Report by Chris H.
Date of Completion: 3-28-19

Note: This report details a penetration test conducted on a virtual system hosted on https://www.hackthebox.eu/ . This system was a lab designed to practice penetration testing techniques, and is not a real-world system with PII, production data, etc.

Target Information

Name	Help
IP Address	10.10.10.121
Operating System	Linux

Tools Used

- Operating system: Kali Linux – A Linux distribution designed for penetration testing
- OpenVPN – An open-source program used for creating a VPN connection to hackthebox.eu servers, which allows for connection to the target.
- Nmap – A network scanner used to scan networks and systems. Discovers hosts, services, OS detection, etc.
- OWASP Dirbuster – A tool that brute forces directories of a webpage and discovers pages and files.
- pwnyShell.php – A simple php web shell that allows for RCE
- exploit.py (https://www.exploit-db.com/exploits/40300) – An python script that takes advantage of a HelpDeskZ arbitrary file upload vulnerability and calculates the location of the the file upload.
- upstream44.c – A kernel exploit used on the linux system that spwns a root shell

Executive Summary

Help is a virtual system hosted on https://www.hackthebox.eu/. I conducted this penetration test with the goal of determining the attack surface, identifying the vulnerabilities and attack vectors, exploiting the vulnerabilities, and gaining root access to the system. All activities were conducted in a manner simulating a malicious threat actor attempting to gain access to the system.

The goal of the attack was to retrieve two files:

1) user.txt – A file on the desktop (Windows) or in the /home directory (Linux) of the unprivileged user. Contents of the file are a hash that is submitted for validation on hackthebox. Successful retrieval of this file is proof of partial access/control of the target.

2) root.txt – A file on the desktop (Windows) or in the /home directory (Linux) of the root/Administrator account. This file contains a different hash which is submitted for validation on hackthebox. Successful retrieval of this file is proof of full access/control of the target.

Summary of Results

This machine was fairly straightforward. After simple enumeration, it is discovered that port 80 (HTTP) is open. A quick dirbuster scan reveals a support page where I was able to submit tickets. The ticket had an optional “Upload File” button, which allowed for a PHP shell to be uploaded. Using an exploit, I was able to locate the page in which the file was stored, and access my PHP shell for code execution. From there, I could get the user.txt flag. I then used a kernel exploit called “upstream44.c” to elevate to root, and get the root.txt flag.

Attack Narrative

The first step with any box is to enumerate and gather as much information as possible. I started with `nmap -sV -A -vv 10.10.10.121` (full results are located in Appendix A). This reveals open ports 22 (SSH), 80 (HTTP), and 3000. Since there are no notable ports with easy exploits (SMB, FTP, etc.) I visited http://10.10.10.121/ to see its webpage. This revealed a default apache webpage, so I decided to start a dirbuster scan on the site.

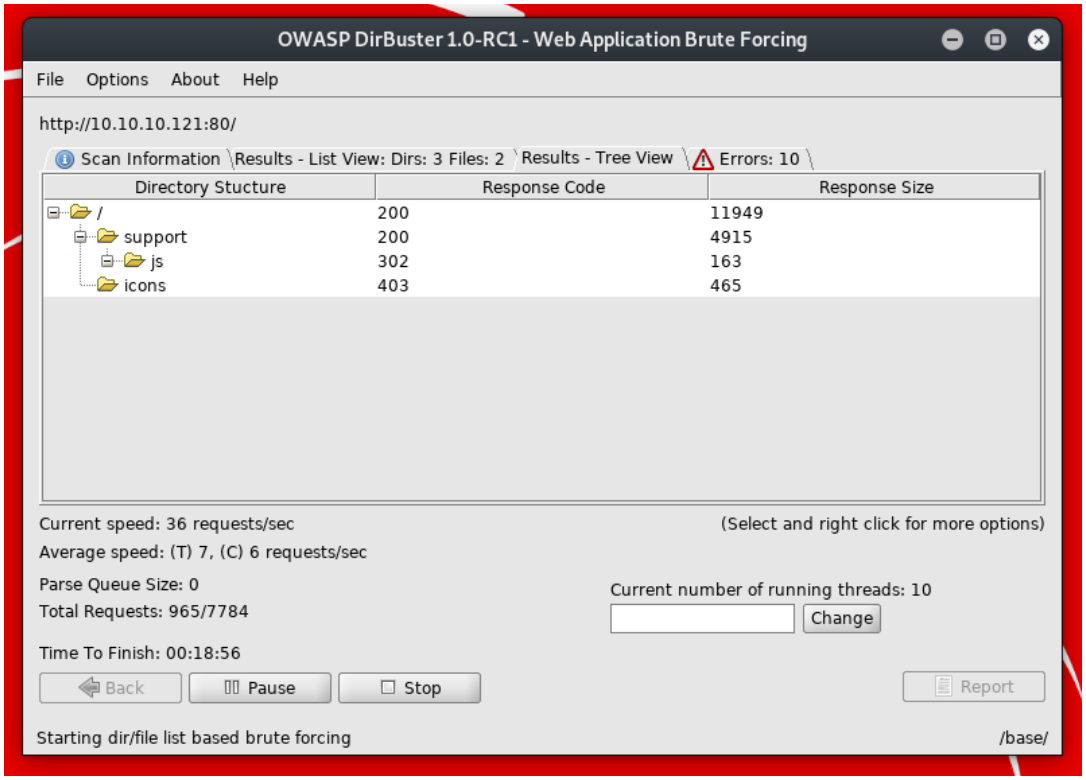

Figure 1

As seen by the output of the dirbuster scan (Figure 1), the tool finds /support with a 200 response code, and a decent response size. This indicates that the page is likely significant.

