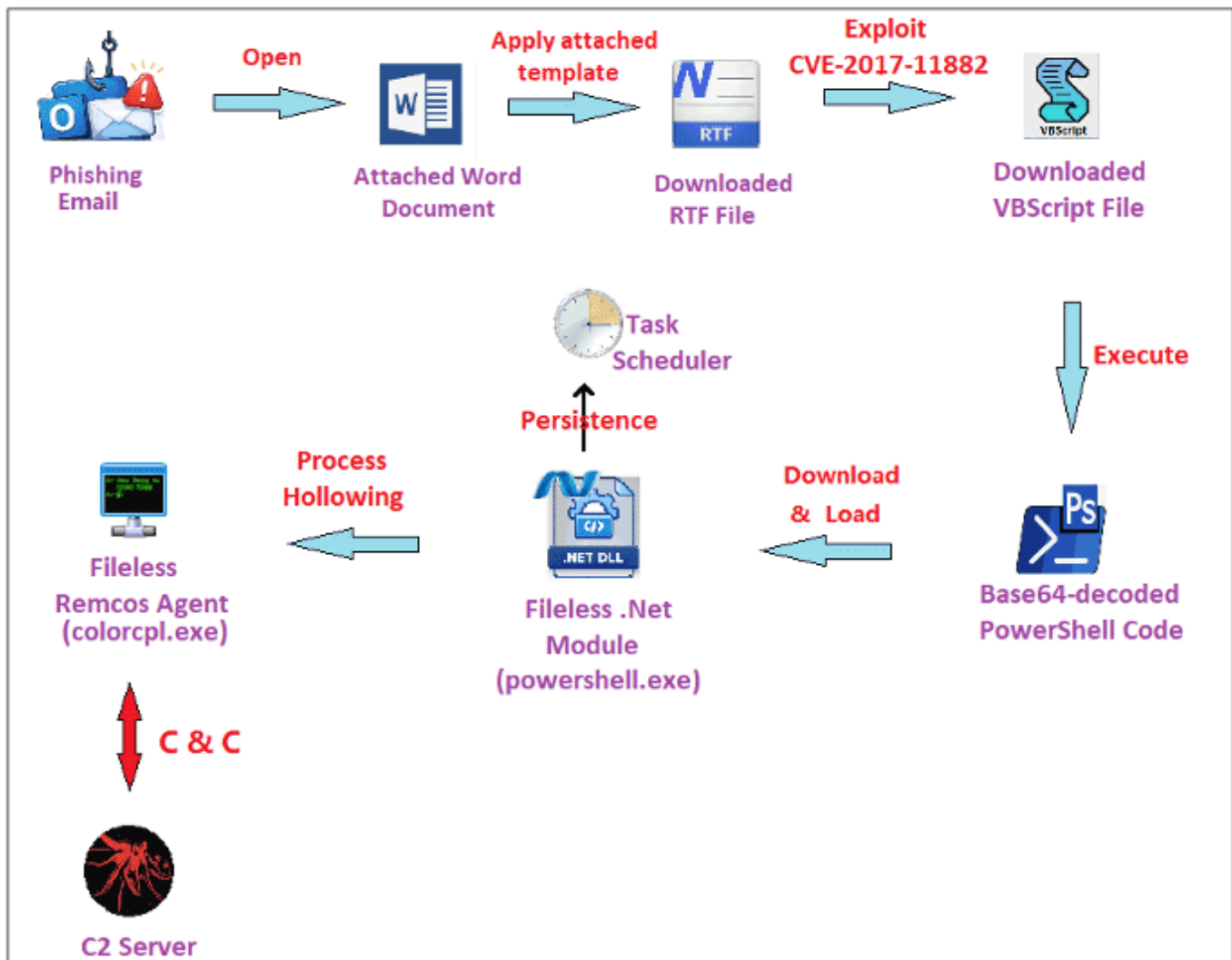


New Remcos Campaign Distributed Through Fake Shipping Document

 fortinet.com/blog/threat-research/new-remcos-campaign-distributed-through-fake-shipping-document

Xiaopeng Zhang Xiaopeng Zhang

January 14, 2026



- Article Contents
- [CVE-2017-11882](#)
- [1. Persistence2. Remcos Payload Loader](#)

By [Xiaopeng Zhang](#) | January 14, 2026

Affected Platforms: Microsoft Windows

Impacted Users: Windows Users

Impact: Full remote control of the victim's computer

Severity Level: High

Background

FortiGuard Labs discovered a new phishing campaign in the wild. The campaign delivers a new variant of Remcos, a commercial lightweight remote access tool (RAT) with a wide range of capabilities, including system resource management, remote surveillance, network management, and Remcos agent management.

I conducted an in-depth investigation into this malicious campaign. This analysis covers how the phishing email initializes the attack, how the attached Word document downloads an RTF file, the vulnerability the attack leverages within the RTF file, the VBScript and PowerShell code, how a fileless .NET module is loaded and executed in a PowerShell process, and how the fileless Remcos agent is downloaded and loaded using process hollowing.

The analysis also details this Remcos variant's internal configuration block, packet structures, and capabilities across six categories, which are illustrated with examples.

Infection Chain

Figure 1 shows the infection chain for the new Remcos variant, illustrating how it is distributed and installed on a victim's machine.

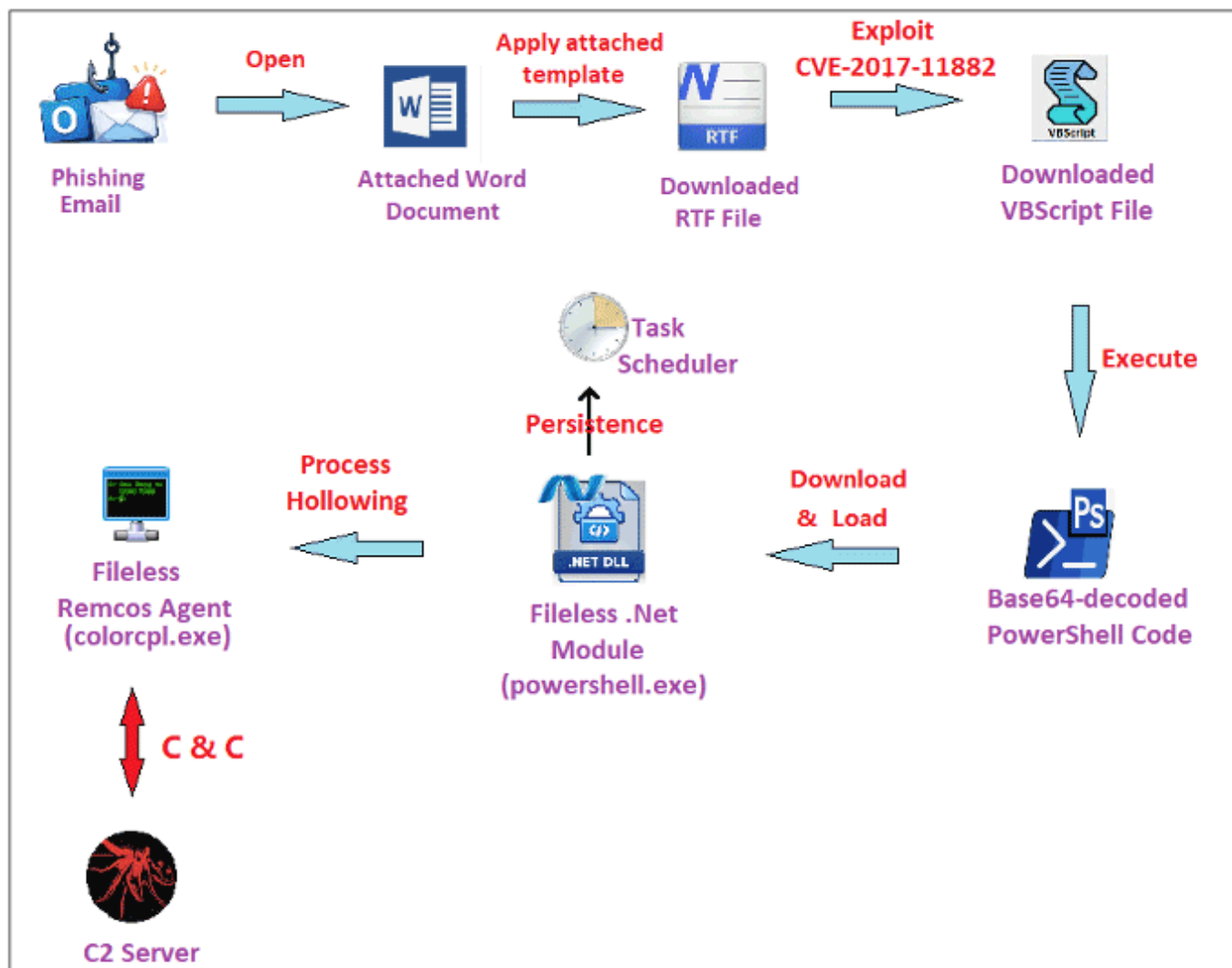


Figure 1: The Remcos variant infection chain.

