Analysis of the Triple Combo Threat of the Kimsuky Group

By Genians

Published: 2025-06-09 · Archived: 2026-04-05 21:45:21 UTC

◈ Executive Summary

- Deployed a covert infiltration strategy using a three-stage communication channel: Facebook, email, and Telegram
- Lured targets with seemingly credible content related to North Korean defector volunteer activities to initiate conversations and deliver malicious files
- Confirmed linkage to the state-sponsored hacking group 'Kimsuky,' which targets defense and North Korea-related activists
- Utilized Korea-specific compressed file formats and encoded malicious scripts, specifically designed to evade security detection patterns
- EDR-based threat hunting and triage can provide visibility

1. Overview

○ The Genians Security Center (GSC) detected an APT (Advanced Persistent Threat) campaign targeting users of Facebook, email, and Telegram in Korea between March and April 2025.

○ The threat actor explored reconnaissance and selected attack targets through two Facebook accounts.

○ According to a joint investigation conducted by Genians threat analysts, the campaign was attributed to the Kimsuky group, a well-known North Korea-affiliated state-sponsored hacking organization. The incident was identified as part of the 'AppleSeed' campaign.

○ Notably, 'AppleSeed' was first introduced during two VB Conferences in October 2019 and 2021 by lead researcher Jae-Ki Kim and colleagues in the sessions titled “Kimsuky group: tracking the king of the spear-phishing” and “Operation Newton: Hi Kimsuky? Did an Apple(seed) really fall on Newton's head?”

○ According to the disclosed presentation materials, this string was found in the PDB (Program Database) path of malicious files developed by the Kimsuky group.

○ Additionally, in November 2021, AhnLab ASEC provided an in-depth analysis of AppleSeed in its report titled “Operation Light Shell,” which documented another Kimsuky attack case.

2. Background

○ Threat activity by the Kimsuky group remains high in Korea. The group is known to use three major tools in their attacks, often under different aliases depending on the variant:

- AppleSeed
- BabyShark (RandomQuery)
- FlowerPower (GoldDragon)

○ Historical examples of AppleSeed often involved executable file extensions (e.g., EXE, PIF). Script-based files (particularly JSE, WSF, and JS) were frequently used, often invoking malicious DLL libraries with Base64-encoded contents.

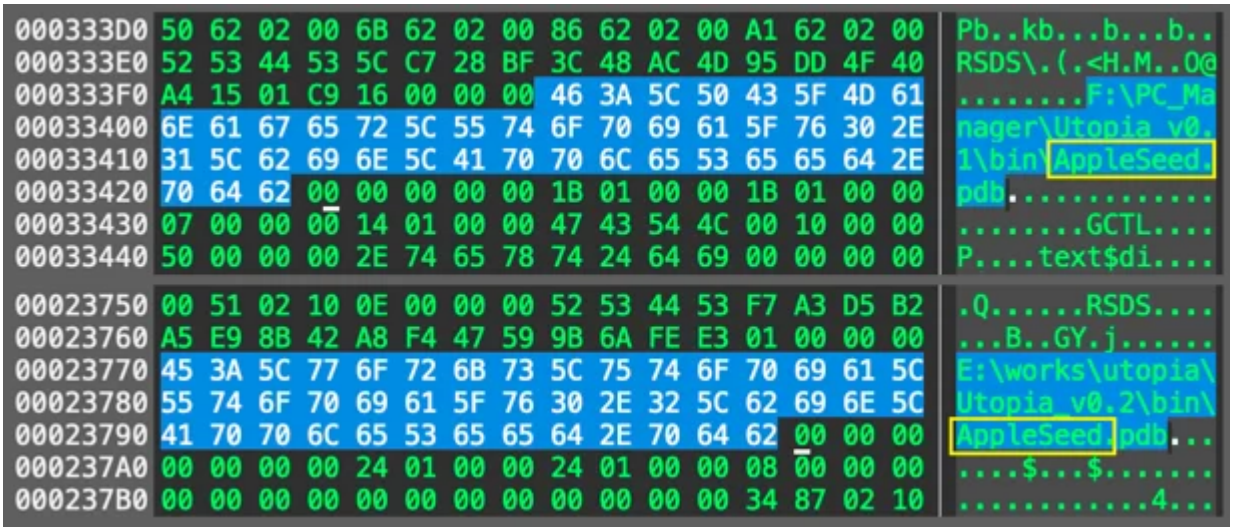
○ Spear phishing attachments frequently used the EGG ALZIP format. Threat actors sometimes recommended using specific decompression tools via email. This serves the dual purpose of evading detection by signature-based security products and encouraging execution on a PC environment rather than a smartphone.

○ Examples of PDB paths containing the 'AppleSeed' string:

No	Bit	PDB Path
1	32	F:\PC_Manager\Utopia_v0.1\bin\AppleSeed.pdb
	64	F:\PC_Manager\Utopia_v0.1\bin\AppleSeed64.pdb
2	32	E:\works\utopia\Utopia_v0.2\bin\AppleSeed.pdb
	64	E:\works\utopia\Utopia_v0.2\bin\AppleSeed64.pdb

[Table 1] PDB Path Information of AppleSeed Malware Files

○ The AppleSeed case under the 'Utopia_v0.1' path was created in May 2019 based on the DLL build date. The 'Utopia_v0.2' version was built between August 2019 and January 2020.



[Figure 2-1] PDB Path of AppleSeed

○ The past activities of this threat actor indicate that targets have primarily included the defense industry and military sectors. During the COVID-19 pandemic, they also launched attacks against vaccine manufacturers. In

addition, there have been continuous attempts to steal information from cryptocurrency exchanges and activists involved in North Korea-related issues.

○ Genians threat analysts discovered a recent AppleSeed attack attempt that persisted for more than two months starting in March 2025 and conducted an in-depth investigation.

○ This report analyzes the most recent AppleSeed attack case, in which the following three access channels were used. The goal is to provide insights and preventative measures against similar security threats through detailed analysis.

- Facebook
- E-Mail
- Telegram

3-1. Facebook-Based Attack Case

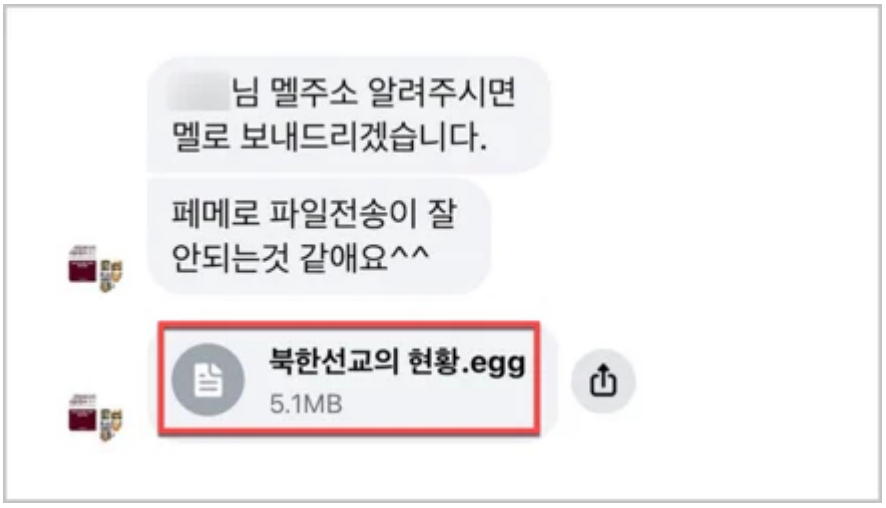
○ The first case involves an attack launched via Facebook. The threat actor used an account named 'Transitional Justice Mission' to send friend requests and direct messages to multiple individuals involved in North Korea-related activities.

[Figure 3-1] Initial contact attempt via Facebook Messenger

○ The actor introduced themselves as either a missionary or a church-affiliated researcher, skillfully approaching the target through Facebook Messenger.

○ Then, by posing as if they were sharing a specific document, they caught the target’s interest and delivered a malicious file.

○ The malicious file was delivered as a password-protected EGG archive.



[Figure 3-2] Malicious File Delivered via Facebook Messenger

○ The attacker also hijacked another Facebook account for their operation. According to the profile data, the account owner claimed to be a graduate of the Korea Air Force Academy.

○ At the time of the malicious activity, the Facebook profile displayed a photo of a Korean man, which was removed after some time.

[Figure 3-3] Message Posing as Inquiry into Defector Volunteer Activities

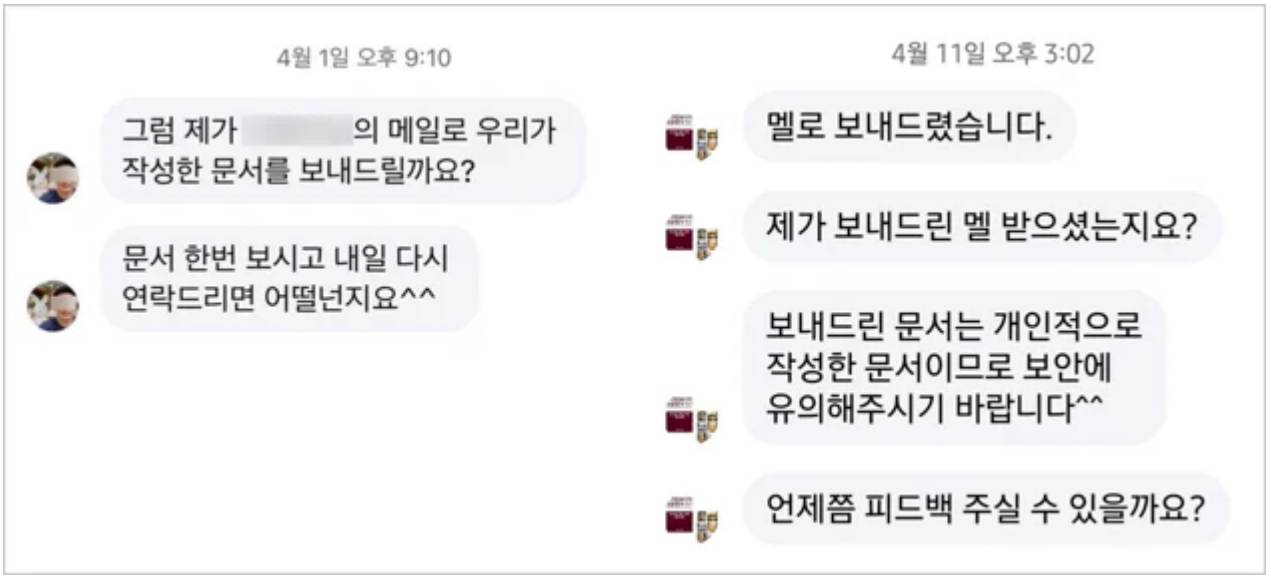
○ In this case, the threat actor approached the victim by pretending to inquire about volunteering for North Korean defectors. The file was sent either directly via Messenger or through follow-up conversations using alternate

delivery methods.

3-2. Email-Based Attack

○ The threat actor also attempted further contact by using the email address obtained through Facebook Messenger conversations.

○ They asked for the target’s email address directly via direct messages, then used it to lure the target into opening a malicious file.



[Figure 3-4] Email Access Attempt via Facebook Messenger

○ Both Facebook accounts mentioned earlier approached the targets in similar ways. Although different accounts were used, the tactics and activity patterns strongly suggest they were operated by the same individual.

○ The malicious files used in the attacks were also structurally identical, and the shared theme of 'volunteer support for North Korean defectors' was consistently used to deceive the recipients.

○ The Korean text in the messages includes informal abbreviations and occasional spelling errors, suggesting that the contents was not generated by AI or translation tools.

○ Based on linguistic analysis, the threat actor is likely a native or highly fluent Korean speaker.

[Figure 3-5] Malicious File Delivered via Email

○ The spear-phishing email contained large attachments or embedded URLs intended to lure the recipient into downloading a file.

○ The files were compressed in the EGG format, and the recipient was instructed to use a specific decompression tool, typically available on PC.

○ This tactic appears intended to prevent access from mobile devices, as the malware is designed to run in a Windows environment.