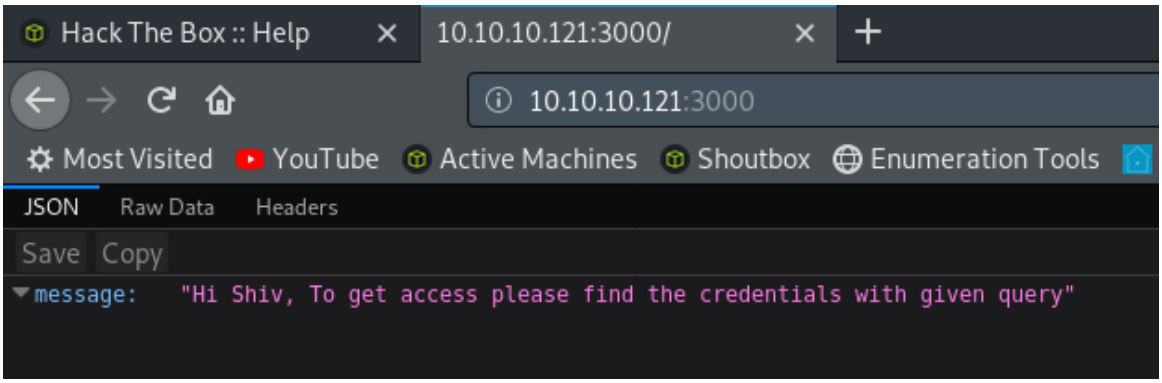

Figure 2

Before checking /support, I visited port 3000 while the dirbuster scan finished. Viewing this page shows the message in Figure 2, which hints at a possible query somewhere to find credentials. This ended up being irrelevant to the compromise of the machine, but still noteworthy.

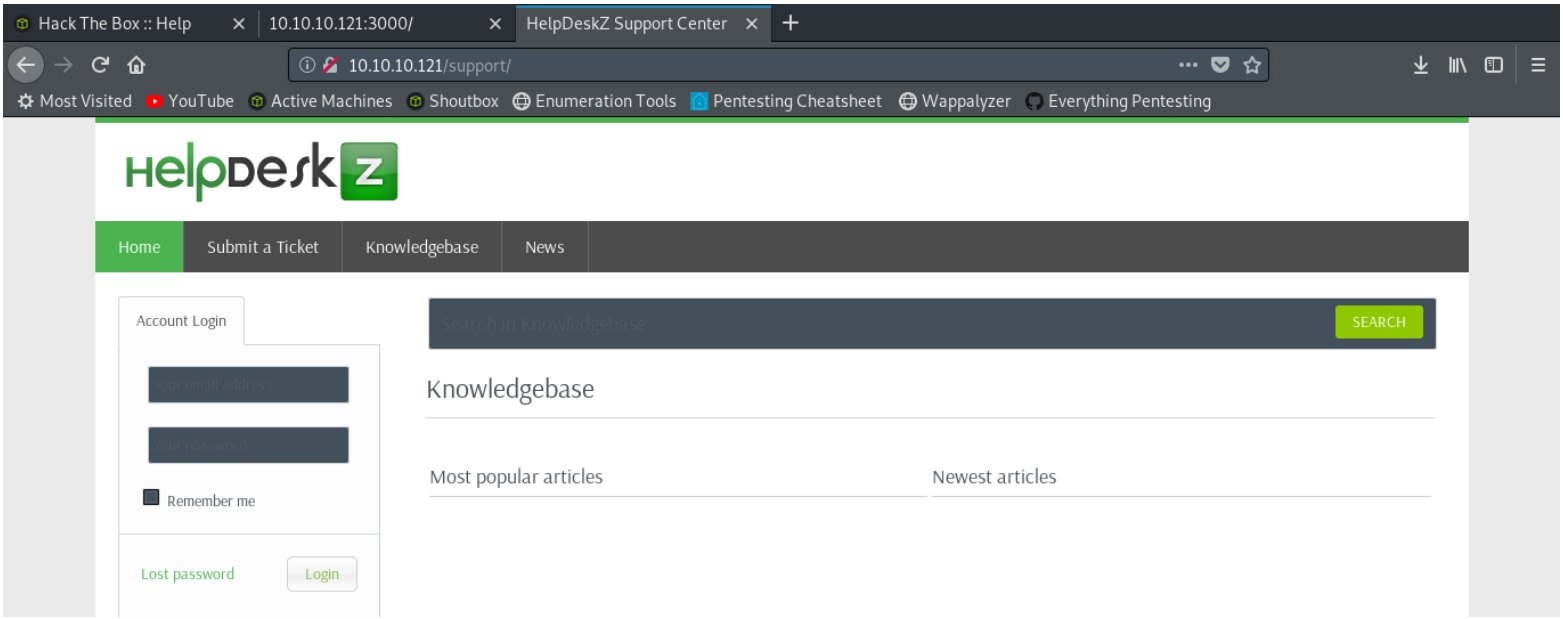

Figure 3

Moving to the http://10.10.10.121/support/ page discovered by dirbuster, I am greeted with a HelpDeskZ support page. “Submit a Ticket” immediately catches my attention, since this could lead to a file upload vulnerability being exploited.