Captured Phishing Email

The captured phishing email, disguised as a message from a shipping company in Vietnam, lures the recipient to open an attached Word file to view an updated shipping document, as shown in Figure 2.

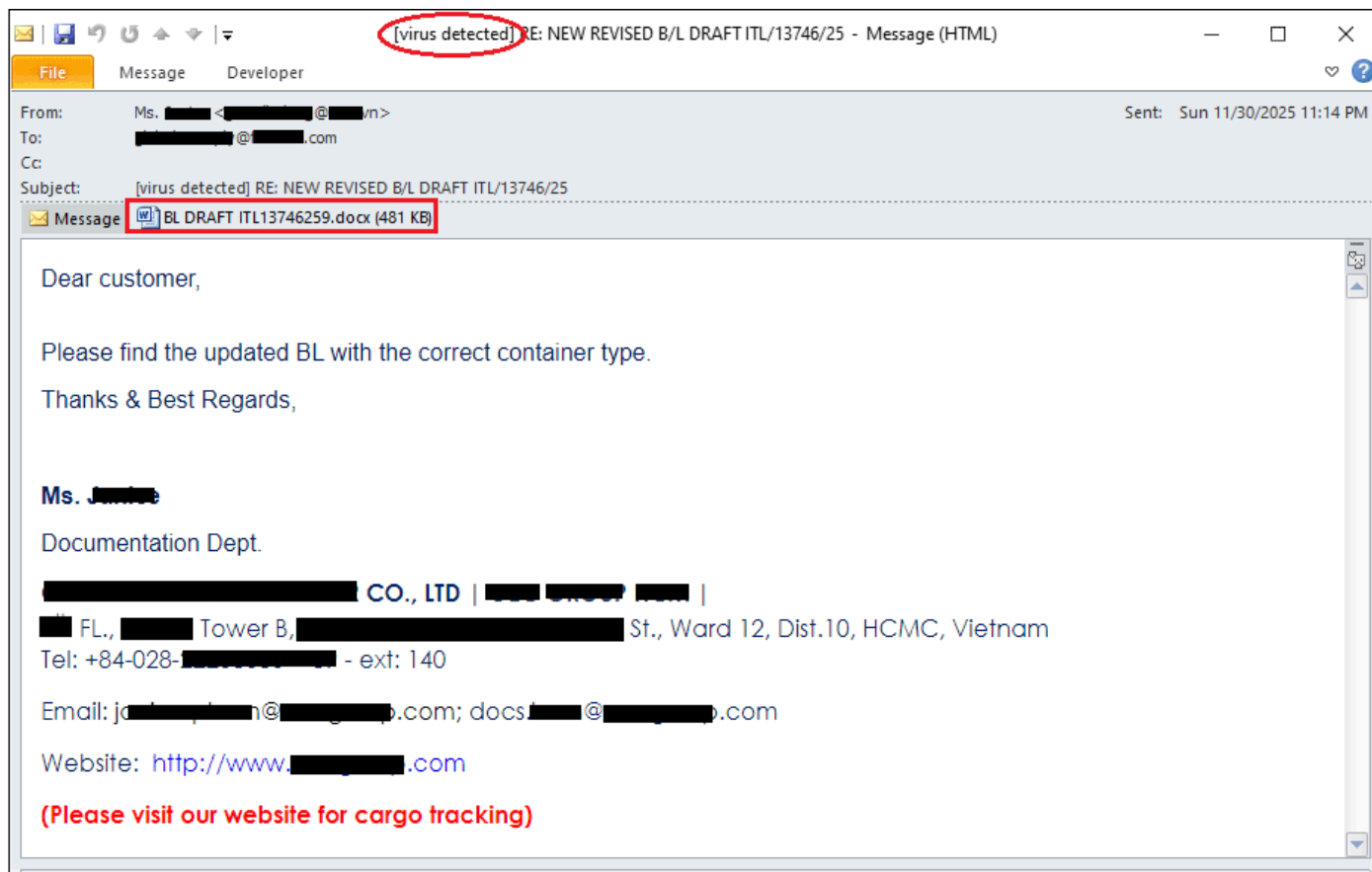


Figure 2: The phishing email contents.

Note that the email in this image has been flagged as “virus detected” by the FortiMail service, so the email would be blocked and quarantined by FortiMail and not delivered to the recipient, thwarting the attack.

Dissecting the Word Document

When examining the internal structure of the attached Word document, we find that it contains an online attachedTemplate option in the settings.xml.rels file, as shown in Figure 3.

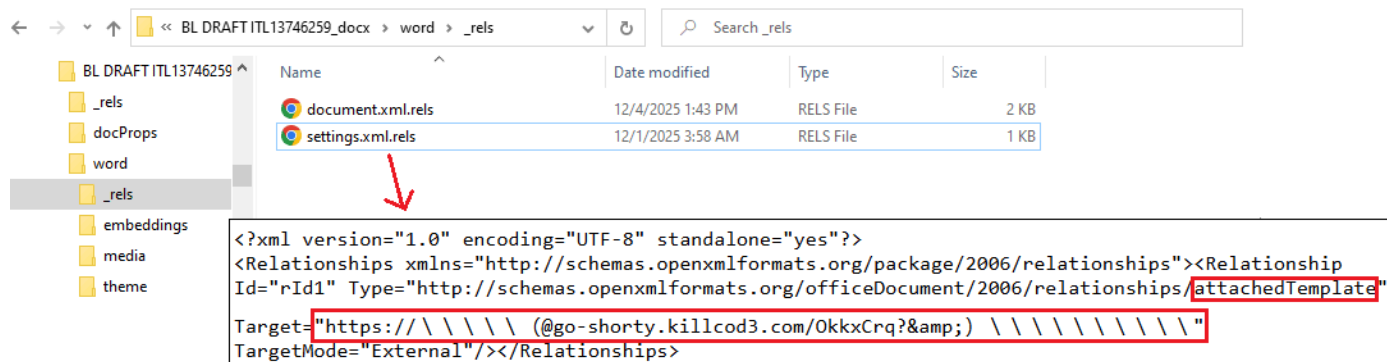


Figure 3: The attachedTemplate definition in settings.xml.rels file.

and the full path of a local file

%AppData%\90g90dfg090asdfxcv0sdf09sdf90200002f0sf0df09f0s9f0sdf0sf00d.vbs is marked.

Finally, the shellcode calls the ShellExecuteW() API to execute the downloaded VBScript file.

Analyzing the VBScript File and PowerShell Code

The VBScript file is lightly obfuscated and contains a Base64-encoded PowerShell code.

Below is a snippet of the VBScript code.

```
Dim millilitres, panzer, undulant, prostrates, sawhorse
Set millilitres = GetObject("winmgmts:root\cimv2")
Set panzer = millilitres.Get("Win32_ProcessStartup").SpawnInstance_
panzer.ShowWindow = 0
Set undulant = millilitres.Get("Win32_Process")
prostrates = 0
sawhorse = undulant.Create(microliths(crummier), banie, panzer, prostrates)
wscript.quit(1)
```

The PowerShell code is assembled into the following command line, which is stored in the variable “crummier.”

powershell -NoProfile -WindowStyle Hidden -Command {Base64-decoded PowerShell code}

Then, the VBScript code creates a WMI object to execute the command line.

