

マルウェアLODEINFOの進化 - JPCERT/CC Eyes

By 喜野 孝太(Kota Kino)

Published: 2020-06-10 · Archived: 2026-04-05 22:57:25 UTC

2020/06/11

- [LODEINFO](#)

[以前](#)のブログで、日本国内の組織を狙ったマルウェアLODEINFOについて紹介しました。JPCERT/CCでは、現在もこのマルウェアを使用した攻撃が活発に行われていることを確認しており、新型コロナウイルスに関連したファイル名などを使って、感染を広げようとする動きが見られます。また、LODEINFOは頻繁にアップデートが行われており、複数の機能が追加・変更されていることを確認しています。

今回は、LODEINFOの一連の攻撃の中で見られた傾向と、アップデート内容について紹介します。

LODEINFOを配信する手口

確認されている全ての攻撃の起点は、添付ファイル付きの標的型攻撃メールです。添付ファイルにはWord文書、またはExcel文書が使用されており、添付ファイルを開いてマクロを有効化することで、内包していたLODEINFOがホスト上に作成、実行されます。標的型攻撃メールに使用されているメールと添付ファイルの内容は、以下のようなものを確認しています。

- 新型コロナウイルスを題材にしたもの
- 日露や日韓の外交を題材にしたもの
- 企業への履歴書や申し込みを装ったもの

攻撃対象の業種としては、メディア系、公共系を確認しています。また、標的型攻撃メールの送信には、フリーのメールアドレス（Gmail等）を使用している傾向が見られます。

LODEINFOのバージョン

以前のブログで紹介したLODEINFOのバージョンは、v0.1.2でしたが、本ブログ執筆時点で確認している最新バージョンは、v0.3.6となっています。また、JPCERT/CCでは以下のバージョンのLODEINFOが存在することを確認しています。

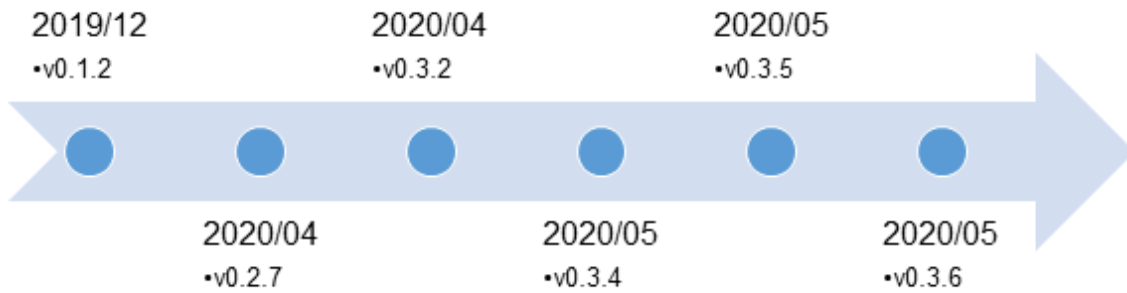


図 1：バージョンの推移

なお、各バージョンで追加された主な機能としては、以下のようなものを確認しています。

| バージョン | 機能追加 |
|--------|------------------------------|
| v0.2.7 | データ送受信フォーマットの一部変更 |
| | 既存コマンドの拡張 (ver) |
| | Mutexの作成 |
| v0.3.2 | 新規コマンドの追加 (print) |
| | 永続化処理の追加 |
| v0.3.5 | 新規コマンドの追加 (rm、ransom、keylog) |