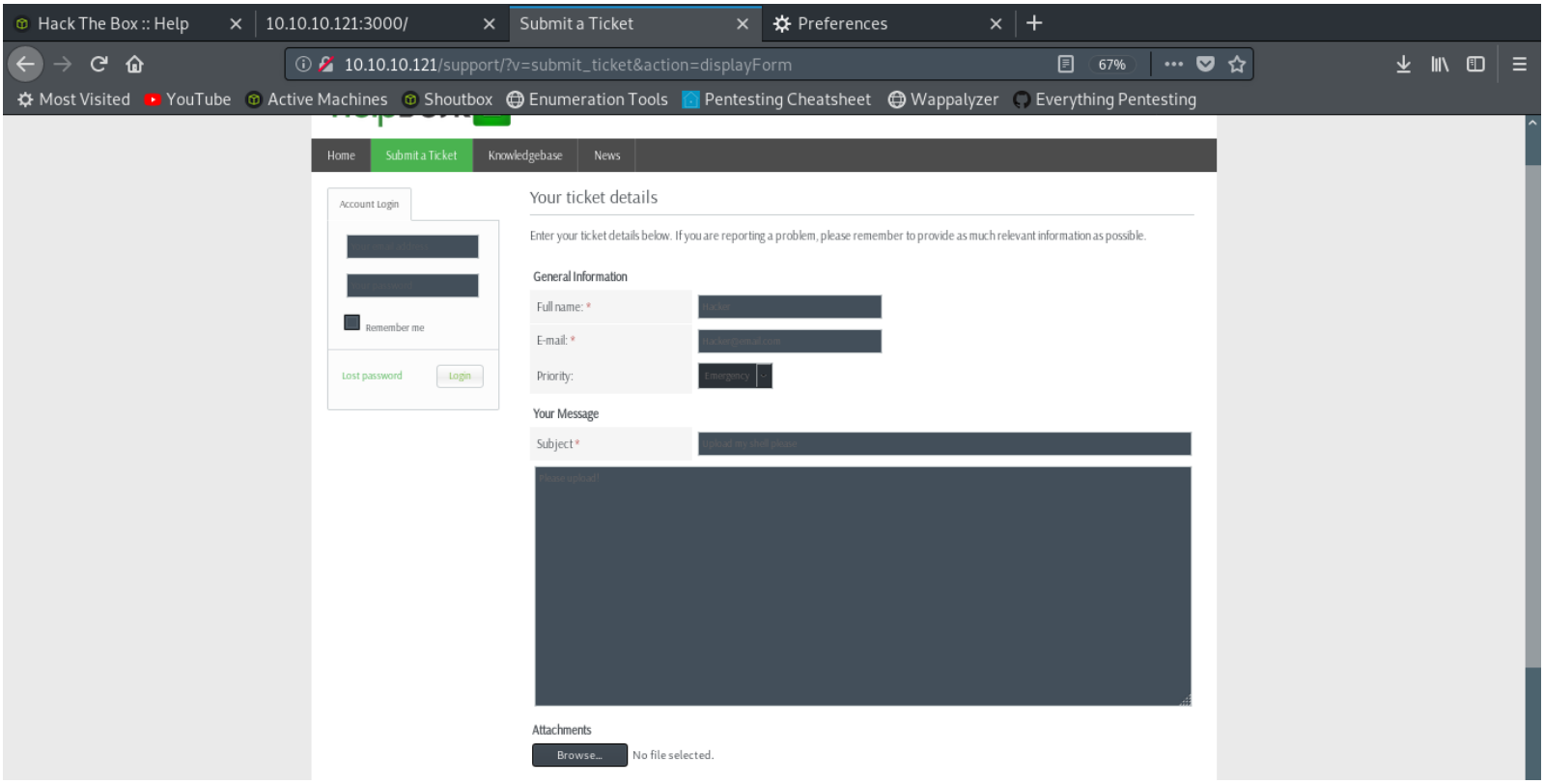

Figure 4

Sure enough, the upload ticket area allows for a file to be attached to the ticket (bottom of Figure 4). A Google search for “HelpDeskZ exploit” reveals many exploits, one of which allows for an arbitrary file upload and gaining RCE on a target. Given the objective, this could be a good way to gain a foothold on the system.

```
if(!isset($error_msg) && $settings['ticket_attachment']==1){
        $uploaddir = UPLOAD_DIR.'tickets/';
        if($_FILES['attachment']['error'] == 0){
                $ext = pathinfo($_FILES['attachment']['name'], PATHINFO_EXTENSION);
                $filename = md5($_FILES['attachment']['name'].time()).".".$ext;
                $fileuploaded[] = array('name' => $_FILES['attachment']['name'], 'enc' => $filename
                $uploadedfile = $uploaddir.$filename;
                if (!move_uploaded_file($_FILES['attachment']['tmp_name'], $uploadedfile)) {
                        $show_step2 = true;
                        $error_msg = $LANG['ERROR_UPLOADING_A_FILE'];
                }else{
```

Figure 5

HelpDeskZ’s source code is available online, and checking the code for how the application handles ticket uploads (Figure 5), it is seen that tickets are stored in “UPLOAD_DIR.’tickets/’. While the upload directory is not known, dirbuster was able to find /support/uploads/ as a location. So, it can be inferred that /support/uploads/tickets is the location of storage.

