ESET discovered an undocumented backdoor used by the infamous Stealth Falcon group

By ESET Research

Archived: 2026-04-05 18:08:12 UTC

Stealth Falcon is a threat group, active since 2012, that targets political activists and journalists in the Middle East. It has been tracked by the Citizen Lab, a non-profit organization focusing on security and human rights, which published an analysis of a particular cyberattack in 2016. In January of 2019, Reuters published an investigative report into Project Raven, an initiative allegedly employing former NSA operatives and aiming at the same types of targets as Stealth Falcon.

Based on these two reports referring to the same targets and attacks, Amnesty International's Senior Technologist, Claudio Guarnieri, has concluded that Stealth Falcon and Project Raven actually are the same group.

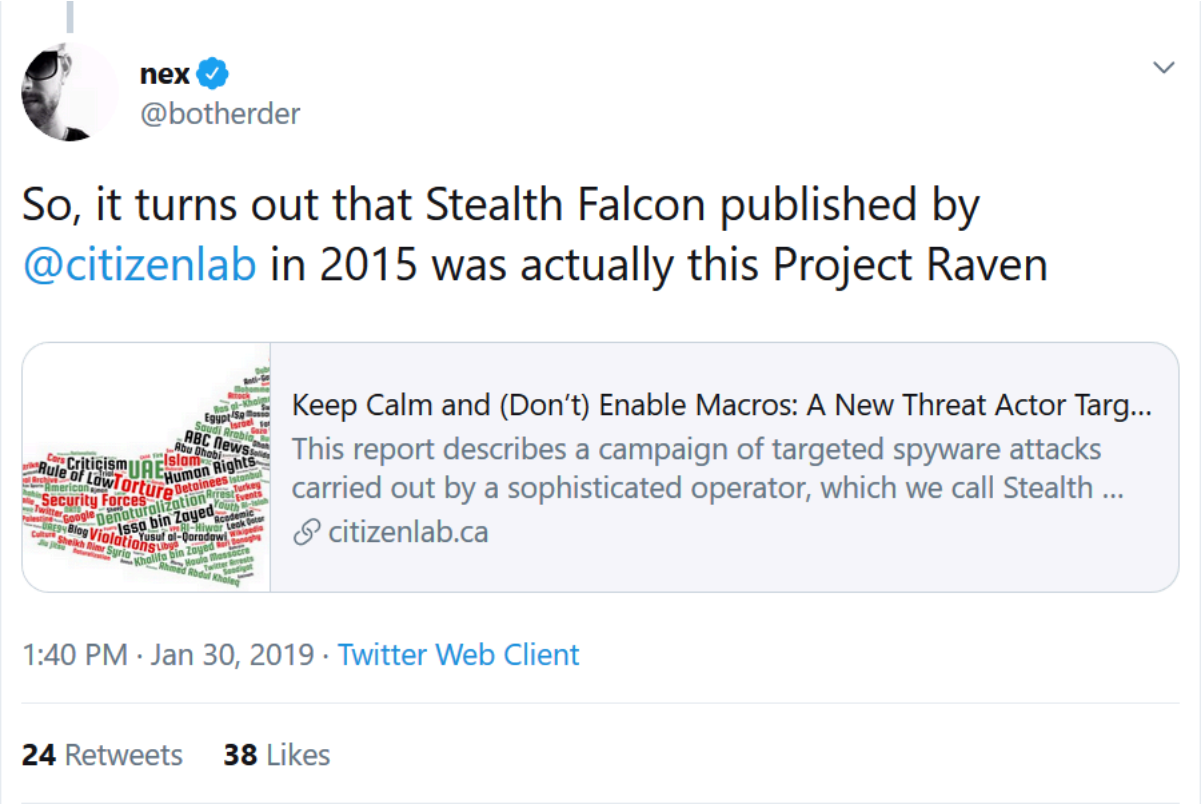

Figure 1. Claudio Guarnieri has connected Stealth Falcon with Project Raven

Some technical information about Stealth Falcon has already been made public – notably, in the already mentioned analysis by the Citizen Lab.

The key component in the attack documented in the Citizen Lab report was a PowerShell-based backdoor, delivered via a weaponized document that was included in a malicious email.

Now, we have found a previously unreported binary backdoor we have named Win32/StealthFalcon. In this article, we disclose similarities between this binary backdoor and the PowerShell script with backdoor capabilities attributed to the Stealth Falcon group. We consider the similarities to be strong evidence that Win32/StealthFalcon was created by this group.

