

APT37's Pretexting-Based Targeted Intrusion: Analysis of Facebook Reconnaissance and Software Tampering Attacks

 genians.co.kr/en/blog/threat_intelligence/pretexting

Genians

April 12, 2026



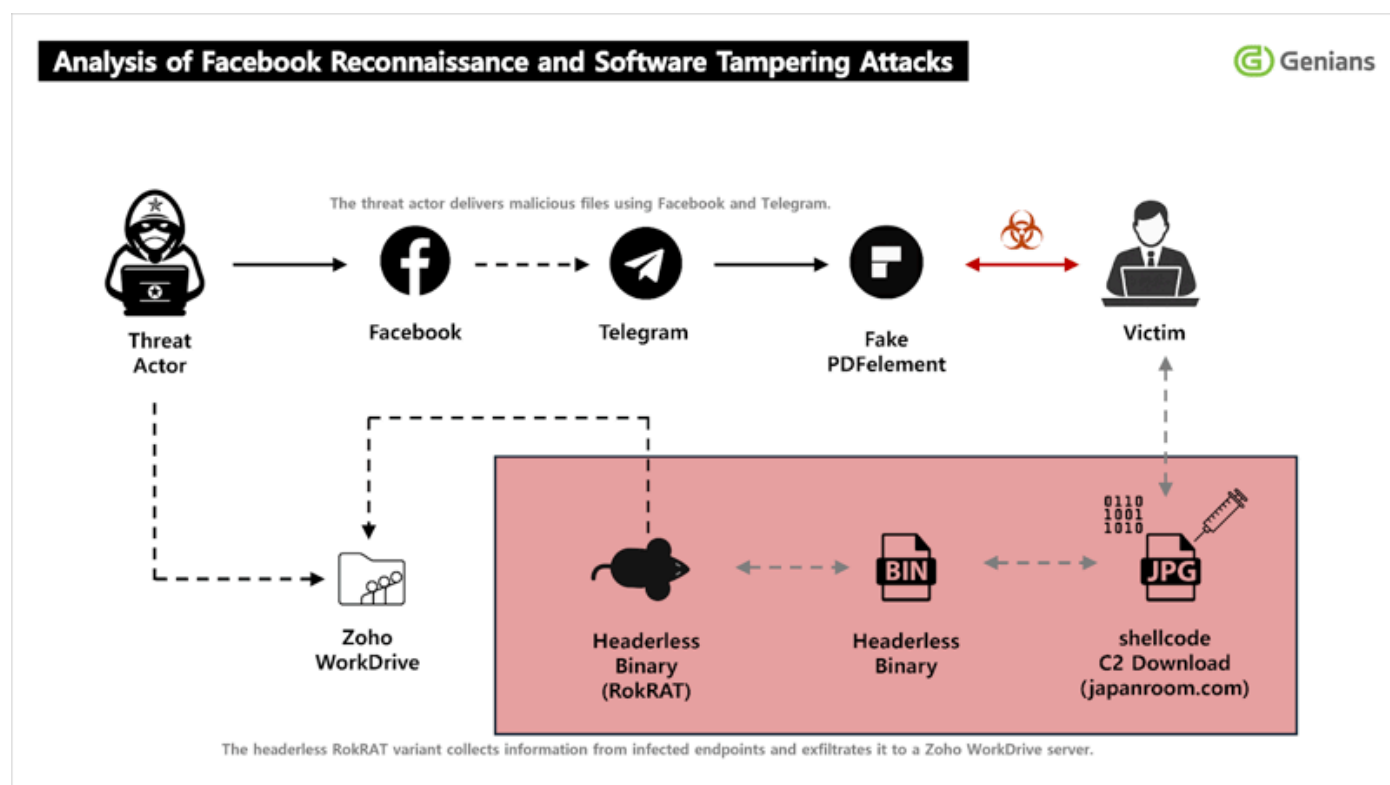
◆ Key Findings

- Conducted reconnaissance using two Facebook accounts claiming to be from Pyongyang and Pyongsong, North Korea
- Built trust by adding targets as Facebook friends, then moved the conversation to Messenger and lured them using specific topics
- Tricked targets into installing a dedicated PDF viewer under the pretense of sharing an encrypted PDF document on military weapons
- Executed shellcode and gained initial access through a carefully tampered Wondershare PDFelement installer
- Delivered follow-up commands through a JPG-disguised payload delivered via the Seoul branch website of a Japanese real estate information service
- Behavior-based EDR is required to detect the identified indicators of compromise (IoCs) and respond to APT evasion techniques

1. Overview

Genians Security Center conducted an in-depth analysis of a targeted intrusion campaign carried out by the APT37 threat actor through a social networking platform.

The analysis showed that the threat actor used two Facebook accounts with their location set to Pyongyang and Pyongsong, North Korea to identify and screen targets. After building trust through friend requests, the actor moved the conversation to Messenger and used specific topics to lure targets as part of the initial social engineering stage of the attack.



[Figure 1-1] Overall Attack Flow

The threat actor used a pretexting tactic, posing as though they were sharing encrypted PDF documents containing technical information on military weapons through Facebook Messenger or Telegram, and then persuading targets to install a dedicated PDF viewer to open them.

Pretexting is a social engineering technique in which a threat actor creates a false scenario or justification in advance and uses it to induce the victim to take a specific action.

In this case, the actor claimed that a dedicated viewer was needed to open the encrypted military documents, leading the victim to install the malicious program.

The PDF viewer was identified as a carefully tampered Wondershare PDFelement installer, which, when executed, activated embedded shellcode and enabled initial compromise.

The actor also appears to have used the Seoul branch website of a Japanese real estate information service as C2 infrastructure, delivering and executing subsequent commands through a malicious payload disguised as a JPG image file as part of a multi-stage attack chain.

This is assessed as a highly evasive strategy that combines legitimate software tampering, abuse of a legitimate website, and file extension masquerading.

This threat intelligence report provides a systematic analysis of the attacker's tactics, techniques, and procedures (TTPs), focusing on key technical artifacts such as Facebook-based target reconnaissance, shellcode execution through a tampered installer package, concealed C2 communication channels, and the delivery mechanism of an image-disguised payload.

In addition to signature-based detection using the identified indicators of compromise (IoCs), the report also highlights the need for an EDR-centered response framework based on endpoint behavior detection to effectively identify attempts to masquerade as legitimate processes and abuse legitimate infrastructure.

2. Background

The state-sponsored threat group APT37 continues to refine its social engineering techniques and evasion strategies in targeted attacks.

In "[Operation. ToyBox Story](#)," reported in May 2025, spear-phishing attacks impersonating a Korean national security strategy think tank and detection evasion tactics using cloud-based C2 channels were identified.

This was followed by the August report, "[RoKRAT Shellcode and Steganographic Threats: Analysis and EDR Response Strategies](#)," which identified an advanced evasion technique in which a malicious payload was hidden inside an image file and executed through shellcode. In December of the same year, "[Operation Artemis: Analysis of HWP-Based DLL Side Loading Attacks](#)" revealed signs of initial compromise through a DLL sideloading technique delivered via an HWP document, in which a malicious DLL was loaded during the normal document viewing process.

The threat actor employed a multi-stage attack chain that combined trust-building through platforms such as Facebook and Telegram, pretexting to induce targets to install malicious software, shellcode execution through a tampered installer package, and concealed C2 communications routed through a legitimate website.

This is assessed as a highly evasive strategy designed to bypass existing security controls by combining tampered legitimate software, abuse of legitimate infrastructure, and JPG file extension masquerading.