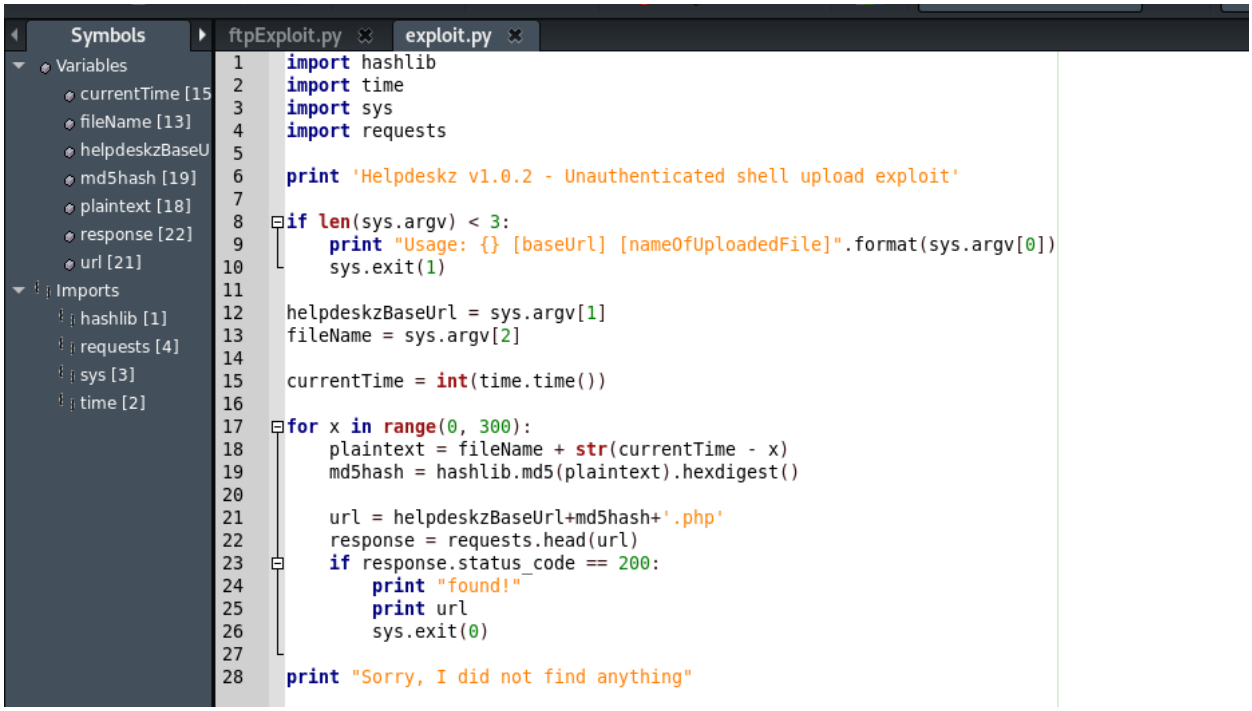

Figure 6

This exploit (Figure 6) was found on exploitdb, and allows the attacker to find the upload location of a web shell that they upload. Referring back to Figure 5, line 5, it is shown the HelpDeskZ uses an MD5 hash of the attachment name combined with the PHP time() function. This function returns the number of seconds since 1/1/1970, also known as unix time. For example, Christmas Day, 2015 at 8:00:00AM converted with the time() function, is 1451030400. If a file named “example.php” was uploaded at that time, HelpDeskZ would name it as follows:

MD5 [“example” + “1451030400”] + “.php”

= 332c2c10a79749e418b85e4ea03a8b27.php

It would then be stored in the location /support/uploads/tickets/. So, the full location would be:

http://10.10.10.121/support/uploads/tickets/332c2c10a79749e418b85e4ea03a8b27.php.

However, this is extremely hard to replicate manually since guessing the exact time of upload as well as having the target’s time synched to the attacking machine makes finding the upload very difficult. Lucky, the script in Figure 6 does the work on its own. It essentially creates a custom wordlist from the hashed name and time, then performs a very precise dirbuster until it reaches a 200 response. This indicates that it has found the upload location.

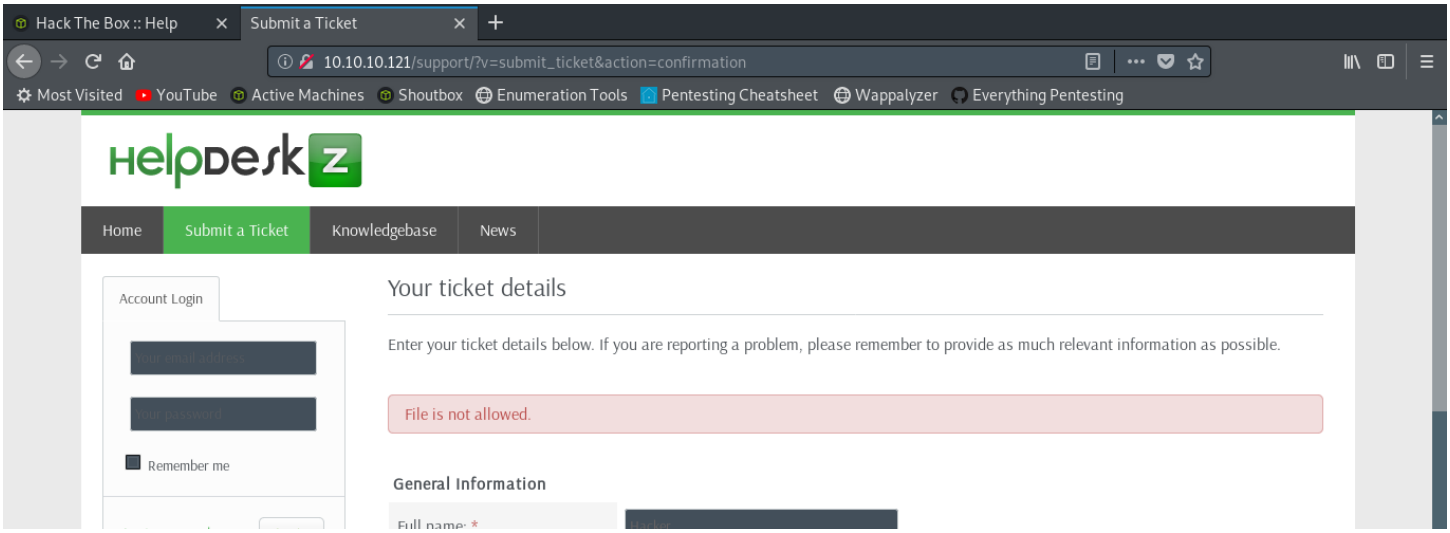

Figure 7

Before using the script, the shell must be uploaded. I used the pwnyShell.php file since this is a very simple and easy php shell. Uploading the shell give this error (Figure 7), however, this is for display only. Online searches reveal that despite the “File is not allowed” error, the file is still processed and stored.