In Figure 7, the Base64-encoded string is shown on the left, while the Base64-decoded PowerShell code appears on the right.

The screenshot shows a Base64-decoding tool with two panes. The left pane contains a long Base64-encoded string. The right pane contains the decoded PowerShell code. Red boxes highlight the following sections of the code:

- Download .Net Module in PNG file:** This section shows the code downloading a .NET module from a remote URL and saving it as a PNG file.
- Extract, Decode and Load the .Net Module:** This section shows the code extracting the .NET module from the PNG file, decoding it, and loading it into memory.
- Call .Net Module's Method - VAI():** This section shows the code calling a method named VAI() on the loaded .NET module.

Figure 7: Base64-decoded PowerShell code.

The PowerShell code downloads a .NET module (.NET DLL) that is transformed and embedded in a normal image file (an optimized_MSI.png, which actually has JPEG data). The Base64-encoded .NET module is appended to the PNG file and enclosed between the markers BaseStart- and - BaseEnd.

After decoding, the code calls the [Reflection.Assembly]::Load() method to load the .NET module into the PowerShell's process and invokes its VAI() method with a large number of parameters, as shown in Figure 7. Figure 8 illustrates the binary structure of the image file.

optimized_MSI.png																	ANSI ASCII	
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
00000000	FF	D8	FF	E2	0C	58	49	43	43	5F	50	52	4F	46	49	4C	yøÿâ	XICC_PROFIL
00000010	45	00	01	01	00	00	0C	48	4C	69	6E	6F	02	10	00	00	E	HLino
00000020	6D	6E	74	72	52	47	42	20	50	50	5A	70	07	CE	00	02	mntrRGB	XYZ î
00000030	00	09	00	06	00	31	00	00	Normal Image Data							53 46 54	1	acspMSET
00000040	00	00	00	00	49	45	43	20	73	52	47	42	00	00	00	00		IEC sRGB
00000050	00	00	00	00	00	00	00	00	00	00	F6	D6	00	01	00	00		öö
00000060	00	00	D3	2D	48	50	20	20	00	00	00	00	00	00	00	00	Ó	HP
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
00000090	00	00	00	00	00	00	00	11	63	70	72	74	00	00	01	50	cp	prt P
...																	...	
00153890	30	48	47	E2	20	DF	ED	AF	68	7B	40	7F	FF	D9	42	61	OHGâ	Bi~h{ @ yÜBa
001538A0	73	65	53	74	61	72	74	2D	54	56	71	51	41	41	4D	41	seStart-	TVqCAAMA
001538B0	41	41	41	45	41	41	41	41	2F	2F	38	41	41	4C	67	41	AAAAEAAAAA	//8AALgA
001538C0	41	41	41	41	41	41	41	41	51	41	41	41	41	41	41	41	AAAAAAAAAÇ	AAAAAAAAA
001538D0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA	
001538E0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA	
001538F0	41	41	41	41	41	41	41	41	67	41	41	41	41	41	34	66	AAAAAAAAAg	AAAAAA4f
00153900	75	67	34	41	74	41	6E	4E	49	62	67	42	54	4D	30	68	uq4AtAnN	IbqBTM0h
...																		
001DA060	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA	
001DA070	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA	
001DA080	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA	
001DA090	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA	
001DA0A0	41	41	41	41	41	41	41	41	2D	42	61	73	65	45	6E	64	AAAAEAAAAA	-BaseEnd

Figure 8: Base64-encoded .NET module hidden within an image file.

The .NET Module

The assembly name of the .NET module (DLL) is Microsoft.Win32.TaskScheduler, which is disguised as a legitimate TaskScheduler-related system module to mislead both the victim and researchers.

Similar .NET modules have been observed in prior analyses and used by multiple malware families, including **DrakCloud** and **Agent Tesla**.

Based on my analysis, this .NET module serves as both a Remcos agent loader and a persistence installer.

The task's action is to execute the following command line every minute:

```
wscript.exe C:\Users\Public\Downloads\eOTJ0Dnny5.vbs
```

The eOTJ0Dnny5.vbs file is a duplicate of

%AppData%\90g90dfg090asdfxcv0sdf09sdf90200002f0sf0df09f0s9f0sdf0sf00d.vbs, which was downloaded earlier in the infection chain. This file was copied by the .NET module before the scheduled task was created.

2. Remcos Payload Loader

The first parameter to the VAI() method is a Base64-encoded string in reverse order. The .NET module decodes it to obtain the URL:

```
hxxps://idliya[.]com/arquivo_20251130221101.txt.
```

The .NET module downloads the file and keeps it in memory. The file contains the Remcos agent payload, which is obfuscated in two stages: Base64 encoding followed by reversing the encoded string. The bottom of Figure 11 shows the Base64-decoded Remcos payload in memory.

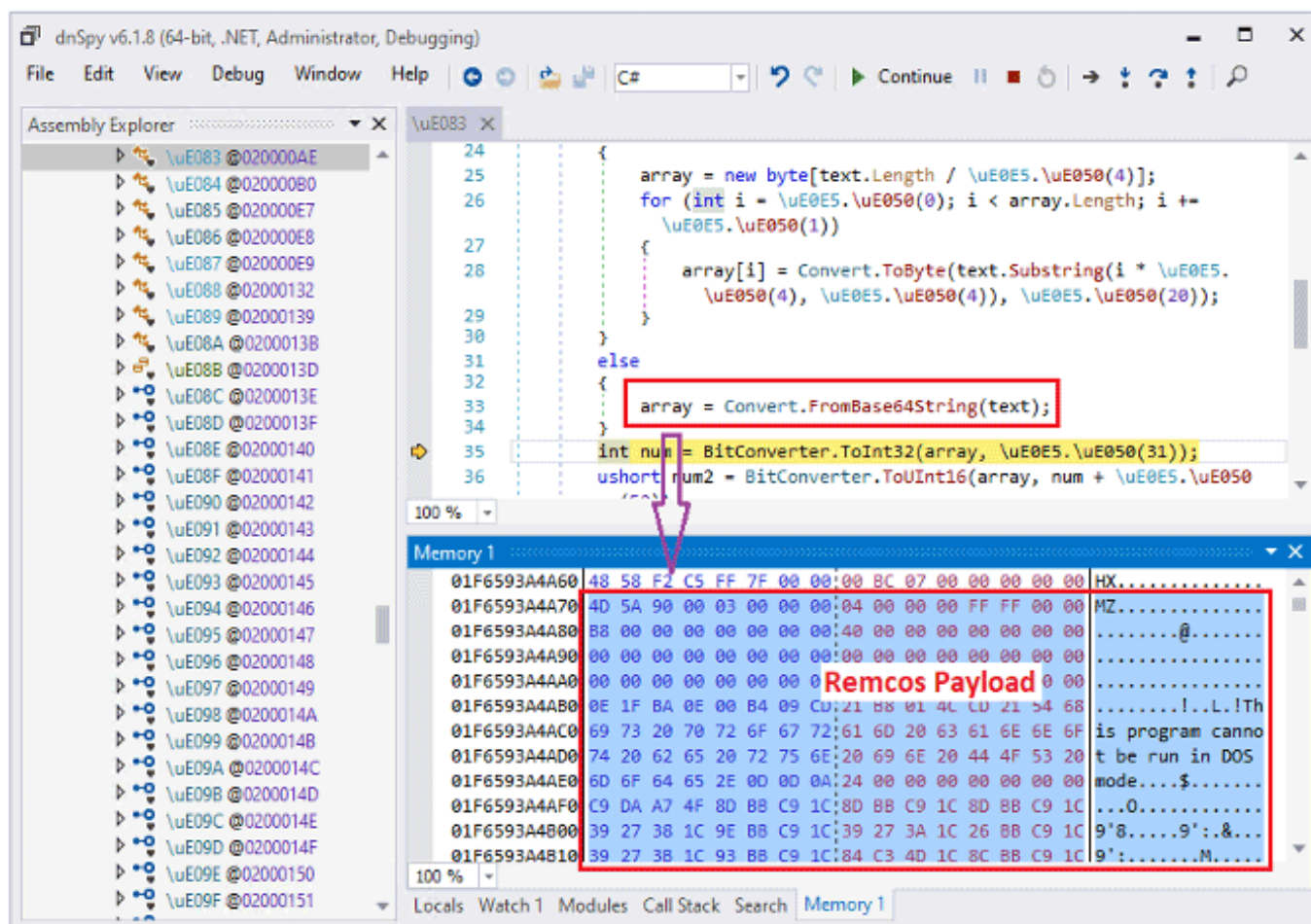
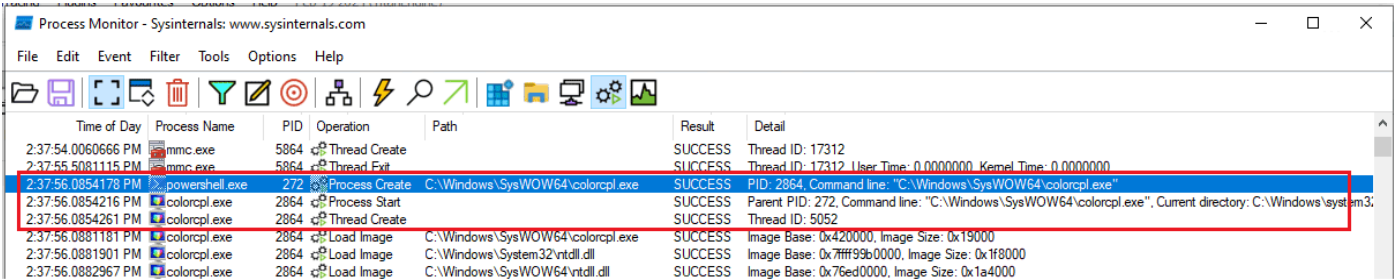