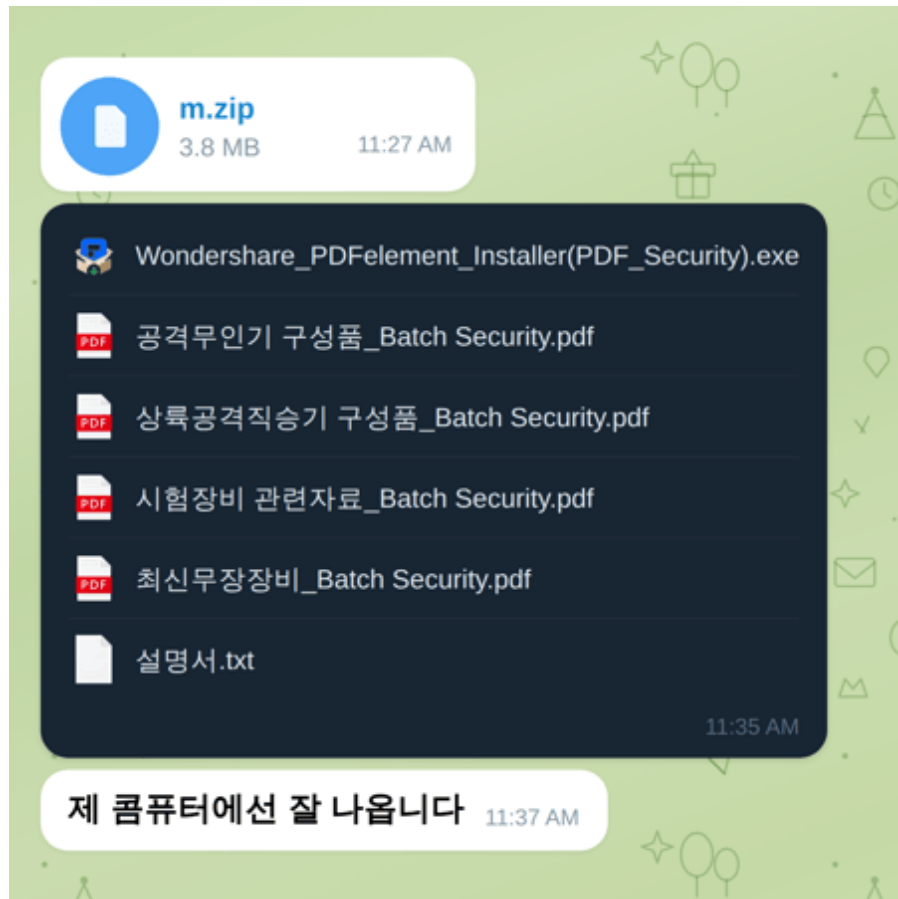
Against this backdrop, this report examines the evolution of APT37's social network-based target reconnaissance and intrusion tactics, and systematically analyzes the technical characteristics of the group's tactics, techniques, and procedures (TTPs).

3. Attack Scenario

In the initial compromise stage, a multi-step social engineering tactic that combined trust-building through social networks with pretexting was identified as the primary attack vector.

The threat actor first connected with targets on Facebook, then built initial trust by exchanging greetings and introductions through one-on-one Messenger conversations.

The actor then engaged the targets by offering confidential materials related to military weapons and, once their interest was established, moved the conversation to a separate, more secure messaging platform such as Telegram to deliver the malicious file.



[Figure 3-1] File Delivery via Telegram

The threat actor delivered a ZIP archive named "m.zip" through Telegram. The archive contained a malicious executable disguised as a PDF viewer, multiple decoy PDF files with military-related document titles, and a TXT file disguised as a user guide.

Notably, the Telegram conversation included the term "컴퓨터," a North Korean-style transliteration of "컴퓨터(*computer*)," while the file "설명서(*instructions*).txt" contained expressions such as "프로그램(*program*)" and "화일(*files*)." These are notable linguistic indicators that may help infer the threat actor's origin or affiliation.

- 1.Wondershare_PDFelement_Installer(PDF_Security)프로그램을 설치합니다.
- 2.Wondershare프로그램을 열고 Batch Security항목을 선택합니다.
- 3.Batch Security에 들어간다음 암호화된 화일들을 선택하여 놓고 복호화 단추를 선택합니다.

[Figure 3-2] Partial View of the Contents of "설명서.txt"

The file delivered through Telegram was provided as an encrypted archive. Analysis showed that, when extracted using a separately shared password, it contained multiple encrypted PDF documents along with an EXE installer.

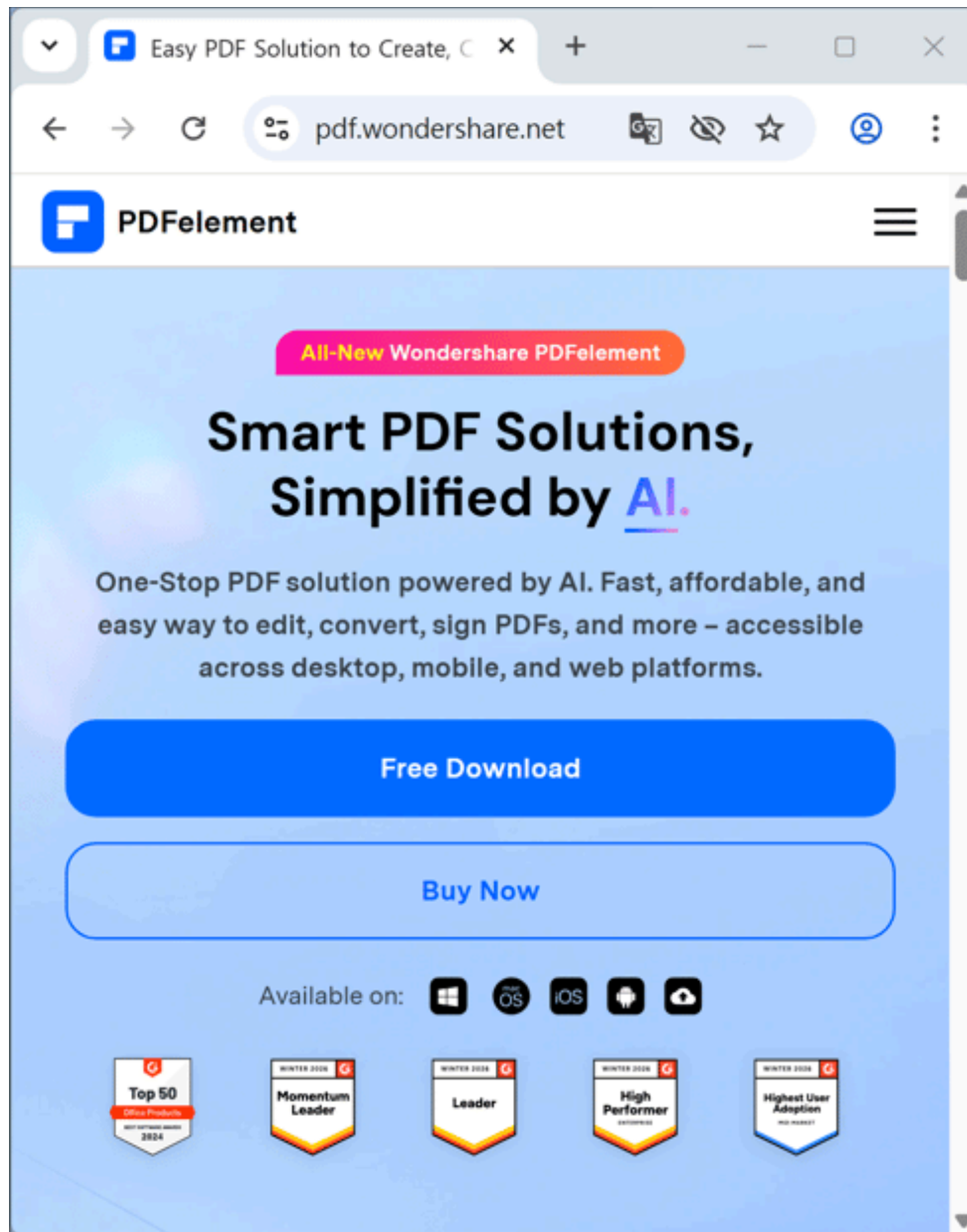
Although the installer closely resembled the PDFelement installer officially distributed by Wondershare, it was confirmed to be a malicious file created by carefully tampering with the legitimate installer.