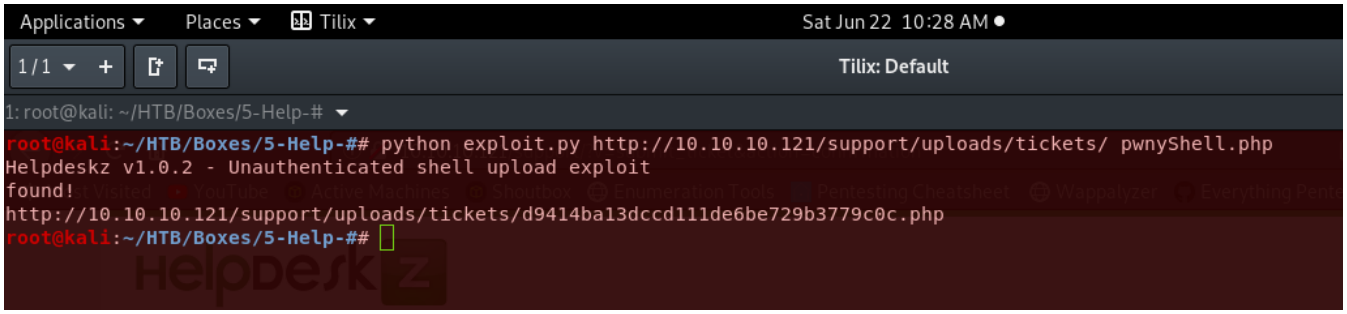

Figure 8

Now, the script can be run (Figure 8). The arguments are 1) the upload directory location, and 2) the name of the file uploaded. After about 30 seconds, the script finds 200 response and displays what URL caused it. The shell is now located.

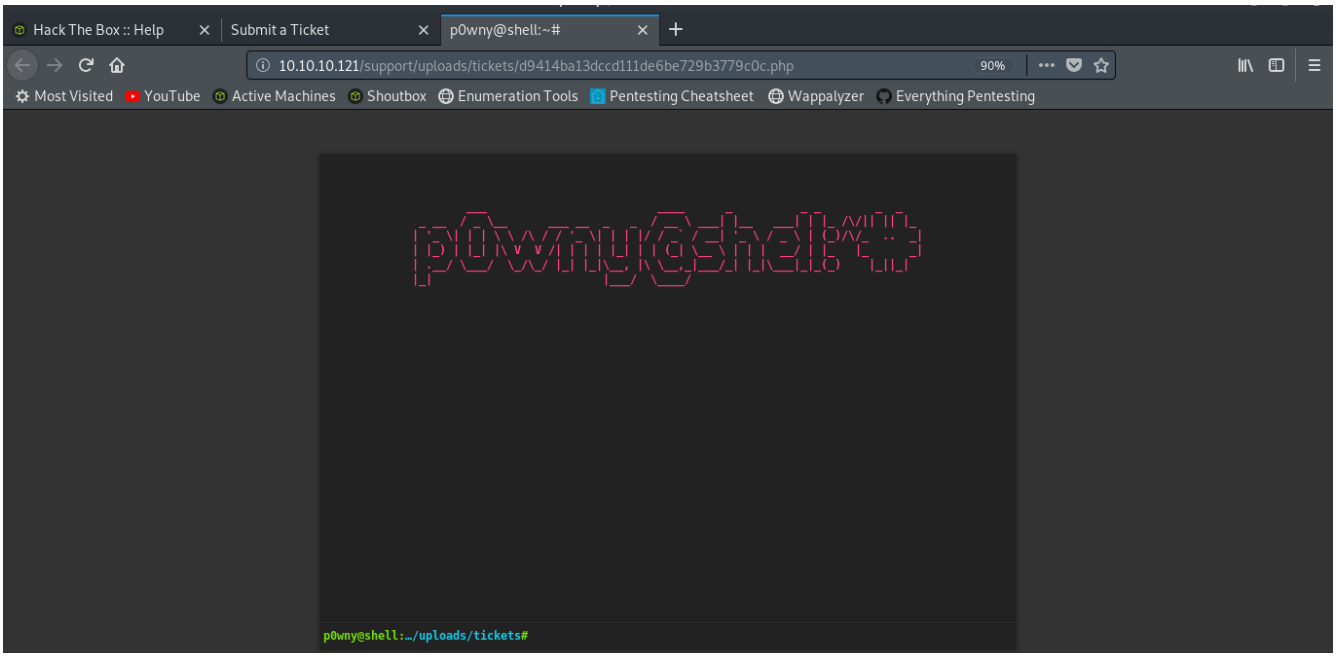

Figure 9

Following the link, the pwnyShell is working (Figure 9). I now have RCE capabilities on the machine.

```
p0wny@shell:/# cd home

p0wny@shell:/home# ls
help

p0wny@shell:/home# cd help

p0wny@shell:/home/help# ls
a
exploit1.c
help
npm-debug.log
user.txt

p0wny@shell:/home/help# cat user.txt
bb8a7b36bdce0c61ccebaa173ef946af

p0wny@shell:/home/help#
```

Figure 10

Changing directories to /home/help, and an ls, this reveals the user.txt flag (Figure 10). Note: the files: a, exploitl.c, help, and npn-debug.log were not placed by me. Since there are others attacking the machine, files are sometimes left behind or placed in the wrong locations.