Figure 11: The Base64-decoded Remcos payload in the PowerShell process memory.

Next, the .NET module deploys the Remcos payload into a newly created colorcpl.exe process using process hollowing. The target process is specified as a parameter to the VAI() method. Figure 12 shows that colorcpl.exe was created by the .NET module while running in the PowerShell process.



The screenshot shows the Process Monitor application with a table of events. A red box highlights the following events:

Time of Day	Process Name	PID	Operation	Path	Result	Detail
2:37:54.0060666 PM	mmc.exe	5864	Thread Create		SUCCESS	Thread ID: 17312
2:37:55.5081115 PM	mmc.exe	5864	Thread Exit		SUCCESS	Thread ID: 17312 User Time: 0.0000000, Kernel Time: 0.0000000
2:37:56.0854178 PM	powershell.exe	272	Process Create	C:\Windows\SysWOW64\colorcpl.exe	SUCCESS	PID: 2864, Command line: "C:\Windows\SysWOW64\colorcpl.exe"
2:37:56.0854216 PM	colorcpl.exe	2864	Process Start		SUCCESS	Parent PID: 272, Command line: "C:\Windows\SysWOW64\colorcpl.exe", Current directory: C:\Windows\system32
2:37:56.0854261 PM	colorcpl.exe	2864	Thread Create		SUCCESS	Thread ID: 5052
2:37:56.0881181 PM	colorcpl.exe	2864	Load Image	C:\Windows\SysWOW64\colorcpl.exe	SUCCESS	Image Base: 0x420000, Image Size: 0x19000
2:37:56.0881901 PM	colorcpl.exe	2864	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7fff99b0000, Image Size: 0x1f8000
2:37:56.0882967 PM	colorcpl.exe	2864	Load Image	C:\Windows\SysWOW64\ntdll.dll	SUCCESS	Image Base: 0x76ed0000, Image Size: 0x1a4000

Figure 12: A new “colorcpl.exe” process created by PowerShell.
As the entry point function of the Remcos payload is called in the colorcpl.exe process, the Remcos agent becomes active on the compromised computer.

Remcos Payload File

Although the Remcos payload file remains fileless throughout the entire campaign, it can be dumped from the PowerShell process’s memory. As shown in Figure 13, the payload is a 32-bit executable (EXE) compiled with Microsoft Visual C++. It is not protected by any packer.

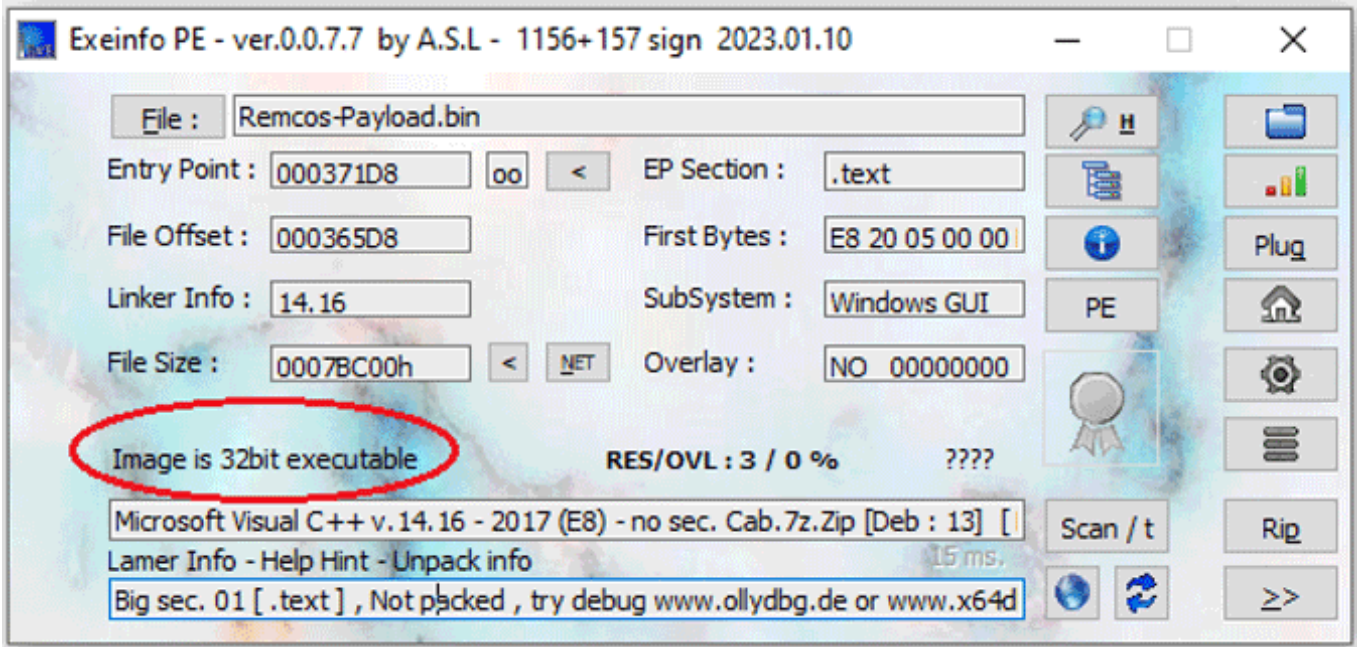


Figure 13: The view of the dumped Remcos payload in a PE analysis tool.
Based on static analysis, this Remcos variant is version 7.0.4 Pro, released on September 10, 2025. The version information is defined in the payload file, shown below.

```
.rdata:00469814      a704Pro      db '7.0.4 Pro',0
```

Remcos Configuration Block

Every Remcos agent contains an encrypted configuration block located in the resource section of the executable file named "SETTINGS." Figure 14 shows the configuration data after decryption and decompression.

colorcpl.exe - PID: 6536 - Module: colorcpl.exe - Thread: Main Thread 11336 - x32dbg

FileViewDebugTracingPluginsFavouritesOptionsHelpFeb 19 2024 (TitanEngine)

CPU

Script

Log

Breakpoints

Memory Map

Call Stack

SEH

Threads

00410291

lea eax,dword ptr ss:[esp+44]

00410295

mov dword ptr ss:[esp+44],edi

00410299

push eax

0041029A

lea edx,dword ptr ss:[esp+64]

0041029E

lea ecx,dword ptr ss:[esp+4C]

004102A2

call colorcpl.4058ED

004102A7

pop ecx

004102A8

test al,al

004102AA

je colorcpl.4102BA

004102AC

lea eax,dword ptr ss:[esp+60]

