


UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

# Reingeniería del Laboratorio de Seguridad Informática

## Análisis, Diseño e Implementación de un *Cyber Range*

Rodrigo Gallardo

Guillermo Guerrero

Tutor:
Gustavo Betarte

Co-Tutores:
Marcelo Rodríguez
Rodrigo Martínez

Usuario Principal:
Juan Diego Campo

Ingeniería en Computación
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Noviembre de 2021

# Agradecimientos

Nos gustaría agradecer a los profesores y miembros del GSI Gustavo Betarte, Marcelo Rodríguez y Rodrigo Martínez por su dedicación y esfuerzo a lo largo de este proyecto, brindando motivación, guía y conocimiento al equipo.

También nos gustaría agradecer al profesor y miembro del GSI Juan Diego Campo, usuario experto y principal desarrollador del LaSI, por acompañarnos desde el comienzo de este proyecto brindando su valioso punto de vista.

Por último, agradecer a los miembros del tribunal María Eugenia Corti, Mercedes Marzoa y Martín Pedemonte; por su dedicación a la hora de evaluar este proyecto.

*Rodrigo Gallardo, Guillermo Guerrero*

A mi madre, Milka; a mi padre, Miguel; a mi hermano, Marcos.

*Rodrigo Gallardo*

A mis padres Andrea y Pedro, y a mi hermano Santiago por acompañarme y apoyarme en estos últimos cinco años.

*Guillermo Guerrero*

# RESUMEN

El objetivo de este trabajo es contribuir a desarrollar una versión actualizada del Laboratorio de Seguridad Informática (LaSI) del Grupo de Seguridad Informática (GSI). Este laboratorio es utilizado para posibilitarle a estudiantes de grado y posgrado un acercamiento práctico, en escenarios realistas, a diversos conceptos de ciberseguridad.

Se llevó a cabo un relevamiento del estado del arte en implementaciones existentes de laboratorios para la enseñanza en ciberseguridad, donde destaca el concepto de *cyber range*, que generalmente consiste en una plataforma donde se pueden llevar a cabo prácticas realistas de defensa y ataque de sistemas informáticos. También se realizó un relevamiento del estado del arte en tecnologías utilizadas en *cyber ranges*.

Por otra parte, se desarrolló un análisis de los requerimientos funcionales y no funcionales que debe satisfacer la infraestructura que da soporte a un laboratorio de seguridad informática.

Del relevamiento realizado se eligió el framework CyTrONE como la solución que mejor se alinea con los requerimientos establecidos y se propone un nuevo diseño del LaSI utilizando esta tecnología. Como parte del proyecto, se modificó la herramienta referida y se integró con Elastic Security. Esta solución final propuesta satisface un 95.8 % de los requerimientos identificados, dejando solo un requerimiento de prioridad baja sin cubrir.

Asimismo, se implementó un prototipo de la solución final propuesta, desplegando un subconjunto de los componentes propuestos en un ambiente distribuido con dos servidores. Finalmente, se rediseñó una práctica del laboratorio del curso de grado/posgrado Fundamentos de Seguridad Informática para migrarla a la nueva plataforma.

Se concluye que el prototipo final cubre las funcionalidades actuales del LaSI más todos los requerimientos nuevos de prioridad alta, y muchos de los de prioridad media y baja.

Palabras claves:
Ciberseguridad, Enseñanza, Cyber Range, Virtualización.

# Tabla de contenidos

# Lista de figuras

# Lista de tablas

# Lista de siglas

**ADR** Anomaly Detection Reasoner 10

**AS** Attack Simulator 10

**CERTuy** Centro Nacional de Respuesta a Incidentes de Seguridad Informática 1

**CLC** Centralized Logging Component 9

**CLF** Cross-Learning Facilities 9

**CRACK** Cyber Range Automated Construction Kit 20, 21

**CS** Countermeasures Simulator 10

**CTF** Capture The Flag 5, 11, 13, 14, 39, 40, 47

**DL** Digital Library 9

**EQL** Event Query Language 57

**ERE** Economic Risk Evaluator 10

**ERM** Economic Risk Models 10

**EVA** Espacio Virtual de Aprendizaje 29

**FSI** Fundamentos de Seguridad Informática 2, 3, 54, 63

**GSI** Grupo de Seguridad Informática 2, 3, 4, 17, 38, 39, 40, 43, 45, 50, 62, 63, 64, 65

**GUI** Graphical User Interface 8, 12

**HIDS** Host Intrusion Detection System 10

**IP** Internet Protocol 32

**IaC** Infraestructure as Code 21

**IaaS** Infrastructure as a Service 10, 16, 21, 22, 39

**KVM** Kernel-based Virtual Machine 17, 19, 36, 37, 50, 51, 63

**LMS** Learning Management System 9, 12, 40, 46

**LXC** LinuX Containers 17, 19, 36, 37

**LaSI** Laboratorio de Seguridad Informática 2, 38, 39, 45, 54, 63

**MS** Monitoring Sensors 10

**MitM** Man In The Middle 28

**NIDS** Network Intrusion Detection System 10

**PE** Performance Evaluator 10

**RLMS** Range Learning Management System 5, 6

**SIEM** Security Information and Event Management 3, 10, 23, 40, 47, 57, 64, 65

**SIM** Simulated Infrastructure Manager 10

**SO** Sistema Operativo 16, 17, 18, 19, 21, 22, 25, 30, 37, 56

**SSH** Secure SHell 48

**TI** Tecnologías de la Información 21

**TM** Training Manager 10

**TOSCA** Topology and Orchestration Specification for Cloud Applications 21

**TSI** Taller de Seguridad Informática 2

**VAT** Vulnerability Assessment Tool 10

**VM** Virtual Machine 15, 16, 18, 19, 20, 22, 37, 44, 49, 50

**VNX** Virtual Networks over LinuX 20, 22

**VSDL** Virtual Scenario Description Language 20, 21

**XML** Extensible Markup Language 20

# Capítulo 1

# Introducción

En la actualidad, la tecnología se encuentra presente en la vida cotidiana de todas las personas y en el funcionamiento normal de las empresas. Por esto, la seguridad informática juega un rol fundamental en proteger los activos de información y otros recursos de los usuarios, ante las numerosas amenazas que proliferan año a año.

Informes estadísticos sobre los incidentes de seguridad informática en el país, generados por el Centro Nacional de Respuesta a Incidentes de Seguridad Informática (CERTuy), indican que en el año 2020 hubo un incremento del 26 % respecto al año anterior. Este incremento es evidenciado a lo largo de los últimos años. A su vez, en el año 2020, se tuvo que 38 de los 2798 incidentes reportados fueron clasificados con prioridad "Alta" o "Muy Alta" [6].

El país necesita una mayor cantidad de recursos humanos especializados en ciberseguridad. Debe destacarse que esta realidad es evidenciada a nivel mundial, debido a que tan solo en el año 2021 se estiman que hay 3,5 millones de puestos de trabajos en ciberseguridad sin cubrir producto de la falta de talentos [38]. Por esta razón, cobra vital importancia la capacitación y entrenamiento de nuevos profesionales en seguridad, para generar mayor conciencia y habilidades en el tema.

La capacitación brindada por los educadores no puede ser únicamente teórica. Es pertinente brindar capacitación y entrenamiento también de forma práctica para que el individuo pueda complementar los conceptos aprendidos y así generar una mayor comprensión de los temas tratados. El problema que surge con la capacitación práctica de temas de seguridad es que debe realizarse en un ambiente seguro y legal, donde el individuo pueda experimentar con

herramientas y tecnologías que potencialmente pueden generar un daño. Esto se debe a que en la seguridad informática no solo es importante, por ejemplo, comprender como prevenir ataques, sino que también cobra vital importancia comprender como se realiza el ataque, y para esto no hay mejor manera que ponerse "en la piel" del atacante.

El Grupo de Seguridad Informática (GSI) cuenta con un laboratorio de ciberseguridad, llamado Laboratorio de Seguridad Informática (LaSI) [1], el cual utiliza para formar a estudiantes de grado y posgrado, poniendo en práctica sobre escenarios realistas los conceptos vistos en cursos como Fundamentos de Seguridad Informática (FSI), Taller de Seguridad Informática (TSI) y los distintos cursos que son dictados en la Especilización de Seguridad Informática del CPAP, el centro de posgrado profesional del InCo. Sin embargo, se busca actualizar la plataforma actual de forma de cubrir ciertos requerimientos insatisfechos y, a su vez, acercar el laboratorio a las tecnologías utilizadas hoy en día por otras implementaciones de laboratorios existentes. Por otro lado, se proyecta que FSI pase a ser parte del currículo obligatorio de la carrera Ingeniería en Computación en el nuevo plan de estudios que se está desarrollando, por lo que se espera un crecimiento significativo en el número de usuarios concurrentes que tiene que poder soportar la infraestructura.

Por lo tanto, el objetivo de este trabajo es rediseñar la actual plataforma del GSI para obtener una nueva versión que cumpla requisitos adicionales a la actual, además de seguir cubriendo o mejorando los requerimientos ya implementados. Para esto, fue necesario realizar, en primer lugar, un análisis de requerimientos e identificar cuáles serían las funcionalidades y características deseables de un laboratorio de ciberseguridad. Estos requerimientos fueron clasificados en tres clases de prioridad: alta, media y baja.

Con el fin de diseñar el nuevo laboratorio, se realizó un relevamiento que cubrió dos ejes principales. En primer lugar, se investigó el estado del arte en herramientas para la enseñanza de ciberseguridad en el resto del mundo, donde cobra gran relevancia el concepto de *cyber range*. Luego, se investigó el estado del arte actual en tecnologías utilizadas en los *cyber ranges*, donde tiene gran importancia la virtualización, una técnica que permite el despliegue de múltiples escenarios virtuales, formados por nodos conectados en red, sobre una misma infraestructura de *hardware*. Ambos relevamientos incluyeron la lectura de material bibliográfico y diverso tipo de material accesible en internet, así como el *testing* de algunas de las herramientas que se consideraron de mayor

interés.

De todas las tecnologías evaluadas, se observó que el *framework* CyTrONE [4] es el que mejor se alinea con los requerimientos del GSI, lo cual llevó a incluir esta tecnología en la implementación del nuevo laboratorio. Como parte del trabajo, también se realizaron ciertas modificaciones necesarias a la herramienta para satisfacer algunos requerimientos específicos del GSI que la versión actual de CyTrONE no lograba cubrir. A su vez, la solución final se integró con Elastic Security [13], un Security Information and Event Management (SIEM) que permite monitorear la actividad de los estudiantes en la plataforma y generar alertas o avisos de forma automática dependiendo de las acciones de los mismos.

Finalmente, se desarrolló y evaluó un prototipo de la solución final, y se desplegó el mismo en una infraestructura distribuida con dos servidores. Para evaluar un caso de uso real de la plataforma, se migró uno de los laboratorios actuales del curso FSI. El prototipo construido cubre todos los requerimientos de alta prioridad identificados, un 57.1 % de los de media prioridad y un 42.8 % de los de baja prioridad. Esto es una mejora en comparación con el laboratorio actual del GSI.

En este prototipo se excluyeron algunos componentes de la solución final. Se concluye que integrando estos componentes al prototipo se lograrían cubrir todos los requerimientos salvo uno de prioridad baja, obteniendo una herramienta completa, robusta, fácil de utilizar, flexible y eficiente.

El resto del informe se divide de la siguiente forma: en el capítulo 2 se presenta un resumen del estado del arte en materia de herramientas y tecnologías para la enseñanza de seguridad informática; en el capítulo 3 se identifican los requerimientos y casos de uso ideales de un *cyber range* y se realiza una comparación de un subconjunto de las tecnologías en base a este análisis; en el capítulo 4 se detalla el diseño de la solución final; en el capítulo 5 se evalúa el prototipo construido en base a la solución final diseñada, para lo cual se migra una de las prácticas actuales del curso FSI a la nueva plataforma; finalmente, el capítulo 6 plantea las conclusiones obtenidas y el trabajo a futuro.

# Capítulo 2

# Estado del Arte

## 2.1. Herramientas para la Enseñanza de Seguridad Informática

En esta sección se presentan los conceptos claves de un *cyber range* y los principales componentes que lo conforman. Además, se introducen tres distintas implementaciones que permiten entender en mayor medida como funcionan estas plataformas, con un énfasis especial en la plataforma CyTrONE que se utilizó como base para la implementación del nuevo *cyber range* del GSI.

Este capítulo está basado en el documento [16], el cual presenta de forma más extensa un estado del arte para estas plataformas.

### 2.1.1. Definición de *Cyber Range*

Los *cyber ranges* son plataformas que permiten simular distintos ambientes conformados por redes, sistemas y aplicaciones. El usuario puede adquirir habilidades prácticas en temas de seguridad de forma interactiva, segura y legal. En estas plataformas se pueden encontrar tres tipos de usuarios: los educadores, que se encargan de generar el material didáctico y los escenarios; los aprendices, que buscan entrenar sus habilidades en los escenarios planteados; y por último se tiene a los administradores que se encargan de administrar la plataforma (usuarios, roles, infraestructura, entre otros).

La idea detrás de un *cyber range* es brindar un aprendizaje desde un punto de vista más práctico. Apoyándose en estas plataformas, los educadores pueden desarrollar escenarios basados en los conceptos teóricos enseñados, para que

los aprendices puedan aplicar el conocimiento generado y desarrollar nuevas habilidades. Pueden existir distintas formas de aprendizaje, por lo que en los *cyber range* se pueden encontrar distintas metodologías de enseñanza [41]. A continuación se introducen brevemente cada una de estas metodologías.

El entrenamiento de ataque simulado sigue un enfoque en el cual el usuario es puesto a prueba al tener que defender una infraestructura de ataques simulados. El sistema registra las acciones tomadas por el usuario y selecciona los ataques en función de estas. De esta forma se puede realizar un entrenamiento específico para cada usuario.

Otro caso es el del entrenamiento basado en roles que trata de plantear distintos escenarios donde cada usuario tiene un rol específico. Los roles más comunes son el rol de atacante y el rol de defensor. A través de estos escenarios cada usuario puede practicar un conjunto específico de habilidades que se corresponden al rol que desempeña en el momento.

Por otra parte, se tiene el entrenamiento basado en ejercicios donde el usuario es puesto a prueba en un escenario que simula una situación real donde debe cumplir con una serie de objetivos. Los ejercicios suelen ser del tipo Capture The Flag (CTF) donde el objetivo es realizar un conjunto de acciones que permiten encontrar una (o más) banderas (que por lo general son archivos con información importante). Se considera que el usuario cumplió con los objetivos del ejercicio cuando encontró estas banderas.

Por último se tiene la gamificación, la cual es una técnica que aplica las mecánicas de los juegos al ámbito educativo buscando el compromiso y motivación del usuario. Se suele aplicar técnicas como plantear los escenarios por niveles y asignar puntajes al cumplir cada objetivo.

Debe destacarse que estas metodologías de enseñanza no son necesariamente excluyentes, pueden resultar muy útiles cuando se aplican en conjunto.

### 2.1.2. Componentes de un *Cyber Range*

En el diseño de un *cyber range* se pueden identificar cinco componentes claves, como se evidencia en la figura 2.1. El Range Learning Management System (RLMS) es un sistema que permite la administración, documentación, reporte, seguimiento y automatización de cursos y programas de entrenamiento. Contiene las características claves de un *cyber range* como son los servicios para la definición de escenarios, gestión de grupos, recolección y reporte de

datos sobre el entrenamiento y estado de la plataforma.

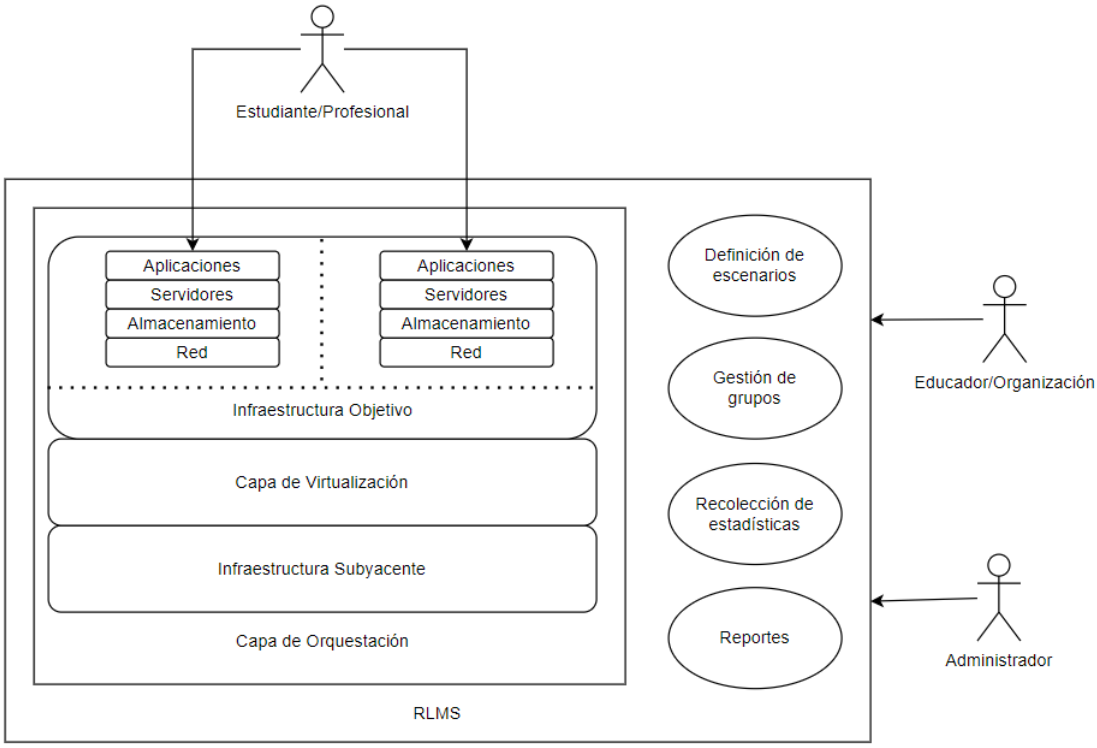


**Figura 2.1:** Componentes claves de un *cyber range*

Luego se tiene una capa de orquestación, la cual recibe entradas desde el RLMS y actúa como gestor de todas las tecnologías, herramientas y servicios que existan en el *cyber range*. También se tiene la tecnología subyacente, que se define como el conjunto de redes, servidores, *routers*, *firewalls*, etc., sobre los que ejecuta el *cyber range*.

