macnica networks

Japan Security Analyst Conference 2018

APTマルウェアに見る不易流行

25 Jan 2018

マクニカネットワークス株式会社
柳下 元
竹内 寛

不易流行

不易

変わることのない本質的なもの

流行

時代の変化に合わせて変化する新しいもの

攻撃者が使うツールであるマルウェアにも「不易」と「流行」部分がある。
マルウェアの不易な本質を把握し、それをインシデント対応で活用できるのではないか。

Agenda

- APT12, BlackBox Binder
- WINNTI
- まとめ

APT12, BlackBox Binder

APT12/IXESHE/Numbered Panda

- 2009年から活動が観測されている中国を拠点とする攻撃者グループ [1]
- 2012年 New York Timesへの侵害 [2]
- 台湾、日本で攻撃を観測 [3]
- Tools (Malwares)
  - IXESHE
  - Etumbot (RIPTIDE)
  - HIGHTIDE, THREEBYTE, WATERSPOUT
  - **BlackBox Binder (2015~)**
    - 配送: Spear Phishing. Outlook msg OLE (手法の解説記事 [4]) , パスワード付きzipファイル等
    - 一次マルウェア: 暗号化されたDLLファイルが埋め込まれており、それがメイン処理を行う。

BlackBox Binder 処理内容

検体1 (2015)

SHA256: bbd90f25d27df4a9936bcf726c12b46ffa8ad214dbd478e641c8e9609798f487

コンパイル日時: 2015/06/09 08:09:40 (UTC)

C2:

107.167.75[.]174

akashi[.]biz[.]tm

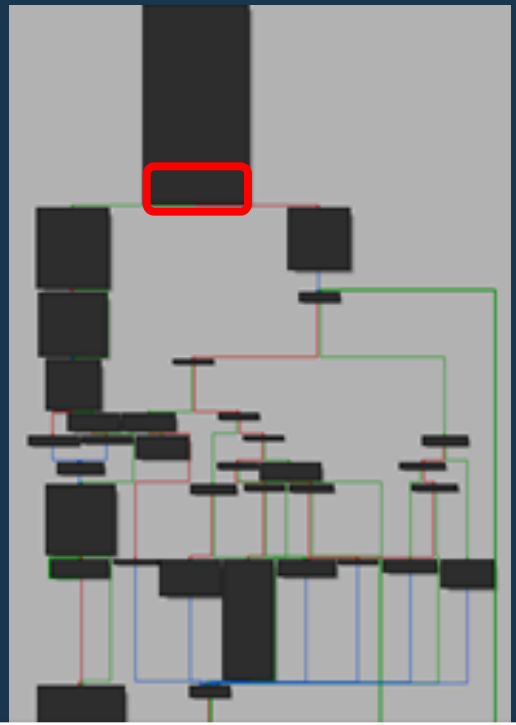
User-Agent (固定):

Mozilla/5.0 (Windows NT 6.1; rv:30.0) Gecko/20100101 Firefox/30.0

ミューテックス: 01-1536

Am I Dropper or Dropped? (ファイル名の確認)

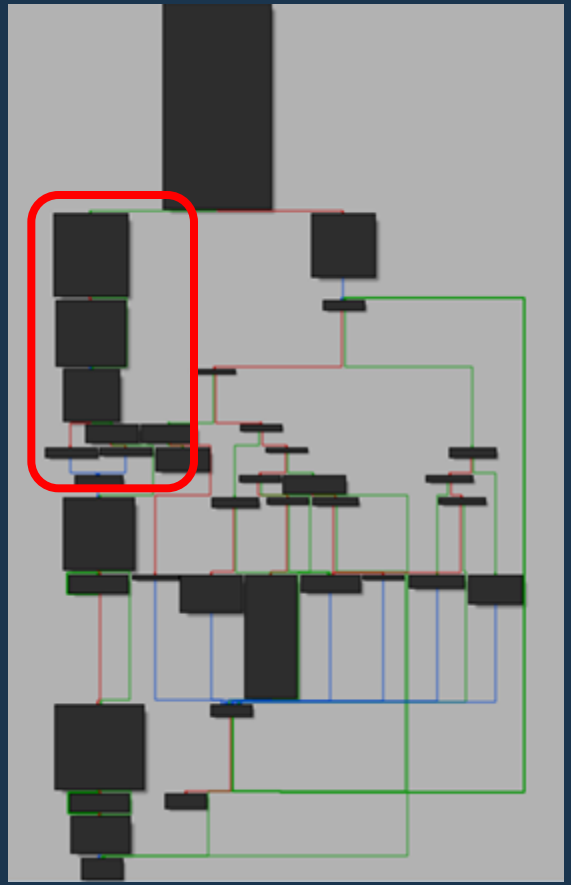
```
00401E77 6A 00                  push    0               ; hModule
00401E79 FF 15 38 90 40 00      call    ds:GetModuleFileNameA
00401E7F 8D 8C 24 BC 04 00 00   lea     ecx, [esp+6CCh+FileName]
00401E86 6A 5C                  push    5Ch             ; int
00401E88 51                     push    ecx             ; char *
00401E89 E8 E2 0B 00 00         call    _strrchr
00401E8E 8B E8                  mov     ebp, eax
00401E90 8D 54 24 2C            lea     edx, [esp+6CCh+var_6A0]
00401E94 45                     inc     ebp
00401E95 B3 72                  mov     bl, 72h
00401E97 52                     push    edx             ; char *
00401E98 55                     push    ebp             ; char *
00401E99 C6 44 24 34 63         mov     [esp+6D4h+var_6A0], 63h ; chrome.exe
00401E9E C6 44 24 35 68         mov     [esp+6D4h+var_69F], 68h
00401EA3 88 5C 24 36            mov     [esp+6D4h+var_69E], bl
00401EA7 C6 44 24 37 6F         mov     [esp+6D4h+var_69D], 6Fh
00401EAC C6 44 24 38 6D         mov     [esp+6D4h+var_69C], 6Dh
00401EB1 C6 44 24 39 65         mov     [esp+6D4h+var_69B], 65h
00401EB6 C6 44 24 3A 2E         mov     [esp+6D4h+var_69A], 2Eh
00401EBB C6 44 24 3B 65         mov     [esp+6D4h+var_699], 65h
00401EC0 C6 44 24 3C 78         mov     [esp+6D4h+var_698], 78h
00401EC5 C6 44 24 3D 65         mov     [esp+6D4h+var_697], 65h
00401ECA C6 44 24 3E 00         mov     [esp+6D4h+var_696], 0
00401ECF E8 7C 65 00 00         call    __strcmpi
00401ED4 83 C4 10               add     esp, 10h
00401ED7 85 C0                  test    eax, eax
00401ED9 0F 85 D1 02 00 00      jnz     loc_4021B0
```



ファイル名が、chrome.exeか？

デコイ

```
lea     edx, [esp+6CCh+var_104]
push    offset aS_S ; "%s.%s"
push    edx ; char *
call    _sprintf
lea     eax, [esp+6D4h+var_104]
lea     ecx, [esp+6D4h+var_310]
push    eax
push    ecx
lea     edx, [esp+6DCh+pszPath]
push    offset aEchoErrorSS ; "echo error >\"%s\\%s\""
push    edx ; char *
call    _sprintf
lea     eax, [esp+6E4h+pszPath]
push    eax
call    run_cmd_exe
mov     ecx, 41h
xor     eax, eax
```



ドロップ

```
push    eax ; char *
call    _sprintf
lea     ecx, [esp+6D4h+pszPath]
lea     edx, [esp+6D4h+Filename]
push    ecx
push    edx
lea     eax, [esp+6DCh+var_418]
push    offset aCopyBSCWindows ;
push    eax ; char *
call    _sprintf
lea     ecx, [esp+6E4h+var_418]
push    ecx
call    run_cmd_exe
add     esp, 24h
lea     edx, [esp+6C4h+pszPath]
push    edx ; pszPath
```

