Craft

Synopsis

Craft is a medium difficulty Linux box, hosting a Gogs server with a public repository. One of the issues in the repository talks about a broken feature, which calls the eval function on user input. This is exploited to gain a shell on a container, which can query the database containing a user credential. After logging in, the user is found to be using vault to manage the SSH server, and the secret for which is in their Gogs account. This secret is used to create an OTP which can be used to SSH in as root.

Skills

- Python code review
- Linux enumeration
- GIT
- Python eval injection
- Vault SSH

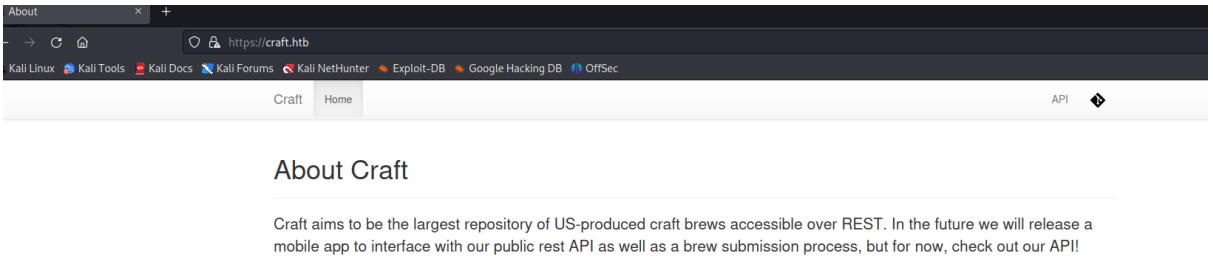
Exploitation

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

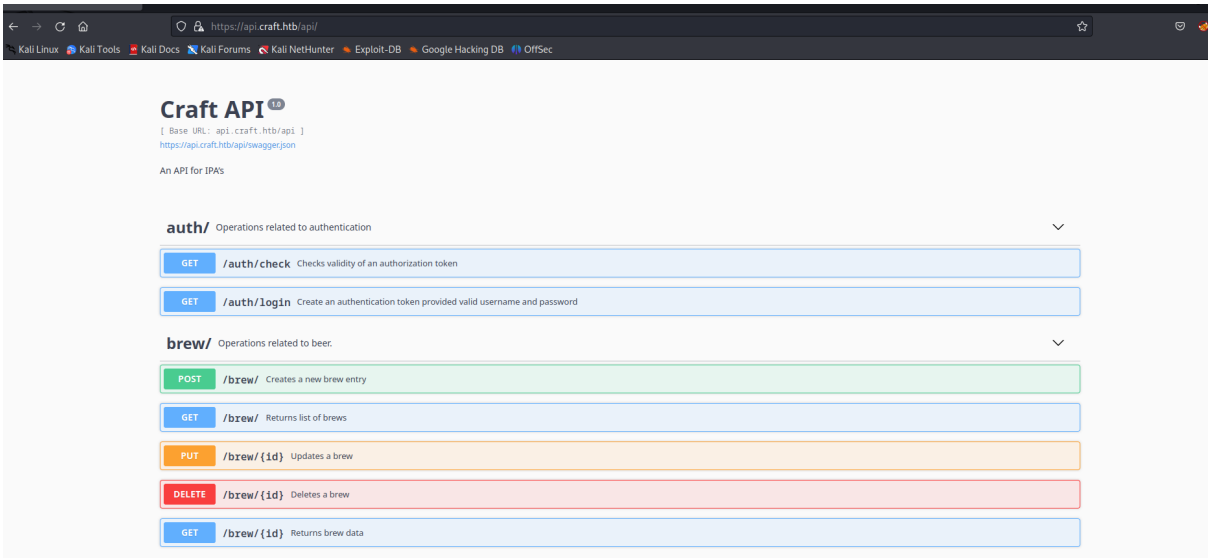
```
└─# nmap -A 10.10.10.110
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-11 06:06 EDT
Stats: 0:03:31 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 99.99% done; ETC: 06:09 (0:00:00 remaining)
Nmap scan report for 10.10.10.110
Host is up (0.088s latency).
Not shown: 998 closed tcp ports (reset)
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
| ssh-hostkey:
|   2048 bde76c22817adb3ec0f0731df3af7765 (RSA)
|   256 82b5f9d1953b6d800f3591862db3d766 (ECDSA)
|_  256 283b2618ecdfb336859c27548d8ce133 (ED25519)
443/tcp open  ssl/http nginx 1.15.8
|_http-title: About
| tls-nextprotoneg:
|_  http/1.1
| ssl-cert: Subject: commonName=craft.htb/organizationName=Craft/stateOrProvinceName=NY/countryName=US
| Not valid before: 2019-02-06T02:25:47
|_Not valid after:  2020-06-20T02:25:47
| tls-alpn:
|_  http/1.1
|_ssl-date: TLS randomness does not represent time
|_http-server-header: nginx/1.15.8
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=7.93%E=4%D=8/11%OT=22%CT=1%CU=35647%PV=Y%DS=2%DC=T%G=Y%TM=64D6091
OS:1%P=x86_64-pc-linux-gnu)SEQ(SP=106%GCD=1%ISR=108%TI=Z%CI=Z%II=I%TS=8)OPS
OS:(O1=M53CST11NW7%O2=M53CST11NW7%O3=M53CNNT11NW7%O4=M53CST11NW7%O5=M53CST1
OS:1NW7%O6=M53CST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN
OS:(R=Y%DF=Y%T=40%W=7210%O=M53CNNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=A
OS:S%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R
OS:=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F
OS:=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%
```