When the installer is executed, the embedded shellcode runs and carries out malicious commands, leading to initial compromise.

4. Detailed Analysis

4-1. Malicious File Flow

The malicious file distributed through Telegram was found to closely mimic legitimate software.



[Figure 4-1] PDFelement Official Website

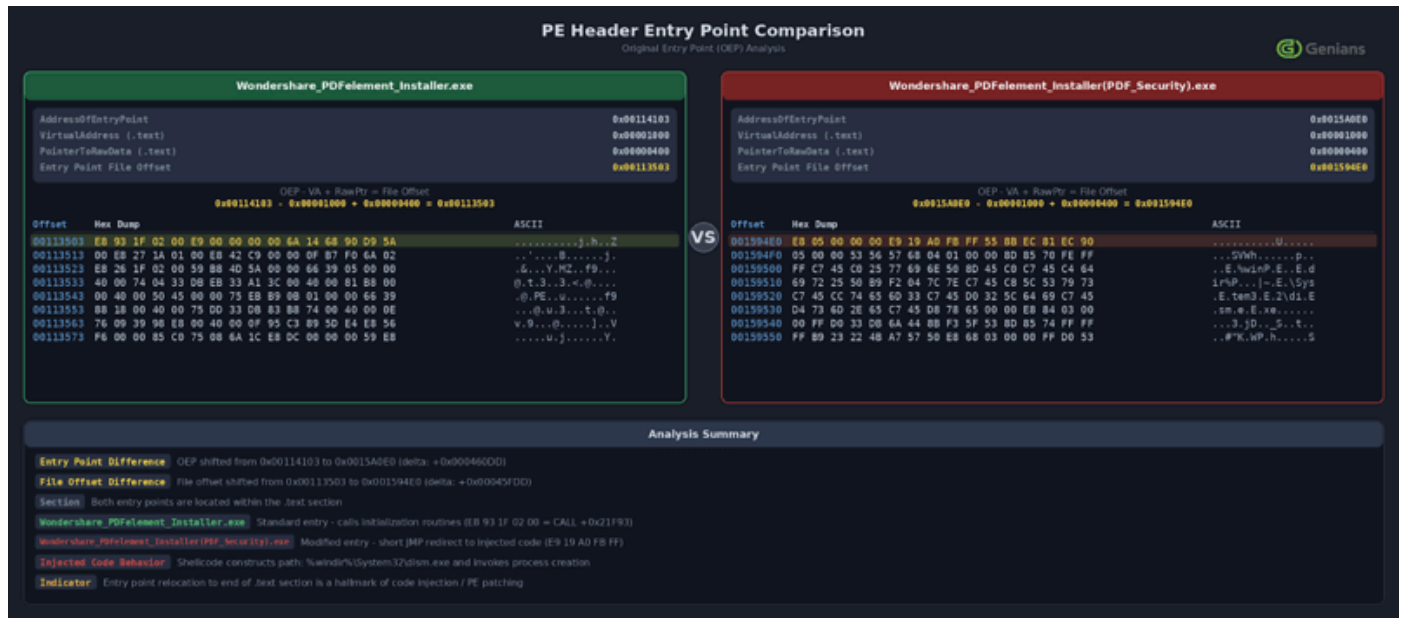
While the legitimate installer provided on Wondershare's official website is named "Wondershare_PDFelement_Installer.exe," the malicious file distributed by the threat actor through Telegram used the name "Wondershare_PDFelement_Installer(PDF_Security).exe."

This indicates a filename masquerading technique in which the phrase "(PDF_Security)" was intentionally added to the original filename to make the file appear to be a legitimate security-related update or enhanced version, leading the target to execute it without suspicion.

In addition, while the legitimate installer distributed through the official website carries a valid Wondershare digital signature that can be used to verify the file's integrity and origin, the tampered installer distributed through Telegram did not contain a digital signature.

This suggests that the original code signature was invalidated during the process of tampering with or independently reconstructing the legitimate binary. The absence of a digital signature is therefore one of the key indicators that the file was not legitimate software distributed through an official channel.

The threat actor retained the main function of the tampered installer and modified only the entry point.



[Figure 4-2] Entry Point Comparison

The legitimate installer has an AddressOfEntryPoint of 0x00114103, located near the beginning of the .text section where the program code resides. This reflects a normal structure in which standard initialization begins. The corresponding file offset is 0x00113503, which marks the normal starting point of execution.

- 0x00113503
 - OEP - VirtualAddress + PointerToRawData
 - 0x00114103 - 0x00001000 + 0x00000400

In contrast, the tampered installer has a modified entry point of 0x0015A0E0.

This corresponds to file offset 0x001594E0 near the end of the .text section, approximately 0x460DD beyond the entry point of the legitimate file.

The threat actor inserted about 2 KB of malicious shellcode into an unused code cave near the end of the .text section, then reset the entry point so that this code would run first when the program was executed.

The execution flow from the tampered entry point, 0x0015A0E0, is as follows.



The shellcode body begins like a typical function. It saves the current stack state with `PUSH EBP` and `MOV EBP, ESP`, allocates 1,424 bytes of temporary workspace on the stack with `SUB ESP, 0x590`, and preserves key register values with `PUSH EBX`, `PUSH ESI`, and `PUSH EDI` to establish the execution environment for the subsequent malicious actions.

8/26

This prevents the path string from appearing in plaintext within the file, helping the malware evade detection techniques that rely on specific string patterns.

Note: DISM (Deployment Image Servicing and Management) is a command-line utility in Microsoft Windows for creating, modifying, and managing system image files such as WIM, VHD, and FFU. It is commonly used by system administrators and advanced users to maintain Windows installation images and offline operating systems, and is included by default in Windows 7 and later.

The shellcode does not directly reference the program's Import Address Table (IAT). Instead, it resolves the addresses of required system functions by traversing the Process Environment Block (PEB), an internal process structure maintained by Windows.

This hash-based function resolution routine is repeatedly used throughout the shellcode to dynamically obtain the addresses of key system functions at runtime, including `GetEnvironmentVariableA`, `CreateProcessA`, `VirtualAlloc`, `VirtualAllocEx`, `WriteProcessMemory`, and `Sleep`.

Once the required function addresses have been resolved, the shellcode calls `ExpandEnvironmentStringsA` to convert "%windir%" in the previously assembled path "%windir%\System32\dism.exe" into the actual system path, such as "C:\Windows."

It then prepares the 68-byte `STARTUPINFO` structure and calls `CreateProcessA` with `CREATE_SUSPENDED(0x4)` set in the creation flags.

As a result, "dism.exe," a legitimate Windows system utility, is created in a suspended state.

A suspended state means that the process has been loaded into memory but has not yet started running, and the shellcode includes a reliability mechanism that retries process creation up to five times at 0.1-second intervals if it fails.

The reason for creating the process in a suspended state is to prepare for injecting malicious code by manipulating its memory space before execution begins.

Once the `dism.exe` process is successfully created, `VirtualAllocEx` is used to allocate a memory region within the process with read, write, and execute permissions enabled (`PAGE_EXECUTE_READWRITE`).

The shellcode then decrypts the encrypted malicious code embedded within it through an XOR loop and uses `WriteProcessMemory` to write the decrypted code into the allocated memory region of "dism.exe."

The same reliability logic is applied at this stage, with up to five retries in the event of failure. At the same time, the shellcode separately prepares the network functionality required for C2 communication.

It dynamically constructs the library name "wininet.dll" on the stack, calls LoadLibraryA to load the library into memory, and then uses VirtualAlloc to allocate a data reception buffer of approximately 10 MB.

Data received from the server is decrypted through an XOR operation and then used in the next stage of the attack. Once all malicious actions have been completed, the shellcode returns with a RET instruction.

Execution is then redirected to the original legitimate entry point through the prearranged JMP 0x00114103 instruction, and the normal Wondershare PDFelement installation process begins.

