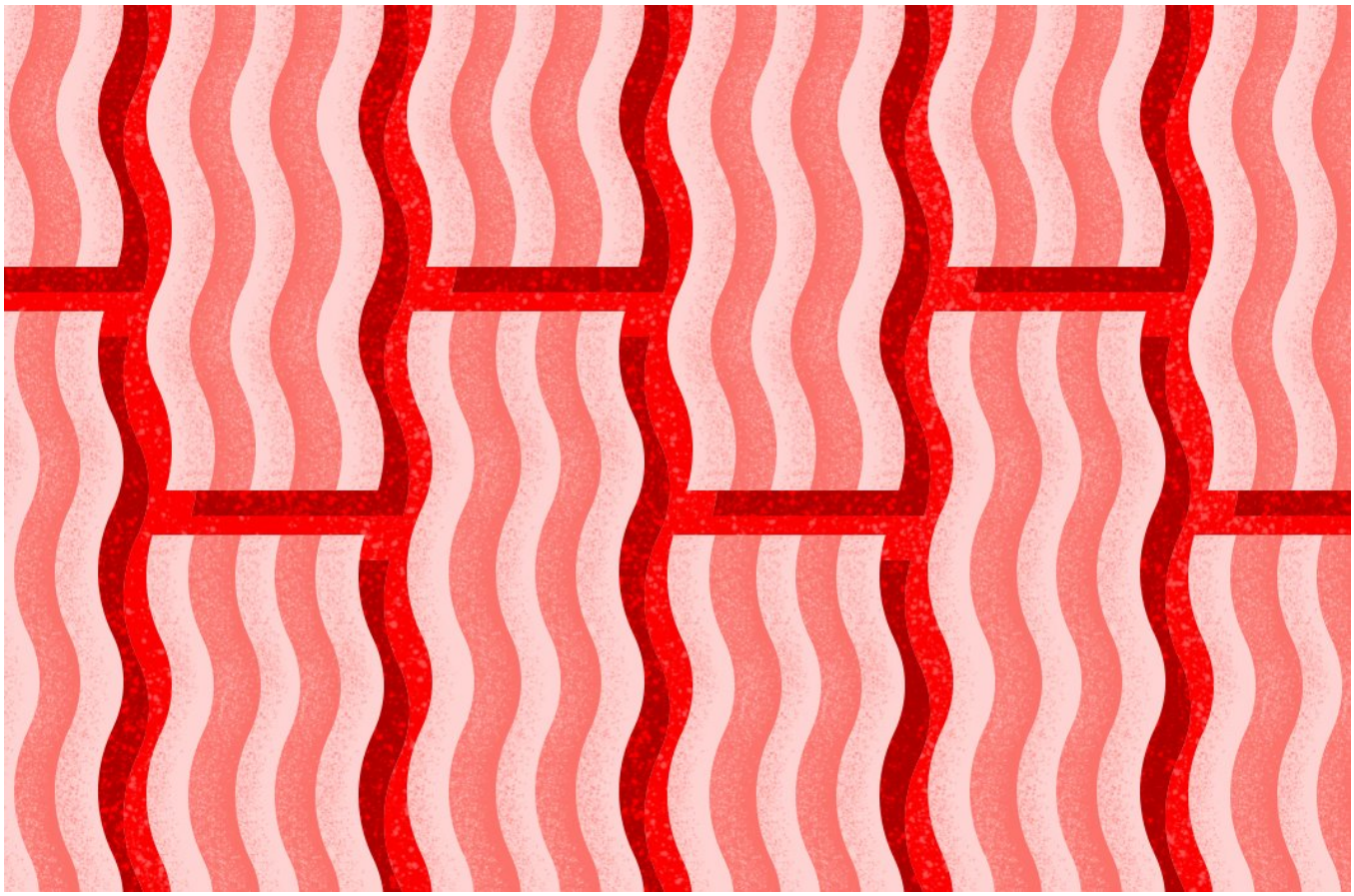
Getting the Bacon from the Beacon

crowdstrike.com/blog/getting-the-bacon-from-cobalt-strike-beacon/

Kareem Hamdan and Lucas Miller

September 29, 2020



In recent months, CrowdStrike® Services has observed a continued increase in the use of Cobalt Strike by eCrime and nation-state adversaries to conduct their operations following the initial access to victims' environments.

Cobalt Strike is a commercially available post-exploitation framework developed for adversary simulations and red team operations and features an easy-to-use interface. Although the vendor uses processes and technology measures in an effort to limit distribution of Cobalt Strike to security professionals, adversaries have also been observed using Cobalt Strike. In the CrowdStrike 2020 Threat Hunting Report, The Falcon OverWatch team reported Cobalt Strike as the #2 most common penetration testing tool observed in the first half of 2020.

A common feature used by adversaries is the Cobalt Strike framework client agent, known as Beacon. The Beacon client agent is executed in the memory space of a compromised system, typically leaving minimal on-disk footprints. This blog discusses CrowdStrike's research and testing of Cobalt Strike's Beacon in an isolated Active Directory domain to identify host-based indicators generated from the use of this tool.

This blog also enumerates and provides an explanation of host-based artifacts generated as a result of executing specific built-in Beacon commands. The artifacts can be used to create detection and prevention signatures in Windows environments, aiding in the positive identification of remnants of Beacon execution.

Beacon Behavior Summary

Adversaries often execute a variety of Beacon commands once they establish a foothold within an environment. Beacon commands can be used to spawn other Beacons on additional systems accessible to the initial Beacon, effectively furthering persistence in the target environment. Beacons can also be leveraged for remote access and execution.

- The execution of the commands highlighted in this blog will generate a variety of Windows security events depending on the context of the command: The Beacon commands `jump psexec` and `jump psexec_psh` will generate an EID 7045 (Service Installation) from `System.evtx`.
- The additional commands will generate an EID 400 event log (PowerShell Engine Startup) from Windows `PowerShell.evtx`.

The majority of PowerShell Engine Startup events generated by Cobalt Strike will have the `HostApplication` field begin with a command prefix. With the default configuration that command prefix is `powershell -nop -exec -bypass -EncodedCommand` . Although this prefix is configurable, CrowdStrike has observed adversaries leverage the default configuration in multiple incident response (IR) engagements.

Beacon Commands

As part of our research, CrowdStrike Services evaluated the following Beacon commands, which are encountered frequently in incident response engagements:

- powershell and powershell-import
- powerpick
- jump psexec
- jump psexec_psh
- jump winrm
- remote-exec wmi
- remote-exec powershell

In the following sections we'll review the purpose behind each of these commands, and the artifacts generated that may be useful for security analysts and threat hunters.