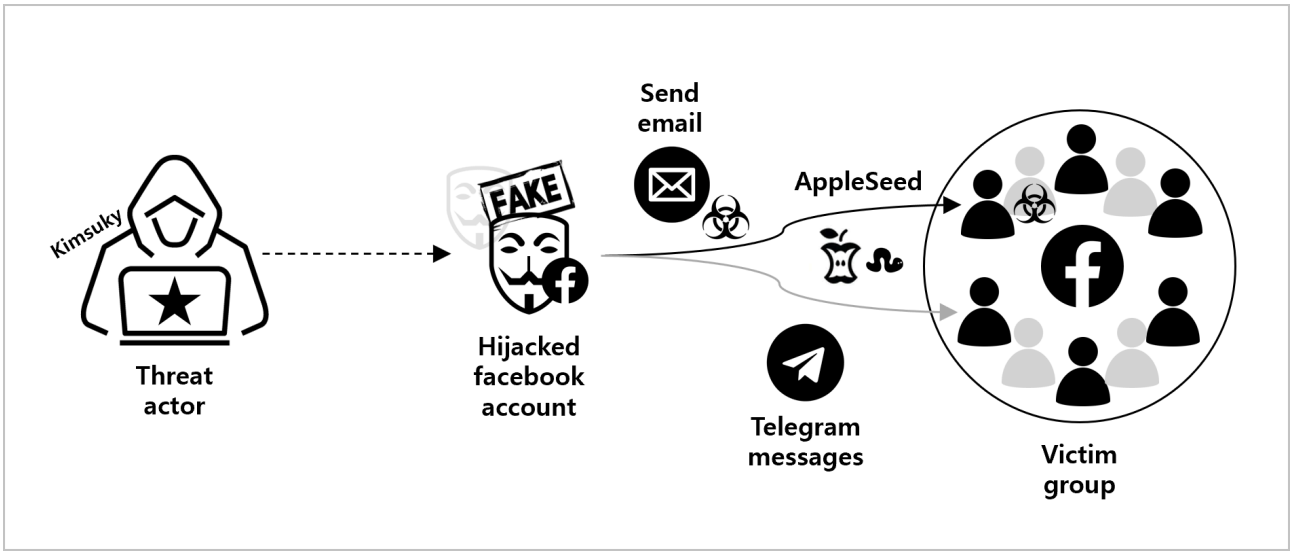
3-3. Telegram-Based Approach

○ The malicious files used in this attack were also structurally identical, consistently using the theme of 'volunteer support for North Korean defectors' to deceive the targets.

[Figure 3-6] Multi-Stage Approach Comparison

○ Analysis of the targeted attack revealed that the threat actor initially made contact via Facebook and email.

○ If the attacker obtained the target's mobile number, they proceeded to contact them through Telegram. Other messaging apps may also have been used. This demonstrates the actor's active and persistent tactics, highlighting the growing variety in defector-themed attacks.



[Figure 3-7] Attack Flow Diagram

○ Based on the observed attack flow, it appears that a specific individual's device was initially compromised. The attacker then monitored the victim and extracted their credentials for SNS and email accounts.

○ With hijacked Facebook access, the attacker impersonated the legitimate owner. Because the Facebook account may have existed for a long time, it draws little suspicion from the victim's contacts. Threats that exploit online friend relationships are difficult to detect from outside. Due to the discreet nature of 1:1 chats over messenger, such threats are difficult to detect and require extra caution.

○ Users should always be wary of unexpected URLs or files, as these may contain threats. Maintaining a habit of vigilance is key to cybersecurity.

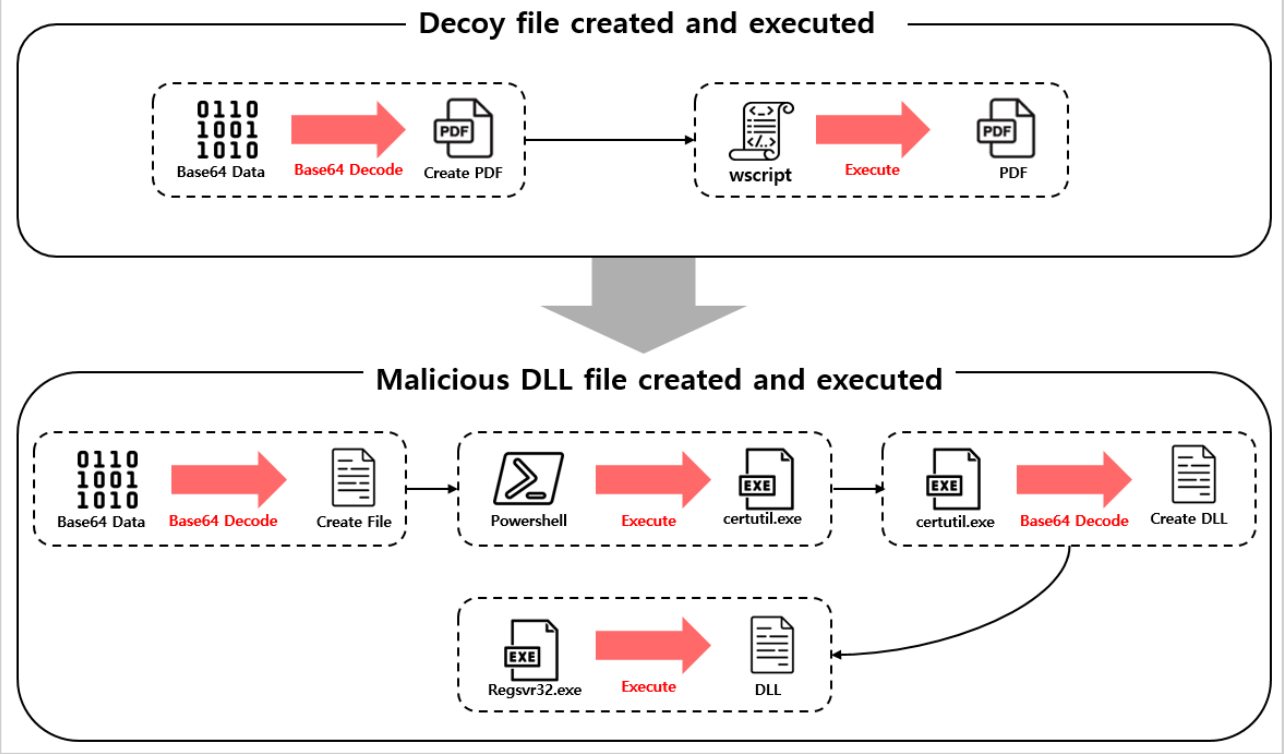
○ This case shows how attackers leverage multiple platforms—Facebook, email, and Telegram—to carry out coordinated multi-channel attacks.

4. Malware Analysis

4-1. Analysis of '탈북민지원봉사활동.jse' File *(Defector Volunteer Support.jse)*

○ The JSE file has a .jse extension and is an obfuscated JScript file that runs under Microsoft's Windows Script Host (WSH).

○ The file named '탈북민지원봉사활동.jse' creates two files upon execution: one is a legitimate-looking PDF document used as a decoy to trick the user, and the other is a malicious DLL file that carries out the actual malicious behavior.



[Figure 4-1] Execution Flow of '탈북민지원봉사활동.jse' *(Defector Volunteer Support.jse)*

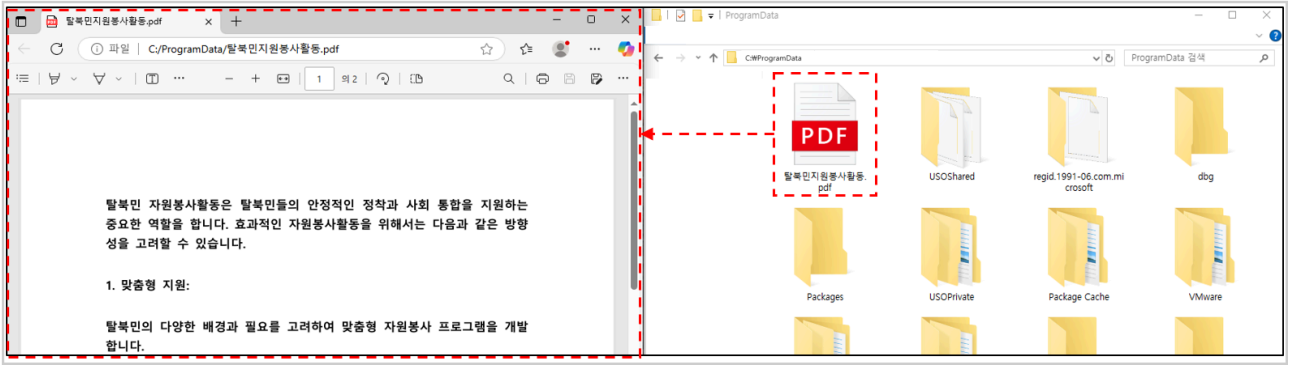
○ Inside the script, the variable xF6hKgM2MlR contains the Base64-encoded data for the PDF file, while the variable guC1USOkKiW holds the name of the file to be created: '탈북민지원봉사활동.pdf'*(Defector Volunteer Support)*.

○ Using the Microsoft.XMLDOM object (xmlDom), the value of xF6hKgM2MlR is decoded and saved as a file at 'C:\ProgramData\탈북민지원봉사활동.pdf'*(Defector Volunteer Support)*, which is then automatically opened using WScript.Shell.

○ This decoy document makes the user believe they are viewing a legitimate file, effectively concealing the malicious behavior.

[Figure 4-2] Decoy File Execution Process

○ When the script is executed, a PDF file is created and opened as shown below.



[Figure 4-3] PDF File Creation and Execution

○ The DLL file's data is Base64 encoded twice. The first decoding is performed using the Microsoft.XMLDOM object (xmlDom), followed by the execution of certutil through PowerShell, completing the two-step decoding process.

○ Once the decoding is complete, the malicious DLL file is saved with the name C:\ProgramData\vmZMXSx.eNwm.

○ The DLL file is executed in silent mode using the command regsvr32.exe /s /n /i:tgvyh!@#12 vmZMXSx.eNwm. This process loads the malicious DLL into the system, where it begins performing its malicious actions.

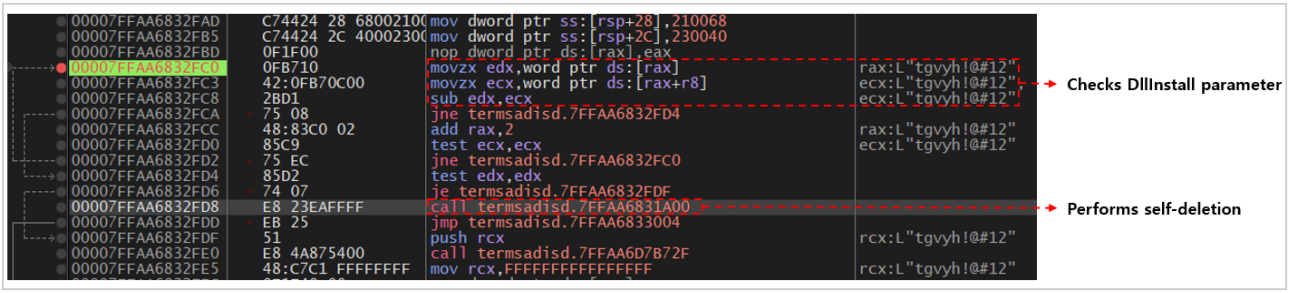
[Figure 4-4] Creation and Execution of Malicious DLL

4-2. Analysis of the vmZMXSx.eNwm File

○ The 'vmZMXSx.eNwm' is a VMProtect-packed DLL. VMProtect is a tool that virtualizes parts of the code, making it difficult to analyze the internal logic with standard debugging and analysis tools. It is commonly used to prevent reverse engineering. The key malicious functionality of the DLL is hidden within the virtualized sections, which limits static analysis.