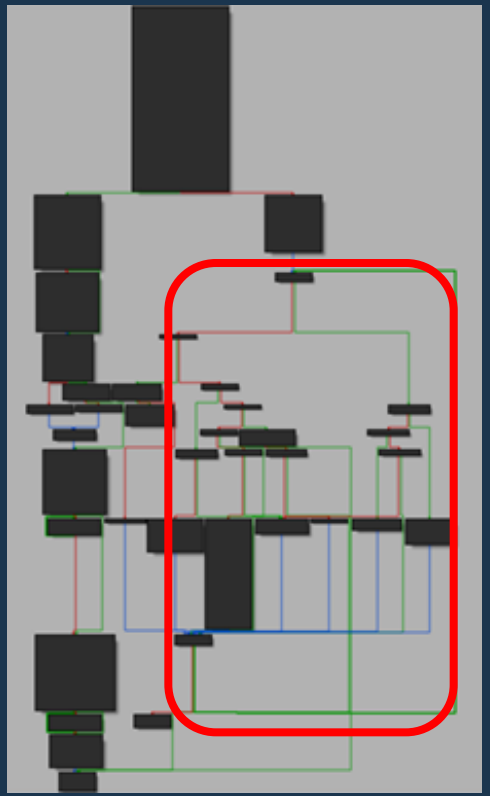
```
; char aS[]
aS              db '"%s"',0             ; DATA XREF: WinMain(x,x,x,x)+69
                align 4
; char aCopyBSCWindows[]
aCopyBSCWindows db 'copy /b "%s" + C:\windows\system32\cmd.exe "%s"',0
; CHAR Operation[]
Operation       db 'open',0             ; DATA XREF: WinMain(x,x,x,x)+5C
                align 4
; char aSS[]
aSS             db '%s\%s',0            ; DATA XREF: WinMain(x,x,x,x)+58
                                        ; WinMain(x,x,x,x)+615↑o
                align 4
```

cmd.exe(コマンドプロンプト)を自己ファイルに追加コピーして、
%temp%にchrome.exeとして保存・実行

アンチアナリシス: ループカウンターフロー

```
    v5 = 0;
    sub_401040(&v41, (int)&v72);
    sub_401190(&v72);
    work_cnt = 1;
    while ( 1 )
    {
      if ( work_cnt > 6903 )
      {
        if ( work_cnt == 7786 )
        {
          dword_410F34 = sub_401500((int)v5, &v11);
          ((void (__cdecl *)(HINSTANCE, signed int, void *, int))dword_410F34)(
            hInstance,
            101,
            &unk_410F38,
            dword_40DEB0);
          goto LABEL_23;
        }
        if ( work_cnt == 8077 )
        {
          sub_401B50(v5);
          goto LABEL_23;
        }
        if ( work_cnt != 9965 )
        {
LABEL_20:
          Sleep(1u);
          ((void (__cdecl *)(signe
          goto LABEL_23;
        }
      }
      else if ( work_cnt == 6903 )
      {
```

work_cnt	処理
2204	コンフィグの復号
3456	ミューテックスの作成
5412	埋め込まれたファイルの復号
7786	復号したコードの処理 パラメータ ・インスタンスハンドル ・コンフィグ: 107.167.75.174.. ・port 443(0x1BB)



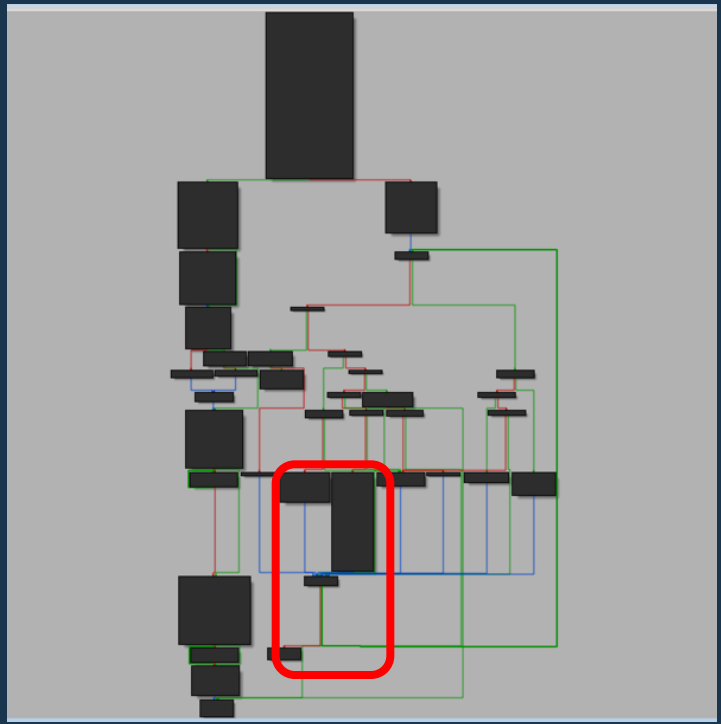
復号

```
00401000
00401000
00401000
00401000                    decrypt         proc near
00401000
00401000                    arg_0           = dword ptr  4
00401000                    arg_4           = dword ptr  8
00401000                    arg_8           = dword ptr  0Ch
00401000
00401000 8B 44 24 0C                        mov     eax, [esp+arg_8]
00401004 99                                 cdq
00401005 83 E2 07                           and     edx, 7
00401008 03 C2                              add     eax, edx
0040100A C1 F8 03                           sar     eax, 3
0040100D 85 C0                              test    eax, eax
0040100F 7E 25                              jle     short locret_401036
```

```
00401011 53                                 push    ebx
00401012 56                                 push    esi
00401013 8B 74 24 0C                        mov     esi, [esp+8+arg_0]
00401017 57                                 push    edi
00401018 8B 7C 24 14                        mov     edi, [esp+0Ch+arg_4]
0040101C 8B D8                              mov     ebx, eax
```

```
0040101E
0040101E                            loc_40101E:
0040101E 6A 01                              push    1
00401020 57                                 push    edi
00401021 56                                 push    esi
00401022 E8 59 00 00 00                     call    des_decrypt
```

DESとXORでコードを復号



AryanRAT

```
;
; Export directory for AryanRATDll.dll
;
                dd 0                    ; Characteristics
                dd 55769CF1h            ; TimeDateStamp: Tue Jun 09 07:59:45 2015
                dw 0                    ; MajorVersion
                dw 0                    ; MinorVersion
                dd rva aAryanratdll_dl  ; Name
                dd 1                    ; Base
```

リモートシェル、プロセス起動、
ファイル読み書き等の基本機能を有するRAT

検体2 (2016)

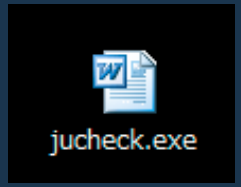
SHA256: 531f7f1ee71fc85d3674bc083e8ecc72108f1f816d82c0b7d2edd775173ea900

※参考 2016 類似検体ハッシュ: ab4f891ee2ae6ac7dce48c530b05aa4565dc38c79ff5dfa00913898167b46095

コンパイル日時: 2016/01/26 06:41:21 (UTC)

C2:

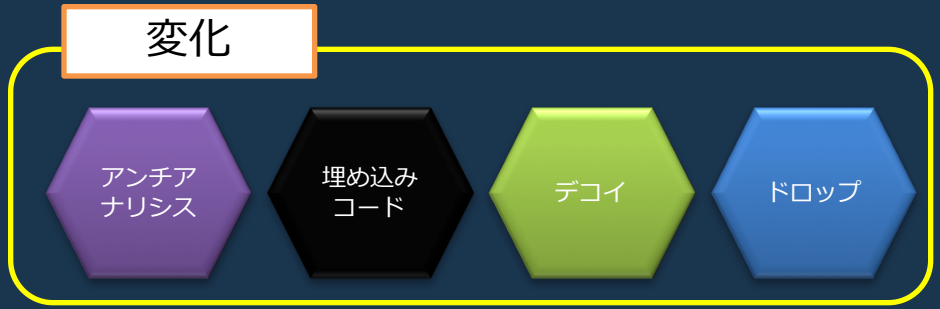
software[.]jelastic.servint[.]net (HTTPS/443)



User-Agent (固定):