004102B0

push eax

004102B1

lea ecx,dword ptr ss:[esp+4C]

004102B5

call colorcpl.401F89

004102BA

push colorcpl.478594

004102BF

lea edx,dword ptr ss:[esp+4C]

Dump 1

Dump 2

Dump 3

Dump 4

Dump 5

Watch 1

[x=] Locals

Address	Hex	ASCTT
00779130	32 31 36 2E 39 2E 32 32 34 2E 32 36 3A 35 31 30	216.9.224.26:510
00779140	31 30 3A 31 1E 7C 1E 1E 1F 7C 52 65 6D 6F 74 65	10:1. ... Remote
00779150	48 6F 73 74 7C 1E 1E 1F 7C 31 7C 1E 1E 1F 7C 00	Host ... 1 ...
00779160	7C 1E 1E 1F 7C 01 7C 1E 1E 1F 7C 01 7C 1E 1E 1F	
00779170	7C 31 7C 1E 1E 1F 7C 31 30 30 30 30 30 7C 1E 1E	1 ... 100000 ...
00779180	1F 7C 00 7C 1E 1E 1F 7C 38 7C 1E 1E 1F 7C 72 00	 8 ... r.
00779190	65 00 6D 00 63 00 6F 00 73 00 2E 00 65 00 78 00	e.m.c.o.s...e.x.
007791A0	65 00 00 00 7C 1E 1E 1F 7C 00 7C 1E 1E 1F 7C 00	
007791B0	7C 1E 1E 1F 7C 30 7C 1E 1E 1F 7C 52 6D 63 2D 4C	... 0 ... Rmc-L
007791C0	43 43 43 54 51 7C 1E 1E 1F 7C 30 7C 1E 1E 1F 7C	CCCTQ ... 0 ...
007791D0	38 7C 1E 1E 1F 7C 6C 00 6F 00 67 00 73 00 2E 00	8 ... 1.o.g.s...
007791E0	64 00 61 00 74 00 00 00 7C 1E 1E 1F 7C 00 7C 1E	d.a.t... ...
007791F0	1E 1F 7C 00 7C 1E 1E 1F 7C 00 7C 1E 1E 1F 7C 31	 1
00779200	30 7C 1E 1E 1F 7C 00 7C 1E 1E 1F 7C 00 00 7C 1E	0
00779210	1E 1F 7C 35 7C 1E 1E 1F 7C 36 7C 1E 1E 1F 7C 53	... 5 ... 6 ... S
00779220	63 72 65 65 6E 73 68 6F 74 73 7C 1E 1E 1F 7C 00	creenshots ...
00779230	7C 1E 1E 1F 7C 00 7C 1E 1E 1F 7C 00 7C 1E 1E 1F	
00779240	7C 00 7C 1E 1E 1F 7C 00 7C 1E 1E 1F 7C 00 7C 1E	
00779250	1E 1F 7C 00 7C 1E 1E 1F 7C 00 7C 1E 1E 1F 7C 00	
00779260	7C 1E 1E 1F 7C 35 7C 1E 1E 1F 7C 31 7C 1E 1E 1F	... 5 ... 1 ...
00779270	7C 4D 69 63 52 65 63 6F 72 64 73 7C 1E 1E 1F 7C	MicRecords ...
00779280	00 7C 1E 1E 1F 7C 30 7C 1E 1E 1F 7C 30 7C 1E 1E	... 0 ... 0 ...
00779290	1F 7C 00 00 7C 1E 1E 1F 7C 00 7C 1E 1E 1F 7C 01	
007792A0	7C 1E 1E 1F 7C 30 7C 1E 1E 1F 7C 00 7C 1E 1E 1F	... 0
007792B0	7C 31 7C 1E 1E 1F 7C 52 00 65 00 6D 00 63 00 6F	1 ... R.e.m.c.o
007792C0	00 73 00 00 00 7C 1E 1E 1F 7C 72 00 65 00 6D 00	s... ... r.e.m.
007792D0	63 00 6F 00 73 00 00 00 7C 1E 1E 1F 7C 00 7C 1E	c.o.s... ...
007792E0	1E 1F 7C 00 7C 1E 1E 1F 7C 37 41 45 41 41 39 34	 7AEAA94
007792F0	44 42 46 41 30 35 31 37 30 33 37 45 46 43 44 31	DBFA0517037EFCDD1
00779300	41 45 38 34 37 32 39 44 45 7C 1E 1E 1F 7C 00 7C	AE84729DE ...

Command:

Commands are comma separated (like assembly instructions): mov eax, ebx

Paused

Dump: 00779130 -> 0077935F (0x0000230 bytes)

Figure 14: Decrypted SETTINGS data.

The configuration items are delimited by [...] (hex str: \x7C\x1E \x1E \x1F\x7C). This variant contains 57 configuration values. It uses these configuration values to control the Remcos agent's behavior on the victim's device.

The configuration block includes the C2 server information, specified as 216.9.224.26:51010:1. It is the IP address, TCP port, and a flag indicating whether TLS is enabled (1) or disabled (0).

The Remcos agent name and register key/mutex name are also defined in the configuration block as RemoteHost and Rmc-LCCCTQ, respectively.

In addition, the configuration block contains numerous flags that automatically enable or disable features, such as screen logging, keylogging, camera and microphone recording, sensitive data collection, and other capabilities.

Finally, several certificate-related values, including public keys, appearing at the end of the configuration block, are used to communicate with the C2 server over the TLS protocol.

Remcos Packet

Once Remcos establishes a connection with the C2 server, it sends the first packet containing basic information collected from the compromised computer, as shown in Figure 15.

To better explain the packet format, I reproduced the communication in a controlled local environment using plaintext packets.