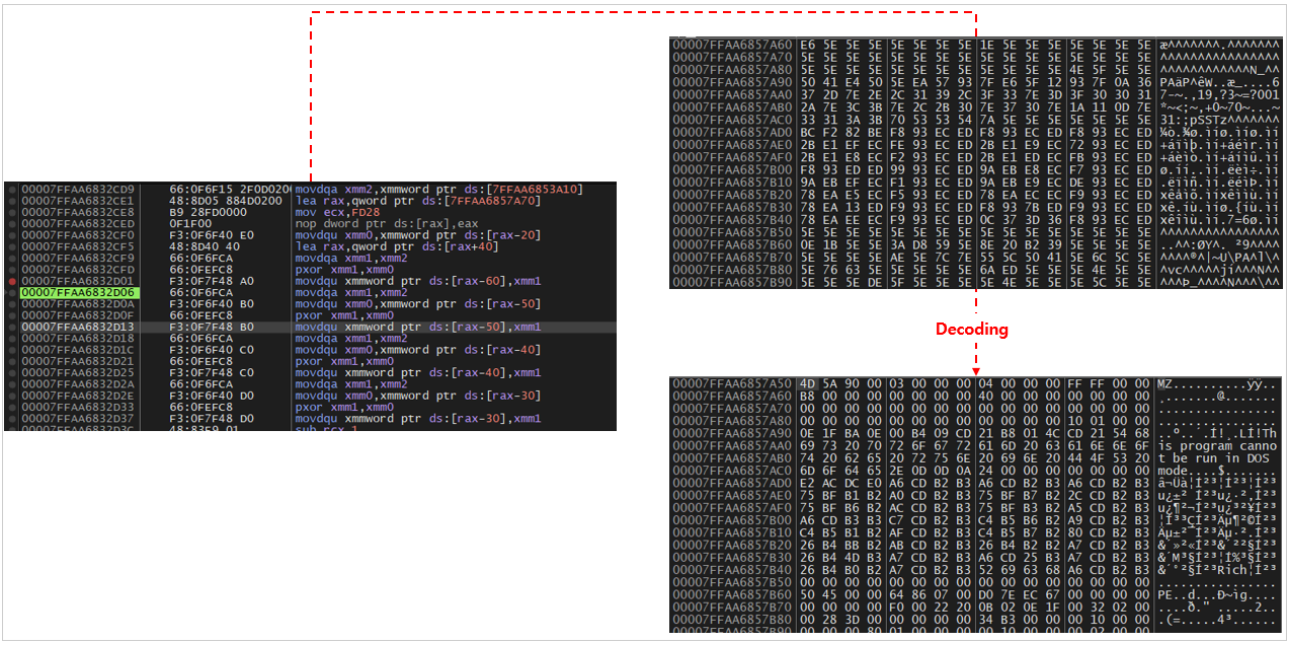
○ When executed with the command 'regsvr32.exe /s /n /i:tgvyh!@#12 vmZMXSx.eNwm', the DllInstall function of the 'vmZMXSx.eNwm' file is called, and the parameter 'tgvyh!@#12' is passed.

○ Once the 'vmZMXSx.eNwm' file is loaded into the 'regsvr32.exe' process, the passed parameter is checked against the string 'tgvyh!@#12'. If the values differ, a batch file is created to perform self-deletion.



[Figure 4-5] Parameter Verification

○ After the parameter verification, the decoding process is performed based on the value located at offset 0xA0 in the '.data' section. This decoding is carried out using an XOR method with a key value of 0x5E. Once decoding is complete, the original DLL binary data, which is not protected by VMProtect, is stored at the same offset in the .data section.

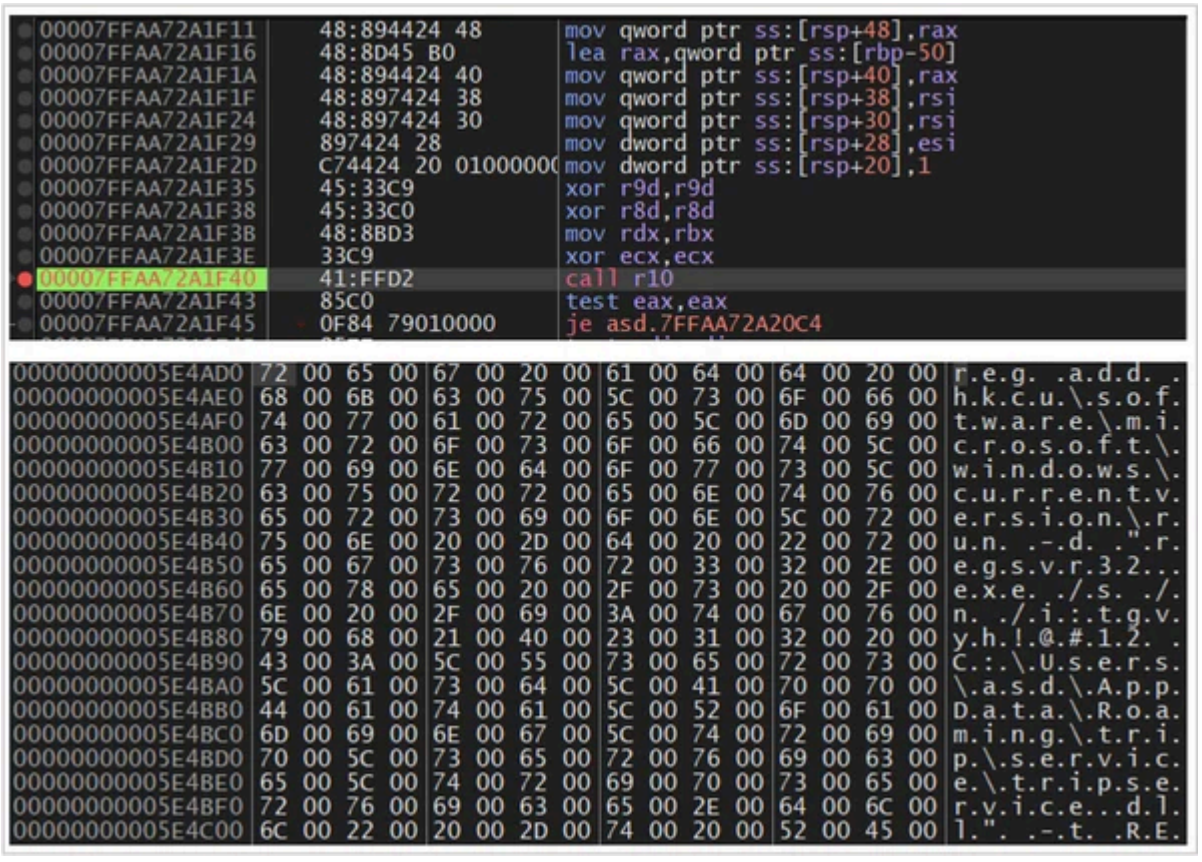


[Figure 4-6] DLL Decoding Process

○ The decoded DLL data is dynamically allocated in virtual memory and relocated. The sections of the DLL are manually organized in memory, and then the 'DllInstall' function of the DLL is called.

[Figure 4-7] DLL Relocation and DllInstall Function Call

○ After the 'DllInstall' function is executed, the same parameter verification process is performed as before. Then, the 'CreateProcessW' function is used to execute additional commands.



[Figure 4-8] Persistence Execution

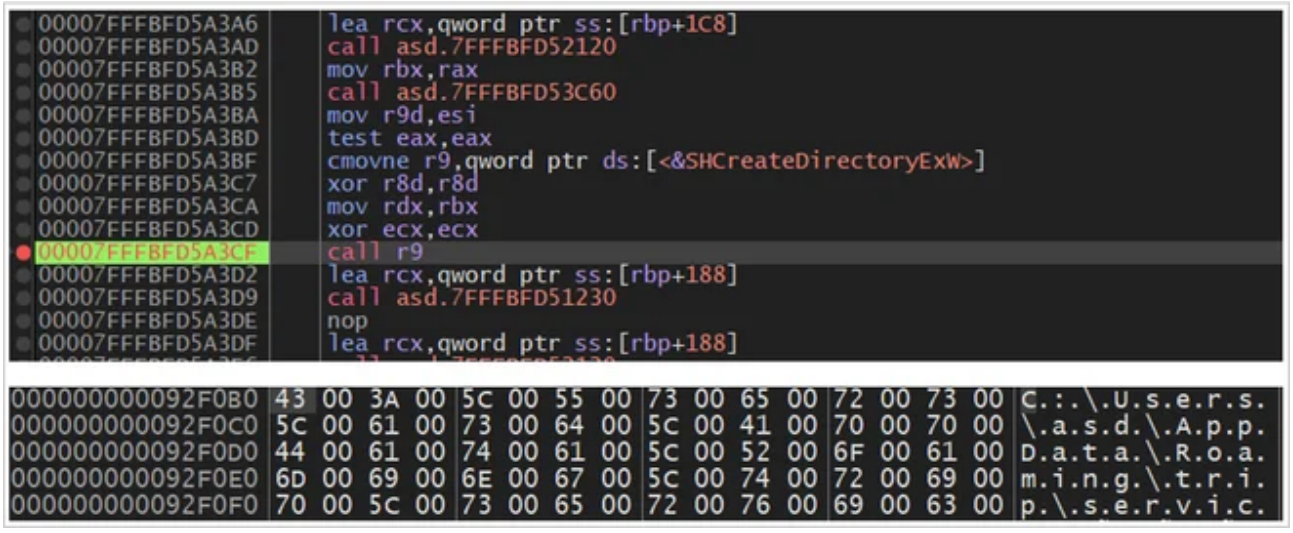
○ The command passed as an argument to 'CreateProcessW' registers the 'TripServiceUpdate' entry in the user execution registry (HKCU\...\Run) and configures the system to automatically execute the malicious DLL through

'regsvr32.exe' every time the system reboots.

reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /t REG_SZ /v "TripServiceUpdate" /d "regsvr32.exe /s /n /i:tgvyh!@#12 C:\Users\[UserName]\AppData\Roaming\trip\service\tripservice.dll" /f

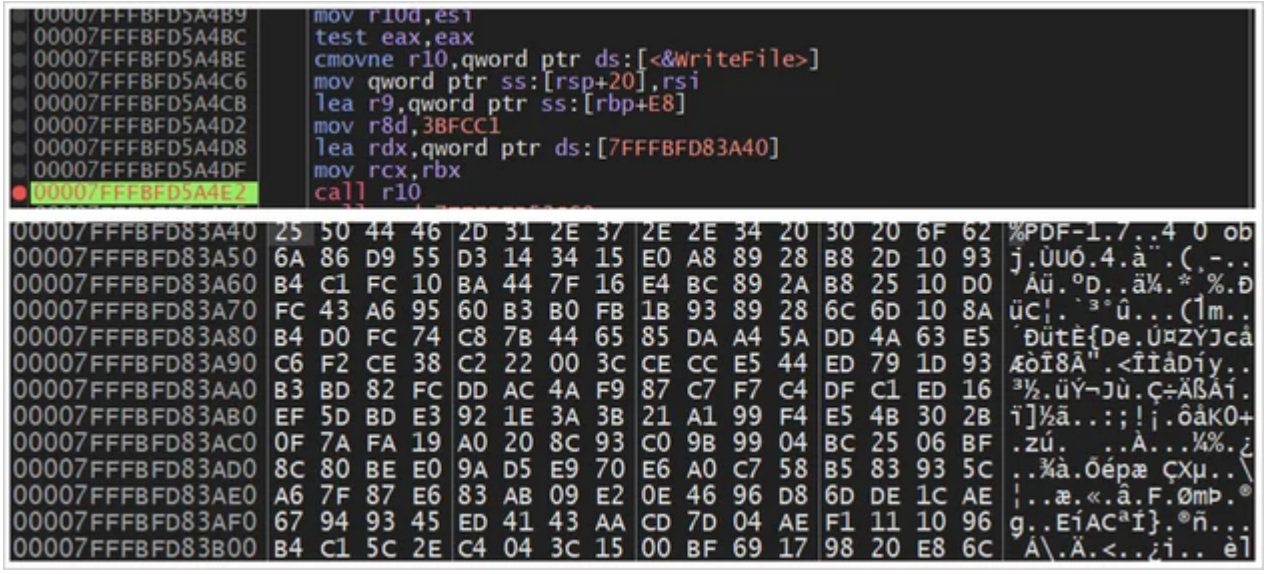
[Table 4-1] Persistence Execution Command

○ Subsequently, a directory is created at ‘C:\Users\[Username]\AppData\Roaming\trip\service\’ to store the malicious DLL (tripservice.dll). This path is referenced by the previously registered auto-execution registry entry (HKCU\...\Run).