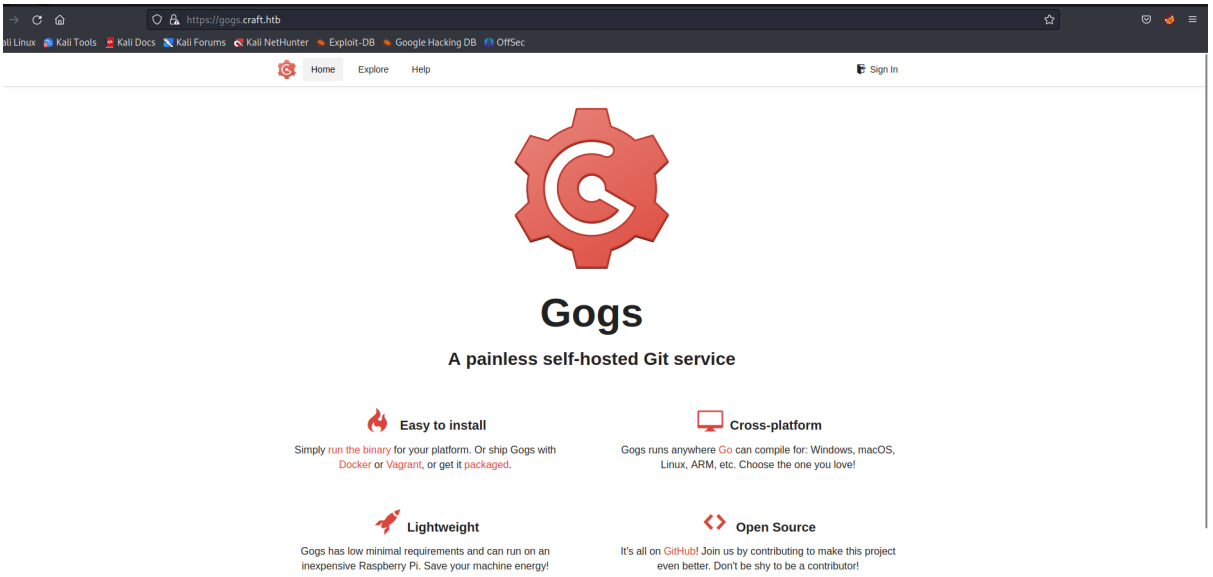
We see only two ports open, so let's start from the web ports

