マルウエアLODEINFOの進化 - JPCERT/CC Eyes | JPCERT コーディネーションセンター公式ブログ

blogs.jpcert.or.jp/ja/2020/06/LODEINFO-2.html

喜野 孝太(Kota Kino)

2020/06/11

マルウエアLODEINFOの進化

LODEINFO

-
- メール

以前のブログで、日本国内の組織を狙ったマルウエアLODEINFOについて紹介しました。
JPCERT/CCでは、現在もこのマルウエ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。

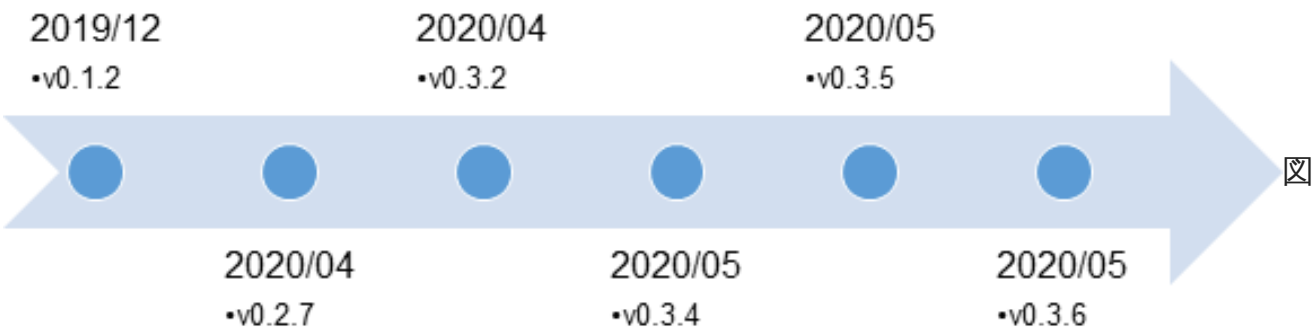


図1：バージョンの推移

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

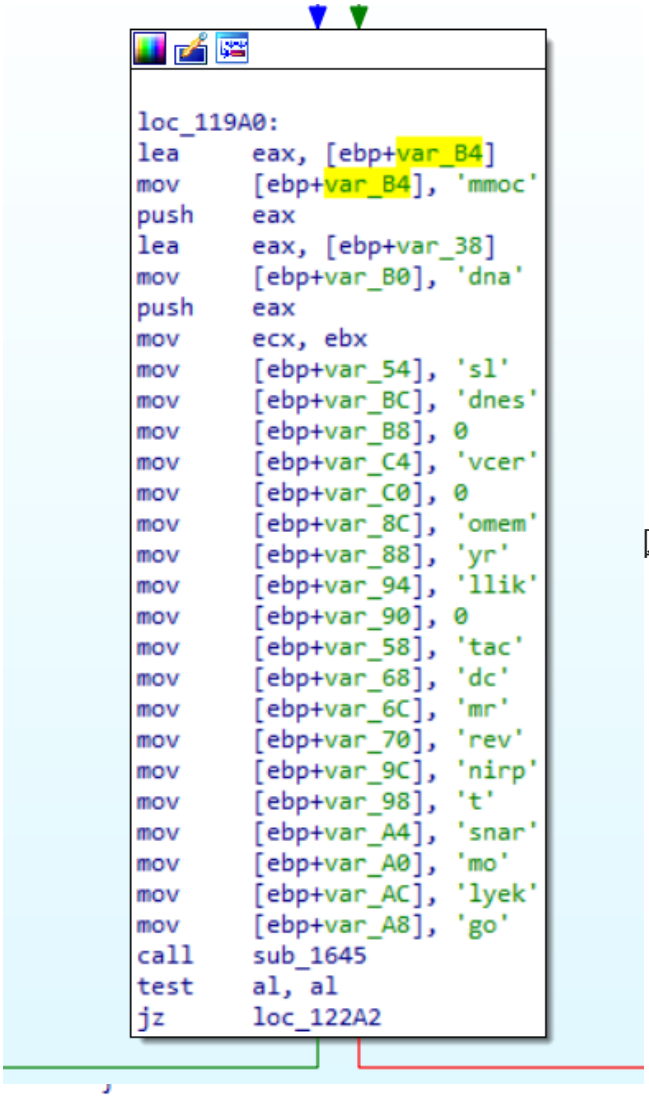
printコマンドは感染ホストのスクリーンキャプチャを取得し、rmコマンドは指定されたファイルの削除を行います。 例えばrmコマンドを実行した場合、ファイル削除後、以下のような実行結果がC&Cサーバ宛てに送信されます。