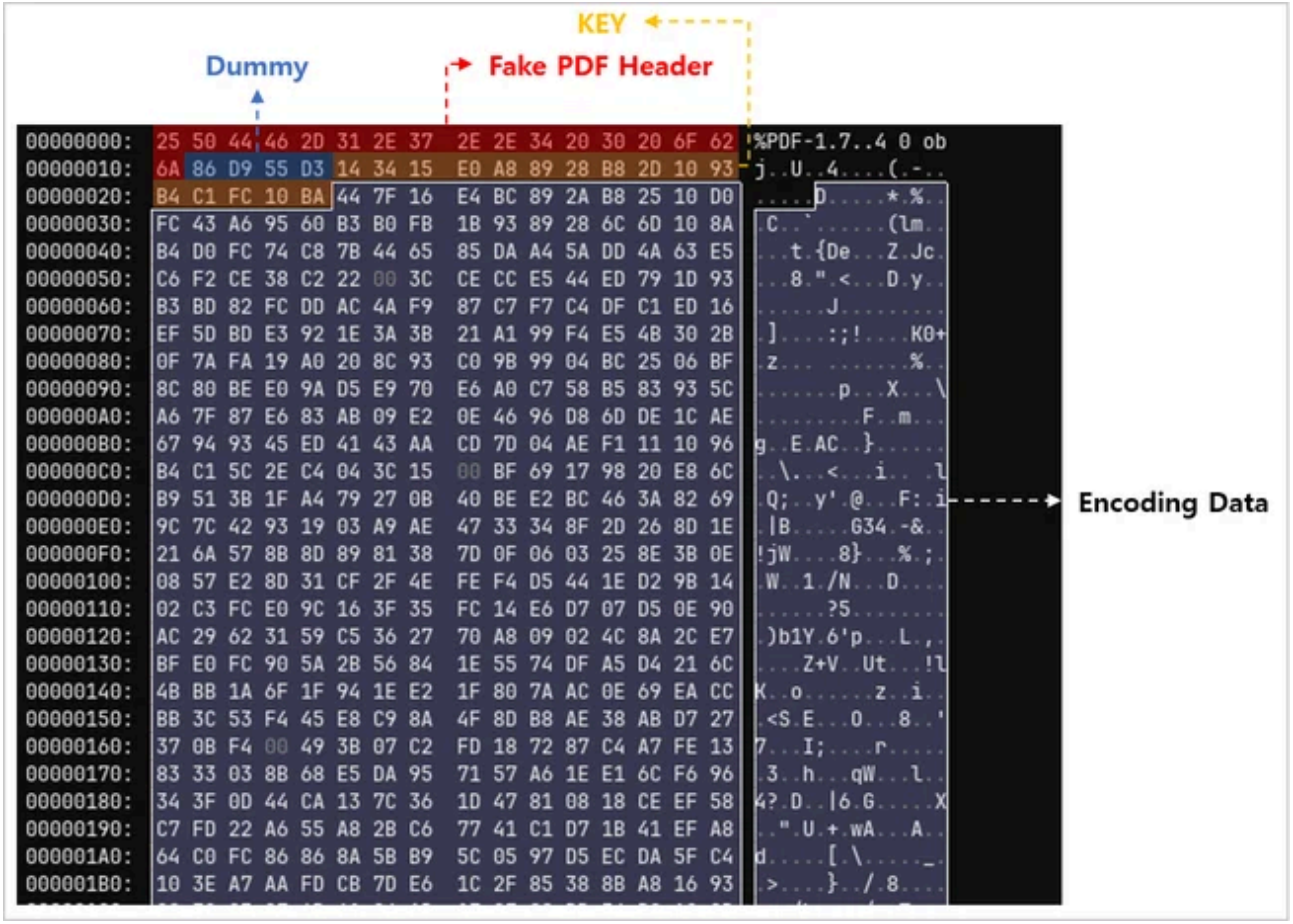
[Figure 4-9] Persistence DLL Directory Creation

○ A temporary file is created at C:\Users\[Username]\AppData\Roaming\temp\{random}.tmp, and the data from the .data section of the malicious code is directly stored in this file.



[Figure 4-10] Random tmp File Creation

○ The stored file is structured as follows: the first 17 bytes are a string designed to disguise the file as a legitimate PDF, followed by 4 bytes of dummy values that are not used in decoding. The next 16 bytes are used as a decoding key, and the remaining area stores the encoded body data using the XOR method.



[Figure 4-11] Random tmp File Structure

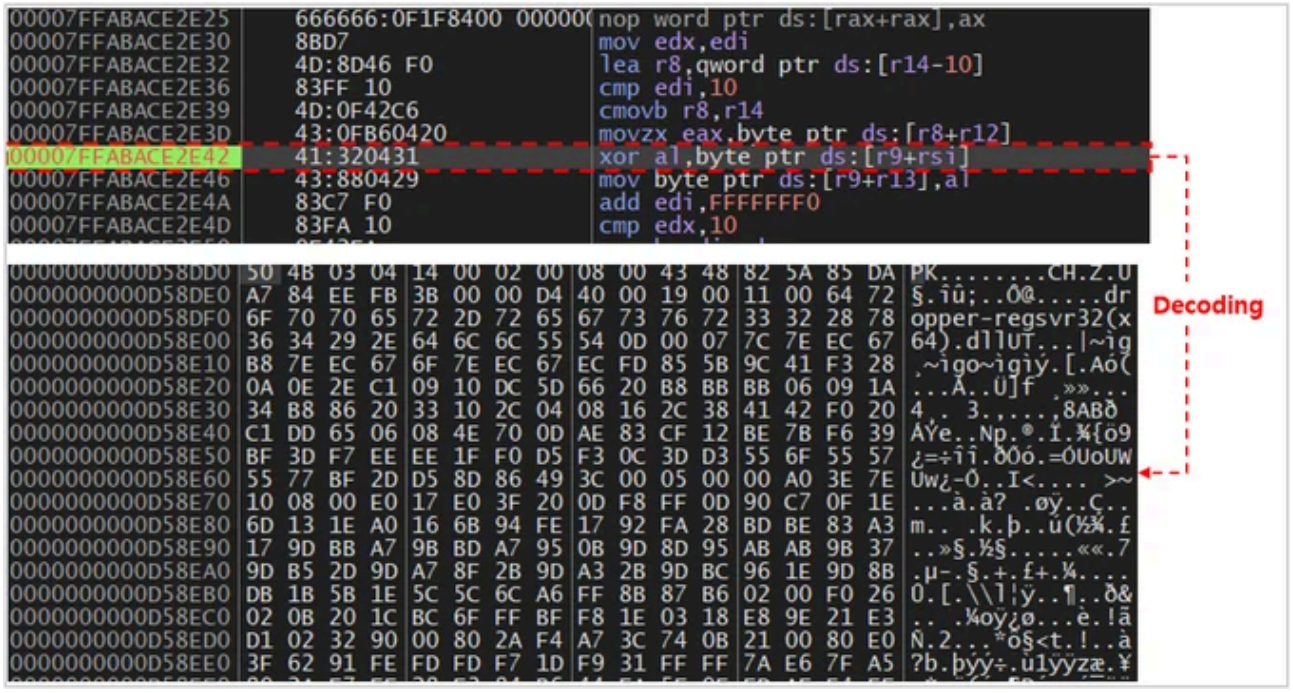
○ The malware retrieves the key value from the created '{random}.tmp' file and repeatedly performs XOR operations with 0x47E04B65.

```
00007FFABACE2D81    8BD7            mov edx,edi
00007FFABACE2D83    4C:6345 C0      movsxd r8,dword ptr ss:[rbp-40]
00007FFABACE2D87    4D:85C0         test r8,r8
00007FFABACE2D8A    7E 22           jle asd.7FFABACE2DAE
00007FFABACE2D8C    0F1F40 00       nop dword ptr ds:[rax],eax
00007FFABACE2D90    0FB64D D8       movzx ecx,byte ptr ss:[rbp-28]
00007FFABACE2D94    42:0FB60422     movzx eax,byte ptr ds:[rdx+r12]
00007FFABACE2D99    48:33C8         xor rcx,rax
00007FFABACE2D9C    C1EB 08         shr ebx,8
00007FFABACE2D9F    41:331C8F       xor ebx,dword ptr ds:[r15+rcx*4]
00007FFABACE2DA3    895D D8         mov dword ptr ss:[rbp-28],ebx
00007FFABACE2DA6    48:FFC2         inc rdx
00007FFABACE2DA9    49:3BD0         cmp rdx,r8
00007FFABACE2DAC    7C E2           jl asd.7FFABACE2D90
```

[Figure 4-12] Decoding Key Generation

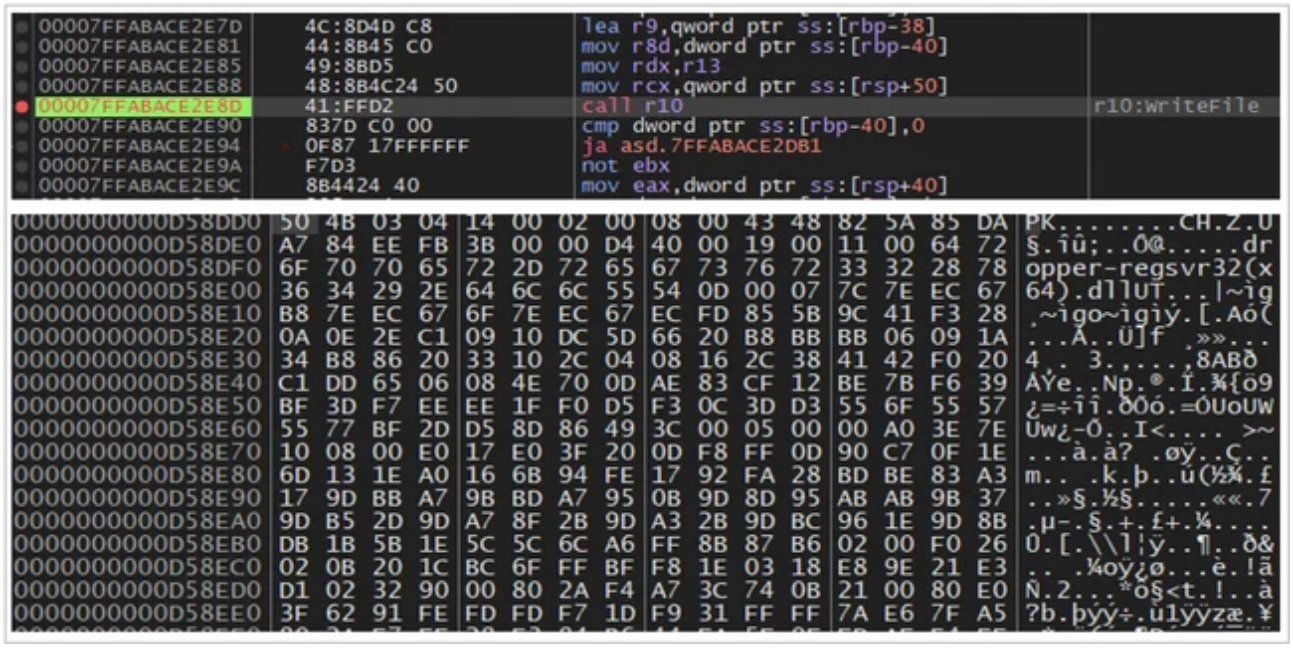
○ The encoded data in the '{random}.tmp' file is read in 4KB chunks, and XOR operations are performed using the previously set key to decode it.

○ The decoded result is ZIP file data.



[Figure 4-13] Data Decoding Process

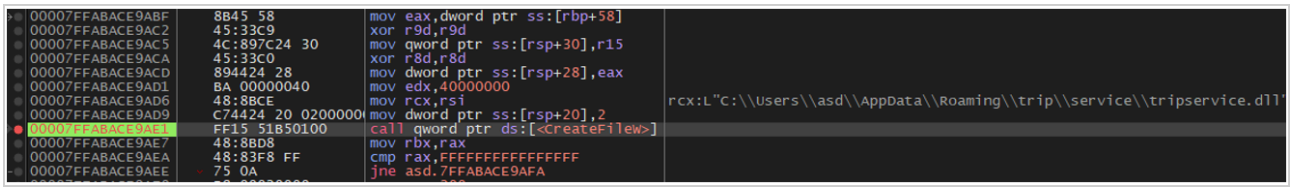
○ Once the decoding process is complete, the decoded data is saved as 'C:\Users\[Username]\AppData\Roaming\temp{random}.tmp.zip'.



[Figure 4-14] ZIP File Creation

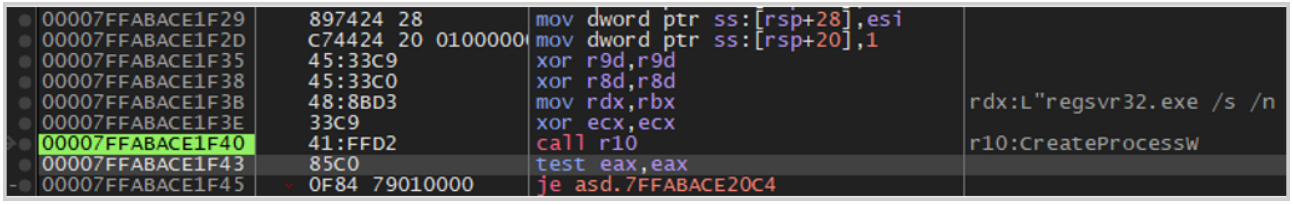
○ After the ZIP file is saved, the '{random}.tmp' file containing the encoded data is deleted.