To the user, the program appears to install normally. In the background, however, a series of malicious actions has already been completed, including process injection into a legitimate process, decryption of the encrypted payload, and the establishment of a communication channel with the external attacker-controlled server.

By carefully combining malicious activity with the normal installation process, the attacker made it extremely difficult for the user to notice anything unusual.

4-2. In-Depth Analysis of the C2 Decryption Logic

The shellcode execution flow consists of 12 stages, each responsible for a distinct function within a specific offset range.

After entering through the tampered entry point in Stage 1, the shellcode sets up its execution environment and dynamically constructs the target path in Stages 2 and 3. Stages 4 through 7 complete the core attack chain, including process creation, memory allocation, payload decryption and remote injection, and thread execution.

In Stage 8, the shellcode returns to the original entry point and continues with the normal installation process, helping avoid user suspicion. Stages 9 and 10 handle HTTP communication with the C2 server and the download of the second-stage payload within the remote thread.

Stages 11 and 12 are support routines used repeatedly throughout the shellcode, providing PEB-based API hash resolution and position-independent address calculation. The table below summarizes the offset range, function name, and key behavior of each stage.

Stage	Offset	Function	Key Operations
1	0x0000-0x000A	Entry Point Redirect	CALL \$+5 to enter the shellcode, then secures a return path to the OEP with JMP 0x00114103

2	0x000A-0x0016	Shellcode Prologue	Creates the stack frame and allocates 1,424 bytes (0x590) of local variable space
3	0x0016-0x005C	Path String Construction	Dynamically constructs the path "%windir%\System32\dism.exe" on the stack using seven 4-byte MOV DWORD PTR instructions
4	0x005C-0x00E3	API Resolution & CreateProcessA	Resolves APIs through PEB hash-based lookup and creates the "dism.exe" process with the CREATE_SUSPENDED (0x04) flag
5	0x00E3-0x011E	VirtualAllocEx	Allocates an executable memory region in the remote process with PAGE_EXECUTE_READWRITE (0x40) permissions
6	0x011E-0x01E1	XOR Decrypt & WriteProcessMemory	Uses a get-EIP stub to obtain the blob address, decrypts it with XOR using the key 0x6D, and writes the payload and C2 URL into the remote process with WriteProcessMemory
7	0x01E1-0x0224	CreateRemoteThread	Creates a remote thread in the "dism.exe" process and begins execution of the injected malicious code
8	0x0224-0x0228	Epilogue & Return to OEP	Restores the registers, returns with RET, and then uses JMP 0x00114103 to resume the normal installation process
9	0x0229-0x02D1	wininet.dll Loader	Dynamically loads "wininet.dll," allocates a 10 MB (0xA00000) download buffer, receives the second-stage payload, verifies the 0x55 0x8B signature, and executes it directly in memory

10	0x02D2- 0x03E4	HTTP C2 Download	Receives data through the sequence InternetOpenA → InternetOpenUrlA → HTTP 200 verification → InternetReadFile loop
11	0x03E5- 0x054F	PEB Walker / API Hash Resolver	Traverses FS:[0x30] → PEB → Ldr → InLoadOrderModuleList, compares combined DLL and function hashes using ROR-13 hashing, and returns the absolute API address
12	0x0550- 0x055E	get-EIP Stub	Dynamically calculates the address of the encrypted data blob (0x055F) based on the current execution location using the CALL/MOV EAX,[ESP]/ADD EAX,0xA pattern - Blob (Binary Large Object)

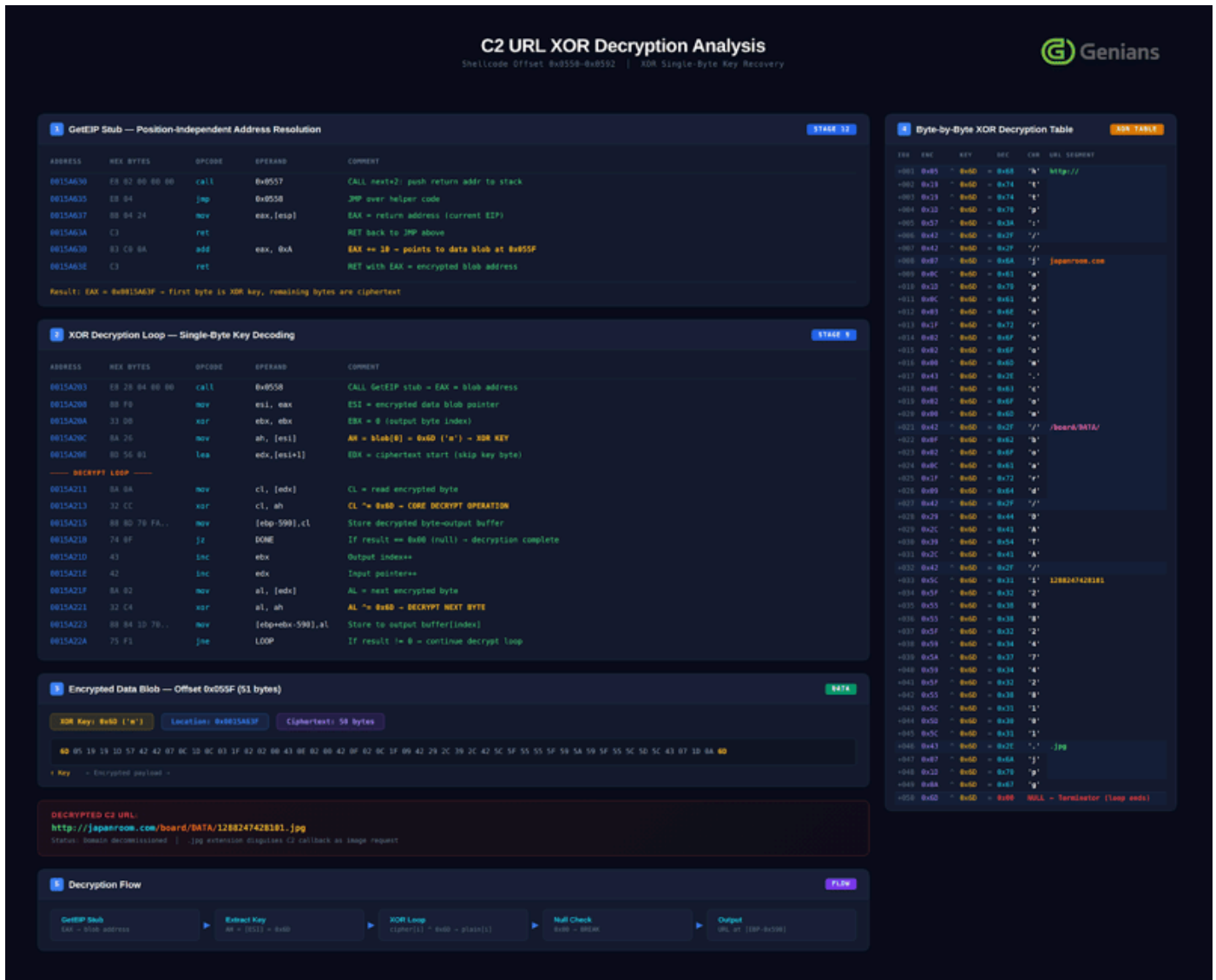
[Table 4-1] Summary of the Overall Shellcode Execution Flow

The C2 URL embedded in the shellcode is protected with single-byte XOR encryption and decrypted at runtime before use.

The infographic below illustrates this decryption process in five sections.

The first section shows the assembly structure of the get-EIP stub at offset 0x0550.

Using the CALL/MOV EAX,[ESP]/ADD EAX,0xA instruction pattern, it dynamically calculates the current execution location and returns in the EAX register the absolute address of the encrypted data blob (0x055F), which is located exactly 10 bytes after the stub.



[Figure 4-4] Analysis of the C2 URL XOR Decryption Process

The second section provides a detailed explanation of the assembly code used in the XOR decryption loop.