The `powershell` and `powershell-import` Commands

Both of these commands have a similar aim: to allow the user to execute PowerShell scripts on the target system. The `powershell` Beacon command executes commands written in PowerShell within the Cobalt Strike framework. When a red teamer or an adversary executes a command within a Beacon session, the operating system will generate an EID 400 event log (PowerShell Engine Startup) on the system that the command is executed on. The `powershell-import` Beacon command imports a PowerShell script into the Beacon session. In several WastedLocker ransomware attacks, CrowdStrike Services[1] observed evidence of the network discovery tool PowerView imported by adversaries shortly after establishing a Beacon on a compromised system. The file system artifacts that are generated will vary depending on whether the `powershell` command is executed before or after the `powershell-import` command.

Artifacts generated before `powershell-import`

Figure 1 shows an example of the EID 400 event log generated by the execution of the `powershell` command before a script has been imported with `powershell-import` . The base64 encoded command decodes to ls, the command that was executed via the `powershell` command.

Observations of `powershell` before `powershell-import` :

- The `HostApplication` field is set to `powershell -nop -exec -bypass -EncodedCommand <base64-encoded-command>`
- The Base64 encoded command decodes to the `<command>` executed

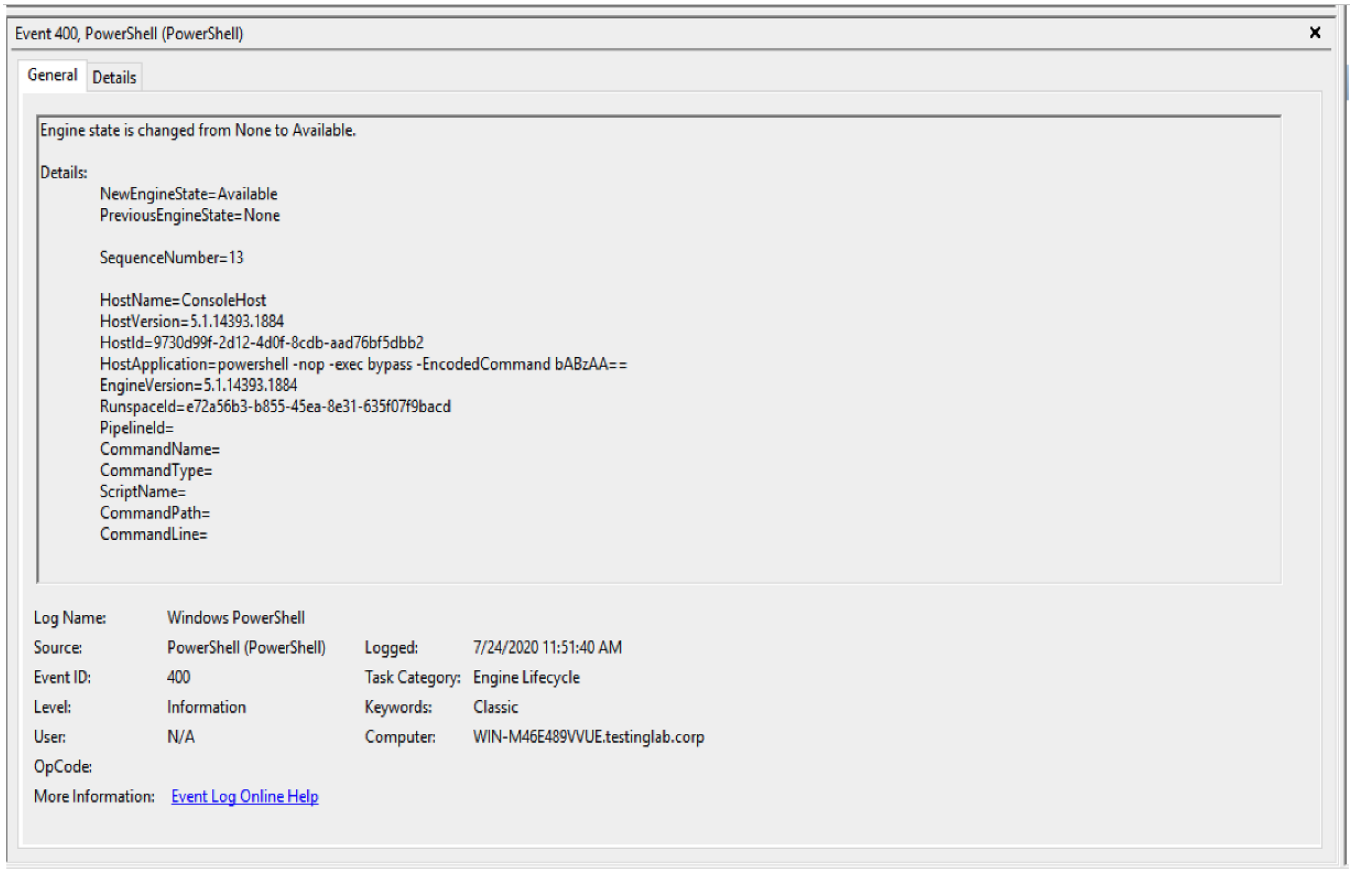


Figure 1. Artifact generated by the `powershell` command before `powershell-import` is executed (click image to enlarge)

An example of the observed artifact as shown in Figure 1:

```
HostApplication=powershell -nop -exec Bypass -EncodedCommand bABzAA==
Decoded Base64 Command: ls
```

Artifacts generated after `powershell-import`

Figure 2, shows an example of the EID 400 generated on the compromised system after execution of the powershell command after a script was imported with powershell-import. The base64 encoded command decodes to IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:22426/'); ls . The IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:22426/') component of the base64 encoded command is how Cobalt Strike manages imported PowerShell scripts within a Beacon session. The rest of the command, after the DownloadString component, is the PowerShell command run by the adversary.

Observations from `powershell` after `powershell-import` :

- The `HostApplication` field is set to `powershell -nop -exec -bypass -EncodedCommand <base64-encoded-command>`
- The base64 encoded command decodes to IEX `(New-Object Net.Webclient).DownloadString('http://127.0.0.1:<ephemeral-port-number>/'); <command>`

