eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN

INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA CYBER RANGE CTF (CAPTURE THE FLAG) CONTRA OBJETIVO ETSI OSM

Estudiante	*Jubeto López, Xabier*
Director	*Jacob Taquet, Eduardo Juan*
Directora	*Astorga Burgo, Jasone*
Departamento	*Ingeniería de Comunicaciones*
Curso académico	*2020-2021*

Bilbao, 21 septiembre 2021

PÁGINA EN BLANCO

RESUMEN TRILINGÜE

Castellano

Este proyecto consiste en diseñar e implementar una plataforma cyber-range. Sobre la cual se realiza una competición "Capture The Flag" contra objetivos de la iniciativa NFV de la ETSI. Esta plataforma trata de crear un método práctico y eficaz para la formación de hackers éticos. Las áreas que se enfatizan en esta formación son la ciberseguridad y la virtualización de servicios y equipos de la red. Adicionalmente, mediante un sistema de evaluación cada participante podrá conocer la calificación de su desempeño en esta plataforma.

English

This project consists of designing and implementing a cyber-range platform. On which a "Capture The Flag" competition is deployed against objectives of the ETSI's NFV initiative. This platform tries to create a practical and effective method for the training of ethical hackers. The areas that are worked in this training are cybersecurity and virtualization of services and network equipment. Additionally, through an evaluation system each participant will be able to know the qualification of his performance on the platform.

Euskara

Proiektu hau cyber-range plataforma bat diseinatzea eta inplementatzean datza. Plataforma honetan "Capture The Flag" lehiaketa ezartzen da ETSIren NFV ekimeneko helburuen aurka. Plataforma hau hacker etikoen heziketarako metodo praktiko eta eraginkorra sortzen saiatzen da. Prestakuntza honetan lantzen diren arloak bi dira, alde batetik, zibersegurtasuna eta, beste aldetik, zerbitzuen eta sareko ekipamenduen birtualizazioa. Gainera, Gainera, balioztatze sistema baten bidez, parte-hartzaile bakoitzak plataforman duen jardueraren kalifikazioa jakin ahal izango du.

ÍNDICE

LISTA DE TABLAS

LISTA DE ILUSTRACIONES

ACRÓNIMOS

CTF	Capture The Flag
ETSI	European Telecommunications Standards Institute
NFV	Network Function Virtualization
OSM	Open-Source Mano
MANO	Management and Orchestration
K8S	Kubernetes
NS	Network Service
VNF	Virtualized Network Function
VIM	Virtualized Infrastructure Manager
VDU	Virtual Deployment Unit
NBI	Northbound Interface

1. INTRODUCCIÓN

Las arquitecturas y tecnologías de red han ido evolucionando poco a poco, con el desarrollo de nuevas invenciones. Este progreso ha hecho que en los últimos años se expandan conceptos y tecnologías que permiten la virtualización de equipos físicos, computación en la nube, etc. Un ejemplo de todos estos avances es la iniciativa NFV de la ETSI, con la cual se trata de impulsar el desarrollo e interconexión de los distintos elementos que componen los sistemas de virtualización. Gracias a estas tecnologías se ha logrado impulsar la arquitectura de redes hacia entornos más dinámicos y versátiles.

Todo este desarrollo ha hecho que las redes se vayan volviendo cada vez más complejas. Sin embargo, la complejidad es antagónica a la seguridad. Por lo tanto, ahora los sistemas deben invertir cada vez mayores recursos en mantener la ciberseguridad. Sin embargo, ninguno de estos sistemas de seguridad es suficiente si no hay alguien dispuesto a probarlos y a mejorarlos. Ante esta necesidad surge el hacking ético.

El hacking ético es un conjunto de principios morales en los que un grupo de personas deciden empezar a atacar equipos con el objetivo exclusivo de descubrir fallos en su seguridad. Una vez descubiertos estas vulnerabilidades, los hackers éticos generan un informe y lo reportan a la organización pertinente para que solucionen dichas vulnerabilidades. Bajo ningún concepto tratan de aprovecharse de esa vulnerabilidad. Estas técnicas son usualmente las más eficaces a la hora de evitar ataques reales, ya que sirven para comprender cómo se podría realizar un ataque maligno y aprender a impedirlo. Por lo tanto, es muy importante que estos hackers éticos dispongan con una formación puntera en tecnologías y conceptos de ciberseguridad.

Con esto en mente, este proyecto trata de crear una plataforma que permita la formación de hackers éticos. Esta formación hace especial énfasis en las tecnologías que se impulsan desde la iniciativa NFV de la ETSI. Para lograr este fin, se ha desarrollado una plataforma cyber-range que simula distintos servicios y se ha diseñado una serie de ejercicios para una competición CTF. En estos ejercicios, los usuarios de esta plataforma trataran atacar sistemas de virtualización como OSM, OpenStack y Kubernetes. Adicionalmente, un sistema de puntuaciones permite conocer el desempeño de estos hackers éticos en su proceso de formación. Con todos estos elementos, este proyecto planea desarrollar e implementar una herramienta versátil y adaptable que pueda formar a múltiples personas en las materias más punteras las áreas de la arquitectura de red y la ciberseguridad.

2. CONTEXTO

Existen dos grandes conceptos que estructuran el proyecto y que necesitan ser contextualizados: las plataformas Cyber-range y la iniciativa NFV de la ETSI.

Las plataformas **Cyber-range** consisten en desarrollar una simulación completa a todos los niveles de una red. La simulación va desde los sistemas físicos de la infraestructura hasta los servicios de alto nivel que se ofrece a los usuarios. Normalmente, se suelen utilizar estas plataformas como entornos controlados de entrenamiento y aprendizaje en áreas de la ciberseguridad.

La **iniciativa NFV** (Network Functions Virtualization) de la ETSI trata de hacer evolucionar las redes mediante las tecnologías de virtualización. Su objetivo es hacer que las redes, infraestructuras y servicios puedan ser desplegadas bajo demanda en cualquier momento y en cualquier lugar, ganando en dinamismo e innovación.

2.1. Cyber-range

En los últimos años se ha ido popularizando el concepto de cyber-range como una herramienta para el aprendizaje y el ensayo de la ciberseguridad. Varias instituciones y organizaciones han definido el objetivo y el funcionamiento de este tipo de plataformas, sin embargo, la definición más extendida es la que ofrece el European Cyber Security Organisation (ECSO).

“Una cyber-range es una plataforma para el desarrollo y uso de entornos simulados. **Este entorno simulado es una representación exacta de la red**, de todos los sistemas físicos, de las aplicaciones y de las infraestructuras de una organización. Lo cual suele incluir la simulación de los ataques, de los usuarios con sus respectivas actividades en la red y los servicios tanto públicos como de terceros.” (European Cyber Security Organisation (ECSO), 2020) [12]

Es decir, es un entorno virtual que simula escenarios reales. El objetivo de estos entornos puede variar dependiendo del caso de uso, pero los principales son la realización de análisis e investigaciones de la seguridad de una red y la educación en materia de seguridad. Esto se debe a que cyber-range permite realizar ataques y comprobaciones de seguridad de una forma segura sin dañar ningún entorno real. Haciendo que sea una herramienta idónea para la formación de alumnos o empleados.

2.1.1. Características

El propio ECSO lista las características más comunes de las plataformas cyber-range. A la hora de implementar las cyber-range, no es necesario que dispongan de todas estas características. En función de los casos de usos, se implementará unas funcionalidades u otras.

En la *Ilustración 1* se muestra en forma de esquema una lista de las características definidas por el ECSO para las plataformas de cyber-range. Se pueden disitinguir dos tipos de características: las funcionalidades que ofrecen las cyber-range y las tecnologias sobre las que se sostienen.

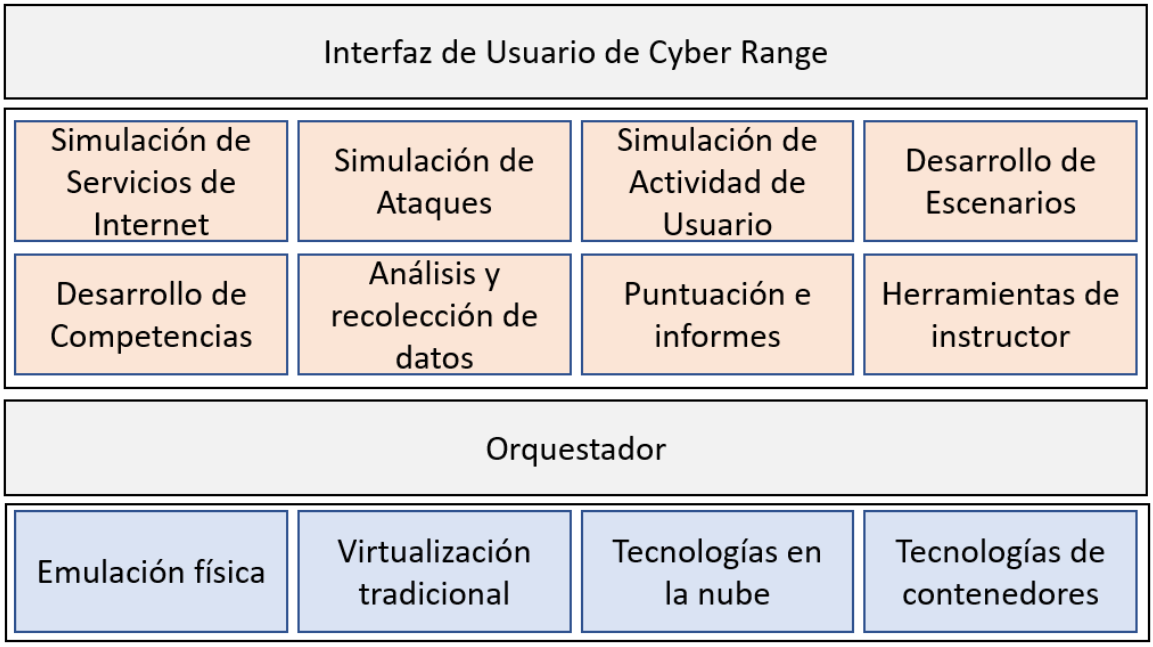

Ilustración 1: Esquema de un Cyber-range según el ECSO [12]

2.1.1.1. Funcionalidades

Las plataformas de cyber-range cuentan con varias funcionalidades que se pueden dividir en ocho grupos: simulación de servicios de internet, simulación de ataques, simulación de actividad de los usuarios, desarrollo de escenarios, desarrollo de competencias, análisis y recolección de datos, puntuación e informes y herramientas de instructor. A continuación, se describen cada uno de ellos:

- **Simulación de servicios de internet**. Consiste en añadir realismo a los escenarios mediante la simulación los servicios disponibles a través de Internet. Entre otros ejemplos, una forma de conseguir este realismo sería la simulación de proveedores de servicio, de repositorios de software o de aplicaciones más específicas, como redes sociales.
- **Simulación de ataques**. Consiste en simular ataques, vulnerabilidades y brechas de seguridad dentro del entorno controlado. Normalmente estos ataques simulados se centran en ataques de librería y en la capacidad de crear o importar ataques personalizados por los usuarios.
- **Simulación de actividad de los usuarios**. Añade una capa de realismo a la simulación de la red mediante la recreación del comportamiento de distintos usuarios. Estos usuarios ficticios pueden ser tanto internos como clientes externos que acceden al entorno simulado. En caso de ser necesario, también se incluye la simulación de usuarios móviles.
- **Desarrollo de escenarios**. Consiste en usar las oportunidades que ofrece las plataformas cyber-range para permitir el desarrollo de diferentes escenarios por terceros. De tal forma que los usuarios puedan desarrollar escenarios para que otros usuarios los utilicen.
- **Desarrollo de competencias**. Consiste en las herramientas que permiten el desarrollo de un sistema de enseñanza sobre la plataforma cyber-range. Permite que una organización pueda crear un programa de formación para desarrollar competencias de sus alumnos o sus trabajadores.
- **Análisis y recolección de datos**. Consiste en la recolección y análisis de los datos que generan los usuarios que utilizan la plataforma cyber-range. Dependiendo de los análisis que se quieran realizar, entre otros, los datos que se registran son: el tráfico generado en la red, el uso de la memoria, los sistemas o servicios a los que acceden los usuarios, etc. El análisis de estos datos permite aprender cómo los usuarios usan la plataforma cyber-range otorgando un feedback útil. El análisis puede utilizar herramientas de inteligencia artificial.

- **Puntuación e informes**. Consiste en la capacidad de puntuar a los usuarios en función de sus actividades o interacciones con la plataforma cyber-range. Suele contar con la posibilidad de monitorizar las actividades de cada usuario y de mostrar el rendimiento de cada uno. Se incluye la capacidad de realizar informes estándar o personalizados que permiten visualizar el rendimiento de los usuarios en la plataforma cyber-range.
- **Herramientas de instructor**. Consiste en una colección de distintas herramientas útiles para un instructor. Estas herramientas tienen mayoritariamente un propósito educacional o de entrenamiento. Un ejemplo de una de estas herramientas sería la imitación (mirroring) de la sesión de un usuario en el equipo del instructor.

2.1.1.2. Tecnologías

Con el objetivo de ofrecer las funcionalidades descritas previamente, las plataformas se sostienen sobre distintas tecnologías. Estas se dividen en tecnologías de orquestación y tecnologías de virtualización.

El **Orquestador** es un elemento imprescindible para cualquier plataforma cyber-range. Es el elemento que realiza la coordinación de los demás sistemas. Es responsable de la creación, configuración, modificación y eliminación de los sistemas virtualizados. Así como, de la automatización de tareas entre la infraestructura virtual y otros componentes internos o externos de cyber-range.

Dependiendo de cada caso de uso, una plataforma cyber-range puede implementarse sobre una o más tecnologías de virtualización. Estas tecnologías de virtualización se pueden distinguir en: emulación de sistemas físicos, virtualización tradicional, tecnologías en la nube y tecnologías de contenedores. A continuación, se describen cada uno de ellos:

- **Emulación de sistemas físicos**. Esta tecnología se encarga de simular el funcionamiento de un sistema hardware de la forma más precisa posible. Suele ser muy poco escalable y el servicio que se despliega sobre él suele contar con las limitaciones típicas del sistema emulado.
- **Virtualización tradicional**. Estas máquinas virtuales en un software que simula el comportamiento de uno o varios sistemas operativos sobre un único equipo físico. Como se pueden desplegar varias máquinas virtuales sobre un único equipo físico, se necesita la figura del hipervisor. El cual, asegura que las máquinas virtuales no interfieren las unas con las otras y que cada una pueda acceder a los recursos físicos que necesite.
- **Tecnologías en la nube**. Esta tecnología abstrae los recursos físicos de toda una infraestructura (almacenamiento, procesamiento, conectividad, etc.) y los pone disponibles de forma transparente para que el hipervisor pueda usar todos los recursos por completo. Ofrecen dinamismo, eficiencia y escalabilidad.
- **Tecnologías de contenedores**. Esta tecnología encapsula en contenedores los distintos servicios desplegados. Cada contenedor cuenta con los recursos y dependencias imprescindibles para que los servicios encapsulados trabajen con total normalidad, como si estuvieran desplegados en un equipo físico tradicional. Este tipo de tecnologías se describen en más detalles en el punto *2.2.4 Kubernetes*

2.1.2. CTF: Capture The Flag

Por un lado, como ya se ha mencionado, las plataformas cyber-range ofrecen un método práctico de aprendizaje de ciberseguridad. Al mismo tiempo, uno de los métodos prácticos más populares para el aprendizaje de ciberseguridad son las competiciones *CTF (Capture The Flag)*. Estos dos conceptos son compatibles, lo cual, hace que se puedan unir para ofrecer una plataforma de aprendizaje más completa. Uniendo estos conceptos logramos tener una infraestructura que permite la formación en entornos controlados con competiciones que ya han demostrado ser eficientes y divertidos a la hora de enseñar ciberseguridad.

Las competiciones *CTF* son un juego para el desarrollo de competencias en ciberseguridad. Suelen estar considerados como una de las mejores formas de aprender sobre ciberseguridad. Ya que permite a los jugadores desarrollar competencias en temáticas distintas de la ciberseguridad de forma totalmente práctica.

Las competiciones *CTF* consisten en realizar distintos ataques en un entorno controlado con el objetivo de obtener una bandera (también llamada *Flag*) que confirme que se ha superado el desafío con éxito. Esta bandera suele ser una cadena de caracteres (*Strings*) que se ha ocultado en un programa, servicio o equipo. El objetivo de los jugadores es atacar dichos elementos de la forma que consideren oportuna para obtener y validar la bandera.

Normalmente, estas competiciones son desafíos que duran un fin de semana y que se superan por equipos. Existen una gran cantidad de organizaciones alrededor del mundo que desarrollan estas competiciones. Cada una tiene distintas características y pueden ser regionales o internacionales, online o en algún lugar geográfico, etc. Aun así, si el jugador lo prefiere, también existen desafíos online individuales y sin límite de tiempo.

Las competiciones *CTF* se suelen dividir en dos tipos principales de competición: Jeopardy y Ataque-Defensa. Los tipos de ataques o retos que se utilizan en cada competición también suele estar diferenciados. El propósito de esta clasificación es permitir al usuario elegir el tipo de competición y el tipo de ataque que más le guste.

2.1.2.1. Jeopardy

Este tipo de competiciones consisten en superar retos de distintas temáticas y obtener puntos en función de la dificultad del reto superado. Normalmente a mayor dificultad, mayor cantidad de puntos. El reto más común consiste en explotar un tipo concreto de servidor, para obtener acceso remoto y obtener un fichero que contiene la bandera, lo que prueba que has atacado el sistema. Después, se introduce la bandera en un formulario y, tras validarla, obtienes puntos en función de la dificultad de reto.

2.1.2.2. Ataque-Defensa

Este tipo de ataque enfrenta a dos equipos. Los equipos deben atacar al otro equipo para ganar el control de una bandera mientras que defiende su propia bandera de los ataques del equipo contrario. Todos los equipos se encargan simultáneamente del ataque y de la defensa de las banderas. En estas competiciones la bandera puede ser un fichero, una cadena de texto o un equipo.

2.1.2.3. Tipos de retos

No existe una clasificación oficial y cada uno de los organizadores de las competiciones puede diferenciar los ataques como desee. Sin embargo, normalmente se suelen agrupar en estos siete tipos: Ingeniería inversa, pwn, criptografía, web, forense y miscelánea. Los cuales consisten en:

- **Ingeniería inversa.** Estos ataques suelen incluir un programa que se ejecuta localmente. El programa ejecuta un algoritmo que comprueba una clave de entrada, si se introduce la clave apropiada, el programa devuelve la bandera. Resolver estos retos requiere hacer ingeniería inversa y entender el algoritmo implementado para deducir la clave de entrada correcta.
- **Pwn**. Estos ataques consisten en ganar el control sobre un equipo externo. El planteamiento del reto suele incluir un programa, una dirección IP y un puerto donde se está ejecutando ese mismo programa. El objetivo del ataque es que jugador deduzca cómo funciona el programa y lo manipule de forma local. Una vez hecho esto, el usuario debe ganar acceso remoto al servidor manipulando el programa del servidor y leer el archivo con la bandera.

- **Criptografía**. Este tipo de ataques trata de romper con mecanismos de criptografía para obtener la bandera. Es de los tipos de juegos que más varía en dificultad. Pueden necesitar simple lógica para romper codificadores simples o técnicas complejas en la que es necesario estar a la última con publicaciones e investigaciones en el área de la criptografía y las matemáticas.
- **Web**: Este tipo de ataques consisten en atacar aplicaciones web. Suelen ser las más sencillas para los novatos, aunque también pueden suponer un desafío para los jugadores más veteranos. En estos ataques los jugadores disponen de una URL y de una directriz sobre dónde buscar la bandera.
- **Forense**. En este tipo de ataques el jugador deberá analizar e investigar un cierto tipo de datos. Los cuales pueden ser: imágenes, registros en memoria, logs, etc. El jugador debe usar distintas herramientas para procesar esos datos y buscar la bandera escondida.
- **Miscelánea.** Este tipo de ataques son los que no pueden ser clasificados dentro de ninguno de los anteriores. Puede que sea porque sea un tipo de ataque nuevo diseñado por la organización de la competición, porque sea una combinación de otros ataques, porque sea un tipo de ataque poco usado en este tipo de competiciones, etc. Suelen consistir en implementaciones ingeniosas para resolver algún problema.

2.2. Iniciativa NFV: Network Functions Virtualization.

A medida que se despliega más infraestructura de red, más costoso se vuelve la innovación, la mejora o el mantenimiento de las redes. Lo cual hace que sea totalmente inviable implementar nuevas tecnologías ingeniosas que rompan con las tendencias establecidas y haga evolucionar el sector de las telecomunicaciones. Bajo este paradigma, una tecnología innovadora y revolucionaria no tiene cabida si para implementarla hay que desechar todas las tecnologías previas, o si implementándola deja obsoletas tecnologías que aún no han sido amortizadas.

Con el propósito de hacer frente a estos inconvenientes, en 2012, surgen iniciativas como la de NFV *(Network Functions Virtualization)* de la ETSI. El objetivo de esta iniciativa es el de desarrollar un nuevo tipo de tecnología que permite virtualizar las funciones de red sobre equipos genéricos. Tiene el propósito de evitar los equipos muy específicos y costosos que constituyen la red en la arquitectura tradicional (routers, switches, cortafuegos, etc.). Este tipo de iniciativas tratan de hacer evolucionar el sector de las telecomunicaciones mediante el abaratamiento del despliegue de nuevas tecnologías e infraestructuras, pasando de equipos físicos a equipos virtuales. Varios años y cientos de publicaciones después, esta iniciativa está siendo consolidada en distintos sectores.

De iniciativas de este tipo surgen soluciones de código abierto como OSM, OpenStack y Kubernetes. Las cuales tienen como objetivo tomar los estándares e implementarlos en un software que permitan a sus usuarios gestionar, orquestar y mantener redes virtuales con todo tipo de funciones de red virtualizadas. Soluciones que sirven para que los administradores de red puedan crear sus propias redes virtuales y ofrecer los mismos servicios que con una red física convencional, pero con un dinamismo que satisfaga las necesidades de los usuarios en cualquier momento y a un coste mucho menor. Gracias a estas tecnologías, surgen conceptos innovadores como: Infraestructura bajo demanda *(IaaS)*, Redes bajo demanda *(NaaS)*, Conectividad o Telecomunicaciones bajo demanda *(TaaS)*, etc.

En las siguientes secciones se analizará la propuesta de la ETSI y los conceptos presentados previamente. Así como varios elementos de virtualización que, de una forma u otra, implementan la iniciativa NFV de la ETSI. Para una descripción más detallada de estos elementos y de su arquitectura, ver *14. ANEXO I: DESCRIPCIÓN DE LAS TECNOLOGÍAS*

2.2.1. Características de NFV: Network Functions Virtualization

Como la organización europea de estándares, el objetivo de la ETSI es la de estandarizar y consolidar la interoperabilidad de diversas tecnologías. Por este motivo, para las tecnologías de virtualización de infraestructura y servicios de red, ha fomentado la iniciativa NFV (Network Functions Virtualization). Con esta iniciativa trata de agrupar las distintas tecnologías de este tipo bajo unas mismas especificaciones, haciendo que sean compatibles entre sí y facilitando un desarrollo más organizado.

La iniciativa NFV de la ETSI consiste en describir unas directrices para una arquitectura que facilite la virtualización de los elementos de la red. Después, cada organización que trate de implementar esta tecnología en su solución es libre de definir más a fondo estos conceptos. Estas directrices tratan sobre desplegar varias funciones de red como entidades de software independientes, denominadas Virtualised Network Functions *(VNF)*, sobre la infraestructura de NFV *(NFVI).* En la *Ilustración 2* se muestra la arquitectura de alto nivel de NFV tal y cómo se representa en la iniciativa de la ETSI.

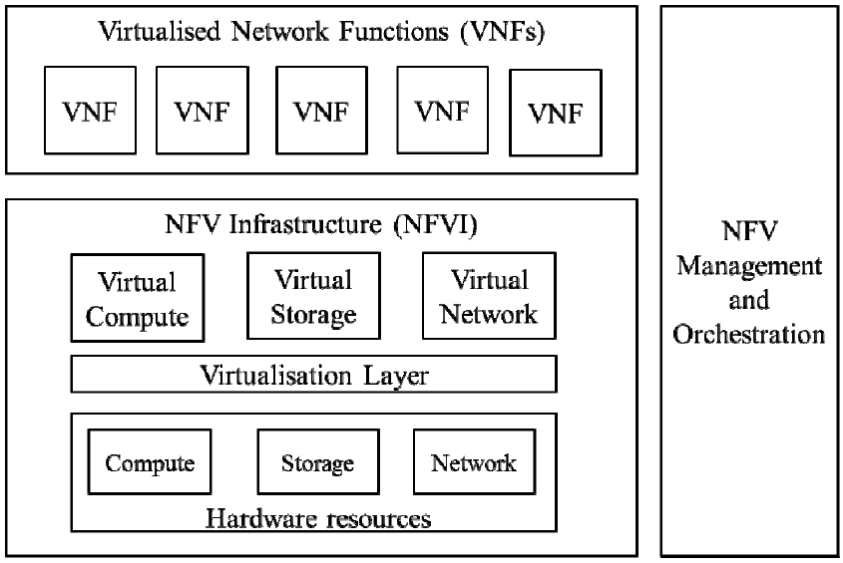

Ilustración 2: Arquitectura de alto nivel de NFV [8] (ETSI, 2014)

Tal y como se puede apreciar en la arquitectura de alto nivel de NFV que se muestra en la Ilustración 2, la arquitectura está compuesta por tres módulos principales, los cuales son: VNFs, NFVI y NFV-Mano. De manera resumida, cada uno de estos elementos se pueden describir como:

- Las **VNFs** (*Virtualized Network Function*) consisten en un grupo de funciones de red virtualizadas. Estas funciones virtualizadas proporcionan un servicio al usuario final. Sus características varían y se adaptan a las necesidades de cada usuario.
- La **NFVI** (*Network Function Virtualization Infrastructure*) es la infraestructura que convierte los recursos hardware en recursos virtuales que puedan usar las VNFs. Lo hace de manera agnóstica tanto para las VNFs como para el usuario final.

- El **NFV-Mano** *(NFV Management and Orchestration)* es un módulo que controla y gestiona los demás módulos. Su trabajo es imprescindible a la hora de sincronizar todos los sistemas que constituyen la arquitectura NFV. Se divide en tres elementos: *Orquestador, VNF Manager y VIM (Virtual Infrastructure Manager)*. El *orquestador* dirige todos los elementos de la arquitectura, el *VNF Manager* se encarga de administrar las funciones de red virtualizadas y el *VIM* administra la infraestructura necesaria para sostener todas las VNFs.

En el apartado *14.1 NFV: Network Functions Virtualization* del *ANEXO I: DESCRIPCIÓN DE LAS TECNOLOGÍAS* se profundiza más al detalle en cada uno de estos módulos que componen la iniciativa NFV de la ETSI. Mostrando un diagrama más detallado de la arquitectura NFV de la ETSI y una descripción más completa de todos los elementos que componen la iniciativa.

2.2.2. OSM: Open Source Mano

OSM es una iniciativa de ETSI para el desarrollo de un software de gestión y orquestación *(Management and Orchestration, MANO)* de código abierto para NFV. Es capaz de ejecutar modelos de código abierto que sean válidos para todas las VNF e independientes de los demás niveles de la arquitectura NFV. Dentro de la arquitectura de NFV, OSM se encarga en exclusiva del módulo NFV-Mano. Para el resto de elementos de la infraestructura NFV es necesario usar otro tipo de software. En la *Ilustración 3* se muestra el lugar que ocupa OSM dentro de la arquitectura NFV.

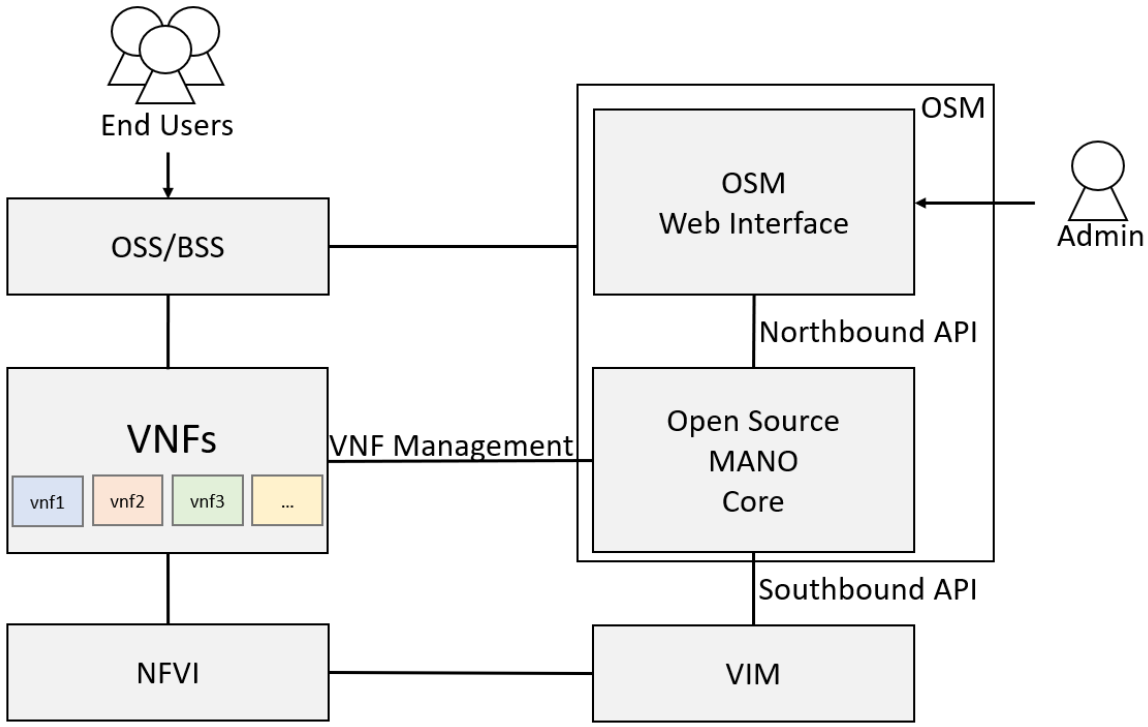

Ilustración 3: Diagrama de OSM dentro de la arquitectura NFV

El objetivo de OSM es el desarrollo de un orquestador de servicios de red de calidad y dirigido por la comunidad. OSM es capaz de dar soporte a una gran cantidad de sistemas virtuales: *NFVI, VIM, WIM, VNF, Network Services y Network Slices*. Al mismo tiempo, también gestiona y orquesta los ciclos de vida, la configuración y demás elementos de las funciones virtualizadas. En los siguientes apartados y en los anexos se detallan en mayor profundidad estos conceptos.