Por otra parte se suele tener la capa de virtualización. Esta capa cobra vital importancia ya que es utilizada para reducir el número de componentes físicos de los que depende el *cyber range* para ejecutar. A través del uso de virtualización se pueden reducir los costos asociados a una infraestructura física. En el capítulo 2.2 se ahonda en detalles sobre tecnologías de virtualización.

Por último, se tendrá la infraestructura objetivo que son el conjunto de redes, servidores y aplicaciones que simulan el ambiente que se hace disponible al usuario final.

En definitiva, el flujo normal de funcionamiento de un *cyber range* es el siguiente. El RLMS se comunica con la capa de orquestación para indicar que se debe generar una nueva infraestructura objetivo. Para esto, la capa

de orquestación gestiona la tecnología subyacente y se apoya en la capa de virtualización que ejecuta sobre esta, para crear los escenarios necesarios. Una vez que estos escenario están desplegados, el usuario final puede acceder a la plataforma para entrenar y desarrollar sus habilidades.

### 2.1.3. Implementaciones de *Cyber Range*

En la literatura [11, 33, 39, 41], se puede encontrar una extensa y variada clasificación para los *cyber ranges* utilizando distintos criterios. Realizar esta clasificación permite determinar los conceptos claves y más importantes que tienen en común. Esto resulta útil para entender el funcionamiento de estas plataformas y poder desarrollar nuevas implementaciones.

La clasificación se realiza en base a como se aplican ciertos conceptos considerados claves para un *cyber range*. A continuación, se presentan algunos de estos conceptos.

En un *cyber range* es muy importante poder abarcar una gran gama de escenarios que permitan desarrollar y entrenar distintas habilidades. La características esenciales en torno a un escenario son:

- Narrativa: para que el escenario tenga sentido debe tener objetivos. Se pueden identificar además consecuencias de acciones tomadas, conflictos y dilemas.
- Dominio: identifica el contexto del escenario.
- Material educativo: ayuda al usuario a completar el escenario y aprender las habilidades esperadas de este. Se pueden utilizar tutorías, demostraciones y sistemas de puntuación.
- Gamificación: incorporación de mecánicas de juegos para mantener el nivel de participación del usuario y su motivación.
- Tipo: cada escenario tiene un tipo asociado y este puede ser estático, teniendo un único objetivo, o dinámico, donde el objetivo varía dependiendo de las acciones del usuario.

Otro de los conceptos esenciales es la gestión de equipos y cuales son soportados por la plataforma. Dentro de un *cyber range* pueden existir distintos equipos caracterizados por un color, donde la idea es que cada uno tiene asignada una función en específico. Como exponentes principales se tiene al equipo rojo y azul con los roles de atacante y defensor respectivamente. Mientras que

el equipo rojo es el encargado de seguir posibles vectores para realizar un ataque, el equipo azul se encarga de la seguridad, estabilidad, escalabilidad y disponibilidad de la infraestructura de red, servicios y aplicaciones dentro del escenario. Es muy útil brindar escenarios en los que estos equipos puedan convivir.

Por último, otra de las características esenciales está relacionada a la infraestructura y las tecnologías utilizadas. Se puede encontrar *cyber ranges* que hacen uso de la nube para el despliegue de los escenarios. También, suelen ser muy utilizadas tecnologías de virtualización que permiten el despliegue de máquinas virtuales y contenedores.

Prestando especial atención a estas tres características, es que a continuación se presentan tres implementaciones de *cyber ranges*.

#### 2.1.3.1. CyberWiser.eu

CyberWiser.eu [2, 14] es un *cyber range* desarrollado por la Unión Europea en conjunto con múltiples organizaciones. Este brinda un ambiente dedicado al entrenamiento de profesionales en el campo de la ciberseguridad. Además, permite la evaluación de organizaciones en relación con las buenas prácticas para la ciberseguridad.

Para el desarrollo de los cursos y el material de entrenamiento, CyberWiser.eu sigue un proceso en el que el primer paso es identificar los usuarios, sus roles y habilidades a entrenar en la plataforma. Como segundo paso se tiene el mapeo de esos roles y habilidades al sistema de entrenamiento de la plataforma. En el tercer paso se describen los cursos utilizando plantillas predefinidas, y por último se generan los cursos y materiales de entrenamiento. Estos cursos son agrupados en cuatro niveles de entrenamiento, cada uno con distintos objetivos de aprendizaje dependiendo del nivel del usuario.

La plataforma está compuesta por distintos componentes interoperables que permiten al usuario la administración y uso de los distintos materiales y escenarios planteados en el *cyber range* y su monitoreo. Estos componentes pueden observarse en la figura 2.2.

La conexión del usuario con la plataforma es a través del portal web. Este es el punto de acceso, a través de una Graphical User Interface (GUI), a la plataforma y todos los recursos y materiales de entrenamiento. Una vez autenticado, el usuario tiene acceso al Cross-Learning Facilities (CLF). Es-

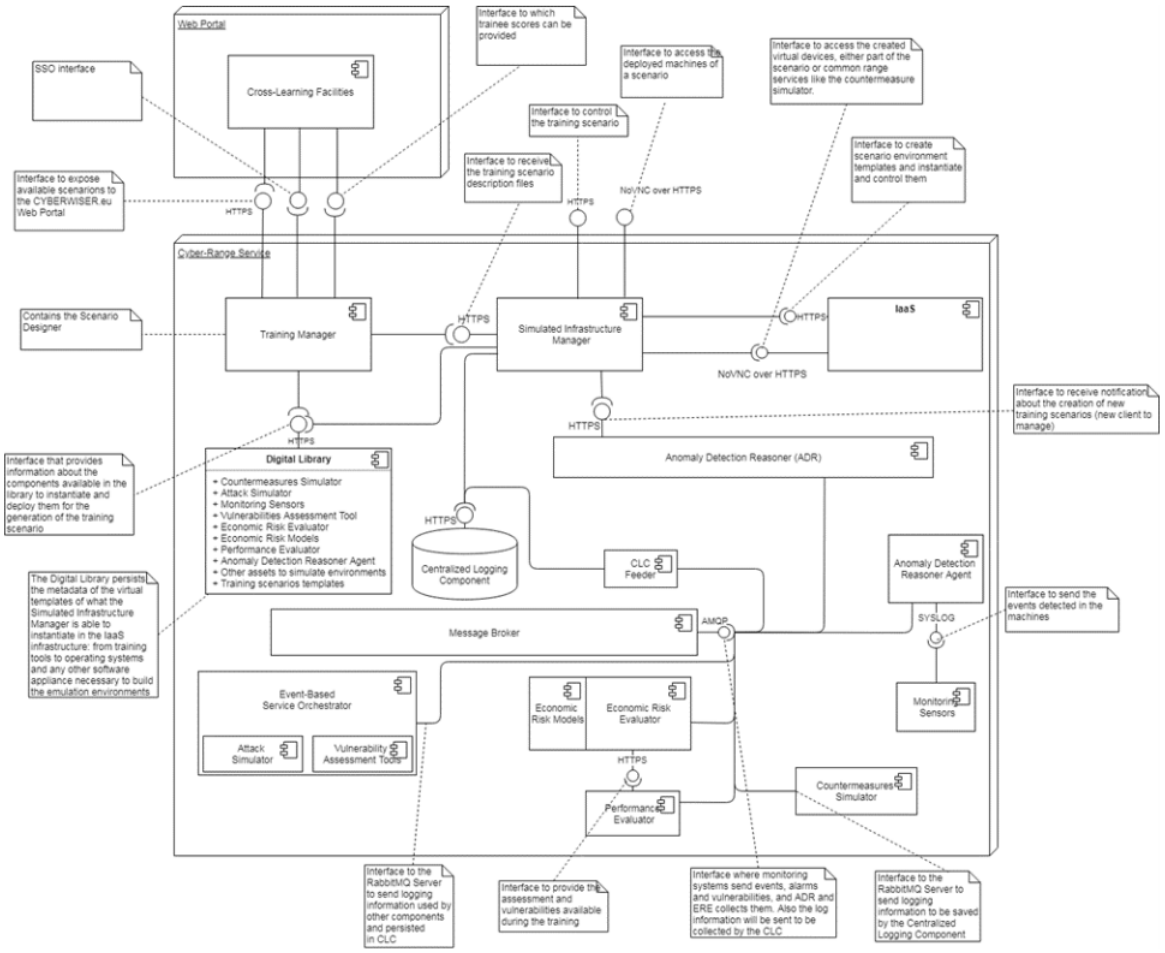


**Figura 2.2:** Diagrama de componentes. Extraído de [2].

te componente está conformado por un Learning Management System (LMS) que gestiona el material de capacitación (documentos, presentaciones, videos y cuestionarios) y brinda acceso a distintos servicios de la plataforma como por ejemplo la gestión de escenarios. Los servicios y funcionalidades que tiene activo el usuario en el CLF dependen de su rol dentro de la plataforma.

El componente de servicios está compuesto por distintos subcomponentes. Los que son considerados más importantes son descriptos a continuación.

El Centralized Logging Component (CLC) es una base de datos que almacena toda la información relativa a los distintos escenarios. El tipo de información almacenada es información de *logging*, puntajes y avances del usuario y eventos ocurridos dentro del escenario.

Digital Library (DL) es el repositorio donde se almacenan las plantillas con la información necesaria para configurar los distintos elementos a desplegar en

un escenario. Además contiene otros componentes necesarios para gestionar los escenarios como son: Countermeasures Simulator (CS), Attack Simulator (AS), Monitoring Sensors (MS), Vulnerability Assessment Tool (VAT), Economic Risk Evaluator (ERE), Economic Risk Models (ERM), Performance Evaluator (PE), Anomaly Detection Reasoner (ADR).

El Training Manager (TM) es el componente responsable de la gestión (creación, edición y eliminación) de los distintos escenarios. Por otra parte, el Simulated Infrastructure Manager (SIM) es responsable por la instanciación y despliegue de los escenarios, y permite el acceso de los usuarios a las máquinas desplegadas.

El componente de Infrastructure as a Service (IaaS) ofrece servicios relacionados con los recursos informáticos virtualizados que pueden solicitarse a demanda, permitiendo la escalabilidad de manera sencilla. Este componente permite la generación, instanciación y control de los escenarios al recibir comunicaciones desde el SIM.

El ADR es un SIEM cuyo objetivo es la detección de amenazas de seguridad. Normaliza, filtra y correlaciona la información provista desde múltiples fuentes, generando alarmas según ciertas reglas. Este sistema recibe los eventos generados por los ADR *agents*, los procesa y genera las alarmas necesarias. A su vez, el agente recibe alertas desde los MS que pueden ser de dos tipos: Network Intrusion Detection System (NIDS) o Host Intrusion Detection System (HIDS).

El componente de VAT realiza escaneos de vulnerabilidad en los objetivos indicados. Por otra parte, el componente de AS permite generar ataques a los objetivos indicados. Estos componentes resultan muy útiles para implementar las actividades de los equipos rojo y azul respectivamente.

Por último, el componente de PE se encarga de evaluar la *performance* y avance de los usuarios dentro del escenario.

#### 2.1.3.2. KYPO

KYPO [20, 21, 22] es un *cyber range open-source* desarrollado por la Universidad de Masaryk en Chequia. “La plataforma de *cyber range* KYPO se basa completamente en enfoques de vanguardia como contenedores, infraestructura como código, microservicios y software de código abierto, incluida la tecnología de proveedor de nube OpenStack”, según [20].

Las prácticas en este *cyber range* están compuestas de cuatro instancias. En una primera instancia se le brinda información al usuario sobre que trata la práctica y la infraestructura del escenario sobre la que estará trabajando. Luego, ocurre una instancia de preguntas que permiten conocer las habilidades y conocimientos previos del usuario. A continuación, se tiene la instancia donde el usuario interactúa con la infraestructura para resolver un ejercicio práctico. Este ejercicio suele ser de tipo CTF organizado en fases. Cuando el usuario encuentra la bandera correcta se le otorga un puntaje y puede avanzar a la siguiente fase. Durante cada fase el usuario tiene la oportunidad de solicitar pistas o conocer directamente la resolución de la fase, lo que le resta puntaje. Por último, puede existir una etapa donde se realizan preguntas relacionadas a las tareas que llevó a cabo el usuario durante la práctica.

Las definiciones de las prácticas, y en particular la definición de escenarios y aquellos sistemas que los constituyen se basan en tecnologías como Packer y Ansible, lo que facilita la automatización de su instanciación.

El diseño de este *cyber range* puede apreciarse en la figura 2.3.

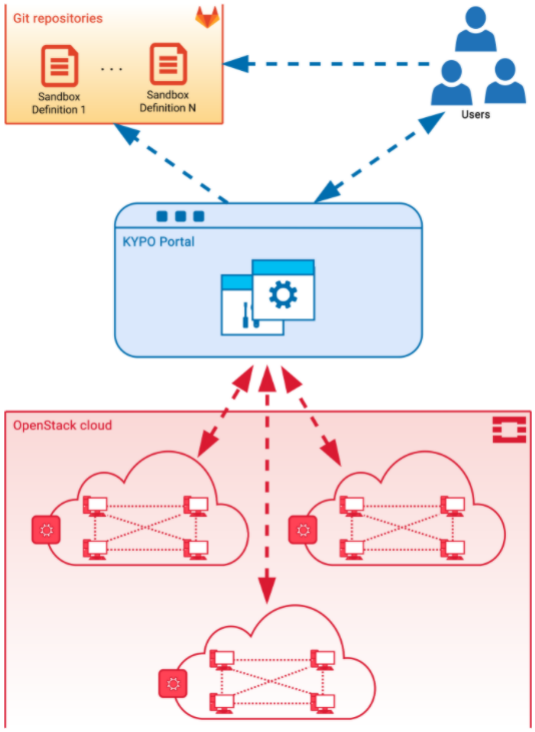


**Figura 2.3:** Diseño de KYPO. Extraído de [21]

Lo primero a destacar es que los usuarios pueden tener distintos roles asignados. Dependiendo del rol se pueden gestionar escenarios, gestionar otros

usuarios y roles, y utilizar los escenarios.

Para conectarse a la plataforma, el usuario interactúa con el portal web. Esta GUI brinda acceso a los distintos servicios ofrecidos por la plataforma. Otro de los detalles a destacar es que la plataforma despliega los escenarios en la nube, lo que permite la flexibilidad y escalabilidad. Por último, los repositorios Git son utilizados para almacenar las definiciones de los escenarios y desde estos la plataforma obtiene las definiciones para instanciarlos.

#### 2.1.3.3. CyTrONE

CyTrONE [4, 5, 30, 32] es un *framewrok open source* diseñado por el Japan Advanced Institute of Science and Technology de Japón. Su objetivo es automatizar las tareas de generación de contenidos y ambientes de entrenamiento. Un usuario provee una *training specification*, de la cual se deriva una descripción de contenido didáctico y una especificación de un escenario. Para esto se tienen tres grandes componentes: CyRIS, CyLMS y CyPROM [30].

CyRIS [32] es definido como una herramienta para facilitar la creación y administración de escenarios abstrayéndose de la infraestructura de *hardware* que se utilice por debajo. Se encarga de gestionar las especificaciones de escenarios generadas e instanciarlas.

Por otra parte, el componente denominado CyLMS [4] se encarga de generar los distintos contenidos didácticos de carácter más teórico que están asociados al escenario para ser gestionados por un LMS. Este contenido incluye material informativo y cuestionarios. La idea del uso del LMS es también permitir que el estudiante entregue las evidencias encontradas en el transcurso de la práctica directamente en esta plataforma, lo que facilita la corrección del trabajo. Para esto, el componente traduce la especificación del contenido realizada por el educador al formato SCORM y lo deja disponible en el LMS de forma automática.

Por último, CyPROM [5] permite la gestión de la evolución de los escenarios para que estos sean dinámicos, aplicando actividades de ataque, defensa y forense. Dada una especificación de como debe evolucionar un escenario, el módulo se encarga de ejecutar automáticamente las tareas necesarias, como pueden ser la ejecución de ataques.

El diseño de CyTrONE puede observarse en la figura 2.4. El *framework* recibe las entradas del usuario instructor y genera la descripción del contenido

para ser procesada por el componente CyLMS, la especificación del escenario para ser procesada por CyRIS y la especificación de la evolución del escenario para ser procesada por CyPROM. De esta forma queda disponible el escenario para el usuario aprendiz.

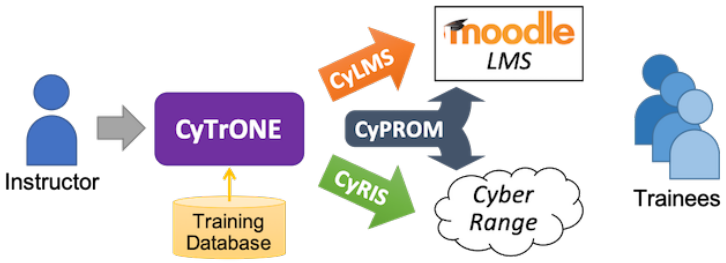


**Figura 2.4:** Diseño de CyTrONE. Extraído de [4]

Este *framework* resultó muy útil para el desarrollo de este proyecto. Más detalles de esta plataforma, en especial del diseño y funcionamiento del módulo CyRIS, son presentados en el capítulo 4.

### 2.1.4. Metodologías para la evaluación

Es de suma importancia para los entrenadores que llevan a cabo prácticas de enseñanza en ciberseguridad evaluar el desempeño de los estudiantes o participantes de la práctica. También, cuando existen muchos caminos para solucionar la tarea propuesta, puede ser de interés conocer el camino que tomó el participante hacia la solución.

En esta sección se presenta una categorización de las formas automáticas de evaluación de los participantes dentro de un *cyber range*: mediante monitoreo o mediante un enfoque CTF.

#### 2.1.4.1. Monitoreo

Este enfoque consiste en monitorear de forma automática la actividad del usuario en el *cyber range*; específicamente en los *guests* que tiene asignados para trabajar. Este monitoreo puede consistir en identificar los comandos ejecutados, archivos descargados, conexiones abiertas, archivos modificados y mucho más. Se puede entonces medir el avance de un estudiante en la práctica dependiendo de si modificó o abrió ciertas evidencias que fueron ocultadas o de si realizó ciertas acciones (por ejemplo, si logró descargar un archivo en una

máquina víctima). Una metodología orientada a este enfoque es llevada a cabo por CyberWiser.eu, descripto en la sección 2.1.3.1.

Mediante el monitoreo constante de la actividad de los participantes en la práctica, se puede conocer con precisión como logró llegar a la solución o en que grado de avance se encuentra el mismo. También se pueden detectar actividades sospechosas por parte de los usuarios o amenazas contra la infraestructura del *cyber range*. Más aún, este enfoque se puede automatizar, reduciendo el proceso de corrección manual del entrenador.