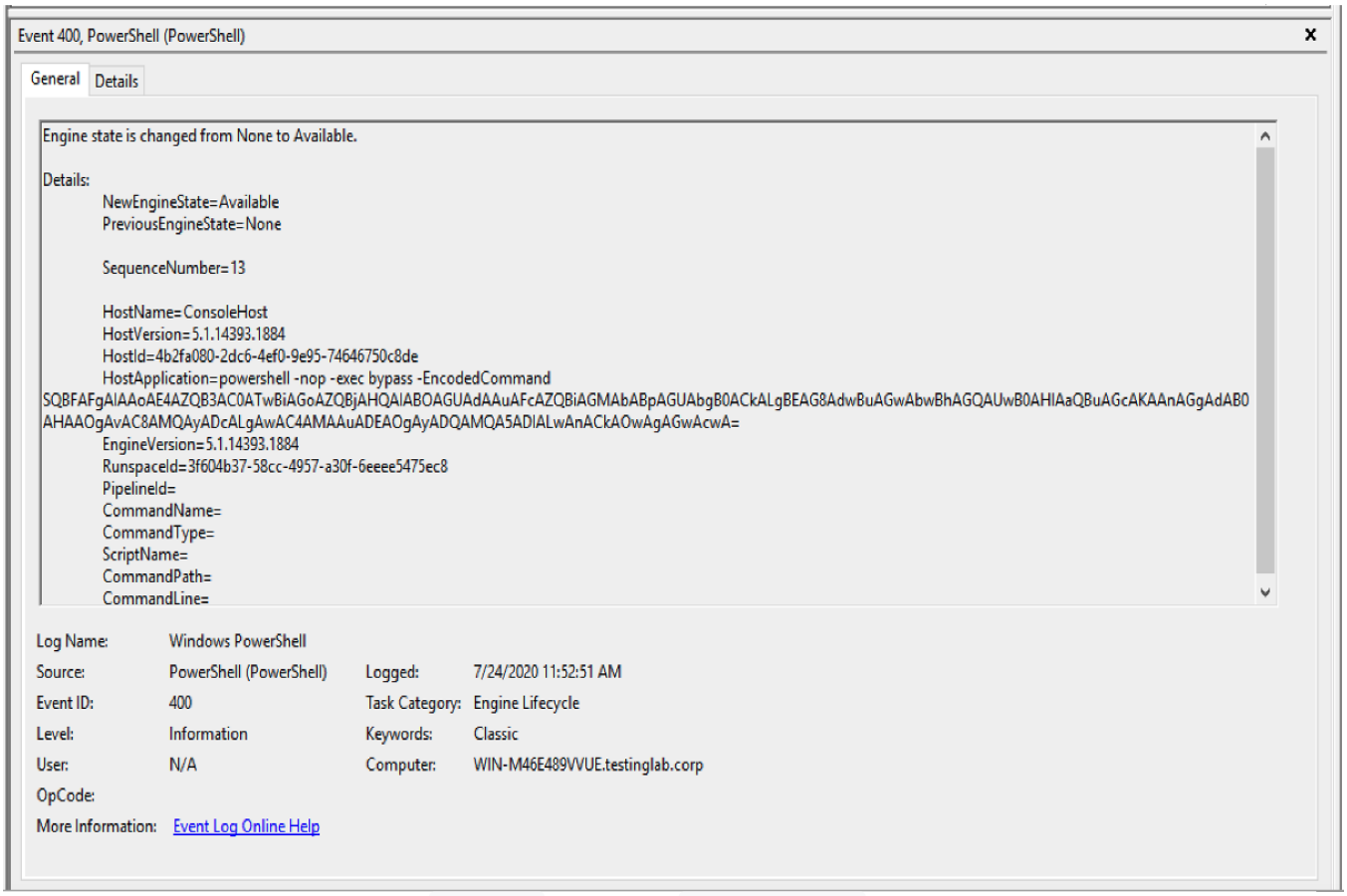


Figure 2. Artifact generated by the `powershell` command after `powershell-import` is executed (click image to enlarge)

An example of the observed artifact as shown in Figure 2:

```
HostApplication=powershell -nop -exec Bypass -EncodedCommand
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAGMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcA
Base64 Command: IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:24192/'); ls
```

The `powerpick` Command

The `powerpick` Beacon command executes unmanaged PowerShell on a compromised system. It provides a way to execute a PowerShell command without invoking `powershell.exe`. When a red teamer or adversary executes the `powerpick` command through a Beacon session, the filesystem will generate an EID 400 event log (PowerShell Engine Startup) on the compromised system.

CrowdStrike observed that the EID 400 event log generated by executing the `powerpick` command will contain a mismatch between the version number in the `HostVersion` and `EngineVersion` event log fields. The event generated will also have the path to the `rundll32.exe` executable in the `HostApplication` field, as it is the default program that a Beacon will use to create a new process.

Observations of `powerpick`:

- `HostName` field is set to `ConsoleHost`
- `HostApplication` field is set to the file path of `rundll32.exe`
- The `HostVersion` and `EngineVersion` fields are set to different values