2.2.2.1. Conceptos clave de OSM

La iniciativa NFV de la ETSI es un proyecto complejo de implementar y de gestionar. En OSM, esta complejidad está planteada para ser manejada por una lógica de capas, cada una representando una plataforma de servicios. Estos servicios, llamados *objetos de servicio,* son creados a demanda y tienen un ciclo de vida. Las plataformas de nivel superior pueden apoyarse en otras de nivel inferior para poder ofrecer un objeto de servicio más complejo. Se pueden diferenciar dos tipos de plataformas en función de su posición en la infraestructura:

- La **Management Function** consume las API de otras plataformas de nivel inferior con el propósito de crear nuevos objetos de servicio, controlar sus ciclos de vida e implementarlos bajo demanda. En general, es el corazón de la plataforma.
- Las **Managed Functions** son las plataformas que se encuentran en niveles inferiores. Tienen una funcionalidad única *(standalone)*. Son las que ofrecen sus servicios a través de una API al *Management Function*.

Aparte de distinguir OSM por niveles, también se pueden diferenciar dos tipos de modos de funcionamiento. Los cuales, aun siendo independientes, coexisten en todo momento dentro del sistema.

- El **Service Platform view** se centra en las funcionalidades propias de la arquitectura y del sistema. Se encarga de suministrar los servicios de la red a través de *objetos de servicio*. Trabaja con peticiones frecuentes y dinámicas, altamente automatizadas y coordinadas. Son difíciles de controlar sin una buena estructura de control.
- El **Platform Operation view** se centra en las operaciones O&M *(Operation & Maintenance)* de las *Management y Managed Functions*. Se encarga de inicializar, ejecutar y mantener cada una de estas funciones. En contraste con la anterior, las peticiones de esta son más esporádicas y más estáticas.

En cuanto a los *objetos de servicio* de más alto nivel que produce, OSM proporciona la capacidad de desarrollar redes bajo demanda *(NaaS)* para su directa explotación por el proveedor de servicios o para su comercialización a terceros. Hay dos tipos de *NaaS* que OSM puede proporcionar: *Network Service (NS) y Network Slice (NetSlice).*

- Las **Network Service (NS)** son una agrupación de varias VNFs. De esta forma, OSM es capaz de otorgar un único objeto de servicio complejo compuesto por múltiples funciones de red virtualizadas que se adaptan a las necesidades de los usuarios. El proceso de creación, ejecución y eliminación está definido en el ciclo de vida de las NS. Este ciclo de vida está dividido en varias fases: las NS primero son definidas mediante descriptores yaml, luego son incorporadas en paquetes de NS, después son instanciadas, a continuación, se ejecuta la NS junto con las primitivas de día 1 y 2, por último, se finaliza la NS.
- Las **Network Slice (NetSlice)** son un concepto introducido por OSM, que busca expandir las características de la iniciativa NFV. A efectos prácticos se puede entender como una particularización de las NS.

Por último, en cuanto a los servicios consumidos, OSM deja que otras soluciones se encarguen de desplegar tanto las VNF como la infraestructura (NFVI), mientras que él se encarga de la administración y orquestación. En este sentido, según la arquitectura de la iniciativa NFV, el NFV-Mano se divide en tres elementos: *Orquestador, VNF Manager y VIM (Virtual Infrastructure Manager)*. Los dos primeros elementos forman parte de la arquitectura de OSM, sin embargo, para las tareas del VIM se apoya en software de terceros. Esto se debe a que, al no proporcionar la infraestructura necesaria, tampoco se encarga de coordinarla.

2.2.3. *OpenStack*

OpenStack es un software de código abierto de procesamiento de VNFs en la nube. La mayor cualidad de OpenStack es su gran facilidad para desplegar instancias de servicios virtuales bajo demanda. Se encarga de convertir los recursos de hardware en recursos virtuales que pueden usar las VNFs. Estos recursos son, entre otros, imágenes para instanciar en las máquinas virtuales, interfaces de red, volúmenes de almacenamiento persistente, etc.

OpenStack es un referente en cuanto a los sistemas de Infraestructura bajo demanda *(IaaS)*. Debido a esto, es de los softwares más populares para la virtualización de servicios en la nube y cuenta con una de las comunidades de desarrolladores más activas del mundo.

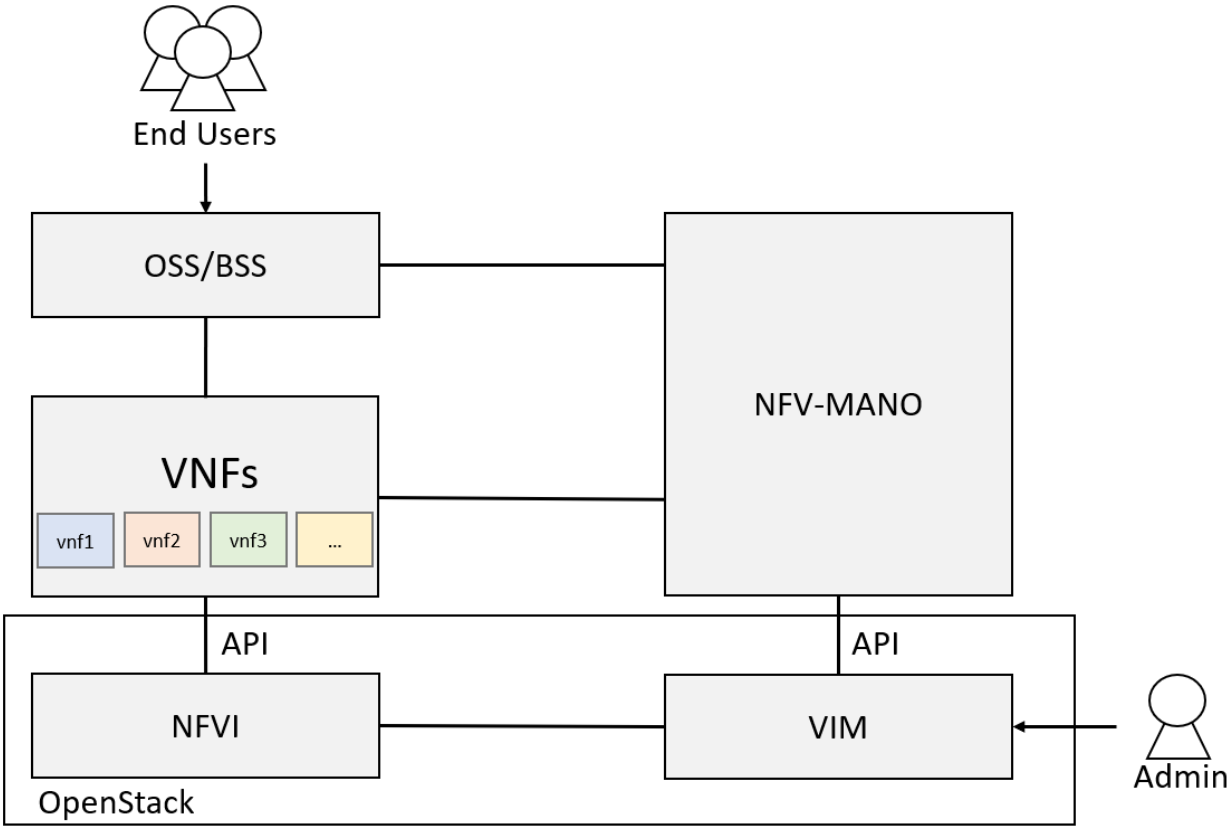

Ilustración 4: Diagrama OpenStack dentro de la arquitectura NFV

Tal y como se puede ver en la *Ilustración 4,* dentro del paradigma desarrollado por la iniciativa NFV de la ETSI, realiza las funciones de un (VIM) *Virtual Infrastructure Management* y de la *NFVI (NVF Infrastructure)*. Es decir, despliega, gestiona y elimina las VNFs que componen los *Network Services* de OSM. Siguiendo siempre las fases del ciclo de vida de cada VNF.

OpenStack dispone de una arquitectura compleja que le concede una gran adaptabilidad ante las peticiones de los usuarios. Esta arquitectura formada por multitud de elementos interconectados se describe con mayor detalle en el punto *14.3: OpenStack* del *ANEXO I: DESCRIPCIÓN DE LAS TECNOLOGÍAS.*

2.2.4. Kubernetes

Kubernetes es una plataforma de contenedores y de microservicios en la nube de código abierto. Orquesta la red, el almacenamiento y la capacidad de cómputo de contenedores de forma automática. De esta forma, permite descargar el trabajo de los usuarios y hace que sea mucho más sencillo implementar plataformas como servicio *(PaaS)* e infraestructura como servicio *(IaaS)* según las necesidades de los usuarios.

Kubernetes utiliza contenedores en vez de máquinas virtuales, lo cual, en comparación, le permite realizar implementaciones a gran escala con un coste de recursos mucho más reducido. De esta forma, se basa en contenedores pequeños con una funcionalidad concreta en vez de en grandes máquinas virtuales. Es decir, se centra en el uso de microservicios frente a las aplicaciones monolíticas tradicionales.

En este contexto, se define una **aplicación monolítica** como un sistema que combina en un único software todos sus servicios, funciones, librerías, etc. De tal forma que todos los componentes se desarrollan y despliegan simultáneamente como una única entidad. Se trata de una estructura pesada de elementos relacionados y centralizados. Por ese mismo motivo, el proceso de creación de nuevos servicios o de actualización de los existentes tiene, en comparación, una mayor dificultad, ya que se debe actualizar toda la aplicación en su conjunto, con sus correspondientes dependencias. Estas aplicaciones monolíticas son las que se han estado implementando tradicionalmente en la industria cuando se desplegaba cualquier software en una máquina virtual o física.

Bajo el mismo contexto, se define una **aplicación de microservicios** como un sistema que descompone y separa una aplicación monolítica en pequeños componentes, los cuales pasan a ser una aplicación totalmente independiente. Estas aplicaciones se sostienen sobre contenedores que virtualizan el nivel del sistema operativo, lo que les permite ser completamente agnósticos a la infraestructura sobre la que se despliegan y, por lo tanto, es posible implementarlos en cualquier equipo. Adicionalmente, los componentes de la aplicación de microservicios se encuentran completamente aislados entre sí, con su propio sistema de archivos, recursos y procesos. Este aislamiento permite que las actualizaciones puedan ser dirigidas a ese componente en exclusiva sin alterar el resto de la red, permitiendo una escalabilidad rápida, sencilla y que se ajuste a las necesidades de cada situación. Las aplicaciones de microservicios son la nueva tendencia en el diseño de redes desplegadas sobre tecnologías de contenedores.

En la *Ilustración 5* se puede observar un esquema de la arquitectura de una aplicación monolítica y de una aplicación de microservicios. Kubernetes, al usar una aplicación de microservicios, consigue realizar las mismas funciones que una aplicación monolítica, pero a un coste computacional significativamente menor. Al mismo tiempo, consigue proveer escalabilidad dinámica en función del estado de la red.

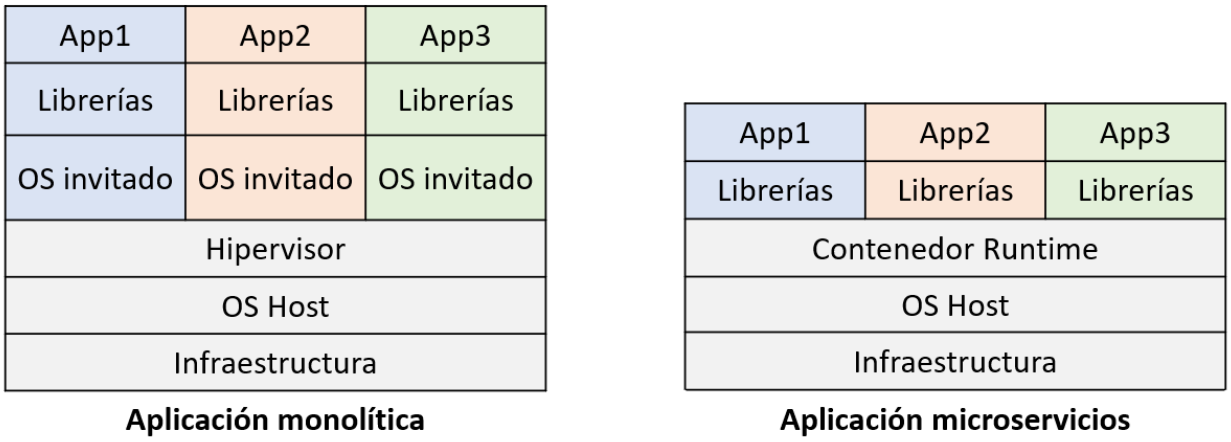

Ilustración 5: Comparativa aplicación monolítica y aplicación de microservicios

Internamente, Kubernetes se organiza en **clústeres**. Los cuales son una agrupación de **nodos**. Estos nodos funcionan con una arquitectura maestro-esclavo. Los maestros se encargan de administrar el *clúster* mientras que los esclavos son los encargados de desplegar y mantener los servicios que ha solicitado el cliente. A más bajo nivel, cada nodo esclavo está formado por *pods*. Los **pods** son la unidad más indivisible de los microservicios y son los encargados de realizar el cómputo de los servicios que ofrece Kubernetes. Para una descripción más detallada de los elementos de la arquitectura ver *14.4 Kubernetes* en *ANEXO I: DESCRIPCIÓN DE LAS TECNOLOGÍAS*.

3. OBJETIVOS Y ALCANCE DEL TRABAJO

El objetivo principal de este trabajo consiste en diseñar e implementar una plataforma cyber-range, sobre la que se despliega una serie de ejercicios de CTF *(Capture The Flag*), los cuales consisten en atacar objetivos que cumplen con la iniciativa NFV de la ETSI. La intención de esta plataforma es el de enseñar hacking ético de forma práctica, mediante el uso de distinto tipos de ataques contra distintos métodos de virtualización, distintas infraestructuras de red y distintas infraestructuras de cómputo.

Este objetivo principal puede ser **dividido en tres partes**. Todas tienen la misma importancia y entran dentro de los requerimientos del proyecto. Aunque se puedan diferenciar entre ellas, son parte intrínseca del proyecto y están relacionadas, por lo que no se pueden separar del objetivo principal del trabajo. Por un lado, se tiene como objetivo el diseñar e implementar una plataforma cyber-range. Por otro lado, utilizar esta plataforma cyber-range para probar y estudiar la seguridad de las soluciones basadas en la iniciativa NFV. Por último, usar la plataforma implementada para desplegar una competición CTF.

La **plataforma cyber-range** se sostiene sobre las soluciones que implementan la iniciativa NFV de la ETSI. Esta plataforma cyber-range simula una red ficticia sobre la que se despliegan los ejercicios que componen la competición CTF. El objetivo de esta cyber-range es simular ataques a los sistemas de virtualización y que sus usuarios puedan desarrollar competencias en el ámbito de la ciberseguridad.

Esta plataforma tiene el objetivo secundario de estudiar la seguridad de los sistemas de virtualización y contenedores. Con este trabajo también se intenta buscar las vulnerabilidades de estos sistemas y explotarlas. El propósito es que tanto los usuarios como el público en general puedan sacar conclusiones de la seguridad de estos sistemas y crear consciencia sobre la necesidad de la ciberseguridad.

Las **soluciones basadas en la iniciativa NFV** de la ETSI consisten en tres softwares: OSM, OpenStack y Kubernetes. De esta forma, la plataforma cyber-range cuenta con un orquestador proporcionado por OSM, un sistema de máquinas virtuales proporcionado por OpenStack y un sistema de contenedores proporcionado por Kubernetes. Los ataques simulados por la plataforma cyber-range y los ejercicios del CTF consistirán en atacar estos tres softwares, haciendo especial énfasis en OSM.

Tanto los ejercicios como los ataques planteados van especialmente dirigidos a las vulnerabilidades que se suelen encontrar en estos softwares. Estas vulnerabilidades pueden ser debidas a errores de administración o al uso de una configuración inapropiada o poco segura. Al tratarse de vulnerabilidades de los sistemas de virtualización, los ataques se centran en la infraestructura de red, la infraestructura de cómputo y los mecanismos de gestión. Dando menos importancia a los ataques de los servicios que se virtualizan.

La **competición CTF** consisten en varios ejercicios, los cuales están diseñados para que el usuario consiga una bandera. Esta bandera puede tratarse de ciertas credenciales o algún tipo de cadena de caracteres. Una vez el usuario haya logrado todas las banderas de los ejercicios, deberá validarlas para poder obtener una calificación.

Adicionalmente, se desarrollará un servidor web que pueda validar y calificar el trabajo de los usuarios en los ejercicios CTF. El servidor web debe ser una herramienta útil tanto para el usuario como para el instructor que supervisa. Por un lado, deberá calificar los esfuerzos del usuario. Por otro lado, deberá mostrar los resultados de todos los usuarios al instructor.

4. BENEFICIOS QUE APORTA EL TRABAJO

4.1. Beneficios sociales

El proyecto trata sobre una plataforma para el aprendizaje de competencias de ciberseguridad. Por lo tanto, el trabajo realizado cuenta con un claro beneficio social, la formación de personas en materia de ciberseguridad. Gracias a la plataforma desarrollada, los usuarios son capaces de comprender cómo funcionan los ataques de forma práctica. Estos ataques se realizan bajo la premisa del hacking ético. Por lo tanto, permite instruir a los usuarios en los procesos de realización de informes y demás mecanismos de protección derivados del hacking ético.

Existen beneficios adicionales, como el de enseñar a los usuarios mecanismos de ciberseguridad. Mediante los ejercicios prácticos planteados, los usuarios aprenden el proceso realizado por los verdaderos atacantes para explotar las vulnerabilidades de entornos reales. Una vez han comprendido la forma en la que se realizan los ataques, los usuarios son capaces de identificar las vulnerabilidades y proporcionar un sistema robusto y eficaz que reduzca la posibilidad de éxito de los ataques.

Un último beneficio sería la capacidad que tiene la plataforma desarrollada de concienciar sobre los peligros a los que están expuestos todos los sistemas informáticos. Los usuarios son ellos mismos los que atacan al entorno diseñado, comprenden los retos que supone realizar estos ataques y que no hace falta ser un experto para realizarlos. Por lo tanto, comprenden que estos mismos ataques pueden ser realizados por cualquiera y a cualquier equipo. En definitiva, todo este proyecto puede sensibilizar de que los ataques son muy comunes, que pueden ser muy dañinos y que se debe invertir en protegerse frente a ellos.

4.2. Beneficios técnicos

La plataforma desarrollada también cuenta con un claro beneficio técnico, la propia plataforma en sí misma. En este proyecto se ha desarrollado desde cero una plataforma de diferentes sistemas de virtualización. Principalmente, esta plataforma se ha plateado para la formación en materias de ciberseguridad. Haciendo posible que distintos usuarios se formen de forma práctica. Adicionalmente, la plataforma se encuentra disponible para que cualquiera se pueda formar.

En relación con lo anterior, la plataforma se ha diseñado para otorgar gran dinamismo y adaptabilidad a los ejercicios que se despliegan sobre ella. Este hecho permite diseñar ejercicios alternativos o adicionales a los que se proponen en este proyecto. Permitiendo a los instructores plantear la formación con otra dificultad, otra perspectiva u otros métodos.

A parte de para la funcionalidad previamente descrita, en vez de usar la plataforma en exclusiva para la formación, es posible utilizarla para desplegar servicios reales dentro de una red corporativa. Al fin al cabo, es una plataforma que combina la orquestación de OSM, las máquinas virtuales de OpenStack y los contenedores de Kubernetes, de forma agnóstica para el usuario final. Por lo tanto, con los descriptores adecuados, es posible desplegar bajo demanda cualquier tipo de servicio y que estos funcionen en cuestión de minutos con una escasa configuración por parte del administrador.

Como beneficio secundario, se encuentra el hecho de que con esta plataforma se pueden identificar las vulnerabilidades y puntos de riesgo de los sistemas de virtualización. Es decir, ayuda a determinar qué configuraciones suponen un mayor riesgo para el sistema y con qué instalaciones el administrador debe estar más cauteloso. También permite comprobar si la configuración que pretendía instalar un administrador va a desembocar en una vulnerabilidad o en un fallo de seguridad.

4.3. Beneficios económicos

El proyecto realizado permite formar a los empleados de las empresas en materia de ciberseguridad, con el objetivo de que estos luego puedan desplegar los servicios de la empresa de forma segura. La formación de sus empleados es un objetivo que toda empresa debe buscar, ya que hace que los empleados sean más productivos y que aumenten los beneficios de la empresa.

Como beneficio añadido, el formar a los empleados en materia de ciberseguridad hace que las vulnerabilidades sean más escasas y que, por lo tanto, los ataques sean menos exitosos. Es importante invertir en esta clase de formación porque los ataques informáticos siempre se traducen a pérdidas económicas. Los ataques pueden causar problemas de todo tipo, por ejemplo: robo de información, denegación de servicios, etc. En general, los ataques hacen que la empresa no se pueda operar, producir o facturar de forma normal y hace que los clientes e inversores pierdan confianza en la empresa. Por lo tanto, la ciberseguridad es algo que toda empresa debe invertir en asegurar.

De manera adicional, esta plataforma desarrollada se puede utilizar como una herramienta para desplegar servicios reales. Esto permite tener una infraestructura y servicios dinámicos que se adaptan a las necesidades de los usuarios sin invertir en hardware específico. Mediante esta tecnología se obtiene una mayor rentabilidad de los servicios desplegados y la amortización de los nuevos servicios es mucho menor.

5. ANÁLISIS DEL ESTADO DEL ARTE

Las plataformas cyber-range son una tecnología que se está popularizando en los últimos años. Debido a esto, cada vez son más las organizaciones que desarrollan alguna implementación. A continuación, estudiaremos algunas de estas implementaciones, analizando el objetivo de dichas plataformas y las características que las definen.

Uno de estos softwares es **Cyberbit** [2]. Este software proporciona una plataforma cyber-range muy completa. Cuenta con un gran catálogo de ejercicios, los cuales enseñan a los empleados de una empresa a enfrentarse a distintos ataques. Esta es una de sus fortalezas, la una gran variedad de tipos de ataques que tiene disponible para que los empleados practiquen. A parte de los ataques, en su catálogo de ejercicios también cuenta con distintos softwares comerciales. De esta forma, los empleados pueden practicar a defenderse en los softwares que utilizan de forma normal en su trabajo.

A parte de lo anterior, Cyberbit tiene la capacidad de simular redes corporativas de pequeño o gran tamaño. Si se contrata el servicio adecuado, permite recrear la red corporativa de la empresa contratante. De esta forma, los empleados de dicha empresa podrán aprender a identificar los ataques que puedan suceder durante sus horas de trabajo.

Por último, Cyberbit proporciona herramientas para supervisar el progreso de los empleados de forma automática. Adicionalmente, provee gráficos útiles del desempeño de los empleados en los ejercicios. Para realizar estos gráficos se basa en múltiples factores, por ejemplo: número de elementos que se han investigado de forma correcta, número de procesos maliciosos detenidos de forma exitosa o número de archivos maliciosos eliminados.

En resumen, Cyberbit proporciona una plataforma muy compleja con multitud de funcionalidades. Sin embargo, está diseñado de tal forma que no resulta especialmente útil para este trabajo desarrollado. Por un lado, Cyberbit está diseñado para simular los ataques y hacer que los empleados se defiendan ante ellos. En cambio, en el trabajo desarrollado, se necesita que sean los empleados los que ejecuten los ataques para poder realizar los ejercicios CTF. Por otro lado, los ejercicios de Cyberbit están diseñados para que varios empleados se dividan en equipos y entre ellos solucionen los ejercicios. En cambio, en el trabajo desarrollado, se platea unos ejercicios más individuales.

Otro software que hace una implementación de esta tecnología es **Cyberranges** [3]. Este software se centra principalmente en crear una réplica de la infraestructura de red de la organización contratante. Después, sobre esta infraestructura simulada entrenan a los trabajadores de la empresa para que sepan proteger y evitar ataques en la red real. Como alternativa, la plataforma también deja a la disposición de la empresa unas librerías con escenarios diversos para que sus empleados puedan entrenar en otros entornos.

A parte de para la formación, esta plataforma permite utilizar la réplica de la red corporativa para más funciones de distintas naturalezas. Una de estas funciones es comprobar la respuesta de la empresa ante distintos ataques y peligros, haciendo un análisis de lo vulnerable que es su la infraestructura de la red. Otra función extra es la de permitir usar la red simulada para comprobar nuevos productos y tecnologías en un entorno seguro sin interferir con la infraestructura real. Una última función es la de usar la red para desplegar una competición de ataque y defensa donde dos grupos de empleados se enfrentan para obtener un objetivo del otro equipo.

El software Cyberranges destaca en crear una réplica fidedigna de la red de una empresa y en formar a los empleados en mantener la seguridad de esta red. Sin embargo, las necesidades de este proyecto no son tanto crear una red ficticia a partir de una ya existente, como desplegar una red interesante para ejecutar una competición de CTF. El proyecto no busca implementar una plataforma que simule a la perfección unos servicios reales, sino que implemente unos sistemas de virtualización que puedan ser objetivos de las prácticas del hacking ético.

Otro software distinto es **Cyberwiser** [4]. Este software utiliza una plataforma cyber-range para ofrecer distintos cursos sobre ciberseguridad. Están clasificados como: *primer, básico, intermedio y avanzado*. Varias empresas y organizaciones públicas utilizan estos cursos para que sus empleados aprendan competencias en materia de ciberseguridad.

El curso más sencillo es *primer*. Está planteado para gente que no tiene ningún conocimiento de ciberseguridad. Este curso enseña conceptos básicos y buenas prácticas. El siguiente curso en dificultad es *básico*. Este curso enseña a los usuarios a evaluar los riesgos y otros peligros para ciberseguridad que pueden ocurrir en una red. Este es el primer nivel que utiliza una plataforma cyber-range para el entrenamiento. A continuación, se encuentra el curso *intermedio*. Este curso enseña a evaluar las vulnerabilidades de la red y las pérdidas económicas derivadas de un ataque. La plataforma cyber-range que utiliza es de hasta 50 máquinas virtuales. Por último, el curso *avanzado* es para empleados que ya tienen experiencia en ciberseguridad, grandes empresas u organizaciones públicas.

Esta solución ofrece unos cursos muy cerrados que quizás sea útil para una organización o individuo que busque formarse en materia de ciberseguridad. Sin embargo, no se ajusta a ninguna de las necesidades del proyecto. El trabajo desarrollado busca poder desplegar distintos ejercicios sobre una red virtual, no usar los cursillos de terceros.

En definitiva, los softwares analizados que implementan las tecnologías de cyber-range cuentan con funcionalidades interesantes. No obstante, ninguno se ajusta a las necesidades del proyecto. La mayor diferencia entre los softwares y el proyecto desarrollado se encuentra en la propia premisa. Mientras que estos softwares se centran en simular entornos reales y realizar los ataque contra los servicios que se despliegan, el proyecto desarrollado busca estudiar las vulnerabilidades del propio sistema de virtualización. Es decir, uno de los objetivos del proyecto es estudiar las vulnerabilidades de los propios sistemas de virtualización y no de los servicios virtualizados.

6. ANÁLISIS DE ALTERNATIVAS

6.1. Sistemas de virtualización

Este análisis de alternativas trata de encontrar las herramientas que más opciones puedan otorgar a la realización de los ejercicios. Entre los objetivos del proyecto se encuentran la realización de ataques contra dos sistemas de virtualización diferentes. Por un lado, ataques contra un sistema de contenedores y, por otro lado, ataques contra un sistema de máquinas virtuales. Por lo tanto, al tratarse de dos tecnologías diferentes, se debe realizar dos análisis de alternativas diferentes, buscando las herramientas de cada tecnología que más alternativas ofrezcan a los ejercicios.

6.1.1. Sistema de contenedores

Existen muchos softwares que implementan tecnologías de contenedores. Entre ellos, los más populares son Docker, Kubernetes y las plataformas cloud. A continuación, se discutirá las ventajas e inconvenientes de cada uno de ellos.

Docker [5] es un software de código abierto que actúa como una aplicación cliente-servidor. Cuenta con una comunidad de desarrolladores muy amplia, lo que le proporciona una gran cantidad de *plugins* e imágenes. A la hora de instalarlo, puede suponer algo complejo de implementar. Pero, una vez configurado, tiene un sistema de administración de contenedores que apenas requiere la participación de un administrador. Lo cual le concede al administrador una reducción en su carga de trabajo. Sin embargo, para poder utilizar este software de forma completa se necesita un proceso de aprendizaje que puede ser largo.

Kubernetes [14] también es un software de código abierto. Actúa bajo una estructura de clúster y una topología maestro-esclavo. Además de ser compatible con las imágenes que han sido creadas para Docker, también tiene una comunidad muy amplia que desarrolla microservicios específicos. Por lo tanto, cuenta con un cátalo de servicios muy amplio. Adicionalmente, escala los servicios automáticamente en función de los recursos utilizados, aunque también, permite al administrador escalar los servicios de forma manual. En comparación, Kubernetes es un software sencillo de instalar.