Mozilla/5.0 (Windows NT 6.1; rv:30.0) Gecko/20100101 Firefox/30.0

ミューテックス: shownews.php



BlackBox

```
.rdata:002348C0 ;
.rdata:002348C0 ; Export directory for BlackBox.dll
.rdata:002348C0 ;
.rdata:002348C0                 dd 0                    ; Characteristics
.rdata:002348C4                 dd 56A7135Bh            ; TimeDateStamp: Tue Jan 26 06:34:03 2016
.rdata:002348C8                 dw 0                    ; MajorVersion
.rdata:002348CA                 dw 0                    ; MinorVersion
.rdata:002348CC                 dd rva aBlackbox_dll    ; Name
.rdata:002348D0                 dd 1                    ; Base
.rdata:002348D4                 dd 2                    ; NumberOfFunctions
.rdata:002348D8                 dd 2                    ; NumberOfNames
.rdata:002348DC                 dd rva off_2348E8       ; AddressOfFunctions
.rdata:002348E0                 dd rva off_2348F0       ; AddressOfNames
.rdata:002348E4                 dd rva word_2348F8      ; AddressOfNameOrdinals
```

エクスポート関数	処理内容
DriverCreate	デコイの表示。別ファイルをドロップ・実行
DriverProc	C2通信処理

デコイ、ドロップ(DriverCreate)

```
    Sleep(0xAu);
  new_allocate_memory = operator new(0xA00001u);
  write_count = 3;
  do
  {
    add_data_to_jucheck(new_allocate_memory, 0xA00000u, &pszPath, 0, 0);
    Sleep(0xAu);
    --write_count;
  }
  while ( write_count );
```

%temp%にjucheck.exeとして保存
0xA00000 x 3 (30MB)バイト追加

C2通信処理 (DriverProc)

```
00401DBC C6 44 24 57 00          mov     [esp+188h+var_131], 0
00401DC1 C6 44 24 40 44          mov     [esp+188h+var_148], 44h ; DriverProc
00401DC6 88 44 24 41             mov     [esp+188h+var_147], al
00401DCA C6 44 24 43 76          mov     [esp+188h+var_145], 76h
00401DCF 88 5C 24 44             mov     [esp+188h+var_144], bl
00401DD3 88 44 24 45             mov     [esp+188h+var_143], al
00401DD7 C6 44 24 46 50          mov     [esp+188h+var_142], 50h
00401DDC 88 44 24 47             mov     [esp+188h+var_141], al
00401DE0 C6 44 24 48 6F          mov     [esp+188h+var_140], 6Fh
00401DE5 C6 44 24 49 63          mov     [esp+188h+var_13F], 63h
00401DEA C6 44 24 4A 00          mov     [esp+188h+var_13E], 0
00401DEF E8 0C F7 FF FF          call    sub_401500
00401DF4 8B 0D B0 98 40 00       mov     ecx, dword_4098B0
00401DFA 8D 54 24 4C             lea     edx, [esp+188h+var_13C]
00401DFE 52                      push    edx
00401DFF 8D 54 24 1C             lea     edx, [esp+18Ch+var_170]
00401E03 51                      push    ecx
00401E04 A3 00 A3 40 00          mov     dword_40A300, eax
00401E09 52                      push    edx
00401E0A 8B 94 24 98 01 00 00    mov     edx, [esp+194h+hInstance]
00401E11 8D 4C 24 64             lea     ecx, [esp+194h+var_130]
00401E15 51                      push    ecx
00401E16 52                      push    edx
00401E17 FF D0                   call    eax             ; BlackBox DriverProc
00401E19 56                      push    esi             ; lpAddress
00401E1A E8 31 FD FF FF          call    sub_401B50
00401E1F 83 C4 20                add     esp, 20h
00401E22 33 C0                   xor     eax, eax
00401E24 5F                      pop     edi
00401E25 5E                      pop     esi
```

C2よりダウンロードしたデータを復号しメモリ上に展開し実行。このDLL自体には、RAT機能はない(Loader)

検体3 (2017)

SHA256: 42da51b69bd6625244921a4eef9a2a10153e012a3213e8e9877cf831aea3eced

コンパイル日時: 2017/08/17 00:35:55 (UTC)

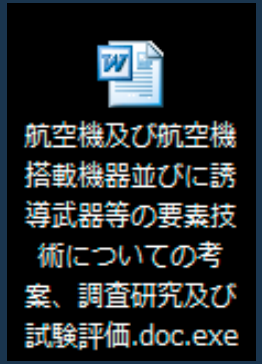
C2:

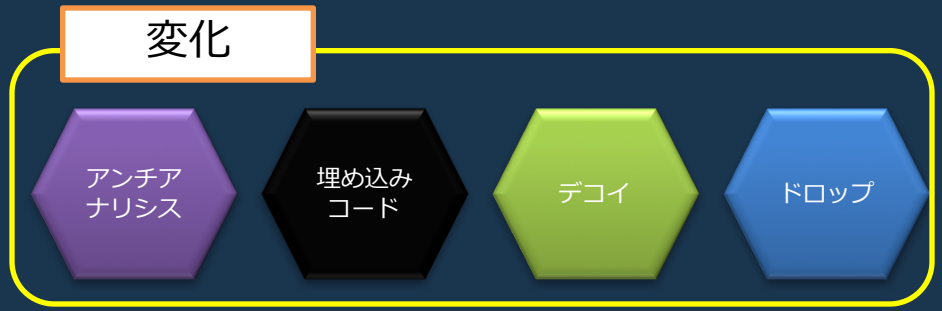
bsksac[.]au-syd.mybluemix[.]net (HTTPS/443)

User-Agent (固定):

Chrome/27.0.1453.94 Safari/537.36

ミューテックス: /jquery.php

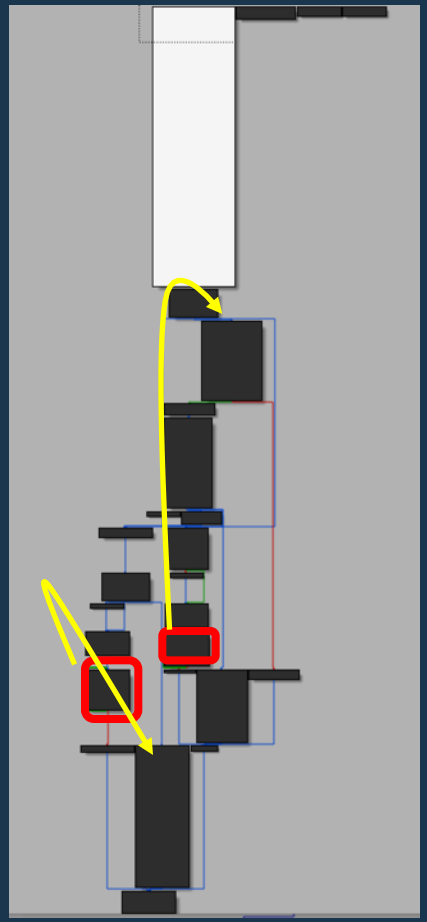




アンチアナリシス: 例外処理

```
quotient = 0;
do
{
  quotient = 100 / (rand() % 101 % 3);
  tick_count = GetTickCount();
  Sleep(300u);
}
while ( GetTickCount() - tick_count >= 300 );
```

アンチサンドボックス解析



BlackBox 2017

```
;
; Export directory for BlackBox.dll
;
                dd 0                    ; Characteristics
                dd 598ABD87h            ; TimeDateStamp: Wed Aug 09 07:45:11 2017
                dw 0                    ; MajorVersion
                dw 0                    ; MinorVersion
                dd rva aBlackbox_dll    ; Name
                dd 1                    ; Base
                dd 3                    ; NumberOfFunctions
                dd 3                    ; NumberOfNames
                dd rva off_248E8        ; AddressOfFunctions
                dd rva off_248F4        ; AddressOfNames
                dd rva word_24900       ; AddressOfNameOrdinals
;
```

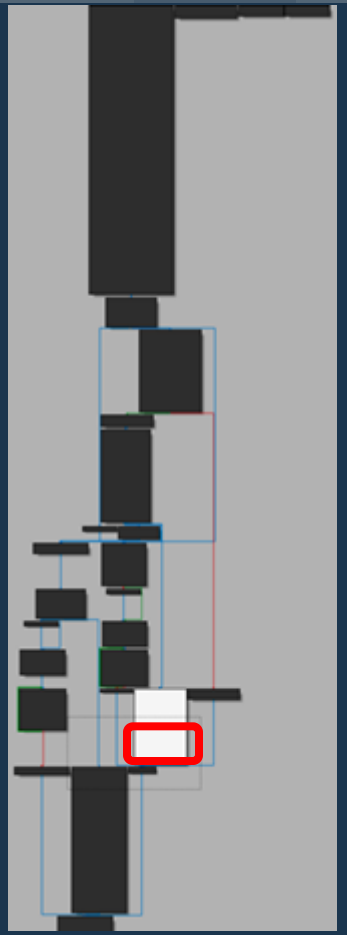
エクスポート関数	処理内容
DriverCreate	デコイの表示。別ファイルをドロップ
DriverOpen	**ドロップしたファイルの実行**
DriverProc	C2通信処理