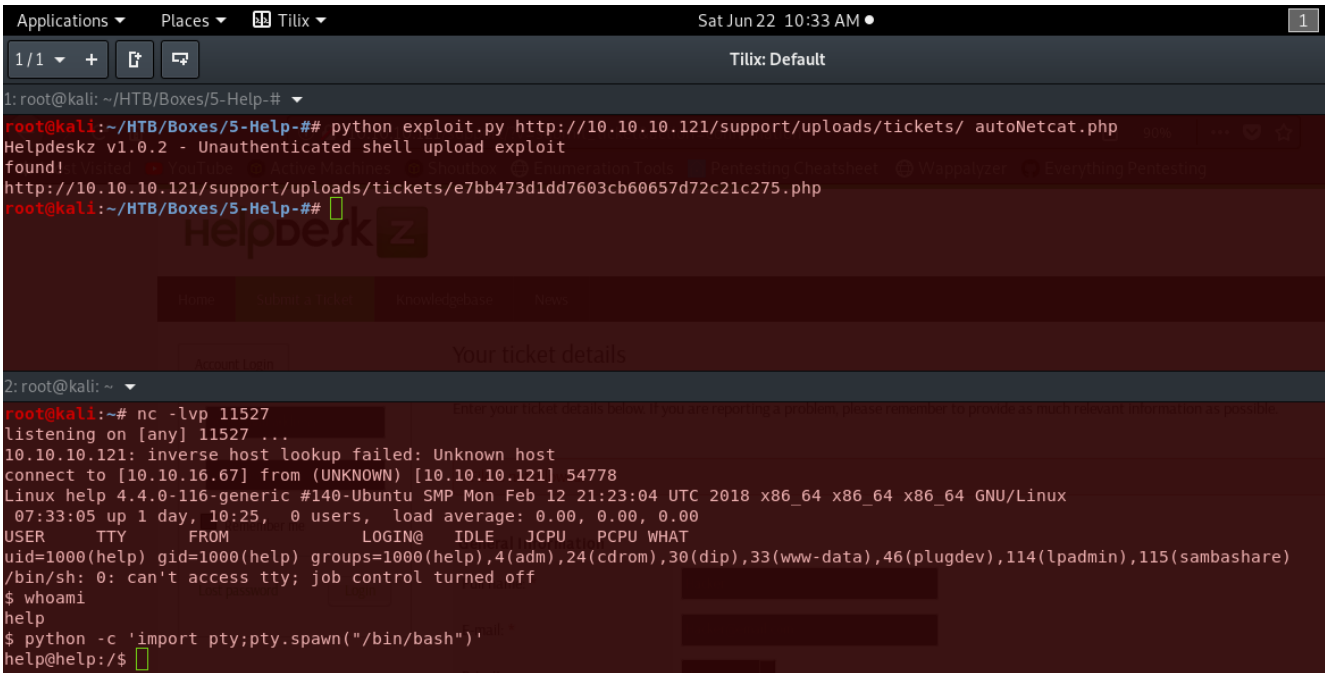

Figure 11

To make this easier, I decided to use my terminal for the RCE instead of the pwnyShell. To do so, I used the ticket upload exploit to upload a php program that executes a netcat to the specified IP address and port number in its code. I modified the code to include my IP and listening port number, uploaded, ran the same exploit, found the location, and opened the link. Doing so caused a connection to be made to my system (Figure 11). Using `whoami` revealed that I am the user help. I then performed a shell upgrade using `python -c 'import pty;pty.spawn(*/bin/bash*)'` to obtain a TTY shell.

Using the command `uname -a` it is revealed that this system is running on 4.4.0-116-generic. A search online shows that this is susceptible to a kernel exploit named upstream44.c, so the exploit code is downloaded and saved.

Repeating the steps used for uploading both php files, I can upload upstream44.c. However, the exploit used to find this is not needed since it is not executed like a php shell.