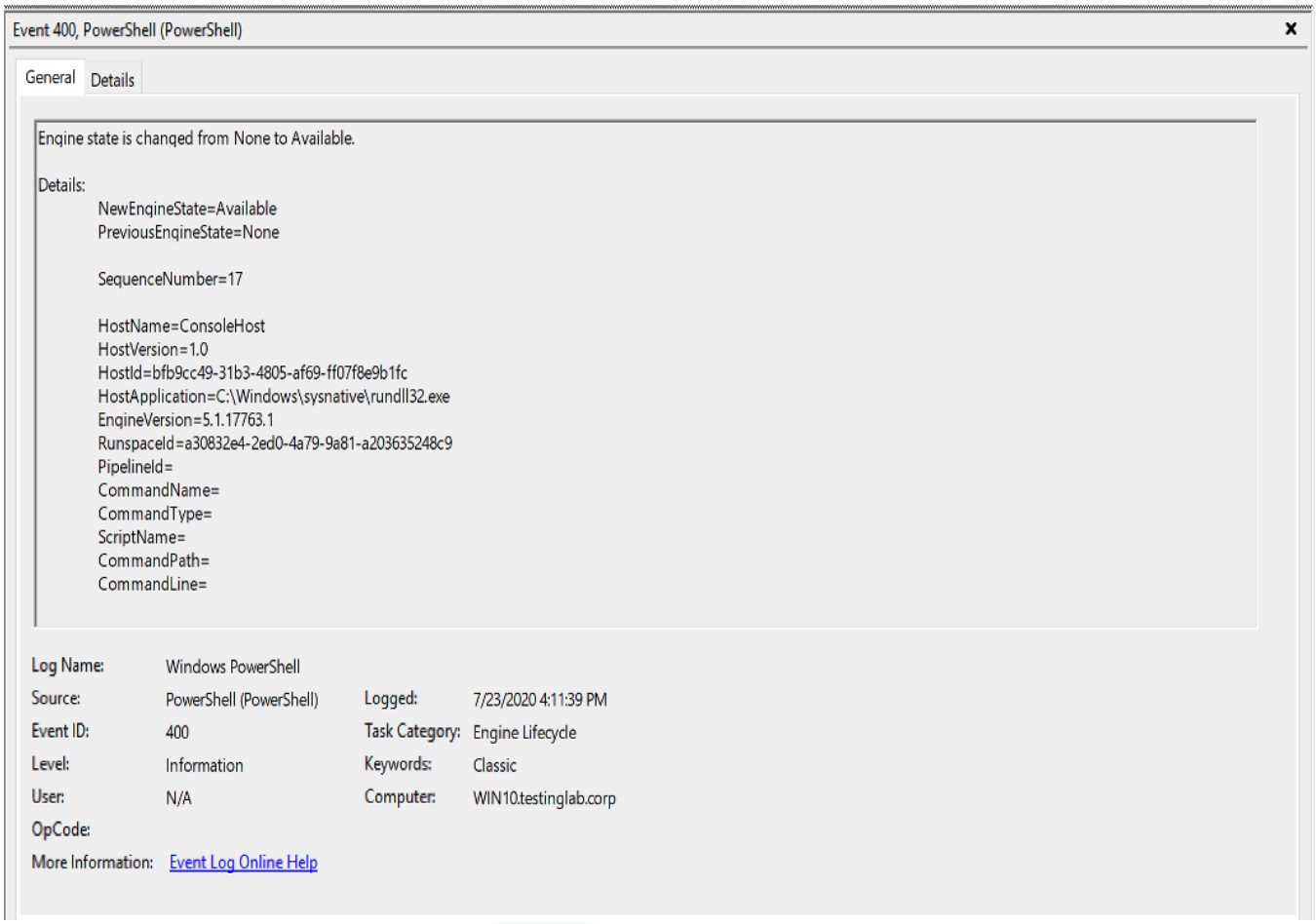


Figure 3. Artifact generated by the `powerpick` Beacon command when executed (click image to enlarge)

An example of the observed artifact as shown in Figure 3:

```
HostName=ConsoleHost
HostApplication=C:\windows\sysnative\rundll32.exe
```

```
HostVersion=1.0
```

```
EngineVersion=5.1.17763.1
```

The `jump psexec` Command

The `jump psexec` Beacon command establishes an additional Beacon on a remote system. When an adversary executes the `jump psexec` command through a Beacon session, the filesystem will generate an EID 7045 event log (Service Installation) on the remote system.

Observations of `jump psexec` :

- The Service Name field is set to `<7-alphanumeric-characters>`
- The Service File Name field is set to `\\127.0.0.1\ADMIN$\<7-alphanumeric-characters>.exe`

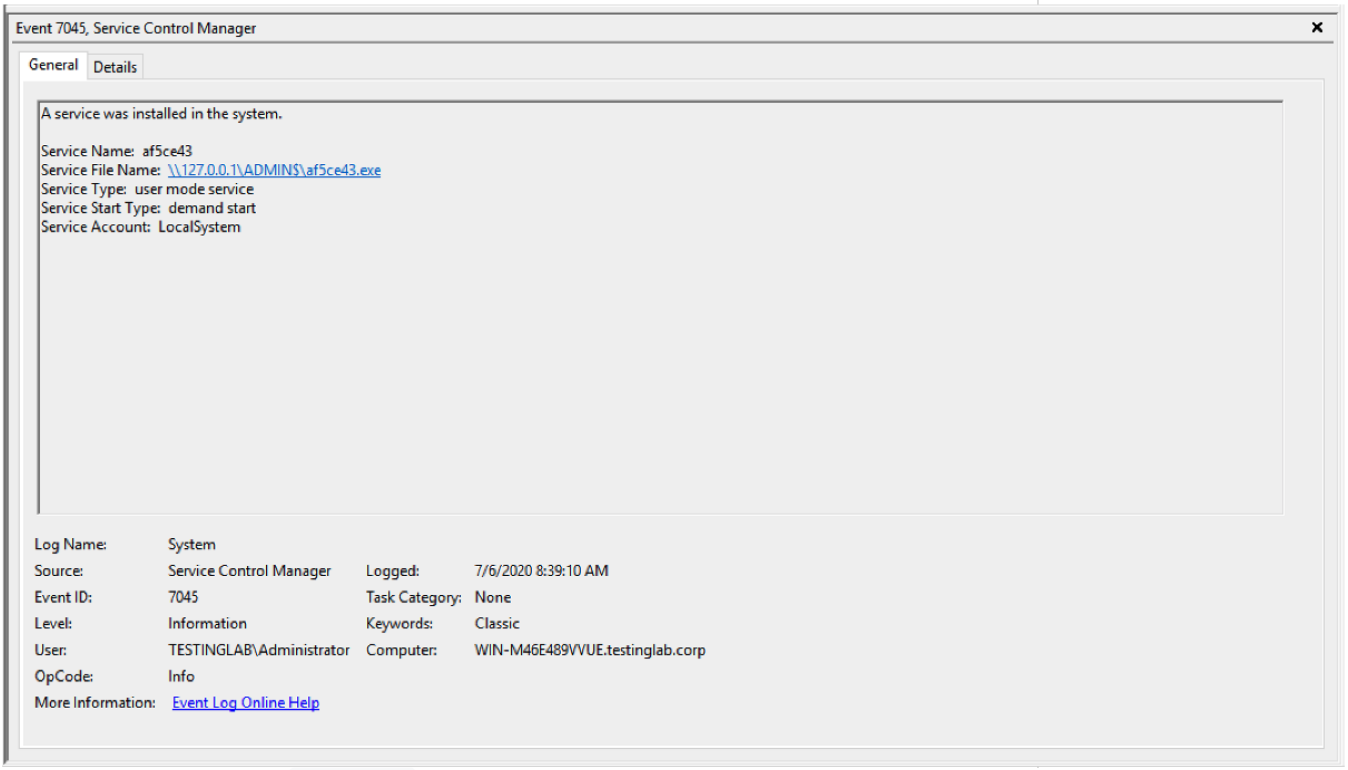


Figure 4. Artifact generated by the `jump psexec` Beacon command when executed on the remote system prior to version 4.1 of Cobalt Strike (click image to enlarge)

An example of the observed artifact as shown in Figure 4:

```
Service Name: af5ce43
Service File Name: \\127.0.0.1\ADMIN$\af5ce43.exe
```

By default, events generated by the `jump psexec` Beacon command using versions of Cobalt Strike prior to version 4.1 will have the `127.0.0.1` localhost string in the value of the "Service File Name," an example of this is `\\127.0.0.1\ADMIN$\7f5747a.exe`. Events generated with version 4.1+ of Cobalt Strike will contain the destination computer's IP address in the "Service File Name" by default and an example of this is `\\10.0.0.16\ADMIN$\9a845c4.exe`. In that example `10.0.0.16` is the IP address assigned to the target system.