The Win32/StealthFalcon backdoor, which appears to have been created in 2015, allows the attacker to control the compromised computer remotely. We have seen a small number of targets in UAE, Saudi Arabia, Thailand, and the Netherlands; in the latter case, the target was a diplomatic mission of a Middle Eastern country. How the backdoor was distributed and executed on the target systems is beyond the scope of this investigation; our analysis focuses on its capabilities and its C&C communication.

C&C communication

In its communication with the C&C server, Win32/StealthFalcon uses the standard Windows component Background Intelligent Transfer Service (BITS), a rather unusual technique. BITS was designed to transfer large amounts of data without consuming a lot of network bandwidth, which it achieves by sending the data with throttled throughput so as not to affect the bandwidth needs of other applications. It is commonly used by updaters, messengers, and other applications designed to operate in the background. This means that BITS tasks are more likely to be permitted by host-based firewalls.

Compared with traditional communication via API functions, the BITS mechanism is exposed through a COM interface and thus harder for a security product to detect. Moreover, this design is reliable and stealthy. The transfer resumes automatically after being interrupted for reasons like a network outage, the user logging out, or a system reboot. Moreover, because BITS adjusts the rate at which files are transferred based on the bandwidth available, the user has no reason for suspicion.

Win32/StealthFalcon can switch the communication between two C&C servers whose addresses are stored in a registry key, along with other configuration values, and can be updated by one of the backdoor commands. In case the backdoor fails to reach out to its C&C servers, the backdoor removes itself from the compromised system after a preconfigured number of failed attempts.

Capabilities

Win32/StealthFalcon is a DLL file that, after execution, schedules itself as a task running on each user login. It only supports basic commands but displays a systematic approach to data collection, data exfiltration, employing further malicious tools, and updating its configuration.

Command name	Functionality
K	Uninstall itself
CFG	Update configuration data
RC	Execute the specified application
DL	Write downloaded data to file

Command name	Functionality
CF	Prepare a file for exfiltration
CFW	Exfiltrate and delete files
CFWD	Not implemented/no operation

Table 1. Backdoor commands

For example, the backdoor's key capability, downloading and executing files, is achieved via regular checks for libraries named "win*.dll" or "std*.dll" in the directory the malware is executed from, and loading these libraries.

Furthermore, Win32/StealthFalcon collects files and prepares them for exfiltration by storing an encrypted copy with a hardcoded prefix in a temporary folder. It then regularly checks for such files and exfiltrates them automatically. After the files have been successfully exfiltrated, the malware safe-deletes all log files and collected files - before deleting the files, it rewrites them with random data - to prevent forensic analysis and recovery of the deleted data.

The configuration values are stored in the HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Shell Extensions registry key. All values are prefixed by the malware's filename (without extension).

Value name suffix	Content
-FontDisposition	Randomly generated, 4-byte victim ID
-MRUData	RC4-encrypted C&C domain
-MRUList	RC4-encrypted C&C domain
-IconPosition	Flag determining which of the C&C domains should be used
-IconDisposition	Number of seconds to sleep after each iteration of contacting the C&C server
-PopupPosition	Counter of failed attempts to reach the C&C servers