○ Then, the stored '{random}.tmp.zip' file is extracted to create the file 'C:\Users[Username]\AppData\Roaming\trip\service\tripservice.dll'.



[Figure 4-15] Persistence DLL File Creation

○ Once the 'tripservice.dll' file is created, the command 'regsvr32.exe /s /n /i:tgvyh!@#12 C:\Users\[Username]\AppData\Roaming\trip\service\tripservice.dll' is executed through the 'CreateProcessW' function.



[Figure 4-16] Executing the 'tripservice.dll' File

○ Finally, a batch file is created to delete both the 'vmZMXSx.eNwm' file and the batch file itself.

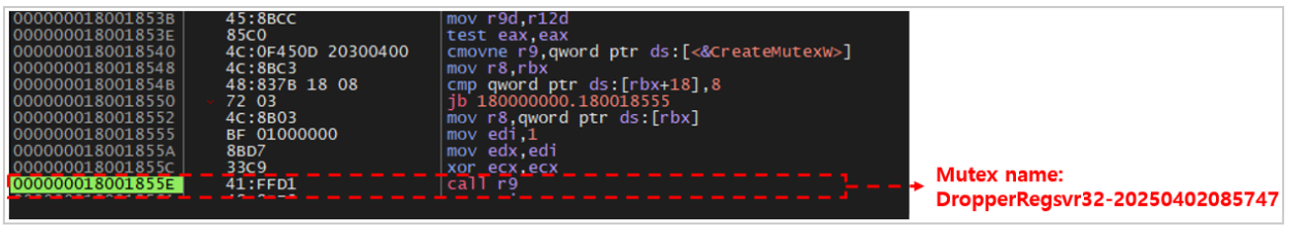
```
:repeat
del "C:\ProgramData\vmZMXSx.eNwm"
if exist "C:\ProgramData\vmZMXSx.eNwm" goto repeat
del "%~f0"
```

[Table 4-2] Batch File Content

4-3. Analysis of the tripservice.dll File

○ Once the 'tripservice.dll' file is loaded by the 'regsvr32.exe' process, the encrypted data stored in the '.data' section is decoded and dynamically allocated in memory. This process is similar to the one used by the 'vmZMXSx.eNwm' file. The code in this memory section then executes the 'DllInstall' function.

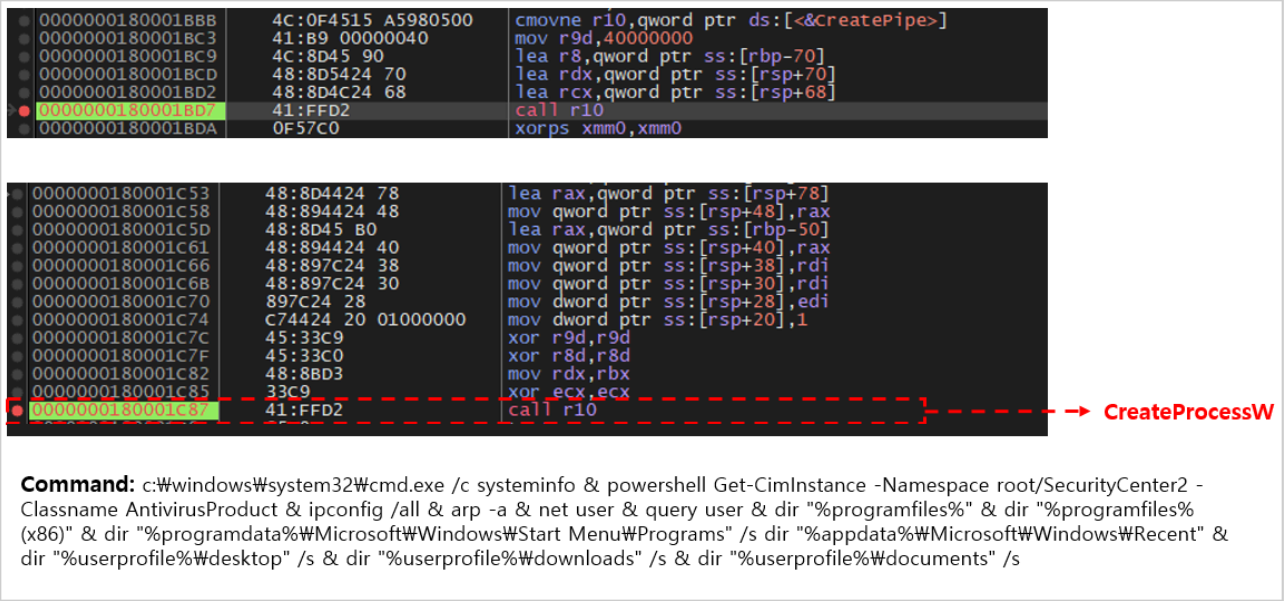
○ When the 'DllInstall' function is executed, a mutex named 'DropperRegsvr32' is created to prevent duplicate instances.



[Figure 4-17] Mutex Creation

○ The code first calls the 'CreatePipe' function to create a pipe and then executes 'CreateProcessW' to launch the command prompt (cmd.exe) and run commands that collect various system information.

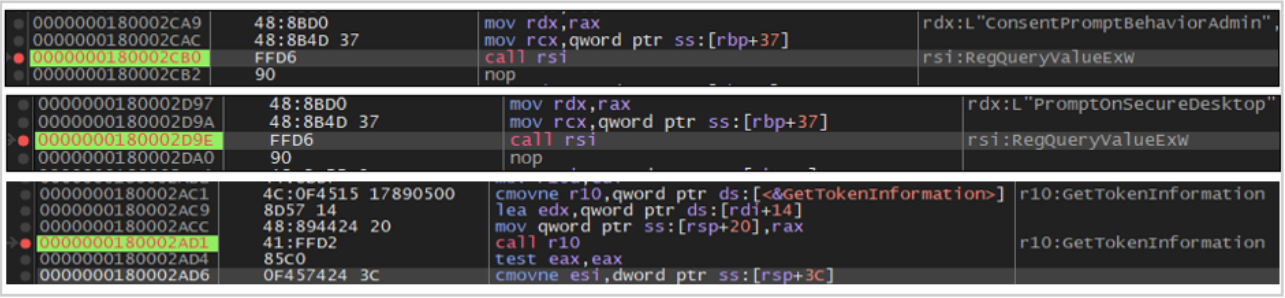
○ The results of these commands are passed through the pipe handle created by 'CreatePipe' and delivered to a memory buffer. These results are then either saved as files or sent to an external server.



[Figure 4-18] Command Execution

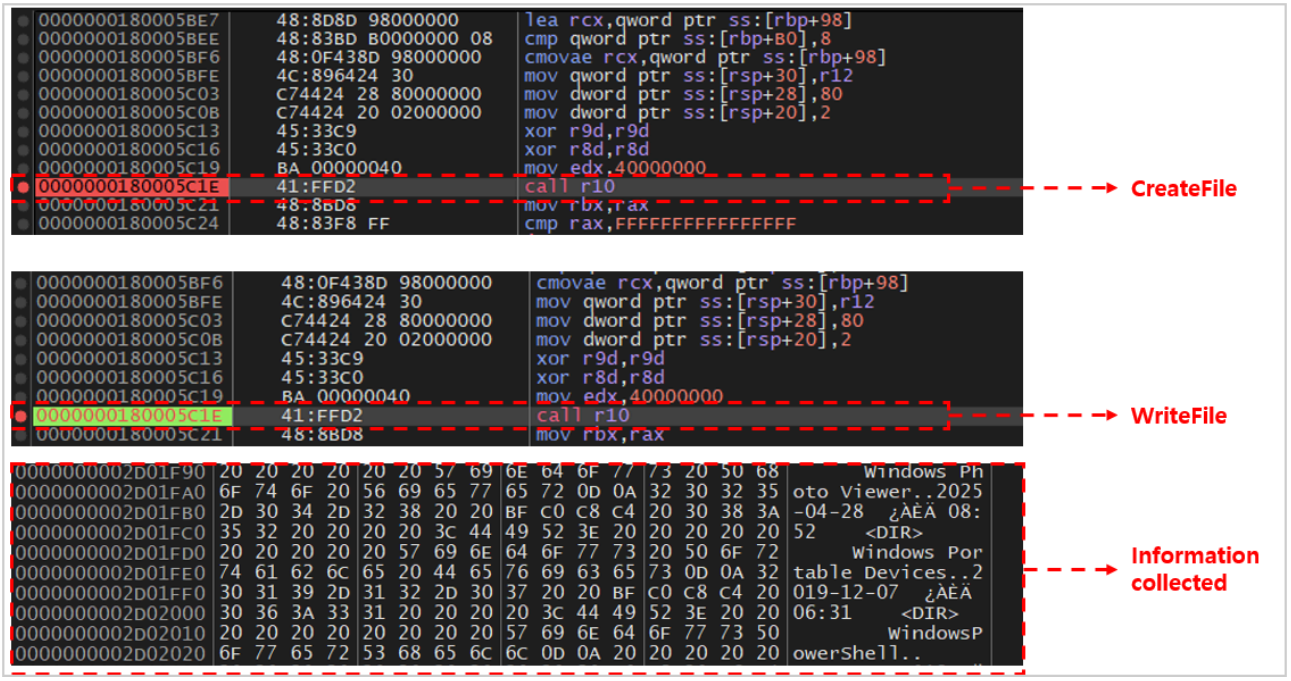
○ After executing the information-gathering commands, the code accesses the registry path 'SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System' to check the values of 'ConsentPromptBehaviorAdmin' and 'PromptOnSecureDesktop'. This checks whether UAC (User Account Control) is enabled.

○ Then, using the OpenProcessToken and GetTokenInformation APIs, the code checks if the currently running process has administrator privileges.



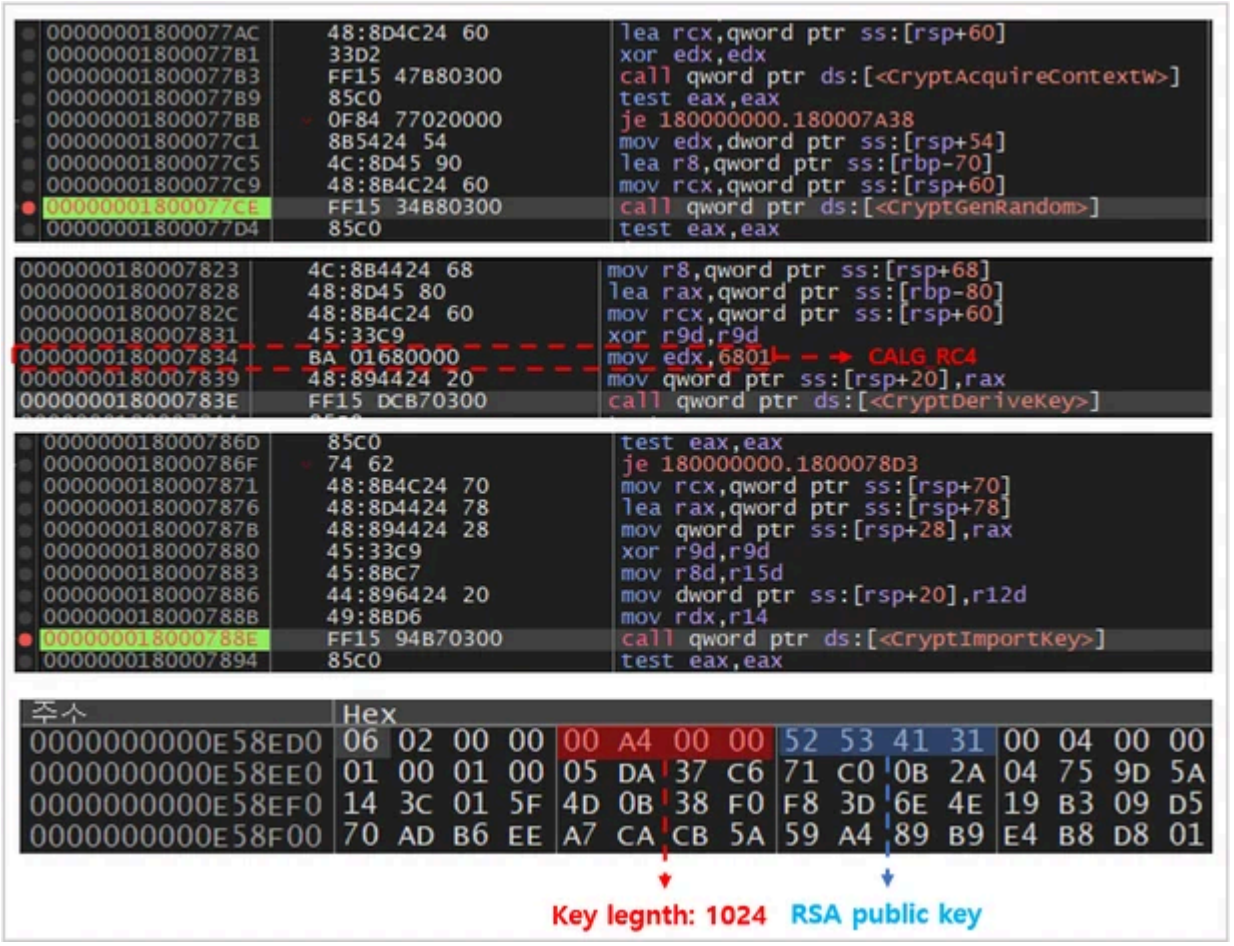
[Figure 4-19] UAC and Administrator Privilege Check