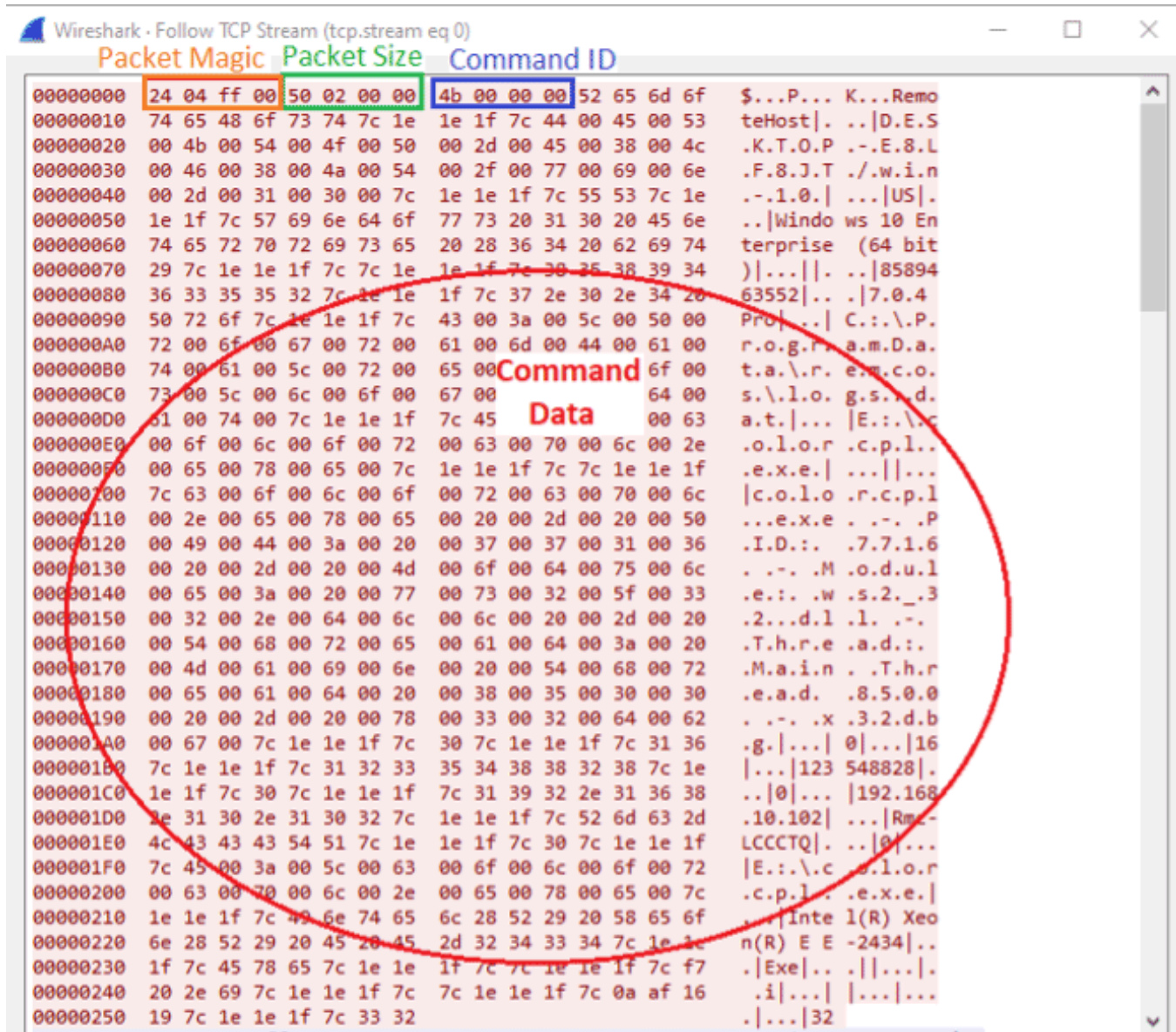


Figure 15: The packet structure of the first packet sent to the C2 server.

The packet begins with a packet magic value 0xff0424, followed by the packet size 0x250, the command ID 0x4b, and the command data, which is encrypted when TLS is enabled.

All the packets follow the same general packet structure:

packet magic | packet size | command ID | command data

This 0x4b command data consists of the following values:

- Remcos agent name
- Victim's computer name + username
- Victim's geographic location
- OS version
- Physical memory size
- Processor information

- Remcos agent version
- Full path of the process running the Remcos agent
- Title of the active application
- Remcos agent type
- Registry key name
- Remcos agent installation time
- C2 server's IP address
- Additional system and agent metadata

Each value within the command data is separated by |...| (hex str \x7C\x1E\x1E\x1F\x7C), as shown in Figure 15.

Figure 16 shows the Remcos Control Center interface on the C2 server, which displays most of the values extracted from the 0x4b command data along with live screen logging output.

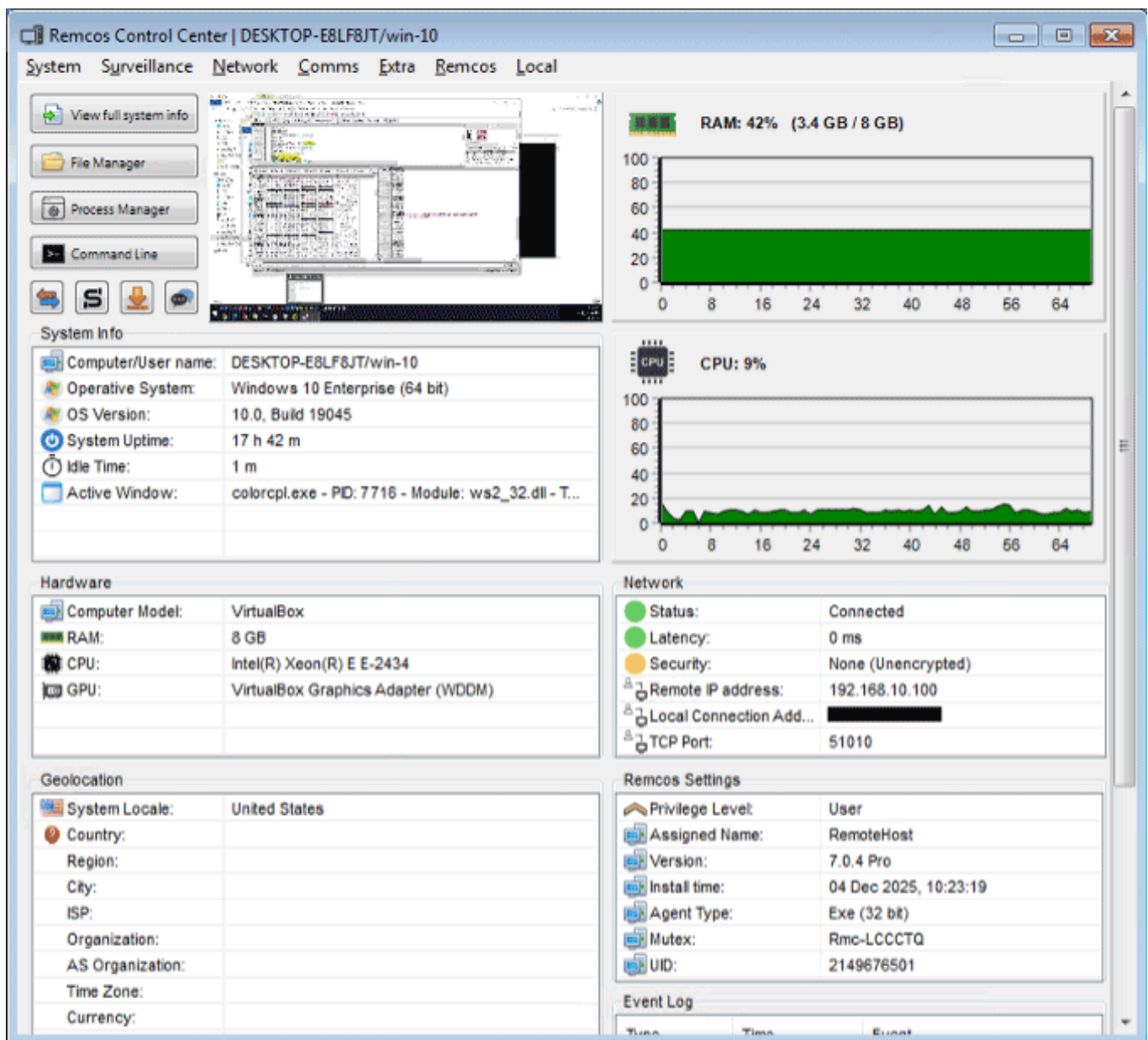


Figure 16: Remcos Control Center interface on the C2 server.

Remcos Features

This Remcos variant provides six major features for remotely managing compromised computers. To support these features, this Remcos implements a total of 211 command IDs.

System:

In this category, Remcos provides Screen Capture, File Manager, File Search, Process Manager, Service Manager, Windows Manager, Registry Editor, Installed Programs, Run System Command, Command Line, Remote Scripting, Clipboard Manager, Set Wallpaper, and Power Manager features.

These features allow the attacker to fully manage and control the victim’s computer system resources. Figure 17 shows an example of the remote Registry Editor interface on the C2 server, which the attacker can use to view and modify keys and values.

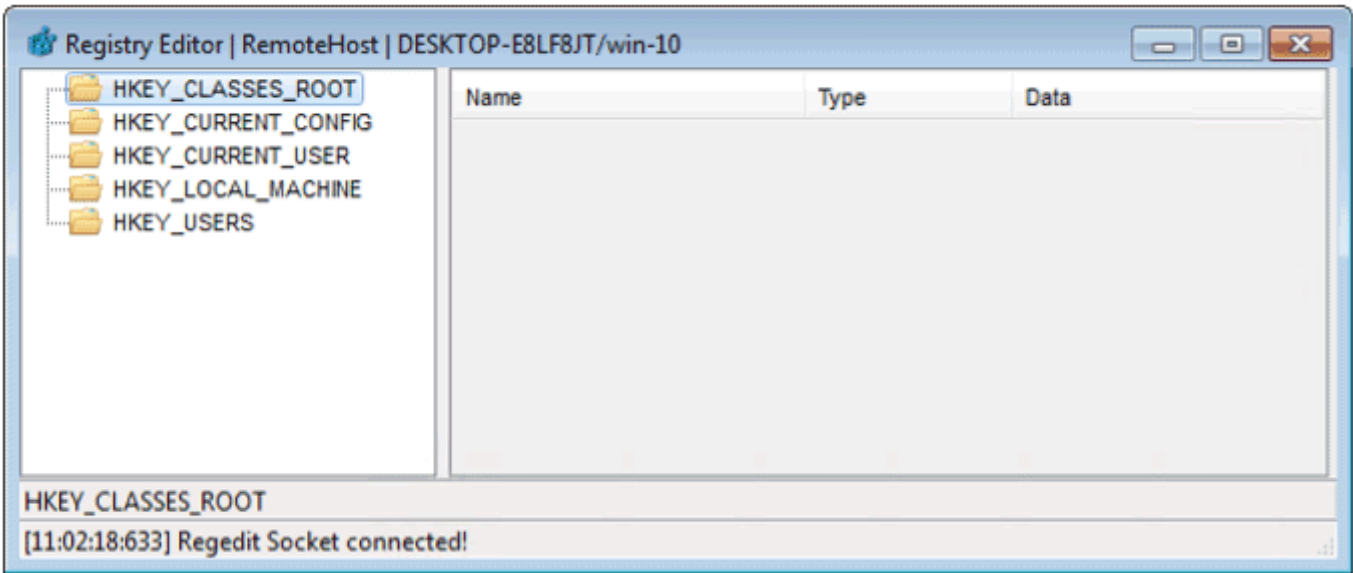


Figure 17: Remote Registry Editor.