デコイ、ドロップ(DriverCreate, DriverOpen)

```
0040225F 8B B5 18 FC FF FF      mov     esi, [ebp+var_3E8]
00402265 56                     push    esi
00402266 E8 A5 F2 FF FF         call    load_library2
0040226B A3 AC 1B 41 00         mov     dword_411BAC, eax
00402270 68 58 05 41 00         push    offset unk_410558
00402275 8D 4D D0               lea     ecx, [ebp+var_30]
00402278 51                     push    ecx
00402279 FF D0                  call    eax             ; Blackbox DriverCreate
0040227B C6 85 F4 FC FF FF 44   mov     [ebp+var_30C], 'D'
00402282 C6 85 F5 FC FF FF 72   mov     [ebp+var_30B], 'r'
00402289 C6 85 F6 FC FF FF 69   mov     [ebp+
00402290 C6 85 F7 FC FF FF 76   mov     [ebp+
00402297 88 9D F8 FC FF FF      mov     [ebp+
0040229D C6 85 F9 FC FF FF 72   mov     [ebp+
004022A4 C6 85 FA FC FF FF 4F   mov     [ebp+
004022AB C6 85 FB FC FF FF 70   mov     [ebp+
004022B2 88 9D FC FC FF FF      mov     [ebp+var_304], bl
004022B8 C6 85 FD FC FF FF 6E   mov     [ebp+var_303], 'n'
004022BF C6 85 FE FC FF FF 00   mov     [ebp+var_302], 0
004022C6 57                     push    edi
004022C7 8D 95 F4 FC FF FF      lea     edx, [ebp+var_30C]
004022CD 52                     push    edx
004022CE 56                     push    esi
004022CF E8 3C F2 FF FF         call    load_library2
004022D4 A3 A8 1B 41 00         mov     dword_411BA8, eax
004022D9 8D 4D D0               lea     ecx, [ebp+var_30]
004022DC 51                     push    ecx             ; mshta.exe
004022DD FF D0                  call    eax             ; Blackbox DriverOpen
004022DF 57                     push    edi
004022E0 56                     push    esi
004022E1 E8 5A F2 FF FF         call    sub_
004022E6 56                     push    esi
004022E7 E8 1F 03 00 00         call    ???@
004022EC 83 C4 30               add     esp, 30h
```

%temp%にmshta.exeとして保存
0xA00000 x3 (30MB)バイト付与

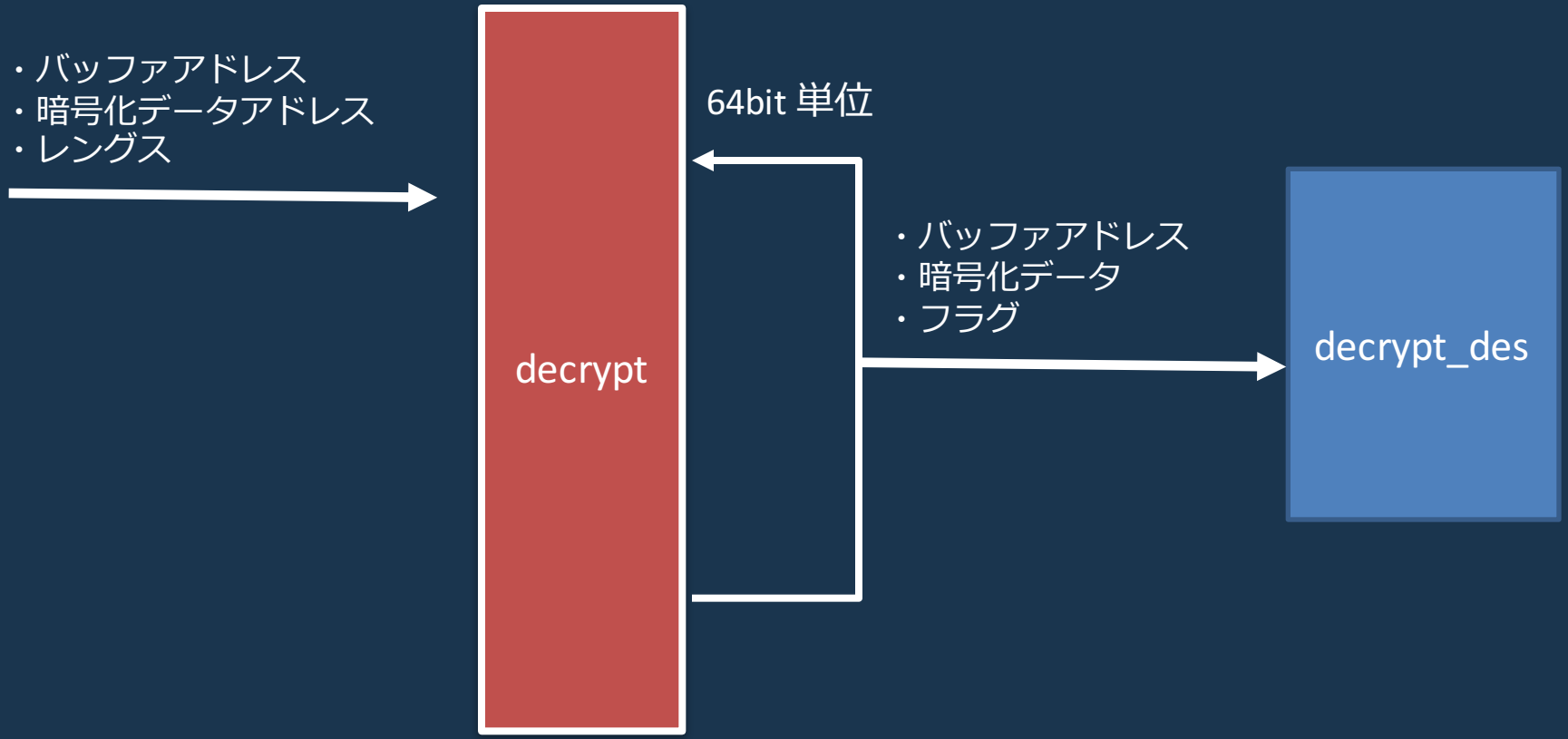
mshta.exeを実行



BlackBox Binderに見る不易

機能	検体1 (2015)	検体2 (2016)	検体3 (2017)
ファイル名の確認	ファイル名確認 chrome.exe	ファイル名確認 jucheck.exe	ファイル名確認 mshta.exe
ドロップ	cmd.exeを追加	動的に確保したメモリ データを追加	動的に確保したメモリ データを追加
復号	DES + XOR ベース	DES + XOR ベース	DES + XOR ベース
アンチアナリシス	ループカウンタフロー	なし	意図的な例外発生
埋め込まれている ファイル	AryanRATDll (RAT)	BlackBox (Loader)	BlackBox (Loader)

BlackBox Binderに見る不易: 復号処理



BlackBox Binder に見る不易: YARA

```
8B 44 24 0C                     mov     eax, [esp+arg_8]
99                              cdq
83 E2 07                        and     edx, 7
03 C2                           add     eax, edx
C1 F8 03                        sar     eax, 3
85 C0                           test    eax, eax
7E 25                           jle     short locret_401096
53                              push    ebx
56                              push    esi
8B 74 24 0C                     mov     esi, [esp+8+arg_0]
57                              push    edi
8B 7C 24 14                     mov     edi, [esp+0Ch+arg_4]
8B D8                           mov     ebx, eax

                loc_40107E:                             ; CODE
6A 01                           push    1
57                              push    edi
56                              push    esi
E8 59 00 00 00                  call    decrypt_des
83 C4 0C                        add     esp, 0Ch
83 C6 08                        add     esi, 8
83 C7 08                        add     edi, 8
4B                              dec     ebx
75 EB                           jnz     short loc_40107E
5F                              pop     edi
5E                              pop     esi
5B                              pop     ebx

                locret_401096:                          ; CODE
C3                              retn
```

```
rule MNC_APT_2017_BLACKBOX_BINDER
{
    strings:
        $api1 = "LoadLibraryA" fullword
        $api2 = "GetProcAddress" fullword
        $decrypt_seq = {8B 44 24 0C 99 83 E2 07 03 C2 C1 F8 03
85 C0 7E 25 53 56 8B 74 24 0C 57 8B 7C 24 14 8B D8 6A 01 57 56
E8 ?? ?? ?? ?? 83 C4 0C 83 C6 08 83 C7 08 4B 75 EB 5F 5E 5B C3}

    condition:
        (uint16(0) == 0x5A4D and
         uint32(uint32(0x3C)) == 0x00004550) and
         filesize < 35MB and
         all of them
}
```