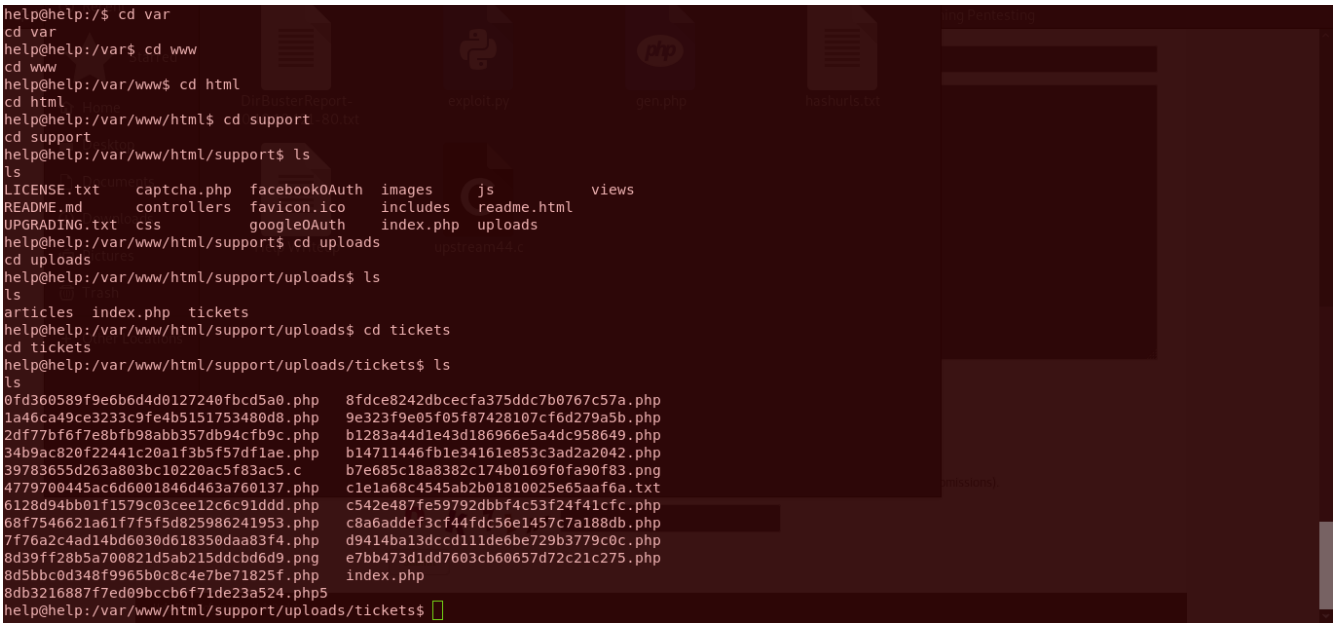

Figure 12

In order to execute the kernel exploit, it must be found first. Since it was uploaded as a ticket, it can be found in the directory: /var/www/html/support/uploads/tickets. An ls command shows dozens of uploaded files, most of which are php files. These came from other people attacking the machine and uploading php shells. However, there is one file with a .c extension (lower left Figure 12, fifth file), and it can be assumed that this is the renamed upstream44.c exploit.

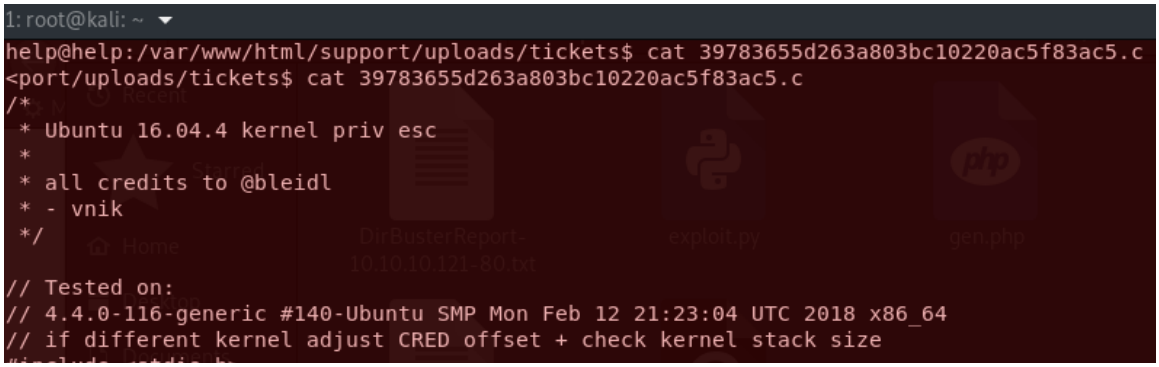

Figure 13

To verify that this is indeed the upstream44.c file, cat is used to print it (Figure 13). Sure enough, it is.

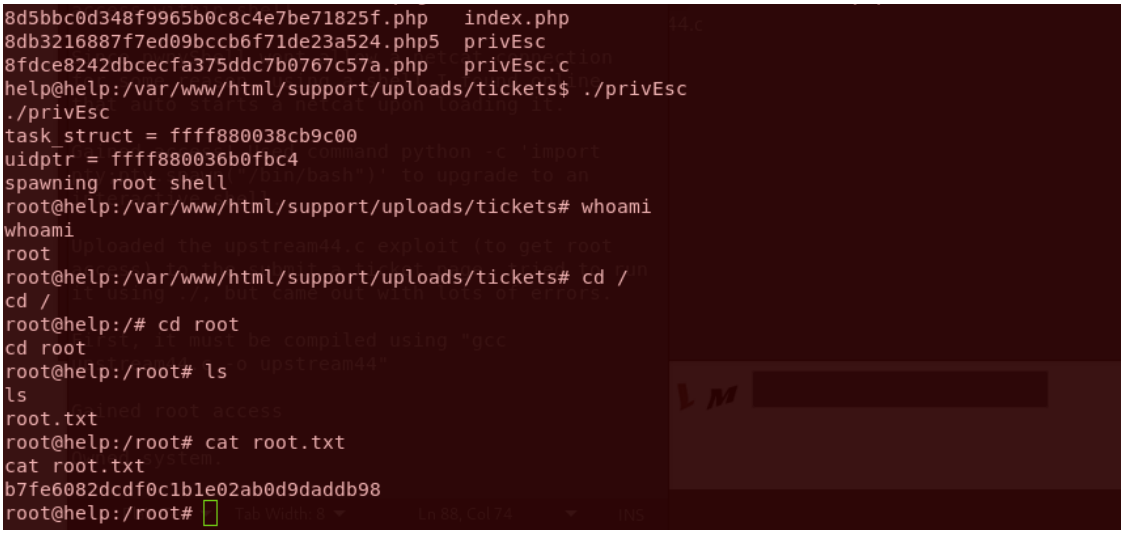

Figure 14

To make life easier, I renamed the file privEsc.c so that I could avoid copy-pasting a hash multiple times. Now, the file is compiled using the command: `gcc privEsc.c -o privEsc`. The file compiles successfully, and it is saved along side the original file. Using `chmod +x privEsc`, the file is given execution permission. Finally, the command `./privEsc` executes it, and a root shell is spawned (Figure 14). A quick navigation to the /root directory and a `cat` command on the root.txt file shows the content, and this box is now fully compromised.

Vulnerability Detail and Mitigation

Vulnerability	Risk	Mitigation
HelpDeskZ File Upload Vulnerability	High	HelpDeskZ v1.0.2 suffers from an arbitrary file upload vulnerability. It would be beneficial to update to the latest version of the program and keep the software updated constantly. Keeping applications and systems patched reduces the likelihood of exploitation from known vulnerabilities. Additionally, fixing the code to improve error handling and disallowing certain file types would reduce the chance of a shell being uploaded.
Ubuntu 4.4.0-116 generic kernel privilege escalation vulnerability	High	The kernel version 4.4.0-116 carries a well known exploit which leads to a root shell being spawned. Keeping this up to date would avoid this exploit entirely.