After extracting the first byte of the blob, 0x6D (ASCII "m"), as the XOR key, the code sequentially XORs each byte starting from the second byte with 0x6D to recover the plaintext. The loop terminates when the result becomes 0x00, or a null byte.

The third section presents the original structure of the encrypted data blob.

Of the 51 bytes located at offset 0x055F, the first byte is the XOR key, the following 49 bytes are the encrypted URL payload, and the final byte, 0x6D, has the same value as the key. When XORed with the key, it produces 0x00, allowing the decryption loop to terminate naturally without separate length information.

The fourth section provides a byte-by-byte XOR decryption table for the full 50-byte sequence.

Each row lists the encrypted value, XOR key, decrypted result, ASCII character, and corresponding URL segment, showing how the protocol (http://), domain (japanroom[.]com), path (/board/DATA/), filename (1288247428101), and extension (.jpg) are restored in sequence.

The fifth section shows the fully decrypted C2 URL.

This URL uses the ".jpg" extension to disguise the request as a normal image file request, allowing it to evade network monitoring and URL filtering.

`http://japanroom[.]com/board/DATA/1288247428101.jpg`

4-3. Second-Stage Payload Analysis

Once the file "1288247428101.jpg" has been successfully downloaded from the C2 server, the shellcode treats the received data not as an image, but as an encrypted second-stage payload.

The shellcode then immediately performs a second round of XOR decryption on the received data.

It uses the first byte of the received data as the XOR key and applies the XOR operation sequentially from the second byte to the last. This is the same single-byte XOR technique used to decrypt the C2 URL.

In the actual file, the first byte is 0x6F, which is used as the key to decrypt the remaining data. Once decryption is complete, the shellcode verifies whether the decrypted data is valid executable code.

Specifically, it checks whether the first byte of the decrypted data is 0x55, corresponding to the PUSH EBP instruction, and whether the second byte is 0x8B, the first byte of a MOV instruction. These two bytes match the beginning of the standard x86 function prologue, 55 8B EC (PUSH EBP / MOV EBP, ESP).

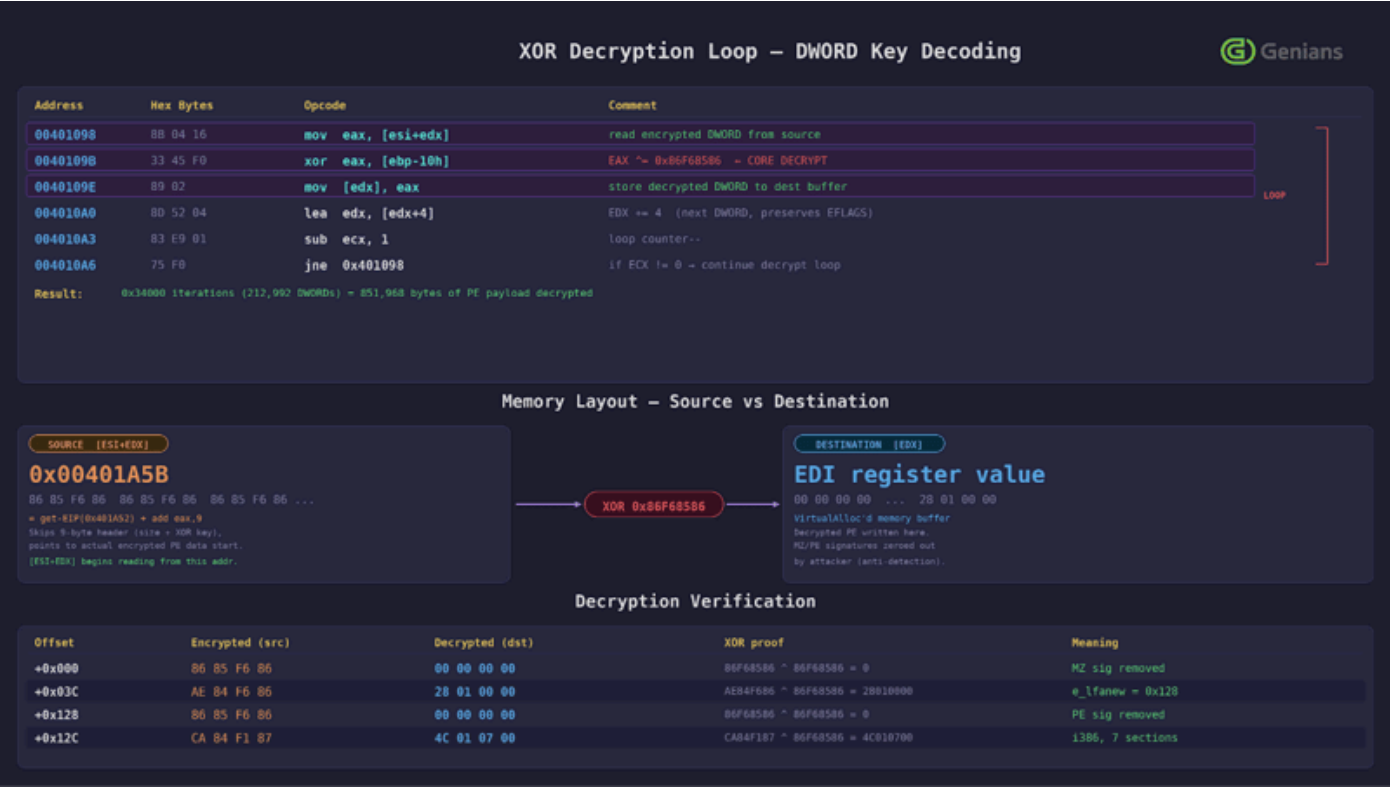
If signature validation fails, the shellcode waits five seconds using Sleep and then retries the download from the C2 server from the beginning.

This retry loop is designed to run indefinitely, allowing the malware to continue attempting the download until it receives a valid payload, even in the event of temporary C2 server issues or transmission errors.

If the signature check succeeds, the shellcode uses LEA EAX, [ESI+1] to calculate the starting address of the actual shellcode by skipping the XOR key byte, and then executes the decrypted second-stage shellcode directly in memory with CALL EAX.

Throughout this process, the second-stage payload is never written to disk as a file. Instead, it is decrypted in memory and executed immediately in a fileless manner.

It is also designed to evade network detection, file-based detection, and static analysis at the same time by disguising the request as a normal image request through a URL with a ".jpg" extension and combining XOR-encrypted transfer data with memory-only execution.



[Figure 4-5] Analysis of the XOR Decryption Loop for the Second-Stage Payload

The shellcode decrypted with the 0x6F XOR key first determines its location in memory. To do this, it uses a technique known as a get-EIP gadget (0x401A47). Unlike a typical program, which can readily identify the address at which it was loaded, shellcode must determine its own current address because it may be loaded at a different location each time.

A get-EIP gadget takes advantage of the x86 processor behavior in which the call instruction automatically pushes the address of the next instruction onto the stack. The shellcode deliberately issues a call to a nearby address and then retrieves the return address from the stack to determine its current location, 0x401A52.

At this location, there is a small 9-byte information block. It contains the total size of the encrypted data, 0xD0000 (851,968 bytes), and the key value used for decryption, 0x86F68586. After reading these two values, the shellcode skips over the 9-byte block and identifies 0x401A5B as the starting address of the actual encrypted data.

Next, the shellcode requests memory allocation from Windows. It calls VirtualAlloc to reserve an empty memory region of size 0xD0400, which is used to store the decrypted output. The address of this region is stored in the EDI register.

The actual decryption process then begins. The shellcode processes the data in 4-byte units (DWORDs) and decrypts the entire payload. It reads each encrypted 4-byte value from the original location at 0x401A5B, XORs it with the key 0x86F68586 to recover the original data, and writes the result to the newly allocated memory referenced by EDI.

Once decryption is complete, the newly allocated memory contains a fully restored PE file. However, to evade detection, the threat actor had intentionally removed the MZ and PE signatures, the most recognizable identifiers of an executable file. Even so, the internal file structures, including the section table and import table, are restored correctly, allowing the shellcode's built-in loader to parse them, map the file into memory, and execute the final malicious payload.