En cuanto al proceso de aprendizaje, se necesita invertir una cantidad de tiempo considerable para comprender la arquitectura, los archivos de configuración y su modo de funcionamiento. Pero una vez se comprenden los archivos de configuración, son muy sencillos de utilizar e implementar. Del mismo modo, configurar un clúster de forma manual requiere conocimientos del sistema y puede ser tedioso. Sin embargo, para aplicaciones genéricas Kubernetes ofrece la opción de usar *Helm-charts*.

Las **plataformas cloud** son un entorno donde se ejecutan aplicaciones de un sistema, normalmente, externo. Estas plataformas ofrecen una gran cantidad de servicios útiles, como Infraestructura como servicio (IaaS), Plataforma como Servicio (PaaS) o Software como Servicio (SaaS). Entre todas estas funcionalidades, los servicios de contenedores son los que interesan para este proyecto. Los sistemas cloud corporativos más típicos, y que también ofrecen un servicio de contenedores son *Microsoft Azure, Google Kubernetes Engine y Amazon Elastic Container Service* (Amazon ECS).

Microsoft Azure [15] ofrece una gran cantidad de servicios útiles para los sistemas en la nube a parte de los servicios de contenedores (PaaS, almacenamiento, computo, inteligencia artificial, etc.). Se trata de una plataforma muy completa y compleja. Incluso ofrece hasta un total de siete servicios de contenedores, permitiendo a los desarrolladores elegir el tipo de contenedor que más se ajuste a sus necesidades.

Google Kubernetes Engine (GKE) [13] es un sistema de contenedores que se sostiene sobre la plataforma Google Cloud. GKE ofrece una plataforma arrendable que implementa Kubernetes de una forma robusta y segura. GKE despliega el clúster de sus clientes dentro de su cloud privado de Kubernetes y realiza el mantenimiento y administración de dicho clúster, asegurando siempre su seguridad. Cabe destacar que Kubernetes es un proyecto de orquestación de código abierto de Google y que, por lo tanto, con GKE tratan de dar un soporte a dicha tecnología. Sin embargo, el soporte proporcionado por GKE no es necesario para utilizar Kubernetes.

Amazon Elastic Container Service (Amazon ECS) [1] es un sistema de contenedores de Amazon que se integra dentro de la plataforma Amazon Web Services (AWS). De forma similar a las anteriores plataformas cloud, Amazon ECS también ofrece un servicio de contenedores a sus clientes. También realiza el mantenimiento y administración de los contenedores dentro de sus plataformas privadas. De esta forma, los desarrolladores pueden centrarse en desarrollar sus aplicaciones, delegando la ejecución y mantenimiento de estas a Amazon ECS.

Entre estas tres opciones hay una que es la que más se ajusta a las necesidades del proyecto. Anteriormente, ya se han expuesto brevemente las características de cada una de ellas. A continuación, se discutirá el motivo por el que estas opciones han sido rechazadas o admitidas.

Por un lado, las plataformas cloud son demasiado extensas para los objetivos de este proyecto. Al no ofrecer únicamente sistemas de contenedores, ofrecen muchos servicios que no son necesarios para el proyecto. Es decir, requieren instalar un sistema muy complejo para luego sólo utilizar una única funcionalidad de todas las que ofrece. Adicionalmente, al ejecutar los contenedores en las respectivas redes privadas de cada empresa, imposibilitan la ejecución de muchas vulnerabilidades que son importantes para los ejercicios. Lo cual hace que, en caso de elegir esta alternativa, los ejercicios sean muy limitados. A consecuencia de estos hechos, no se considera que las plataformas cloud se ajusten a las necesidades del proyecto y, por lo tanto, se descarta esta opción.

Por otro lado, Docker ofrece una plataforma que se ajusta muy bien a los requerimientos del proyecto, ya que cuenta con un gran repositorio de distintas imágenes y configuraciones. Adicionalmente, al ser una plataforma que se instalaría en la red del laboratorio, permite al desarrollador del proyecto una gran maniobrabilidad en el diseño e implementación de los diferentes ejercicios. Es una opción que podría haber sido válida para implementar si no fuera porque se considera que Kubernetes aporta mayores ventajas.

Por último, Kubernetes, al ser compatible con las imágenes de Docker, ofrece sus mismas ventajas. Al mismo tiempo, incluye funciones exclusivas de Kubernetes. Al ser también de código abierto, otorga al desarrollador la misma maniobrabilidad que Docker. Además, la opción de evitar la tediosa instalación de los servicios del clúster mediante el uso de Helm-charts, permite realizar ejercicios interesantes para la competición CTF que se plantea. Adicionalmente, no se considera que el proceso de aprendizaje sea un punto negativo, dado que, todas las opciones conllevan un proceso de aprendizaje. Por lo tanto, se considera que **Kubernetes** es la alternativa que más se ajusta a los requerimientos del proyecto.

6.1.2. Sistema de máquinas virtuales

En cuanto a sistemas de máquinas virtuales en redes de cómputo en la nube, existen tres alternativas principales: VMware ESXi, XEN y OpenStack. Estas dos alternativas son los softwares más usados para este tipo de sistemas. A continuación, se analizará en más detalle cada una de estas dos alternativas.

XEN [20] es un software de un hipervisor. Estos tipos de softwares son sistemas que permiten la ejecución de múltiples sistemas operativos huésped simultáneamente dentro de un único equipo físico anfitrión. Este hipervisor en concreto se ejecuta directamente encima del nivel físico del equipo, sin la necesidad de ningún sistema operativo que lo soporte. Con este software se puede crear un servidor que virtualice una gran cantidad de sistemas operativos, pero no proporciona ningún servicio adicional a este. No ofrece Infraestructura como Servicio (IaaS) y la administración, despliegue y mantenimiento de las distintas máquinas virtuales recae completamente en el administrador del sistema, lo cual puede ser tedioso cuando se cuenta con un número elevado de virtualizaciones.

VMware ESXi [19] es otro software de hipervisor. Al igual que XEN, también se ejecuta directamente sobre el nivel físico del servidor en el que se despliega. Sin embargo, este software forma parte de otro software más completo que ofrece la compañía VMware para la virtualización de servidores y de centros de datos. Este otro software es VMware vSphere y se trata de un software propietario que necesita una licencia para poder ser usado.

OpenStack [18] es un software de código abierto con una gran comunidad, la cual desarrolla multitud de herramientas para este software. Estas herramientas, por ejemplo, facilitan la automatización de las instancias, la interoperabilidad entre distintos elementos de la red, etc. Adicionalmente, a través de sus múltiples APIs, OpenStack permite que programas externos envíen comandos a los distintos elementos que componen OpenStack. De esta forma, OpenStack ofrece servicios altamente personalizables y adaptables que cumplen todas las necesidades de sus usuarios. Otra ventaja que ofrece es su escalabilidad. OpenStack facilita el trabajo de administrador en este sentido, ya que con pocas operaciones es posible desplegar nuevas instancias de servicios o réplicas de los antiguos.

Sin embargo, los múltiples componentes y la arquitectura compleja de OpenStack hace que la instalación de un servidor OpenStack sea muy compleja. Normalmente, este proceso de instalación, además de un administrador experto, requiere invertir muchos recursos y mucho tiempo en él.

Otro punto a favor a tener en cuenta, es que, el grupo de investigación donde se ha realizado el trabajo cuenta con investigadores que conocen este software, los cuales pueden ofrecer su ayuda en caso de ser necesario. Esta ayuda puede facilitar el proceso de aprendizaje necesario para aprender a utilizar el software y agilizar el desarrollo del proyecto.

Entre estas tres alternativas hay una que se adapta mejor al trabajo a realizar. A continuación, se discutirá lo conveniente que son estas alternativas para el proyecto.

Por un lado, XEN es un software sencillo y fácil de implementar, pero que no alcanza las necesidades del proyecto. Al tratarse únicamente de un software hipervisor y al no realizar las funciones de VIM, se considera que no es la opción que más se adapta a lo que propone la iniciativa NFV de la ETSI. Por lo tanto, los ejercicios que se pudieran proponer sobre este software no llegan a encajar bien con los requerimientos del proyecto. Por todo lo anterior, no se considera que XEN sea una opción válida para el proyecto.

Por otro lado, VMware ESXi cuenta con una plataforma muy profesional y efectiva. Sin embargo, para este proyecto es preferible utilizar software de código abierto, no propietario, que permita analizar en profundidad la arquitectura del sistema, con el objetivo de poder encontrar alguna vulnerabilidad. Se considera que este criterio es crucial a la hora de elegir una buena alternativa, por lo tanto, se descarta la opción de utilizar VMware ESXi.

Por último, OpenStack cuenta con las ventajas de ser código abierto y con las herramientas diseñadas por la comunidad. Además, proporciona unas plataformas de APIs que resultan muy interesantes por múltiples razones. Por ejemplo, permite la interoperabilidad de forma sencilla con el resto de softwares que se utilizan en el proyecto. Del mismo modo, permite acceder a diferentes puntos de la arquitectura de OpenStack y estudiar las posibles vulnerabilidades que pueden ser explotadas en un ejercicio de la competición CTF. Teniendo en cuenta todos los detalles de las alternativas previamente expuestas, se considera que la alternativa que mejor se adapta al trabajo a realizar es el uso del software **OpenStack**.

6.2. Despliegue de la infraestructura de máquinas virtuales

Debido a que se ha elegido previamente a OpenStack como la alternativa adecuada para el proyecto, ahora se debe realizar otro estudio de alternativas en lo relativo al despliegue de esta infraestructura. Esto se debe a que existen tres alternativas de despliegue que deben ser estudiadas por ser interesantes para el trabajo a realizar. La primera alternativa es usar el servidor que se utiliza actualmente en la red de producción del laboratorio. La segunda, es montar un servidor OpenStack nuevo en un equipo propio. La última alternativa es utilizar Microstack en vez de OpenStack.

La primera opción es utilizar el **servidor OpenStack de la red de producción** del laboratorio. De este modo, se parte directamente de un servidor que tiene garantías de funcionar correctamente desde el primer día del desarrollo. Por lo tanto, con esta alternativa, al no tener que realizar la instalación del servidor, es más rápido el proceso de aprendizaje e inicio de desarrollo de servicios. Sin embargo, esta alternativa también cuenta con un gran inconveniente. Debido a que la naturaleza de este proyecto incluye la realización de ataques al sistema de virtualización, el servidor del laboratorio puede resultar comprometido. Lo cual, dañaría tanto la propia infraestructura como el trabajo que realizan el resto del personal del laboratorio en el servidor.

La segunda opción es **montar un servidor OpenStack privado en un equipo propio**. Esta alternativa tiene las características opuestas a la alternativa anterior. Por un lado, se evita el explotar las vulnerabilidades de un servidor que es usado por otros miembros del laboratorio. Haciendo posible realizar el proyecto sin comprometer el trabajo de terceras personas. Por otro lado, la instalación de un servidor OpenStack desde cero es costoso y requiere invertir una cantidad significativa de tiempo para poder desplegarlo de forma correcta.

La última opción es **montar un servidor de Microstack**. Microstack ofrece un servicio muy similar a OpenStack a una escala menor. Esta versión de OpenStack es más sencilla de instalar y permite realizar ataques contra ella sin dañar el servidor OpenStack de producción. Sin embargo, al ser una versión más simple, tiene unos inconvenientes, sobre todo en la forma que gestiona los recursos de red. Microstack despliega las máquinas virtuales en una red virtual interna y les otorga una conexión con internet haciendo NAT. Por lo tanto, puede resultar complejo implementar los ejercicios en este sistema.

A la hora de discutir qué alternativa se ajusta más a los objetivos del proyecto, la opción de montar un servidor de Microstack es la que más inconvenientes presenta. Esto se debe a que sus limitaciones técnicas lastran demasiado la forma de diseñar los ejercicios del proyecto. Por lo tanto, no se considera que sea una buena alternativa.

Por otro lado, montar un nuevo servidor OpenStack necesitaría invertir demasiado tiempo en una fase de instalación que no es relevante para los objetivos del proyecto. Por lo tanto, aunque se trate de una alternativa muy atractiva, no se considera que sea la mejor alternativa para este proyecto.

Por último, la alternativa que más se ajusta a los objetivos del proyecto es la de **usar el servidor OpenStack de la red de producción**. Gracias al mecanismo que incorpora OpenStack de dividir a los usuarios en *proyectos* o *tenants* es posible separar el trabajo de cada usuario de forma que no se molesten entre ellos. De esta forma, al no interferir con otros usuarios, se pueden evitar la mayoría de las posibles vulnerabilidades causadas por los ejercicios de la competición CTF.

7. ANÁLISIS DE RIESGOS

Se han determinado cinco posibles riesgos que pueden afectar al desarrollo del proyecto. Por lo tanto, se ha realizado este análisis de riesgos con el propósito de cuantificar el efecto que tendrían estos riesgos en caso de producirse. Los riesgos identificados son:

A. Los ataques afectan a la red de producción.
B. Fallo de hardware o de la infraestructura física.
C. Desviaciones con respecto a la planificación.
D. Desviación con respecto al presupuesto.
E. Diseño de ejercicios no implementables.

En la Tabla 1, se han clasificado los riesgos detectados en función de su probabilidad y el impacto que tendrían en el proyecto.

Probabilidad \ Impacto	Alto	Medio	Bajo
Alta	**A**		
Media		**E**	**C**
Baja	**B**	**D**	

Tabla 1: Análisis de riesgos

A. Los ataques afectan a la red de producción.

Este riesgo ocurre cuando los ataques que se realizan contra el servidor OpenStack afecta a los demás usuarios del servidor. Se considera que tiene una probabilidad relativamente alta de suceder, ya que los ataques son una acción que se está realizando de forma activa y continuada durante todo el proyecto. Al mismo tiempo, se considera que, en caso de ocurrir, el impacto sería alto, ya que podría afectar a otros usuarios ajenos al proyecto.

Las consecuencias de este riesgo podrían derivar en varios problemas como la denegación de servicios o perdidas de información, entre otros. Para solucionar estas consecuencias, se podría realizar reinicios del servidor, revertir el servidor a un estado anterior con un backup o tener que modificar alguna configuración.

Para evitar este riesgo, se debe ser muy cuidadoso con el tipo de ejercicios a implementar. Diseñándolos de tal forma que no supongan un riesgo real para el trabajo de otras personas. Adicionalmente, se debe intentar separar lo máximo posible el entorno de trabajo de los usuarios de la cyber-range de los usuarios ajenos al proyecto. Para realizar esta acción, ya se ha comentado que existe la posibilidad de realizar *tenants* independientes, los cuales tienen este mismo objetivo.

B. Fallo de hardware o de la infraestructura física.

Este riego ocurre cuando aparece un fallo de hardware o de la infraestructura física, como por ejemplo en la red eléctrica, en la red de comunicaciones o en los equipos utilizados. Estos fallos pueden ser derivados de acontecimientos puntuales como cortes de luz, saturación de la capacidad de cómputo, etc. La probabilidad de que estos tipos de riesgos sucedan en muy baja. Sin embargo, el impacto que tendrían estos sucesos puede suponer un gran impacto.

Este tipo de fallos son totalmente ajenos al proyecto y, por lo tanto, no se puede realizar ninguna acción para reducir la probabilidad o evitar que sucedan. Sin embargo, se puede invertir en reducir el alto impacto que tendrían. Mediante técnicas de backup y de redundancia, se puede lograr que el impacto de estos fallos sea mínimo y que, en caso de ocurrir, se pueda recuperar la producción normal en cuestión de pocos instantes.

C. Desviaciones con respecto a la planificación.

Este riesgo es un riesgo de gestión y sucede cuando se excede los plazos que se tenían previstos en la planificación inicial. Es un riesgo con una probabilidad media de suceder, ya que este proyecto incluye un periodo de aprendizaje de hasta tres softwares complejos diferentes, cada uno con sus características y propiedades. Este periodo de aprendizaje es un periodo largo y complejo. El cual, es importante para comprender los sistemas y a posteriori diseñar los ejercicios. Por lo tanto, el ralentizarse en este proceso puede suponer desajustar toda la planificación prevista.

Dado que se ha iniciado este proyecto con la suficiente antelación, en caso de que sucedan retrasos, estos no repercutirían el desempeño del proyecto más allá de retrasar su fecha de finalización. Este hecho no supondría un problema, ya que, la planificación inicial cuenta con un margen de tiempo previo a la entrega del trabajo.

D. Desviación con respecto al presupuesto.

Este riesgo es un riesgo de gestión y sucede cuando se excede el presupuesto planteado para el proyecto. Es un riesgo con una probabilidad baja de suceder ya que los equipos que se van a utilizar ya se encuentran disponibles en el laboratorio y el software a utilizar es, en su mayoría, de código abierto. Por lo tanto, hay pocos gastos que puedan realmente desviar el proyecto del presupuesto planteado.

Sin embargo, en caso de suceder, puede tener un impacto moderado en el proyecto. Dependiendo del caso, podría llegar a forzar a modificar algún elemento del planteamiento, del diseño o del despliegue del proyecto. Lo cual podría resultar con un proyecto diferente del actualmente presentado.

E. Diseño de ejercicios no implementables.

Este riesgo plantea el suceso de que un ejercicio diseñado no pueda ser implementado en el cyber-range diseñado. Es un riesgo con una probabilidad media de suceder y con un impacto medio también. Esta posición en mitad de la tabla se debe a que es un riesgo que es muy dependiente del problema de implementación que lo genere.

En caso de suceder, podría derivar a dos situaciones diferentes. Por un lado, se podría considerar totalmente imposible de implementar. En este caso, sería necesario replantear este ejercicio o desecharlo completamente. Por otro lado, se podría necesitar un cambio del escenario para que el ejercicio sea implementable. En este caso, sería necesario replantear el diseño del escenario y volver a desplegarlo con las modificaciones pertinentes.

8. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

8.1. Arquitectura del despliegue del escenario

A la hora de describir el escenario implementado para la solución, se pueden distinguir dos escenarios. Por un lado, el escenario de las infraestructuras físicas que contienen los servicios virtualizados sobre los que se despliegan las plataformas cyber-range con las competiciones CTF. Por otro lado, el escenario de máquinas virtuales y contenedores en el que se ejecutan los ejercicios diseñados para los usuarios.

Adicionalmente, también se ha diseñado un servidor Web que sirve para validar los ejercicios de la competición. Este servidor se despliega sobre un NS y se puede considerar parte del escenario.

8.1.1. Arquitectura de la infraestructura general

Este escenario está implementado en la red del laboratorio y, por lo tanto, hace uso de los equipos y recursos que se encuentran disponibles en esta red. El escenario cuenta con cuatro equipos físicos, los cuales son un servidor OpenStack, un servidor OSM, un equipo de monitorización y un clúster de Kubernetes. Estos cuatro equipos se dividen en dos redes separadas por un router que también da acceso a Internet. Adicionalmente, también se cuenta con un switch, colocado entre el router y el servido OSM, al que se le conecta el equipo de escucha. En la Ilustración 6 se puede observar un diagrama de la arquitectura descrita previamente.

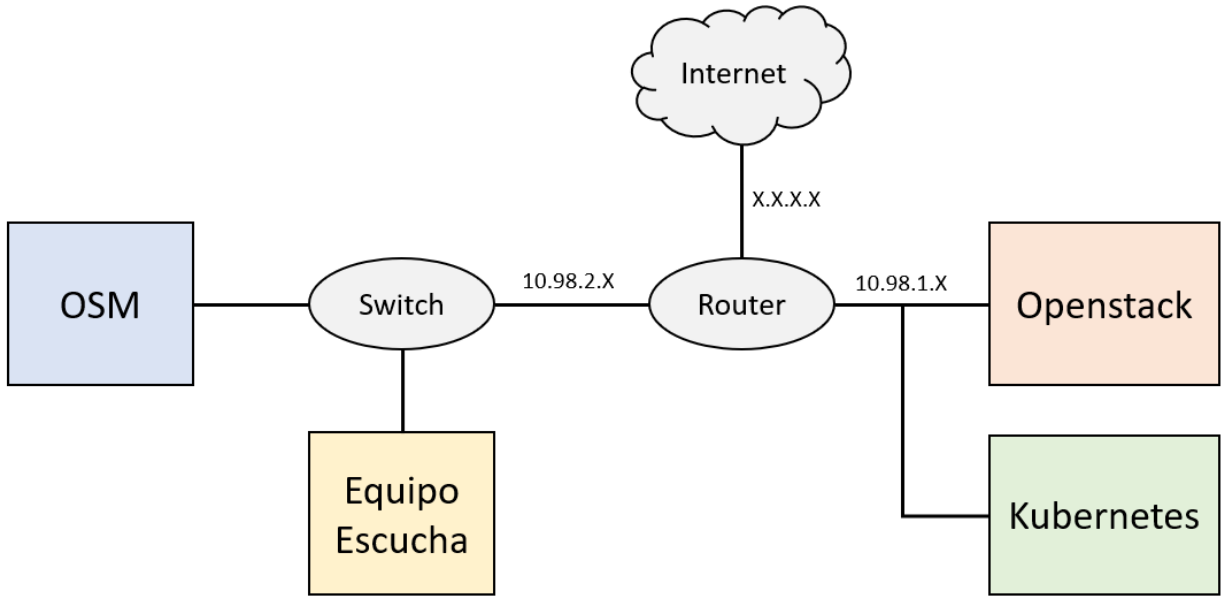

Ilustración 6: Arquitectura de la infraestructura física desplegada

En los siguientes apartados se describirán detalladamente las características de estos elementos de la infraestructura.

8.1.1.1. Servidor OpenStack

Como se ha mencionada en el apartado *6.2 Despliegue de la infraestructura de máquinas virtuales* el servidor OpenStack que se utilizará para la plataforma cyber-range es el que se encuentra en la red de producción del laboratorio. Por lo tanto, la única operación que será necesaria hacer cómo paso previo al desarrollo de los ejercicios será la creación de un nuevo *tenant*. Un *tenant* es sinónimo de *proyecto* en la terminología de OpenStack, para obtener una descripción más detallada del funcionamiento y terminología de OpenStack ver apartado *14.3: OpenStack* del anexo. Este *tenant* separa las plataformas cyber-range de este proyecto de los proyectos realizados por otros miembros del laboratorio y permite realizar los ejercicios sin miedo de interferir en sus trabajos.

Una vez se haya completado la fase de aprendizaje del software y de diseño de los ejercicios de la competición CTF y de los elementos de la cyber-range, se procederá al despliegue de estos. El diseño de los ejercicios será descrito más adelante en el apartado *8.2: Descripción de los ejercicios del CTF*. Para realizar el despliegue serán necesarios principalmente estos elementos: *Imágenes, volúmenes y Security Groups.*

Las **imágenes** son elementos que se cargan en el servidor para luego ser instanciados en distintas máquinas virtuales. De esta forma, se pueden desplegar de forma rápida y sencilla servicios que ya han previamente diseñados. Para los ejercicios diseñados será necesario tres imágenes de Ubuntu. Una imagen debe contar con un servidor web y se usará para la evaluación de los ejercicios. Otra imagen con servicios de red como SSH, net-tools, etc. Una última imagen con la configuración más básica posible, para poder instalar servicios desde OSM.

Los **volúmenes** serán necesarios para un ejercicio y deben ser creados previamente. Una vez creados en el servidor OpenStack, deben ser asignados a una instancia del cyber-range. En la instancia, deben ser montados con permisos exclusivos de administrador y en el volumen se debe crear un fichero. Cada usuario debe tener un volumen, por lo que se debe repetir esta acción para cada usuario de la plataforma cyber-range.

Los **Security groups** permiten el acceso de cierto tipo de tráfico a una determinada instancia. Funcionan de manera similar a un cortafuegos tradicional, permitiendo el transcurso de cierto tipo de tráfico y denegando otro. Para la plataforma cyber-range se deberán definir tres tipos de Security groups: Uno permisivo con la mayoría del tráfico, otro restrictivo y uno que tan sólo permita el tráfico HTTP.

8.1.1.2. Clúster Kubernetes

El primer paso es la instalación del clúster de Kubernetes. No es una tarea excesivamente compleja, pero puede llevar su tiempo al administrador del sistema. Para ello, primero, se debe instalar *Docker, Kubelet, Kubeadm y Kubectl*. Después, se debe inicializar el clúster y configurar los archivos *Configmap y Kubeconfig*. Por último, se debe gestionar los recursos de almacenamiento. En la guía de usuario [14] se describe de forma detallada los pasos de la instalación. Para una descripción más detallada de los elementos de Kubernetes ver el apartado *14.4 Kubernetes* del anexo.

Una vez se ha completado el proceso de instalación, el aprendizaje del software y el diseño de los ejercicios, se debe preparar el clúster para desplegar la plataforma cyber-range. El ejercicio referente a Kubernetes, el cual se describe en el apartado *8.3.4: Ejercicio 4: Helm-chart*, consta de un Helm-chart personalizado que debe ser instalado. Un Helm-chart es una forma rápida de instalar un servicio estándar en un clúster de Kubernetes. Para la instalación, se sube el Helm-chart personalizado a un repositorio, el cual se vincula al servidor OSM, y se instala desde dicho servidor. Este método de instalación permite un despliegue rápido para los múltiples usuarios de la plataforma cyber-range.

En caso de ser necesario para los ejercicios, los usuarios también pueden recibir un entorno aislado dentro del clúster de Kubernetes. La herramienta que hace posible esta acción se denomina *Namespaces*. De manera análoga a los *tenants* de OpenStack, con los *Namespaces* es posible dividir los trabajos de los distintos usuarios para que estos no interfieran entre sí y no se molesten en sus ejercicios.

8.1.1.3. Servidor OSM

Al igual que en el anterior caso, el primer paso es la instalación del servidor OSM. Un servidor OSM requiere de muchos recursos de procesamiento y de almacenamiento, por lo que es aconsejable contar con un hardware que cumpla estas necesidades. El proceso de instalación lleva tiempo, pero está prácticamente automatizado, lo cual permite instalar OSM mientras se realizan otras tareas.

Una vez se ha terminado de instalar el servidor, se debe vincular con el clúster de Kubernetes y con el servidor OpenStack. Para ello, hay que asegurar que las *APIs* de los tres elementos son accesibles a través de sus interfaces de red, que cuentan con usuarios o *tenants* válidos y que no haya ningún cortafuegos o *security group* denegando las transmisiones. Finalmente hay que crear una nueva cuenta de *VIM* en el servidor OSM. Esta cuenta guarda todos los datos necesarios para desplegar un servicio en el equipo OpenStack o Kubernetes. Es una forma de aligerar el trabajo del administrador, vinculando los servidores y evitando el repetir información redundante cada vez que se despliega un servicio.

Una vez se ha terminado el proceso de instalación, el proceso de vinculación y el proceso de aprendizaje, es cuando se inicia el proceso de diseño de los ejercicios. Los ejercicios diseñados son traducidos a descriptores yaml que deben ser cargados en el servidor OSM y, posteriormente, instanciados. Para conocer mejor este proceso de instanciación y los detalles de los elementos del servidor OSM, ver el apartado *14.2 OSM: Open Source MANO* del anexo.

8.1.1.4. Equipo de monitorización y switch

El equipo de monitorización se trata de un equipo físico tradicional, con Ubuntu como sistema operativo, que se encuentra conectado a un switch. Este switch se encuentra entre el servidor OSM y el router que conecta con la red del servidor OpenStack y el clúster de Kubernetes. El objetivo de este equipo es ser el punto de monitorización de las comunicaciones entre OSM y los demás sistemas de virtualización. Para realizar esto, es necesario configurar el switch de tal forma que envíe una copia del tráfico de la interfaz del servidor OSM a la interfaz del equipo de monitorización. Adicionalmente, para poder recoger ese tráfico, será necesario instalar un software de monitorización, como por ejemplo *tcpdump* o *Wireshark*.

Los usuarios de la plataforma cyber-range deberán acceder a este equipo como primer paso para realizar los ejercicios. Por lo tanto, se debe preparar un usuario en el sistema para cada uno de ellos. De esta forma, cada uno podrá realizar las escuchas sin interferirse los unos en los otros.

8.1.2. Arquitectura de los ejercicios diseñados

Se ha diseñado un *Network Service* (NS) que agrupa todos los recursos necesarios para que un usuario realice los ejercicios de la competición CTF. De esta forma, se logra poder desplegar ejercicios a demanda en función del número de usuarios participando en la competición. Este NS consta de un cyber-range con los ejercicios dos, tres y cuatro. El ejercicio 1 no tiene ningún elemento que desplegar porque es un ataque que se realiza sobre equipos físicos. En la Ilustración 7 se muestra de manera esquemática la forma en la que está compuesto el NS del escenario de los ejercicios.