Sin embargo, las herramientas que permiten realizar este monitoreo constante suelen tener altos requerimientos de *hardware*, lo cual puede ser especialmente desfavorable para laboratorios de enseñanza en el entorno educativo que cuenten con fuertes restricciones presupuestales.

#### 2.1.4.2. Enfoque CTF

Como lo dice su nombre, este enfoque es común en las competencias del estilo CTF, donde diversos equipos compiten por resolver problemas relacionados a la ciberseguridad en un tiempo acotado. Este enfoque es el adoptado por KYPO, descripto en 2.1.3.2, y por CyTrONE, mediante los módulos CyLMS y CyPROM, descriptos en 2.1.3.3.

En términos generales, el enfoque consiste en ocultar diversas *flags* en los *guests* del escenario y medir el avance de los mismos dependiendo de las *flags* que alcanzan. Cuando un equipo encuentra una *flag*, la envía a la plataforma donde se hospeda el *cyber range* y en esta se verifica que la misma sea la *flag* adecuada para el equipo, ya que se pueden tener *flags* distintas para cada uno. Luego, se pueden desbloquear nuevas actividades a realizar a medida que se suministran las *flags* adecuadas, acumulando puntos por cada una de ellas. A su vez, algunas competencias permiten a los participantes solicitar "pistas" sobre como llegar a la *flag*, a costo de una reducción en los puntos.

La evaluación de los participantes también puede incluir cuestionarios, donde los mismos deben contestar preguntas referentes a la práctica o sobre la metodolgía o técnica aplicadas para resolverla. Un ejemplo más avanzado de esto es KYPO, que permite customizar la dificultad de las distintas etapas de la práctica dependiendo de las respuestas de los estudiantes en los cuestionarios.

Este enfoque es menos costoso en cuanto a *hardware* que el anterior. Solo se necesita de una plataforma donde desplegar los cuestionarios y evaluar

las respuestas de los participantes para medir y monitorear su avance en las prácticas. A su vez, la creación de los cuestionarios o de las *flags* y su evaluación se puede automatizar en gran parte, como en el caso de las herramientas KYPO y CyTrONE.

Una de las falencias de este enfoque es que no se puede saber a ciencia cierta el camino que llevó al practicante hacia la solución. Otro problema es que, en contextos académicos, para garantizar la individualidad en la realización de las prácticas, puede ser necesario distribuir *flags* distintas para cada equipo, lo cual puede complejizar más la implementación del mecanismo de evaluación. Este problema no está presente en las competencias de CTF, ya que los equipos que compiten entre si no tienen intenciones de revelar las *flags* descubiertas a los contrarios.

## 2.2. Tecnologías utilizadas en *Cyber Ranges*

Esta sección presenta algunos de los conceptos relevantes relacionados a virtualización, así como un subconjunto de las tecnologías evaluadas para asistir en la creación y despliegue de un *cyber range*. Un relevamiento más extenso de estas herramientas se incluye en [17].

### 2.2.1. Virtualización

La virtualización refiere a la abstracción de los recursos físicos en objetos lógicos. Una máquina virtual o Virtual Machine (VM) ejecuta sobre un *host* físico y, mediante un hipervisor, permite la virtualización de los recursos de *hardware* como procesador, memoria y red. Se pueden tener varias VM ejecutando sobre el mismo *host*, con distintos sistemas operativos instalados (incluso distintos a los del *host*) y utilizando distintas porciones del *hardware* disponible.

El hipervisor se puede considerar como una capa de *software* que se encarga de la virtualización del *hardware*, brindando el ambiente de ejecución y gestión para las VM. Existen distintos tipos de hipervisores, así como distintas técnicas de virtualización. Se describirán solo aquellos tipos y técnicas que se consideran relevantes.

Un hipervisor de tipo 1 ejecuta directamente sobre el *hardware*, sin ningún Sistema Operativo (SO) como intermediario. Se puede pensar como un SO más

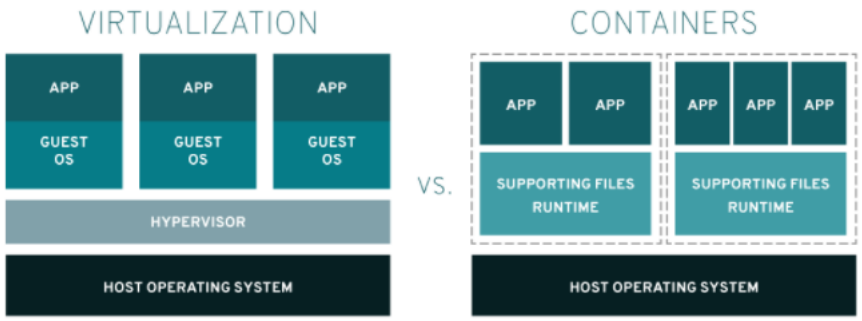


**Figura 2.5:** Comparación entre técnicas de virtualización de VM y de contenedores. Extraído de [12]

simple cuyo propósito es brindar el ambiente de ejecución para desplegar VMs. Un hipervisor de tipo 2 ejecuta sobre un SO instalado, siendo indistinguible de los demás procesos del sistema.

Para virtualizar los recursos, los hipervisores pueden emplear distintas técnicas. La técnica de *full-virtualization* toma todas las instrucciones del sistema operativo *guest* y las traduce a instrucciones capaces de ejecutar en el *hardware* del *host*; esta técnica se puede acelerar, como se menciona más adelante, para obtener mejores resultados de *performance*. Otra técnica conocida es la de *paravirtualization*, en la cual los sistemas operativos *guest* son versiones de sistemas operativos comerciales las cuales fueron modificadas para que las llamadas al sistema se hagan directo al hipervisor.

Una técnica que ha cobrado una importante popularidad en los últimos años es la virtualización a nivel de SO o *containerization*, en la cual se crean distintas instancias del *user space* de un sistema operativo, las cuales ejecutan sobre un mismo *kernel*. Esto es una diferencia en cuanto a hipervisores tipo 1, en los cuales el *kernel* del sistema operativo *guest* también es virtualizado. Esta diferencia les da ciertas ventajas y desventajas que se discutirán más adelante. En la figura 2.5 se observa una comparación entre las técnicas de virtualización de VM tradicionales y las técnicas de contenedores.

Otro concepto importante a introducir es el de la computación en la nube. Las nubes permite agrupar y compartir recursos de *hardware* de forma escalable para ejecutar distintas cargas de trabajo. En el contexto de este trabajo, resulta de particular interés el concepto de IaaS, en el cual un proveedor de nube ofrece a usuarios recursos de cómputo, almacenamiento y red, pudiendo ser utilizados para desplegar cualquier tipo de *software*, desde SOs hasta aplicaciones.

En el contexto de este trabajo se cree que el uso de VMs y contenedores

desplegados en una nube que ofrezca servicios de IaaS puede ser una buena solución para la plataforma del grupo GSI, por lo que a continuación se describen tecnologías y herramientas que permitan esto.

### 2.2.2. Tecnologías de gestión de VM y contenedores

En esta sección se presentan tres tecnologías de virtualización: Kernel-based Virtual Machine (KVM), Xen y LinuX Containers (LXC). Se eligen estas ya que se considera que cada una es representativa de distintas técnicas de virtualización, las cuales se pueden comparar en términos generales en base a estas tecnologías. Estas técnicas son *full-virtualization*, *paravirtualization* y *containerizaiton*. También se presenta Libvirt, una herramienta potente para el manejo de máquinas virtuales y contenedores. En el capítulo 3 se comparan estas tecnologías en base a los requerimientos identificados.

#### 2.2.2.1. KVM

KVM [19] convierte a un SO Linux tradicional en un hipervisor de tipo 1. Utiliza técnicas de *full-virtualization* aceleradas. Para esto, utiliza el emulador de *hardware* QEMU, el cual funciona en base a *full-virtualization*. KVM reconoce aquellas instrucciones de la CPU virtual generada por QEMU que pueden ser ejecutadas directamente en *hardware* y que no necesitan traducción, acelerando la técnica de *full-virtualization*.

**Características importantes**

- Es *open-source*, y se encuentra integrado en el *kernel* de Linux desde la versión 2.6.20, por lo cual es fácil de instalar.
- Requiere de extensiones de *hardware* para la virtualización (Intel VT o AMD-V).
- Utiliza SELinux y SVirt para la seguridad y aislamiento de las máquinas virtuales, aplicando políticas MAC.
- Al utilizar *full-virtualization*, soporta la virtualización de una gran variedad de sistemas operativos comerciales Linux, Windows e incluso algunas versiones de Android.
- Tiene soporte para *live-migration* de máquinas virtuales.
- Puede utilizar técnicas de *copy-on-write* de imágenes de disco.

- Funcionalidades de Kernel Samepage Merging, lo cual permite que procesos que pueden estar en distintas VM compartan páginas de memoria, y *memory ballooning*.

#### 2.2.2.2. Xen

Xen [44] es un hipervisor de tipo 1 que puede utilizar técnicas de *full-virtualization* o *paravirtualization*. El hipervisor es gestionado a partir de un *guest* específico, usualmente una versión de Linux modificada, el cual es denominado "Domain-0". A partir de este se pueden crear nuevos *guests* con menos privilegios, llamados "Domain-Us" (por usuario). En modo *paravirtualization*, estos *guests* cuentan con SOs modificados específicamente para ejecutar en el hipervisor Xen e invocar sobre este las *system calls* que necesiten.

**Caracterisiticas importantes**

- Es *open-source* y lleva una gran trayectoria de soporte.
- En modo *paravirtualization*, no requiere de extensiones de *hardware* para la virtualización.
- En modo *paravirtualization*, se obtiene una *performance* casi nativa ya que las *system calls* son despachadas directamente al hipervisor, quien se encarga de ejecutarlas.
- En modo *paravirtualization*, requiere de SO portados específicamente para ejecutar sobre el hipervisor. Esto reduce la cantidad de sistemas operativos *guests* que se pueden desplegar. Actualmente existen versiones modificadas de Linux, FreeBSD y OpenBSD para ejecutar sobre Xen. Aunque no existen versiones portadas de Windows, el mismo se puede desplegar en modo *full-virtualization*, si se cuenta con extensiones de *hardware* para la virtualización.
- Tiene soporte para *live-migration* de máquinas virtuales.
- Soporta *memory balloning*. Funcionalidades similares a *Kernel Samepage Merging* aún se encuentran en desarrollo.

#### 2.2.2.3. LXC

LXC [26] es una herramienta que permite aplicar virtualización a nivel de SO o *containerization* sobre el *kernel* de Linux. Permite a los usuarios la creación y administración de contenedores que buscan ser lo más parecido posible a una VM pero sin necesitar ejecutar otro *kernel*.

**Características importantes**

- Es *open-source*.
- *Performance* casi nativa y menor *overhead* en comparación con una VM.
- Soporte para funcionalidades de *copy-on-write* mediante *snapshots*.
- Permite definir contenedores privilegiados o no privilegiados. En los primeros el UID 0 del contenedor es mapeado al UID 0 del *host*, y la seguridad es aplicada a través de los mecanismos anteriormente descriptos. En cambio, el UID 0 de los contenedores no privilegiados es mapeado a un usuario sin privilegios de *root* fuera del contenedor.
- Utiliza AppArmor, SELinux y otras herramientas más para brindar una capa extra de seguridad.

#### 2.2.2.4. Libvirt

Libvirt [25] es un conjunto de *software open-source* que permite la gestión de máquinas virtuales y otras tecnologías de virtualización. Incluye un *deamon*, un cliente de línea de comandos y una API disponible en distintos lenguajes de programación como C, Pyhton y Perl, entre otros.

Permite trabajar con distintos hipervisores como KVM, Xen, y tecnologías de contenedores como LXC. Permite desacoplarse de la tecnología de virtualización utilizada y trabajar con varias de estas al mismo tiempo a través de una interfaz común.

Las principales funciones que ofrece Libvirt son:

- Gestión de VM: permite gestionar el ciclo de vida de una VM ya que se puede iniciar, pausar, guardar, restaurar, migrar y eliminar una VM.
- Gestión de almacenamiento: permite crear imágenes en distintos formatos (ejemplos son qcow2 y vmdk), crear particiones de disco, entre otras operaciones.
- Gestión de red: se pueden gestionar interfaces de red físicas y lógicas.
- Soporte remoto: todas las funcionalidades pueden ser accedidas de forma remota si la máquina cuenta con el *deamon* en ejecución.

### 2.2.3. Herramientas para la creación de escenarios virtuales

Se considera un escenario virtual a un subconjunto de máquinas virtuales o contenedores conectados a través de redes virtuales. En el contexto de un *cyber range*, los escenarios virtuales tienen el propósito de asemejarse a infraestructuras reales, por lo que las máquinas del escenario pueden contar con vulnerabilidades, herramientas de ataque, archivos importantes y más.

Existen diversas herramientas que permiten, mediante algún lenguaje especial, definir escenarios virtuales, especificando las características de los *guests* que los componen, las redes que los conectan y configuraciones necesarias en el *host*. Algunas funcionalidades adicionales que son deseables en este tipo de herramientas son la capacidad de poder instalar aplicaciones de forma automática en los *guest*, configurar reglas de *firewall* y más.

Virtual Networks over LinuX (VNX) es una herramienta *open-source* que cuenta con un lenguaje de definición de escenarios basado en Extensible Markup Language (XML) [42]. Una vez que el usuario define la especificación del escenario, la herramienta la valida y realiza el despliegue de las maquinas virtuales y redes detalladas en la especificación.

ADLES es otra herramienta *open-source*, diseñada para asistir en la especificación y despliegue de escenarios virtuales para prácticas de seguridad informática, abstrayéndose de la infraestructura donde el escenario será desplegado [23]. También cuenta con un lenguaje de especificación de escenarios basado en YAML, el cual la herramienta se encarga de *parsear* y desplegar en la infraestructura determinada por el usuario en la especificación del escenario. Actualmente, solo hay soporte para desplegar VMs utilizando VMware Vsphere, pero la herramienta se puede extender para tolerar otras infraestructuras.

Similar a ADLES, se tiene a Virtual Scenario Description Language (VSDL) [8] y Cyber Range Automated Construction Kit (CRACK) [36], los cuales también fueron diseñado teniendo en mente el despliegue de escenarios para un *cyber range*.

Una particularidad del lenguaje de definición de escenarios de VSDL es que permite especificar vulnerabilidades presentes en los *guests* que conforman el escenario y configurarlos automáticamente para que tengan estas vulnerabilidades. La especificación del escenario se traduce a *scripts* de *Packer* y *Terraform*, ambos herramientas de Infraestructure as Code (IaC), para configurar y

desplegar las máquinas virtuales sobre una infraestructura de *OpenStack*.

El lenguaje de definición de escenarios de CRACK esta basado en el lenguaje de IaC Topology and Orchestration Specification for Cloud Applications (TOSCA). Se observa que VSDL Y CRACK ambos utilizan herramientas de IaC para asistir en el despliegue y configuración de sus escenarios.

En el capítulo 2 se presentaron en mayor detalle CyberWiser.eu, KYPO y CyTrONE, los cuales son *frameworks* más amplios que los vistos en esta sección, pero que también cuentan con lenguajes específicos de definición de escenarios virtuales para prácticas de ciberseguridad.

### 2.2.4. Herramientas de IaC

Debido a que diversas infraestructuras de *cyber range* utilizan herramientas de IaC para asistir en el despliegue de los escenarios virtuales y en la configuración de los *guests*, resulta de interés indagar sobre las distintas soluciones existentes para estos propósitos.

Packer [31] y Libguestfs [24] son dos herramientas que permiten la creación y customización de imágenes de SO para máquinas virtuales o contenedores. Ambos permiten crear usuarios en la imagen, instalar *software*, copiar/extraer archivos y más. *Packer* recibe las acciones a realizar sobre la imagen a través de una especificación en un lenguaje propio y Libguestfs mediante comandos. Para el "aprovisionamiento" de las imágenes (es decir, las configuraciones mencionadas), *Packer* puede interactuar con Ansible, Puppet y otras herramientas similares.

Terraform [18], Ansible y Puppet [34] permiten la especificación, despliegue y configuración de una infraestructura de Tecnologías de la Información (TI). Mediante lenguajes sencillos se puede especificar los nodos participantes de una red, las redes que los unen, configuraciones que se exigen en los nodos y demás acciones. Luego, estas herramientas se encargan de realizar el despliegue y asegurarse de que las configuraciones especificadas se realicen y se mantengan en los nodos.

Terraform y Ansible en particular cuentan con *plugins* que permiten el despliegue de las infraestructuras especificadas en proveedores de IaaS como OpenStack u OpenNebula. Experimentando con el *plugin* que permite el despliegue en OpenNebula, se observó que estas funcionalidades aún están en desarrollo y no permiten explotar por completo los servicios que ofrece este pro-

veedor, además de contar con ciertos problemas que dificultan su utilización: documentación incompleta, actualizaciones inestables y fallas en la creación de escenarios virtuales.

### 2.2.5. Herramientas de despliegue e IaaS

Para gestionar los recursos de *hardware* disponibles de forma eficiente y accesible, existen diversas tecnologías que facilitan la gestión de los escenarios virtuales sobre arquitecturas distribuidas. Estas herramientas son útiles si se desean mantener altos grados de confiabilidad, escalabilidad y usabilidad, abstrayendo al usuario de la tecnología de virtualización subyacente.

Como se vió en la sección 2.2.3, herramientas como ADLES y VNX, además de proveer un lenguaje de especificación de escenarios, también estan capacitadas para realizar el despliegue de los mismos. ADLES despliega las maquinas virtuales utilizando VMware Vsphere, mientras que VNX se comunica con Libvirt y permite el despliegue de máquinas virtuales y contenedores utilizando algunos de los hipervisores soportados por Libvirt. VNX también tiene soporte para despliegue distribuido de los escenarios mediante su versión distribuida, EDIV.

Por otro lado, existen soluciones *open-source* más completas que permiten la creación de una nube que ofrezca servicios de IaaS como CloudStack [3], OpenStack [29] y OpenNebula [28]. Lo que estas tecnologías tienen en común es que se abstraen de la tecnología de virtualización, permitiendo desplegar VMs y contenedores en una gran variedad de hipervisores. Cuentan con un gran número de interfaces que permiten gestionar de forma amigable todos los recursos de *hardware* disponibles: almacenamiento de las imágenes de SOs, recursos de red, recursos utilizados por las VMs o contenedores, usuarios, permisos de la plataforma y mucho más.