WINNTI

本日お話ししたいこと

- 2016年と2017年に観測したWINNTI
- WINNTIの流行(変化している点)と不易(変化しない点)

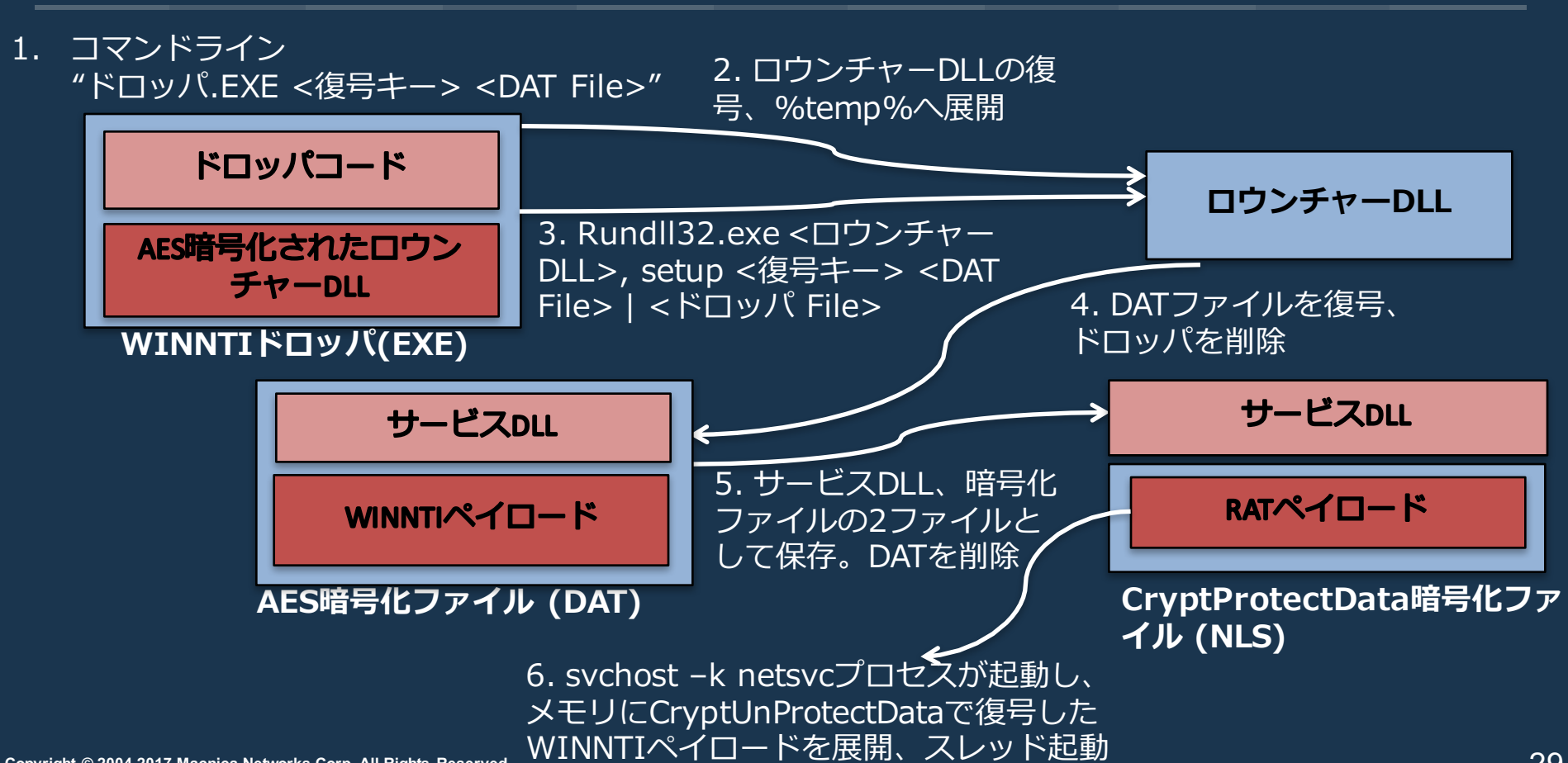
WINNTI 概要

- 標的
  - ゲーム 2013 [5]
  - メディア 2015 [6]
  - 化学、eコマース、投資ファーム、エレクトロニクス、テレコム 2016 [7]
- マルウェア
  - WINNTI, PlusUnit, ZxShell, Kriskynote, Kernel Rootkit
- 攻撃グループ名称
  - WINNTI, (Axiom) [8]

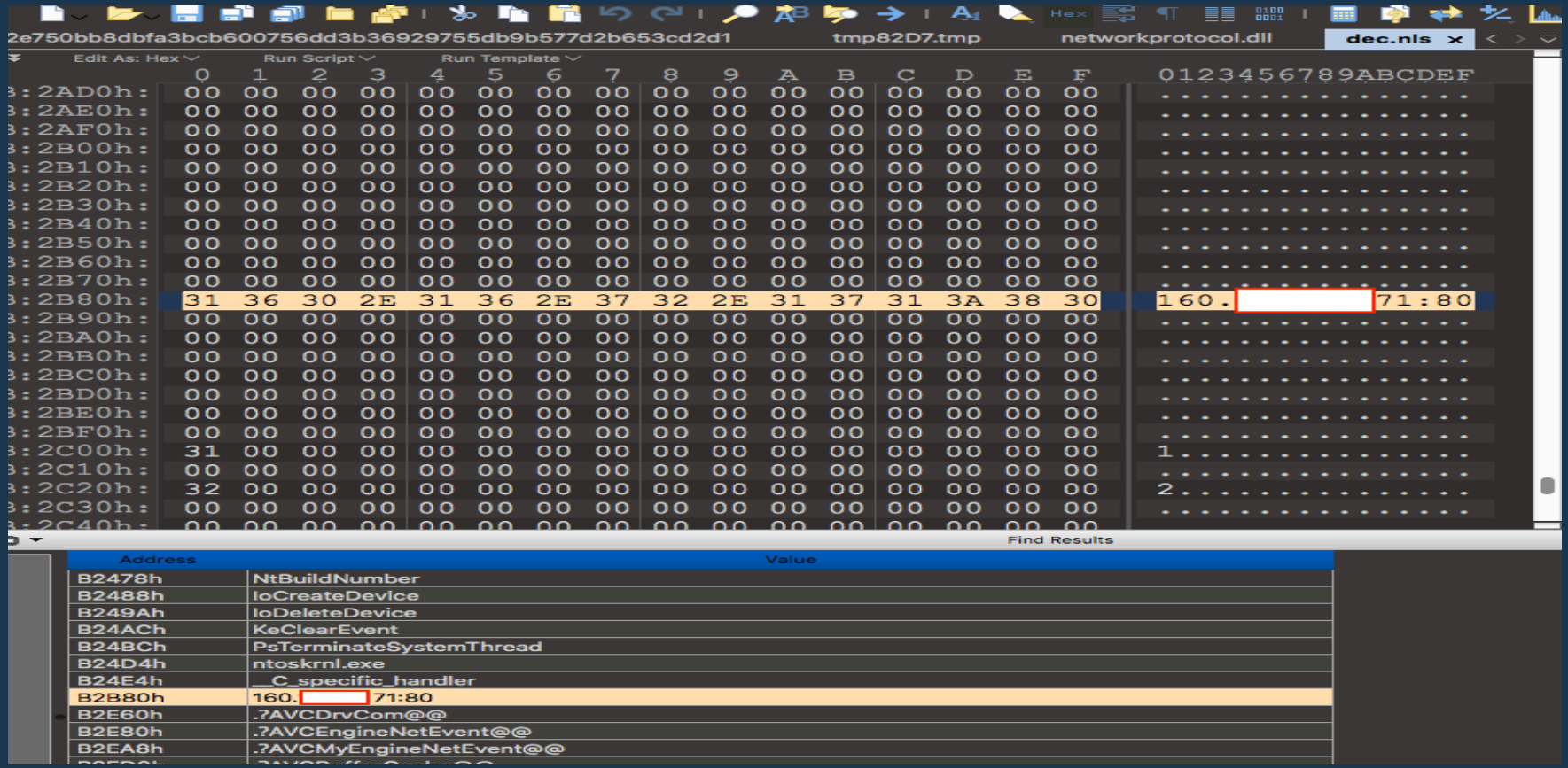
WINNTI (2016検体)