Observations of `jump psexec` after version 4.1 of Cobalt Strike:

- The Service Name field is set to `<7-alphanumeric-characters>`
- The Service File Name field is set to `\\<System-IPAddress>\ADMIN$\<7-alphanumeric-characters>.exe`

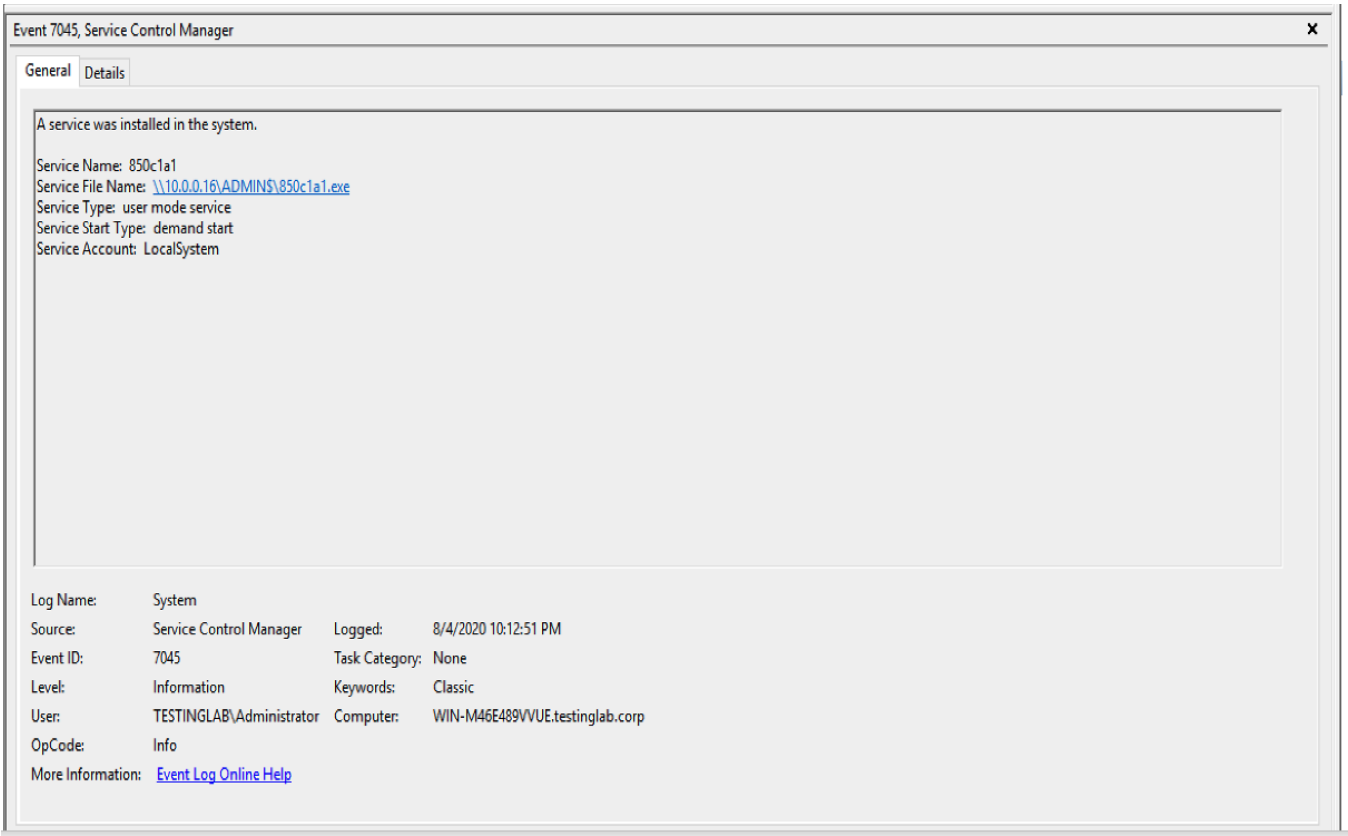


Figure 5. Artifact generated by the `jump psexec` Beacon command when executed on the remote system created by version 4.1+ of Cobalt Strike (click image to enlarge)

The `jump psexec_psh` Command

The `jump psexec_psh` command establishes an additional Beacon on a remote system via the Windows Service Control Manager. The `jump_psexec` command creates and starts a service that executes a base64 encoded PowerShell Beacon stager, which generates an EID 7045 event log (Service Installation) on the remote system.

The EID 7045 event log created by the `jump psexec_psh` command has a seven-character alphanumeric value for the "Service Name" field of the created event. The "Service File Name" field starts with the default Cobalt Strike prefix for PowerShell services `%COMSPEC% /b /c start /b /min powershell -nop -w hidden -encodedcommand` .

Observations of `jump psexec_psh` :

- The Service Name field is set to `<7-alphanumeric-characters>`
- The Service File Name field is set to `%COMSPEC% /b /c start /b /min powershell -nop -w hidden -encodedcommand <base64-encoded-command>`
- The base64 encoded command decodes to a PowerShell stager for a Cobalt Strike Beacon

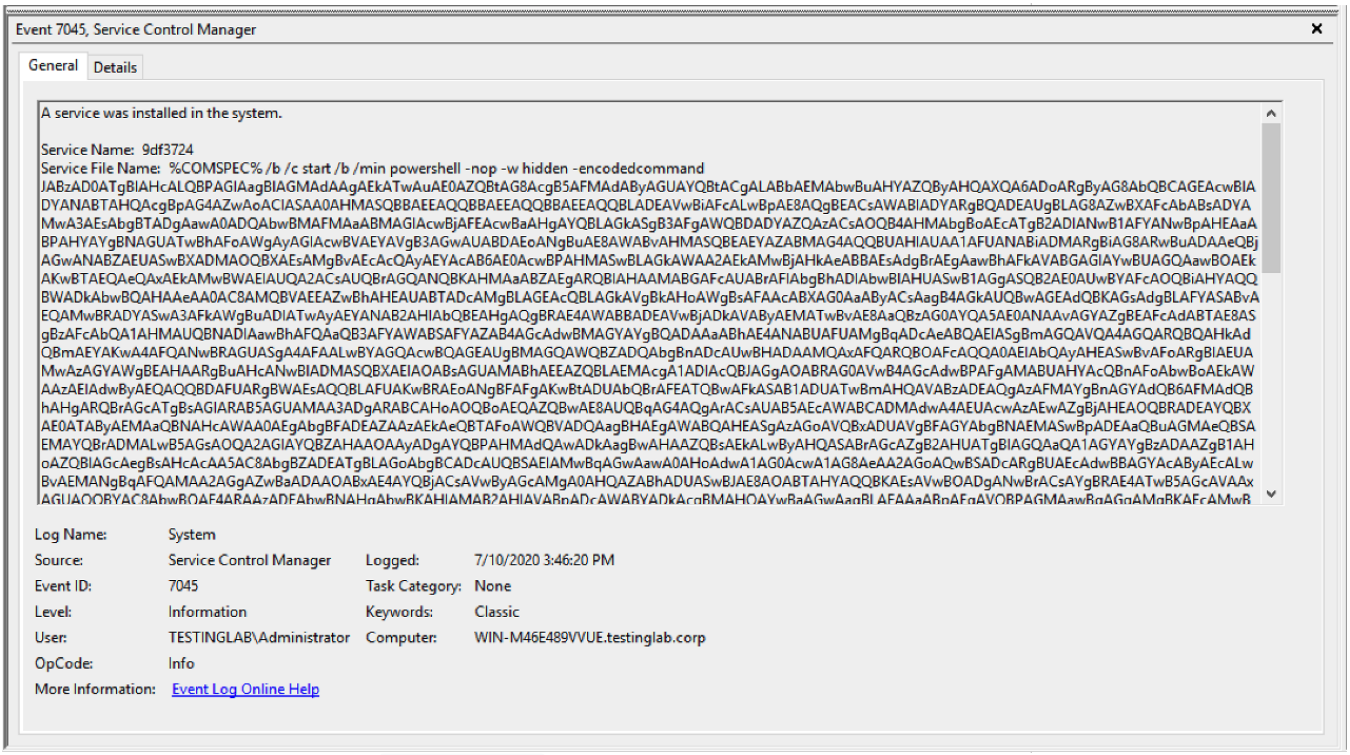


Figure 6. Artifact generated by the `jump psexec_psh` Beacon command when executed on the remote system (click image to enlarge)