Se experimentó con OpenNebula para evaluar la utilidad de una de estas herramientas como infraestructura para un *cyber range*. Se eligió OpenNebula por su facilidad de instalación y por tener bajos requerimientos de recursos en comparación con OpenStack y CloudStack. Las conclusiones extraídas de probar OpenNebula es que es una solución que aún no ha alcanzado un nivel de madurez aceptable, pero que sin embargo es fácil de instalar, liviana y cuenta con diversas funcionalidades que podrían ser de utilidad para el despliegue y gestión de escenarios virtuales. Sin embargo, el propósito de estas herramien-

tas es más general y abarcativo que los de un *cyber range*, y se observa que soluciones más especificas al contexto de trabajo, como CyTrONE, se ajustan mejor y con mayor facilidad a los requerimientos establecidos.

### 2.2.6. Elastic Stack

Elastic Stack [13] es un *stack* tecnológico compuesto por cuatro proyectos *open source*: Elasticsearch, Logstash, Kibana y Beats. Elasticsearch es un motor de búsqueda y analítica. Logstash permite implementar *pipelines* para la ingesta de datos, la transformación y el envío a un destino. Kibana es un visualizador de datos. Y por último, los Beats son agentes que permiten la recolección y envío de datos.

En este proyecto cobra vital importancia el caso de uso de seguridad: Elastic Security, que es una solución de SIEM. Un SIEM es un sistema que permite recopilar información y eventos de seguridad, normalizar y correlacionar dichos datos, y generar alertas ante posibles amenazas de seguridad. Además se tiene el *endpoint* que es el agente que se despliega en el *host* para realizar la recopilación de información que es enviada al SIEM y que permite la prevención de amenazas como pueden ser *malware* y virus.

Para esto, Elastic Security recolecta información de red (conexiones abiertas, tráfico, conexiones SSH), contenido de *logs* (por ejemplo, *logs* de auditoría), comandos ejecutados en las maquinas, uso de recursos y mucho más.

# Capítulo 3

# Análisis

En este capítulo se presenta el análisis desarrollado para la nueva plataforma para el laboratorio de seguridad informática. Se presentan los usuarios de la plataforma, los requerimientos y los principales casos de uso. Se comparan las tecnologías de virtualización detalladas en la sección 2.2.2 en base a los requerimientos identificados. Por último, se presenta un análisis de los distintos *cyber ranges* investigados con foco en los requerimientos identificados y otras cuestiones fundamentales.

## 3.1. Operativa de la plataforma

El principal cometido de la plataforma es brindar un ambiente seguro y legal en el que se puedan desarrollar conocimientos y habilidades sobre temas de seguridad informática. Estos conocimientos pueden estar relacionados a explotación de vulnerabilidades, uso de herramientas de ataque, *hardening* de sistemas, entre otros.

Para cumplir con este objetivo se identifican tres usuarios fundamentales:

- Estudiante: tiene acceso a las instancias de escenarios y los utiliza para adquirir conocimientos y desarrollar sus habilidades.
- Educador: debe poder diseñar escenarios, asignar instancias de escenarios a estudiantes o grupos de estudiantes. Además debe poder realizar la evaluación del trabajo del estudiante.
- Administrador: gestiona los componentes de *hardware* y *software* de la plataforma, así como a los usuarios.

Un escenario es una especificación de un conjunto de sistemas que interactúan entre si, donde se espera que el estudiante pueda desarrollar una práctica para adquirir conocimientos específicos.

El escenario está compuesto por cinco secciones principales: datos generales, definición de *guests*, definición de redes, definición de evidencias y definición de reglas. La estructura de un escenario se puede visualizar en la figura 3.1.

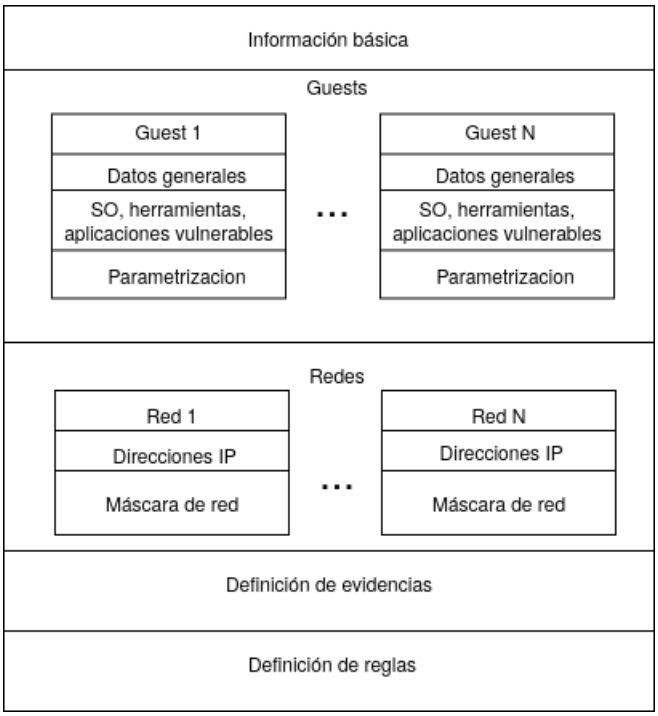


**Figura 3.1:** Estructura general de una especificación de escenario y sus componentes.

La sección de información básica describe al escenario.

La definición de *guests* abarca todos los *guests* participantes del escenario. Cada *guest* cuenta con las siguientes subsecciones:

- Datos generales: estos datos permiten identificar al *guest* dentro del escenario y sirven para su instanciación. Ejemplos son: identificador, rol (atacante o víctima), ruta a la imagen base.
- Sistema operativo, herramientas y aplicaciones vulnerables: identifican el SO, herramientas y aplicaciones vulnerables con las que cuenta el *guest*. Se entiende por aplicación vulnerable a aquella que tenga una vulnerabilidad conocida y que se espera que el estudiante explote durante la práctica. Una herramienta es cualquier otra aplicación que ayuda al estu-

diante a llevar a cabo la práctica, por ejemplo: herramientas de análisis, herramientas de ataque, entre otras.

- Parametrización: esta subsección describe configuraciones con las que debe contar el *guest*. Ejemplos son: usuarios, archivos, *script* a ejecutar, etc. Se distinguen aquellos archivos que forman la evidencia que el estudiante debe entregar una vez finalizada la práctica.

La sección de redes identifica a las redes del escenario que permiten la comunicación entre los *guests* de este. Para cada red se debe indicar la dirección de red, máscara, y dirección de cada *guest* en esa red. Además se pueden definir reglas para controlar el tráfico que transita la red.

En la sección de definición de evidencias se debe indicar la función de validación que se encarga de comparar la evidencia entregada por el estudiante contra la versión original para validar su trabajo. Como ejemplo de función de validación se tiene la comparación de dos archivos para determinar que su contenido es igual.

Por último, en la sección de definición de reglas se especifican las reglas utilizadas para lograr el monitoreo constante del trabajo del estudiante y el estado de la plataforma durante la realización de la práctica.

La figura 3.2 presenta una máquina de estados que representa el ciclo de vida de un escenario. El ciclo de vida comienza cuando un educador desea especificar un escenario. Una vez especificado, es almacenado en el repositorio de escenarios donde se puede seguir modificando.

A partir de la especificación del escenario (en particular utilizando la sección de definición de *guests*) se puede crear las imágenes base de uno o más *guests*. Además, se toma como entrada aplicaciones vulnerables, herramientas y las imágenes base desde sus respectivos repositorios. Las imágenes generadas son almacenadas en el repositorio de imágenes base donde se pueden seguir modificando.

Luego que fueron generadas las imágenes base se realiza la instanciación del escenario, por parte del educador, tomando como entrada su especificación y el número de instancias a crear. Al finalizar la etapa se tiene todas las instancias indicadas prontas para ser utilizadas por los estudiantes.

La siguiente etapa involucra la corrección del trabajo del estudiante por parte del educador. La evidencia entregada por el estudiante es validada contra la versión original según la función de validación indicada en la especificación

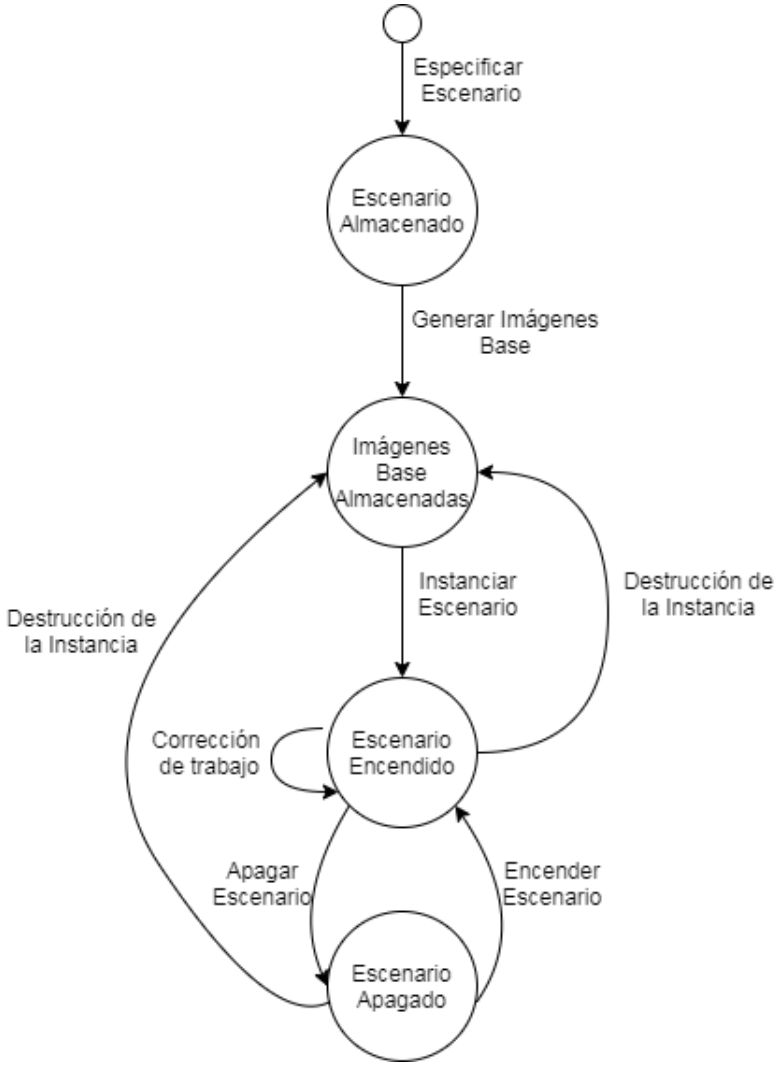


**Figura 3.2:** Estados por los que transita un escenario.

del escenario.

Por último, se tiene la destrucción de la instancia la cual elimina todos los elementos generados durante la instanciación. Esta acción también es realizada por el educador.

Durante todo este proceso se tiene la figura del usuario administrador que debe asegurar el correcto funcionamiento de los distintos componentes de *hardware* y *software* para permitir la normal operativa de la plataforma.

## 3.2. Requerimientos

A continuación se listan los requerimientos funcionales y no funcionales identificados, señalando su prioridad. La prioridad del requerimiento hace alusión a la necesidad que este sea provisto por el prototipo final a implementar.

### 3.2.1. Requerimientos funcionales

#### RF01 - Creación de escenarios

La plataforma debe permitir a un educador definir escenarios realistas donde se puedan llevar a cabo prácticas de seguridad informática como por ejemplo ataques Man In The Middle (MitM), *hardening* de sistemas y *password cracking*. Se debe poder especificar los *guests* que participan en el escenario, las redes y la parametrización de dichos *guests*. Estos escenarios se deben poder almacenar en un repositorio de escenarios. Prioridad: **Alta**.

#### RF02 - Instanciación de escenarios

La plataforma debe permitir a un educador instanciar varios clones de un escenario tomado del repositorio y habilitarlos para determinados equipos de estudiantes. Prioridad: **Alta**.

#### RF03 - Control de instancias de escenarios

La plataforma debe permitir a un educador visualizar el estado de una instancia particular de un escenario, eliminar dicha instancia, desplegarla nuevamente con la configuración base del escenario y suspender o apagar las máquinas de la instancia. Prioridad: **Alta**.

#### RF04 - Acceso a instancias de escenario

La plataforma debe permitir a un estudiante acceder a las instancias de escenarios habilitados para su equipo y conectarse a la estación o estaciones de trabajo. Prioridad: **Alta**.

#### RF05 - Monitoreo de la plataforma

La plataforma debe permitir al administrador monitorear el estado de la plataforma a nivel global o a nivel de instancias de escenario. Se debe poder

visualizar el uso de recursos y las actividades de los usuarios. Prioridad: **Media**.

### RF06 - Gestión de usuarios

La plataforma debe permitir al administrador crear un grupo compuesto de estudiantes, asignándole a cada uno sus credenciales para el acceso a la plataforma o a las instancias de escenarios habilitados. También debe permitir crear equipos de estudiantes dentro de ese grupo. Por otra parte, debe permitir crear usuarios educadores y asignarlos a grupos. Además debe permitir crear usuarios administradores. Prioridad: **Media**.

### RF07 - Autenticación delegada

La plataforma debe permitir a los usuarios autenticarse utilizando servicios externos. En particular la autenticación se podría realizar utilizando las credenciales del usuario de la plataforma Moodle [27] utilizada por la Facultad de Ingeniería en el Espacio Virtual de Aprendizaje (EVA) [40]. Prioridad: **Media**.

### RF08 - Progresión de escenario

La plataforma debe permitir a un educador especificar la forma en que un escenario progresa. Se debe permitir la ejecución de ciertas acciones, como por ejemplo ataques automáticos, mientras el estudiante lleva a cabo una práctica. Prioridad: **Baja**.

### RF09 - Seguimiento de las instancias de escenarios

La plataforma debe permitir a un educador realizar un seguimiento de cada instancia de escenario particular. Se puede evaluar el avance del equipo en la realización de las tareas del escenario, monitorear la actividad del equipo en la plataforma, analizar el tiempo de trabajo y los recursos que tomó completar la práctica. Prioridad: **Baja**.

### RF10 - Apagado automático de instancias de escenario

La plataforma debe suspender o apagar las máquinas de la instancia si el estudiante no se encuentra trabajando en esta por un período de tiempo configurable. Prioridad: **Baja**.

**RF11 - Control de instancias de escenarios por estudiantes**

La plataforma debe permitir a un estudiante visualizar las instancias que tiene habilitadas y ver su avance en la realización de las tareas de cada instancia. Prioridad: **Baja**.

### 3.2.2. Requerimientos no funcionales

**RNF01 - Aislamiento**

La plataforma debe estar aislada de los ambientes de producción. Además, los escenarios desplegados deben mantenerse aislados uno de otros y de la infraestructura base de la plataforma. Prioridad: **Alta**.

**RNF02 - Realista**

La plataforma debe permitir la creación e instanciación de escenarios lo más realistas posibles. Se debe permitir la utilización de distintos SO como Linux, Windows y Android. Prioridad: **Alta**.

**RNF03 - Interfaz de Consola**

La plataforma debe proveer una interfaz de consola para educadores y administradores ya que estos son usuarios expertos. Prioridad: **Alta**.

**RNF04 - Acceso Remoto**

La plataforma debe permitir el acceso remoto por parte de todos los usuarios. Prioridad: **Alta**.

**RNF05 - Usabilidad**

La plataforma debe ser fácil de utilizar y permitir la operativa para cada tipo de usuario. En este sentido se espera que cada usuario pueda utilizar la plataforma sin problemas luego de la lectura de las instrucciones de uso teniendo en cuenta que para los usuarios estudiantes, educadores y administradores se espera un conocimiento en computación de nivel bajo, medio y alto respectivamente. Prioridad: **Alta**.

**RNF06 - Facilidad para la Especificación de Escenarios**

La plataforma debe proveer un lenguaje de especificación de escenarios intuitivo y entendible por el usuario educador del cual se espera un nivel de conocimiento de computación de nivel medio. Prioridad: **Alta**.

**RNF07 - Escalable**

La plataforma debe permitir el escalamiento vertical y horizontal de forma fácil y sin modificaciones en el *software*. Prioridad: **Media**.

**RNF08 - Performance**

La plataforma debe permitir, como mínimo, la instanciación y funcionamiento concurrente de 30 escenarios, compuestos de tres máquinas cada uno, de forma óptima. En el futuro se debe permitir el escalamiento para soportar el trabajo concurrente de 60 escenarios. Prioridad: **Media**.

**RNF09 - Mantenible**

La plataforma debe ser fácil de mantener y extender sus funcionalidades por un equipo diferente al equipo de desarrollo inicial. Prioridad: **Media**.

**RNF10 - Despliegue**

La plataforma debe abstraerse de la infraestructura base de forma que el despliegue pueda realizarse de forma sencilla en distintas configuraciones de *hardware*. Prioridad: **Media**.

**RNF11 - Escenarios Portables**

La plataforma debe permitir definir escenario portables que puedan ser compartidos entre distintas instancias de la plataforma. Prioridad: **Baja**.

**RNF12 - Configurable**

La plataforma debe ser fácilmente configurable. Debe permitir la actualización de sistemas operativos, aplicaciones y otras tecnologías utilizadas. Prioridad: **Baja**.

**RNF13 - Interfaz Gráfica**

La plataforma debe proveer una interfaz gráfica para todos los usuarios. Esta interfaz permite la definición de escenarios, control de escenarios, entre otras funcionalidades. Prioridad: **Baja**.

## 3.3. Casos de Uso

Se presentan los casos de usos que deben ser satisfechos por la plataforma, con el fin de brindar más detalle de la operativa. Tener en cuenta que se presentan únicamente aquellos casos de uso de las funcionalidades consideradas críticas que se corresponden con las principales acciones de un usuario educador en la plataforma.

### 3.3.1. Caso de Uso 1: Especificar un escenario.

**Descripción**

El caso de uso comienza cuando el educador desea crear una nueva especificación de escenario. Una vez realizada la especificación, es almacenada en el repositorio de escenarios.

**Pre-condiciones**

No tiene.

**Flujo de Eventos**

1. El educador declara en la especificación de escenarios aspectos generales: nombre y descripción del escenario.
2. El educador declara en la especificación de escenarios los *guests* participantes: imagen base del sistema operativo, herramientas a instalar, rol del *guest* en el escenario, grupos y archivos.
3. El educador declara en la especificación del escenario las redes participantes: nombre, rango de Internet Protocol (IP)s y filtrado de tráfico.
4. El educador declara en la especificación del escenario la evidencia con la que determinar si el estudiante completó el escenario: dónde se encuentra la evidencia y la función de validación a utilizar.

5. El educador declara en la especificación del escenario las reglas utilizadas para lograr el monitoreo constante del trabajo del estudiante.
6. El educador declara en la especificación del escenario los distintos parámetros que varían según cada instancia del mismo escenario y como deben parametrizarse.
7. El educador le indica a la plataforma que desea almacenar la nueva especificación del escenario.