Surveillance:

This category enables the attacker to monitor the victim’s activities using features such as Camera, Microphone, Keylogger, ScreenLogger, Browser History, Password Recovery, Cookie Recovery, and Activity Notification.

Figure 18 shows an example of a screen-logging packet. The C2 server sends command 0x10, after which the Remcos agent captures the victim’s screen into a JPEG file and sends the picture file back to C2 server.

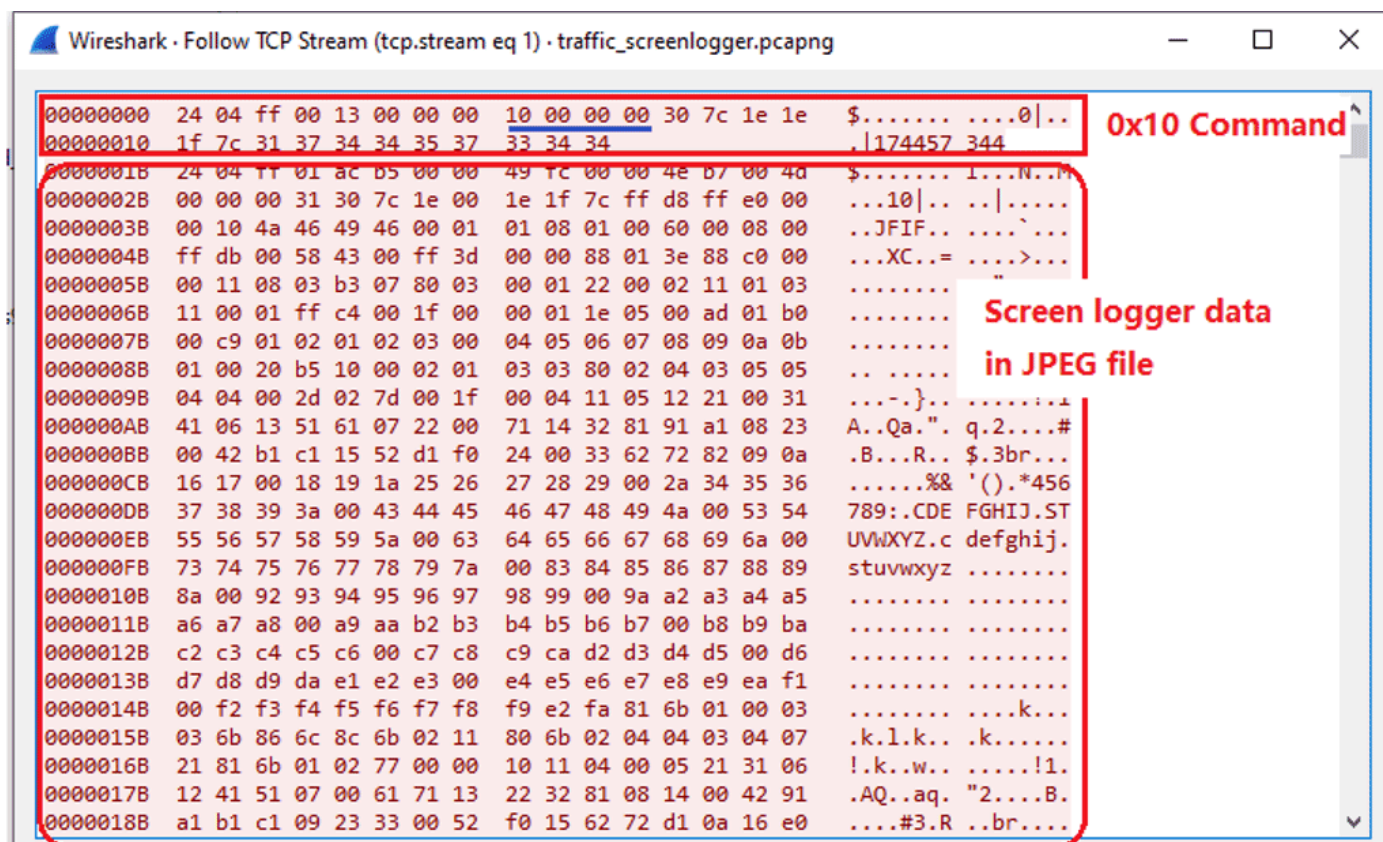


Figure 18: Screen logger packet.

Network:

Remcos provides network-related features, such as Network Monitor, Proxy, Downloader, DNS redirection, Open Webpage, and IP Info. It can remotely display the status of processes using network resources (LISTENING, ESTABLISHED).

It can also configure direct or reverse SOCKS5 proxies on the compromised system. Using the Downloader, the Remcos agent can download and execute additional executable files. It can also force the victim's default web browser to open a specified webpage.

In addition, Remcos can block the victim from accessing a website via the DNS Redirection feature. Figure 19 shows how to block an example website (e.g., www.test.com) from the victim's computer. It actually modifies the hosts file on the victim's computer.

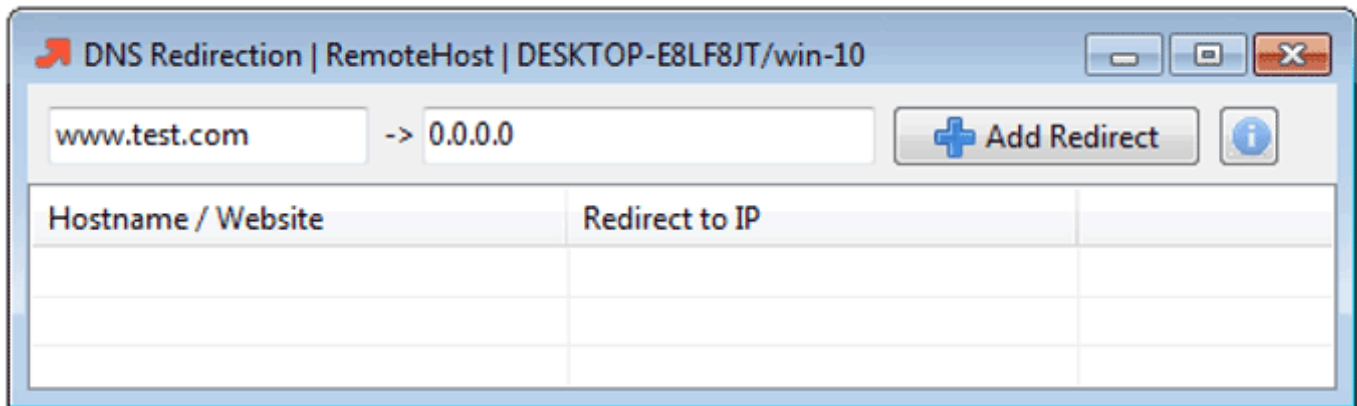


Figure 19: Set to block a website on the victim's computer.

Comms:

This category contains Chat, MessageBox, and Audio Player features. These features provide interactive communication between the victim and the attacker. Figure 20 shows the Chat feature in use on the C2 server.