- 標的
  - エレクトロニクス
- 検体
  - 64bit ドロッパ
  - SHA256:
    5ebf39d614c22e750bb8dbfa3bcb600756dd3b36929755db9b577d2b653cd2d1

WINNTI (2016検体動作)



WINNTI (2016検体 C2 CryptUnprotectData [9])



WINNTI (2016検体C2トラフィック)

```
POST / HTTP/1.1
Content-Type: application/octet-stream
Host: 160.      .171
Content-Length: 52
Inflate-Length: 52
Cache-Control: no-cache
Connection: Keep-Alive
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)

....2...2....NX..>...+......B...3...0.g =0.So#\. "'.HTTP/1.1 200 OK
Server: nginx
Date: Wed, 05          10:43:02 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding
Vary: Accept-Encoding
```

WINNTI (2016検体 Kernel Driver)

Windows Filtering Platform (WFP)ベース FWPS, FWPM

WINNTI Kernel Driver

```
v0 = DeviceObject;
v14 = 0;
v15 = 0;
v3 = 0;
memset(&Dst, 0, 0x2Cu);
v5 = 1;
v1 = FwpmEngineOpen0(0, 10, 0, &v3, &fwpm_filter0);
if ( v1 < 0 )
  goto LABEL_16;
v14 = 1;
v1 = FwpmTransactionBegin0(fwpm_filter0,  0);
if ( v1 < 0 )
  goto LABEL_16;
v15 = 1;
memset(&v6, 0, 0x2Cu);
v6 = -1318876260;
v8 = -1366029130;
v9 = -1101491021;
v7 = 1178164397;
v10 = L"Transport-Data  Proxy Sub-Layer";
v11 = L"Sub-Layer for use by Transport-Data  Proxy callouts";
v12 = 0;
v13 = 0;
v1 = FwpmSubLayerAdd0(fwpm_filter0,  &v6, 0);
if ( v1 < 0
```

WDKトラフィックキャプチャ

```
Tl_drv.c
541
542     Callouts
543
544 -- */
545 {
546     NTSTATUS status = STATUS_SUCCESS;
547     FWPM_SUBLAYER0 TLInspectSubLayer;
548
549     BOOLEAN engineOpened = FALSE;
550     BOOLEAN inTransaction = FALSE;
551
552     FWPM_SESSION0 session = {0};
553
554     session.flags = FWPM_SESSION_FLAG_DYNAMIC;
555
556     status = FwpmEngineOpen0(
557                 NULL,
558                 RPC_C_AUTHN_WINNT,
559                 NULL,
560                 &session,
561                 &gEngineHandle
562                 );
563     if (!NT_SUCCESS(status))
564     {
565         goto Exit;
566     }
567     engineOpened = TRUE;
568
569     status = FwpmTransactionBegin0(gEngineHandle, 0);
570     if (!NT_SUCCESS(status))
571     {
572         goto Exit;
573     }
574     inTransaction = TRUE;
575
576     RtlZeroMemory(&TLInspectSubLayer, sizeof(FWPM_SUBLAYER0));
577
578     TLInspectSubLayer.subLayerKey = TL_INSPECT_SUBLAYER;
579     TLInspectSubLayer.displayData.name = L"Transport Inspect Sub-Layer";
580     TLInspectSubLayer.displayData.description =
581         L"Sub-Layer for use by Transport Inspect callouts";
582     TLInspectSubLayer.flags = 0;
583     TLInspectSubLayer.weight = 0; // must be less than the weight of
584                                   // FWPM_SUBLAYER_UNIVERSAL to be
585                                   // compatible with Vista's IpSec
586                                   // implementation.
587
588     status = FwpmSubLayerAdd0(gEngineHandle, &TLInspectSubLayer, NULL);
589     if (!NT_SUCCESS(status))
```

WINNTI (2016検体 Kernel Driver)

トラフィックの改変 (秘匿)

FwpsInjectTransportReceive(Send)Async0

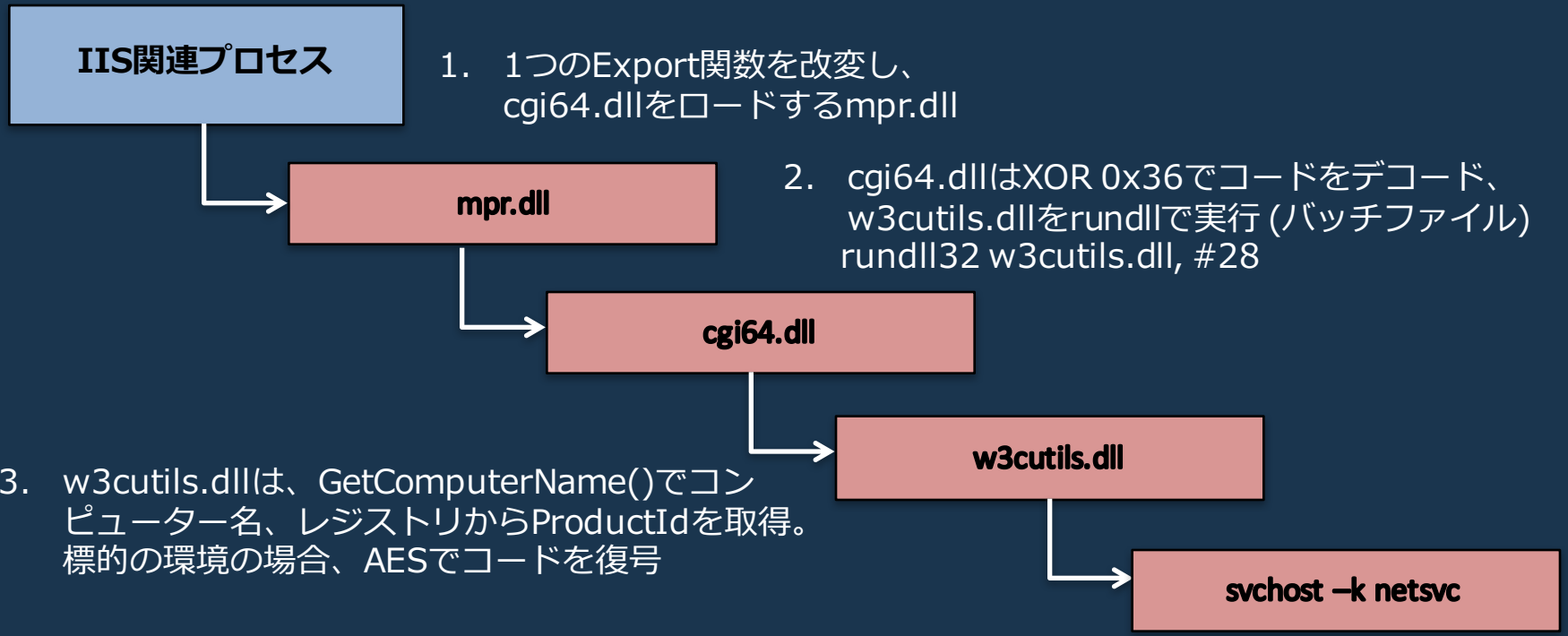
```
sub_401240();
ThreadHandle = 0;
v2 = PsCreateSystemThread(&ThreadHandle, 0x1FFFFFu, 0, 0, 0, StartRoutine, 0);
if ( v2 >= 0 )
{
 v2 = ObReferenceObjectByHandle(ThreadHandle, 0, 0, 0, &dword_40881C, 0);
 ZwClose(ThreadHandle);
 if ( v2 >= 0 )
 {
  ThreadHandle = 0;
  v2 = PsCreateSystemThread(&ThreadHandle, 0x1FFFFFu, 0, 0, 0, sub_401480, 0);
  if ( v2 >= 0 )
  {
   v2 = ObReferenceObjectByHandle(ThreadHandle, 0, 0, 0, &dword_40820C, 0);
   ZwClose(ThreadHandle);
   if ( v2 >= 0 )
   {
    DriverObject->DriverUnload = (PDRIVER_UNLOAD)sub_404E00;
    return v2;
```

```
KeAcquireInStackQueuedSpinLock(&dword_408258, &LockHandle);
  injdest01 = P;
  P = *(PVOID *)P;
  *((_DWORD *)P + 1) = &P;
  KeReleaseInStackQueuedSpinLock(&LockHandle);
  if ( injdest01[2] )
  {
    v3 = FwpsInjectTransportReceiveAsync0(
        injhandle01,
        0,
        0,
        0,
        2,
        injdest01[11],
        injdest01[27],
        injdest01[28],
        injdest01[12],
        sub_405300,
        injdest01);
    if ( v3 < 0 )
      FwpsFreeCloneNetBufferList0(injdest01[12], 0);
    v2 = v3;
    injdest01[12] = 0;
```