#### Post-condiciones

Se generó una especificación de un escenario y se almacenó en el repositorio de escenarios.

### 3.3.2. Caso de Uso 2: Generación de imágenes base

#### Descripción

El caso de uso comienza cuando el educador desea generar las imágenes base de un escenario a partir de una especificación de escenario. Una vez finalizado el caso de uso, se generaron las imágenes de los distintos *guests* del escenario.

#### Pre-condiciones

Existe la especificación de escenario en el repositorio de escenarios.

#### Flujo de Eventos

1. El educador le indica a la plataforma que desea generar las imágenes base del escenario.
2. Automáticamente, la plataforma genera las imágenes base para cada *guest* del escenario y las almacena en un repositorio de imágenes de sistema operativo.

#### Post-condiciones

Se generaron las imágenes base de los *guests* del escenario y se almacenaron en un repositorio de imágenes base.

### 3.3.3. Caso de Uso 3: Instanciación de un escenario

#### Descripción

El caso de uso comienza cuando el educador desea instanciar un escenario para un número de grupos determinado. Se crean tantas instancias del escenario como fueron solicitadas y cada instancia se parametriza según lo indicado en la especificación del escenario.

#### Pre-condiciones

Existe la especificación de escenario en el repositorio de escenarios. Existen las imágenes base correspondientes.

#### Flujo de Eventos

1. El educador le indica a la plataforma que desea instanciar un escenario para una cierta cantidad de grupos.
2. Para cada grupo se crea una instancia del escenario y se parametriza la misma.

#### Post-condiciones

Se crearon las instancias del escenario. Las instancias ya son accesibles por los estudiantes.

### 3.3.4. Caso de Uso 4: Iniciado (Apagado) de Escenario

#### Descripción

El caso de uso comienza cuando el educador desea iniciar (apagar) instancias del escenario. Una vez finalizado el caso de uso los elementos de las instancias seleccionadas se encuentran prendidos (apagados).

#### Pre-condiciones

Existe una especificación de un escenario. Existe al menos una instancia del escenario instanciada.

#### Flujo de Eventos

1. El educador le indica a la plataforma que desea iniciar (apagar) instancias de un escenario.
2. La plataforma, para cada instancia seleccionada del escenario, prende (apaga) todos los elementos de las instancias seleccionadas.

#### Post-condiciones

Se iniciaron (apagaron) los elementos de las instancias seleccionadas del escenario.

### 3.3.5. Caso de Uso 6: Destrucción de las instancias

#### Descripción

El caso de uso comienza cuando el educador desea destruir instancias del escenario. Una vez finalizado el caso de uso todos los elementos creados al momento de instanciar el escenario son destruidos.

#### Pre-condiciones

Existe una especificación de un escenario. Existe al menos una instancia del escenario disponible.

#### Flujo de Eventos

1. El educador le indica a la plataforma que desea destruir instancias de un escenario.
2. La plataforma, para cada instancia seleccionada del escenario, elimina todos los elementos creados al momento de la instanciación.

#### Post-condiciones

Se destruyeron las instancias seleccionadas del escenario.

### 3.3.6. Caso de Uso 7: Corrección del trabajo en cada instancia

#### Descripción

El caso de uso comienza cuando el educador desea validar el trabajo realizado por los estudiantes para cada instancia de un escenario. Para esto, la plataforma verificará las evidencias según lo declarado en la especificación del escenario.

#### Pre-condiciones

Existe una especificación de un escenario, y al menos una instancia de éste. La evidencia generada por el estudiante y el resultado esperado están disponibles.

#### Flujo de Eventos

1. El educador le indica a la plataforma que desea verificar el trabajo realizado por los estudiantes para cada instancia del escenario.
2. La plataforma, para cada instancia del escenario, verifica la evidencia contra el resultado esperado utilizando la función de validación declarada en la especificación del escenario.

#### Post-condiciones

Se corrigió el trabajo de los estudiantes para todas las instancias del escenario.

## 3.4. Análisis de las tecnologías de virtualización

En la sección 2.2.2 se identificaron tres tecnologías de virtualización (KVM, Xen y LXC), cada una representativa de una técnica de virtualización distinta. En esta sección se comparan las mismas en base a los requerimientos identificados. Varios autores comparan estas tecnologías en términos de *performance*, *scheduling* de los recursos, aislamiento, usabilidad y demás factores [7, 15,

37, 43, 46]. Aquí se presentan las conclusiones relevantes extraídas de estos trabajos; una comparación de mayor extensión se presenta en [17].

Uno de los requisitos de la plataforma a desarrollar es que esta debe poder escalar de manera adecuada; es decir, debe poder soportar un gran número de usuarios concurrentes (RNF07 y RNF08). A su vez, otro requisito importante es el aislamiento de los recursos de cada VM (RNF01), y un *scheduling* equitativo de los recursos disponibles entre las VMs que se encuentran ejecutando (RNF05 y RNF08). También son necesarios escenarios realistas (RNF02), por lo que es preferible poder virtualizar una gran variedad de SOs.

En primer lugar, se observa que al utilizar técnicas de *full-virtualization*, KVM permite virtualizar una mayor variedad de SOs, facilitando satisfacer el requerimiento RNF02. En contraste, Xen con *paravirtualization* solo soporta SOs portados y LXC solo soporta contenedores Linux.

Las tecnologías basadas en contenedores son las que obtienen el mejor desempeño, pero al costo de un pobre aislamiento de los recursos, lo cual puede llegar a presentar serios problemas; por ejemplo, *starvation*. KVM y Xen en modo *paravirtualization* tienen resultados casi nativos en virtualización de CPU. KVM desempeña mejor en cuanto a memoria, aunque sufre degradaciones en cuanto a acceso a disco. Xen tiene resultados casi nativos en la mayoría de los casos. Ambas soluciones escalan bien a medida que se agregan VMs concurrentes, aunque en algunos casos el desempeño de KVM es mejor. Del trabajo presentado en [15] se concluye que KVM puede estar mejor equipado para soportar un laboratorio virtual de seguridad informática, debido a que presenta mejores resultados en la ejecución concurrente de tareas típicas dentro de escenarios virtuales de un laboratorio.

Del análisis realizado, se concluye que KVM se alinea mejor que las otras tecnologías con los requerimientos establecidos para un *cyber range*. Permite virtualizar una gran variedad de sistemas operativos, lo cual es importante para obtener escenarios realistas (RNF02). Si bien no tiene la mejor eficiencia computacional de todas las soluciones, demuestra comportarse de manera adecuada con un gran número de VMs concurrentes, ejecutando tareas típicas de un laboratorio de ciberseguridad (RNF07 y RNF08). Y por último, a diferencia de los contenedores, tiene un buen aislamiento de los recursos entre VMs y realiza un *scheduling* equitativo de los mismos (RNF01 y RNF08).

## 3.5. Análisis de *cyber ranges*

En la tabla 3.1 se puede observar una comparación de las distintas tecnologías evaluadas para el diseño e implementación del nuevo laboratorio del GSI. Se incluye el laboratorio actual, LaSI, para observar los requisitos insatisfechos. También se incluye OpenNebula, el cual fue descripto en 2.2.5 y fue evaluado de forma práctica. Luego, se incluyen KYPO, CyberWiser.eu y CyTrONE, tres de las implementaciones de *cyber range* que se consideraron de mayor interés y potencial. Finalmente, se agrega a la comparación la solución final que será introducida en mayor detalle en el capítulo 4 de diseño. Esta solución final está compuesta por una versión modificada de CyTrONE, la cual se extendió para agregar un mayor control sobre las instancias de los escenarios, junto con un despliegue del Elastic Stack. En la figura 3.3 se resume la cantidad de requisitos satisfechos por cada tecnología.

Es importante destacar que este análisis comparativo fue realizado en base a investigación de la documentación de las herramientas y en ciertos aspectos carece de una verificación práctica de que cierto requerimiento sea realmente satisfecho. Para OpenNebula, CyTrONE y Elastic Stack, sí se realizaron pruebas prácticas que permitieron recabar mayor información.

El LaSI solo cumple con un 37.5 % de los requisitos identificados y un 70 % de los requisitos con prioridad alta. Aunque el laboratorio funciona correctamente y logra su objetivo de brindar educación práctica en ciberseguridad a muchos alumnos, se observa que hay lugar para la mejora en cuanto a usabilidad de la plataforma, gestión de los escenarios, corrección automática y otros requisitos de menor prioridad.

La principal dificultad que presenta el LaSI es la definición de escenarios, debido a que no cuenta con un lenguaje que permita especificar los componentes de un escenario y su configuración. La definición de los escenarios es realizada manualmente por los educadores generando directamente las imágenes base de las máquinas que participan en este. Esto a su vez, genera que la tarea de actualizar o modificar los escenarios no sea nada sencilla. Por otra parte, la evaluación del trabajo del estudiante no se encuentra completamente automatizada, ya que el educador debe implementar *scripts* que se encarguen de validar las evidencias entregadas por los estudiantes.

La herramienta OpenNebula cumple con un 58.3 % de los requisitos identificados y también con un 70 % de los requisitos con prioridad alta. Aunque

OpenNebula esta pensado para ser utilizado en el despliegue de IaaS, presenta bastante potencial para ser utilizado en un *cyber range*, o al menos como plataforma base, como en el caso de CyberWiser.eu.

KYPO fue presentado en la sección 2.1.3.2. Cumple con un 70.8 % de todos los requerimientos y un 70 %, 71.4 % y 71.4 % de los de prioridad alta, media y baja respectivamente. Cuenta con un lenguaje de definición de prácticas, el cual puede procesar para desplegar las instancias de entrenamiento. Esto permite compartir escenarios desarrollados entre distintas instalaciones de KYPO. También permite que estudiantes y educadores puedan monitorear en tiempo real el avance en la realización de las prácticas, satisfaciendo los requerimientos RF09 y RF11.

Al utilizar la plataforma OpenStack como base para el despliegue de los escenarios virtuales, KYPO satisface diversos requerimientos no funcionales como RNF01, RNF07, RNF08 y RNF10. Sin embargo, los requerimientos de *hardware* de OpenStack y su dificultad de configuración hacen que se considere que KYPO no satisface el requerimiento RNF12.

KYPO es *open-source* y cuenta con una extensa y organizada documentación, estando a su vez desplegado sobre una herramienta open-source con gran soporte como *OpenStack*, por lo que se cree que es mantenible; cumpliendo con el requerimiento RNF09.

El enfoque de KYPO está en sesiones de entrenamiento cortas (menos de un día) del tipo CTF, a diferencia del enfoque del LaSI que es mantener los escenarios levantados durante semanas para que los estudiantes puedan realizar las prácticas. Por esta razón se cree que KYPO no cuenta con un control preciso de instancias de escenarios, RF03.

CyberWiser.eu, presentado en la sección 2.1.3.1, cumple con un 66.6 % del total de los requerimientos y un 70 %, 57.1 % y 71.4 % de los de alta, media y baja prioridad respectivamente. CyberWiser.eu utiliza OpenNebula como infraestructura base para el despliegue de los escenarios y comparte mucha de las fortalezas que KYPO posee. Sin embargo, la principal desventaja de CyberWiser.eu es que no es de código abierto y no se puede acceder a la herramienta, lo cual hace que algunos requerimientos no sean satisfechos y otros no se conozca la respuesta. También imposibilita el hecho de poder utilizarlo para implementar el laboratorio del GSI.

Sin embargo, el diseño de CyberWiser.eu presenta una herramienta potente y flexible de la cual se pueden extraer ideas. En particular, en cuanto a

metodología de evaluación, CyberWiser.eu toma el enfoque de monitoreo constante descrito en la sección 2.1.4 mediante el uso de un SIEM. Este enfoque tiene diversas ventajas entre las cuales destacan: monitorear el avance de los estudiantes para llegar a la solución, pudiendo reconocer cuando los mismos alcanzan ciertas banderas y que acciones tomaron para alcanzarlas; monitorear la plataforma en busca de actividad extraña o intentos de trampa; analizar el uso de recursos de la plataforma en general. Esta metodología es diferente a la utilizada por KYPO, la cual cae en la categoría de CTF mencionada en la sección 2.1.4. Idealmente, una combinación de ambas metodologías de evaluación podría sumar las ventajas de cada una, resultando en un laboratorio más completo y flexible.

Por último, CyTrONE, presentado en la sección 2.1.3.3, cumple con un 83.3 % del total de los requerimientos y un 90 %, 85.7 % y 71.4 % de los de prioridad alta, media y baja respectivamente. CyTrONE es *open-source* y cuenta con una extensa y amigable documentación, por lo cual es mantenible y fácil de adaptar a las necesidades del GSI.

El componente CyRIS permite desplegar escenarios realistas y flexibles a partir de una descripción de escenario sencilla. El componente CyPROM permite añadir progresión al escenario, modificando el mismo en base al avance del estudiante. El componente CyLMS permite integrar la plataforma con un LMS para subir contenido educativo y a su vez realizar cuestionarios relacionados a las prácticas, llevando a cabo una metodología de evaluación del estilo CTF, como se mencionó en la sección 2.1.4. Finalmente, el *framework* CyTrONE integra todas estas soluciones y permite accederlas desde una única interfaz fácil de utilizar, sumando a su vez la gestión de usuarios de la plataforma.

Se destaca que CyTrONE, aunque no cubra todos los requerimientos identificados, es la solución más completa de todas las evaluadas.

En el siguiente capítulo se introduce el diseño de la solución final, la cual explota las fortalezas de las demás tecnologías presentadas hasta el momento.

| | LaSI | OpenNebula | KYPO | CyberWiser.eu | CyTrONE | Solución Final |
|---|---|---|---|---|---|---|
| RF01 - Creación de escenarios | Si | Si | Si | Si | Si | Si |
| RF02 - Instanciación de escenarios | Si | Si | Si | Si | Si | Si |
| RF03 - Control de instancias de escenarios | No | Si | No | ND | No | Si |
| RF04 - Acceso a instancias de escenario | Si | Si | Si | Si | Si | Si |
| RF05 - Monitoreo de la plataforma | No | Si | Si | Si | No | Si |
| RF06 - Gestión de usuarios | No | Si | Si | Si | Si | Si |
| RF07 - Autenticación delegada | No | No | No | No | Si | Si |
| RF08 - Progresión de escenario | No | No | Si | Si | Si | Si |
| RF09 - Seguimiento de las instancias de escenarios | No | No | Si | Si | No | Si |
| RF10 - Apagado automático del escenario | No | Si | No | ND | No | No |
| RF11 - Control de instancias de escenarios por estudiantes | No | No | Si | Si | Si | Si |
| RNF01 - Aislamiento | Si | Si | Si | Si | Si | Si |
| RNF02 - Realista | Si | No | Si | Si | Si | Si |
| RNF03 - Interfaz de Consola | Si | Si | No | ND | Si | Si |
| RNF04 - Acceso remoto | Si | Si | Si | Si | Si | Si |
| RNF05 - Usabilidad | No | No | ND | ND | Si | Si |
| RNF06 - Facilidad para la especificación de escenarios | No | No | Si | Si | Si | Si |
| RNF07 - Escalable | No | Si | Si | Si | Si | Si |
| RNF08 - Performance | Si | Si | ND | ND | Si | Si |
| RNF09 - Mantenible | Si | No | Si | No | Si | Si |
| RNF10 - Despliegue | No | Si | Si | Si | Si | Si |
| RNF11 - Escenarios portables | No | No | Si | No | Si | Si |
| RNF12 - Configurable | No | No | No | Si | Si | Si |
| RNF13 - Interfaz Grafica | No | Si | Si | Si | Si | Si |

**Tabla 3.1:** Comparación de distintas tecnologías evaluadas para el diseño del nuevo laboratorio del GSI. ND significa "No Disponible" y se utiliza en los casos en los que esa información no se pudo obtener a partir de la documentación.

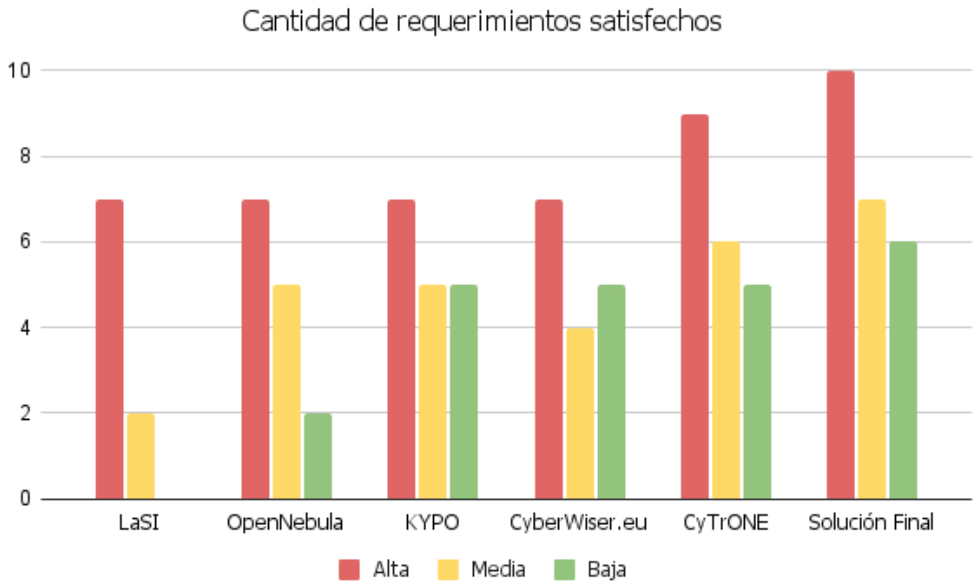


**Figura 3.3:** Cantidad de requerimientos satisfechos por cada una de las tecnologías evaluadas.

# Capítulo 4

# Diseño

En este capítulo se presenta el diseño de la plataforma. En primer lugar se presenta en mayor detalle el diseño de CyRIS, que fue utilizado como base para implementar el prototipo. Luego, se introducen las mejoras aplicadas a este componente para cubrir uno de los requerimientos con prioridad alta y la integración de Elastic Security a la solución pensada. Finalmente, se presenta la solución final diseñada, para la cual se hace un análisis teórico de los requerimientos que cumple y de la cual se extrae el prototipo final.

## 4.1. Diseño de CyRIS

A partir del relevamiento de las herramientas realizado, se identificó que el *framework* CyTrONE es un gran punto de partida para la implementación del nuevo laboratorio del GSI debido a que cumple una gran cantidad de los requerimientos identificados. Además, tiene la ventaja de ser *open-source* y de estar bien documentado, lo que facilita su modificación y extensión.