Opening the browser have us the following page

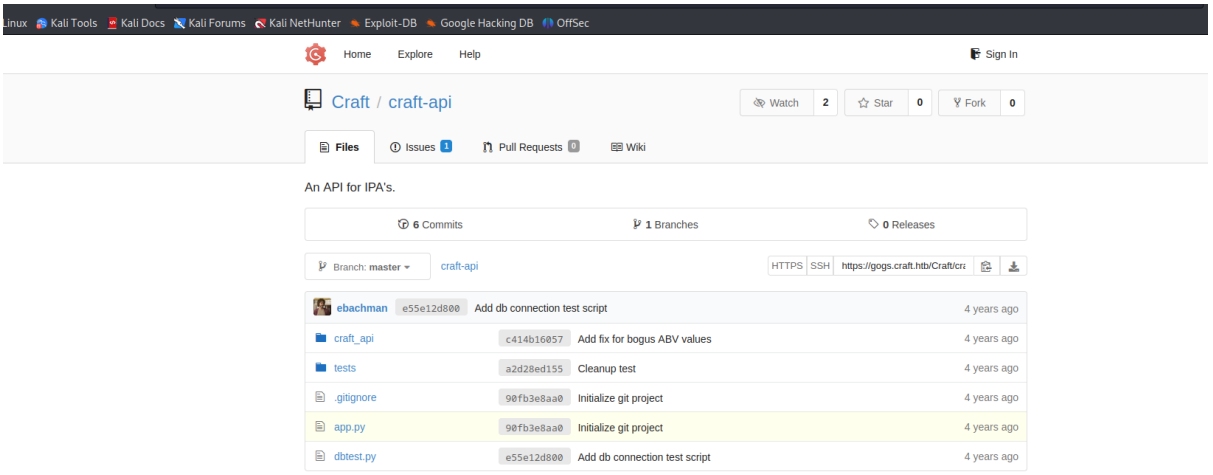


From that page we can access API docs as well as GOGS

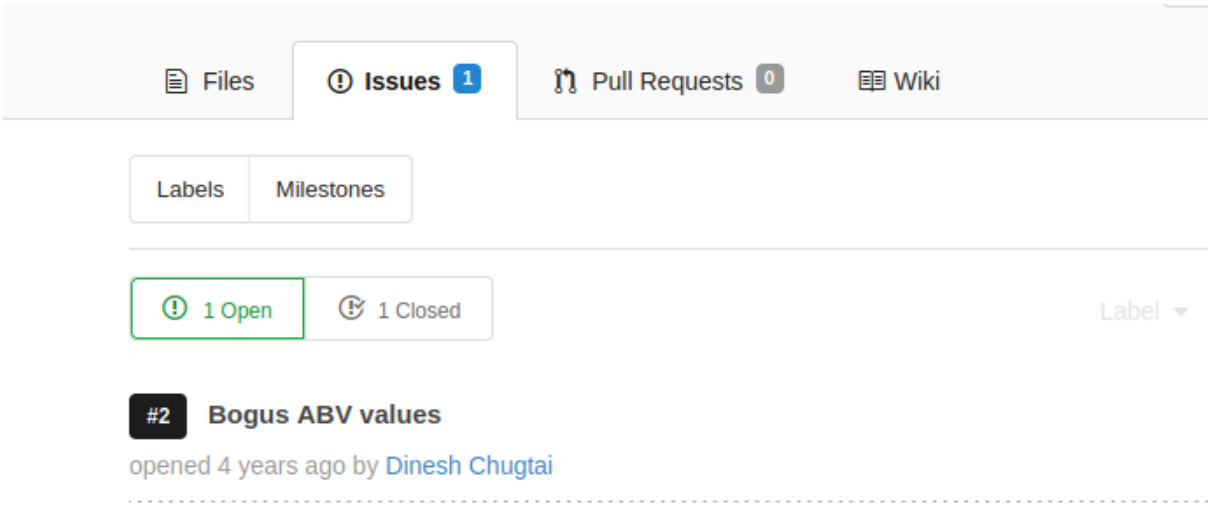




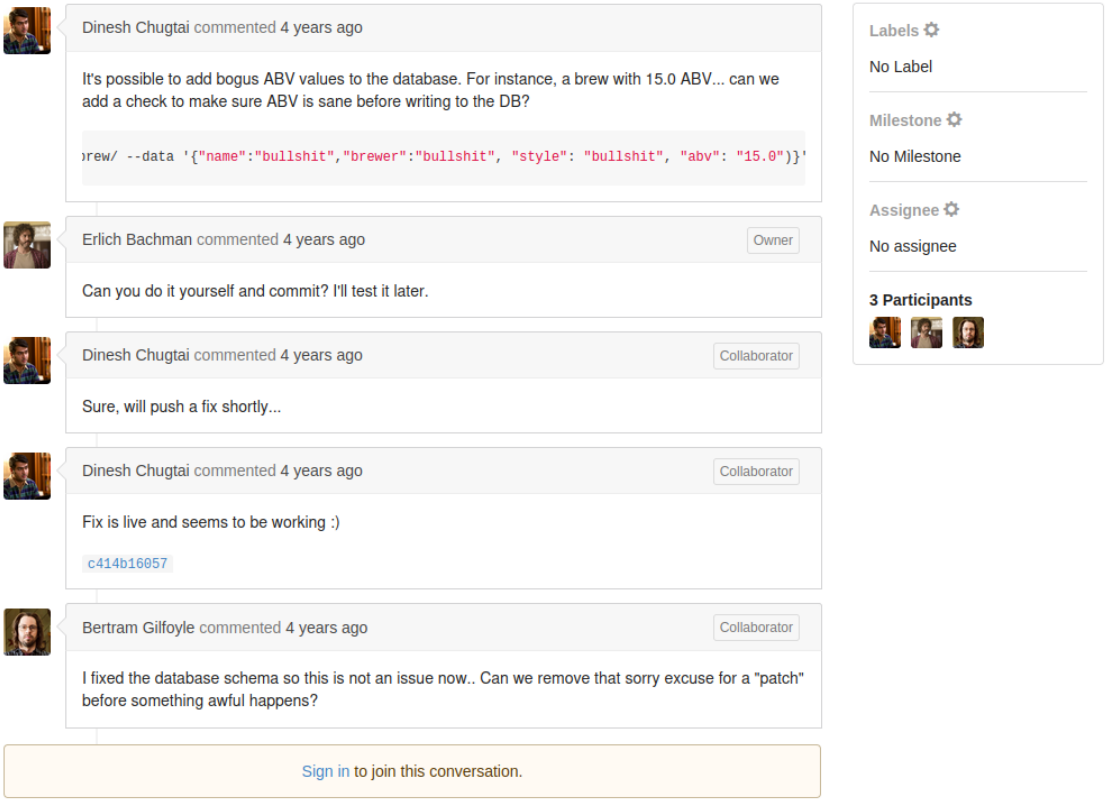
There is not much we can do with API right now, so we started exploring GOGS where we found a python source code of the application