新規コマンドの追加

現時点での最新バージョン (v0.3.6) では、前回のv0.1.2の検体から以下のコマンドが追加されています。

- print
- rm
- ransom
- keylog

printコマンドは感染ホストのスクリーンキャプチャを取得し、rmコマンドは指定されたファイルの削除を行います。例えばrmコマンドを実行した場合、ファイル削除後、以下のような実行結果がC&Cサーバ宛てに送信されます。

```
1590318292|932|080027D50FB0|DESKTOP-J783225C:\Users\Public\Pictures\Sample Pictures\Chrysanthemum.jpg
C:\Users\Public\Pictures\Sample Pictures\Desert.jpg: OK.
```

```
C:\Users\Public\Pictures\Sample Pictures\desktop.ini: OK.  
C:\Users\Public\Pictures\Sample Pictures\Hydrangeas.jpg: OK.  
C:\Users\Public\Pictures\Sample Pictures\Jellyfish.jpg: OK.  
C:\Users\Public\Pictures\Sample Pictures\Koala.jpg: OK.  
C:\Users\Public\Pictures\Sample Pictures\Lighthouse.jpg: OK.  
C:\Users\Public\Pictures\Sample Pictures\Penguins.jpg: OK.  
C:\Users\Public\Pictures\Sample Pictures\Tulips.jpg: OK.
```

ransom、keylogコマンドについては、現時点で確認しているバージョンでは未実装となっており、コマンドを実行しても以下のような結果だけがC&Cサーバ宛てに送信されます。

```
1590318292|932|080027D50FB0|DESKTOP-J783225Not available
```

ただし、コマンド名から推測すると、将来的にファイルの暗号化やキーログ機能が搭載される可能性があるかもしれません。



図 2 : コマンドの一覧

```

}
else if ( (unsigned __int8)aa_EqualsCommand(&recv_command, ransom) )
{
    strcpy(var_23C, "Not available");
    aa_SetDataToSend(var_23C, 0);
}
else if ( (unsigned __int8)aa_EqualsCommand(&recv_command, keylog) )
{
    strcpy(v117, "Not available");
    aa_SetDataToSend(v117, 0);
}
else
{

```

図 3 : ransom、keylogコマンド処理部

データ送受信フォーマットの一部変更

LODEINFOは、AESとBASE64を組み合わせてデータの暗号化を行っていますが、データをBASE64デコードした後のオフセット0x45の位置には、AESで暗号化されたデータのサイズが記載されています。

```

00000000: 0c86 a3a9 c739 955b 89a6 3c2f af7e a6b1 .....9.[...</.~..
00000010: 7400 0000 566c 7e3b 5e60 b32d a9ce 8192 t...Vl~;^`.~....
00000020: 5c1d dceb 9125 e3b1 5052 1e4d 631c e887 \....%..PR.Mc...
00000030: 55d2 a20d a7b2 7ab8 79ff 0ef2 629e 7e5f U....z.y...b.~_
00000040: 50fd e803 6920 0000 002f 263a e9eb 99c7 P...i .../&:....
00000050: 14e0 3649 19ab dd8f 183e e985 19e9 38f6 ..6I.....>....8.
00000060: 46a1 3077 990b 19d7 1f39 0000                F.0w.....9..

```

図 4 : 変更前のデータフォーマット

前回のv0.1.2の検体では、該当部分にデータサイズがそのまま記載されていましたが、v0.2.7以降の検体からは、オフセット0x49の位置に新しく1byteのXORキーが記載されるようになり、オフセット0x45のデータサイズにはXORキーでエンコードされた値が記載されるようになっています。

```

00000000: f720 4e40 9f33 3c20 1370 750c 4aec 8862 . N@.3< .pu.J..b
00000010: b400 0000 b20d 25ed 3728 9a29 b9db 9d08 .....%.7(.)....
00000020: ea2d 40c3 8816 b83a 5f49 69d8 4341 5fd9 .-@....:_Ii.CA_
00000030: ac28 defe 761c 7c36 79ec a9ba c04e ce11 .(.v.|6y...N..
00000040: 5755 ea5c 38db 8b8b 8b8b cb24 c354 4678 WU.\8.....$.TFx
00000050: ba98 b91f 072c a124 6062 df1a 7ba1 d800 .....,$`b..{...
00000060: 2177 0f40 4495 06af d64d 1d10 c416 ad36 !w.@D...M.....6
00000070: e420 dd37 c82d 03eb d00a 36d4 9471 79d0 . .7.-....6..qy.
00000080: 6c23 b72a ba19 b6dc fd94 e5c7 17d3 8155 l#.*.....U
00000090: e4c7 f0a5 4e06 8d2c be44                ....N...D