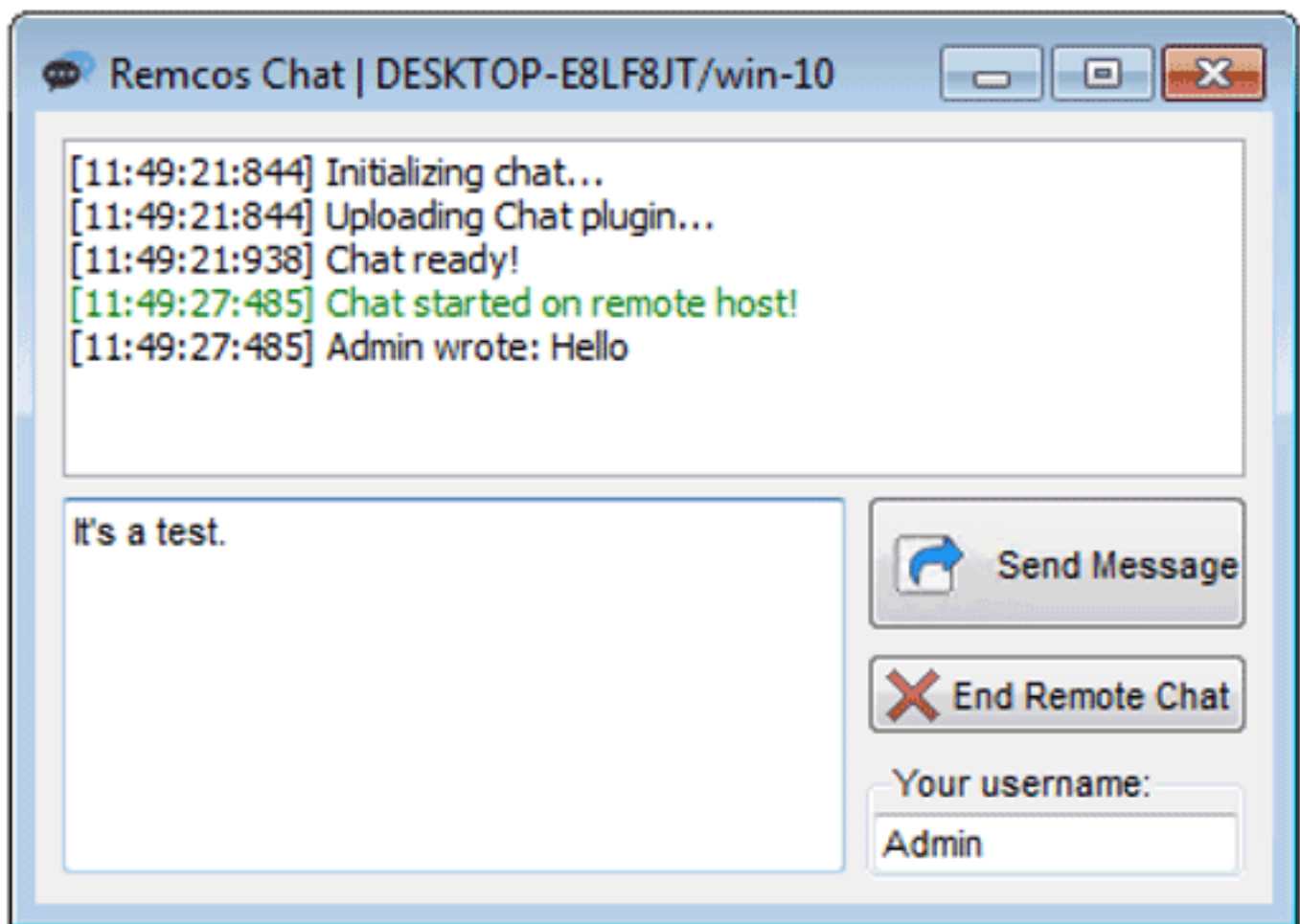


Figure 20: Remote chat with the victim.

Extra:

The Extra category includes DLL Loader, Logins Cleaner, and Extra Functions features. These features support the attacker with the ability to remotely load and execute a given DLL and clear saved credentials, browser history, and cookies on the compromised computer. An attacker can also control the victim's mouse and keyboard, monitor, CD Drive, Windows Taskbar, and Start Button.

Remcos:

This category's functions manage the Remcos agent: Rename, Elevate to Admin, Elevate to System, Ping, Reconnection, Restart, Show, Update, Close, and Uninstall.

[Previous analysis of Remcos RAT](#) has already documented most of these command IDs and their corresponding features in detail, so they are not repeated in this analysis. Please refer to that analysis for additional information on the command IDs.

Summary

This analysis introduces a newly observed phishing campaign that delivers a fileless variant of the Remcos RAT. The infection chain begins with a phishing email impersonating a Vietnamese shipping company which lures victims into opening a malicious Word document. The document then retrieves a remote RTF file, exploits a vulnerability, and executes VBScript and PowerShell code, resulting in in-memory loading of a .NET module that serves as both a loader and a persistence mechanism for the Remcos payload.

The Remcos payload (version 7.0.4 Pro) is never written to a local file. It is downloaded into memory and injected into a legitimate system process via process hollowing. Examining its encrypted configuration block reveals C2 server details, feature flags, and TLS-related certificates used for secure communication.

This analysis also breaks down the Remcos features across six categories, including system management, surveillance, networking, communication, and agent control, which give the attacker full remote control of the compromised systems.

Fortinet Protections

Fortinet customers are already protected from this campaign with FortiGuard's AntiSPAM, Web Filtering, IPS, and AntiVirus services as follows:

The URLs associated with this malicious campaign are rated as **Malicious Websites** by the FortiGuard Web Filtering service.

FortiMail recognizes the phishing email as "virus detected." In addition, real-time anti-phishing provided by FortiSandbox embedded in Fortinet FortiMail, web filtering, and antivirus solutions provides advanced protection against both known and unknown phishing attempts.

FortiGuard Antivirus service detects the Word document, RTF file, and the fileless Remcos payload file with the following AV signatures.

XML/Agent.EDC!tr.dldr

MSOffice/CVE_2017_11882.DMP!exploit

W32/Rescoms.B!tr

Fortinet FortiGate, FortiMail, FortiClient, and FortiEDR support the FortiGuard AntiVirus service, and the FortiGuard AntiVirus engine is part of each solution. As a result, customers using these products already have up-to-date protections.

To stay informed of new and emerging threats, you can [sign up](#) to receive future alerts.

We also suggest that our readers complete the free [NSE training: NSE 1 – Information Security Awareness](#), which includes information on Internet threats and is designed to help end users identify and protect themselves against phishing attacks.

The [FortiPhish Phishing Simulation Service](#), together with [Fortinet's Security Awareness and Training Service](#), uses real-world phishing scenarios to train and test employees against common social engineering tactics. By improving users' ability to recognize and respond to suspicious content—especially during high-risk periods of distraction or urgency—these services help reduce the risk of successful phishing and malware attacks.

If you believe this or any other cybersecurity threat has impacted your organization, please contact our [Global FortiGuard Incident Response Team](#).

IOCs

URLs:

hxxp://66[.]179[.]94[.]117/157/w/w.doc

hxxp://66[.]179[.]94[.]117/157/fsf090g90dfg090asdfxcv0sdf09sdf90200002f0sf0df09f0s9f0sdf0sf00ds.vbe

hxxps://idliya[.]com/assets/optimized_MSI.png

hxxps://idliya[.]com/arquivo_20251130221101.txt

Relevant Sample SHA-256:

[BL DRAFT ITL13746259.docx]

7798059D678BCA13EEEEBB44A8DB3588E4AA287701AEDE94B094B18F33B58F84

[w.doc]

A35DD25CD31E4A7CCA528DBFFF37B5CDBB4076AAC28B83FD4DA397027402BADD

[fsf090g90dfg090asdfxcv0sdf09sdf90200002f0sf0df09f0s9f0sdf0sf00ds.vbe]

E915CE8F7271902FA7D270717A5C08E57014528F19C92266F7B192793D40972F

[Remcos payload]

94CA3BEEB0DFD3F02FE14DE2E6FB0D26E29BEB426AEE911422B08465AFBD2FAA