The compile timestamp has been zeroed out, intentionally hindering attempts to trace when the payload was built. One of the most notable features of this payload is its abuse of a legitimate cloud service as a command-and-control (C2) channel. It receives attacker commands and exfiltrates stolen data through the OAuth2 API of Zoho WorkDrive, with two sets of OAuth authentication tokens hardcoded in the binary for this purpose.

Because this communication relies on a legitimate cloud service, it is difficult to distinguish from ordinary business traffic, allowing it to effectively evade detection by network security devices. The main malicious functions carried out by the payload are as follows.

- Captures screenshots by determining the desktop resolution with GetDC and GetSystemMetrics, copying the full screen with CreateCompatibleBitmap and BitBlt, converting it with the GDI+ JPEG encoder, and transmitting it to the C2 server.
- Supports remote command execution in the format of cmd.exe /c "%s", allowing it to run arbitrary system commands delivered by the C2 server.
- Collects host and user information with GetComputerNameW and GetUserNameW, and enumerates disk configuration with GetLogicalDrives and GetDiskFreeSpaceExW.
- Performs system reconnaissance by querying the external service ipinfo[.]io/json to identify the victim system's public IP address and geolocation.
- Uses the CryptoPP library to support AES-CBC symmetric encryption and RSA public-key encryption.

To evade security products, it enumerates running processes with CreateToolhelp32Snapshot and Process32NextW and checks for the presence of 360Tray.exe, a component of the Qihoo 360 security product.

It can also place a file disguised as the legitimate software updater "OfficeUpdate.exe" in the AppData path, depending on the command received.

In addition, it contains 21 different User-Agent strings to disguise its network traffic as normal web browser activity.

Client_id	1000.PZCKAWFHJPS9A6WU5G*****
	1000.LS8YMLB2Z0BSHSL3VW*****
Refresh_Token	1000.c637b3e7d74c2d678663454d16311b15.12b060c2ad23ecafc959*****
	1000.085128b4e96633c82beb2101f5c525e4.6ee7ce2c1952f14794d0*****
Client_Secret	a9e22f09e5e54c7ee7b3e9ac4340d7*****
	4cb82c2055249110fc86893de90a2d*****

[Table 4-2] Zoho WorkDrive Credentials

The malware collects the computer name, user name, Windows version, IP address and geolocation of the infected system, including country and city, SMBIOS hardware identifiers, the list of running processes, desktop screenshots in JPEG format, and documents and smartphone audio recordings with specific extensions such as ".DOC," ".XLS," ".PPT," ".PDF," ".HWP," ".TXT," ".M4A," and ".AMR," then exfiltrates them to Zoho WorkDrive cloud storage.

The collected data is encrypted with AES-256-CBC before transmission.

In addition, the hardcoded string "**#FBI#TOOLKIT#GIDRA@TEAM, @-IV-FBI-SERVER2**" embedded in the binary is used for AES decryption of the response data when an HTTP GET request succeeds.

0AC840	72	00	6D	00	61	00	6C	00	00	00	00	2E	00	58	00	r.m.a.l.....X.	
0AC850	4C	00	53	00	00	00	00	00	2E	00	44	00	4F	00	43	L.S.....D.O.C.	
0AC860	00	00	00	00	2E	00	50	00	50	00	54	00	00	00	00	P.P.T.....	
0AC870	2E	00	54	00	58	00	54	00	00	00	00	2E	00	4D	00	..T.X.T.....M.	
0AC880	34	00	41	00	00	00	00	00	2E	00	41	00	4D	00	52	4.A.....A.M.R.	
0AC890	00	00	00	00	2E	00	50	00	44	00	46	00	00	00	00	P.D.F.....	
0AC8A0	2E	00	48	00	57	00	50	00	00	00	00	41	00	6C	00	..H.W.P.....A.l.	
0AC8B0	6C	00	00	00	47	00	45	00	54	00	00	00	68	00	74	l...G.E.T...h.t.	
0AC8C0	74	00	70	00	3A	00	2F	00	2F	00	31	00	32	00	37	t.p.:././..1.2.7.	
0AC8D0	2E	00	30	00	2E	00	30	00	2E	00	31	00	00	00	00	..0...0...1.....	
0AC8E0	2F	00	00	00	4F	00	66	00	66	00	69	00	63	00	65	/...0.f.f.i.c.e.	
0AC8F0	55	00	70	00	64	00	61	00	74	00	65	00	2E	00	65	U.p.d.a.t.e...e.	
0AC900	78	00	65	00	00	00	00	00	2F	00	6A	00	73	00	6F	x.e...../..j.s.o.	
0AC910	6E	00	00	00	69	00	70	00	69	00	6E	00	66	00	6F	n...i.p.i.n.f.o.	
0AC920	2E	00	69	00	6F	00	00	00	63	6F	75	6E	74	72	79	..i.o...country.	
0AC930	63	69	74	79	00	00	00	00	69	70	00	00	00	00	00	city....ip.....	
0AC940	00	00	00	00	5F	04	26	0E	06	07	08	82	68	83	A5	&.....h...	
0AC950	3A	4B	8A	6A	23	46	42	49	23	54	4F	4F	4C	4B	49	54	:K.j#FBI#TOOLKIT
0AC960	23	47	49	44	52	41	40	54	45	41	4D	00	40	2D	49	56	#GIDRA@TEAM.@-IV
0AC970	2D	46	42	49	2D	53	45	52	56	45	52	32	11	00	00	00	-FBI-SERVER2....
0AC980	00	00	00	00	00	00	00	80	FF	FF	FF	FF	FF	FF	FF	7F	

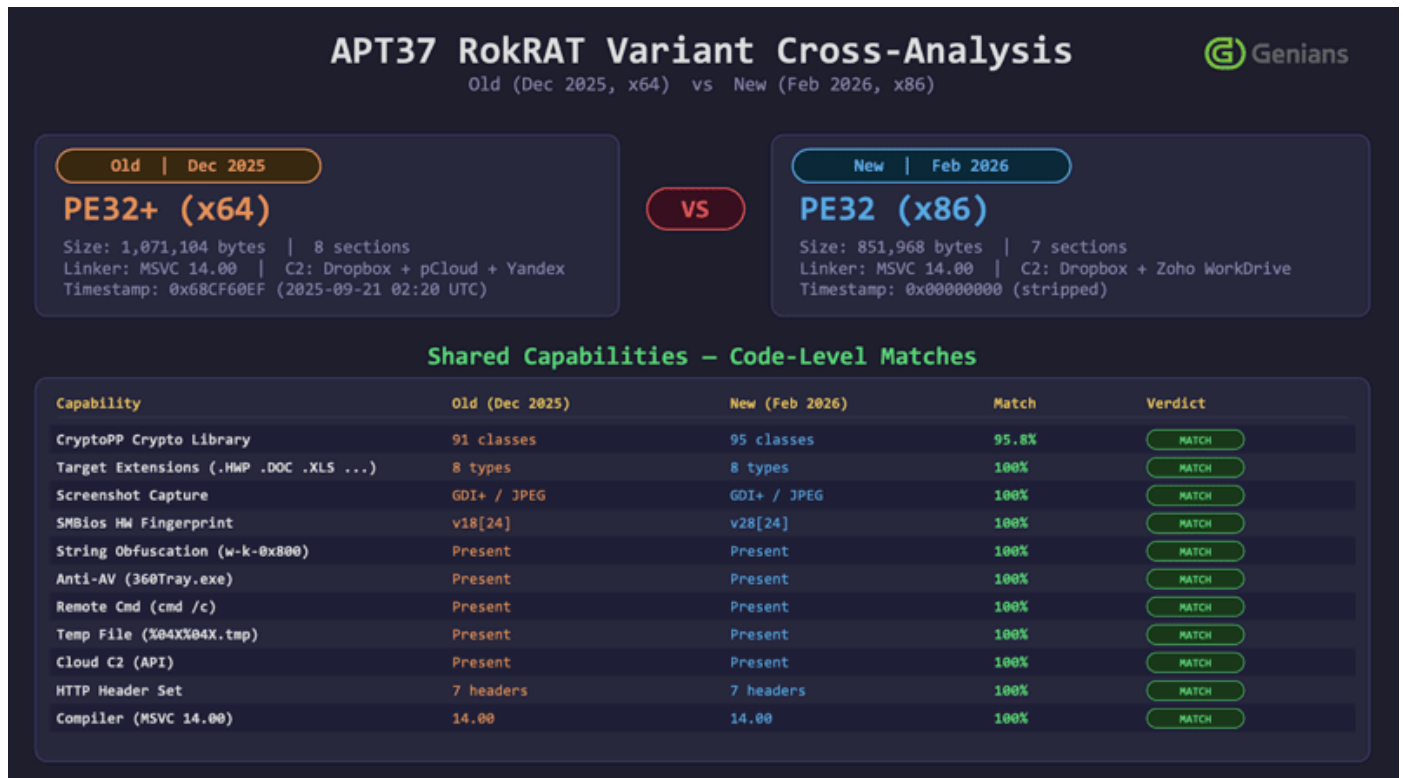
[Figure 4-6] Strings Embedded in the Binary