```

図 5 : 変更後のデータフォーマット

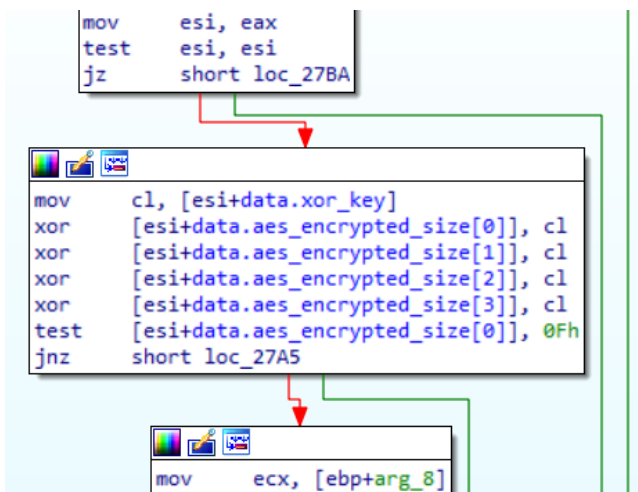


図 6 : XORの処理部

上記の変更によって、以前に紹介したHTTP POSTリクエストのデータを復号するコードが正常に動作しなくなっているため、以下に対応したコードの一部を記載します。

```

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)
    xor_key = postdata[0x34]
    decrypt_size = int.from_bytes([b ^ xor_key for b in postdata[0x30:0x34]], byteorder="little")
    dec_data = cipher.decrypt(postdata[0x35:0x35+decrypt_size])

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

    return dec_data

encrypted_data = "njgGCEgbkXQIgexSrDm307QAAADuSiTM6xoP8ResYAybhHoRx9W-Ulw_ealn9gIEjvsZzqQXG8vn3QYoIfi

```

```
KEY = a2b_hex("7306ED96A7D75BAB94C4F15AAF0A9E61690F0E300FEA9135764C206580DF2970")
IV = a2b_hex("D5C5376805264812B3ED88BE4A614A1A")

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

LODEINFOの起動方法の変化

以前のLODEINFOでは、Word文書のマクロを有効化した際に、LODEINFOのDLLファイルをホスト上に作成し、rundll32.exe を使って実行していました。しかし、v0.3.2以降からは、DLLファイルと一緒に正規のWindows実行ファイルを作成し、DLLサイドローディングを使ってLODEINFOを実行するように手法が変わっています。

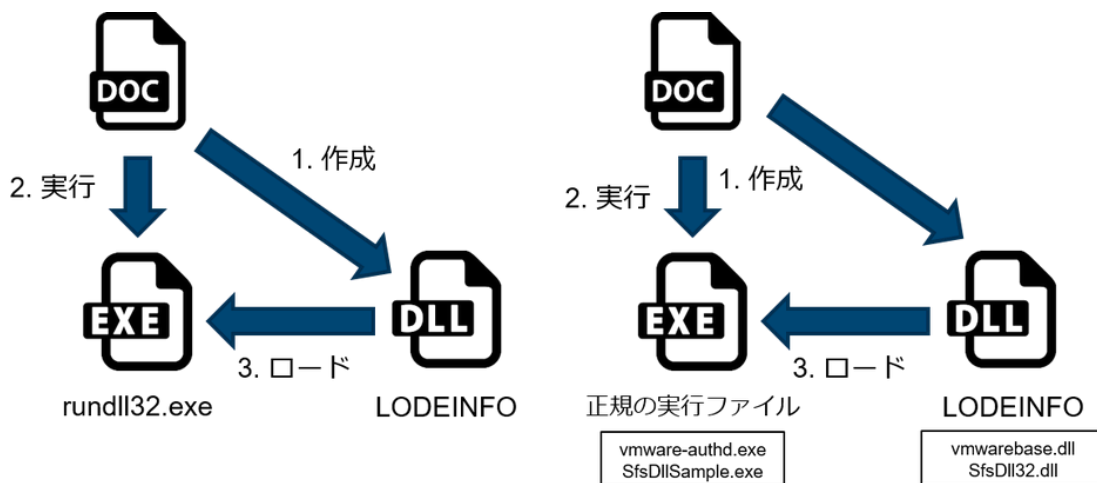


図 7：起動方法の変化（左: rundll32.exeによる実行、右: DLLサイドローディングによる実行）

LODEINFOの通信特徴

LODEINFOはUser-Agentが検体内部でハードコードされており、v0.2.7までは以下が使用されています。

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90
```