And there was something in the Issuess tab,

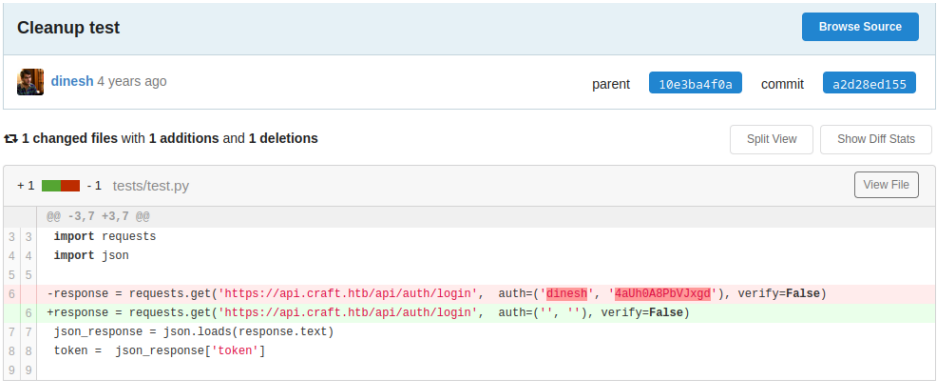


Closer look of the issue informed as about problem with the “ABV” which is taken from the user and that user authentication is handled by the JWT token

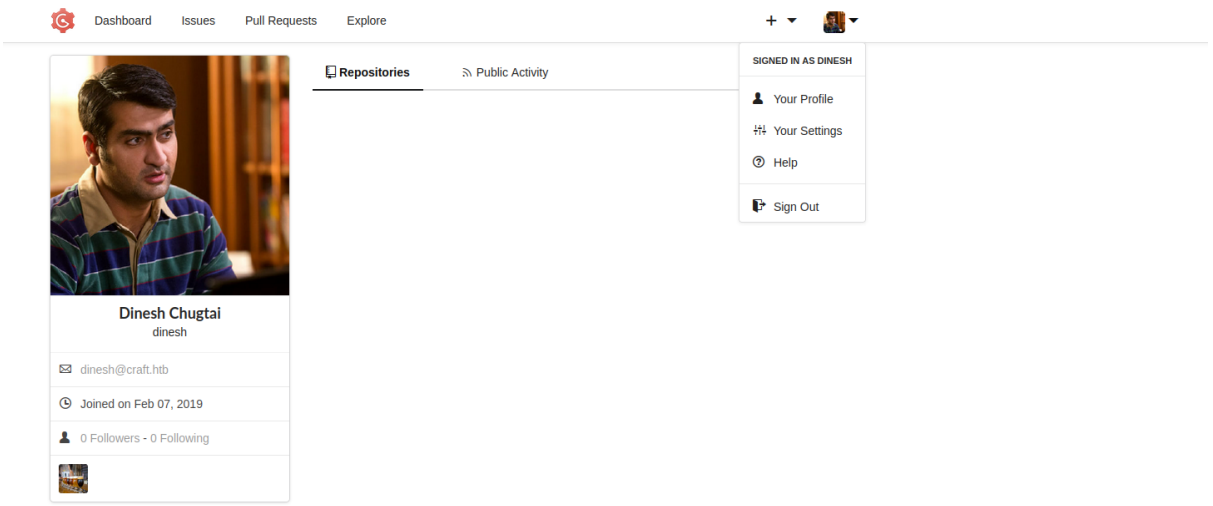


Inspecting the problematic code showed us that the ABV value is taken from the user and not sanitised and also put in the EVAL function, (eval is a dangerous function because abused can lead to the remote code execution)

We continued the enumeration of the repository and in the history of commits we found credentials for a user Danesh