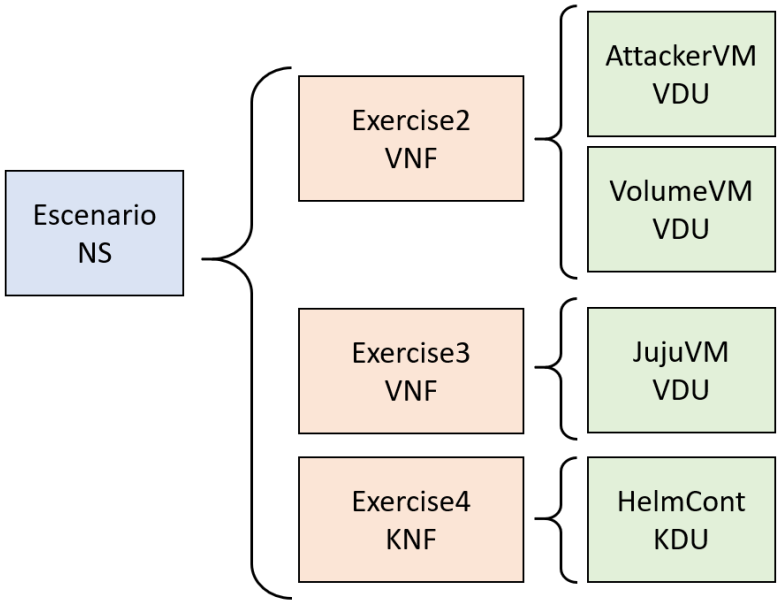

Ilustración 7: Esquema del Network Service que compone el cyber-range

8.1.2.1. NS del escenario

El *Network Service* (NS) del escenario está compuesto por tres *Virtualized Network Function* (VNF). En la Tabla 2Tabla 2: Descriptor abreviado del NS del Escenario se muestra el contenido resumido del descriptor *yaml* que lo conforma. Como se puede observar, y como se ha mencionado anteriormente, el ejercicio uno no está definido dentro del descripto *yaml* porque no es necesario ninguna infraestructura virtual para realizarlo.

```
nsd:
  […]
      vnf-profile:
      - id: '1'
        virtual-link-connectivity:
        - constituent-cpd-id:
           - constituent-base-element-id: '1'
             constituent-cpd-id: vnf-mgmt-ext
          virtual-link-profile-id: mgmtnet
        vnfd-id: exercise2-vnf
      - id: '2'
        virtual-link-connectivity:
        - constituent-cpd-id:
          - constituent-base-element-id: '2'
            constituent-cpd-id: vnf-mgmt-ext
          virtual-link-profile-id: mgmtnet
        vnfd-id: exercise3-vnf
      - id: '3'
        virtual-link-connectivity:
        - constituent-cpd-id:
          - constituent-base-element-id: '3'
            constituent-cpd-id: mgmt-ext
          virtual-link-profile-id: mgmtnet
        vnfd-id: exercise4-knf
    id: scenario-ns
    […]
```

Tabla 2: Descriptor abreviado del NS del Escenario

Como se puede apreciar en la Tabla 2, en el descriptor del NS se indica la cantidad de VNFs que va a componer este objeto de servicio, tres en este caso. Cada una de estas VNFs contienen varios elementos: un identificador diferente, el nombre del VNF o KNF a instanciar, una interfaz de red llamada *connection point descriptor* (CPD) y una red llamada *mgmtnet* que, a la hora de la instanciación, será sustituida por la red 10.98.1.X del laboratorio.

8.1.2.2. VNF del ejercicio 2

Tal y como se muestra en la Ilustración 7 y se describe en la Tabla 3: Descriptor abreviado del VNF del ejercicio 2

, el VNF del ejercicio 2 está compuesto por dos *Virtualized Depolyment Unit* (VDU). Una VDU es el elemento más básico de las funciones virtualizadas de OSM. Se ha optado por desplegar dos VDU para que cada una de ellas contenga una máquina virtual, por un lado, la que se usará para adjuntar el volumen y, por otro lado, la que se usará para que el atacante pueda leer el volumen.

```
vnfd:
  […]
  id: exercise2-vnf
  mgmt-cp: vnf-mgmt-ext
    […]
  vdu:
  - id: attackerVM
    int-cpd:
    - id: attackerVM-eth0-int
      […]
    sw-image-desc: ubuntu-focal
    virtual-compute-desc: VM-compute
    virtual-storage-desc: VM-storage
  - id: volumeVM
    int-cpd:
    - id: attackerVM-eth0-int
      […]
    sw-image-desc: ubuntu-focal
    virtual-compute-desc: VM-compute
    virtual-storage-desc: VM-storage
  version: '1.0'
  virtual-compute-desc:
  - id: VM-compute
    virtual-memory:
      size: 1.0
    virtual-cpu:
      num-virtual-cpu: 1
  virtual-storage-desc:
  - id: VM-storage
    size-of-storage: 10
```

Tabla 3: Descriptor abreviado del VNF del ejercicio 2

De este descriptor es interesante señalar que es aquí donde se definen las interfaces y los nombres que deberán ser usados como referencia en el nivel de NS. También se especifica la imagen que usarán las VDU, Ubuntu para los dos casos. Finalmente, se definen los recursos que usaran estas dos VDU. Como van a ser dos máquinas virtuales sencillas, se ha optado por otorgarles 1GB de RAM, 1 CPU y 10GB de memoria de almacenamiento.

8.1.2.3. VNF del ejercicio 3

Para el ejercicio 3 se usará un VNF de un único VDU. Sin embargo, este VNF es más complejo que el anterior, ya que cuenta con un *Juju-charm.* Lo cual, al contrario que en el ejercicio anterior, le permite realizar operaciones de día uno y dos. En la Tabla 4 se puede observar la línea 4 que corresponde a la *sección "lcm-operations-configuration"*, esta sección contiene el diseño del *Juju-charm* de este ejercicio. Sin embargo, para evitar la repetición, este diseño será explicado detalladamente en el apartado *8.3.3: Ejercicio 3: Juju-charm*.

```
1   vnfd:
2     […]
3     lcm-operations-configuration:
4     […]
5     id: exercise3-vnf
6     mgmt-cp: vnf-mgmt-ext
7     sw-image-desc:
8     - id: ubuntu
9       […]
10    vdu:
11    - cloud-init-file: cloud-config.txt
12      id: jujuVM
13      int-cpd:
14      - id: jujuVM-eth0-int
15        […]
16      sw-image-desc: ubuntu
17      virtual-compute-desc: jujuVM-compute
18      virtual-storage-desc: jujuVM-storage
19    virtual-compute-desc:
20    - id: jujuVM-compute
21      virtual-cpu:
22        num-virtual-cpu: 2
23      virtual-memory:
24        size: 4.0
25    virtual-storage-desc:
26    - id: jujuVM-storage
27      size-of-storage: 40
```

Tabla 4: Descriptor abreviado del VNF del ejercicio 3

Obviando la parte del diseño del *Juju-charm*, que ha sido omitido en la Tabla 4, el resto del descriptor es muy similar al de los apartados anteriores. La imagen que se utiliza también es de Ubuntu, pero se trata de un Ubuntu más sencillo, ya que delega las instalaciones a las primitivas del *Juju-charm*. Adicionalmente, al tener que ejecutar más programas que las máquinas virtuales anteriores, se le ha asignado más recursos, en concreto: 4GB de RAM, 2 GPU y 40GB de almacenamiento.

8.1.2.4. KNF del ejercicio 4

Por último, para el ejercicio 4 se utiliza un *Kubernetes Network Function* KNF en vez de un VNF ya que este ejercicio se instancia sobre Kubernetes en vez de sobre OpenStack. A efectos prácticos, bajo el punto de vista de un administrador de OSM, los dos tipos de descriptores son prácticamente idénticos y sólo les diferencia un pequeño cambio de terminología. En este caso, el descriptor es muy simple porque se utiliza un *Helm-chart* para que se encargue de la instalación del servicio.

```
1   vnfd:
2     […]
3     id: exercise4-knf
4     k8s-cluster:
5       nets:
6       - id: mgmtnet
7     kdu:
8     - name: helmCont
9       helm-chart: CyberRange-MT/helm-charts/hackssh
10    mgmt-cp: mgmt-ext
11    version: '1.0'
```

Tabla 5: Descriptor abreviado del KNF del ejercicio 4

En la Tabla 5 se puede observar el contenido del descriptor KNF. Este KNF está compuesto por un único *Kubernetes deployment unit* (KDU) al cual se le especifica el nombre y el repositorio donde se encuentra el Helm-chart que se debe instalar durante la instanciación. A diferencia de los casos anteriores, este descriptor no necesita que se le asigne los recursos para el contenedor, ya que esto lo realiza Kubernetes de forma automática.

8.1.3. Arquitectura del NS del servidor Web

Adicionalmente al NS previamente descrito, también se ha creado otro NS independiente para desplegar un servidor Web. Este servidor Web aloja la página Web que los usuarios necesitarán para poder validar sus puntuaciones y que el instructor utilizará para conocer las calificaciones de los usuarios. En este apartado se detallará el modo en el que se ha diseñado el NS y el VNF que sostienen este servicio, para ver la forma en la que se ha diseñado el servidor ver apartado *8.4: Verificación de las respuestas.*

Se ha optado por utilizar una VNF de OpenStack para alojar este servidor por múltiples propósitos. Uno de ellos es que, una vez se cuenta con un servidor OpenStack, es más cómodo desplegar servicios en una VNF que en un equipo físico. Otro motivo es que, de este modo, se cuenta con las ventajas de los sistemas NFV como despliegue de instancias a demanda, entre otros. En la Ilustración 8 se muestra de manera esquemática la forma en la que está compuesto el NS del servidor Web.

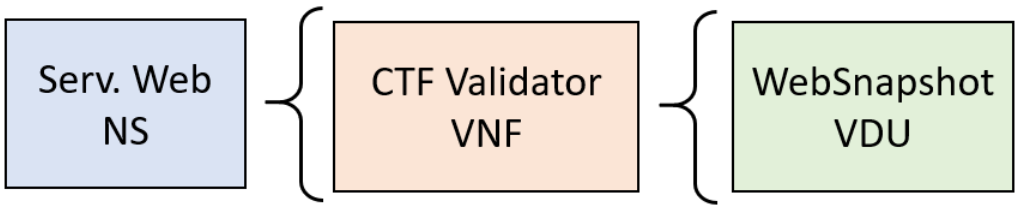

Ilustración 8: Esquema del Network Service que compone el servidor Web

A diferencia del escenario anterior, este NS es bastante simple. Cuenta con un único VNF, llamado *CTF Validator*. En la Tabla 6 se muestra el descriptor de este NS. Como en el caso anterior, este descriptor detalla el interfaz de red y definen la red a la que se conecta esta interfaz. A la hora de instanciar el NS, esta red es la misma que la del laboratorio. También tiene un campo que especifica el nombre del VNF que se instancia junto al NS.

```
nsd:
  […]
       virtual-link-connectivity:
       - constituent-cpd-id:
         - constituent-base-element-id: '1'
           constituent-cpd-id: vnf-mgmt-ext
         virtual-link-profile-id: mgmtnet
         vnfd-id: CTFvalidator-vnf
  id: webserver-ns
  version: '1.0'
  virtual-link-desc:
  - id: mgmtnet
    mgmt-network: true
  vnfd-id:
  - CTFvalidator -vnf
```

Tabla 6: Descriptor abreviado del NS del servidor Web

Este VNF, a su vez, cuenta con un único VDU, el cual se llama *Web Snapshot*. El descriptor de este VNF se muestra en la Tabla 7. La imagen que se utiliza para instanciar este servicio es una *Snapshot* que se ha creado con anterioridad de otra instancia. Una *Snapshot* es una instantánea de un momento determinado de una instancia, que se utiliza para luego poder desplegar esa misma instancia con exactamente los mismos datos y configuraciones. De esta forma, se ahorra al administrador el tener que modificar una imagen desde cero para tener una copia de una instancia personalizada. El proceso seguido en este caso, ha sido diseñar, desarrollar y desplegar un servidor web para luego crear una *Snapshot* y poder clonar tantos NS cómo se deseen.

```
vnfd:
  […]
  id: only1vm-vnf
  mgmt-cp: vnf-mgmt-ext
  sw-image-desc:
  - id: webserver_snapshot
    […]
  vdu:
  - id: webVM
    cloud-init-file: cloud-config.txt
    sw-image-desc: webserver_snapshot
    virtual-compute-desc: webVM-compute
    virtual-storage-desc: webVM-storage
    int-cpd:
    - id: mgmt-eth0-int
      […]
  virtual-compute-desc:
  - id: webVM-compute
    virtual-cpu:
      num-virtual-cpu: "2"
    virtual-memory:
      size: "4.0"
  virtual-storage-desc:
  - id: webVM-storage
    size-of-storage: "40"
```

Tabla 7: Descriptor abreviado del VNF del servidor Web

8.2. Descripción de los ejercicios del CTF

Los ejercicios diseñados hacen uso de muchos tipos de ataques. Al implementar ejercicios variados, se consigue que los usuarios participantes desarrollen competencias múltiples en el ámbito de la ciberseguridad. Sin embargo, aunque se ha diseñado ataques concretos para solucionar estos ejercicios, es importante darle al usuario la capacidad de probar otros métodos. Ya que, como suele pasar en todas las competiciones CTF, los ataques planteados quizás no sean los mismos que realicen todos los usuarios. Es habitual que los usuarios acaben realizando otro tipo de ataques que no habían sido considerado por los instructores y que sean igualmente válidos para completar los ejercicios. Por lo tanto, es importante remarcar que, aunque se han diseñado ejercicios vulnerables a algunos ataques, pueden que también sean vulnerables a otros que no se han considerado.

8.2.1. Tipos de ataques diseñados

A continuación, se describe al detalle la naturaleza de estos ataques diseñados, dividiéndolo en distintos tipos.

8.2.1.1. Robo de credenciales

Un robo de credenciales sucede cuando un usuario no autorizado logra credenciales de un usuario ajeno a él, por ejemplo, una contraseña. Una vez obtenida estas credenciales el atacante podría realizar varios ataques distintos, como un ataque de suplantación y hacerse pasar por el usuario atacado. Los motivos por los que se realizan este ataque pueden ser diversos, ya que, el atacante puede estar interesado en distintos objetivos. Este ataque puede ser usado para obtener los privilegios de otra persona en el sistema, para acceder a datos privados, etc.

En los ejercicios planteados se realizarán dos veces un ataque de robo de credenciales. Cada una de estas dos llevando un método diferente. Uno de estos ataques consistirá en robar las credenciales de inicio de sesión de un usuario en el servidor OpenStack. Para realizar este ataque deberá escuchar el tráfico sin cifrar que se transmite por la red. El otro ataque consistirá en hacerse con las credenciales de inicio de sesión de un usuario en una VNF de OpenStack. Esta VNF actúa como un equipo Linux y se deberá utilizar una herramienta que logre romper la contraseña mediante un ataque de fuerza bruta.

8.2.1.2. Man in the middle

Un ataque de *man in the middle* sucede cuando un atacante se cuela en la comunicación de dos equipos. De esta forma, tiene al alcance realizar dos ataques. Por un lado, puede limitarse a escuchar la comunicación de forma pasiva o, por otro lado, puede manipularla para que los demás equipos de la comunicación actúen como el atacante desea.

En los ejercicios planteados el atacante se aprovechará de las comunicaciones sin cifrar para hacerse con el token de autenticación que intercambian el servidor OSM y el servidor OpenStack. De esta forma, el atacante podrá hacerse pasar por el servidor OSM y obligar al servidor a realizar las operaciones que él desee. Se ha comprobado que el servidor OpenStack es muy vulnerable a este tipo de ataques, ya que, cuenta con una gran variedad de APIs que permiten a programas externos realizar múltiples operaciones. Por lo tanto, una vez el atacante se ha hecho con el token de autenticación puede realizar prácticamente cualquier operación en el servidor OpenStack, como si tuviera permisos de administrador.

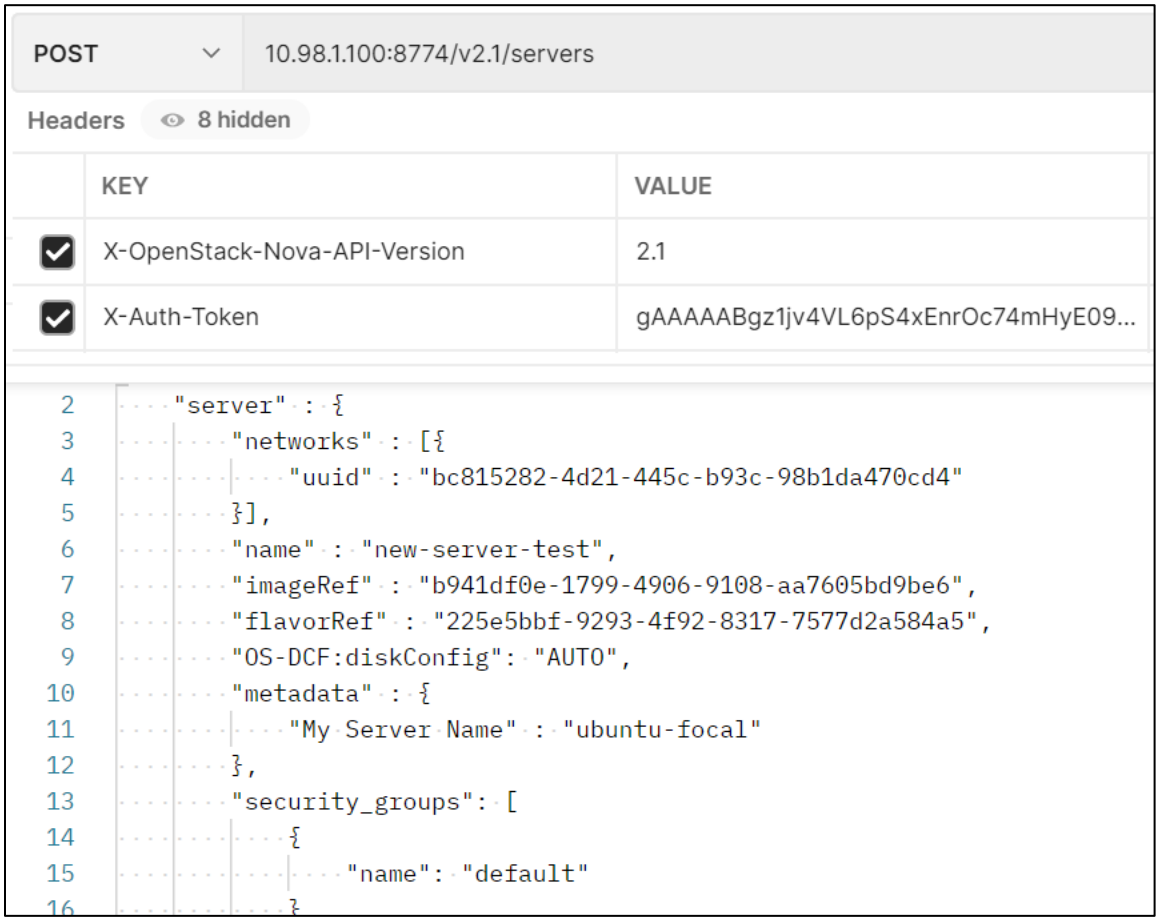

Ilustración 9: Petición HTTP para la creación de VNF

En la Ilustración 9 se muestra un ejemplo de la petición HTTP que puede realizar el atacante para crear una VNF en el servidor OpenStack. Como se puede observar, esta petición tiene varios campos. El *Path* contiene el destino de la petición HTTP. En la cabecera HTTP se encuentra un campo llamado "*X-Auth-Token*" el cual contiene una cadena de 183 caracteres. Este campo es el token de autenticación que usará el atacante para hacerse pasar por el servidor OSM y que ha obtenido previamente escuchando la conversación entre OpenStack y OSM. Por último, el campo *Body* contiene todos los datos para, en este caso, crear la VNF que desea el atacante.

8.2.1.3. Robo de datos

Los ataques de robo de datos consisten en la obtención de forma ilícita de información a la cual el atacante no debería tener acceso. Estos ataques se suelen apoyar en otros y pueden ser de múltiples naturalezas y contra múltiples objetivos. Estos tipos de ataques pueden ser discretos y pasar desapercibido o puede ser directos y evidentes para el usuario atacado.

En un ejercicio, se plantea el robo de datos de un fichero que se encuentra dentro de un volumen sin encriptar. Este volumen está adjuntado a una VNF ajena al atacante. Como este atacante no tiene acceso a esa VNF, deberá forzar el robo del volumen mediante un ataque de *man in the middle*. En este caso, el ataque diseñado sería evidente para el usuario atacado, ya que, el atacante hace que el volumen sea inaccesible en la VNF original.

8.2.1.4. Alteración de la ejecución

Este tipo de ataque tiene como objetivo la alteración del funcionamiento habitual del equipo objetivo. Esta alteración puede ser leve y cambiar el estado del equipo para introducir una vulnerabilidad o puede ser grave y hacer que el equipo cumpla con una función totalmente diferente a la que fue originalmente diseñada.

En los ejercicios diseñados, esta alteración del diseño original se debe a, principalmente, dos ataques. Por un lado, un ataque para la instalación de un software malicioso que el usuario objetivo no desea. Por otro lado, un ataque para la iniciación de servicios que introducen vulnerabilidades al equipo objetivo.

8.2.1.5. Infraestructura de red comprometida

Estos ataques tienen como objetivo la alteración de la infraestructura y buen funcionamiento de la red. Para conseguirlo, existen varios métodos, como, la creación de vulnerabilidades en equipos de la red. Este tipo de ataques suelen ser un paso previo en el objetivo de conseguir otros ataques, ya que, buscan manipular la red para obtener un beneficio.

En los ejercicios diseñados se realizará un ataque para la modificar el modo de funcionamiento de un equipo físico de la red (un *switch*) para poder escuchar las comunicaciones entre otros equipos de la red. También se realizará un ataque para cambiar las normas de los *security groups*, los cuales actúan como corta fuegos, y permitir que el tráfico no autorizado alcance equipos que no debería alcanzar.

8.2.1.6. Descubrimiento

Los ataques de descubrimiento tienen como objetivo la obtención de información sobre la red y los equipos que la componen. Busca obtener conocimientos sobre la topología de la red y sobre las características y servicios de los elementos de la red.

Los ataques diseñados buscan, por un lado, obtener información de la red física y de los equipos físicos. Por ejemplo, conocer los servicios que tienen disponibles cada uno de los equipos conectados a una red, mediante el mapeo de la red. Y, por otro lado, obtener información de las funciones virtualizadas y demás elementos virtuales de la plataforma cyber-range. Por ejemplo, obteniendo listas con información de las VNFs, volúmenes y *security groups* de OpenStack.

8.2.1.1. Escalada de privilegios

Los ataques de escalada de privilegios consisten en la obtención de unos privilegios que no deberían estar disponibles para el atacante. En otras palabras, sucede cuando un atacante gana acceso a lugares a los que debería tener un acceso restringido.

El ataque diseñado con este fin consiste en la obtención de acceso y permisos de administrador en el equipo anfitrión desde un equipo virtualizado de su interior. Mediante este ataque, el atacante no logra capacidad de ejecución en el equipo anfitrión, pero sí que logra capacidad de lectura y escritura en todos los ficheros del equipo anfitrión. Lo cual, en la práctica, es suficiente para comprometer todos los servicios y VNFs del anfitrión. Pudiendo, incluso, alterar otras VNFs del mismo anfitrión. Este tipo de ataques también puede ser considerado como un ataque de movimiento lateral, ya que desde un equipo se logra alterar otros equipos de la misma red.

8.2.2. Descripción de la implementación de ataques con herramientas de instalación

Debido a la mayor complejidad de este tipo de ataques, es interesante comentar la forma en que se ha implementado los ataques con Helm-chart y Juju-charm.

8.2.2.1. Helm-chart

Tal y como se menciona en el apartado *14.4.2: Helm-chart* del anexo, un Helm-chart es una herramienta de Kubernetes que facilita la instalación de servicios en el clúster. Estos Helm-charts serán utilizados para la ejecución de un ejercicio. Para el cual, se supone que se han desarrollado unos acontecimientos previos que justifican el estado de este Helm-chart. Los acontecimientos se detallan a continuación.

Un atacante con malas intenciones desarrolla un Helm-chart, el cual, en principio, instala un servidor de SSH en un clúster de Kubernetes. Sin embargo, cuenta con una vulnerabilidad que le permite realizar una escalada de privilegios. Este Helm-chart, aparte de instalar el servidor SSH, también crea un volumen mal configurado que permite al pod acceder a los directorios del equipo anfitrión con permisos de administrador. Esto sucede porque, al instalar este Helm-chart, se crea un volumen persistente con *Hostpath*, lo cual, permite al pod tener acceso de administrador a un directorio específico del anfitrión para leer y escribir sus datos. Normalmente, cuando se realizan este tipo de operaciones, se suelen realizar varias configuraciones para mantener la seguridad. Por ejemplo, establecer el acceso como sólo de lectura o utilizar un sistema de almacenamiento externo que no comprometa al anfitrión. Sin embargo, como el propósito de este Helm-chart es crear una vulnerabilidad, estas configuraciones no se realizan.

Una vez desarrollado este Helm-chart, el atacante lo carga en un repositorio con el objetivo de que un administrador de un clúster instale su charm. Suele ocurrir que un administrador inexperto o confiado instale a través de Helm-chart programas sin detenerse a investigar el *deployment* que lo compone. Bajo este supuesto, el administrador habría instalado, sin modificar, el Helm-chart del atacante.

El Helm-chart de este ejercicio en concreto está compuesto por los elementos que se pueden apreciar en la *Ilustración 10*. Siendo estos: un *configmap* que sirve como archivo de configuración, un *deployment* que indica los elementos a instanciar, un *service* que provee de conectividad a los elementos instanciados, un *volume* que sirve de almacenamiento y un archivo *values* que permite estandarizar y automatizar el proceso de instalación. Todos estos elementos son archivos yaml.

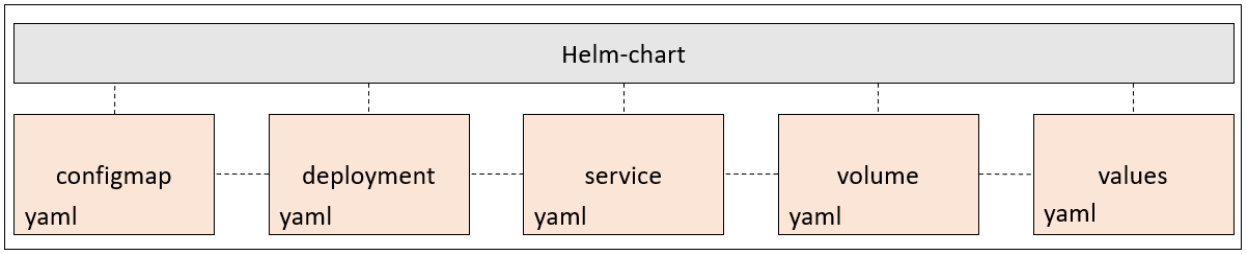

Ilustración 10: Diagrama de los elementos que componen el Helm-chart

A continuación, se detallan el contenido relevante de estos elementos para el ejercicio. Por un lado, los campos que se muestran en la *Tabla 8* indican al *deployment* que existe un volumen que debe ser instalado adicionalmente a los demás elementos del servicio. El nombre de este volumen se especifica en el archivo *values*.

```
[…]
- name: etc-storage
  persistentVolumeClaim:
   claimName: {{.Values.persistence.pvcname }}
[…]
```

Tabla 8: Instalación del volumen dentro del archivo deployment

Por otro lado, en la *Tabla 9* se muestra el contenido del fichero *volume*. Este fichero describe cómo se debe instalar y adjuntar un volumen de almacenamiento persistente. Este volumen se guarda en una dirección del equipo anfitrión. Por lo tanto, lo que se logra con este volumen es que un pod tenga acceso a una región del equipo anfitrión. Los detalles de la ubicación del volumen se especifican en el archivo *values*.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: {{ .Values.persistence.pvname }}
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: {{ .Values.persistence.hostPath }}
---------------------------------------------
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: {{ .Values.persistence.pvcname }}
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

Tabla 9: Contenido archivo volume

Por último, en la Tabla 10 se muestra el contenido relevante para la instalación del Helm-chart y para la ejecución del ejercicio. En el primer bloque se muestra la imagen que se instancia para crear el servicio SSH. En el segundo bloque se muestra la interfaz de red que tendrá el servicio, la cual es importante conocer para la ejecución del ejercicio. En el tercer bloque se muestra los datos necesarios para la creación del volumen, por ejemplo, la ubicación de este.

```
Image:
  name:sickp/alpine-sshd
  tag: latest
  pullPolicy: Always
```

```
Service:
  type: LoadBalancer
  port: 22
  extport: 32222
```

```
Persistence:
  enabled: true
  pvname: hackssh-etc-pv
  pvcname: hackssh-etc-pvc
  hostPath: "/etc"
  podPath: "/root/hostfolder"
```

Tabla 10: Contenido destacado del archivo values

Por lo tanto, la idea de este ataque es que el atacante puede acceder al pod con el usuario que se crea en la instalación del Helm-chart y utilizar la escalada de privilegios para comprometer a todos los equipos e instancias de la red. Mediante este ataque no se puede ejecutar ningún programa en el equipo anfitrión, sin embargo, se puede modificar programas o crear un cronjob que ejecute el programa por el atacante. En la práctica, el atacante gana el control total sobre el anfitrión. En los ejercicios se ha optado por un ataque más inofensivo y se ha diseñado para que con este ataque se gane información del anfitrión leyendo un archivo del directorio */etc*.

8.2.2.2. Juju-charm

Juju-charm es una herramienta similar a Helm-chart, en el sentido de que también permite la instalación de programas de forma automática. Sin embargo, es una herramienta más compleja y versátil que Helm-chart. Dado que está diseñada para ser utilizada en varios sistemas de virtualización, como OpenStack o Kubernetes, y porque permite operaciones de día 1 y 2.

Como se ha mencionado en el apartado *14.2 OSM: Open Source MANO* del anexo, las operaciones de día cero son las que se realizan junto con la instanciación de la imagen, las de día uno son las que se realizan de forma automática justo después de la instanciación y las de día dos son las que se realizan en cualquier momento posterior. Estas operaciones, también llamadas primitivas, son realizadas gracias a herramientas como Juju-charm. Con ellas, un desarrollador puede, entre otras opciones, diseñar al detalle qué servicios quiere instalar en una determinada instancia o que programas quiere ejecutar de forma automática. De esta forma, permite facilitar en gran medida la administración posterior de estas instancias, automatizando gran parte del proceso. El desarrollo de estas Juju-charms se realiza en el lenguaje de programación Python.