5. Threat Attribution

5-1. APT37 RokRAT Comparison

Genians Security Center (GSC) has continuously tracked APT37 campaign activity through a range of Cyber Threat Intelligence (CTI) analysis reports and has accumulated key characteristics and case examples of the group's major cyber operations.

These findings serve as an important reference not only for understanding individual incidents, but also for analyzing campaign-level activity observed over an extended period.



[Figure 5-1] Comparison with the RokRAT Variant Identified in December 2025

APT37's RokRAT is a backdoor focused on reconnaissance and information collection, with capabilities such as system information gathering, command execution, and screenshot capture. Its core functionality has remained relatively stable and has been reused repeatedly across multiple operations over time. A comparison with the variant identified in December last year shows a very high level of code similarity.

At the initial compromise stage, the group has consistently relied on LNK shortcut files and malicious HWP documents as primary delivery vectors. Through spear-phishing, it impersonates credible entities such as university professors, broadcast writers, and think tank events to gain the trust of its targets, then executes the final payload only in memory in a fileless manner.

In this campaign, however, the actor used a carefully tampered installer for a dedicated document viewer and actively leveraged Zoho WorkDrive cloud services.

In particular, its delivery and evasion techniques have become markedly more sophisticated over time. What began with PowerShell-based shellcode loading evolved into DLL sideloading through abuse of legitimate Sysinternals utilities, payload concealment through JPEG steganography, and multi-stage XOR decryption chains of up to three layers, making its method of concealing malicious activity behind legitimate documents and processes increasingly refined.

This shows that RokRAT has focused less on changing its core functionality and more on evolving its delivery, execution, and evasion chain.

Its C2 infrastructure has also continued to abuse the APIs of legitimate cloud services such as Dropbox, pCloud, Yandex Cloud, and Zoho WorkDrive to disguise malicious traffic as normal activity. In addition, evidence that the same account IDs were registered with different cloud services on the same date suggests that the actor systematically operated a coordinated multi-cloud infrastructure.

Accordingly, this activity is assessed to be attributable to the APT37 threat group based on sufficient technical evidence linking the campaign to the group.

5-2. Facebook and Zoho-Related Findings

The Facebook accounts used by the threat actor, "**richardmichael0828**" and "**johnsonsonphilia0414**," were both confirmed to have been created on November 10, 2025. The fact that both accounts were created on the same date suggests that they may have been operated by the same actor or organization.

From a network infrastructure perspective, the actor appears to rely primarily on Astrill VPN, with the U.S.-based IP address **38.32.68[.]195** repeatedly observed across related activity.

Although this IP has limited value as a standalone indicator because it falls within a VPN service range, past attack infrastructure associated with the same IP has been identified in analysis reports by Recorded Future and Silent Push, including "[PurpleBravo's Targeting of the IT Software Supply Chain](#)", "[Silent Push Pivots into New Lazarus Group Infrastructure, Acquires Sensitive Intel Related to \\$1.4B ByBit Hack and Past Attacks](#)." This supports the possibility that the infrastructure was reused in connection with a specific threat group.

In the initial access stage, the threat actor was observed approaching targets through Facebook accounts listing **Pyongsong and Pyongyang, North Korea, as their places of birth**. After establishing trust, the actor directed targets to external channels such as Telegram and email to deliver malicious files, following a typical social engineering-based attack flow.

Meanwhile, APT37 was previously observed using the account "**scott.snyder@zoho.com**" in a spear-phishing attack in 2017, and since then, multiple cases of Zoho services being abused have continued to emerge.

In particular, the account "**leon91729@zoho.com**" was additionally identified in the second half of 2025. During the same period, malware using Zoho WorkDrive as C2 infrastructure was discovered along with decoy documents written in the North Korean typeface "**Chollima**", and evidence was also found that the account "**kingtiger1970**" had been used during registration.

In addition, the debugging string "**JinHyok**" was identified during C2 communication. As an identifying element observed in North Korea-linked threat groups, it may serve as an important clue for attribution.

6. Conclusion

This threat was identified as a multi-stage social engineering attack in which the actor first built trust with targets through Facebook-based reconnaissance, then moved the conversation to external messaging platforms such as Telegram to induce them to execute a malicious file.

The threat actor lowered the target's guard by claiming that a dedicated program was required to view encrypted military documents, then induced the target to run a tampered Wondershare PDFelement installer disguised as legitimate software. This executable carried out the initial compromise through embedded shellcode and was designed to execute additional payloads in stages.

In particular, the abuse of the Seoul branch website of a Japanese real estate information service as C2 infrastructure, along with the delivery of follow-on commands through a payload disguised as a JPG image file, was identified as an advanced concealment technique that makes the activity difficult to distinguish from normal traffic.

In addition, this campaign follows a multi-stage attack chain that begins with social network-based initial access, moves to pretexting-based inducement to execute malware, proceeds through shellcode-based initial compromise, and culminates in follow-on command execution using an image-disguised payload. This attack chain closely resembles APT37's established tactics. In particular, when considered together, the North Korean linguistic and regional characteristics, attack flow, and infrastructure operation methods suggest that the campaign was likely carried out by the same cluster of threat actors or a closely associated group.

Given these characteristics, responding to this threat requires not only action based on the identified IoCs, but also behavior-based analysis across the full attack chain, including SNS-based contact and relationship building, execution of executables disguised as legitimate software, shellcode execution and follow-on payload loading, C2 communication routed through legitimate websites, and execution of concealed payloads disguised as image files.

Accordingly, organizations should establish an EDR-centered response framework capable of detecting the threat actor's tactics, techniques, and procedures (TTPs) in an integrated manner, from initial compromise through execution, persistence, C2 communication, and follow-on command activity. In particular, to respond to advanced evasion techniques such as masquerading as legitimate processes and abusing legitimate infrastructure, it is essential to complement single-event detection with threat hunting and correlation-based analysis across related behaviors.

6-1. Countermeasures for Social Network-Based Initial Access

Because this threat relies on establishing contact through social networking platforms such as Facebook before initiating the attack, traditional email-focused security policies alone are not sufficient to effectively prevent initial compromise.

In particular, the actor used social engineering techniques that approached targets with specific themes such as North Korea, military affairs, technical cooperation, and high-profit deals, built trust, and then induced them to execute malicious files. This indicates that user behavior can directly affect an organization's security posture.

Organizations should therefore establish security guidelines for non-business contact and file sharing through social networking and messaging platforms, define clear source verification procedures for externally received files, and implement policies that restrict the execution of files unrelated to work.

In addition, **organizations need to strengthen user awareness training on key techniques used in this attack**, including encrypted archives delivered with passwords shared separately and external requests to install legitimate-looking software.