○ The results of the system information collection commands executed via the ‘CreateProcessW’ function are transmitted through the pipe and saved as a file at the following path: C:\Users\[Username]\AppData\Roaming\temp\{random}.tmp

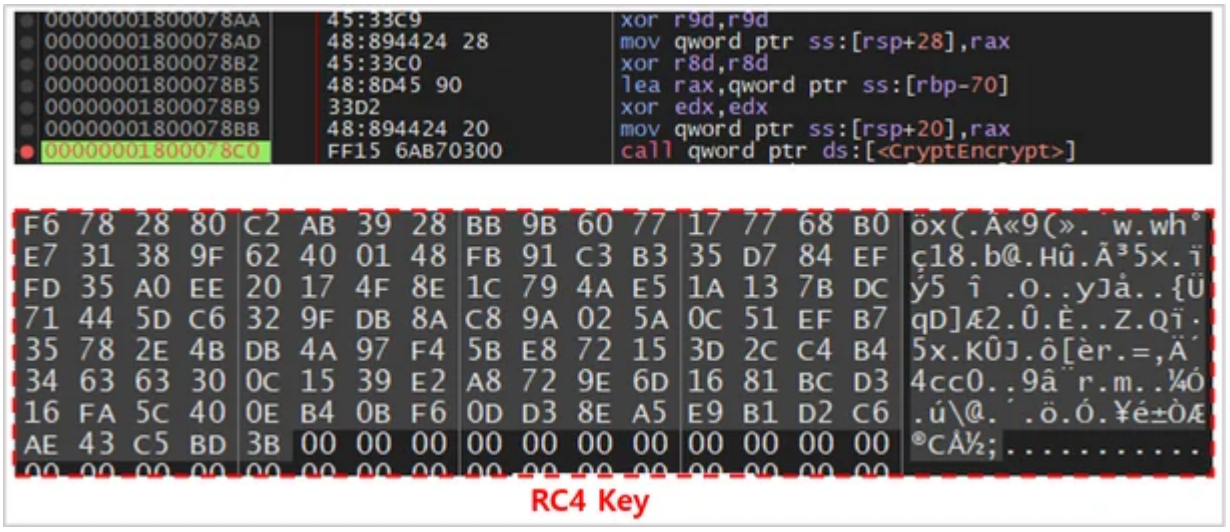


[Figure 4-20] Saving Collected Data

○ The 'CryptGenRandom' function generates 117 bytes of random data. The 'CALG_RC4' algorithm is then specified, and the 'CryptDeriveKey' function is used to generate an RC4 session key. After that, the 'CryptImportKey' function loads a 1024-bit RSA public key, which is used to encrypt the RC4 session key.



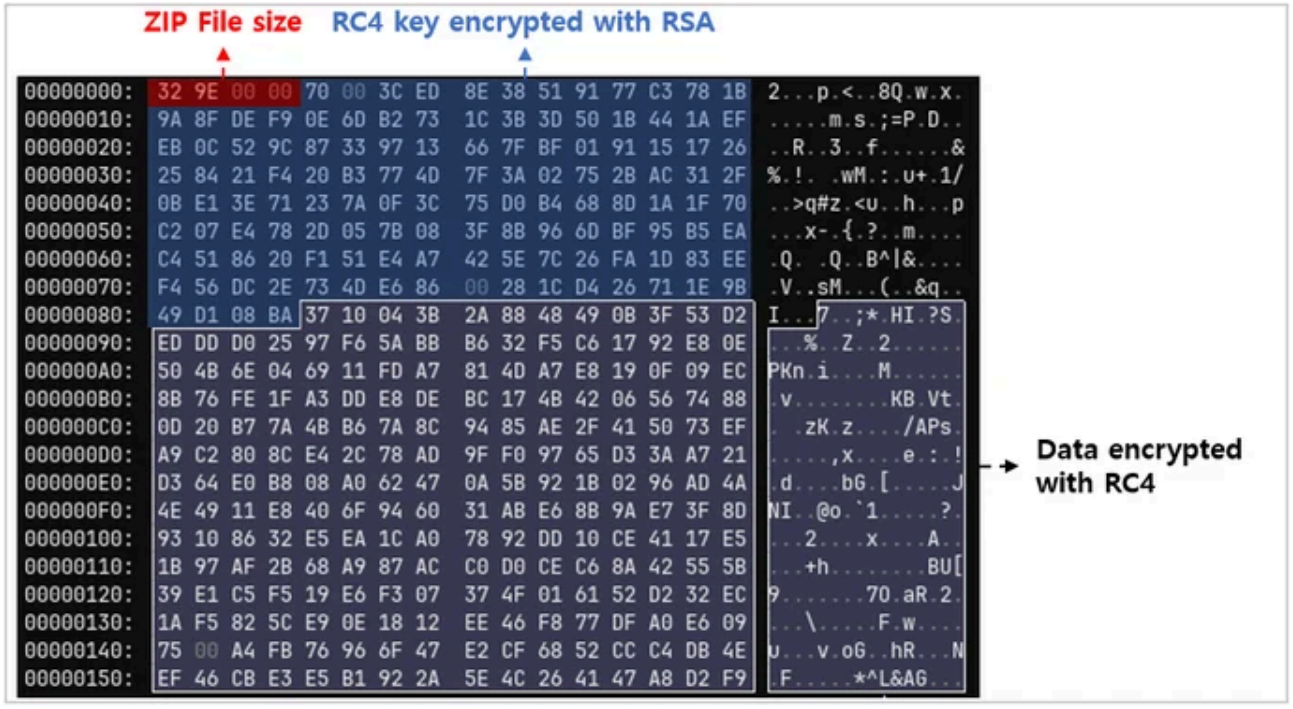
[Figure 4-21] Encryption Key Configuration



[Figure 4-22] RSA Encryption of RC4 Key

○ The '{random}.tmp' file collected in the previous step is compressed into a ZIP archive named '{random}.tmp.zip'.

○ The ZIP file is then encrypted, producing a new file named '{random}.tmp.zip.enc'. This file consists of the following three components: the size of the ZIP file, the RC4 session key encrypted with RSA, and the ZIP data encrypted using RC4.

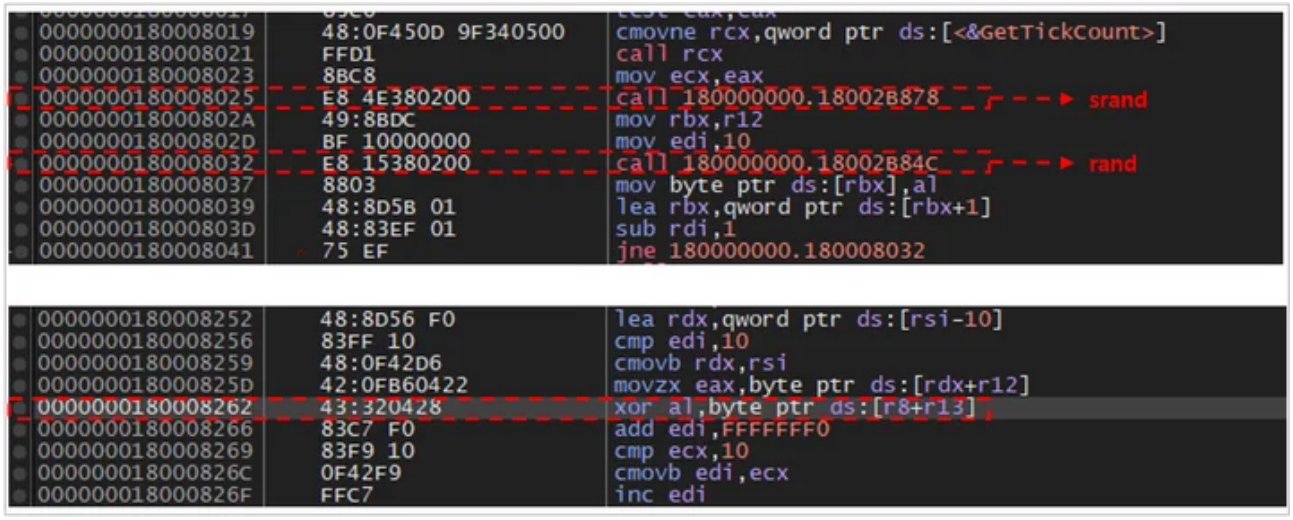


[Figure 4-23] Structure of the '{random}.tmp.zip.enc' File

○ The following steps are performed to encode the '{random}.tmp.zip.enc' file.

○ The value obtained from the 'GetTickCount' function is used as the seed for the 'srand' function. Based on this, the 'rand' function is called 16 times to generate a total of 16 bytes of random data.

○ This random value is used as a key to perform XOR encryption on the data within the .enc file. The generated key is applied in a cyclic manner throughout the encryption process.

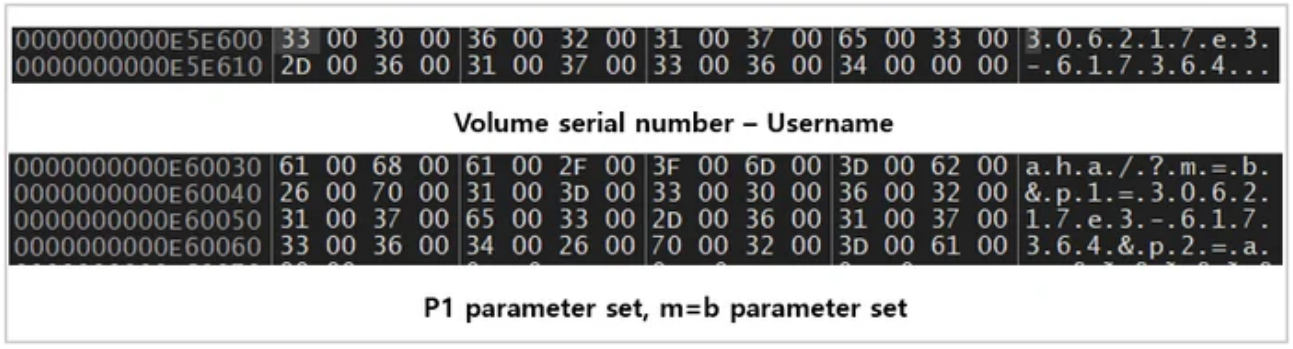


[Figure 4-24] Encoding of the '{random}.tmp.zip.enc' File

○ A file named '{random}.pdf' is created, consisting of the PDF header, 4 bytes of dummy data, a 16-byte XOR key, and the encoded contents of the .enc file. The overall structure matches that of ‘[Figure 4-11] Random tmp File Structure’.

○ A unique identifier string is generated based on the infected system's drive volume serial number and the username. The username is converted to hexadecimal, one character at a time, and the final string is formatted as 'VolumeSerial-Username(in hex)'.

○ The generated string is included as the value of the 'p1' parameter in an HTTP request which is sent to the C2 server. The 'm' parameter with a value of 'b' indicates a data transmission.



[Figure 4-25] 'p1' and 'm' Parameter Configuration

○ An HTTP request is sent to the 'woana.n-e[.]kr' domain, including the previously defined parameters and the data from the '{random}.pdf' file, which is formatted as 'multipart/form-data'.