En el ejercicio propuesto, se utilizará un Juju-charm al que un desarrollador ha decidido concederle demasiada libertad de ejecución. La razón de esto es la que se expone a continuación. Se parte del supuesto de que un desarrollador ha decidido automatizar la creación de una instancia para un servidor web. Para ello, parte de una imagen de Ubuntu y, mediante un Juju-charm, hace que como operación de día uno se instale e inicie un servicio de *apache2*. El desarrollador, decide mantener la posibilidad de realizar estas operaciones como primitivas de día dos, por si en un futuro necesitara instalar otro servicio de forma rápida y sencilla. Por lo tanto, está permitiendo que como operaciones de día uno y dos se pueda instalar e iniciar cualquier servicio.

Bajo esta premisa, el atacante del ejercicio deberá acceder al servidor OSM e instalar el software que considere oportuno aprovechándose de esta vulnerabilidad, que el propio desarrollador ha introducido sin ninguna mala intención. Ante este ataque, el administrador no puede hacer realmente nada para evitarlo, aparte de eliminar la instancia y crear otra con un Juju-charm más seguro.

En la *Ilustración 11* se muestran los elementos que componen el Juju-charm que se ha desarrollado para este ejercicio. Estos elementos son: un archivo *actions* que lista todas las primitivas que puede ejecutar el Juju-charm, un archivo de configuración, un archivo con metadatos del charm, un programa en Python que implementa la ejecución de las primitivas y varios enlaces simbólicos y archivo de operación.

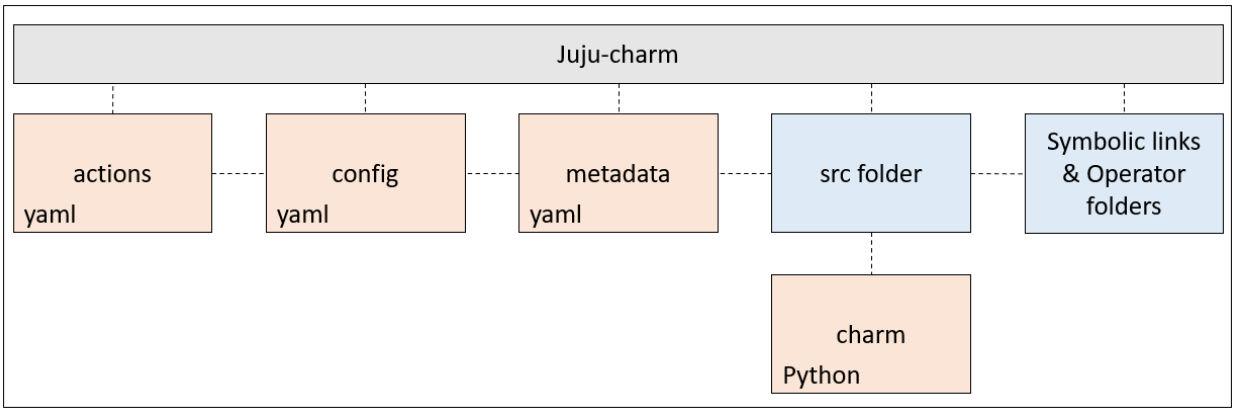

Ilustración 11: Diagrama de los elementos que componen el Juju-charm

A pesar de implementarse en el programa Python, es más sencillo de ver los detalles de estas primitivas en el archivo *actions*. Por lo tanto, en la *Tabla 11* se muestra el contenido relevante del archivo *actions*. Como se puede observar, las dos primitivas tienen una cadena de carácter como variables de entrada. Esto es importante ya que el atacante se aprovechará de esta capacidad para instalar e iniciar determinados servicios. Aparte de estas primitivas, también se ha desarrollado otras para crear archivos y para detener servicios. Sin embargo, estas primitivas no son utilizadas en los ejercicios planteados, por lo que no se incluyen en la *Tabla 11.*

```
1  […]
2  start-service:
3    description: "starts the service of the vnf"
4    params:
5      servicename:
6        description: "The name of the service to start."
7        type: string
8        default: ""
9    required:
10     - servicename
11 install-service:
12   description: "install a service on the vnf"
13   params:
14     servicename:
15       description: "The name of the service to install."
16       type: string
17       default: ""
18   required:
19     - servicename
20 […]
```

Tabla 11: Contenido relevante del archivo actions

8.3. Ejecución de la plataforma cyber-range

En este apartado se detalla la forma de resolver los ejercicios, paso a paso, desde el punto de vista del usuario de la competición CTF, al cual se le referirá como *atacante*. Cabe señalar que este es el proceso de ataques diseñado, pero que es posible que existan otros métodos no considerados para llegar a los mismos resultados. Debido a esto, se considera que cualquier otro método que consiga el objetivo es igualmente válido al planteado en el diseño de la solución. El objetivo de todos los ejercicios es obtener una o varias cadenas de caracteres. Estas cadenas varían en longitud y carácter dependiendo de los ejercicios.

8.3.1. Ejercicio 1: Credenciales

El primer ejercicio consiste en atacar la infraestructura de red para obtener una vulnerabilidad y utilizarla para atacar el servidor OSM. El objetivo del ataque es obtener las credenciales de inicio de sesión del administrador del servidor OSM. Para obtener estas credenciales el atacante se deberá aprovecharse de que el inicio de sesión se realiza con el protocolo HTTP sin cifrar. Por lo tanto, como las comunicaciones no se realizan sobre una comunicación segura, como HTTPS, es posible para el atacante leer con facilidad el contenido de los datagramas transmitidos.

El primer paso que deberá realizar el atacante es acceder al switch, que se encuentra entre el servidor OSM y el resto de la red. En la Ilustración 12 se específica la localización de este switch dentro de la topología de la red. Una vez ha accedido al switch, deberá configurarlo para que redirija todo el tráfico del servidor OSM por un puerto al que tiene un equipo conectado. De esta forma puede escuchar todas las comunicaciones que se realizan entre el servidor OSM y cualquier otro equipo.

Periódicamente, el administrador realizará un inicio de sesión ante el servidor OSM. Por lo tanto, el segundo paso es realizar una captura del tráfico redirigido durante un período de tiempo. Durante ese periodo, el atacante podrá capturar el datagrama HTTP que contiene las credenciales de inicio se sesión. En la Ilustración 12 se muestra un ejemplo de esta captura realizada con la herramienta Wireshark. Estas credenciales son la bandera que se busca para completar el ejercicio.

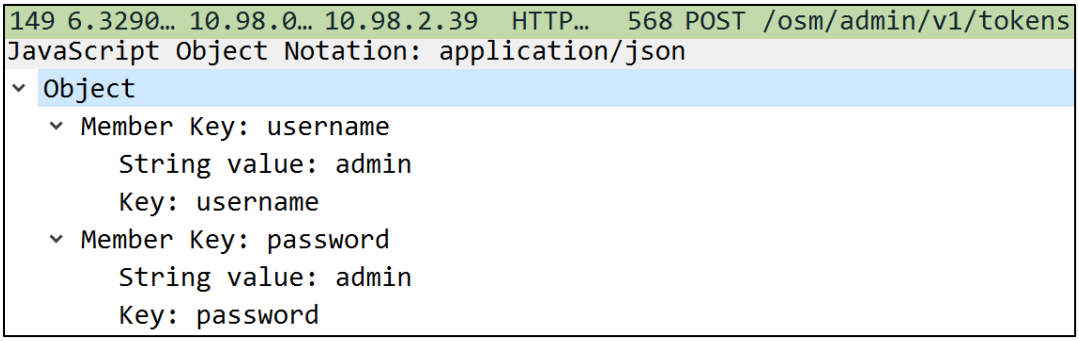

Ilustración 12: Mensaje inicio de sesión OSM

8.3.2. *Ejercicio 2: Volúmenes*

El segundo ejercicio consiste en aprovecharse del ataque realizado en el primer ejercicio para atacar el servidor OpenStack y sus instancias. El objetivo de este ejercicio es robar la información que contiene un volumen que se encuentra adjunto a una instancia en OpenStack. La información de este volumen no se encuentra cifrada.

El primer paso de este ejercicio es repetir la escucha de las comunicaciones de OSM. En este caso se debe capturar los datagramas que se envían regularmente entre el servidor OSM y el OpenStack. En estos datagramas se envía sin cifrar el token de autenticación que autentica ese mensaje cómo un mensaje de un usuario legítimo. Este token consiste en una cadena de caracteres que se envía en múltiples datagramas y que cambia cada pocos minutos, por lo tanto, es importante realizar una escucha cada pocos minutos para obtener el token actualizado cada vez. En la Ilustración 13 se muestra una captura de este mensaje, realizada con la herramienta Wireshark.

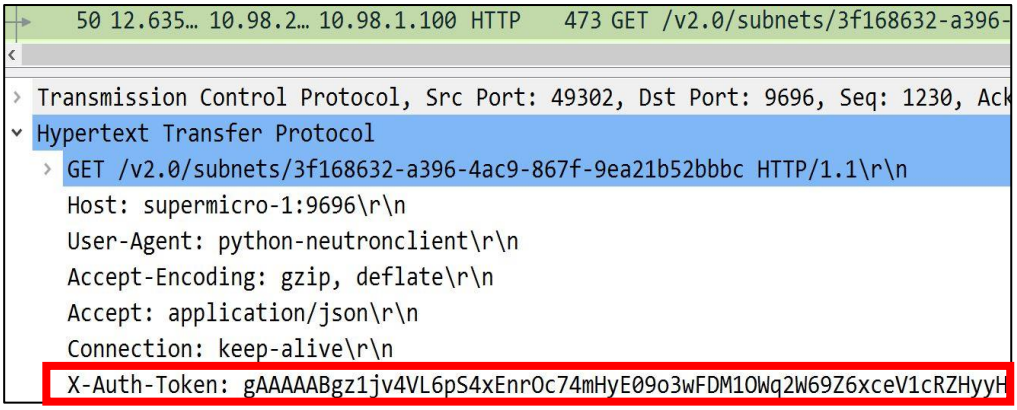

Ilustración 13: Mensaje con el token de autenticación

Una vez obtenido el token de autenticación, el siguiente paso es obtener información acerca de las instancias y los volúmenes que se encuentran desplegados en el servidor OpenStack. Esta información se puede obtener atacando la API de OpenStack. Para ello, se debe usurpar la identidad del servidor OSM y enviar un datagrama con el token de autenticación junto con un comando con la orden a ejecutar por el servidor OpenStack. Por lo tanto, para conseguir el objetivo propuesto, el atacante debe enviar dos datagramas, uno con el comando para listar las instancias y otro con el comando para listar los volúmenes. Una vez el servidor OpenStack haya recibido los datagramas, responderá a cada uno de ellos con la respectiva lista que se solicita.

Cuando el atacante se ha hecho con estos dos listados, obtiene una información importante para poder ejecutar la última parte de este ejercicio. Para poder proseguir con el ejercicio, primero necesitará analizar los datos para saber qué instancia cuenta con un volumen adjunto y, posteriormente, conocer el identificador de dicho volumen e instancia. Tras analizar estos datos, el atacante observa que no tiene acceso a la instancia del volumen, por lo tanto, debe realizar los siguientes ataques.

Primero, el atacante debe enviar un datagrama al servidor OpenStack con un comando de "*detachment*" del volumen de la instancia original. En la Ilustración 14 se muestra el contenido de este mensaje, el cual contiene el identificador de la instancia, el comando *DELETE* y el campo *os-volume_attachments*. Una vez realizado este ataque, el volumen ya no está adjunto a la instancia y queda libre para adjuntarse a cualquier otro volumen. Cuando se realiza esta operación, el volumen no se elimina ni se pierde ninguno de los datos que contiene.

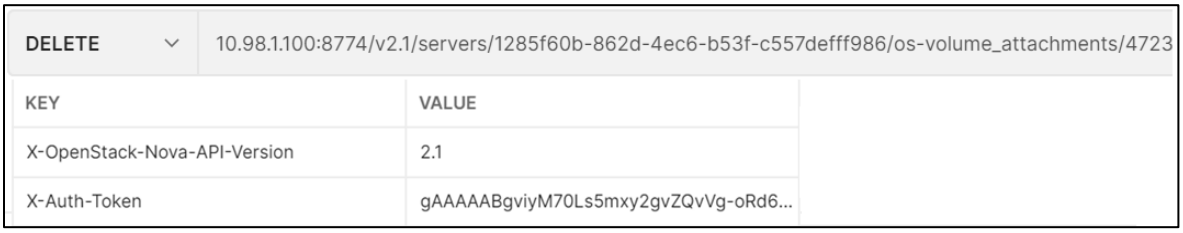

Ilustración 14: Mensaje de "detachment" del volumen

A continuación, el atacante deberá recurrir a la lista de las instancias que ya había obtenido y buscar el indicador de una instancia a la que sí que tiene acceso. Una vez encontrado, el atacante deberá enviar un nuevo datagrama. Esta vez con el token de autenticación, el identificador del volumen, el identificador de la instancia a la que tiene acceso y el comando de adjuntar dichos elementos. En la Ilustración 15 se muestra el contenido de este mensaje.

Ilustración 15: Mensaje de "atachment" del volumen

El último paso de este ejercicio es montar el volumen adjuntado en la nueva instancia a la que el atacante tiene acceso y obtener la información que contiene un fichero de su interior. La cadena de caracteres que se encuentra en dicho fichero es la bandera que se busca en este ejercicio. En la Ilustración 16 se muestra el proceso para realizar esta acción.

Ilustración 16: Comandos para el robo del volumen

8.3.3. *Ejercicio 3: Juju-charm*

El tercer ejercicio trata sobre explotar una vulnerabilidad del servidor OSM para poder atacar las instancias que se encuentran en el servidor OpenStack. El objetivo de este ejercicio es obtener las credenciales de inicio de sesión de una instancia. Para se hará uso de un Juju-charm.

Los primeros dos ataques de este ejercicio son similares a los del ejercicio anterior. Primero, se debe escuchar las comunicaciones entre el servidor OSM y el servidor OpenStack para obtener el token de autenticación. Después, solicitar un listado actualizado de las instancias desplegadas.

Durante el diseño del escenario, se ha creado un *security group* que actúa como cortafuegos de la instancia y que complica la resolución del escenario. Este *security group* restringe la casi totalidad el tráfico entrante y saliente de la instancia. Por lo tanto, el atacante debe darse cuenta de esta configuración y solicitar un listado con la información de los *security groups*. Una vez tiene esta información, debe obtener el identificador del *security group* y enviar un mensaje para crear una nueva regla que, al menos, permita el tráfico del protocolo SSH en el puerto 22. Para este ataque también valdría permitir todos los tipos de tráfico en todos los puertos. En la Ilustración 17 se muestra un ejemplo del mensaje que se debe enviar para crear la nueva regla, se puede observar que contiene el identificador del *security group* y una descripción de las características del tráfico a permitir.

POST 10.98.1.100:8774/v2.1/os-security-group-rules

```
{
    "security_group_rule": {
        "parent_group_id": "b7e15ae7-66f0-4b4a-a5ed-959176f0a25f",
        "ip_protocol": "tcp",
        "from_port": 22,
        "to_port": 22,
        "cidr": "0.0.0.0/0"
    }
}
```

Ilustración 17: Mensaje de modificación de los Security Groups

Una vez se ha solucionado este impedimento, se puede proseguir con el ejercicio. Para ello, se debe hacer uso de las credenciales de inicio de sesión del servidor OSM que se han obtenido en el primer ejercicio. Si el atacante no ha completado dicho ejercicio, debe conseguir ahora las credenciales. Con estas credenciales debe iniciar sesión en el servidor OSM, acceder a una VNF y ejecutar unas primitivas de juju-charm.

Como ya se ha mencionado en el apartado *8.2.2.2: Juju-charm*, un juju-charm permite ejecutar primitivas para realizar operaciones de día uno y dos. En este caso, el juju-charm está diseñado para ejecutar dos primitivas en concreto: instalar un programa e iniciar un servicio. El atacante debe aprovecharse de las vulnerabilidades que introduce esta herramienta para instalar un servidor SSH e iniciar el servicio. En la Ilustración 18 se muestra un ejemplo de cómo se instala el servidor SSH utilizando las primitivas mencionadas.

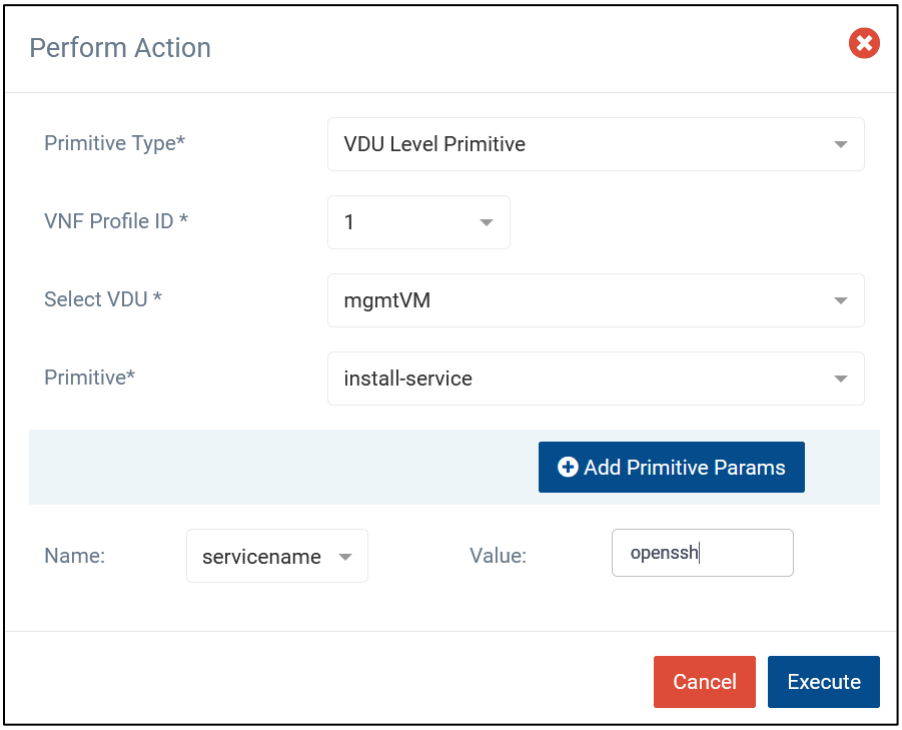

Ilustración 18: Uso Juju-charm desde servidor OSM

Si el atacante logra ejecutar las primitivas, habrá conseguido manipular la VNF, instalando y ejecutando software no deseado por el administrador. El objetivo de esta manipulación es posibilitar el uso de herramientas que necesitan el protocolo SSH para realizar ataques de adivinación de contraseñas. Por lo tanto, como último paso del ejercicio, el atacante debe explotar esta nueva vulnerabilidad introducida para obtener las credenciales de inicio de sesión de la instancia. Para ello el atacante puede utilizar herramientas como *hydra o patator*. Estas herramientas permiten realizar ataques de fuerza bruta contra objetivo que tienen un servidor SSH. Para facilitar un poco el proceso, se les entregará a los usuarios unos archivos con una lista de combinaciones de usuarios y contraseñas. De esta forma los atacantes podrán realizar ataques de biblioteca en vez de ataques de fuerza bruta. En la Ilustración 19 se muestra un ejemplo de este proceso. El usuario y contraseña que se obtiene con esta operación son la bandera que se necesita para completar el ejercicio.

```
┌──(kali㉿kali)-[~/ficherosHydra]
└─$ patator ssh_login host=10.98.1.106 user=FILE1 password=FILE0 0=pwd_comb.txt 1=usrfile.txt -x ignore:mesg='Authentication failed.'
06:46:08 patator    INFO - Starting Patator 0.9 (https://github.com/lanjelot/patator) with python-3.9.1 at 2021-07-01 06:46 EDT
06:46:08 patator    INFO -
06:46:08 patator    INFO - code  size    time | candidate                          |   num | mesg
06:46:08 patator    INFO - -----------------------------------------------------------------------------
06:46:12 patator    INFO - 0     39     0.057  osm4u:ubuntu                       |    12 | SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.1
06:47:59 patator    INFO - Hits/Done/Skip/Fail/Size: 1/325/0/0/325, Avg: 2 r/s, Time: 0h 1m 50s
```

Ilustración 19: Resultado ataque fuerza bruta

8.3.4. *Ejercicio 4: Helm-chart*

El último ejercicio tiene como objetivo realizar una escalada de privilegios y obtener información del equipo anfitrión. Es decir, este ejercicio consiste en que el atacante ataque el clúster de Kubernetes para obtener el nombre de los usuarios del equipo anfitrión que sostiene el propio clúster de Kubernetes. Para realizar este ataque, tal y como se ha mencionado en el apartado *8.2.2.1: Helm-chart*, se parte del hecho de que el administrador del clúster ha instalado un Helm-chart con unas vulnerabilidades específicas. Aunque el ejercicio no lo contempla por motivos prácticos, se supone que este Helm-chart ha sido diseñado por el atacante para lograr una puerta trasera en un pod del clúster. Por lo tanto, al atacante se le informa de las características de este Helm-chart y parte con el conocimiento que este Helm-chart abre el puerto 32222. Haciendo uso de este conocimiento, debe empezar el ejercicio realizando un ataque de descubrimiento de la red para saber qué equipo tiene abierto dicho puerto. En la Ilustración 20 se muestra un ejemplo abreviado de este proceso de descubrimiento.

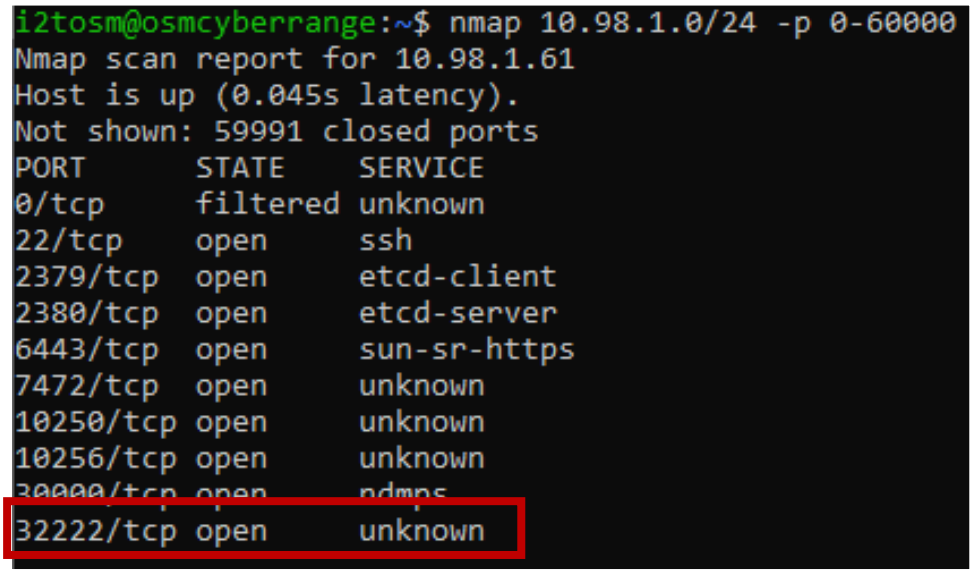

Ilustración 20: Resultado de mapeado de la red

Una vez descubierta la localización y la identidad del pod vulnerable, el atacante puede acceder a él. Esto se debe a que el Helm-chart instala un usuario con una contraseña por defecto y, como se parte de que el administrador no ha modificado el Helm-chart, el atacante puede usar este usuario para entrar con facilidad.

Una vez se ha accedido al pod, el atacante accede a la carpeta donde se encuentra la vulnerabilidad. Esta carpeta en realidad se trata de un volumen virtual que redirecciona al atacante a una dirección del anfitrión del clúster de Kubernetes. Como el atacante tiene permisos de administrador en el pod, en esta dirección del anfitrión también tiene permisos de administrador. En caso del ejercicio, el volumen virtual redirige a la carpeta "*/etc*" del equipo anfitrión. En la Ilustración 21 se muestra un ejemplo del proceso seguido por el atacante.

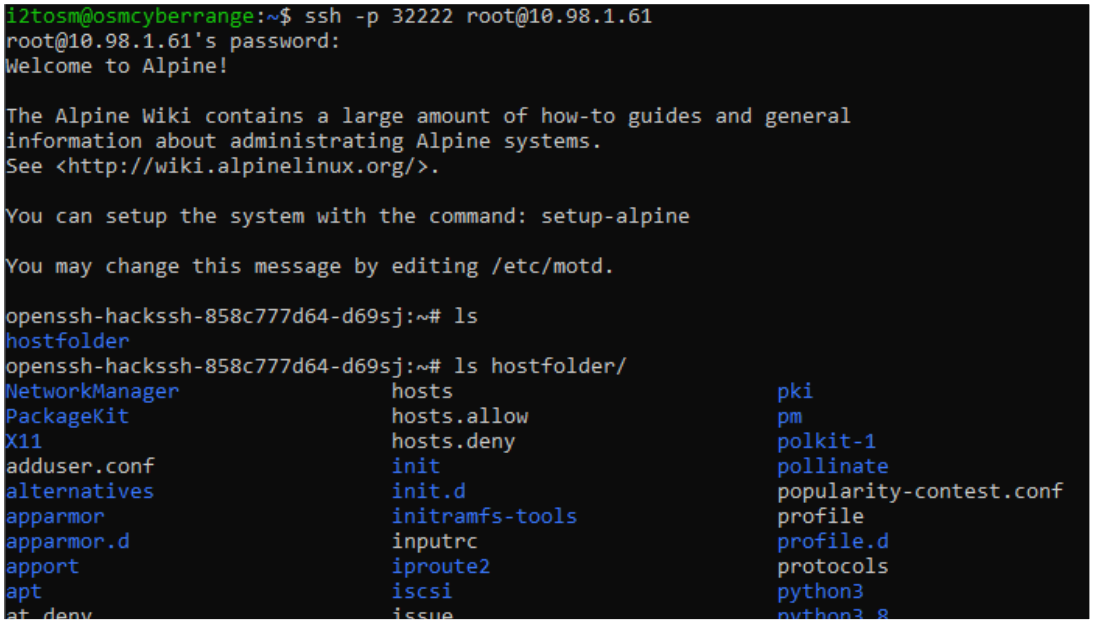

Ilustración 21: Escalada de privilegios

Una vez realizado todo lo anterior, el atacante debe leer el fichero *"/passwd"* del equipo anfitrión para buscar un usuario que empieza con *"flag-"*. El nombre de este usuario es la bandera que se necesita para completar el ejercicio. En la Ilustración 22 se muestra el contenido de este fichero junto con el usuario mencionado.

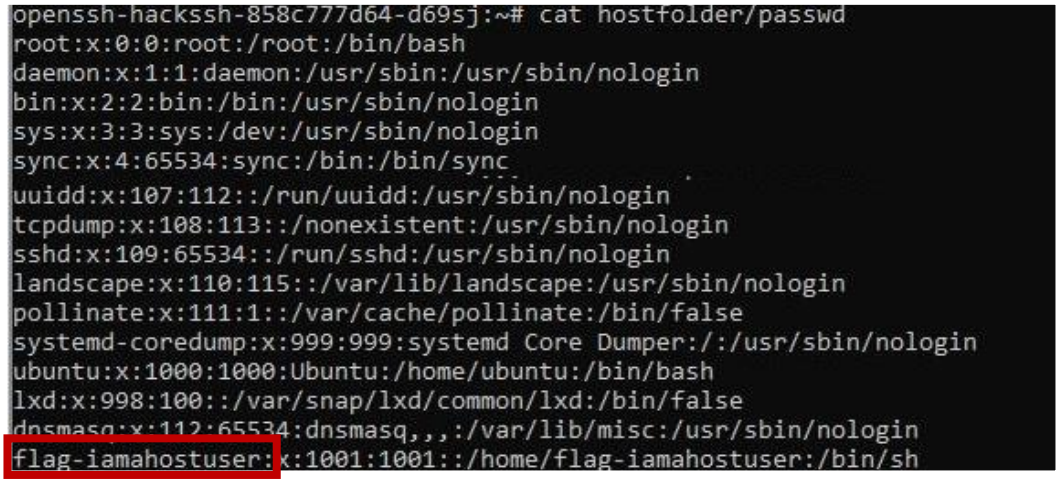

Ilustración 22: Obtención de información del host

8.4. Verificación de las respuestas

Para el sistema de verificación de las respuestas se ha diseñado un servidor web. Este servidor permite a los usuarios conocer los enunciados de los ejercicios y validar sus respuestas. Al mismo tiempo, permite que el instructor del ejercicio conozca las puntuaciones de todos los usuarios que realizan los ejercicios. En los siguientes apartados se describe en mayor detalle todos los elementos y características de este servidor.

8.4.1. Arquitectura del servidor Web

El servidor es un proyecto de Web dinámico que se ha desplegado sobre un servidor *Tomcat*. Se ha configurado el servidor para que use el protocolo HTTPS para mantener las comunicaciones seguras. En cuanto a su diseño, el servidor sigue una arquitectura MVC (Modelo, Vista y Controlador). Según esta arquitectura, el programa se divide en tres planos separados. En la Ilustración 23 se muestra cómo estos elementos se dividen entre estas tres categorías.