また、v0.3.2以降では以下が使用されています。

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.100
```

C&Cサーバのインフラには、様々な国のISPが利用されています。

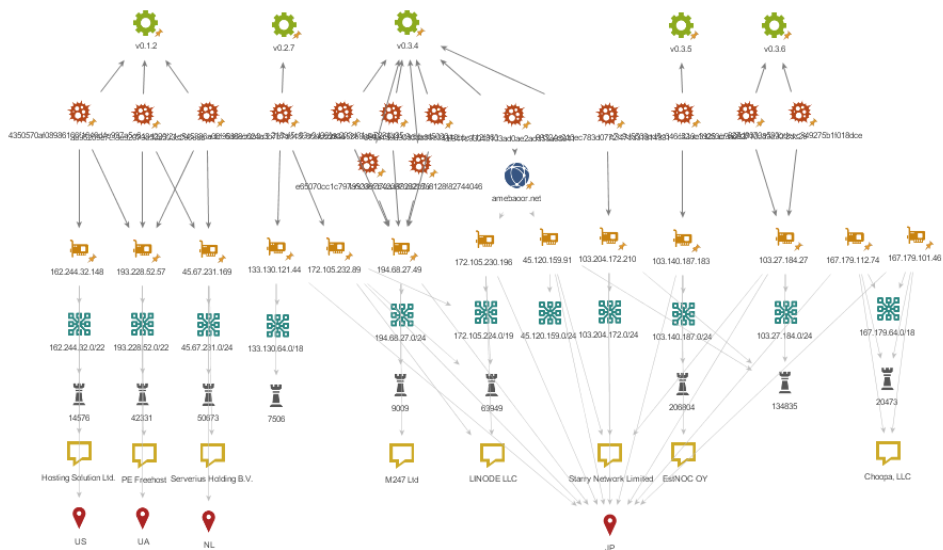


図 8 : C&Cサーバのインフラ分布

おわりに

マルウェアLODEINFOの開発は頻繁に行われており、同様に攻撃も継続して確認されています。今後もこのマルウェアを使用した攻撃が続く可能性がありますので、引き続き注意が必要です。

なお、今回解説した検体のハッシュ値をAppendix A、新たに確認した通信先をAppendix Bに記載しています。Appendix B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、佐條 研

Appendix A 検体のハッシュ値

- 65433fd59c87acb8d55ea4f90a47e07fea86222795d015fe03fba18717700849 (v0.3.6)
- 8c062fef5a04f34f4553b5db57cd1a56df8a667260d6ff741f67583aed0d4701 (v0.3.5)
- 1cc809788663e6491fce42c758ca3e52e35177b83c6f3d1b3ab0d319a350d77d (v0.3.2)

Appendix B 通信先

- 103.27.184.27
- 103.140.187.183
- 103.204.172.210
- 133.130.121.44
- 167.179.101.46
- 167.179.112.74
- 172.105.232.89
- 194.68.27.49
- www.amebaor.net



[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



[JSAC2026 開催レポート～DAY 2～](#)