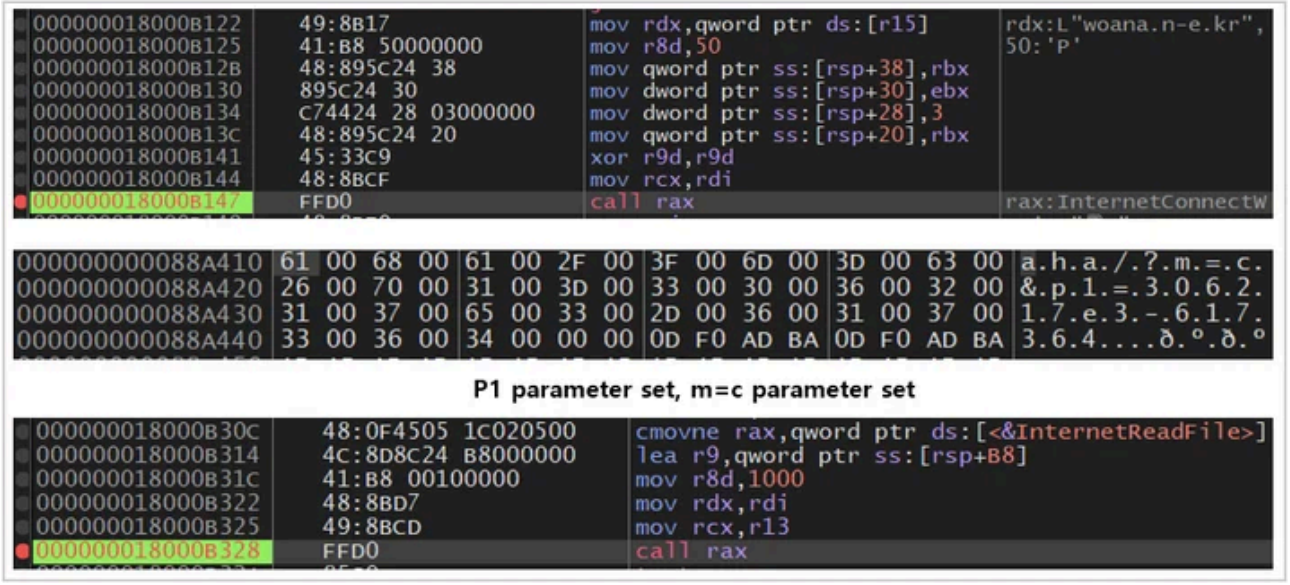
[Figure 4-26] Transmission of Collected Data

○ Once the transmission is complete, a new thread is created to send another HTTP request to the 'woana.n-e[.]kr' domain. The 'p1' parameter remains the same, while the 'm' parameter is set to 'c'.

○ Setting the 'm' parameter to 'c' indicates data reception. The 'woana.n-e[.]kr' domain responds by returning data that contains commands.

○ Upon receiving the commands, the malware saves them to a file at the following path using the 'InternetReadFile' function:
C:\Users\[Username]\AppData\Roaming\temp\{random}.tmp

○ The command is then executed in the same way as before, and the result is sent back via a request with the parameter set to 'm=b'.



[Figure 4-27] Receiving Command Data

○ The malware maintains a loop structure that continuously communicates with the 'woana.n-e[.]kr' domain at regular intervals to send and receive commands. Upon initial execution, it sends collected system information to the 'woana.n-e[.]kr' domain, using the 'p1' parameter to include the unique identifier string and setting the 'm' parameter to 'b'.

○ It then creates a new thread and performs a request with the same 'm' parameter set to 'c'. This indicates command reception, and the response received from the 'woana.n-e[.]kr' domain is saved as a file.

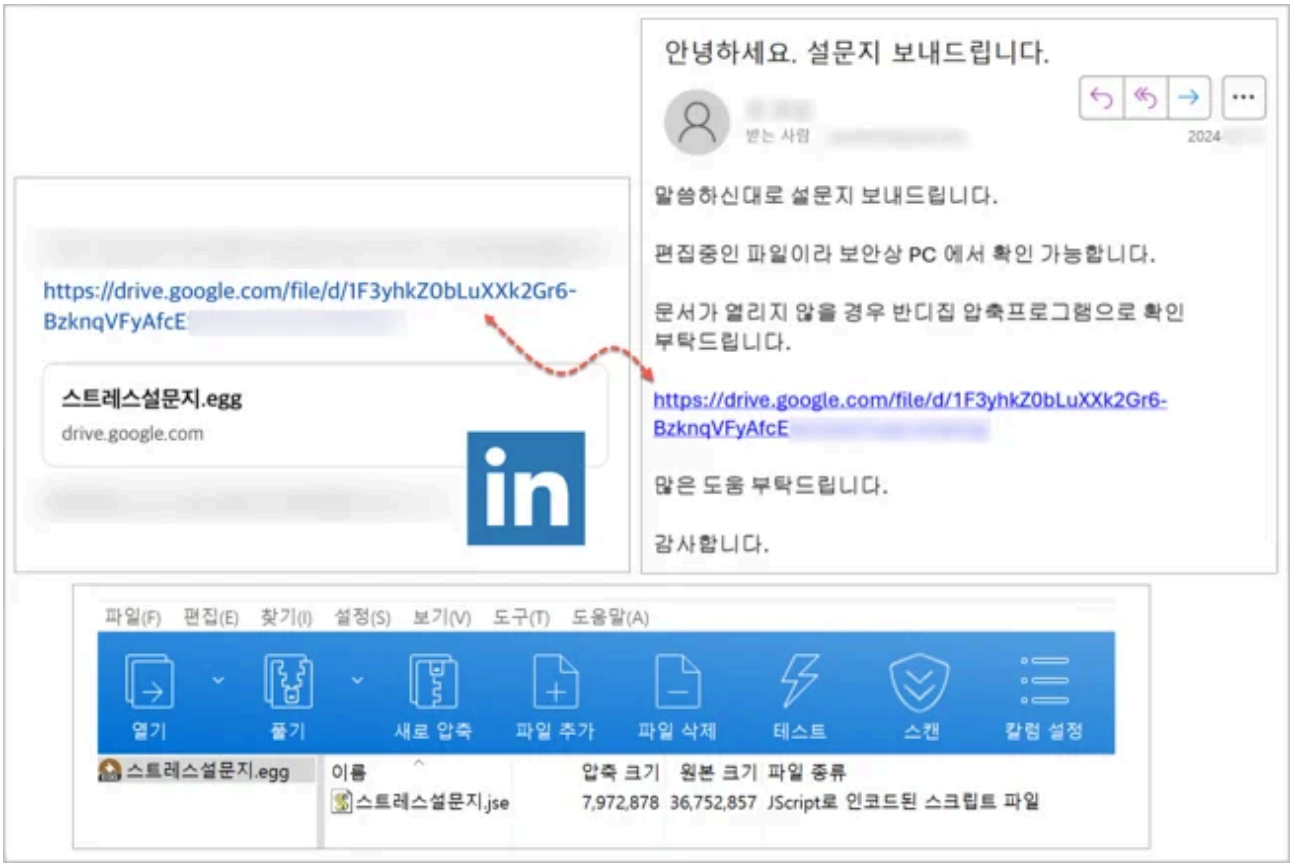
○ The saved file contains executable commands or scripts. The process of executing these commands and the method of transmission are the same as described in ‘[Figure 4-18] Command Execution’ through ‘[Figure 4-26] Transmission of Collected Data’.

○ This malware is a remote access trojan (RAT) that is executed through a DLL loaded via 'regsvr32' and collects system information using RC4 and RSA encryption along with a PDF disguise technique, receives and executes commands from the C2 server, and sends the results back.

5. Similar Variant Cases

5-1. Spear Phishing Similarity Comparison

○ A review of the threat actor’s past activities shows that, in addition to Facebook, there have also been cases of initial access via LinkedIn.



[Figure 5-1] Example of an attack conducted via LinkedIn

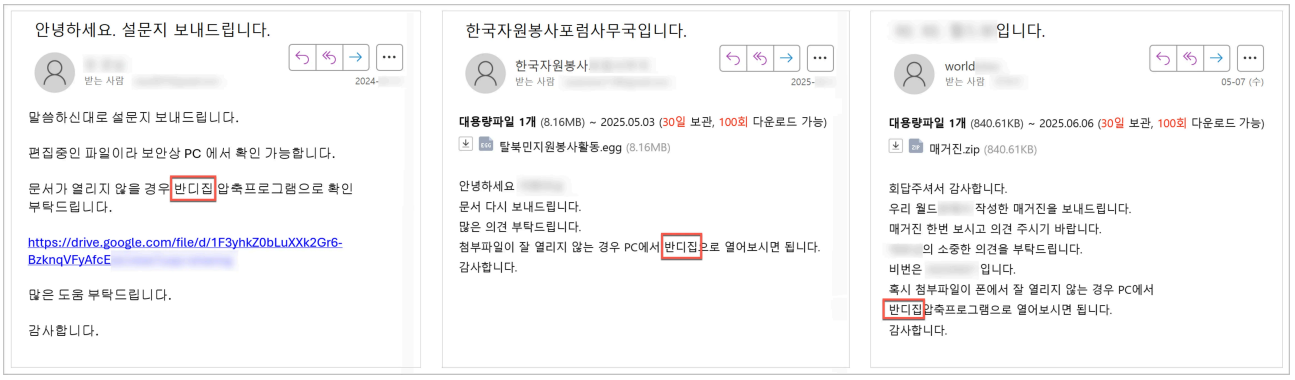
○ In an actual case from 2024, the attacker disguised themselves as a military researcher to approach a graduate of the Korea Naval Academy.

○ LinkedIn, a leading social media platform for professional networking and recruitment, is used to select targets.

○ On LinkedIn, individuals' affiliations, work experience, technical skills, and achievements by field are often publicly available. The platform also allows attackers to search for individuals in specific fields and reach out via direct messages.

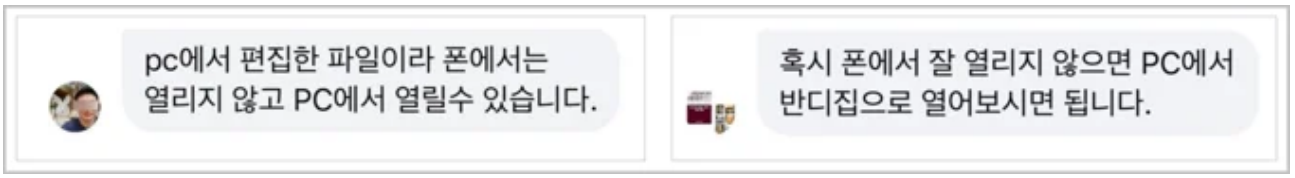
○ A comparison of spear phishing incidents carried out last year and this year shows that both campaigns attempted to lure targets into using the Korean file compression tool Bandizip. This appears to serve two main purposes:

- To ensure that the archive is extracted on a Windows PC rather than a smartphone
- To evade security detection by using encrypted archives in the EGG format



[Figure 5-2] Comparison of file compression instructions

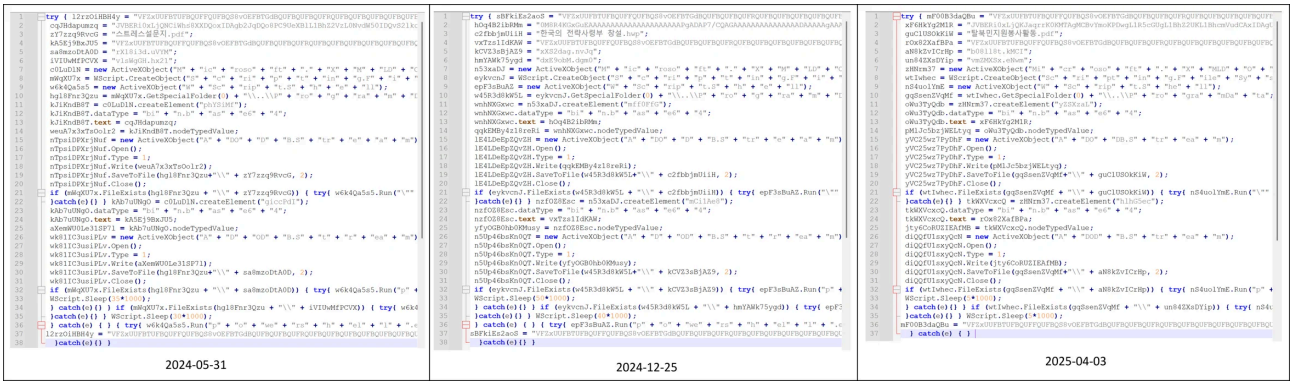
○ Notably, similar phrase has been observed not only in emails but also in messenger conversations.