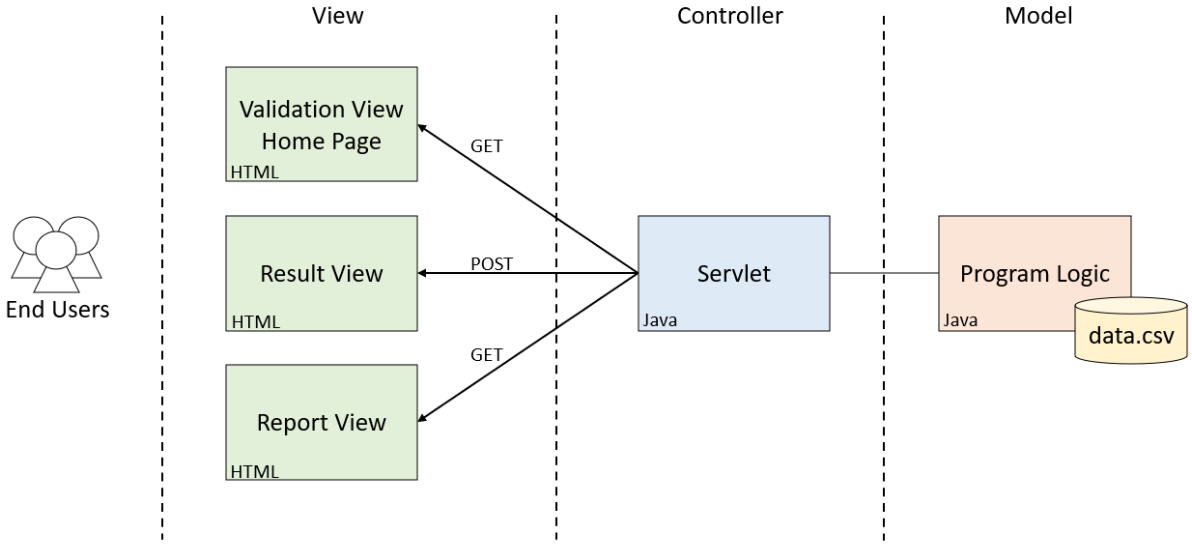

Ilustración 23: Arquitectura del software del servidor Web

Por un lado, el modelo incorpora la lógica del programa. Está escrito en el lenguaje de programación Java. Se encarga de ejecutar cualquier algoritmo y cualquier procesamiento necesario. Es aquí donde, por ejemplo, se calcula la puntuación final que ha conseguido cada usuario. También es el encargado de guardar y gestionar los datos de los usuarios. Para este proyecto, el modelo guarda y lee los datos de los usuarios en un archivo CSV.

Por otro lado, el controlador es el intermediario entre la vista y el modelo. Responde a las peticiones que el usuario final hace desde la vista e invoca al modelo cuando es necesario realizar algún procesamiento. En este proyecto se ha utilizado un Servlet de Java para realizar la tarea del controlador.

Por último, la vista se encarga de representar todos los datos a los usuarios finales. Este plano cuenta con las páginas web que se representan en el navegador web de los usuarios. Están escritas en el lenguaje de programación HTML, pero utilizan Java para hacerlas dinámicas. De esta forma, varían en función de los datos a representar. En los próximos apartados se detallarán el contenido de estas tres páginas web.

8.4.2. Página de evaluación

La página de evaluación es la página principal del servidor Web. Empieza con la cabecera que se muestra en la Ilustración 24. Esta cabecera contiene el logotipo, unos enlaces rápidos a los ejercicios y otros apartados, el título del servidor web y una descripción de la competición CTF.

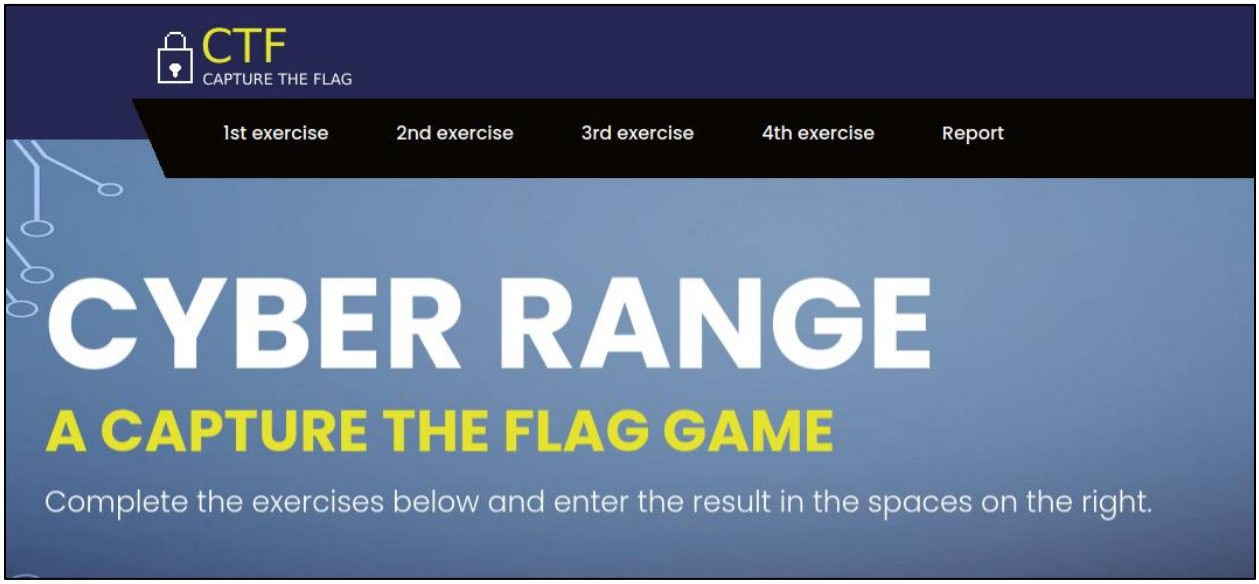

Ilustración 24: Página principal

Después de la cabecera, hay una sección para que los usuarios de la competición se registren dentro del sistema. Para realizar esta acción el usuario únicamente debe introducir un nombre que lo identifique. En la Ilustración 25 se muestra la sección donde los usuarios se registran.

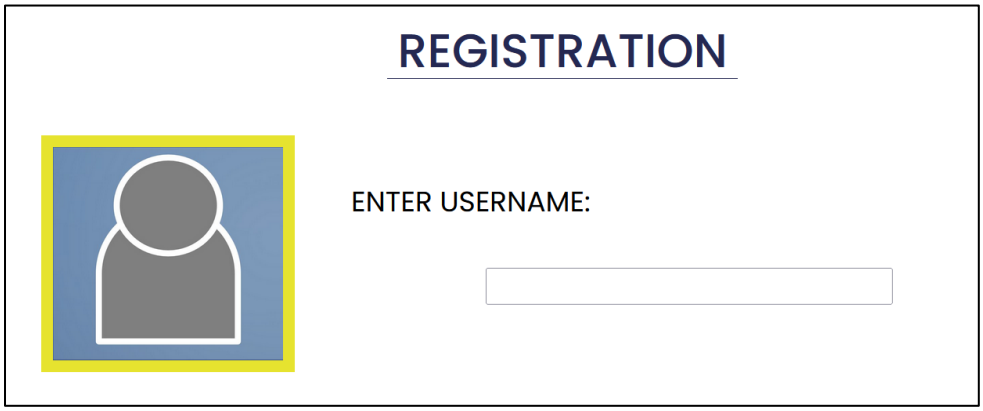

Ilustración 25: Registro de usuarios

Tras registrarse, los usuarios verán los ejercicios de la competición CTF. Cada uno de estos ejercicios cuenta con un título y un pequeño enunciado que da indicaciones generales de cómo realizar el ejercicio. Es costumbre en estos tipos de competiciones, el ofrecer a los usuarios la mínima cantidad de información posible. De esta forma, se logra que los usuarios realicen los ejercicios de maneras imaginativas y diferentes.

Junto al enunciado se encuentra uno o varios campos para introducir las banderas que se hayan descubierto al realizar el ejercicio. Estas banderas son los resultados de todos los ataques que se realizan y es el objetivo que debe perseguir el usuario.

En total, hay cuatro ejercicios distintos. En la Ilustración 26 se muestra, como ejemplo, el primero de estos ejercicios.

Ilustración 26: Enunciado de los ejercicios

Al final de todas las páginas web del servidor, tras las demás secciones, se encuentra el pie de página. Esta sección cuenta con otra pequeña descripción, el logotipo y los mismos enlaces rápidos que se muestran en la cabecera. En la Ilustración 27 se muestra este pie de página para el caso de la página de evaluación.

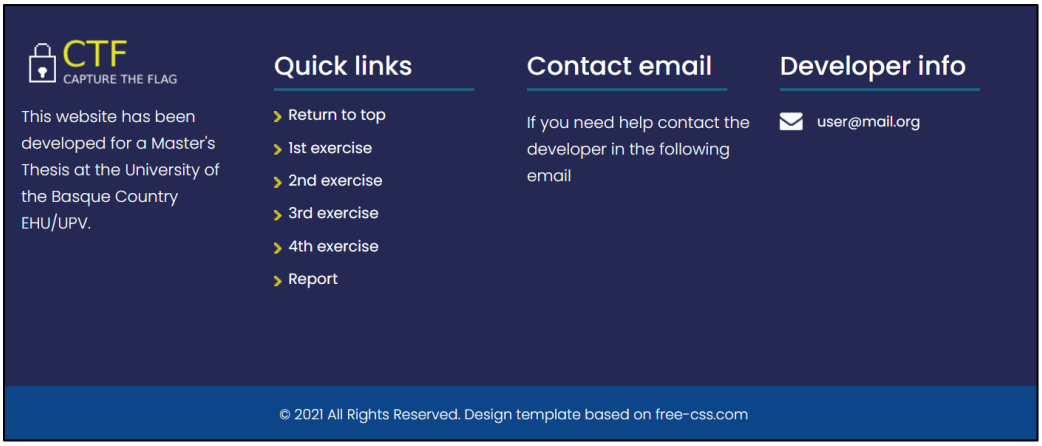

Ilustración 27: Pie de página de las páginas del servidor web

8.4.3. Página de resultados del usuario

En esta página se muestran los resultados que ha obtenido el usuario. Para acceder a ella los usuarios deben completar los ejercicios y enviar los resultados al servidor. Tras esto, el servidor procesa los resultados obtenidos y devuelve esta página como respuesta. En la página, tal y como se muestra en la Ilustración 26, se representa el nombre del usuario y la puntuación obtenida. Como hay cuatro ejercicios, los resultados se muestran sobre cuatro. Acompañando a la calificación numérica también se muestra la calificación en texto ("*Fail", "Poor", "Pass", "Good" y "Excelent*"). A esta página web solamente tiene acceso el usuario una vez ha enviado sus datos.

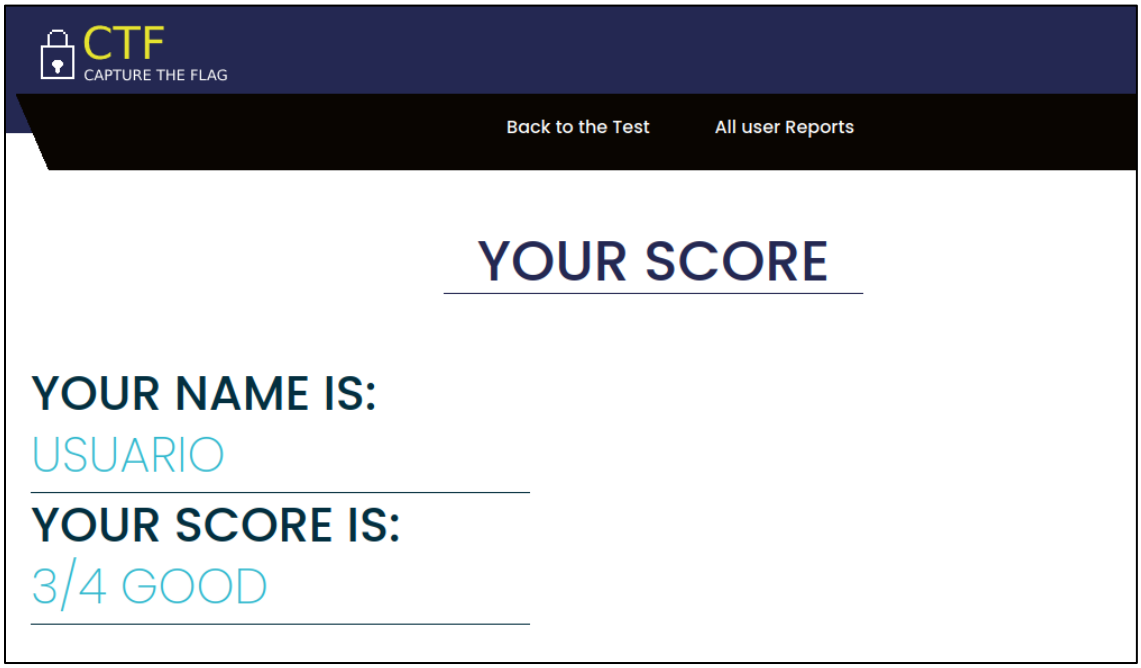

Ilustración 28: Resultados de un usuario

8.4.4. Página del informe general

Esta página web muestra un resumen del desempeño de cada usuario. Lista a todos los usuarios que han participado en la competición y muestra su puntuación total obtenida junto con la puntuación de cada ejercicio. En la Ilustración 29 se muestra esta página web. Como se ha mencionado, el resultado total es sobre cuatro ya que hay cuatro ejercicios. Adicionalmente, en el desglose, los ejercicios muestran un uno cuando son correctos y un cero cuando son incorrectos.

En esta página web los usuarios podrán ver sus calificaciones y compararlas con las de sus compañeros. Sin embargo, esta página web es especialmente útil para que el instructor de la competición CTF pueda evaluar a todos los usuarios que han participado.

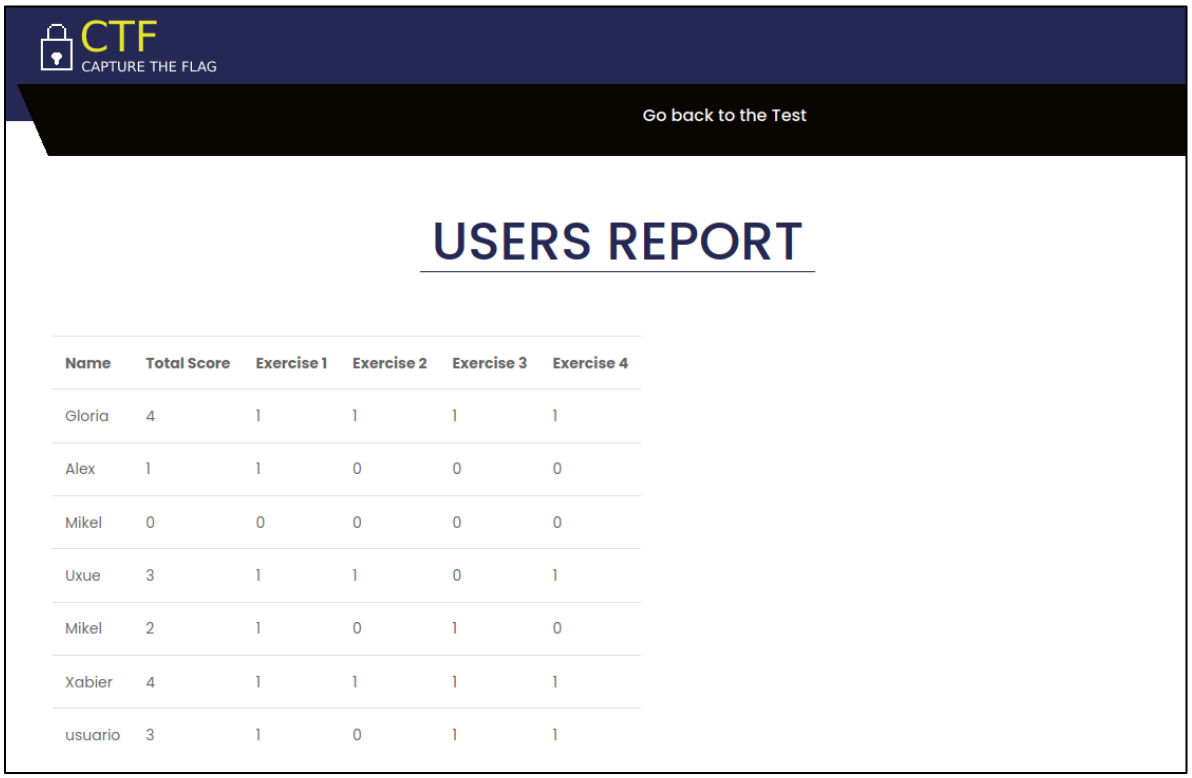

Ilustración 29: Informe de todos los usuarios

9. DESCRIPCIÓN DE FASES Y TAREAS

De acuerdo con la planificación inicial, el proceso de desarrollo de este proyecto puede ser dividido en seis fases diferentes. La primera fase es la del despliegue de la infraestructura e instalación del software requerido para el resto del proyecto. La segunda fase es el período de formación de los entornos previamente desplegados e instalados. La tercera fase corresponde con el diseño de la cyber-range y de los ejercicios de la competición CTF. La cuarta fase es la de la instalación y despliegue de todos los elementos previamente diseñados. La quinta fase es la implementación de un sistema de evaluación. Por último, la fase de gestión del proyecto es una fase que se realiza durante el transcurso de todo el trabajo.

A continuación, se exponen estas fases con su correspondientes de forma más detallada. Cada una de estas fases son representadas en sus correspondientes fechas, según la planificación, en el apartado *10: DIAGRAMA DE GANTT*.

9.1. Despliegue de la infraestructura y el software inicial

Esta fase consiste en la instalación y despliegue de todos los elementos que serán necesarios para realizar el proyecto. Se puede dividir en dos tareas diferentes. Por un lado, la instalación de los sistemas de virtualización. Por otro lado, el despliegue de la infraestructura de red.

9.1.1. Despliegue de infraestructura de red

La primera tarea consiste en el despliegue de infraestructura de red física. Esta infraestructura es necesaria para desplegar sobre ella los distintos elementos que componen la cyber-range. Adicionalmente, también es necesaria para garantizar la conectividad de todos los equipos y para realizar ciertos ejercicios de la competición CTF.

9.1.2. Instalación sistemas de virtualización

La segunda tarea consiste en la instalación del software de virtualización y de sus correspondientes dependencias. Los principales softwares a instalar son el orquestador OSM y el sistema de contenedores Kubernetes. Los dos sistemas son de código abierto y, por lo tanto, hay extensas indicaciones de cómo instalarlos en sus correspondientes documentaciones [17] [14]. El sistema de máquinas virtuales OpenStack no será necesario instalar porque se utilizará el que ya se encuentra disponible en la red de producción.

9.2. Formación

Después de terminar con la fase de instalación y despliegue de todos los elementos necesarios para la realización del proyecto, es necesario invertir tiempo y recursos en aprender a utilizar dichos elementos. Esta fase puede ser dividida en tres tareas, una tarea por cada sistema de virtualización.

9.2.1. Formación OSM

Esta tarea consiste en comprender las características del principal software del proyecto. Es importante saber utilizar al detalle la plataforma OSM porque actúa como el orquestador de todos los demás sistemas y, por lo tanto, controla todos los servicios de la plataforma cyber-range. Adicionalmente, es importante conocer bien este software debido a que es el que utilizará el administrador para desplegar todos los servicios del proyecto.

Parte de esta tarea corresponde al estudio de la iniciativa NFV de la ETSI. De esta forma, se parte de la comprensión del marco global al que pertenecen todas las plataformas del proyecto. Una vez terminado con esta fase previa, la tarea deriva a la forma en la que OSM adapta las especificaciones del marco global a su propia solución. Por último, se estudia la forma en la que se componen los descriptores que conforman las funciones de red y la forma de desplegarlos. Comprender estos descriptores es necesario para poder crear otros nuevos que cumplan con los requerimientos de cada uno de los servicios a instanciar.

9.2.2. Formación OpenStack

Esta tarea consiste en comprender el funcionamiento del software OpenStack. Este es un elemento relevante dentro del proyecto, ya que varios de los ejercicios diseñados son operaciones que se realizan contra él. OpenStack se encarga de realizar las funciones del VIM dentro de la arquitectura de la iniciativa NFV de la ETSI y, por lo tanto, se encarga de gestionar las máquinas virtuales del proyecto.

Aparte de la arquitectura del software, hay varios elementos de OpenStack que son cruciales para los ejercicios y que tienen que ser entendidos. Estos elementos, entre otros, son los *security groups*, los *tenants*, las imágenes, las *snapshots*, etc.

9.2.3. Formación Kubernetes

Esta tarea consiste en comprender el funcionamiento del software Kubernetes. Al igual que OpenStack, también se encarga de realizar las funciones del VIM dentro de la arquitectura de la iniciativa NFV de la ETSI. Sin embargo, en vez de encargarse de gestionar las máquinas virtuales, se encarga de gestionar los contenedores. Este software será necesario para un ejercicio de la competición CTF.

Para poder diseñar ejercicios interesantes es importante comprender la arquitectura de los elementos de Kubernetes. Estos elementos clave son, entre otros, los *deployments*, los *services*, los *Helm-chart*, etc.

9.3. Diseño de los elementos y servicios de la solución

Una vez completado el proceso de formación, es la hora de diseñar los elementos y servicios que componen el proyecto. En esta fase se puede distinguir dos tareas principales. Por un lado, el diseño de la infraestructura necesaria para desplegar un cyber-range. Por otro lado, los ejercicios y servicios que utilizarán la infraestructura anteriormente mencionada.

9.3.1. Diseño plataforma cyber-range

Esta tarea se centra en la planificación y desarrollo de una topología interesante para un proyecto de estas características. El diseño tiene que tener en mente que esta infraestructura será utilizada por el resto de elementos y que, por lo tanto, deberá cumplir con sus necesidades. Por este motivo, es importante conocer los recursos y elementos que se tienen disponibles a la hora de realizar el diseño.

9.3.2. Diseño ejercicios de la competición CTF

Esta tarea consiste en la planificación y desarrollo de los ejercicios que se realizan durante la competición CTF. Los ejercicios deben tener relación con la virtualización de servicios y con los conceptos de la iniciativa NFV de la ETSI. Adicionalmente, además de cumplir con lo anterior, deben ser diversos para que los usuarios practiquen distintas competencias.

En esta tarea es importante plantear múltiples tipos de ejercicios y tipos de ataques para disponer de una colección de ideas a utilizar. Para este propósito son útiles herramientas como la lluvia de ideas, etc.

9.4. Despliegue de la solución diseñada

Esta fase consiste en desplegar todos los diseños que se han realizado en la fase anterior. Al igual que en el anterior, en esta fase también hay dos tareas. Por un lado, el despliegue de la infraestructura diseñada. Por otro lado, el despliegue de los ejercicios diseñados.

9.4.1. Despliegue de infraestructura de la plataforma cyber-range

Esta tarea consiste en traducir los diseños de la infraestructura de la fase anterior en descriptores yaml. Después, estos descriptores son cargados en el servidor OSM y, a través de este, son desplegados como instancias en el servidor OpenStack y en el clúster de Kubernetes.

9.4.2. Despliegue de ejercicios de la competición CTF

Esta tarea consiste en instalar y configurar todo lo necesario para que los usuarios puedan realizar los ejercicios diseñados. En esta tarea se implementa en la solución cada uno de los ejercicios, con sus correspondientes características particulares. Como último paso, es necesario guardar todas las configuraciones e instalaciones realizadas para que este proceso pueda ser automatizado. De esta forma, se logra que el proceso de creación de ejercicios sea mucho más sencillo para el administrador del sistema.

9.5. Implementación del sistema de evaluación

Una vez realizadas todas las fases anteriores se dispone de una plataforma cyber-range con varios ejercicios, los cuales deben ser evaluados. En esta fase, se debe diseñar e implementar un sistema de evaluación que permita calificar el desempeño de los usuarios en la plataforma cyber-range. Esta fase está compuesta de una única tarea que contiene todo este proceso de desarrollo.

Para el sistema de evaluación se ha optado por el desarrollo de un servidor web, accesible para todos los usuarios e instructores. Para poder implementarlo, será necesario diseñar y desplegar dos elementos. Por un lado, la infraestructura necesaria para sostener el servidor web. Por otro lado, la programación necesaria para que el servidor web califique los resultados de los usuarios y se los muestre al instructor.

9.6. Gestión del proyecto

Esta fase de gestión que es esencial en cualquier tipo de proyecto. Todas las tareas que componen esta fase son importantes para la gestión del proyecto y se ejecutan de forma continuada durante la totalidad del desarrollo del proyecto. Estas tareas son las siguientes tres: seguimiento del proyecto, seguimiento del presupuesto y documentación

9.6.1. Seguimiento del proyecto

El seguimiento del proyecto es una tarea que se centra en supervisar que el desarrollo del proyecto cumple con la planificación estipulada. Busca comprobar que el desarrollo del proyecto alcanza los distintos hitos planteados. Otro objetivo de esta tarea es hacer frente o evitar cualquier desvío en la planificación. En caso de suceder estos desvíos respecto a la planificación, en esta tarea se busca minimizar los daños que puedan tener en el proyecto.

9.6.2. Seguimiento del presupuesto

De forma similar a la tarea anterior, en el seguimiento del presupuesto se hace frente o evita cualquier desvío respecto al presupuesto estipulado. Es decir, en esta tarea se busca comprobar que el desarrollo del proyecto va cumpliendo el presupuesto diseñado. Del mismo modo, en caso de haber un desvío respecto al presupuesto, en esta tarea se buscará la alternativa más eficiente que trastoque mínimamente el presupuesto planteado.

9.6.3. Documentación

Esta última tarea consiste en anotar los progresos y los resultados que se van realizando durante el proceso de desarrollo de las demás fases. De esta forma, se puede llevar un registro fiable de que se cumplen los requisitos y se alcanzan los objetivos planteados. Adicionalmente, también consiste en la redacción de la memoria final durante el transcurso del proyecto y al final de este.

10. DIAGRAMA DE GANTT

En la Ilustración 30 se muestra las fases y tareas del apartado 9: DESCRIPCIÓN DE FASES Y TAREAS distribuidas en según deben ser completadas de acorde a la planificación. A continuación, se detalla la posición de cada fase y tarea dentro del diagrama de Gantt.

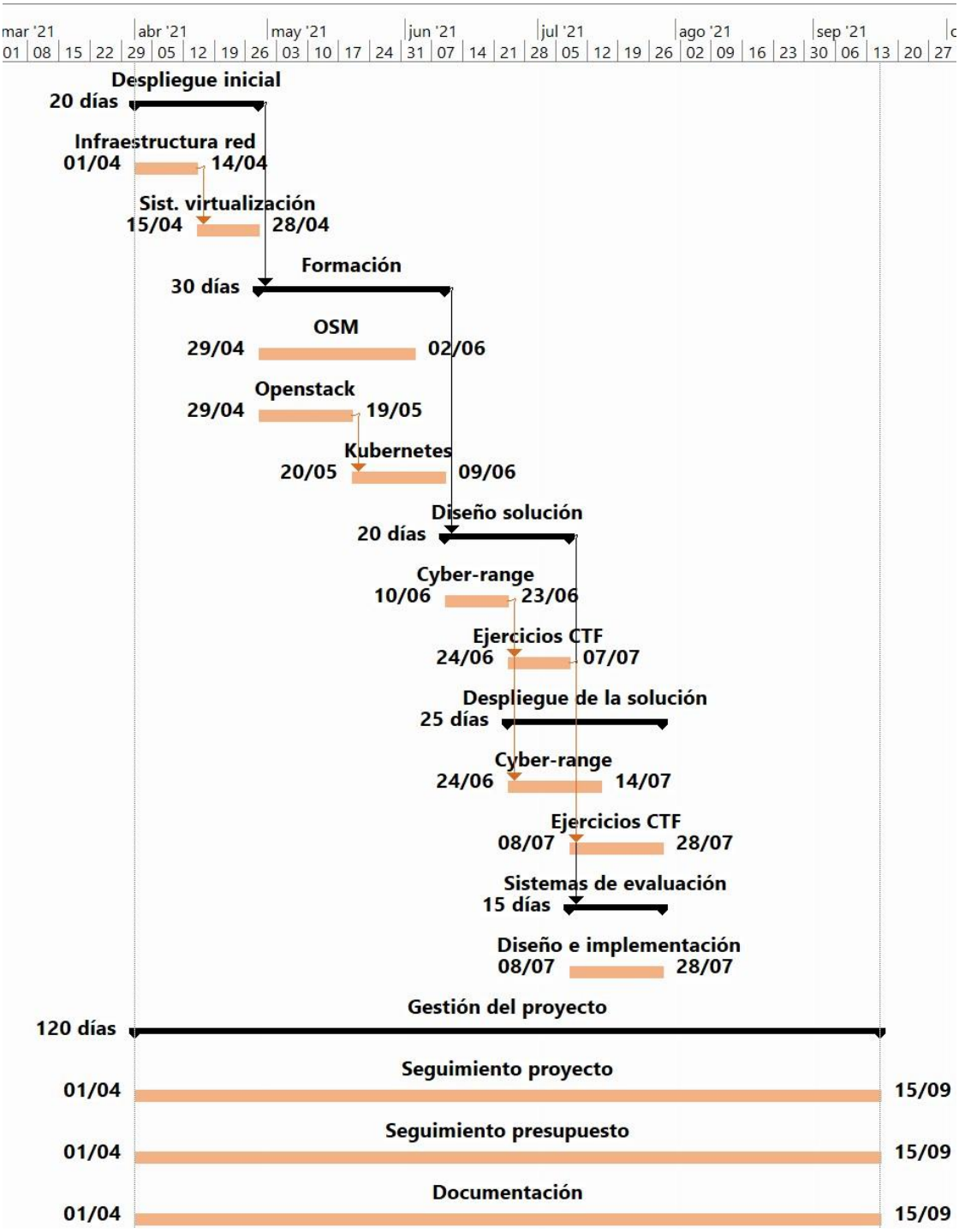

Ilustración 30: Diagrama de Gantt

10.1. Despliegue de la infraestructura y el software inicial

Esta fase se realiza al inicio del proyecto, debido a que este despliegue es necesario para comenzar con el proyecto. La duración total de esta fase es de 20 días. De estos 20 días, dos semanas corresponden a la tarea de despliegue de infraestructura de red y otras dos semanas a la tarea de instalación de sistemas de virtualización.

10.2. Formación

La fase de formación es dependiente de la fase anterior y, por lo tanto, se debe esperar a que termine para empezar con esta nueva fase. En total, la formación se extiende durante 30 días.

La formación de OpenStack está planteada para durar 3 semanas. La de Kubernetes empieza en cuanto se termina la de OpenStack y dura otras 3 semanas. Al mismo tiempo que se realizan estas formaciones, también se realiza la de OSM durante 5 semanas. Que estas formaciones coincidan se debe a que existen conceptos que deben ser aprendidos en más de una formación de forma simultánea.