Para este proyecto se utiliza principalmente el componente denominado CyRIS, que como fue presentado en la sección 2.1.3.3, es definido como una herramienta para facilitar la creación y administración de escenarios abstrayéndose de la infraestructura de *hardware* que se utilice por debajo.

Este componente permite la instanciación de escenarios a partir de una especificación. Esta especificación se encuentra escrita en lenguaje YAML [45] y presenta tres secciones principales:

- *Host settings*: en esta sección se identifican los *hosts* que se utilizaran para desplegar las instancias del escenario.

- *Guest settings*: se especifican los sistemas que participan del escenario así como las configuraciones necesarias (como por ejemplo *software* a instalar) y acciones a ejecutar (por ejemplo, la ejecución de ataques).
- *Clone settings*: esta sección identifica los *guest* que participan de la instancia así como la topología de la red del escenario. Además se debe indicar cuantas instancias del escenario y en que *host* son desplegadas.

En la figura 4.1 se puede observar el flujo de la instanciación de los escenarios. A partir de una descripción del escenario y una base de datos de imágenes base, el sistema configura y despliega las instancias del escenario. Para llevar a cabo esta tarea, el primer paso es instanciar VMs base para realizar en ellas las configuraciones indicadas en la especificación. Una vez que se realizaron las configuraciones necesarias, estas VMs base son clonadas y distribuidas entre los distintos *hosts* para generar las instancias de los escenarios. En este último paso, se configuran además las redes que conforman al escenario.

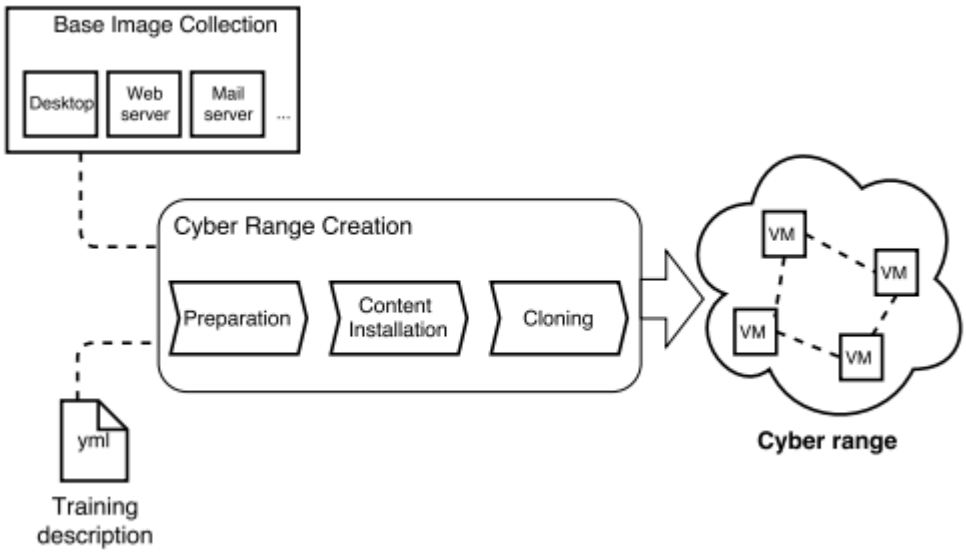


**Figura 4.1:** Flujo de trabajo de CyRIS. Extraído de [32]

## 4.2. Mejoras implementadas

### 4.2.1. Gestión de las instancias de escenarios

Realizando pruebas con el *framework* CyTrONE y específicamente con el componente CyRIS (en sus versiones más recientes), se observó que éste no per-

mite manejar los escenarios instanciados con el nivel de granularidad necesario. Por ejemplo, si se especifica un escenario y se despliegan múltiples instancias de dicho escenario, solo se pueden eliminar todas las instancias a la vez, pero no instancias individuales. Eliminar instancias individuales es una funcionalidad vital provista por el LaSI, utilizada frecuentemente por los docentes del GSI. Si un estudiante daña de forma irreparable su instancia de escenario, dicha instancia se elimina y al estudiante se le da acceso a una instancia nueva.

Por otro lado, CyRIS no cuenta con una interfaz sencilla que permita apagar y prender los *guests* que conforman instancias individuales del escenario. Este control puede ser útil para tener instancias ociosas apagadas, en el caso de que los estudiantes no estén accediendo a ellas o en el caso de que se hayan creado instancias adicionales para asignar a estudiantes que pierdan las suyas.

Estas razones hacen que CyRIS por si solo no satisfaga el requerimiento RF03. En un intercambio de correos electrónicos con el principal desarrollador de CyRIS, Razvan Beuran [35], se presentaron los objetivos del trabajo de este proyecto y se declaró la intención de utilizar la herramienta CyRIS en la implementación del nuevo LaSI. Gracias a este intercambio, se reveló que un control con mayor precisión de las instancias de un escenario sería un gran aporte, dado que actualmente esta gestión se resuelve de forma manual.

Para cubrir estas falencias se diseñó e implementó el *script* `instance_management`. Este *script* permite listar el estado de los *guests* de un conjunto de instancias de un escenario y también permite prenderlos, apagarlos o destruirlos. Para conocer en que *hosts* están desplegados los *guests* así como otra información importante sobre el escenario y la plataforma, `instance_management` consume información de dos archivos:

- Uno es el archivo `range_details-crXX.yml`, el cual es un archivo que se auto-genera cuando se instancia por primera vez el escenario con identificador `XX`, y que contiene información relevante a los *guests* que conforman las instancias y en que *hosts* físicos están desplegadas dichas instancias. Fue necesario modificar el *script* que genera este archivo al momento de la instanciación, para que se incluya en dicho archivo también el nombre del usuario de administración de los *hosts* donde se despliega el escenario. Este cambio solo requirió una línea y no afectó en absoluto el resto de las funcionalidades.
- Otro archivo que se consume es el de configuración de CyRIS, del cual

se extrae información sobre las rutas a los directorios de instalación de CyRIS y de los escenarios.

Aunque `instance_management` posee su propia interfaz de consola para acceder a las funcionalidades mencionadas, también se modificó la interfaz principal de CyRIS (el *script* `cyris.py`) para poder acceder a estas funcionalidades desde la misma, unificando la instanciación, destrucción y gestión de los escenarios en un solo *script*.

El *script* fue implementado en lenguaje Python para integrarse de forma fácil con el resto de la plataforma. Este fue diseñado como un módulo independiente que no requiere de modificaciones mayores en el código actual de la herramienta. La justificación para esto es que CyRIS aún se encuentra en desarrollo, y al no conocer en que *features* se esta trabajando y que partes del código están siendo modificadas, implementar `instance_management` como un módulo independiente facilitará su integración con nuevas modificaciones a la plataforma. Instalar este *script* es tan sencillo como copiarlo en la carpeta *main* de la instalación de CyRIS, junto con una pequeña modificación en el código que se comentará más adelante.

Algunos cuidados adicionales se tuvieron que llevar a cabo al momento de encender los *guests* de una instancia del escenario. Ciertas configuraciones como establecer las reglas del *firewall* o los *gateways* de la red son realizadas de forma no persistente al momento de la instanciación por parte de CyRIS. Esto implica que al apagar las instancias, las configuraciones se pierden. Por lo tanto, al momento del encendido de los *guests* en una instancia, se les debe reconfigurar los *gateways* en sus interfaces de red y las reglas del *firewall* según lo especificado en la descripción del escenario original. Al momento, `instance_management` lleva a cabo estas reconfiguraciones. Sin embargo, se observa que este trabajo se podría simplificar considerablemente si se introdujeran en CyRIS las modificaciones necesarias para hacer que estas configuraciones fueran permanentes al momento que se instancia por primera vez el escenario.

### 4.2.2. Monitoreo del trabajo del estudiante

El componente CyRIS por si solo no provee funcionalidades para la evaluación del trabajo realizadon por un estudiante. Para esto se cuenta con el componente CyLMS que provee de la integración con un LMS como Moodle,

donde el estudiante puede contestar cuestionarios referentes a las metodologías o técnicas aplicadas para cumplir con los objetivos y entregar evidencias de que ha realizado el trabajo correctamente.

Para lograr este último punto, las prácticas se deben implementar siguiendo la metodología CTF, descripta en la sección 2.1.4.

Una de las falencias identificadas de este enfoque es que no se puede saber a ciencia cierta el camino que llevo al estudiante hacia la solución; si se puede, por ejemplo, introducir distintas *flags* en los distintos caminos que puede tomar un estudiante hacia la solución final, y evaluar que camino tomó dependiendo de que *flag* suministra.

Otro de los problemas es que la evaluación es en cierto modo binaria, pues esta se basa en que el estudiante entregue o no las *flags* correctas. No se puede evaluar el caso en el que este haya realizado cierto avance pero no llegue a encontrar la *flag*.

Por otra parte, se tiene el problema en que el educador no puede conocer de forma inmediata el avance de un estudiante dentro de un escenario, ya que se debe esperar a que este entregue las *flags* o responda los cuestionarios que por lo general se dan al final de la práctica. Si se tiene la capacidad de conocer en tiempo real el avance del estudiante, entonces se podrían implementar mecanismos en los que la plataforma reaccione a las acciones del estudiante y se generen modificaciones automáticas en los escenarios. Por ejemplo, podría darse el caso que el estudiante presenta un avance más que satisfactorio dentro del escenario. Teniendo esta información en tiempo real podrían tomarse acciones para modificar la dificultad del resto de la práctica generando así que el escenario se adapte a las habilidades del estudiante y presente un mayor desafío.

Para sobreponerse a estos problemas es que se plantea agregar la metodología de evaluación en el que el monitoreo del avance del estudiante sea constante, la cual fue descripta en la sección 2.1.4. Para cumplir con este objetivo se ha utilizado la tecnología denominada Elastic Stack, descrita brevemente en la sección 2.2.6 y en mayor detalle en [17].

La idea de utilizar dicha tecnología es desplegar en los sistemas participantes de un escenario el *endpoint* para recopilar las acciones realizadas por un estudiante. Esta información es enviada al SIEM donde será analizada y se podrán generar alertas ante ciertos eventos claves. De esta forma se puede automatizar la corrección de las prácticas conociendo en tiempo real el avance

del estudiante y teniendo la posibilidad de, en base a este avance, brindar *feedback* con menos demoras con el fin de mejorar la experiencia de aprendizaje; por ejemplo, en caso de que el estudiante este realizando acciones incorrectas continuamente, el educador puede notar esto y asistir al estudiante. Por otra parte, conocer el avance en tiempo real puede ser útil para modificar el escenario con el objetivo de adaptarse a las habilidades del estudiante y por lo tanto generar un entrenamiento personalizado.

Se debe destacar que estas ventajas se logran a costo de una mayor necesidad de infraestructura, ya que desplegar este sistema cuenta con considerables requerimientos de *hardware*. Por otra parte, también es necesario implementar de forma manual las reglas que se utilizan para la detección de los eventos de seguridad.

### 4.2.3. Fallas encontradas en CyRIS

Realizando pruebas con CyRIS en un ambiente distribuido con dos servidores, como es explicado en mayor detalle en el capítulo 5, se encontraron ciertas fallas, las cuales se corrigieron como parte del desarrollo.

Para la configuración automática de los *guests*, los *hosts* en los que se despliegan deben tener acceso Secure SHell (SSH) sin contraseña a los mismos. Para esto, se copia la clave pública del *host* en el archivo `authorized_keys` de cada uno de los *guest* que soporta. Sin embargo, debido a una falla en el programa de instanciación, los *guests* desplegados en los servidores esclavos obtienen una copia de la clave pública del *host* maestro y no del servidor esclavo donde están desplegados, lo cual inhabilita el acceso SSH sin contraseña e impide la correcta instanciación.

Para lidiar con esta falla, debido a restricciones temporales, se optó por la solución más sencilla. En el archivo de descripción del escenario a desplegar se agregan dos tareas de configuración para cada *guest*. Una de estas tareas copia un *script* al *guest* y la otra lo ejecuta. El *script* se encarga de agregar la clave pública del servidor esclavo al archivo `authorized_keys` del *guest* y de borrar el *script* al terminar. De esta forma, la clave queda correctamente copiada a los *guests* y se permite el acceso SSH sin contraseña desde el *host* donde esta desplegado. Se remarca que esto es solo un atajo para evitar la falla y no una solución de la misma.

La otra falla identificada también esta relacionada al despliegue distribuido.

Todas las instancias de un escenario tienen un identificador, el cual las permite distinguir entre si y el cual se incluye en la definición de las redes de la instancia; la IP de una interfaz de red en una instancia se define como:

`id_escenario.id_instancia.id_segmento.id_miembro`

Donde `id_escenario` es igual para todas las instancias, `id_instancia` es el identificador de cada instancia, `id_segmento` es el identificador del segmento de red en la topología del escenario e `id_miembro` es el identificador de la interfaz participante en ese segmento.

El problema encontrado es que durante la instanciación, los identificadores se repiten entre las distintas instancias desplegadas en cada *host*. Esto es un problema ya que distintas instancias pueden tener asignadas las mismas IP, lo cual dificulta fuertemente las tareas de monitoreo de Elastic Security y también hace indistinguibles una instancia de otra, lo cual trae problemas a las nuevas funcionalidades desarrolladas que requieren poder identificar instancias particulares.

Esta falla fue corregida modificando el *script* `cyris.py` para que asigne correctamente identificadores distintos a todas las instancias de la plataforma, independientemente de si están desplegadas en *hosts* distintos.

Por otra parte, se detectó una falla en la tarea que permite configurar el *firewall* local de cada *guest*. El problema se debe al reinicio del *firewall* mientras se realiza la configuración de la VM base, lo que ocasiona que el proceso de instanciación falle. Para solucionar este problema se decidió omitir el inicio del *firewall* en la VM base y dejar que este ocurra cuando se inicia cada clon.

Por último, para permitir que las instancias desplegadas en el servidor remoto puedan comunicarse con Elastic Security, se decidió agregar a la tabla de ruteo del servidor donde se tiene desplegado Elastic Security rutas a las redes de las instancias desplegadas en el servidor remoto con este como *gateway*. Para esto se creó un *script* que debe ser ejecutado manualmente para estos casos. Además, es necesario configurar correctamente el *firewall* local de cada servidor permitiendo la comunicación entre estos.

Se crearon dos *pull requests* al repositorio público de la herramienta CyRIS con las modificaciones realizadas:

- El *pull request* [10] contiene las extensiones descriptas en la sección 4.2.1 y la modificación al *script* `cyris.py` que asigna correctamente identificadores distintos a todas las instancias de la plataforma.

- El *pull request* [9] contiene las modificaciones necesarias para omitir el inicio del *firewall* en la VM base.

El *script* que configura las rutas necesarias para la comunicación con Elastic Security no fue disponibilizado como un *pull request* ya que la configuración de estas rutas depende fuertemente del ambiente utilizado para el despliegue, que no necesariamente será igual al ambiente utilizado para el despliegue del prototipo.

## 4.3. Diseño de la solución final

En esta sección se presenta el diseño de la solución final ideada para el nuevo laboratorio del GSI, teniendo en consideración lo mencionado en las secciones anteriores de este capítulo.

En el diagrama de la figura 4.2 se puede observar el *stack* tecnológico que conforma la solución junto con el componente externo, Moodle, y los actores.

El componente CyTrONE, con las modificaciones mencionadas en 4.2.1, se encarga de la gestión de los escenarios, de la gestión del contenido de las prácticas en la plataforma Moodle y de la progresión de los escenarios. Para el despliegue y gestión de las máquinas y redes virtuales involucradas en los escenarios, el componente CyRIS hace uso de la librería Libvirt, utilizando KVM como tecnología de virtualización.

El componente Elastic Security consume información proveniente de los escenarios virtuales desplegados para implementar el monitoreo automático mencionado en 4.2.2.

El componente externo, Moodle, es un LMS en el cual se puede desplegar contenido referente a los cursos y prácticas, así como conectarse a los escenarios a través de noVNC para obtener una interfaz gráfica. Para la gestión de este contenido en Moodle, CyTrONE cuenta con el componente CyLMS. Con estas funcionalidades, este contenido se podría desplegar en la plataforma Moodle de Facultad de Ingeniería.

### 4.3.1. Comparación con otras tecnologías

En esta sección se vuelve la atención a la tabla 3.1 para comparar la solución propuesta con las demás soluciones evaluadas. Se observa que esta evaluación esta basada principalmente en la información recolectada a partir de

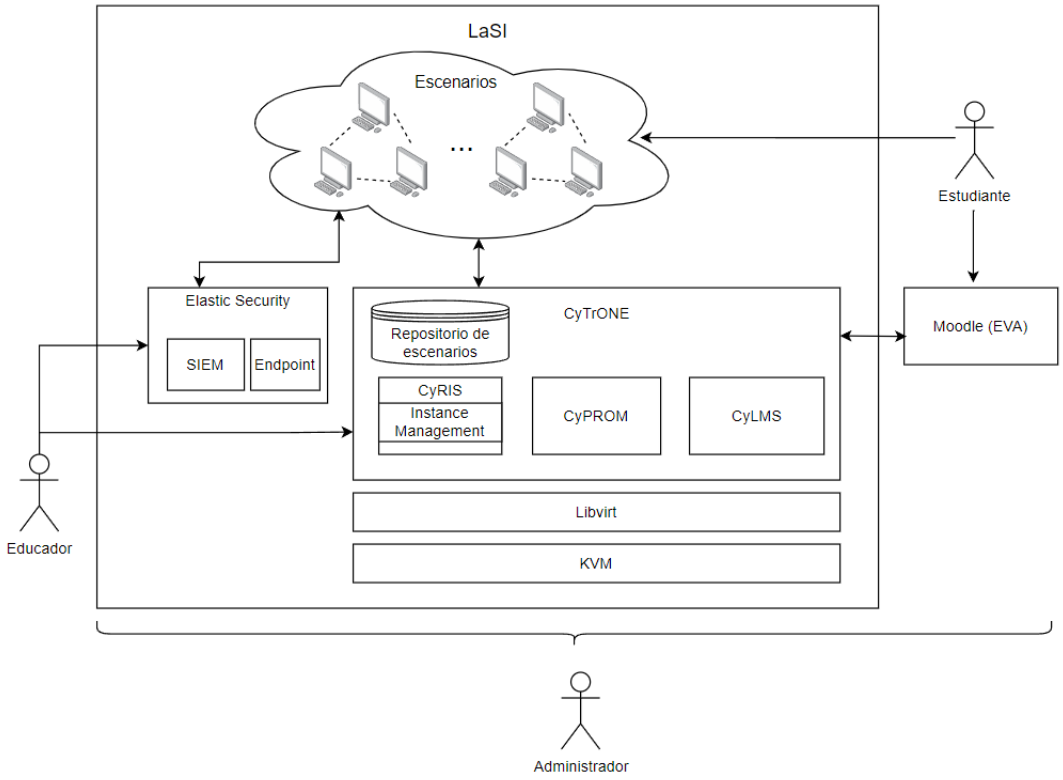


**Figura 4.2:** Diseño de la nueva solución pensada para el LaSI

las especificaciones de las herramientas y en base a pruebas realizadas con los componentes separados. Una evaluación más profunda y realista es llevada a cabo para el prototipo final en el capítulo 5.