[Figure 5-3] Comparison of Facebook message wording

5-2. JSE Script Similarity Comparison

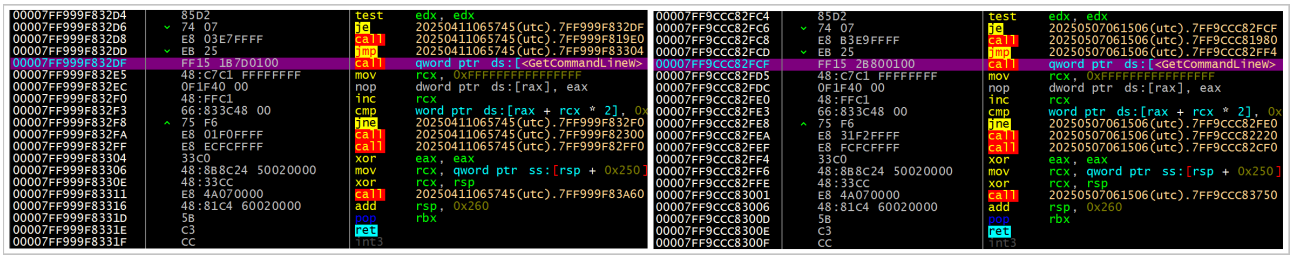
○ A comparison of the cases from May and December 2024 and April 2025 shows that malicious scripts were used in an almost identical pattern, indicating that the threat actor is likely relying on an automated tool for script generation.



[Figure 5-4] Structural comparison of malicious scripts

5-3. DLL Malware Similarity Comparison

○ This figure shows a comparison of functions from malware samples used in attacks in April and May 2025. Although the threat actor modifies the code depending on the variant, samples from similar timeframes share structural similarities.



[Figure 5-5] Function-level comparison of DLL-based malware

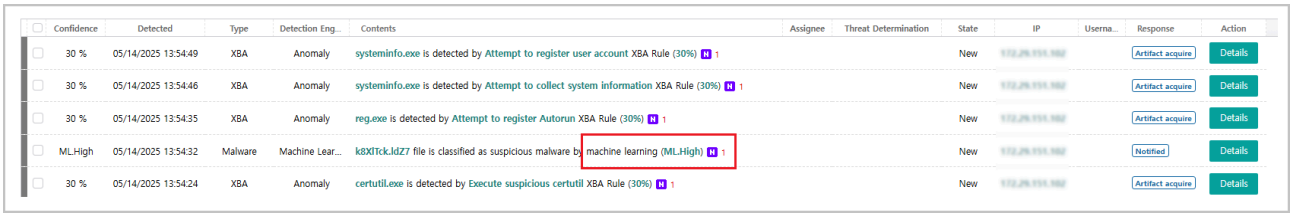
6. Conclusion

○ Nation-state APT attacks are typically carried out in a highly covert manner, with only a small number of cases publicly disclosed.

○ Email-based spear phishing attacks remain highly active. With nothing more than the target's email address, attackers can launch swift and stealthy tailored attacks. In addition, various methods now include the use of social networking platforms and personal messaging apps.

○ The cases described in this report represent only a portion of the broader threat landscape. Sophisticated threat actors continue to diversify their script patterns to evade detection by traditional security products. As such, it is becoming increasingly difficult to accurately detect new, modified threats using signature-based methods alone.

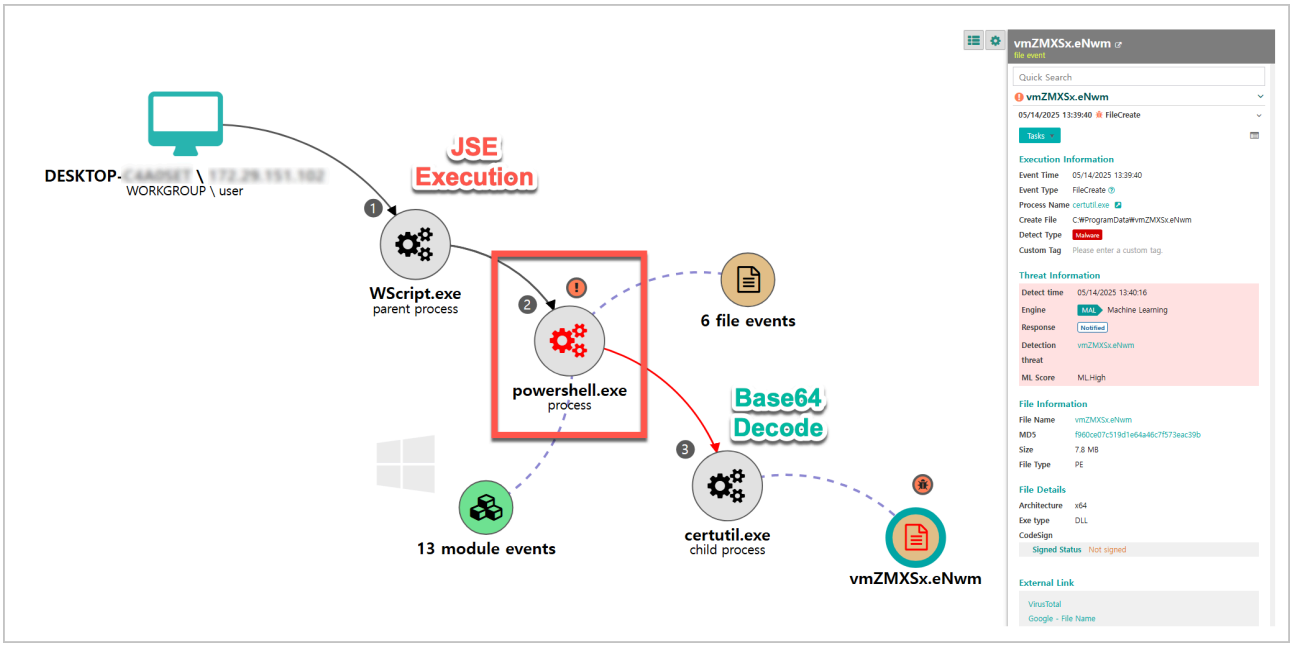
○ The Genian EDR solution not only comes equipped with built-in behavior-based detection rules (XBA) capable of identifying previously unknown threats, but also leverages machine learning–based threat modeling for rapid response and defense.



[Figure 6-1] Machine learning–based detection by Genian EDR

○ In fact, the 'AppleSeed' variant used by the Kimsuky group was detected immediately at the initial execution stage through Genian EDR's machine learning technology.

○ Malicious files in JSE format are typically executed via the WScript.exe process, which is followed by a series of threat activities triggered through PowerShell.exe commands.

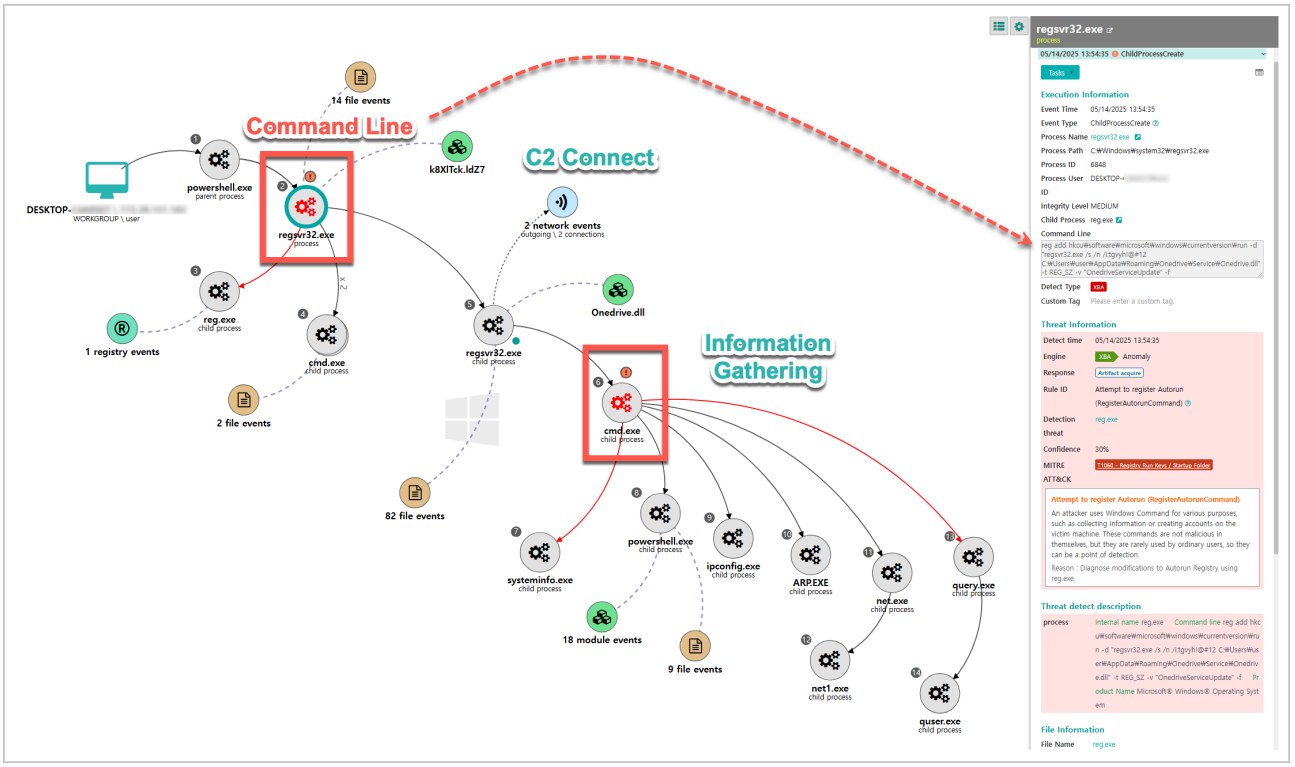


[Figure 6-2] Execution events of the JSE script

○ Genian EDR provides enhanced visibility into attack storylines by clearly mapping parent-child process relationships on the endpoint where the threat was introduced.

○ In addition, it enables immediate identification of Base64-encoded data embedded within the script being decoded via the CertUtil.exe process.

○ Beyond visualizing the threat execution flow, it also supports proactive threat hunting through per-endpoint ‘event investigation’ and ‘LIVE search’.



[Figure 6-3] Threat visibility enabled by Genian EDR

○ With insights from EDR detections, security administrators can efficiently monitor and manage abnormal activity on affected endpoints.

○ Genian EDR makes it easy for administrators to view the exact command-line arguments used during the execution of 'AppleSeed' through its detailed information panel. In addition, built-in MITRE ATT&CK mappings provide a more structured and informed approach to threat management.

○ By adopting EDR, security teams can actively respond to a wide range of threats targeting internal endpoints. Key event data is retained for each device, making it easy to review past activity over specific timeframes. This helps streamline evidence collection and identify the root cause of incidents more effectively.

7. Indicator of Compromise

- **MD5**

2f6fe22be1ed2a6ba42689747c9e18a0

5a223c70b65c4d74fea98ba39bf5d127

7a0c0a4c550a95809e93ab7e6bdcc290

46fd22acea614407bf11d92eb6736dc7

568f7628e6b7bb7106a1a82aebfd348d

779f2f4839b9be4f0b8c96f117181334

07015af18cf8561866bc5b07e6f70d9a

7756b4230adfa16e18142d1dbe6934af

8346d90508b5d41d151b7098c7a3e868

30741e7e4cdd8ba9d3d074c42deac9b1

537806c02659a12c5b21efa51b2322c1

afadab22f770956712e9c47460911dad

b9c2111c753b09e4cc9d497f8fd314fc

b128c5db5d973be60f39862ba8bfb152

bfb02dee62c38c3385df92b308499b31

ca3926dc6c4b2a71832a03fba366cbcd

ec9dcef04c5c89d6107d23b0668cc1c1

f4d59b1246e861a2a626cb56c55651f0

f14f332d4273de04ba77e38fd3dcff90

f960ce07c519d1e64a46c7f573eac39b

fb3c652e795f08cc2529ed33ec1dc114

fe8626e7c3f47a048c9f6c13c88a9463

1ae2e46aac55e7f92c72b56b387bc945

2a388f3428a6d44a66f5cb0b210379a0

- **C2**

afcafe.kro[.]kr

dirwear.000webhostapp[.]com

download.uberlingen[.]com

hyper.cadorg.p-e[.]kr

jieun.dothome.co[.]kr

nauji.n-e[.]kr

nocamoto.o-r[.]kr

nomera.n-e[.]kr

onsungtong.n-e[.]kr

peras1.n-e[.]kr

update.screawear[.]ga

vamboo.n-e[.]kr

woana.n-e[.]kr