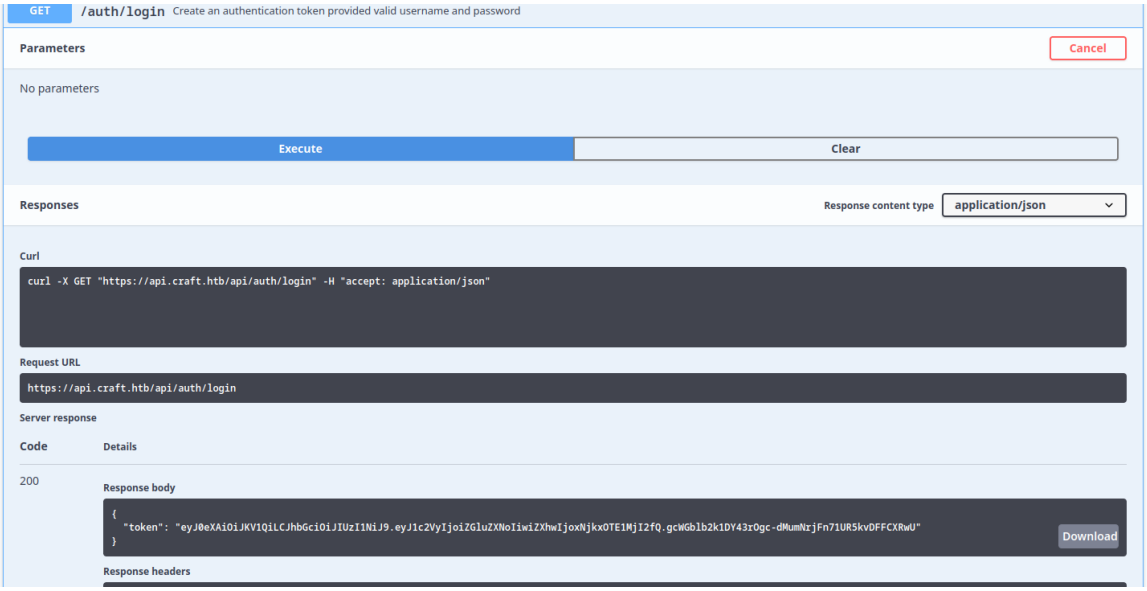


We logged as this user to the GOGS

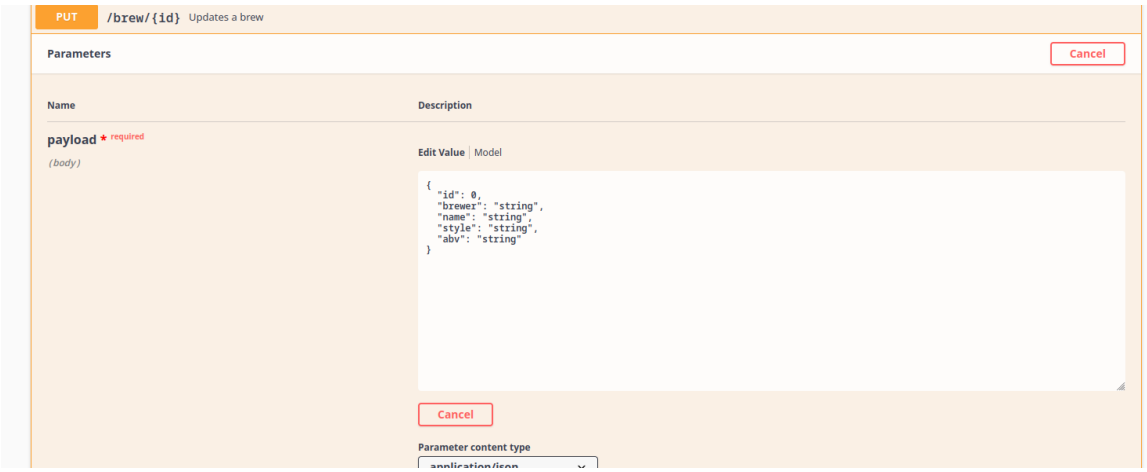


But it didn’t give us any new accesses

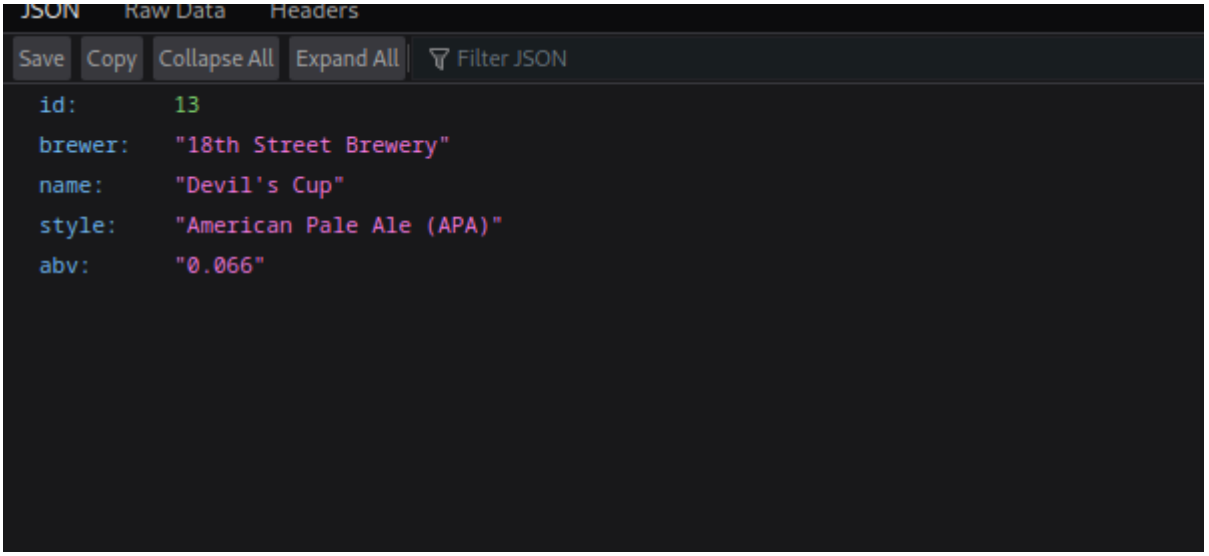
Yet, as we remembered the authentication to the application is handled by JWT token, so we returned