Appendix 1: Full Nmap Results

```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-14 17:03 EDT
NSE: Loaded 148 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 17:03
Completed NSE at 17:03, 0.00s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 17:03
Completed NSE at 17:03, 0.00s elapsed
Initiating Ping Scan at 17:03
Scanning 10.10.10.121 [4 ports]
Completed Ping Scan at 17:03, 0.17s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 17:03
Completed Parallel DNS resolution of 1 host. at 17:03, 0.01s elapsed
Initiating SYN Stealth Scan at 17:03
Scanning 10.10.10.121 [1000 ports]
Discovered open port 80/tcp on 10.10.10.121
Discovered open port 22/tcp on 10.10.10.121
Increasing send delay for 10.10.10.121 from 0 to 5 due to 13 out of 41 dropped probes since last
increase.
Discovered open port 3000/tcp on 10.10.10.121
Completed SYN Stealth Scan at 17:03, 11.64s elapsed (1000 total ports)
Initiating Service scan at 17:03
Scanning 3 services on 10.10.10.121
Completed Service scan at 17:04, 11.91s elapsed (3 services on 1 host)
Initiating OS detection (try #1) against 10.10.10.121
Retrying OS detection (try #2) against 10.10.10.121
Retrying OS detection (try #3) against 10.10.10.121
Retrying OS detection (try #4) against 10.10.10.121
Retrying OS detection (try #5) against 10.10.10.121
Initiating Traceroute at 17:04
Completed Traceroute at 17:04, 2.76s elapsed
Initiating Parallel DNS resolution of 2 hosts. at 17:04
Completed Parallel DNS resolution of 2 hosts. at 17:04, 0.01s elapsed
NSE: Script scanning 10.10.10.121.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 17:04
Completed NSE at 17:04, 7.24s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 17:04
Completed NSE at 17:04, 0.00s elapsed
Nmap scan report for 10.10.10.121
```

```
Host is up, received echo-reply ttl 63 (0.29s latency).
Scanned at 2019-03-14 17:03:46 EDT for 56s
Not shown: 997 closed ports
Reason: 997 resets
PORT     STATE SERVICE REASON         VERSION
22/tcp   open  ssh     syn-ack ttl 63 OpenSSH 7.2p2 Ubuntu 4ubuntu2.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 e5:bb:4d:9c:de:af:6b:bf:ba:8c:22:7a:d8:d7:43:28 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCZY4jlvWqpdi8b
JPUnSkjWmz92KRwr2G6xCttorHM8Rq2eCEAe1ALqpgU4
4L3potYUZvaJuEIsBVUSPlsKv+ds8nS7Mva9e9ztlad/
fzBlyBpkiYxty+peoIzn4lUNSadPLtYH6khzN2PwEJYtM/
b6BLlAAY5mDsSF0Cz3wsPbnu87fNdd7WO0PKsqRtHpok
jkJ22uYJoDSAM06D7uBuegMK/sWTVtrsDakb1Tb6H8+D0y6ZQoE7XyHSqD0OABV3ON39G
zLBOnob4Gq8aegKBMa3hT/Xx9Iac6t5neiIABnG4UP03gm207oGIFHvlElGUR809Q9
qCJ0nZsup4bNqa/
|   256 d5:b0:10:50:74:86:a3:9f:c5:53:6f:3b:4a:24:61:19 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAy
NTYAAABBBHINVMyTivG0LmhaVZxiIESQuWxvN2jt87kY
iuPY2jyaPBD4DEt8e/1kN/4GMWj1b3FE7e8nxCL4PF/lR9XjEis=
|   256 e2:1b:88:d3:76:21:d4:1e:38:15:4a:81:11:b7:99:07 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHxDPln3rCQj04xFAKye
cXJaANrW3MBZJmbhtL4SuDYX
80/tcp   open  http    syn-ack ttl 63 Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: OPTIONS GET HEAD POST
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
3000/tcp open  http    syn-ack ttl 63 Node.js Express framework
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Site doesn't have a title (application/json; charset=utf-8).
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.70%E=4%D=3/14%OT=22%CT=1%CU=4330
6%PV=Y%DS=2%DC=T%G=Y%TM=5C8AC1E
OS:A%P=x86_64-pc-linux-gnu)SEQ(SP=104%GCD=1%ISR=10C%TI=Z%CI=I%II=I%
TS=8)SEQ
OS:(SP=104%GCD=1%ISR=10C%TI=Z%CI=I%TS=A)SEQ(SP=
104%GCD=1%ISR=10C%TI=Z%CI=I)
OS:OPS(O1=M54BST11NW7%O2=M54BST11NW7%O3=M54BN
NT11NW7%O4=M54BST11NW7%O5=M54B
OS:ST11NW7%O6=M54BST11)WIN(W1=7120%W2=7120%W
3=7120%W4=7120%W5=7120%W6=7120)
```

OS:ECN(R=Y%DF=Y%T=40%W=7210%O=M54BNNSNW7%CC=
Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%
OS:F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=
40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T
OS:5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=
OS:Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%
A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF
OS:=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G
%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40
OS:%CD=S)

Uptime guess: 0.000 days (since Thu Mar 14 17:04:27 2019)
Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=260 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 135/tcp)
HOP RTT ADDRESS
1 747.10 ms 10.10.16.1
2 747.17 ms 10.10.10.121

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 17:04
Completed NSE at 17:04, 0.00s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 17:04
Completed NSE at 17:04, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 56.62 seconds
Raw packets sent: 1382 (68.670KB) | Rcvd: 1211 (56.062KB)