Ultimately, this threat relies on abusing user trust rather than exploiting a technical vulnerability, making a user awareness-based response framework essential beyond simple malware detection.

In an environment where attacks increasingly abuse elements that appear legitimate, user judgment and security awareness become critical lines of defense that directly affect an organization's overall security posture.

6-2. Countermeasures for Detecting the Execution of Tampered Legitimate Software

In this attack, a tampered Wondershare PDFelement installer disguised as legitimate software was used.

This is a common technique that can evade simple hash-based detection and reputation-based blocking.

Accordingly, the following threat behaviors should be prioritized in hunting based on event logs recorded by EDR:

- Abnormal process creation immediately after the execution of legitimate software
- Invocation of script interpreters such as PowerShell or CMD after the installer is executed
- External network communication unrelated to the user's execution context
- Process tree structures that deviate from a normal installation flow

In particular, attacks using tampered installers can be effectively identified by analyzing detection logs based on behavior chains in which a legitimate program is followed by abnormal activity.

6-3. Threat Hunting and Continuous Monitoring Framework

Because advanced threat groups such as APT37 are designed to remain hidden and persist over long periods, response measures centered on one-time detection have inherent limitations.

Organizations should therefore operate a proactive detection framework based on threat hunting and use continuous monitoring to identify signs of potential compromise.

In particular, hidden threats should be identified through comprehensive analysis of abnormal execution activity following SNS-based contact, behavioral changes after the execution of legitimate software, recurring process patterns over time, and records of external communications.

In addition to retrospective analysis based on historical log data, organizations should also maintain a framework that supports tracing the initial point of compromise through long-term behavioral tracking.

Such a threat hunting and continuous monitoring framework plays an important role in minimizing detection gaps against advanced APT attacks.

6-4. Countermeasures for C2 Communication and Image-Disguised Payloads

The threat actor attempted to evade detection by using C2 communication routed through a legitimate website and payloads disguised as JPG files. This is a common technique that makes network-based detection more difficult.

This calls for the following detection strategies:

- Identify abnormal data transfer patterns following access to legitimate websites
- Analyze unusual response sizes or structures associated with image file requests
- Detect periodic external communications unrelated to user activity
- Identify abnormal encoding or hidden data patterns in network traffic

In particular, network behavior-based analysis capable of identifying abnormal activity within legitimate services, rather than simply blocking domains, is critical. It is also important to check for attempts to access Zoho WorkDrive.

6-5. Integrated Response Strategy with "Genian Insights E"

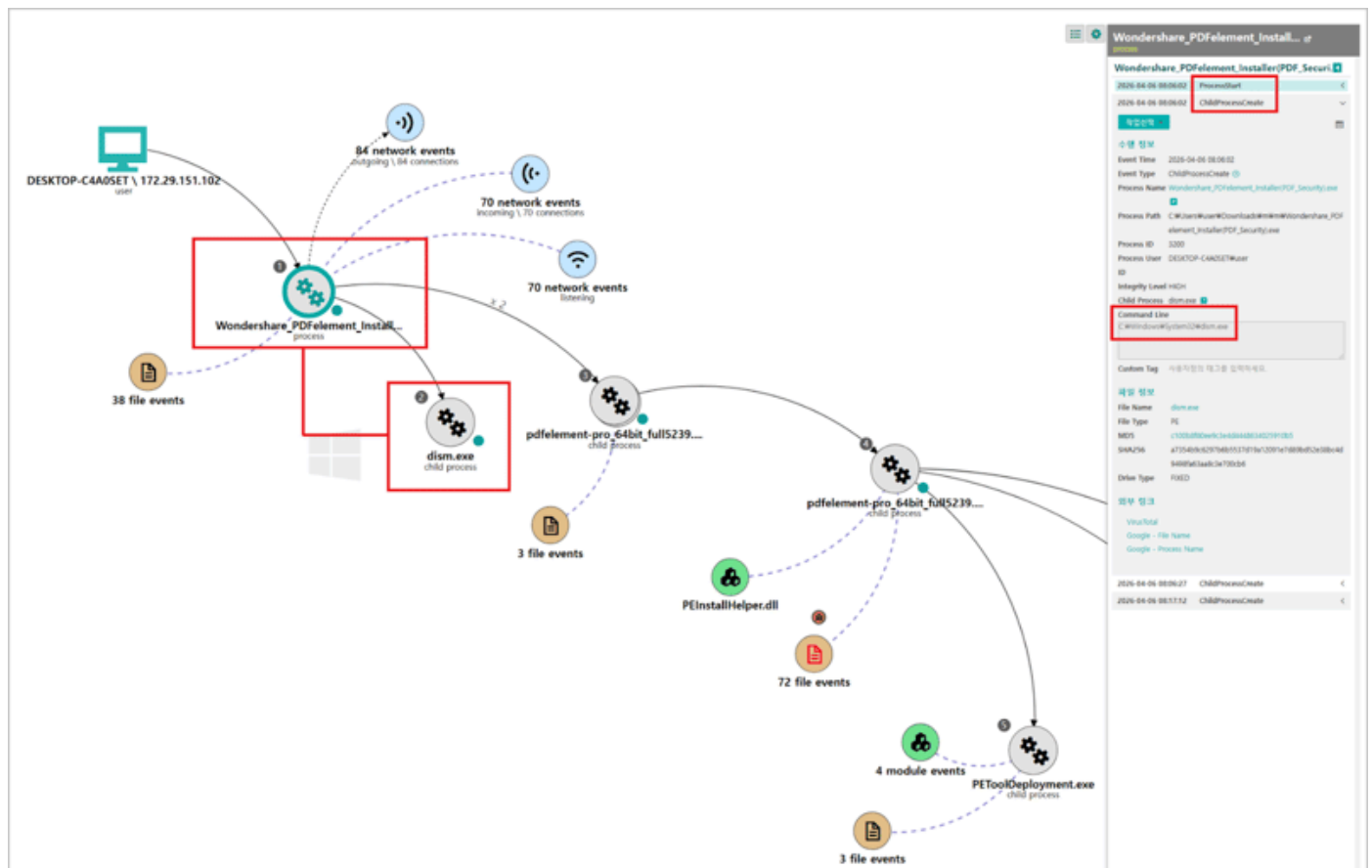
This threat has been identified as a complex attack that combines social engineering, masquerading as legitimate software, memory-based execution, and abuse of legitimate infrastructure, making it difficult to respond effectively with a single security solution. An integrated response framework centered on EDR is therefore needed to provide visibility across the full attack chain, from initial execution to follow-on activity, and to support process tree-based anomaly detection and correlation analysis across related events.

["Genian Insights E"](#) is an integrated endpoint security platform built on a single-agent architecture to address a wide range of threats.

It provides **EDR** (Endpoint Detection and Response) for real-time behavior-based detection and threat response, **Anti-Virus** for signature-based malware detection and remediation, and **Anti-Ransom** capabilities that block file encryption activity in real time while supporting automatic backup and restoration of important documents. It also includes **Device Control** capabilities that help prevent internal data leakage by controlling the use of storage media such as USB drives and external hard disks.

In particular, it is important to go beyond analyzing files, processes, and network events separately and instead analyze the entire attack flow as a single, connected context.

This allows suspected endpoints to be quickly isolated and helps prevent further spread. By combining IoC-based detection with behavior-based detection, organizations can also improve detection accuracy. As a result, EDR can serve as a core security control for enabling early detection and proactive response to multi-stage APT threats such as this one.



[Figure 6-1] Identifying the Initial Infection Flow with EDR

Using the Attack Storyline feature in Genian Insights E, it is possible to observe that when the process "Wondershare_PDFelement_Installer(PDF_Security).exe" starts, "dism.exe" is launched as a child process.

28d0143718153bf04c1919a26bb70c2d

36be2cbb59cd1c3f745d5f80f9aee21c

C2

japanroom[.]com

38.32.68[.]195

222.122.49[.]15