to the API and with Danesh credentials we generated the new JWT token



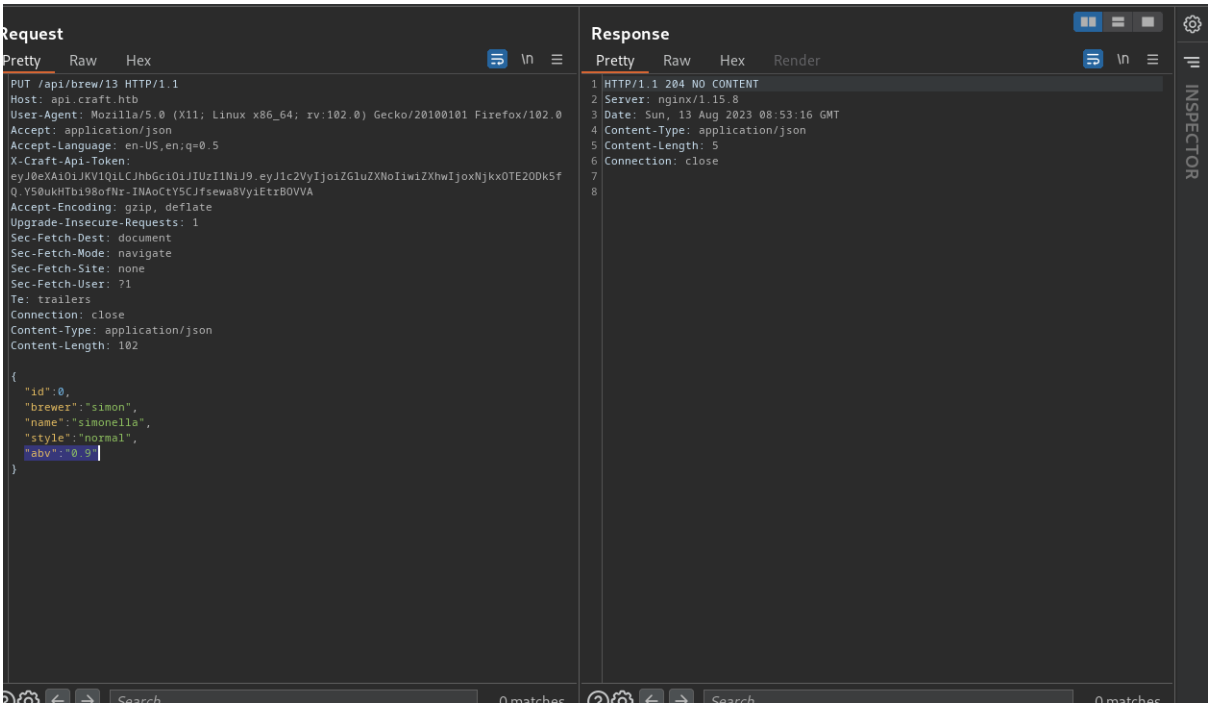
API offers PUT method to update values, including the value of ABV (which we know is vulnerable to remote code execution due to presence of dangerous function and unsanitized user's input)