An example of the observed artifact as shown in Figure 6:

```
Service Name: 9df3724
Service File Name: %COMSPEC% /b /c start /b /min powershell -nop -w hidden -encodedcommand JABzA<Redacted>
```

The `jump winrm` Command

The `jump winrm` Beacon command establishes a Beacon on a remote system utilizing the Windows Remote Management (WinRM) interface (native on all Windows devices). When the `jump winrm` Beacon command is executed by an adversary through a Beacon session, the filesystem will generate an EID 400 event log (PowerShell Engine Startup) on the compromised system. The event created will contain the Cobalt Strike PowerShell command prefix in the `HostApplication` field. The generated event is not affected by the usage of any of the PowerShell-related Beacon commands.

Observations of `jump winrm` on the compromised system:

- The `HostApplication` field is set to `powershell -nop -exec -bypass -EncodedCommand <base64-encoded-command>`
- The base64 encoded command decodes to `IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:<ephemeral-port-number>/')`

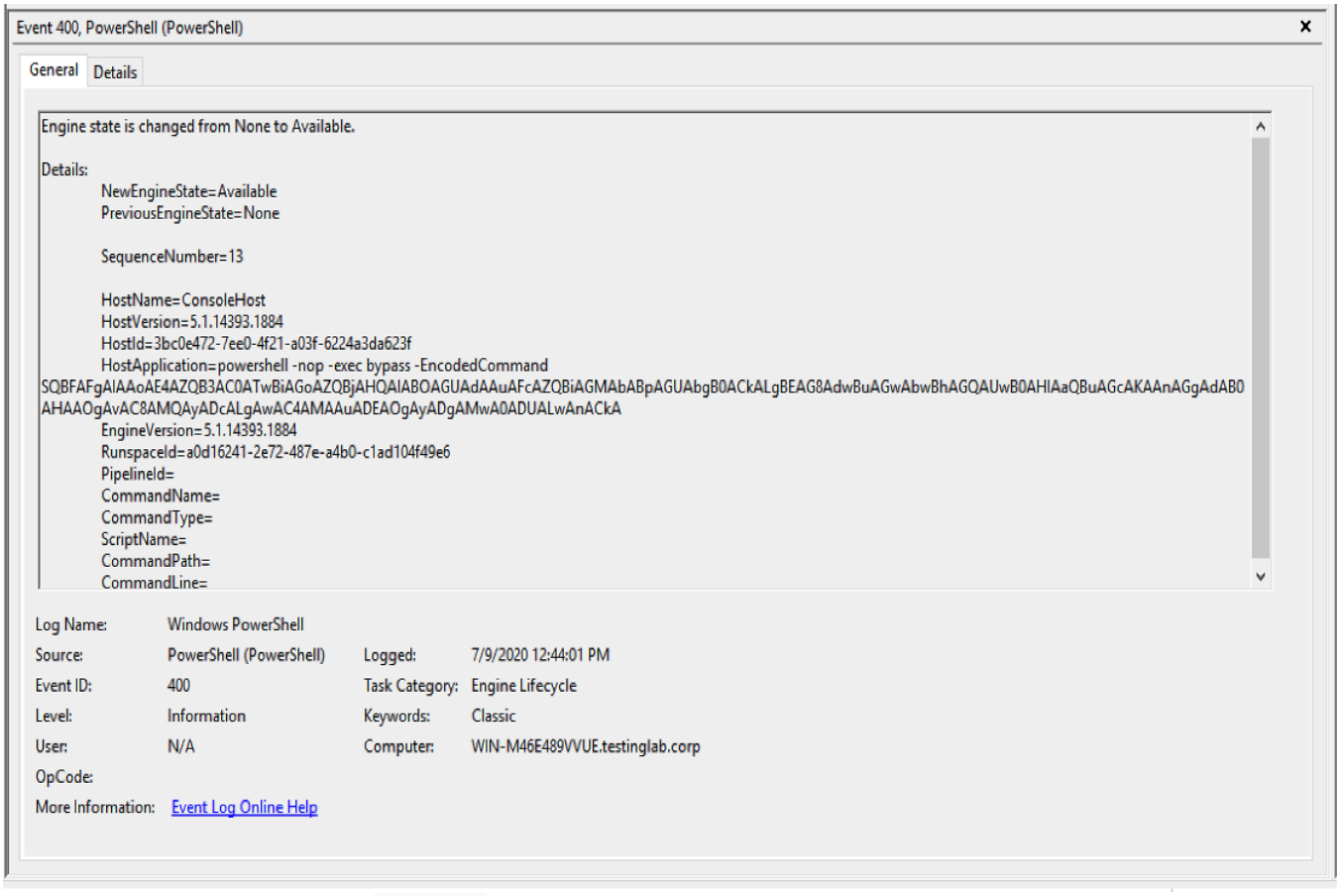


Figure 7. Artifact generated by the `jump winrm` Beacon command when executed, on the compromised system (click image to enlarge)

An example of the observed artifact as shown in Figure 7:

```
HostApplication=powershell -nop -exec bypass -EncodedCommand
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAGMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcA
Base64 Command: IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:28345/')
```

If a WinRM listener is not present on the remote system when the `jump winrm` command is executed, Cobalt Strike will create an EID 400 event log on the remote system, as shown in Figure 7.