El único requerimiento que la solución final no satisface es el de apagado automático de los escenarios, el cual tiene prioridad baja. Cumple un 95.8 % del total de requerimientos y un 100 %, 100 % y 85.7 % de los de alta, media y baja prioridad.

La última versión de CyRIS, con su lenguaje de especificación de escenarios y el despliegue de los mismos utilizando Libvirt y KVM permite satisfacer los requerimientos RF01, RF02, RF04, RNF01, RNF02, RNF06 y RNF11. El despliegue en múltiples *hosts* también ayuda a satisfacer los requerimientos RNF07, RNF08 y RNF10. Las mejoras implementadas permiten hacer un control de las instancias de escenarios desplegadas, satisfaciendo el requerimiento RF03.

La integración de Elastic Security permite visualizar las acciones de los estudiantes en la plataforma, generar avisos cuando se alcance cierta bandera o avance en una práctica, detectar acciones malintencionadas, analizar el uso

de recursos y mucho más. Esto permite cubrir ampliamente los requerimientos RF05, RF09 y RF11.

El componente CyPROM está específicamente diseñado para gestionar la progresión de los escenarios, permitiendo ejecutar ataques automáticos o tomando acciones sobre el escenario en base al avance y respuestas de los estudiantes. En este sentido, permite cubrir los requerimientos RF08, RF09, RF11 y RNF02.

CyLMS utilizado en conjunción con un despliegue de Moodle, permite disponibilizar contenido educativo referente a las prácticas en dicha plataforma, dar acceso a los *guests* de los escenarios mediante interfaz gráfica y crear cuestionarios relacionados a los conceptos para medir el aprendizaje de los estudiantes. Esto satisface los requerimientos RF04, RF06, RF09, RF11, RNF04 y RNF13. A su vez, gracias al despliegue del material y acceso a los escenarios en Moodle, se cumple el requerimiento de autenticación delegada, que ninguna de las otras tecnologías puede satisfacer sin cambios.

CyTrONE, aparte de agrupar los demás componentes y proveer una interfaz común para acceder a ellos, también ofrece funcionalidades de gestión de los usuarios estudiantiles. Todas las funcionalidades son accesibles mediante interfaces de consola fáciles de utilizar y bien documentadas. Por otro lado, Elastic Security cuenta con la interfaz gráfica, Kibana, que ayuda a analizar los datos mediante gráficas y diagramas. Estas interfaces ayudan a cubrir uno de los requerimientos de alta prioridad, la usabilidad.

# Capítulo 5

# Experimentación

En este capítulo se presenta y describe la evaluación del prototipo implementado. Se plantea el despliegue del prototipo y su uso en la instanciación de un escenario desarrollado por el grupo.

## 5.1. Evaluación del prototipo

El prototipo implementado se puede resumir en los componentes CyRIS (que permite el despliegue de escenarios a partir de una especificación) y Elastic Security (que permite el monitoreo de las actividades del estudiante dentro del la instancia del escenario, facilitando la corrección de su trabajo). Además, el componente CyRIS cuenta con la extensión realizada por el grupo, denominada `instance_management`, que permite un manejo más fino de los escenarios instanciados. Esta extensión se encarga de gestionar los escenarios permitiendo realizar las acciones de: prendido, apagado, eliminación y consulta del estado de las instancias de los escenarios. También se incluyen las correcciones mencionadas en la sección 4.2.3.

Para el prototipo final a evaluar no se tiene en cuenta la integración con la plataforma Moodle de la Facultad de Ingeniería, ya que esta requiere de modificaciones en el componente externo que están fuera del alcance de este trabajo. A su vez, el componente CyPROM no es evaluado ya que, luego de un intercambio con los desarrolladores de la herramienta, se descubrió que ciertas funcionalidades de CyPROM aún no se encuentran estables.

Para poder evaluar el prototipo se realizó su despliegue en dos servidores, como se indica en la figura 5.1. Se observa el servidor maestro que cuenta con

la instancia del prototipo propiamente dicha, CyRIS, `instance_management` y Elastic Security. Este servidor gestiona el despliegue de las instancias, que pueden darse en ambos servidores, y se encarga de proveer las funcionalidades de monitoreo a través de Elastic Security. Por otra parte se tiene el servidor esclavo que se utiliza exclusivamente para el despliegue de instancias, dicho servidor debe contar únicamente con la instalación de CyRIS. Además, ambos servidores deben contar con las tecnologías de virtualización utilizadas (Libvirt y KVM).

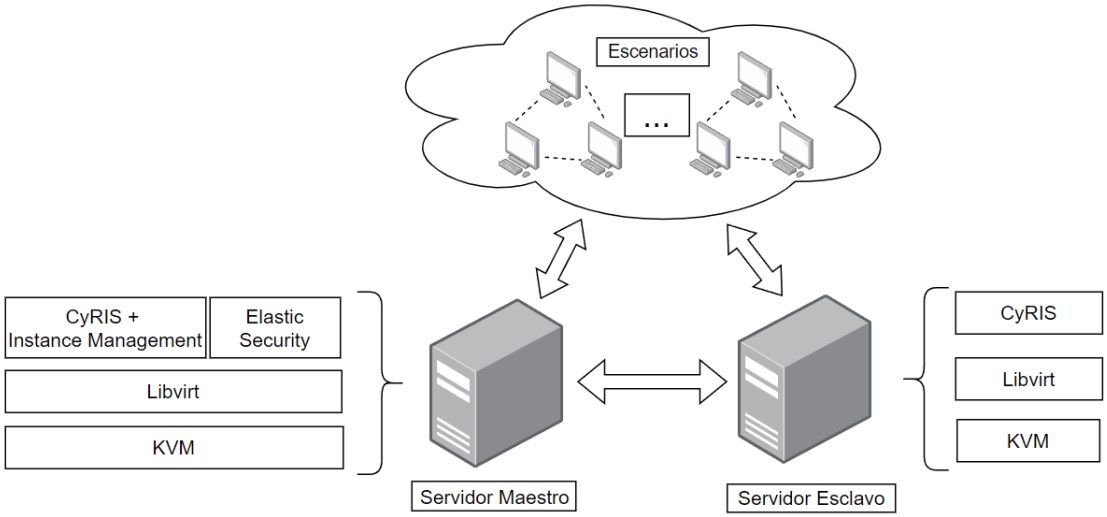


**Figura 5.1:** Diagrama de despliegue del prototipo

Para evaluar las capacidades del prototipo de forma práctica, se implementó la adaptación del escenario denominada "Acceso a shell en forma reversa", la cual es una práctica utilizada en el curso de grado y posgrado FSI. La idea es poder mostrar, a través de este ejemplo, que las prácticas actualmente implementadas para el LaSI también pueden ser implementadas por el nuevo prototipo haciendo uso de las nuevas funcionalidades que este provee. En particular, se busca demostrar la facilidad de uso del lenguaje de especificación de escenarios y la utilidad del monitoreo del trabajo del estudiante.

Esta práctica tiene como objetivo mostrar los problemas que surgen cuando no se utiliza filtrado de paquetes salientes de un *firewall*. El escenario está conformado por dos sistemas: la máquina atacante a la que el estudiante tiene acceso y desde donde lanzará el ataque, y la máquina víctima que cuenta con la instalación del servidor Apache en el que se encuentra desplegada una versión vulnerable de la aplicación `awstats`, que permite la ejecución remota de

código. Además, esta máquina carece de una política de *firewall* correctamente configurada. Por lo tanto, el objetivo del estudiante es, explotando la vulnerabilidad de `awstats`, lograr obtener acceso remoto a una consola de comandos en la máquina víctima.

El primer paso para lograr implementar la práctica, es generar su especificación. A continuación se presenta un extracto de la especificación del escenario.

```
- guest_settings:
  - id: victim
    basevm_host: host_1
    basevm_config_file: /home/cyuser/images/basevm/basevm.xml
    basevm_type: kvm
    tasks:
    # Install necessary software using official repositories
    - install_package:
      - package_manager: yum
        name: httpd
    - install_package:
      - package_manager: yum
        name: wget
    - install_package:
      - package_manager: yum
        name: policycoreutils-python
    # Copy vulnerable version of awstats to the vm
    - copy_content:
      - src: /home/cyuser/repository/vulnerable_software/awstats-6.2
        dst: /root
    # Install vulnerable version of awstats
    - execute_program:
      - program: /root/awstats-6.2/install.sh
        interpreter: bash
                              ...
- clone_settings:
 - range_id: 192
   hosts:
   - host_id: host_1
     instance_number: 1
```

```
guests:
- guest_id: attacker
  number: 1
  entry_point: yes
- guest_id: victim
  number: 1
  topology:
   - type: custom
     networks:
     - name: network
       members: attacker.eth0, victim.eth0
```

En la sección de *guest settings* se observan las tareas ejecutadas para lograr la configuración deseada del sistema denominado *victim*. En particular se aprecian las acciones de instalar aplicaciones desde los repositorios oficiales, el copiado de contenido (como puede ser la versión vulnerable de `awstats` por no encontrarse en repositorios oficiales) y la ejecución de *scripts* creados por el grupo para llevar a cabo tareas de instalación y configuración. Por otra parte, en la sección *clone settings*, se define la cantidad de instancias del escenario, en que *host* deben ser desplegadas y como se conforma la red del escenario. En este caso es una red muy sencilla que permite la comunicación directa entre los dos sistemas del escenario.

Se destaca que el prototipo facilita en gran medida el armado del escenario y permite de manera muy sencilla generar nuevas versiones del escenario (por ejemplo al utilizar otro SO u otra versión de la aplicación) con la simple modificación de la especificación. Además, tener un lenguaje para la especificación de escenario permite compartir estos escenarios entre instancias del prototipo de manera muy sencilla. Este es un punto muy importante, ya que el equipo se comunicó con los creadores de CyTrONE y en el intercambio mantenido se mostró interés en que los grupos de trabajo compartan los escenarios implementados.

Por otra parte, se desarrolló una regla para Elastic Security que permite detectar las acciones claves que realiza el estudiante y que implican la resolución correcta de la práctica. Esta regla permite entonces generar una alerta que implica que el estudiante ha cumplido con los pasos necesarios para completar el escenario. Los pasos que debe seguir el estudiante para completar la práctica son dos. En primer lugar debe descargar en la máquina víctima

un *script* escrito en Perl que se utilizará posteriormente para generar la *shell* reversa. Por lo tanto, el evento que se debe detectar es la conexión generada entre la máquina víctima y la máquina atacante, y la creación en el *filesystem* de la máquina víctima (en el directorio /tmp) del *script* descargado en dicha conexión. El segundo paso que realiza el estudiante es la ejecución del *script*. Para esto, se debe detectar la ejecución de dicho *script* y la generación de una conexión desde la máquina víctima hacia la máquina atacante. Se destaca que en ambos pasos, el estudiante explota la vulnerabilidad de `awstats` que permite la ejecución remota de código.

Para detectar estos eventos, se mapean las acciones del estudiante a una regla de Elastic Security de tipo Event Query Language (EQL). Esta regla permite detectar la ocurrencia de los eventos, en el orden específico, y generar una alerta. En la figura 5.2 se puede observar la alerta generada por el SIEM.

Si bien esto automatiza en gran medida el proceso de evaluación, se requiere la intervención del educador, ya que este debe evaluar a partir de las alertas generadas y los datos recolectados del trabajo del estudiante que este haya completado efectivamente la práctica, porque podría tratarse de un caso de copia. Para detectar esto se podría, por ejemplo, analizar el tiempo transcurrido entre las distintas acciones claves que llevan al estudiante a completar la práctica.

Elastic Security no solamente permite obtener datos sobre las acciones del estudiante, sino que también datos del estado de los distintos sistemas de una instancia del escenario y la propia plataforma. Por lo tanto, esto permite a los usuario educadores y administradores conocer el uso de recursos que hace cada instancia del escenario desplegado. Estos datos pueden ser muy útiles para ajustar la asignación de recursos y mejorar así la experiencia del usuario estudiante al interactuar con la plataforma. En la figura 5.3 se presenta la vista a la que tiene acceso el educador para consultar el estado de la plataforma.

Por otra parte, en la figura 5.4 se puede observar la vista para el usuario educador cuando interactúa con la extensión implementada, `instance_management`, para conocer el estado del escenario instanciado.

Para concluir con la evaluación del prototipo, se propone verificar el cumplimiento de los requerimientos identificados en la fase de análisis. En la tabla 5.1, se listan dichos requerimientos y se indica si el prototipo cumple con estos o no. Con los colores rojo, amarillo y verde se identifican a los requerimientos con prioridad alta, media y baja respectivamente.

| Requerimiento | Verifica |
|---|---|
| RF01 - Creación de escenarios | Si |
| RF02 - Instanciación de escenarios | Si |
| RF03 - Control de instancias de escenarios | Si |
| RF04 - Acceso a instancias de escenario | Si |
| RF05 - Monitoreo de la plataforma | Si |
| RF06 - Gestión de usuarios | No |
| RF07 - Autenticación delegada | No |
| RF08 - Progresión de escenario | No |
| RF09 - Seguimiento de las instancias de escenarios | Si |
| RF10 - Apagado automático del escenario | No |
| RF11 - Control de instancias de escenarios por estudiantes | No |

| | |
|---|---|
| RNF01 - Aislamiento | Si |
| RNF02 - Realista | Si |
| RNF03 - Interfaz de Consola | Si |
| RNF04 - Acceso remoto | Si |
| RNF05 - Usabilidad | Si |
| RNF06 - Facilidad para la especificación de escenarios | Si |
| RNF07 - Escalable | Si |
| RNF08 - Performance | NA |
| RNF09 - Mantenible | Si |
| RNF10 - Despliegue | Si |
| RNF11 - Escenarios portables | Si |
| RNF12 - Configurable | Si |
| RNF13 - Interfaz Grafica | No |

**Tabla 5.1:** Evaluación de prototipo según los requerimientos identificados. Los requerimientos en rojo tienen prioridad alta, en amarillo tienen prioridad media y en verde tienen prioridad baja.

| Servidor | CPU | Memoria | Disco |
|---|---|---|---|
| Maestro | 4 (2.00GHz) | 15 GB | 196 GB |
| Esclavo | 4 (2.00GHz) | 4 GB | 67 GB |

**Tabla 5.2:** Características de *hardware* de los servidores utilizados

Se puede observar que la totalidad de los requerimientos con una alta prioridad son cubiertos por el prototipo. Cabe destacar que la gran mayoría de los requerimientos catalogados con prioridad alta ya eran satisfechos por el componente CyRIS. Sin embargo, para cumplir con el requerimiento RF03 se debió desarrollar la extensión denominada `instance_management`. Por otra parte, se cumple con un 57.1 % de los requerimientos de prioridad media y 42.8 % de los de baja prioridad. En particular, el agregado de la herramienta Elastic Security permite el cumplimiento de los requerimientos RF05 y RF09 catalogados con una prioridad media y baja respectivamente.

Por último, se quiere destacar que en la prueba realizada debido a las restricciones de *hardware* (tabla 5.2), solamente se pudieron desplegar 10 instancias del escenario de forma que la usabilidad del usuario estudiante no se vea afectada. Si bien se realizaron intentos de desplegar un número mayor de instancias, la experiencia resultó mala para un usuario estudiante. Se llegó a la conclusión que el cuello de botella es la memoria disponible en los servidores. Por esta razón, al requerimiento RNF08 se lo marcó como NA en la tabla, ya que no se pudo medir de forma fidedigna para ser comparado con la solución actual. Sin embargo, dado que tanto el LaSI como el prototipo utilizan las mismas tecnologías de virtualización se cree que con los recursos suficientes de *hardware* se puede llegar a desplegar el mismo número de instancias en ambas plataformas, teniendo en cuenta que el prototipo va a requerir de más recursos por el uso de la tecnología Elastic Security.