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



```
loc_119A0:
lea     eax, [ebp+var_B4]
mov     [ebp+var_B4], 'mmoc'
push    eax
lea     eax, [ebp+var_38]
mov     [ebp+var_B0], 'dna'
push    eax
mov     ecx, ebx
mov     [ebp+var_54], 'sl'
mov     [ebp+var_BC], 'dnes'
mov     [ebp+var_B8], 0
mov     [ebp+var_C4], 'vcer'
mov     [ebp+var_C0], 0
mov     [ebp+var_8C], 'omem'
mov     [ebp+var_88], 'yr'
mov     [ebp+var_94], 'llik'
mov     [ebp+var_90], 0
mov     [ebp+var_58], 'tac'
mov     [ebp+var_68], 'dc'
mov     [ebp+var_6C], 'mr'
mov     [ebp+var_70], 'rev'
mov     [ebp+var_9C], 'nirp'
mov     [ebp+var_98], 't'
mov     [ebp+var_A4], 'snar'
mov     [ebp+var_A0], 'mo'
mov     [ebp+var_AC], 'lyek'
mov     [ebp+var_A8], 'go'
call    sub_1645
test    al, al
jz      loc_122A2
```

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

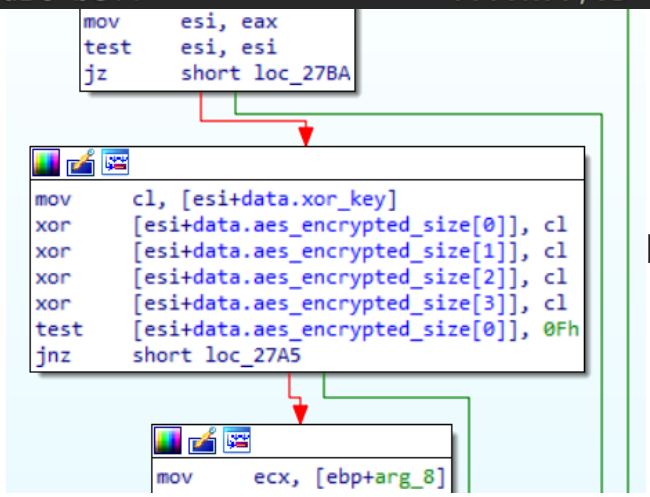


図6：XORの処理部

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 decypt_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 = "njgGCEgbkXQIgexSrDm3O7QAAADuSiTM6xoP8ResYAybhHoRx9W-
Ulw_ealn9gIEjvsZzqQXG8vn3QYoIfmNmO4viy0rFkZGRkaN6IX4HXa-
cdyoRLWkIYxVPI9Ciu8sDP1PK0x6gDH556OYX8GMdejk40daIbiwY3ERd0qL8jRawpwBHht7Sps_hwoZfeks-
ly5sw2Y9RqtUQ.."