Observations of an event created by `jump winrm` on the remote system:

The `HostApplication` field is set to `<path-to-PS-executable> -Version <PS-Version> -s -NoLogo -NoProfile`

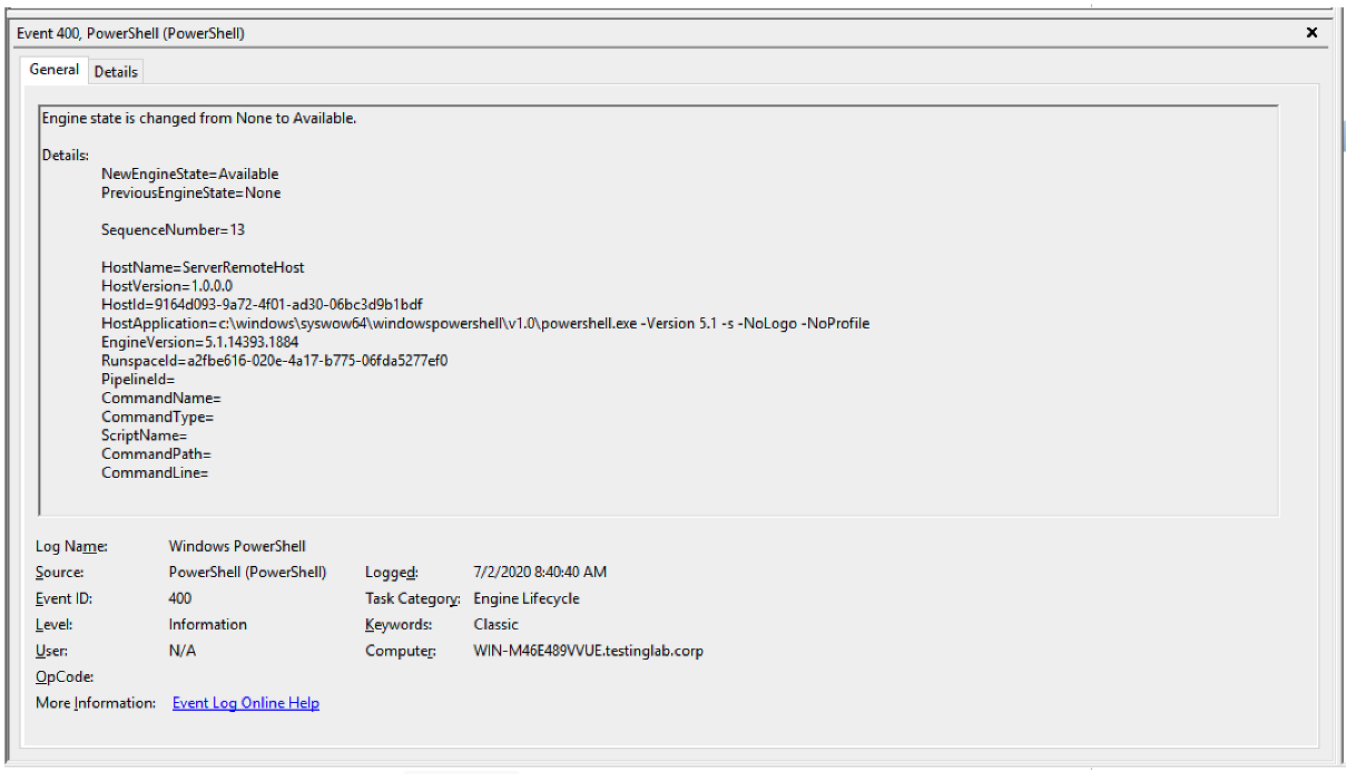


Figure 8. Artifact generated by the `jump winrm` Beacon command when executed on the remote system (click image to enlarge)

An example of the observed artifact as shown in Figure 8:

```
HostApplication=c:\windows\syswow64\windowspowershell\v1.0\powershell.exe -Version 5.1 -s -NoLogo -NoProfile
```

The `remote-exec wmi` Command

The `remote-exec wmi` Beacon command executes a command on a remote system via WMI. When the `remote-exec wmi` command is executed, the filesystem will generate an EID 400 event log (PowerShell Engine Startup) on the compromised system with the standard Cobalt Strike PowerShell command prefix in the `HostApplication` field.

Observations of `remote-exec wmi`:

- The `HostApplication` field is set to `powershell -nop -exec Bypass -EncodedCommand <base64-encoded-command>`
- The base64 encoded command decodes to `Invoke-WMIMethod win32_process -name create -argumentlist '<command>' -ComputerName <target>`

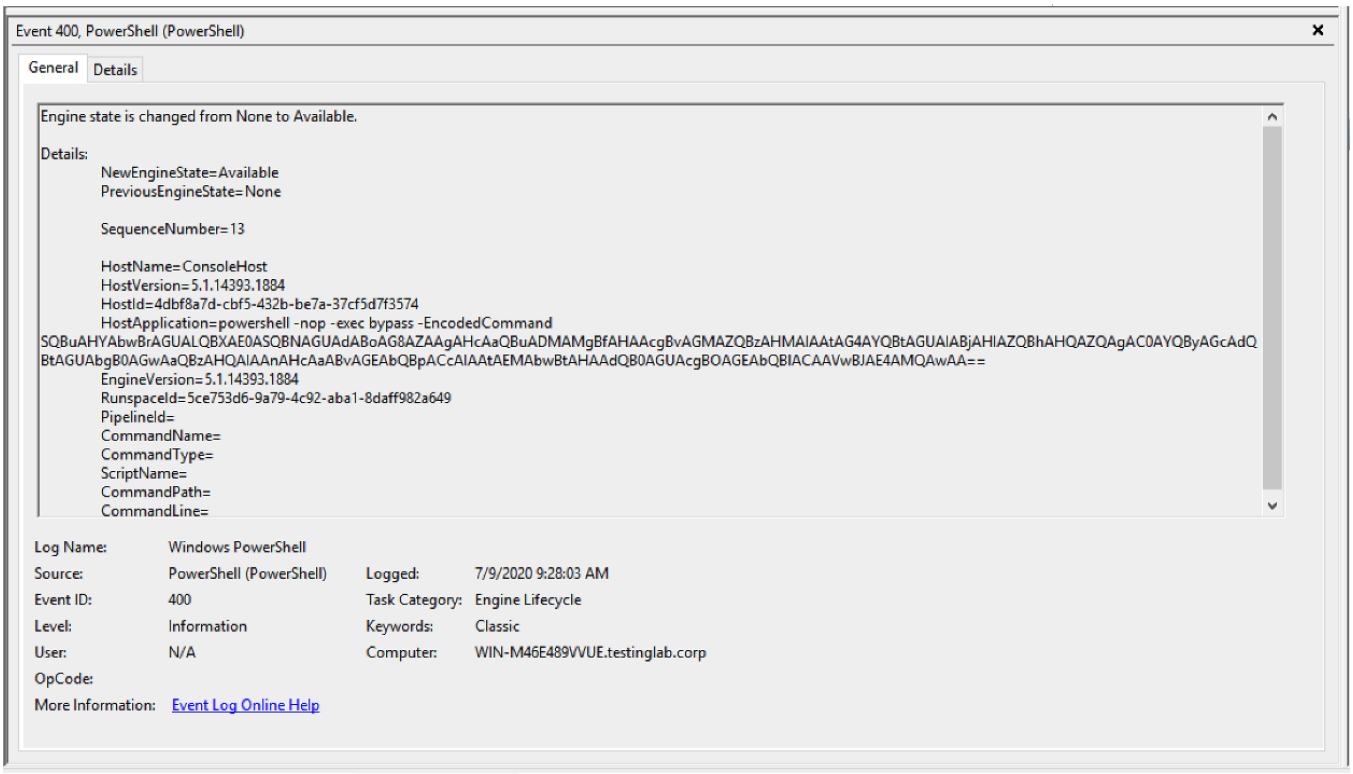


Figure 9. Artifact generated by the `remote-exec wmi` Beacon command when executed on the compromised system (click image to enlarge)