Table 2. Configuration data stored in registry

Possible trick to evade detection

Of interest is a function that is executed before any malicious payload is started, and which seems redundant. It references 300+ imports, but does not use them at all. Instead, it always returns and continues with the payload afterward, without condition checks that would suggest it is an anti-emulation trick.

```
.text:10003E1C ; int __cdecl av_bypass_fake_reference_imports()
.text:10003E1C av_bypass_fake_reference_imports proc near
.text:10003E1C                                         ; CODE XREF: MAIN+15↓p
.text:10003E1C                 mov     eax, flag_bypass_av
.text:10003E21                 cmp     eax, 1
.text:10003E24                 jnz     end
.text:10003E2A                 xor     ecx, ecx
.text:10003E2C                 cmp     func_addr, ecx
.text:10003E32                 jz      end
.text:10003E38                 mov     eax, ds:GetProcessWindowStation
.text:10003E3D                 mov     func_addr, eax
.text:10003E42                 cmp     eax, ecx
.text:10003E44                 jz      end
.text:10003E4A                 mov     eax, ds:GetUserObjectInformationW
.text:10003E4F                 mov     func_addr, eax
.text:10003E54                 cmp     eax, ecx
.text:10003E56                 jz      end
.text:10003E5C                 mov     eax, ds:GetLastActivePopup
.text:10003E61                 mov     func_addr, eax
.text:10003E66                 cmp     eax, ecx
.text:10003E68                 jz      end
.text:10003E6E                 mov     eax, ds:GetActiveWindow
.text:10003E73                 mov     func_addr, eax
.text:10003E78                 cmp     eax, ecx
.text:10003E7A                 jz      end
.text:10003E80                 mov     eax, ds:MessageBoxW
.text:10003E85                 mov     func_addr, eax
.text:10003E8A                 cmp     eax, ecx
.text:10003E8C                 jz      end
.text:10003E92                 mov     eax, ds:ImmReleaseContext
.text:10003E97                 mov     func_addr, eax
.text:10003E9C                 cmp     eax, ecx
.text:10003E9E                 jz      end
.text:10003EA4                 mov     eax, ds:ImmSetCompositionWindow
.text:10003EA9                 mov     func_addr, eax
.text:10003EAE                 cmp     eax, ecx
.text:10003EB0                 jz      end
.text:10003EB6                 mov     eax, ds:ImmGetContext
.text:10003EBB                 mov     func_addr, eax
.text:10003EC0                 cmp     eax, ecx
.text:10003EC2                 jz      end
.text:10003EC8                 mov     eax, ds:ImmGetCompositionStringW
.text:10003ECD                 mov     func_addr, eax
.text:10003ED2                 cmp     eax, ecx
.text:10003ED4                 jz      end
.text:10003EDA                 mov     eax, ds:ImmSetCompositionFontA
.text:10003EDF                 mov     func_addr, eax
.text:10003EE4                 cmp     eax, ecx
.text:10003EE6                 jz      end
.text:10003EEC                 mov     eax, ds:PlaySoundA
.text:10003EF1                 mov     func_addr, eax
.text:10003EF6                 cmp     eax, ecx
.text:10003EF8                 jz      end
.text:10003EFE                 mov     eax, ds:GetTickCount
.text:10003F03                 mov     func_addr, eax
.text:10003F08                 cmp     eax, ecx
.text:10003F0A                 jz      end
.text:10003F10                 mov     eax, ds:MulDiv
```

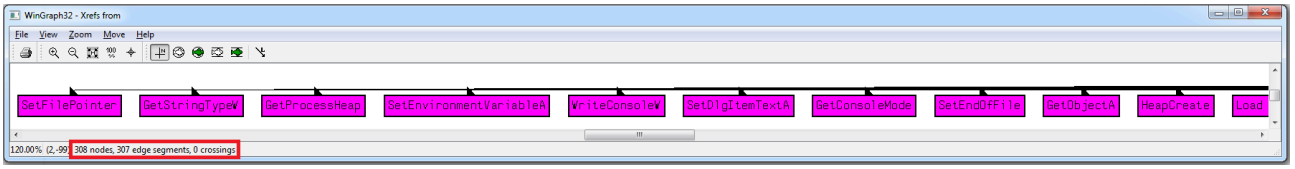

Figure 2. A function referencing hundreds of unused imports, possibly added to avoid detection of the malware

We don't know the precise intention of this function, but we suspect it is either some attempt to evade detection, or some leftover from a larger framework used by the malware authors.

Links to Stealth Falcon

Both Win32/StealthFalcon and the PowerShell-based backdoor described in the Citizen Lab analysis share the same C&C server: the address windowsearchcache[.]com was used as a "Stage Two C2 Server Domain" in the backdoor analyzed by the Citizen Lab, and also in one of the versions of Win32/StealthFalcon.

Both backdoors display significant similarities in code – although they are written in different languages, the underlying logic is preserved. Both use hardcoded identifiers (most probably campaign ID/target ID). In both cases, all network communication from the compromised host is prefixed with these identifiers and encrypted with RC4 using a hardcoded key.

For their C&C server communication, they both use HTTPS but set specific flags for the connection to ignore the server certificate.

Conclusion

We discovered and analyzed a backdoor with an uncommon technique for C&C communication – using Windows BITS – and some advanced techniques to hinder detection and analysis, and to ensure persistence and complicate forensic analysis. Similarities in the code and infrastructure with a previously known malware by Stealth Falcon drive us to the conclusion that the Win32/StealthFalcon backdoor is also the work of this threat group.