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

decrypted_data = decypt_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。

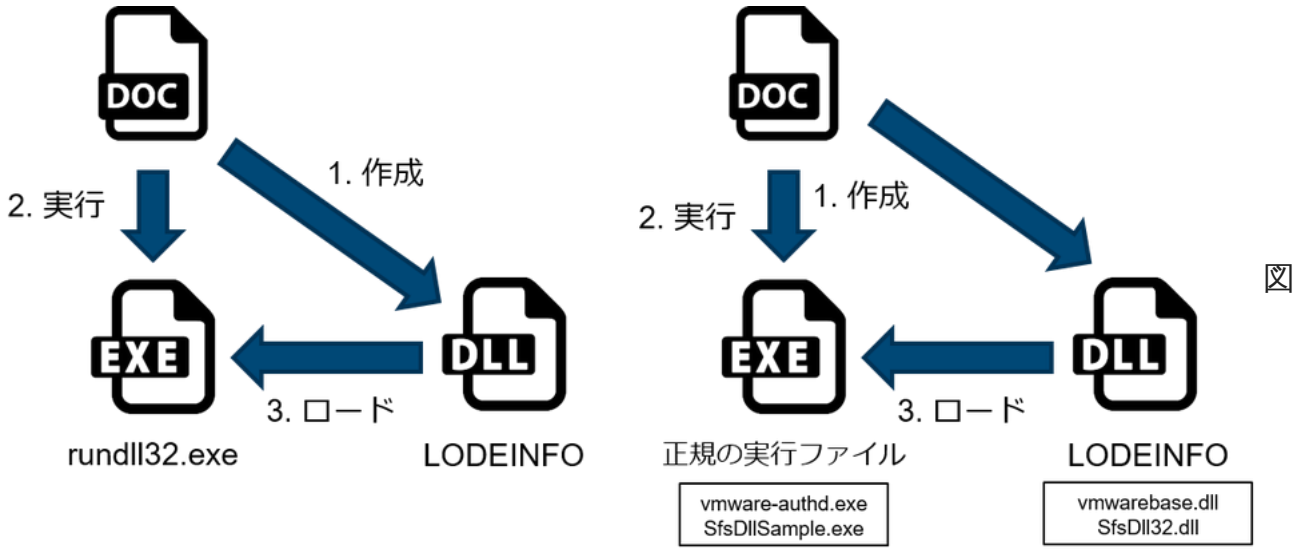


図7：起動方法の変化（左: rundll32.exeによる実行、右: DLLサイドローディングによる実行）

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 Safari/537.36
```

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/70.0.3538.102 Safari/537.36 Edge/18.18363
```