An example of the observed artifact as shown in Figure 9:

```
HostApplication=powershell -nop -exec Bypass -EncodedCommand
SQBuAHYAbwBrAGUALQBXAE0ASQBNAGUAdABoAG8AZAAgAHcAaQBuADMAMgBfAHAAcgBvAGMAZQBzAHMAIAAtAG4AYQBtAGUAIABjAHIAZQBhAHQAZQAgAC0A
Base64 Command: Invoke-WMIMethod win32_process -name create -argumentlist 'whoami' -ComputerName WIN10
```

The `remote-exec powershell` Command

The `remote-exec powershell` Beacon command executes a command on a remote system via PowerShell remoting from a compromised system. When the `remote-exec powershell` command is executed, the filesystem will generate an EID 400 event log (PowerShell Engine Startup) on the compromised system. The event created will contain the standard Cobalt Strike PowerShell command prefix in the `HostApplication` field.

Observations of `remote-exec powershell`:

- The `HostApplication` field is set to `powershell -nop -exec Bypass -EncodedCommand <base64-encoded-command>`
- The Base64 encoded command decodes to `Invoke-Command -ComputerName <target> -ScriptBlock { <command> }`

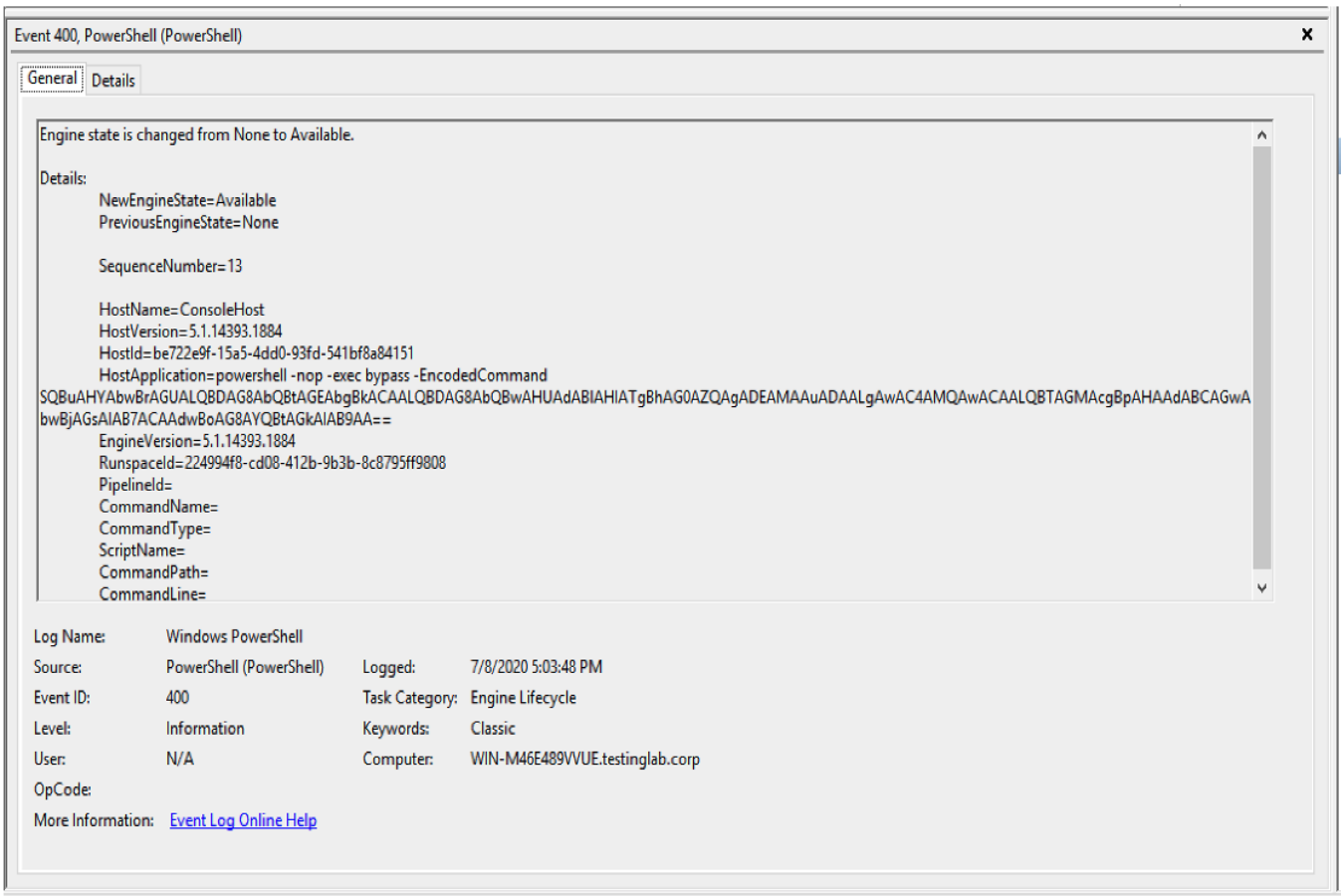


Figure 10. Artifact generated by the `remote-exec powershell` Beacon command when executed on the compromised system (click image to enlarge)

An example of the observed artifact as shown in Figure 10:

```
HostApplication=powershell -nop -exec Bypass -EncodedCommand
SQBuAHYAbwBrAGUALQBDAG8AbQBtAGEAbgBkACAALQBDAG8AbQBwAHUAdABlAHIATgBhAG0AZQAgADEAMAAuADAALgAwAC4AMQAwACAALQBTAGMAcgBpAHAA
Base64 Command: Invoke-Command -ComputerName 10.0.0.10 -ScriptBlock { whoami }
```

Conclusions

Although Cobalt Strike provides the operator a degree of freedom to configure some of the previously mentioned commands, those features are not always leveraged by adversaries. Due to the high prevalence of Cobalt Strike in contemporary intrusions, CrowdStrike recommends collecting EID 400 (PowerShell Engine Startup) and EID 7045 event logs (Service Installation) for monitoring and alerting in a centralized security information and event management (SIEM) platform.

CrowdStrike also recommends upgrading to the most recent version of PowerShell and disabling previous versions, as PowerShell is backward compatible. While these additional security measures do not provide full visibility into Cobalt Strike activity, they can aid in its detection.

[1] CrowdStrike has previously reported on adversaries that use Cobalt Strike, such as COBALT SPIDER.

Additional Resources

- *Learn more about the CrowdStrike Services team and how it can help your organization improve your cybersecurity readiness.*
- *Read about the powerful CrowdStrike Falcon® platform by visiting the webpage.*
- *Test CrowdStrike next-gen AV for yourself. Start your free trial of Falcon Prevent™ today.*