WINNTI (2017検体)

- 標的
  - 化学
- 検体
  - 64bit ローンチャDLL
  - SHA256:
    d3c6d513ca7e649287bd63cef865419b2b4a0ef9e48664e2905f700d0db9587
    02df9535c6fcfdeeec199135b9f7a0c417bded67e5d28aa0f42827b672af4194
    d16e01dbb894a40ff0c8b3f6b25a41d190db03c15c432ac50c3784a9880d376e
  - C2通信なく、別の感染端末から操作

WINNTI (2017検体動作)



1. 1つのExport関数を改変し、cgi64.dllをロードするmpr.dll
2. cgi64.dllはXOR 0x36でコードをデコード、w3cutils.dllをrundllで実行 (バッチファイル)
   rundll32 w3cutils.dll, #28
3. w3cutils.dllは、GetComputerName()でコンピューター名、レジストリからProductIdを取得。標的の環境の場合、AESでコードを復号
4. 復号したコードをsvchostにインジェクション、起動

WINNTI (2017検体Sysmon チェック)

- Sysmon.exeのプロセス有無
- プロセス有の場合、Sysmonイベント停止？

```
__int64 SysmonChk_OpenProc_WriteF__()
{
  unsigned int v0; // ebx
  __int64 v1; // rbx

  if ( (unsigned int)GetVersionEX__() < 4 )
    return 0i64;
  v0 = Sysmoncheck__((__int64)"sysmon.exe", 0);
  if ( v0 )
  {
    if ( !(unsigned int)OpenEventCloseHandle__((__int64)"Global¥¥BFE_Notify_Event_{65a097fe-6102-446a-9f9c-55dfc3f411016}") )
      WriteFBySwith_OpenProc_CreateThread____(v0, (__int64)qword_225BC80, (unsigned __int64)&unk_16000, 0i64, 0, 1u);
    v1 = CreateEvent1__((__int64)"Global¥¥BFE_Notify_Event_{65a097fe-6102-446a-9f9c-55dfc3f411014}");
    kernel32_Sleep(5000i64);
    if ( v1 )
      ((void (__fastcall *)(__int64))kernel32_CloseHandle)(v1);
  }
  return 0i64;
}
```

```
__int64 __fastcall OpenEventCloseHandle__(__int64 BFE_Event__)
{
  __int64 handle0; // rax

  handle0 = kernel32_OpenEventA(1i64, 0i64, BFE_Event__);
  if ( handle0 )
  {
    ((void (__fastcall *)(__int64))kernel32_CloseHandle)(handle0);
    handle0 = 1i64;
  }
  return handle0;
}
```

WINNTI (2017検体 Kernel Driver)

- ラテラル操作用 (C2通信なしにRAT操作)
- ¥¥Device¥¥NullをRATが作成、ドライバがポインタを利用

WINNTI RAT

```
v5 = a3;
v6 = a1;
v7 = a4;
v8 = a2;
nullhandl0 = CreateFileA("¥¥¥¥.¥¥Nul", 3221225472i64, 3i64, 0i64,
if ( nullhandl0 == -1 )
  return 0i64;
if ( v5 )
{
  LODWORD(v11) = v7;
  result = DeviceIoControl(nullhandl0, &DeviceIoCtrl_Arg02, v6, v8, v
}
else
{
  LODWORD(v11) = 0;
  result = DeviceIoControl(nullhandl0, &unk_156003, v6, v8, 0i64, v1
}
```

Kernel Driver

```
...
{
  RtlInitUnicodeString(&DestinationString, L"¥¥Device¥¥Null");

  v1 = IoGetDeviceObjectPointer(&DestinationString, 1u, (PFILE_OBJECT *)&Object,
&DeviceObject);

  if ( (v1 & 0x80000000) == 0 )
  {
    DrvObj0 = DeviceObject->DriverObject;
    if ( DrvObj0 )
    {
      qword_14000A228 = (__int64 (*)(void))DrvObj0->MajorFunction[14];
      DrvObj0->MajorFunction[14] =
(PDRIVER_DISPATCH)Probe4ReadWrite_IofCompleteIRP;
      result = v1;
    }
    else
{
...
```

WINNTI (2017検体 Kernel Driver)

NdisSendNetBufferLists (Buffer size 1024)

```
v7 = NDIS_HANDL01;
v6->Next = 0i64;
LODWORD(Irp) = 0;
v8 = NdisAllocateNetBufferAndNetBufferList(v7, 0i64, 0i64, v6, Irp, v2);// Alloc and Init NET_BUFFER_LIST

if ( v8 )
{
  if ( VirtualAddress )
  {
    memmove(VirtualAddress, v3, v2);
    v9 = NDis_Dst_TargetAdapter;
    *(_QWORD *)(v8 + 120) = NDis_Dst_TargetAdapter;

    NdisSendNetBufferLists(v9, v8, 0i64, 1i64);// target adapter, NET_BUFF_LIST, PortNum, SendFlag

    if ( (unsigned int)dword_14000A060 > 0 && (unsigned __int16)word_14000A064 > 0u )
      SpinLockRelated__01_0(v3, (unsigned int)v2, 0);
    goto LABEL_18;
  }
  goto LABEL_11;
}
....
```

WINNTI (2017検体 Kernel Driver)

NdisCopyFromNetBufferToNetBuffer

```
while ( 1 )
{
  v15 = *((_DWORD *)v12 + 6);
  if ( v15 - 1 <= 0x63F )
  {
    NET_BUFF_struct01 = (_DWORD *)NdisAllocateNetBufferMdlAndData(NDIS_HANDLE01);
    v17 = NET_BUFF_struct01;
    if ( !NET_BUFF_struct01 )
      goto LABEL_20;

    NET_BUFF_struct01[6] = 0;
    NET_BUFF_struct01[10] = 0;
    NET_BUFF_struct01[4] = 0;
    LODWORD(v22) = 0;

    if ( (signed int)NdisCopyFromNetBufferToNetBuffer(NET_BUFF_struct01, 0i64, v15, v12, v22, &v27) < 0
      || v15 != v27 )                 // Dst, DstOffst, Bytes, Src, SrcOffst, Bytes
    {
...
```

WINNTI (不易の部分)

スレッド先頭箇所の処理

CreateEvent, GetWindowsVersion, OpenWindowStationA (if Old Windows)

WINNTI RAT 2016

```
...
char v31; // [rsp+17Ah] [rbp+7Ah]
char v32; // [rsp+250h] [rbp+150h
char v33; // [rsp+251h] [rbp+151h

if ( (unsigned int)CreateEvent__((__int64)"Sys_Win32_Event") )
{
  Kernel32_SetEvent(qword_1800B7270);
  kernel32_Sleep(10000i64);
  kernel32_WaitForSingleObject(qword_1800B7270, 0xFFFFFFFFi64);
}
dword_1800B721C = 0;
v0 = Get_VerifVersionInfo__();

if ( v0 <= 3 )
{
  user32_GetProcessWindowStation();
  v1 = user32_OpenWindowStationA("winsta0", 0i64, 0x2000000i64);
  if ( v1 )
    user32_SetProcessWindowStation(v1);
}
...
```