```

*key = 0x027c7680,
*key[4] = 0xd15933c2,
*key[8] = 0x04c7283c,
*key[12] = 0x0000796c,
iv[0] = 0x1247a421,
iv[4] = 0x04803a8c,
iv[8] = 0x30780579,
iv[12] = 0x9f3888d7,
v1 = m_ret_arg1Offset@350(a1 + 3);
if ( !((V3-CryptAcquireContext)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x18, 0xf0000000) )
return 0;
v3 = m_ret_arg1Offset@350(a1 + 3);
handlehashobj = a3 + 1;
if ( !((V3-CryptCreateHash)(*a3, 0x0004, 0, 0, a1 + 1) )
{
LABEL_0:
if ( !*a4 )
return 0;
v6 = m_ret_arg1Offset@350(a4 + 3);
(v6->CryptInitializeContext)(*a4, 0);
return 0;
}
if ( !CryptHashData(*handlehashobj, key, 16u, 0)
|| (v8 = m_ret_arg1Offset@350(a1 + 3),
v8 + a3 + 2,
!(v8->CryptDeriveKey)(*a1, 0x0004, *handlehashobj, 0x000000, a1 + 2) )// CALC_AES_128
{
if ( *handlehashobj )
{
v5 = m_ret_arg1Offset@350(a1 + 3);
(v5->CryptDestroyHash)(*handlehashobj);
}
goto LABEL_0;
}
v10 = m_ret_arg1Offset@350(a4 + 3);
(v10->CryptSetKeyParam)(*v8, 3, 0x0000, 0); // SP_PADDING = 0x00000077
v11 = m_ret_arg1Offset@350(a4 + 3);
(v11->CryptSetKeyParam)(*v8, 1, 0x, 0); // IV = parameter
v12 = m_ret_arg1Offset@350(a4 + 3);
(v12->CryptSetKeyParam)(*v8, 4, 0x0000, 0); // SP_MODE = CBC
return *v9;

```

[攻撃グループAPT-C-60による攻撃のアップデート](#)

```

python parse_cross2beacon_config.py beacon.bin
[+] Decoded Config Data
Offset 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Encode to ASCII
000000 29 01 00 00 7f 00 00 01 b3 15 00 00 09 00 00 00 ).....
000010 31 32 37 2e 30 2e 30 2e 31 00 00 00 0c 01 00 127.0.0.1.....
000020 00 2d 2d 2d 2d 2d 42 45 47 49 4e 20 50 55 42 4c -----BEGIN.PUBL
000030 49 43 20 4b 45 59 2d 2d 2d 2d 2d 2d 0a 4d 49 47 66 IC.KEY-----,MIGF
000040 4d 41 30 47 43 53 71 47 53 49 62 33 44 51 45 42 MA0GCSqGS1b3DQEB
000050 41 51 55 41 41 34 47 4e 41 44 43 42 69 51 4b 42 AQUAA4GNADCB1QKB
000060 67 51 43 4e 53 33 38 6c 48 50 32 56 33 4a 44 34 gQcNS381HP2V3JD4
000070 47 54 39 55 63 61 4c 68 41 6b 70 4d 64 51 41 47 GT9UcalhAkPmDQAG
000080 52 6e 36 4e 77 36 52 48 6e 56 35 54 2f 69 48 4a Rn6Nw6RHnVST/1HJ
000090 2b 7a 48 4c 48 38 32 71 37 58 4b 6d 6f 2b 72 55 +zHLH82q7Xkmo+rU
0000A0 2b 49 7a 59 70 58 6e 57 55 37 70 4d 73 69 53 64 +IzYpXmU7pMs1Sd
0000B0 71 2b 63 52 78 4d 6f 54 4c 6d 68 4e 6f 71 32 55 +cRkMoTlMhNoq2U
0000C0 54 57 4b 39 6f 39 52 6f 64 63 5a 7a 5a 58 73 6b TwK9o9RodcZtZxsk
0000D0 62 4d 37 54 7a 4b 37 55 5a 6a 79 61 70 54 49 4a bM7TzK7UZjyapTIJ
0000E0 66 63 71 36 42 57 4d 64 73 4d 78 36 67 48 34 4f fcq6BwMdsMx6gH40
0000F0 73 6c 42 2f 35 77 6e 63 33 77 51 78 55 62 4f 61 s1B/Swnc3wXubOa
000100 71 45 6f 6b 4b 6f 72 5a 77 6d 68 55 33 77 49 44 qEokKorZumHU3wID
000110 41 51 41 42 0a 2d 2d 2d 2d 2d 45 4e 44 20 50 55 AQAB-----END.PU
000120 42 4c 49 43 20 4b 45 59 2d 2d 2d 2d 2d 41 41 41 BLIC.KEY-----AAA
000130 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....
[+] Config Data
C2: 127.0.0.1:5555
PUBLICKEY: -----BEGIN PUBLIC KEY-----
MIGFMA0GCSqGS1b3DQEB AQUAA4GNADCB1QKBgQCNS381HP2V3JD4GT9UcalhAkPmDQAGRn6Nw6
RHnVST/1HJ+zHLH82q7Xkmo+rU+IzYpXmU7pMs1Sdq+cRkMoTlMhNoq2UTwK9o9RodcZtZxsk
bM7TzK7UZjyapTIJfcq6BwMdsMx6gH40s1B/Swnc3wXubOaqEokKorZumHU3wIDAQAAB
-----END PUBLIC KEY-----

```

[Cobalt Strike Beaconの機能をクロスプラットフォームへと拡張するツール「CrossC2」を使った攻撃](#)