Indicators of Compromise (IoCs)

ESET detection name

Win32/StealthFalcon

SHA-1

31B54AEBDAF5FBC73A66AC41CCB35943CC9B7F72
50973A3FC57D70C7911F7A952356188B9939E56B
244EB62B9AC30934098CA4204447440D6FC4E259
5C8F83CC4FF57E7C67925DF4D9DAABE5D0CC07E2

RC4 keys

258A4A9D139823F55D7B9DA1825D101107FBF88634A870DE9800580DAD556BA3
2519DB0FFEC604D6C9A655CF56B98EDCE10405DE36810BC3DCF125CDE30BA5A2
3EDB6EA77CD0987668B360365D5F39FDCF6B366D0DEAC9ECE5ADC6FFD20227F6
8DFFDE77A39F3AF46D0CE0B84A189DB25A2A0FEFD71A0CD0054D8E0D60AB08DE

Note: Malware derives a second RC4 key by XORing each byte of the hardcoded key with 0x3D.

Host-based indicators

Malware file names

ImageIndexer.dll
WindowsBackup.dll
WindowsSearchCache.dll
JavaUserUpdater.dll

Log file name patterns

%TEMP%\dsc*
%TEMP%\sld*
%TEMP%\plx*

Registry keys/values

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Shell Extensions
X-MRUList
X-MRUData
X-FontDisposition
X-IconDisposition
X-IconPosition
X-PopupPosition
X is the malware's filename (without extension).

Network indicators

BITS job names

WindowsImages-
WindowsBackup-
WindowsSearchCache-
ElectricWeb

C&C servers

footballtimes[.]info
vegetableportfolio[.]com
windowsearchcache[.]com
electricalweb[.]org
upnpdiscover[.]org

MITRE ATT&CK techniques

Tactic	ID	Name	Description
Execution	T1059	Command-Line Interface	Malware uses cmd.exe to execute some commands.
	T1106	Execution through API	Malware uses CreateProcessW API for execution.
	T1085	Rundll32	Malware uses rundll32.exe to load the backdoor DLL.
	T1053	Scheduled Task	Malware schedules rundll32.exe to be executed on each login, and subsequently to load the backdoor DLL.
Persistence	T1053	Scheduled Task	Malware establishes persistence by scheduling a task that loads the backdoor on each user login.
Defense Evasion	T1197	BITS Jobs	Malware uses BITS file transfer mechanism for network communication, in an attempt to avoid detection.
	T1140	Deobfuscate/Decode Files or Information	Strings are encrypted with a custom XOR cipher.
	#rowspan#	#rowspan#	Configuration data and log files are encrypted with RC4, using a hardcoded key.
	T1107	File Deletion	Malware overwrites files with random data, and deletes them after exfiltration.
	T1036	Masquerading	Malware attempts to disguise itself by using seemingly-legitimate file

Tactic	ID	Name	Description
			names.
	T1112	Modify Registry	Malware stores its configuration in a registry key.
	T1027	Obfuscated Files or Information	Strings are encrypted with a custom XOR cipher.
	#rowspan#	#rowspan#	Configuration data and log files are encrypted with RC4, using a hardcoded key.
Discovery	T1063	Security Software Discovery	Malware terminates itself if McAfee Agent binary (cmdagent.exe) is detected.
Collection	T1074	Data Staged	Malware stores collected data in a temporary folder in files named with a hardcoded prefix.
	T1005	Data from Local System	Malware has a command to collect/steal a file from the compromised system.
Command and Control	T1008	Fallback Channels	Malware is able to communicate with two C&C servers; it also supports switching to a different C&C server using a backdoor command.
	T1105	Remote File Copy	Malware uses BITS Jobs for C&C communication.
	T1005	Standard Cryptographic Protocol	Malware encrypts C&C communication using RC4 with a hardcoded key.
Exfiltration	T1020	Automated Exfiltration	Malware automatically exfiltrates files in a temporary folder in files named with a hardcoded prefix.
T1022	Data Encrypted	Malware encrypts the collected data using RC4 with a hardcoded key, prior to exfiltration.	

Tactic	ID	Name	Description
	T1041	Exfiltration Over Command and Control Channel	Malware exfiltrates data over the C&C channel.

Source: https://www.welivesecurity.com/2019/09/09/backdoor-stealth-falcon-group/