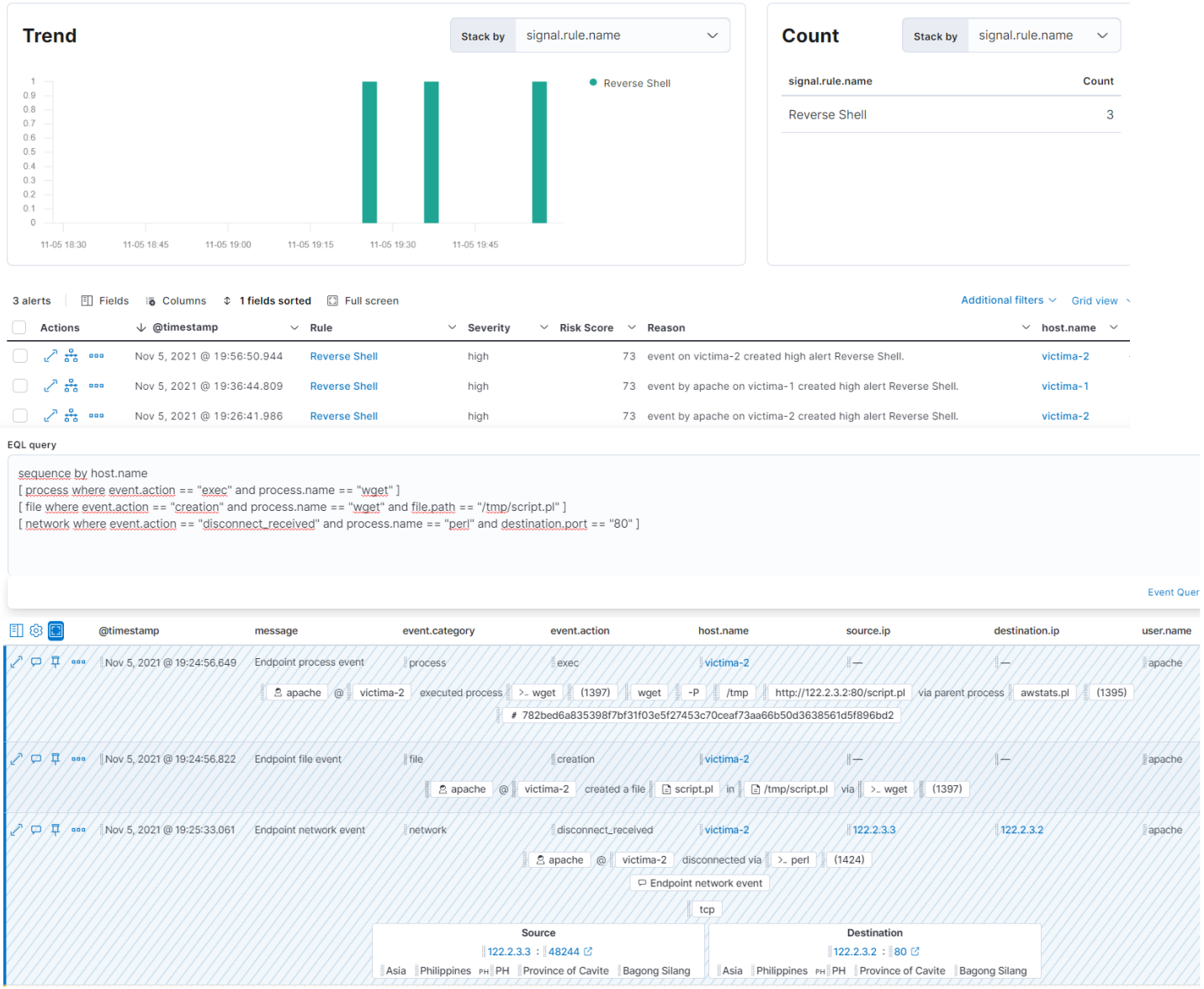


**Figura 5.2:** Vista de análisis de acciones realizadas por el estudiante dentro del escenario

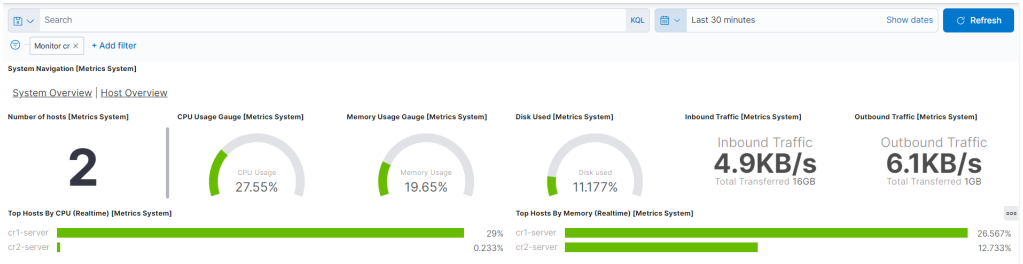


**Figura 5.3:** Vista de consumo de recursos de la plataforma

```
cyuser@cr1-server:~/cyris$ ./main/cyris.py list 122 1-3,8-9 CONFIG
################################################################################
CyRIS v1.2: Cyber Range Instantiation System
################################################################################
* INFO: cyris: Parse the configuration file.
* INFO: cyris: Check that prerequisite conditions are met.

Host                       | Instance   | Guest name     | Guest number   | Status
-------------------------------------------------------------------------------------
cr1-server.prototipo.com.uy| 1          | attacker       | 1              | running

cr1-server.prototipo.com.uy| 1          | victim         | 1              | running

cr1-server.prototipo.com.uy| 2          | attacker       | 1              | running

cr1-server.prototipo.com.uy| 2          | victim         | 1              | running

cr1-server.prototipo.com.uy| 3          | attacker       | 1              | running

cr1-server.prototipo.com.uy| 3          | victim         | 1              | running

cr2-server.prototipo.com.uy| 8          | attacker       | 1              | running

cr2-server.prototipo.com.uy| 8          | victim         | 1              | running

cr2-server.prototipo.com.uy| 9          | attacker       | 1              | shut off

cr2-server.prototipo.com.uy| 9          | victim         | 1              | shut off
```

**Figura 5.4:** Vista generada por `instance_management` sobre estado del escenario

# Capítulo 6

# Conclusiones

La educación en ciberseguridad está cobrando creciente importancia en el mundo, y con ella crece en popularidad el concepto de *cyber range*. En el relevamiento del estado del arte sobre herramientas para la enseñanza en ciberseguridad se observó que muchas universidades, empresas y gobiernos hacen uso de infraestructuras de *cyber range* para conducir entrenamientos y pruebas. Sin embargo, a pesar de la proliferación de estas instalaciones, es difícil conseguir material técnico sobre la arquitectura y diseño de las herramientas, debido a que la mayoría son propietarias o privadas.

Del subconjunto de herramientas a las cuales se tiene acceso al material técnico, se puede observar una variedad de implementaciones y decisiones de diseño bastante amplia. Por ejemplo, se pueden variar las tecnologías utilizadas, las funcionalidades y flexibilidad ofrecidos, la metodología de evaluación y más. Pero, a pesar de esta gran variedad, la mayoría de las herramientas analizadas se ajustan a la descripción general de un *cyber range* dada en la sección 2.1.2, compartiendo todas el objetivo común de brindar un ambiente seguro y legal para llevar a cabo entrenamientos prácticos de ciberseguridad. A su vez, todas estas distintas soluciones sirvieron de inspiración para proponer un nuevo diseño del laboratorio de seguridad del GSI, el cual intenta combinar las mejores cualidades de las implementaciones de *cyber ranges* vistas.

Por otro lado, una desventaja importante de las herramientas analizadas es la incapacidad de poder compartir escenarios de práctica entre instituciones educativas. En un mundo ideal, existiría un lenguaje unificado de especificación de escenarios, el cual se podría desplegar en cualquier herramienta independiente de su implementación. Si bien en este proyecto se analizaron diversos

lenguajes independientes de especificación de escenarios (ver capítulo 2), como trabajo a futuro se podría generar un estándar que unifique estos lenguajes y facilite compartir escenarios entre *cyber ranges*.

Se destaca el inmenso peso que tienen las tecnologías de virtualización en las implementación de un *cyber range* funcional y eficiente. Al proveer la capacidad de desplegar múltiples escenarios virtuales aislados sobre una misma infraestructura de *hardware*, la virtualización permite llevar a cabo prácticas realistas, evitando grandes costos de un despliegue físico y aprovechando al máximo el *hardware* disponible.

En cuanto a tecnologías de virtualización, también existe una gran variedad de implementaciones y soluciones, cada una con ventajas y desventajas claramente definidas. De todas las soluciones analizadas, KVM es la tecnología que mejor se adapta a los requerimientos de aislamiento, realismo, eficiencia, flexibilidad y utilidad de un *cyber range*. También se destaca Libvirt, el cual provee una interfaz sencilla y potente para desplegar y gestionar *guests* utilizando distintas tecnologías de virtualización; en particular, KVM. Estas tecnologías tienen la ventaja agregada de ser *open source* y gratuitas, lo cual es importante en un contexto académico donde posiblemente no abunden los recursos.

La migración de la práctica del curso FSI al prototipo de laboratorio evidencia la usabilidad y potencial del mismo. Con una sencilla especificación de escenario se pueden desplegar, de forma distribuida en dos servidores, numerosas instancias del escenario, las cuales se pueden gestionar y controlar fácilmente a través de una interfaz. A su vez, con la integración de Elastic Security se puede monitorear toda la plataforma y los escenarios desplegados y detectar automáticamente cuando los estudiantes completan la práctica, lo que facilita el trabajo de corrección por parte de los docentes y permite mantener un registro de las acciones de los estudiantes. Se cree que el nuevo laboratorio no solo presenta una mayor facilidad de uso que el LaSI, sino que ofrece mayor cantidad de funcionalidades y flexibilidad en la creación de escenarios, lo cual permite definir prácticas más ricas en conocimiento con un menor esfuerzo por parte del docente.

Del prototipo final se excluyeron dos módulos de CyTrONE, CyLMS y CyPROM. CyLMS fue excluido ya que el LMS utilizado por los cursos del GSI es Moodle, e integrar CyLMS con Moodle podría requerir de modificaciones en dicha plataforma, las cuales están fuera del alcance de este trabajo. CyPROM

fue excluido debido a que ciertas funcionalidades aún no se encuentran estables. Sin embargo, los desarrolladores del componente han informado que se le asignó un nivel de prioridad alto y se espera una nueva versión a comienzos del año siguiente, debido al interés que mostró el GSI en contar con el componente completamente funcional.

Se debe señalar que el prototipo tiene amplio margen de mejora. Se espera en un futuro poder expandir sus funcionalidades al agregar los componentes CyLMS para generar una integración con la plataforma Moodle de la Facultad de Ingeniería, y CyPROM para gestionar la evolución de los escenarios. Esto permitirá cumplir con los requerimientos RF06, RF07, RF08, RF11 y RF13.

También se podría generar una mayor integración entre el componente CyRIS y Elastic Security definiendo un lenguaje utilizado para especificar las acciones o evidencias que se buscan recolectar del trabajo del estudiante. Luego, esta especificación podría traducirse de forma automática en reglas que deben aplicarse en el SIEM para generar una mayor automatización de las tareas del usuario educador.

En cuanto al requerimiento RF10, se considera que para lograr su cumplimiento se deben implementar modificaciones en el código de CyTrONE. Sin embargo, dado que el LaSI es desplegado en la infraestructura propia de la Faculta de Ingeniería, y no en una infraestructura de nube en las que por lo general se debe pagar por los recursos consumidos, se cree que este requerimiento no debe ser considerado crítico, ya que no aportaría una funcionalidad importante.

Por otra parte, evaluando los casos de uso especificados, se tiene que el prototipo cumple de forma directa con los casos de uso 3, 4 y 5. Si bien cumple con la configuración de imágenes base, como esto se realiza al momento de instanciar el escenario y no como un paso aparte, no se puede afirmar que cumpla completamente con este caso de uso. Sin embargo, realizando una pequeña modificación en el componente CyRIS para que brinde la opción al educador de generar únicamente las imágenes base, se estará cumpliendo con este caso de uso.

Por último, en cuanto al caso de uso 1, ya que actualmente en la especificación del escenario no se indica la evidencia que debe ser recolectada, no se cumple completamente con este caso de uso. De forma similar ocurre con el caso de uso 7 ya que si bien se generó un grado de automatización de la corrección de los escenarios, al no darse una comparación según la evidencia indicada

en la especificación, no se cumple de forma completa con el caso de uso. Se cree que para cumplir con estos casos de uso se podría, como fue mencionado anteriormente, generar una mayor integración entre CyRIS y Elastic Security especificando la evidencia a recolectar y que esta especificación se traduzca a reglas a desplegar en el SIEM.

El resultado de aplicar todas estas mejoras al prototipo final sería una herramienta completa, robusta, fácil de utilizar, flexible y eficiente, y posicionaría al laboratorio del GSI entre uno de los mejores equipados de la región.

# Bibliografía


[1] Juan Diego Campo et al. "Framework for IT Security Learning". Facultad de Ingeniería, UDELAR. 2009.

[2] Antonio Álvarez. *CyberWiser.eu. D2.5Platform Design, Final Version.* Inf. téc. 2019. URL: https://www.cyberwiser.eu/content/d25-platform-design-final-version.

[3] *Apache CloudStack.* [Online; Accessed: 24 de abril de 2021]. URL: https://cloudstack.apache.org/.

[4] R. Beuran et al. "Integrated framework for hands-on cybersecurity training: CyTrONE". En: *ScienceDirect Computer and Security* 78 (2018), pp. 43-59.

[5] Razvan Beuran et al. "Realistic Cybersecurity Training via Scenario Progression Management". En: *IEEE European Symposium on Security and Privacy Workshops* (2019), pp. 67-76.

[6] Centro Nacional de Respuesta a Incidentes de Seguridad Informática. *Estadísticas de incidentes de Seguridad Informática 2020.* [Online; Accessed: 5 de abril de 2021]. URL: https://www.gub.uy/centro-nacional-respuesta-incidentes-seguridad-informatica/datos-y-estadisticas/estadisticas/estadistica-incidentes-seguridad-informatica-2020.

[7] Andrea Chierici et al. "A quantitative comparison between xen and kvm". En: *Journal of Physics: Conference Series.* Vol. 219. 4. 2019.

[8] G. Costa et al. "Automating the Generation of Cyber Range Virtual Scenarios with VSDL". En: *The Italian Conference on CyberSecurity* (2018).

[9] *crond-jaist/cyris: disable fw reboot on base vm and default chain rules #12.* [Online; Accessed: 17 de nobiembre de 2021]. URL: https://github.com/crond-jaist/cyris/pull/12.

[10] *crond-jaist/cyris: instance mgmt script and fixes #11*. [Online; Accessed: 17 de nobiembre de 2021]. URL: https://github.com/crond-jaist/cyris/pull/11.

[11] National Initiative for Cybersecurity Education (NICE) - Cyber Range Project Team. *The Cyber Range: A Guide. Guidance Document for the Use Cases, Features, and Types of Cyber Ranges in Cybersecurity Education, Certification and Training*. Inf. téc. 2020.

[12] *Diferencias entre los contenedores y las máquinas virtuales*. [Online; Accessed: 24 de abril de 2021]. URL: https://www.redhat.com/es/topics/containers/containers-vs-vms.

[13] Elastic. *¿Qué es el ELK Stack?* [Online; Accessed: 26 de octubre de 2021]. URL: https://www.elastic.co/es/what-is/elk-stack.

[14] Gencer Erdogan. *CyberWiser.eu. D4.4Trainging Material, Final version*. Inf. téc. 2020. URL: https://www.cyberwiser.eu/content/d44training-material-final-version.

[15] Charles David Graziano. "A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project". En: (2011).

[16] Rodrigo Gallardo y Guillermo Guerrero. "Investigación de Herramientas para la Enseñanza de Seguridad Informática". Facultad de Ingeniería, UDELAR. 2021.

[17] Rodrigo Gallardo y Guillermo Guerrero. "Investigación del Estado del Arte en Tecnologías utilizadas en *Cyber Ranges*". Facultad de Ingeniería, UDELAR. 2021.

[18] *Introduction to Terraform*. [Online; Accessed: 1 de mayo de 2021]. URL: https://www.terraform.io/intro/index.html.

[19] *Kernel Virtual Machine (KVM)*. [Online; Accessed: 24 de abril de 2021]. URL: https://www.linux-kvm.org/page/Main_Page.

[20] *KYPO Cyber Range Platform*. [Online; Accessed: 17 de abril de 2021]. URL: https://docs.crp.kypo.muni.cz/.

[21] *KYPO Cyber Range Platform: Conceptual Architecture*. [Online; Accessed: 17 de abril de 2021]. URL: https://docs.crp.kypo.muni.cz/basic-concepts/conceptual-architecture/.

[22] *KYPO Cyber Range Platform: Platform Components*. [Online; Accessed: 17 de abril de 2021]. URL: https://docs.crp.kypo.muni.cz/installation-guide/platform-components/.

[23] D. Conte de Leon et al. "ADLES: Specifying, deploying, and sharinghands-on cyber-exercises". En: *ScienceDirect: Computers and Security* 74 (2018), pp. 12-40.

[24] *Libguestfs*. [Online; Accessed: 13 de junio de 2021]. URL: https://libguestfs.org/.

[25] *Libvirt Virtualization API*. [Online; Accessed: 24 de abril de 2021]. URL: https://libvirt.org/.

[26] *Linux Containers - LXC - Introduction*. [Online; Accessed: 17 de noviembre de 2021]. URL: https://linuxcontainers.org/lxc/introduction/.

[27] *Moodle*. [Online; Accessed: 24 de Octubre de 2021]. URL: https://moodle.org/?lang=es.

[28] *OpenNebula Overview*. [Online; Accessed: 24 de abril de 2021]. URL: https://docs.opennebula.io/6.0/overview/opennebula_concepts/opennebula_overview.html.

[29] *OpenStack*. [Online; Accessed: 24 de abril de 2021]. URL: https://www.openstack.org/software/.

[30] Cyber Range Organization, Design (CROND). Japan Advanced Institute of ScienceTechnology. *CyTrONE User Guide*. Inf. téc. 2021. URL: https://github.com/crond-jaist/cytrone/releases.

[31] *Packer Documentation*. [Online; Accessed: 13 de junio de 2021]. URL: https://www.packer.io/docs.

[32] Cuong Pham et al. "CyRIS: a cyber range instantiation system for facilitating security training". En: *Proceedings of the Seventh Symposium on Information and Communication Technology*. 2016, pp. 251-258.

[33] Shaani Priyadarshini. "Features And Architecture Of The Modern Cyber Range: A Qualitative Analysis And Survey". Tesis de maestría. Newark, Delaware, USA: University of Delaware, 2018.

[34] *Puppet*. [Online; Accessed: 24 de abril de 2021]. URL: https://puppet.com/.

[35] *Razvan Beuran - Home Page*. [Online; Accessed: 17 de noviembre de 2021]. URL: http://www.jaist.ac.jp/~razvan/index.html.

[36] E. Russo et al. "Building Next Generation Cyber Ranges with CRACK". En: *The Italian Conference on CyberSecurity* (2018).

[37] Aravindh Sampathkumar. "Virtualizing Intelligent River®: A comparative study of alternative virtualization technologies". En: (2013).

[38] Steve Morgan. *Cybersecurity Talent Crunch To Create 3.5 Million Unfilled Jobs Globally By 2021*. [Online; Accessed: 5 de abril de 2021]. URL: https://cybersecurityventures.com/jobs/.

[39] L. Topham et al. "Cyber Security Teaching And Learning Laboratories: A Survey". En: *Information Security: An International Journal* 35 (2016), pp. 51-80.

[40] UDELAR. *Entorno Virtual de Aprendizaje (EVA)*. [Online; Accessed: 11 de abril de 2021]. URL: https://eva.fing.edu.uy/.

[41] E. Ukwandu et al. *A Review of Cyber-Ranges and Test-Beds: Current and Future Trends*. Inf. téc. 2020.

[42] *VNX: creación de escenarios de red virtuales distribuidos*. [Online; Accessed: 26 de abril de 2021]. URL: https://pdfslide.tips/reader/f/vnx-creacion-de-escenarios-de-red-virtuales-distribuidos.

[43] Miguel G Xavier et al. "Performance evaluation of container-based virtualization for high performance computing environments". En: *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE. 2013, pp. 233-240.

[44] *Xen Project*. [Online; Accessed: 24 de abril de 2021]. URL: https://xenproject.org/.

[45] *YAML*. [Online; Accessed: 24 de Octubre de 2021]. URL: https://yaml.org/.

[46] Kejiang Ye et al. "Performance Combinative Evaluation From Single Virtual Machine To Multiple Virtual Machine Systems." En: *International Journal of Numerical Analysis Modeling* 9.2 (2012).