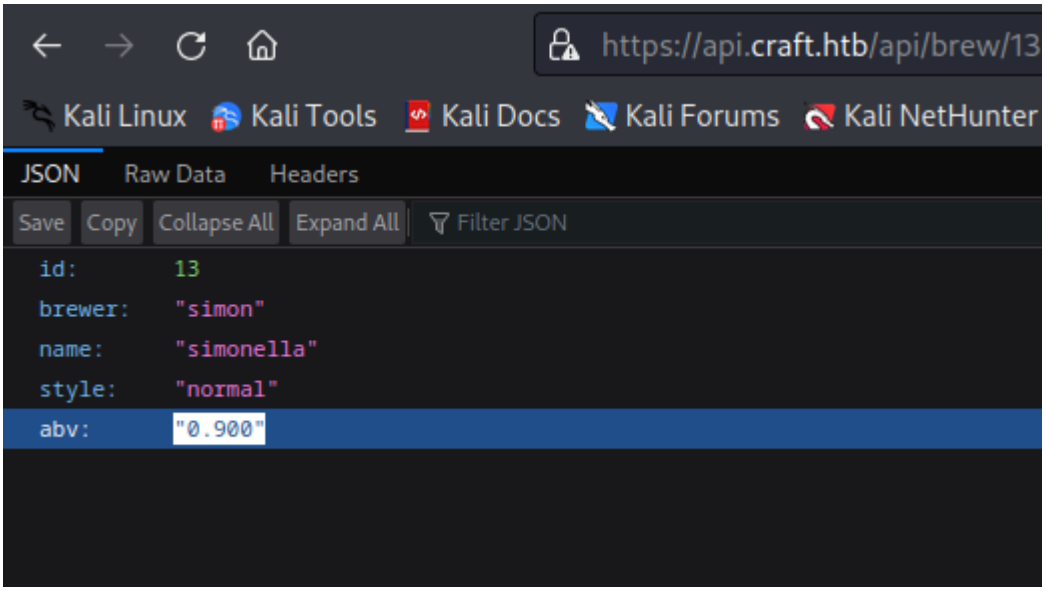
The original value for the brew is as follow



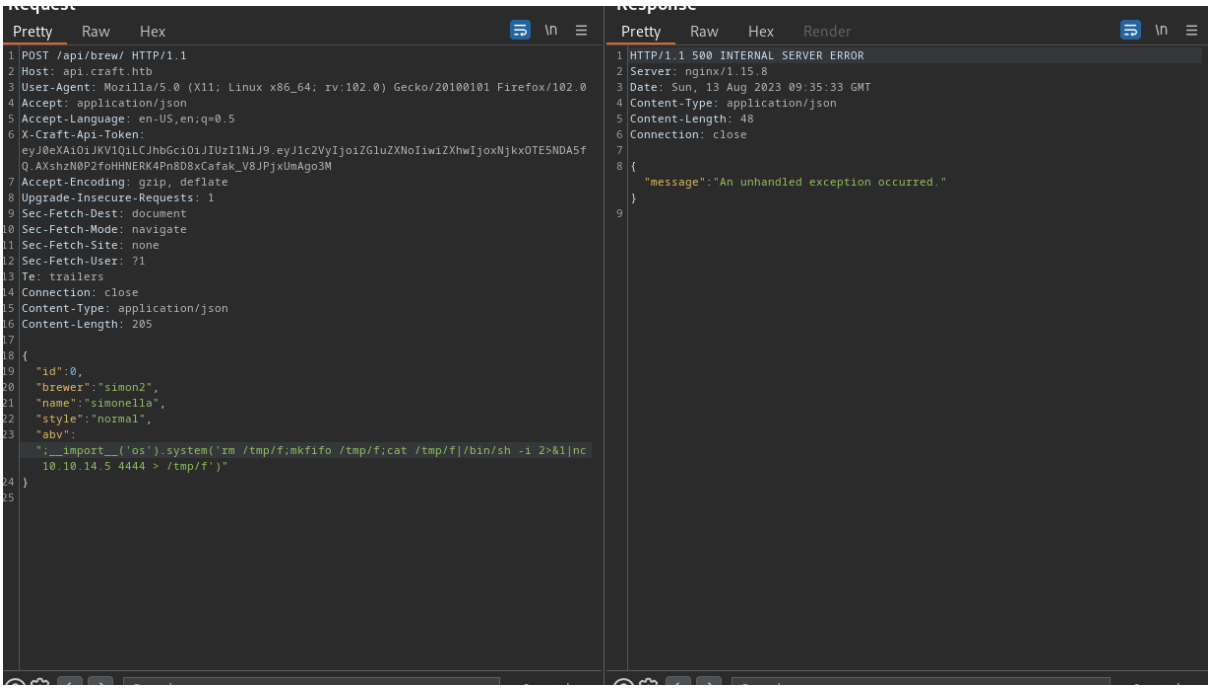
We sent the request to update this value (our request contains forged JWT token to ensure that we sent the request as Danesh)