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。

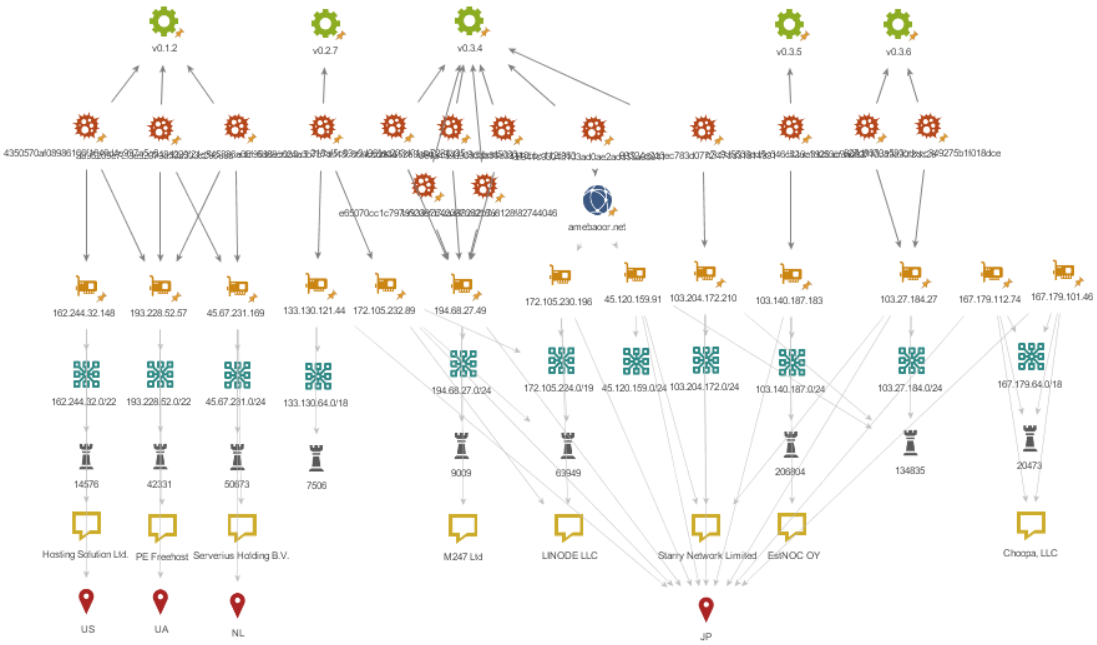


図 8：C&Cサーバのインフラ分布

おわりに

マルウエア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。 今後もこのマルウエ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.amebaoor.net

-
- メール

この記事の筆者

喜野 孝太(Kota Kino)

2019年8月より現職。主に、マルウェア分析・フォレンジック調査に従事。

このページは役に立ちましたか？

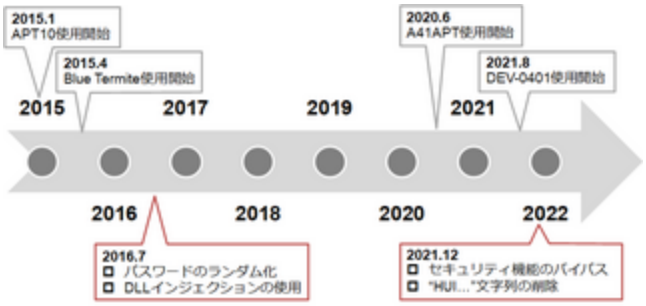
0人が「このページが役に立った」と言っています。

その他、ご意見・ご感想などございましたら、ご記入ください。

こちらはご意見・ご感想用のフォームです。各社製品については、各社へお問い合わせください。

javascriptを有効にすると、ご回答いただけます。 ありがとうございました。

関連記事

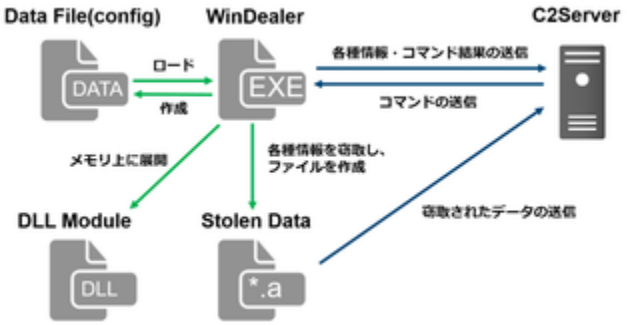
- 

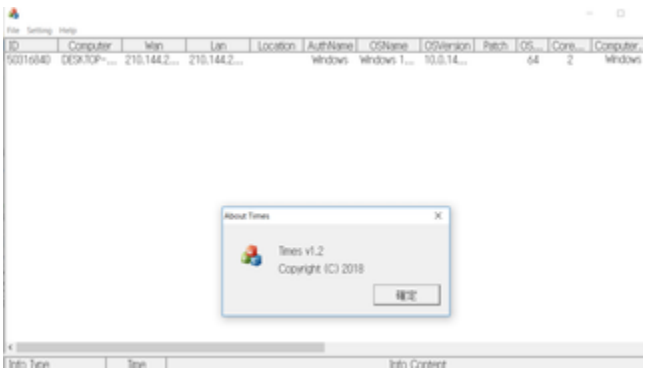
  HUI Loaderの分析

- 

  Anti-UPX Unpackingテクニック

- 

モバイル端末を狙うマルウェアへの対応FAQ
- 

攻撃グループLuoYuが使用するマルウェアWinDealer
- 
攻撃グループBlackTechが使用するマルウェアGh0stTimes

≪ 前へ
トップに戻る
次へ ≫