10.3. Diseño de los elementos y servicios de la solución

Una vez terminada con la fase de formación, se empieza con la fase de diseño de la solución. Esta fase dura un total de 20 días. De estos 20 días, dos semanas corresponden al diseño de la plataforma cyber-range y otras dos semanas al diseño de los ejercicios de la competición CTF.

10.4. Despliegue de la solución diseñada

Cada una de las tareas de despliegue de esta fase se ejecutan cuando se ha completado su tarea correspondiente de diseño. En total, esta fase se desarrolla durante 25 días. Cada una de las tareas de despliegue dura 3 semanas.

10.5. Implementación del sistema de evaluación

Esta fase se ejecuta en cuanto termina la fase 10.3 y simultáneamente a la fase 10.4. La fase tiene una duración de 15 días, los cuales corresponden íntegramente a la única tarea de esta fase.

10.1. Gestión del proyecto

La fase de gestión del proyecto se realiza durante el transcurso de todo el proyecto. Se inicia simultáneamente a la primera tarea y se extiende más allá de la última, hasta que la memoria final ha sido terminada. En total, esta fase dura 120 días.

11. PRESUPUESTOS Y COSTES

El presupuesto se ha planteado bajo la perspectiva de un trabajo de fin de master. Esta perspectiva se ajusta al trabajo actual. Se plantea a un único ingeniero de telecomunicaciones realizando el trabajo, con la asistencia de los directores del proyecto para la realización de la gestión y coordinación del proyecto.

La duración de este proyecto es de seis meses. El tiempo invertido en este proyecto se ha calculado en base al número de créditos ofrecidos para la realización de un trabajo de fin de master, actualmente estimado en 600 horas.

11.1. Horas Internas

Trabajador	N.º	Coste Horario	Horas	Coste
Ingeniero de telecomunicaciones	1	30,00 €	600	18.000,00 €
Directores del proyecto	2	50,00 €	30	3.000,00 €
Total				**21.000,00 €**

Tabla 12: Horas internas proyecto ingeniería

11.2. Amortizaciones

Activo	Coste de Adquisición	Vida útil (meses)	Tiempo de uso (meses)	Amortización
Portátil Dell Vostro	1.100,00 €	48	6	137,50 €
Servidor OSM	1.500,00 €	48	6	187,50 €
Servidor Openstack	10.000,00 €	96	6	625,00 €
Total	**12.600,00 €**			**950,00 €**

Tabla 13: Amortizaciones proyecto ingeniería

11.3. Gastos

Concepto	Coste
Curso sobre NFV y OSM	15,00 €
Total	**15,00 €**

Tabla 14: Gastos proyecto ingeniería

11.4. Coste Total

Concepto	Coste Total
Horas internas	21.000,00 €
Amortizaciones	950,00 €
Gastos	15,00 €
Subcontrataciones	0,00 €
Subtotal	**21.965,00 €**
Imprevistos (5%)	1.098,25 €
TOTAL	**23.063,25 €**

Tabla 15: Coste total proyecto ingeniería

12.CONCLUSIONES

Con la solución planteada se ha conseguido cumplir con los objetivos marcados y se ha satisfecho todas las necesidades del proyecto. Este trabajo ha desarrollado una herramienta útil y eficaz que permite realizar la formación de hackers éticos en el área de la virtualización de servicios y de la iniciativa NFV de la ETSI. Adicionalmente, se ha diseñado como un proceso de aprendizaje totalmente práctico, donde se premia la originalidad y la creatividad. Esto hace que sea un proceso mucho más ameno e interesante para los usuarios de la plataforma que una formación exclusivamente teórica.

Se ha logrado diseñar e implementar una plataforma cyber-range versátil fácilmente modificable y ampliable que permite sostener una competición CTF. También se ha logrado desarrollar e implementar una serie de ejercicios interesantes para dicha competición CTF. Por último, también se ha logrado desarrollar un sistema de evaluación que permite el guiado y la calificación del desempeño de los usuarios que participan en el proceso de formación.

12.1. Trabajo futuro

Como se ha mencionado, con la solución propuesta se ha completado todos los objetivos iniciales que se habían planteado para el trabajo. Sin embargo, ahora se abren unos objetivos nuevos que podrían servir para la mejora y evolución de la plataforma. Con este propósito, se pueden plantear distintos objetivos futuros en cada una de las áreas de este trabajo. Uno de estos objetivos podría ser la mejora del sistema de evaluación, introduciendo distintas estadísticas y la capacidad de crear informes más detallados del desempeño de los usuarios de la cyber-range. Otro objetivo podría ser el comprobar el desempeño de la cyber-range con usuarios reales, con el objetivo de encontrar posibles fallos del diseño y áreas de mejora. Un último objetivo se podría centrar en el desarrollo de más ejercicios, de distintos tipos y de distintas dificultades.

13. BIBLIOGRAFÍA

[1] Amazon. (2021). *Amazon Elastic Container Service: Developer Guide*. Obtenido de https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html

[2] Cyberbit. (2021). *Plataforma cyber-range de Cyberbit*. Obtenido de https://www.cyberbit.com/platform/cyber-range/

[3] Cyberranges. (2021). *Plataforma cyber-range de Cyberranges*. Obtenido de https://cyberranges.com/

[4] CyberWiser. (2021). *Plataforma cyber-range de CyberWiser*. Obtenido de https://www.cyberwiser.eu/cyberwisereu-basic

[5] Docker. (2021). *Documentación*. Obtenido de https://docs.docker.com/engine/

[6] ETSI. (2013). *Network Functions Virtualisation(NFV); Use Cases.* GS NFV 001 (V1.1.1).

[7] ETSI. (2014). *Network Functions Virtualisation(NFV); Management and Orchestration.* GS NFV-MAN 001.

[8] ETSI. (2014). *Network FunctionsVirtualisation(NFV); ArchitecturalFramework.* GS NFV 002 (V1.1.1).

[9] ETSI. (2014). *Network FunctionsVirtualisation(NFV); Virtual Network Functions Architecture.* GS NFV-SWA 001 (V1.1.1).

[10] ETSI. (2015). *Network FunctionsVirtualisation(NFV); InfrastructureOverview.* GS NFV-INF 001.

[11] ETSI. (October 2012). Network Functions Virtualisation(NFV): Introductory Whitepaper. *SDN and OpenFlow World Congress.*

[12] European Cyber Security Organisation (ECSO). (2020). *Understanding Cyber Ranges: From Hype to Reality.*

[13] Google. (2021). *Documentación de Google Kubernetes Engine*. Obtenido de https://cloud.google.com/kubernetes-engine/docs

[14] Kubernetes. (2021). *Documentación de Kubernetes*. Obtenido de https://kubernetes.io/es/docs/concepts/

[15] Microsoft. (2021). *Página web del producto Azure*. Obtenido de https://azure.microsoft.com/es-es/product-categories/containers/

[16] Open Source MANO. (2019). *Whitepaper de la arquitectura*. Obtenido de https://osm-download.etsi.org/ftp/Documentation/201902-osm-scope-white-paper/#!index.md

[17] Open Source MANO. (2020). *Guia de usuario*. Obtenido de https://osm.etsi.org/docs/user-guide/index.html

[18] Openstack. (2018). *OpenStack Documentation*. Obtenido de https://docs.openstack.org/ussuri/

[19] VMware. (2021). *Documentación vSphere y ESXi*. Obtenido de https://docs.vmware.com/es/VMware-vSphere/index.html

[20] Xen project. (2020). *Wiki de Xen project*. Obtenido de https://wiki.xenproject.org/wiki/Main_Page

14.ANEXO I: DESCRIPCIÓN DE LAS TECNOLOGÍAS

14.1. NFV: Network Functions Virtualization

NFV es una iniciativa de la ETSI con el propósito de estandarizar la interoperabilidad de esta tecnología. Esta iniciativa está dirigida por la comunidad en un grupo de especificación industrial de la ETSI *(ETSI ISG)* compuesto por varios proveedores de red. El objetivo de NFV es la virtualización de todos los niveles de la arquitectura de red usando un hardware estándar.

Con esta virtualización es posible implementar diferentes funcionalidades de red bajo demanda. Es decir, se implementan nuevas funciones o servicios en el momento que son necesarios y se eliminan cuando ya no son de utilidad. Otra característica interesante es la clonación de las funciones de red, la cual dota de gran adaptabilidad a la red. Por ejemplo, cuando la carga de usuarios en un servidor sea elevada, se pueden desplegar nuevos servicios, o clonaciones de los actuales, para mejorar la experiencia de los usuarios.

Sin embargo, al tratarse de un control más centralizado que hace uso de la capacidad de cómputo de un equipo, la escalabilidad puede resultar compleja. Por este motivo esta tecnología no sustituye a todos los elementos de una infraestructura de red a gran escala, sino que se centra en implementaciones de redes corporativas, redes de campus, redes de sistemas de servicios en la nube y demás redes pequeñas.

NFV se complementa con SDN *(Software Defined Networks)* pero no es dependiente de SDN. Por un lado, SDN busca definir por software la topología de la red mediante el uso de switches genéricos, de tal forma que usando equipos físicos y software la topología sea variable y modificable. Por el otro lado, NFV busca realizar la virtualización de la topología similar a SDN, pero con el añadido de que también virtualiza equipos y funciones de la red.

14.1.1. Arquitectura

A continuación, se profundizará en cada uno de los módulos que componen la especificación NFV de la ETSII. En la *Ilustración 31* se muestra un diagrama detallado de la arquitectura NFV de la ETSI.

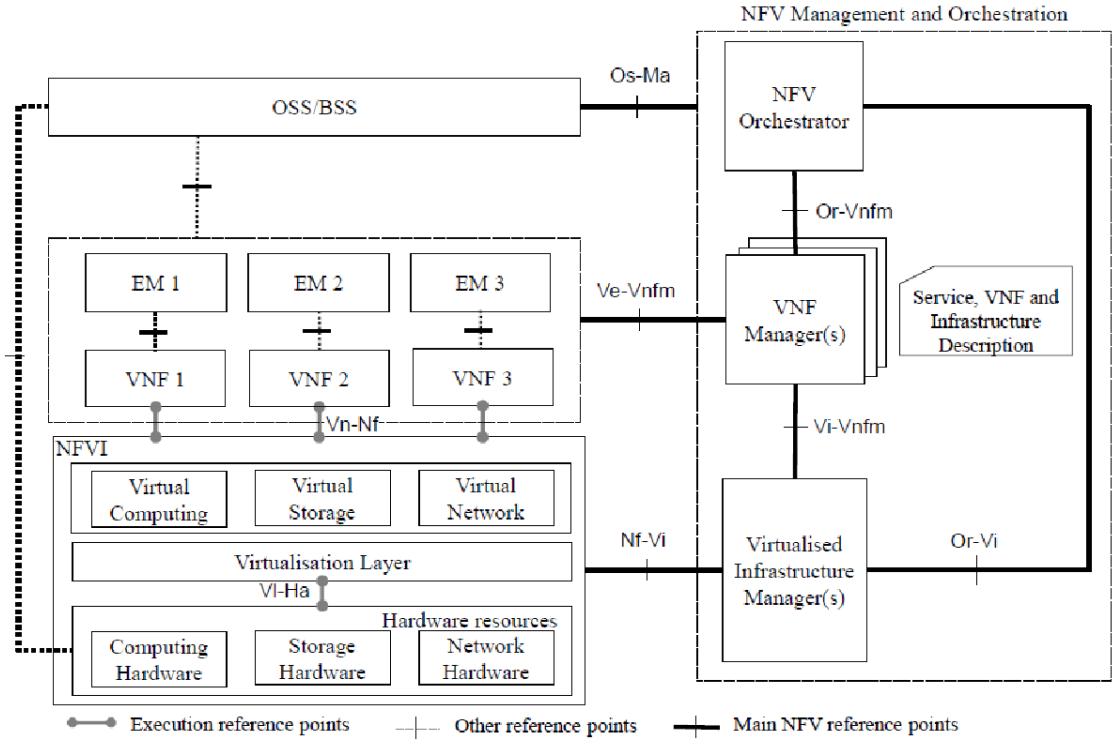

Ilustración 31: Arquitectura de referencia de NFV [9]

14.1.1.1. OSS/BSS: Operation Support System/Business Support System

Es el sistema que da soporte al software de negocio y al software del usuario final. Aquí se encuentra el software de operación y las bases de datos de gestión, de operación de la red, de clientes, etc. Como tal no forma parte de la arquitectura de NFV. Es el software que ya tiene una empresa y que se ejecuta sobre NFV. El usuario final no percibe la diferencia entre operar sobre equipos virtuales NFV o físicos tradicionales.

Las órdenes que se dan desde el nivel OSS/BSS son transmitidas al orquestador. El orquestador, a su vez, las traduce y transmite a las correspondientes funciones virtualizadas de la red. Al mismo tiempo, el orquestador también interactúa con la infraestructura virtualizada y hace que las órdenes se propaguen por la infraestructura que sostiene todo el sistema.

14.1.1.2. MANO: Management and Orchestration

Esta entidad es la encargada de introducir, gestionar u orquestar funciones virtualizadas de red (*VNF*) o servicios de red (*NS*). A su vez, hace que todas estas funciones sean compatibles o traducibles sobre la infraestructura virtualizada disponible.

MANO está compuesto por tres elementos. Cada uno de estos elementos tienen una doble interacción. Por un lado, interaccionan entre sí dentro del sistema de gestión y orquestación. Por otro lado, gestiona y orquesta una parte de la arquitectura NFV.

El **NFVO** *(NFV Orchestrator)* se encarga de que los otros elementos de MANO funcionen correctamente en base a las demandas realizadas por el software de usuario en el OSS/BSS. Es decir, se ocupa de que no haya incoherencias, bucles, etc. Gestiona y controla el VIM y el VNFM.

El **VNFM** *(VNF Management)* gestiona el ciclo de vida de las VNF dinámicamente usando descriptores. Estos descriptores consisten en atributos y requerimientos VNF. En otras palabras, con los descriptores VNFM se encarga de crear, modificar y finalizar las VNF. Estos descriptores son configurables para actuar de manera dinámica en función de los requerimientos del usuario.

El **VIM** *(Virtual Infrastructure Manager)* se encarga de gestionar la infraestructura y recursos hardware y software de NFVI. Contiene un registro con la relación o traducción de los recursos virtuales a los recursos físicos. En otras palabras, este elemento conoce qué equipos físicos están siendo utilizados y también sabe qué equipos lógicos están usando esos equipos físicos. Por lo tanto, asegura que haya soporte físico suficiente para nuevas peticiones intentando optimizar el uso de los recursos disponibles.

14.1.1.3. VNF: Virtualized Network function

En este módulo se agrupan las funciones virtualizadas de la red *(VNF)*. Cada una de las VNF, usualmente, cuenta con la lógica de una función de red concreta. Por ejemplo: un cortafuegos, un control de acceso, etc.

Las VNF son completamente agnósticas al hardware y se muestran completamente ajenas a él. Es decir, las funciones de red no muestran ninguna diferencia si se ejecutan sobre sistemas físicos o lógicos. Por lo tanto, las VNF, a nivel lógico, tienen las mismas características y funciones que las funciones físicas de red *(PNF)*. En otras palabras, un firewall lógico actúa exactamente igual que un firewall físico.

Aunque la iniciativa de la ETSI establezca que lo común sea que cada VNF esté compuesta por una única máquina virtual, es posible que una función de red haga uso de varias máquinas virtuales. Se suele optar por esta alternativa cuando las funciones son demasiado complejas y necesitan más de una máquina virtual.

En este nivel se encuentran los **EMS** *(Element Manager System).* Estos elementos se encargan de gestionar los elementos de red con el framework de OSI: FCAPS *(Fault, Configuration, Accounting, Performance and Security)*.

Adicionalmente, permiten la integración de las VNF con el VNFM y el OSS. No es necesario que los EMS se encuentren desplegados para todos los casos. Se pueden desplegar VNF sin usar EMS.

14.1.1.4. NFVI: NFV Infrastructure

Este módulo está formado por el bajo nivel de la arquitectura. Son los recursos hardware y software sobre los que se sostiene las VNF y toda la arquitectura. Se puede diferenciar dos niveles dentro del NFVI, el nivel de los recursos hardware y el nivel de los recursos virtualizados.

Por un lado, los recursos hardware se dividen en tres tipos: recursos de procesamiento, de almacenamiento o de red. Estos recursos tienen un localizador llamado PoP *(Punto de presencia)* que permite localizar dónde se encuentran los recursos físicos de cada VNF.

Por otro lado, en una capa de virtualización, se transforman los equipos físicos en recursos virtuales. Esta transformación es equivalente a la operación realizada por cualquier hipervisor de cualquier sistema de virtualización. Estos recursos virtuales se presentan como entidades independientes a cada VNF. Es decir, como si cada VNF tuviera ese procesamiento (microprocesador), ese almacenamiento (disco duro) y esa red (enlaces) en exclusiva.

14.1.2. Ciclo de vida

Las especificaciones de la iniciativa NFV plantean un ciclo de vida para las VNFs. Este ciclo de vida planteado no entra en mucho detalle, ya que deja en manos de las soluciones el implementar y decidir los detalles de este ciclo de vida. Aun así, se puede distinguir varias fases o etapas generales.

En la primera fase se realiza la instalación. Esta fase consiste en la definición de los volúmenes, paquetes o imágenes que constituyen las funciones de red. En la segunda fase se realiza la instanciación de las imágenes que se han definido en el paso previo. Una VNF se instancia a partir de una imagen ya definida, de tal forma que se pueda modificar o clonar sin alterar a la imagen base. Por último, se encuentra la fase de terminación de la instancia. En esta fase se elimina la VNF sin alterar las demás VNFs o las imágenes base.

14.1.3. Soluciones software

El propósito de la iniciativa de la ETSI es que de las especificaciones de NFV surjan soluciones software que ofrezcan la tecnología NFV al público general. Unos ejemplos de estas soluciones serían Open Source Mano y OpenStack. Relacionándolo con la arquitectura de NFV mostrada en la *Ilustración 31*, Open Source Mano realiza las funciones del orquestador y del VNFM y OpenStack realiza las funciones del VIM y de la capa de virtualización de NFVI.

14.2. OSM: Open Source MANO

OSM está diseñado en una arquitectura de capas. Consume servicios de los niveles inferiores y ofrece servicios más complejos a los niveles superiores. Al mismo tiempo, tiene dos tipos de modo de funcionamiento, *el Platform Operator view* que se centra en que la plataforma funcione correctamente y el *Service Platform view* que se encarga de crear y suministrar los objetos de servicio que necesita el usuario.

14.2.1. Arquitectura (Service Platform view)

Como ya se ha mencionado, la arquitectura de OSM se divide en capas. Cada una de estas consume servicios de plataformas de nivel inferior y, combinando estos servicios, ofrece un servicio más complejo a la plataforma de nivel superior. Esta estructura de capas se puede ver en la forma en la que se componen los *Network Services (NS)*.

OSM consume servicios de plataformas VIM *(Virtual Infrastructure Manager)* y de plataformas SDN *(Software Defined Network)* para desplegar los *Network Services*. Estos servicios consumidos son las VNFs, las cuales se crean consumiendo distintas máquinas virtuales llamadas VDU *(Virtual Deployment Unit)*.

Mediante este procedimiento, OSM es capaz de crear un servicio de *red bajo demanda* y ofrecerlo bajo un *objeto de servicio*. Existen dos tipos de objetos de servicio NaaS: Network Services y Network Slice. En la *Ilustración 32* se muestra un diagrama detallado de toda la arquitectura de OSM.

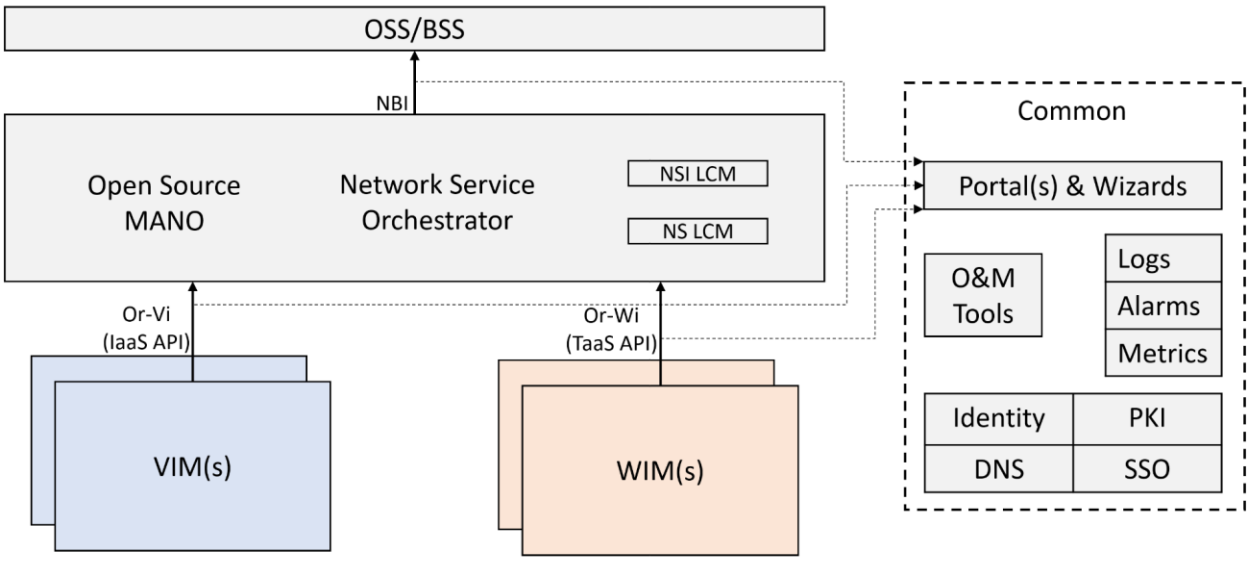

Ilustración 32: Arquitectura de OSM

Como se puede observar en la *Ilustración 32*, se puede dividir la arquitectura en tres bloques principales. A continuación, se desarrollará con más detalles estos bloques.

Primero, se encuentra el **Network Service Orchestrator**. El cual se encarga de controlar los ciclos de vida de los distintos *Network Services* y de los distintos *Network Slices* ofrecidas bajo demanda a través del *Northbound interface*. Es el bloque del que OSM se hace cargo.

Después, se encuentran varios **VIM**s que trabajan como administradores de la *Virtual Infrastructure Platform*. A través de la API *Or-Vi* ofrecen infraestructura bajo servicio *(IaaS)* a los niveles superiores. Si hubiera una necesidad

de conectividad que VIM no pudiera suministrar se puede hacer uso de la asistencia SDN. OSM delega la gestión de este bloque a otro software como, por ejemplo, OpenStack o Kubernetes.

Por último, se encuentran varios **WIM**s que trabajan como administradores de SDN *(Software Defined Networks).* A través de la API *Or-Wi* ofrecen conectividad o telecomunicaciones como servicio *(TaaS)* a los niveles superiores. OSM también delega la gestión de este bloque a otro software como, por ejemplo, OpenStack o Kubernetes.

14.2.2. Servicios ofrecidos por el Northbound Interface (NBI): Network Services (NS)

Un *Network Service* agrupa en un solo objeto de servicio un conjunto de VNFs interconectadas que pueden desplegarse sobre varias tecnologías, a diferentes niveles de centralización y en diversas áreas geográficas. OSM va más allá de lo que inicialmente define ETSI en NFV, permitiendo combinar en una sola NS funciones de red virtuales, físicas e incluso híbridas. OSM ofrece las NS creadas al OSS/BSS, para que se ejecuten los servicios del usuario.

Al mismo tiempo, a través de un interfaz web accesible desde internet, permite que el administrador se conecte al servidor OSM para que ejecute tareas de despliegue, mantenimiento, configuración o eliminación de NS. Estas operaciones de administrador están contempladas dentro del ciclo de vida de las NS.

La iniciativa NFV de ETSI plantea un boceto de **los ciclos de vida** de las VNF. Pero, realmente, deja en manos de las soluciones software la forma de implementar estos ciclos de vida. En el caso de OSM, las VNF prestan sus servicios a los NS y son los NS los que son presentados al usuario en el OSS/BSS. Por lo tanto, OSM implementa la gestión de los ciclos de vida de los NS y, de forma indirecta, de las VNF. Los ciclos de vida de los NS que implementa OSM son los que se muestran en la *Ilustración 33* y se describen en detalle a continuación.

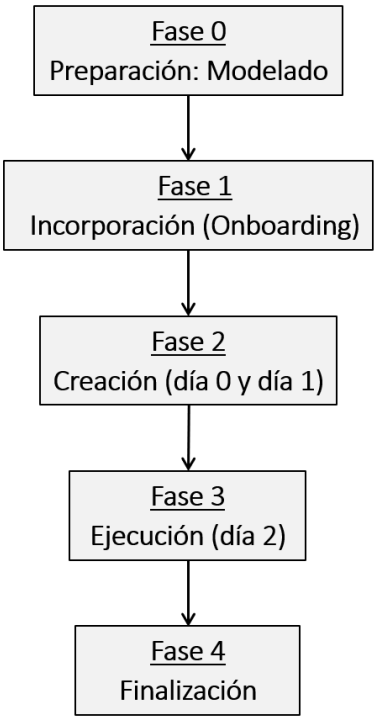

Ilustración 33: ciclos de vida de NS de OSM

14.2.2.1. Fase 0 - Preparación: Modelado

La fase 0 es la de preparación y modelado de la NS. Esta fase es más bien un prerrequisito y como tal no forma parte del ciclo de vida. Sin embargo, es importante para las fases posteriores. En esta fase, el IM *(Information Model)* de OSM proporciona los **descriptores yaml** necesarios para crear una plantilla del NS. Estos descriptores contienen descripciones de la topología interna, recursos necesarios, ciclos de vida de las funciones de red, etc. Una vez un *Network Service* se ha modelado, conforma un *paquete Network Service*. Este paquete sirve como una plantilla que puede ser modificada para incorporar atributos de una instancia NS en concreto.

A pesar de ser un prerrequisito, el diferenciar esta fase de las demás proporciona varias ventajas. Por un lado, permite replicar las VNF y las NS en cualquier punto de la infraestructura las veces que sean necesarias. Por otro lado, permite desplegar NS diferentes modificando ligeramente alguno de los parámetros de los paquetes NS.

14.2.2.2. Fase 1 - Incorporación (Onboarding)

En la fase 1 se realiza la operación de incorporación. Consiste en subir a los servidores OSM los paquetes NS una vez ya están listos. De esta forma, se permite posteriormente desplegar los NS en base a los paquetes. En esta fase OSM ofrece una API a través de su interfaz NorthBound que permite operaciones CRUD (Create, Read, Update, Delete) sobre los paquetes NS. Adicionalmente, se realizan las comprobaciones necesarias para validar los paquetes incorporados.

14.2.2.3. Fase 2 – Creación del NS (día 0 y día 1)

En esta fase se instancian las NS a partir de los paquetes que se han diseñado y cargado en las fases previas. OSM ofrece una API que permite operaciones CRUD sobre las instancias NS. Estas operaciones, a diferencia de la fase previa, no consisten en cargar y validar los paquetes NS, sino que son operaciones sobre las instancias que se crean a partir de los paquetes. Para el caso de una creación o instanciación de una NS, OSM interactúa con varias plataformas de servicio de niveles inferiores para crear el objeto de servicio de la instancia NS.

En esta fase se distinguen dos tipos de operaciones. Por un lado, las **operaciones de día 0**, que consisten en la creación y puesta en marcha de las instancias NS. Por otro lado, las **operaciones de día 1**, que consisten en la ejecución de primitivas de forma automática inmediatamente después de la creación de las instancias de NS.

14.2.2.4. Fase 3 – Ejecución del NS (día 2)

En esta fase, la instancia NS se convierte en el único objeto relevante para las operaciones restantes. Estas operaciones se dividen en tres distintas categorías y se ejecutan en el momento en el que son solicitadas. Primero, las **operaciones comunes** del ciclo de vida son operaciones estándares aplicables a cualquier NS. Segundo, **operaciones de día 2** son operaciones sobre alguna funcionalidad específica que ofrece una determinada instancia NS. Usualmente suelen ser solicitadas por el cliente y pueden consistir en primitivas de configuración, de monitorización, etc. Tercero, las **operaciones internas** son operaciones que no tiene por qué solicitarlas el cliente, pero que se ejecutan siguiendo unas políticas internas.

Adicionalmente, dentro de la monitorización, OSM permite varias métricas y alarmas para elementos relacionados con el nivel VDU-PDU o específicas del nivel de aplicación. También se permite realizar alarmas bajo demanda.

14.2.2.5. Fase 4 – Finalización del NS

Esta fase es la última del ciclo de vida de las instancias NS. Cuando se inicia esta fase comienza el proceso de finalización de la instancia NS. Finalizar una instancia NS libera los recursos que se le había asignado. Sin embargo, existe la opción de conservar ciertos elementos de la NS que estaban almacenados y que no se deseen eliminar, por ejemplo, los volúmenes.

14.2.3. Servicios ofrecidos por el Northbound Interface (NBI): Network Slices (NetSlice)

3GPP presenta un concepto llamado *Network Slices*, clave en tecnologías 5G entre otras. Este concepto tiene varias similitudes con la iniciativa NFV y, por lo tanto, OSM ha hecho que su solución implemente las *Network Slices* como una particularización de los *Network Services.* De esta forma, las *Network Slices* se pueden entender como varios NS especializadas. Sin embargo, la forma de definir el ciclo de vida de las *Network Slices* presenta ligeras diferencias, pero la finalidad es análoga.

3GPP define de forma ligeramente distinta a ETSI NFV la relación de las *Network Slices* con los *Network Services*. La principal diferencia es que para 3GPP los *Network Slice* están compuestos por varios *Network Subnets*.