```

00 00 05 0C 0A 04 00 movsx eax, cs:num7
00 0F 02 C8 movd xmm1, eax
00 0F 02 C8 cvtdq2pd xmm1, xmm1
00 0F 05 DC 0A 04 00 movsx eax, cs:num1
00 0F 04 C8 movd xmm0, eax
00 0F 04 C8 cvtdq2pd xmm0, xmm0
00 0F 04 C8 addsd xmm0, xmm0
00 0F 04 C8 subsd xmm1, xmm0
00 0F 04 C8 mulsd xmm1, xmm2
00 0F 05 00 00 movsd [rbp+410h+phPrev], xmm1
00 0F 04 C8 call ret3
00 0F 04 C8 movsx r9d, al
00 0F 04 C8 call ret0
00 0F 04 C8 movsx ecx, al
00 0F 04 C8 imul r9d, ecx
00 0F 04 C8 call ret7
00 0F 04 C8 movsx eax, al
00 0F 04 C8 add eax, r9d
00 0F 04 C8 movsx ecx, cs:num9
00 0F 04 C8 add eax, ecx
00 0F 04 C8 movsx ecx, cs:num8
00 0F 04 C8 xor edx, edx
00 0F 04 C8 div ecx, ecx
00 0F 04 C8 movsx ecx, cs:num1
00 0F 04 C8 cmp eax, ecx
00 0F 04 C8 jz short loc_7FF8581895C0
00 0F 04 C8 call ret3
00 0F 04 C8 movsx edx, al
00 0F 04 C8 movsx eax, cs:num8
00 0F 04 C8 imul edx, eax
00 0F 04 C8 lea r8d, [rdi+rdx*2]
00 0F 04 C8 add r8d, r8d
00 0F 04 C8 call ret3
00 0F 04 C8 movsx ecx, al
00 0F 04 C8 sub r8d, ecx
00 0F 04 C8 call ret6
00 0F 04 C8 movsx ecx, al
00 0F 04 C8 add r8d, ecx
00 0F 04 C8 movsx ecx, cs:num3
00 0F 04 C8 add ecx, r8d

```

[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DslogdRAT](#)

Source: <https://blogs.jpCERT.or.jp/ja/2020/06/LODEINFO-2.html>