Dissecting Agent Tesla: Unveiling Threat Vectors and Defense Mechanisms | Idan Malihi

Archived: 2026-04-05 22:30:13 UTC

AgentTesla Execution Diagram

The **WSF script** is written in **JavaScript** and embedded within the batch file. It is executed by the **cscript** command. The script is obfuscated, making it difficult to read and understand the first stage's code. This is a common technique used by malware authors to evade detection and analysis. After the de-obfuscation process, the script executes a **PowerShell** payload, which is also obfuscated.

After running the script, two processes were launched: the **cscript** process, which executes the WSF script, and the **PowerShell** process, which runs the payload.

The PowerShell payload sets the system's security protocol to TLS 1.2 and loads the Microsoft.VisualBasic assembly. The script then repeatedly pings Google until an internet connection is detected. Upon connection, it creates a new WebClient object to download and execute a script from https://didaktik-labor.de/mx1.jpg.

After replacing the '**A**' character with '**00**' in the payload, we can understand that the first payload is coded in **binary**.

It is a **PowerShell** function that is responsible for decompressing the other two payloads that exist in the second stage using **Gzip**.

The first payload decompresses the second and third payloads using **Gzip**. The second payload is stored in the variable **y74gh00rffd** and contains an obfuscated payload in which the script replaces the '**EV**' characters with '**0x**'. This suggests that the second payload is in **hexadecimal** format.

After decompressing the second payload, the script produces a **DLL file**.

The **eSQy** variable contains an obfuscated payload, in which the script replaces the '**EV**' characters with '**0x**'. This suggests that the third payload is in **hexadecimal** format, also.

After decompressing the third payload, the script produces an **EXE** file. The file is written in **.NET** and runs on a 32-bit architecture.

After decompressing the script and dropping the DLL and EXE files, it extracts the **Black** function from the **toooyou** module in the DLL file and then uses the **InstallUtil** tool to execute the EXE file.

The **Black** function executes code using the **calli** instruction, repeatedly calling a method until a certain condition is met. This is part of a larger obfuscated malware process that decompresses, drops, and executes malicious payloads (EXE and DLL files) on the target system. In conjunction with the rest of the script, this function's

purpose is to evade detection and execute malicious binaries and operations. Furthermore, the **Black** function gets two variables as values; the first value that the function gets is the **InstallUtil.exe**, and the second value is the **eSQy** variable, which executes the **EXE** file.

At the beginning of the malware debugging, the malware configures the **ServicePointManager** to allow **HTTPS** connections using **SSL 3.0**, **TLS 1.0**, **TLS 1.1**, and **TLS 1.2** protocols.

After the malware sets up the HTTPS connections, it prepares the execution process by ensuring that only one copy of the current process is running and terminating any other copies of the same process. It does this by comparing the process IDs of all running copies of the process and terminating those with different IDs from the current process.

The malware checks for the **%appdata%\Roaming** path in the endpoint in the **StartupDirectoryPath** variable for dropping the gnxLZ.exe file.

Then, the malware enumerates the username and the hostname of the current endpoint.

The malware gathers detailed system information, including the current time, computer name, operating system, username, RAM, CPU, and external IP address. Threat actors can utilize this information to profile victims, plan further attacks, or sell on dark net forums.

After stealing all the intended information, the malware organizes the data from the victim's computer and then sends it to the attacker's C2 server.

Then, the malware configures and opens an SMTP connection with the attacker's SMTP mail server.

The attacker's SMTP server information:

The malware connects to the SMTP server using port 587 to 94.237.43.240.

The SMTP connection packets in Wireshark:

The SMTP server's IP address is 94.237.43.240. The first packet sent from the SMTP server indicates that it is hosted on the stablehost.com website.

The malware exfiltrates the data to the SMTP server over TLSv1.2 (HTTPS).

An example of a stolen data from a victim:

MITRE ATT&CK

Yara Rule

Yara Detection

Snort Rule

Snort Detection