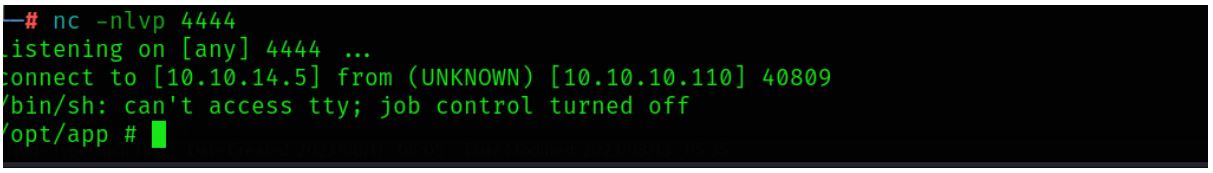
And our malicious request was accepted - the values were updated

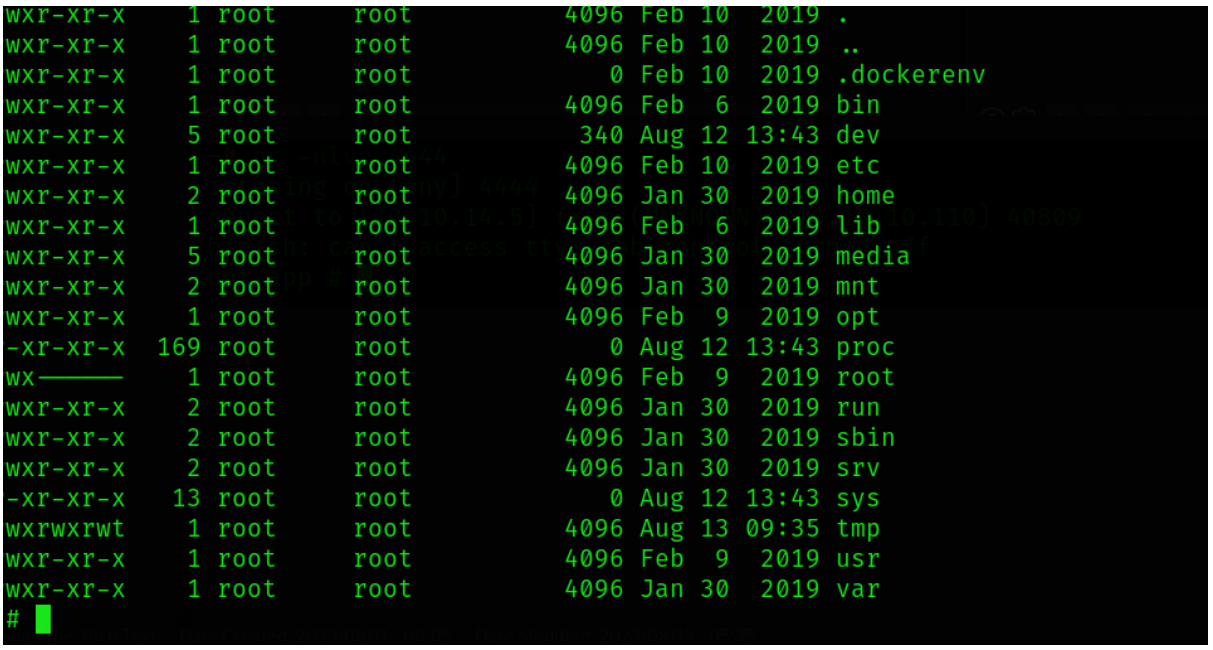


The next step was to get a reverse shell on the system, by passing a malicious command as the ABV value



And we got on the system




wxr-xr-x 1 root root 4096 Feb 10 2019 .
wxr-xr-x 1 root root 4096 Feb 10 2019 ..
wxr-xr-x 1 root root 0 Feb 10 2019 .dockerenv
wxr-xr-x 1 root root 4096 Feb 6 2019 bin
wxr-xr-x 5 root root 340 Aug 12 13:43 dev
wxr-xr-x 1 root root 4096 Feb 10 2019 etc
wxr-xr-x 2 root root 4096 Jan 30 2019 home
wxr-xr-x 1 root root 4096 Feb 6 2019 lib
wxr-xr-x 5 root root 4096 Jan 30 2019 media
wxr-xr-x 2 root root 4096 Jan 30 2019 mnt
wxr-xr-x 1 root root 4096 Feb 9 2019 opt
-xr-xr-x 169 root root 0 Aug 12 13:43 proc
wx------ 1 root root 4096 Feb 9 2019 root
wxr-xr-x 2 root root 4096 Jan 30 2019 run
wxr-xr-x 2 root root 4096 Jan 30 2019 sbin
wxr-xr-x 2 root root 4096 Jan 30 2019 srv
-xr-xr-x 13 root root 0 Aug 12 13:43 sys
wxrwxrwt 1 root root 4096 Aug 13 09:35 tmp
wxr-xr-x 1 root root 4096 Feb 9 2019 usr
wxr-xr-x 1 root root 4096 Jan 30 2019 var
#