14.2.4. Servicios ofrecidos por el Southbound Interface (Or-Vi): VIM

OSM orquesta y administra las *Virtualized Infrastructure Manager* (VIM) a través del **Southbound Interface**. Es decir, orquesta un **software externo** que permite controlar el *Virtual Infrastructure Platform*. Gracias a esta orquestación se consigue ofrecer *Infrastructure as a Service (IaaS)* a los clientes y usuarios de OSM. Por lo tanto, OSM deja que el VIM se encargue de controlar los recursos de computación, de red y de almacenamiento, mientras que él se encarga de la interoperabilidad y la orquestación.

VIM es un elemento externo y agnóstico a OSM. Por lo tanto, VIM crea recursos virtuales con *Enhanced Platform Awareness (EPA)*, sin embargo, no puede suministrar conectividad entre otros recursos que han desplegado otras VIM u otros elementos. Para estos casos, OSM se encarga de suministrar esta conectividad con la ayuda de un controlador SDN. A esta técnica se la conoce como *SDN Assist*. Cuando esta asistencia ocurre, VIM se encarga de desplegar las máquinas virtuales y la red, mientras que el controlador SDN se encarga de realizar la conectividad entre los elementos.

Entre otras opciones, los softwares más reconocibles que realizan las funciones de VIM son: OpenVIM, OpenStack, VMware y Kubernetes.

14.2.5. Servicios ofrecidos por el Southbound Interface (Or-Wi): WIM

Se define WIM como *WAN Infrastructure Managers*. Los cual es lo mismo que un controlador SDN (SDNC) que controla los recursos disponibles en un *Software-Defined Transport Platform*. WIM se encarga de suministrar *conexiones bajo demanda* y ofrecer el ciclo de vida de estas al nivel superior a través de una API. Con estos controladores es posible realizar múltiples tipos de conexiones involucrando diferentes tipos de tecnologías.

14.3. OpenStack

A pesar de que OpenStack cumpla con los requerimientos de la iniciativa NFV, no hace uso de los mismos términos en su documentación. En la documentación oficial [18] a las VNFs o a las *Network Services* se les suele llamar servicios virtualizados o simplemente instancias.

14.3.1. Arquitectura

La arquitectura de OpenStack es un sistema complejo de varios elementos interconectados. Cuenta con nueve componentes principales, los cuales son: Keystone, Horizon, Glance, Nova, Neutron, Swift, Cinder, Heat y Ceilometer. La disposición de estos elementos dentro de la arquitectura se muestra en la *Ilustración 34.* En los siguientes apartados se describen detalladamente cada uno de ellos.

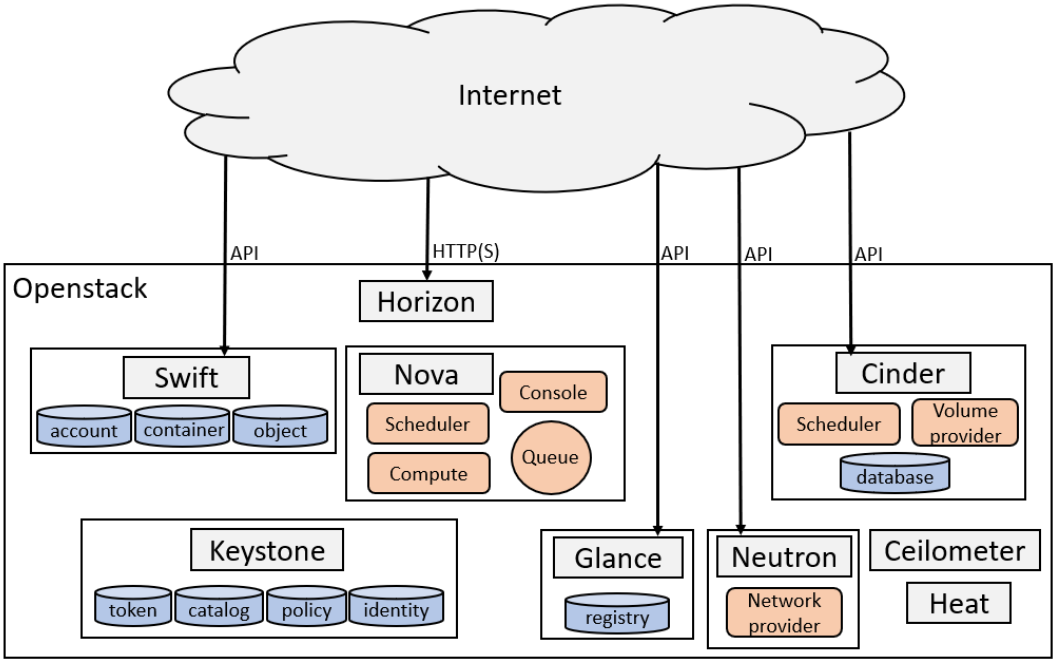

Ilustración 34: Arquitectura simplificada de OpenStack

14.3.1.1. Keystone

Proporciona el servicio de autenticación para que el cliente o usuario pueda autenticarse y pueda acceder al resto de elementos de la arquitectura. La autenticación se basa en el uso de unos tokens de seguridad. Una vez los usuarios han introducido sus credenciales, Keystone responde con un token que contiene los permisos y privilegios de cada usuario.

A través de este componente se puede determinar a qué proyecto o tenant pertenecen cada usuario. En la documentación de OpenStack [18], se usan de forma análoga a los tenants y los proyectos, siendo definidos como un grupo de usuarios con unos permisos y privilegios concretos.

Como se puede apreciar en la *Ilustración 34,* para proporcionar este servicio se hace uso de 4 elementos. Primero, el *Token Backend* administra los tokens que se usan para la autenticación del usuario. Segundo, el *Catalog Backend* ofrece un registro endpoint que se usa para un descubrimiento de servicios virtualizados en ejecución. Todos los servicios virtualizados de OpenStack están conectados a estos endpoints. Por lo que mediante este elemento se

permite listar todos los servicios y especificar a cuáles tiene acceso un determinado usuario. Tercero, el *Policy Backend* comprueba que los usuarios no rompen ninguna política o regla. Por último, el *Assignment Backend* provee datos acerca de las reglas, les da nombre y características a estas.

14.3.1.2. Glance

Glance es el componente encargado de almacenar y desplegar las instancias de los servicios. Contiene todas las imágenes de los servicios virtualizados que se pueden ofrecer. Al mismo tiempo permite a los usuarios subir, buscar y obtener imágenes de máquinas virtuales bajo demanda. Es un repositorio de imágenes de máquinas virtuales y de metadatos.

El usuario puede acceder al *Glance Registry*. Este registro está conectado a la base de datos de las imágenes de las máquinas virtuales y de sus respectivos metadatos. Desde este lugar es donde el usuario decide qué imágenes desplegar.

14.3.1.3. Nova

Nova es el componente donde ocurre el procesamiento y ejecución de los servicios virtualizados que se han desplegado desde Glance. Es un componente que se encuentra abstraído por completo del resto de elementos de la arquitectura con el propósito de permitir que la capacidad de procesamiento del sistema sea fácilmente escalable bajo demanda.

Nova está diseñado sobre una arquitectura basada en mensajes. Por lo tanto, cuando tiene que ejecutar el procesamiento de cualquier proceso, lo hace en el orden que le llegan los mensajes a la cola de mensajes a través de su API. Sin embargo, existe la opción de que el usuario acceda a Nova directamente a través de una terminal.

La arquitectura de mensajes está compuesta por varios elementos, Los más importantes son: una *API de acceso* con la que se comunica con los otros servicios, un *Scheduler* que organiza y crea la cola de procesamiento y un *módulo de cómputo* que procesa los mensajes de la cola.

14.3.1.4. Neutron

Neutron es el componente que se encarga de proporcionar la comunicación entre las distintas instancias de los servicios y con el exterior. Por lo tanto, no se puede considerar como un elemento totalmente independiente, debido a que se encuentra distribuido entre los demás servicios virtualizados con el objetivo de ofrecer *Network as a Service*.

Es el componente encargado de suministrar interfaces de red a los servicios virtualizados y de conectar esas interfaces a la red que ha indicado el usuario. Del mismo modo, permite a los usuarios crear subredes y routers virtuales y hacer que sus servicios virtualizados se puedan conectar a ellas bajo demanda. La única limitación para realizar esta operación es que el usuario tenga acceso al tenant o proyecto adecuado. Para la autenticación y la autorización delega en Keystone.

14.3.1.5. Swift

Swift es uno de los dos componentes de almacenamiento de los que dispone la arquitectura de OpenStack. Este servicio de almacenamiento guarda los datos convirtiéndolos en objetos. Permite almacenar cualquier tipo de archivo, imagen u otro dato, convirtiéndolo previamente en objetos.

Al almacenar los datos en Swift, el usuario recibe una dirección de referencia en la cual sus datos han sido guardados. Sin embargo, en ningún momento recibe la dirección real del lugar en el que los datos han sido realmente almacenados. Por lo tanto, permite que el usuario no se tenga que preocupar por la administración de la memoria. Como punto negativo, si el usuario extraviara la dirección de referencia, le sería imposible localizar sus datos almacenados.

Este componente está compuesto por un Swift-proxy que está conectado a tres bases de datos. Las cuales contienen las *cuentas de usuarios* (administradores, clientes, tenant, etc.), contenedores que almacenan todo tipo de *metadatos* y los *objetos guardados* (archivos y datos de todo tipo).

14.3.1.6. Cinder

Cinder es el segundo componente de almacenamiento de los que dispone la arquitectura de OpenStack. A diferencia de Swift, en este componente se guardan los datos por bloques. Lo que le hace más similar a los diseños de discos duros clásicos. Un usuario usando el almacenamiento de Cinder sabe en todo momento la dirección real de la memoria en la que sus datos están almacenados.

Cinder está compuesto por una API que lo comunica con los demás servicios, un *Schedule* que organiza el acceso a memoria de las peticiones de lectura y escritura, un *volumen* que organiza los bloques de la memoria y sirve de backup y, finalmente, una *base de datos* que almacena todos los metadatos de los archivos almacenados en los bloques.

14.3.1.7. Horizon

Horizon es el primer elemento al que se accede y el único que, en principio, visualiza el usuario. Es una interfaz gráfica accesible a través del navegador web y a través de internet. De este modo, permite a los usuarios acceder de forma remota al servidor OpenStack sin usar ningún software que no sea un navegador web.

Cuando un usuario se conecta a Horizon, envía una petición a Keystone para autenticarse. Una vez autenticado, el usuario puede acceder a cualquier componente de OpenStack o a cualquier servicio virtualizado que tenga accesible desde su tenant. También permite diseñar, modificar, eliminar o desplegar redes virtuales bajo demanda según las necesidades del usuario.

14.3.1.8. Ceilometer

Ceilometer es el componente de OpenStack que se encarga de la monitorización de las instancias desplegadas por los usuarios y por OpenStack. Mediante esta monitorización obtiene los datos que utiliza para la creación de estadísticas y logs de todos los servicios.

Algunas de las estadísticas que realiza son la contabilización del uso que hacen los usuarios de los servicios disponibles, la contabilización de qué tipos de servicios son los que más se usan y el registro de los errores que ocurren en el sistema.

14.3.1.9. Heat

Por último, Heat es el componente de OpenStack que se encarga de la orquestación de todo el sistema. Por lo tanto, es el que realiza todas las tareas de coordinación de todas las instancias de los servicios desplegados. Este componente, define cómo se consumen los recursos, cómo se activan los recursos, solicita a Keystone que autentique los tokens de los usuarios, envía los tokens a sus respectivos servicios, etc.

14.3.1.10. Comunicación entre los componentes de OpenStack.

Cuando un usuario accede al servidor de OpenStack, lo hace conectándose a la interfaz Horizon a través de internet y del navegador web. Antes de permitirle acceder, la interfaz Horizon le solicitará al usuario que se autentique enviando una petición a Keystone. Si las credenciales introducidas por el usuario son correctas, Keystone devolverá un token de autenticación con el que el usuario podrá acceder a todos las instancias y recursos que sean accesibles para su perfil con su rol correspondiente.

Una vez autenticado, el usuario puede realizar varias operaciones sobre los servicios virtualizados (Configurar, crear, eliminar, etc.). Si decide instanciar un servicio virtualizado nuevo, el mensaje de la petición de creación saldrá de la interfaz Horizon hacia el módulo Nova. En Nova se procesará esta petición y devolverá un primer mensaje de confirmación si la petición es correcta. Posteriormente, Nova enviará una petición a Keystone para validar que el usuario tiene acceso a la imagen solicitada.

Al mismo tiempo, mientras se espera a la respuesta de Keystone, Nova envía una petición a Neutron para que cree la red, los elementos de la red o las interfaces necesarias para esa nueva instancia. Una vez Neutron ha finalizado, responde a la petición de Nova y, adicionalmente, guarda en el módulo Cinder los elementos que ha definido (redes, interfaces, routers, etc.) para que actúe de backup.

Una vez que Nova disponga de la confirmación de la legitimidad del usuario por parte de Keystone y de los elementos de red necesarios, iniciará el proceso de instanciación de la imagen. Para ello, primero solicitará a Cinder un volumen de almacenamiento. A continuación, le solicitará a Glance la imagen del servicio virtualizado a ejecutar. Para ello, Glance debe conocer si la imagen se guarda en Swift o en Cinder, obtenerla de uno de estos módulos y ofrecérsela a Nova. Finalmente, Nova desplegará la instancia de la imagen. El usuario podrá acceder a esta nueva instancia a través del hipervisor que se encuentra dentro de la interfaz Horizon.

14.4. Kubernetes

Kubernetes es una plataforma de contenedores y de microservicios en la nube de código abierto. Fue presentada por Google en 2014 tras varios años de desarrollo y uso interno. Con el tiempo, Kubernetes ha ido desarrollándose hasta convertirse en uno de los estándares más populares de la industria en cuando a software de administración de contenedores. Todo parece indicar que Kubernetes tiene un futuro muy prometedor como sistema de gestión de servicios en la nube.

14.4.1. Arquitectura

Kubernetes está diseñado como una arquitectura maestro-esclavo. En esta arquitectura un nodo maestro se encarga de gestionar qué servicios se despliegan en qué nodos workers. Un maestro puede trabajar con un gran número de nodos workers. Sin embargo, cuando este número ya se hace difícil de administrar se puede desplegar otros nodos master adicionales que faciliten la gestión.

La agrupación de distintos nodos es lo que forma el clúster de Kubernetes. Para permitir una accesibilidad sencilla al usuario final, el clúster debe permanece totalmente agnóstico al usuario. Es decir, el usuario no debe saber si el servicio se está ejecutando sobre un clúster o sobre un equipo físico. En la *Ilustración 35* se muestra un diagrama de la arquitectura de un clúster de Kubernetes con sus principales elementos.

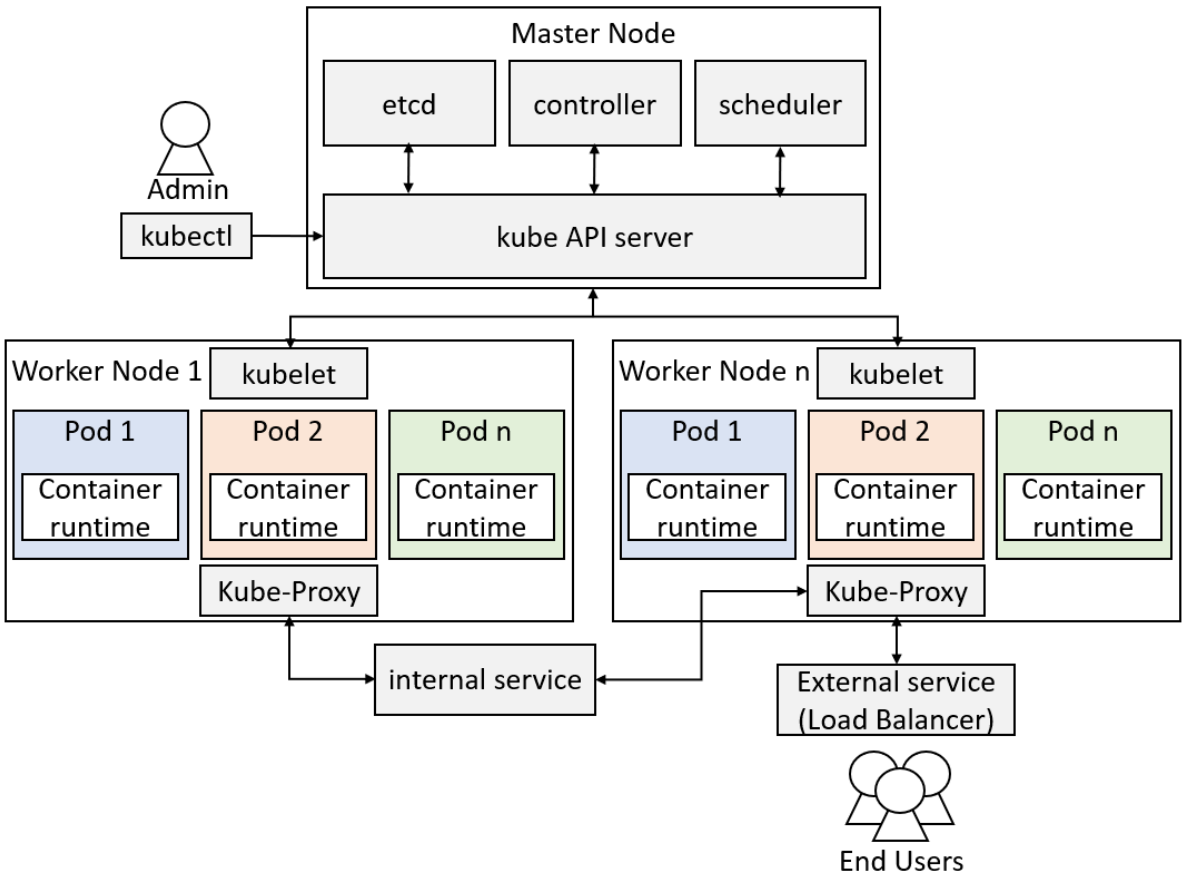

Ilustración 35: Arquitectura de Kubernetes

14.4.1.1. Nodos worker

Los nodos workers son los que realizan la función de esclavos del paradigma maestro-esclavo de esta arquitectura. Reciben órdenes del master para realizar operaciones de distintos tipos, por ejemplo, desplegar nuevos servicios o eliminar alguno ya existente. Hay múltiples nodos workers dentro de cada clúster de Kubernetes y cada uno puede ser tanto físico como virtual.

Cada uno de los nodos workes está compuesto por múltiples pods. Los pods son la unidad más pequeña e indivisible de Kubernetes. Son los pequeños componentes en los que se han dividido las aplicaciones monolíticas para crear los microservicios. Sobre estos componentes se sostiene todo el servicio virtualizado.

A parte de los pods, en cada nodo worker hay otros tres elementos: Kubelet, Kube-Proxy y un sistema de contenedores.

El sistema de **contenedores** realiza las tareas de más bajo nivel y se encarga de que los servicios virtualizados sean agnósticos a la infraestructura de Kubernetes. Para desplegar este elemento, Kubernetes delega en software de terceros. Existen varias alternativas, pero el más generalizado y el más utilizado suele ser Docker.

Las tareas de planificación *(scheduler)* del nodo son realizadas por **Kubelet**. Este elemento se encarga de realizar las tareas de gestión y mantenimiento tanto del propio nodo como de los contenedores que están ejecutándose. Cuando recibe la orden por parte del master, también se encarga de desplegar o finalizar los pods que se le hayan indicado.

Las tareas de encaminamiento y enrutado de la red son realizadas por **Kube-Proxy**. Este elemento no funciona como una interfaz de red convencional, ya que ese trabajo lo comparte con el módulo Service. Este elemento sirve como punto de acceso y de salida de todo el tráfico que vaya encaminado desde o hacia los pods de un nodo a otro. Procura que el tráfico generado o recibido se encamine por la red de la forma más ordenada y eficiente posible. De manera complementaria, el Kube-Proxy se debe conectar a un Service para obtener una dirección IP y obtener una conectividad plena entre los distintos elementos de la red y con el exterior.

14.4.1.2. Plano de control del Clúster (master)

El master es el elemento de la arquitectura de Kubernetes que da las ordenes, controla el estado del clúster y gestiona a los nodos workers. Su trabajo es el de gestionar y orquestar todo el clúster de Kubernetes. Es decir, no realizan ninguna tarea de virtualización, ni sostiene ninguna instancia. Sus tareas consisten en planificar los pods, monitorizar el clúster, unir nuevos nodos al clúster, etc. Si el clúster crece en complejidad y la cantidad de nodos se vuelve difícil de administrar, existe la posibilidad de añadir uno o varios masters más al clúster, reduciendo la carga de trabajo y mejorando la operabilidad.

Un nodo master cuenta con cuatro elementos principales: Un planificador *(scheduler),* un controlador, una API de alto nivel y un sistema de almacenamiento llamado *etcd.*

El administrador y el cliente acceden al clúster de Kubernetes haciendo peticiones a través de una **API de alto nivel** que se encuentra dentro del master para desplegar, gestionar o eliminar servicios virtualizados. La herramienta que usa el usuario para enviar peticiones por línea de comandos a esta API se llama **Kubectl**. Para que una petición de usuario o una orden sea validada y se redirija al planificador, primero se debe autenticar su procedencia. Para ello realiza funciones de autenticación y desvela si el administrador o cliente es un usuario legítimo. Una vez autenticado, las peticiones o comandos son después enviadas al planificador.

El **planificador** *(scheduler)* se encarga de leer las peticiones que se reciben por la API y seleccionar el pod que debe ejecutar el comando solicitado por el usuario. Para tomar esta decisión se basa en la cantidad de recursos solicitados por la petición. De tal forma que buscará el nodo menos saturado que cuente con esos recursos con el propósito de evitar sobresaturar un nodo más que otro. El planificador solamente se encarga de repartir la carga de trabajo entre los distintos nodos, por lo tanto, las tareas de despliegue, gestión o eliminación las realiza el elemento Kubelet de cada nodo.

El **controlador** se encarga de monitorizar el estado de los pods del clúster, prestando especial atención a los cambios que puedan ocurrirles. De tal forma que, por ejemplo, detecte cuando un pod ha terminado de forma abrupta o

inesperada. Cuando esto ocurra, lo replanifica indicando al planificador que inicie un proceso de recuperación, para desplegar un pod que sustituya al que haya dejado de funcionar.

Etcd es el sistema de almacenamiento persistente y distribuido que usa el master para guardar información del clúster. Por ejemplo, en *etcd* se almacena los cambios de estado que han experimentado los nodos de la red, los recursos disponibles en cada nodo, etc. Los demás elementos que componen el master se apoyan en los datos que se almacenan en *etcd* para realizar sus funciones.

14.4.1.3. Pods

Los *pods* son los elementos más básicos de Kubernetes. Cada uno de ellos conforma un microservicio y son tratados como servicios virtuales independientes. Normalmente, cada pod ejecuta una única aplicación sencilla y es el cúmulo de varios pod los que dan un servicio complejo al usuario final.

Los *pods* también sirven para crear una capa de separación entre el contenedor sobre el que se ejecuta el servicio virtualizado y el propio sistema de procesamiento del nodo. Haciendo que los contenedores se encuentren más aislados con el sistema del nodo y entre sí. De esta forma, el desarrollador sólo tiene que operar con la capa de Kubernetes.

14.4.1.4. Service

Asignar una conectividad IP tradicional puede llegar a ser una tarea complicada con unos elementos tan dinámicos como los pods. Si siguiera una conectividad tradicional, la interfaz de red se eliminaría cuando, por ejemplo, un pod terminara abruptamente y otro nuevo lo reemplazara. Bajo este pretexto, los paquetes en tránsito se perderían y habría que empezar un nuevo proceso de descubrimiento. Para evitar este problema existen los *Service*.

El *Service* es un elemento que se encarga de mantener la conectividad IP entre los distintos elementos del clúster. Su funcionamiento es sencillo, abstrae las interfaces de red de los pods. De tal forma que en vez de asignar direcciones IP a los pods, se les asigna a los *Services* y luego los pods se asocian a estos *Services*. Los ciclos de vida de estos elementos tampoco están relacionados. De esta forma, si un pod se reinicia, puede seguir manteniendo sus comunicaciones y su dirección IP, lo que permite una conectividad mucho más estable y robusta.

Esta solución permite gran versatilidad. Hace posible que haya más dinamismo dentro del clúster. Por ejemplo, con el propósito de aliviar la carga de trabajo de un pod, los *Service* permiten que se asocie otro pod al mismo *Service* y que este haga un balanceo de carga entre esos pods.

Los *Services* se dividen en dos tipos. Por un lado, los **internal Services** permiten la conectividad con los demás pods del clúster. A estos services normalmente se les asigna una dirección interna que no es enrutable desde internet. Por otro lado, los **external Services**, también llamados *LoadBalancers* en la documentación de Kubernetes, tienen asignada una dirección externa que es enrutable desde internet. Por estos *Services* es por donde entra y sale todo el tráfico del clúster.

14.4.1.5. Ingress

El Ingress es un elemento de la red del clúster que recuerda a un *DNS* de una red tradicional. Normalmente se considera como un elemento previo al *External Service*. Este elemento realiza la traducción de nombres de dominio a direcciones IP, de tal forma que los pods sean accesibles a través de una URL en vez de una dirección IP.

Adicionalmente, el Ingress añade una capa de seguridad a las comunicaciones, haciendo que el acceso a las distintas APIs sean por https en vez de por http.

14.4.1.6. ConfigMap

El *ConfigMap* es un elemento que se encarga mantener y administrar la configuración de la aplicación. Establece unos parámetros que el resto de elementos de la red deben cumplir. Por ejemplo, lista el rango de direcciones que se pueden asignar a un *external service*, asigna la URL por la que se deben conocer a los servicios desplegados, etc. Permite que estos parámetros sean modificables sin alterar el funcionamiento o ciclo de vida de los Pods.

14.4.1.7. Secret

ConfigMap guarda y define muchos parámetros de configuración. Sin embargo, al tratarse de un elemento accesible por cualquiera, no es el lugar indicado para almacenar información delicada, como credenciales. Para ese propósito existe *Secret*. El *Secret* es un elemento que almacena la información delicada de los servicios virtualizados del clúster, como los usuarios y las contraseñas. Para asegurar la confidencialidad, estos datos se guardan encriptados con un *encoder base64*.

14.4.1.8. Deployments

El administrador no se dedica a diseñar y definir cada uno de los pods del clúster. Eso es una tarea que realiza el master en función de la complejidad del servicio demandado por el administrador. En cambio, el administrador se encarga de diseñar el servicio a un nivel mayor de abstracción. Es decir, lo que el administrador diseña son los *Deployments* y no los pods.

El *Deployment* es un **descriptor yaml** del servicio a desplegar. Estos descriptores son modificables por el usuario para crear o modificar los servicios deseados. Permiten especificar el número de réplicas que se van a desplegar de un servicio, la imagen a desplegar, las credenciales de usuario que utilizar, los metadatos, etc. Adicionalmente, permite, una vez desplegado el servicio, realizar algún cambio en el *Deployment* y que este se refleje en el servicio ya desplegado.

14.4.1.9. Namespace

El *Namespace* es una forma de organizar el clúster de Kubernetes. Divide a los usuarios del clúster en distintos grupos y les asigna un nombre, un *Namespace*. Para un usuario de un equipo son visibles todos los servicios desplegados por el resto de miembros de su equipo, pero no son visibles los de los demás equipos. Esto permite trabajar a multitud de usuarios sobre un mismo clúster sin miedo a sobrescribir el trabajo de otro usuario ajeno a su proyecto.

14.4.1.10. Volúmenes

Kubernetes no cuenta con un sistema de almacenamiento persistente. Para almacenar datos que no deben perderse se debe acudir a software de terceros o a un sistema de almacenamiento físico. Del mismo modo, deben ser gestionado mediante herramientas y tecnologías ajenas a Kubernetes. Una vez el sistema de almacenamiento está montado, Kubernetes hace uso de **descriptores yaml** creados por el usuario para asociar volúmenes de almacenamiento a los pods desplegados que los necesiten.

14.4.1.1. Statefulset

Cuando se vaya a desplegar un servicio que requiera de almacenamiento persistente se usa un *Statefulset* en vez de un *Deployment*. El funcionamiento de *Statefulset* es análogo al del *Deployment*, con el añadido de que permite a los pods y sus réplicas acceder a los datos de una forma ordenada. De esta forma, se evita que dos pods diferentes escriban sobre el mismo dato simultáneamente o que ocurra cualquier otro tipo de inconsistencia de la información.

14.4.2. Helm-chart

En el diseño inicial de Kubernetes, cada vez que un usuario pretendía desplegar un nuevo servicio debía previamente diseñar, configurar y desplegar todos y cada uno de los elementos del clúster. Con el propósito de evitar este procedimiento tedioso cada vez que se vaya a desplegar un nuevo servicio, se crearon los *Helm-chart*.

Los *Helm-chart* consisten en un grupo de descriptores *yaml* estandarizados y listos para un despliegue genérico de un servicio concreto. Están planteados para dar un nivel de abstracción aún mayor al usuario y que este sólo tenga que descargar un Helm-chart y desplegarlo para tener un servicio en funcionamiento, sin tener que detenerse a

diseñar o configurar ningún elemento de Kubernetes. A la hora de desplegar estos descriptores genéricos, Helm-chart admite parámetros de personalización para ajustar mejor el servicio a las necesidades del usuario.

La comunidad de Kubernetes, aprovechándose de las ventajas que trae esta funcionalidad, ha creado una gran cantidad de repositorios de Helm-charts, como *bitnami* o *stable* entre otros. Dentro de cada repositorio hay varios Helm-chart de aplicaciones software populares, por ejemplo: openldap, mongodb, mysql, moodle, etc. A parte de la gran variedad que ofrece la comunidad, también existe la posibilidad de que cada usuario cree su propio Helm-chart. De esta forma, le será mucho más sencillo desplegar las aplicaciones propietarias que desee.