WINNTI RAT 2017

```
char v24; // [rsp+75h] [rbp-Bh]

if ( (unsigned int)GetVersionEX__() <= 2 )
{
  user32_GetProcessWindowStation();
  v0 = user32_OpenWindowStationA("winsta0", 0i64, 0x2000000i64);
  if ( v0 )
    user32_SetProcessWindowStation(v0);
  }
  v5 = 'bolG'; // EventName
  v6 = 'B¥¥la'; v7 = 'N_EF'; v8 = 'fito';
  v9 = 'vE_y'; v10 = '_tne'; v11 = '0D7{';
  v12 = '3AF0'; v13 = 'BF-C'; v14 = '4-CD';
  v15 = '-D8A'; v16 = 'BEEA'; v17 = '5F3-';
  v18 = '84A5'; v19 = '2D09'; v20 = 32065; v21 = 0;
  if ( (unsigned int)CreateEvent__((__int64)&v5) )
  {
    kernel32_SetEvent(0i64);
    kernel32_Sleep(300i64);
    kernel32_WaitForSingleObject(0i64, 0xFFFFFFFFi64);
  }
```

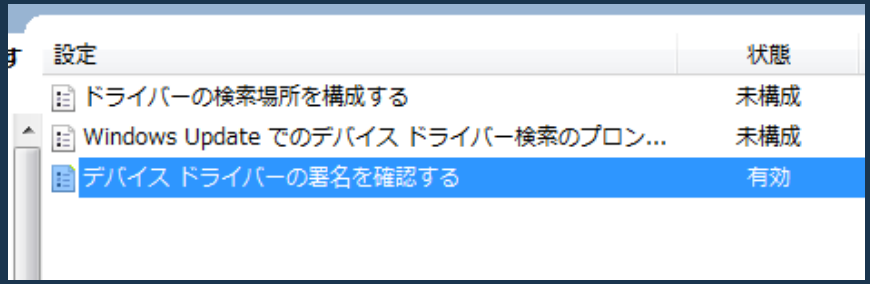
WINNTI (不易の部分)

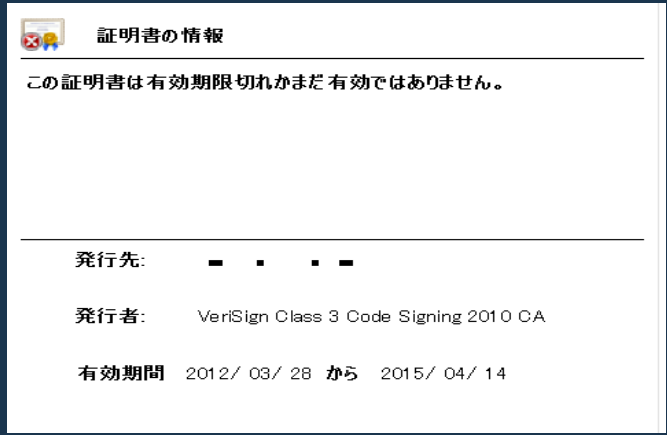
- カーネルドライバのインストール方法
- ファイル保存先 (すぐ削除) : ¥WINDOWS¥TEMP¥XXXX.tmp

```
if ( v2 > 3 )
 {
   v4 = 40960; //Size
   v5 = &MZ01;   // Driver for 7 or above x64
 }
 else
 {
   v4 = 22016; //Size
   v5 = &MZ02; // Driver for 2003 or below
 }
 Create_WriteFile__(v5, v4, v9);
 Load_Driver__((__int64)v9, (__int64)&v7); // RegCreateKey(%Service), NtLoadDriver(), RegDeleteKey()
 kernel32_SetFileAttributesA(v9, 128);
 kernel32_DeleteFileA(v9);
```

WINNTI (不易の部分)

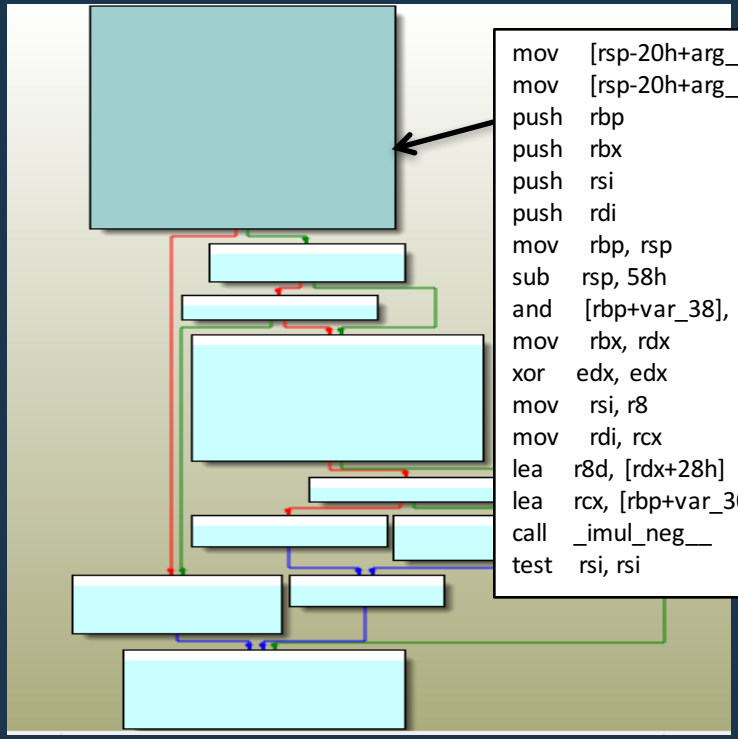
- RegCreateKey(サービス), LoadDriver()によるサービス作成・開始
  - sc queryから秘匿　(SCMを経由しない)
- DeleteFile()でドライバファイルの削除
  - AVに検知される確率の低下
  - driverqueryから秘匿
- 中華民國某企業の失効したデジタル署名
  - 64bit OS – 失効したデジタル署名でも動作
  - ドライバの署名が失効したものかは確認しない？
- その他
  - GMERで不検出





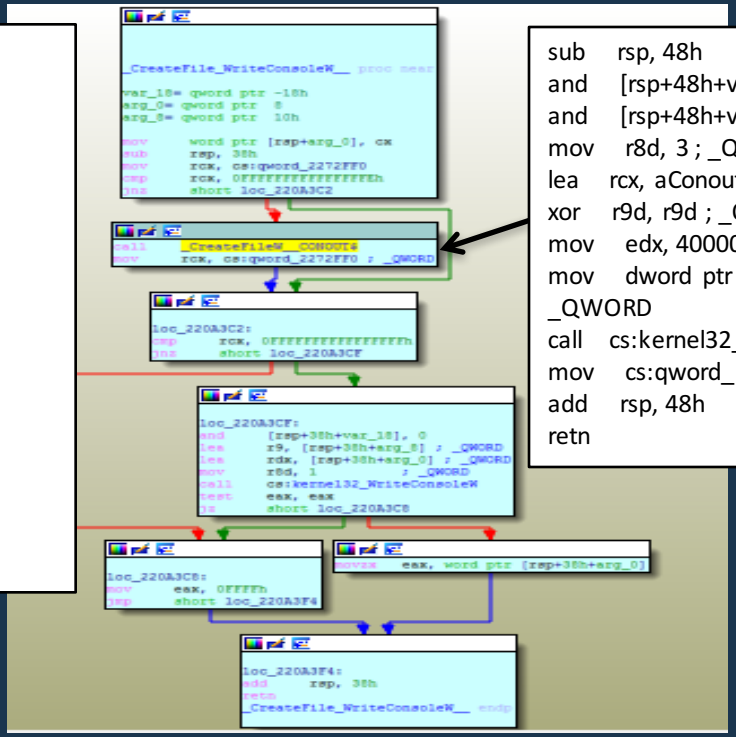
WINNTI (不易の部分例 アセンブラレベル)

プロセスオープンとスレッド作成



```
mov    [rsp-20h+arg_10], r8
mov    [rsp-20h+arg_18], r9
push   rbp
push   rbx
push   rsi
push   rdi
mov    rbp, rsp
sub    rsp, 58h
and    [rbp+var_38], 0
mov    rbx, rdx
xor    edx, edx
mov    rsi, r8
mov    rdi, rcx
lea    r8d, [rdx+28h]
lea    rcx, [rbp+var_30]
call   _imul_neg__
test   rsi, rsi
```

コンソール処理箇所CONOUT$



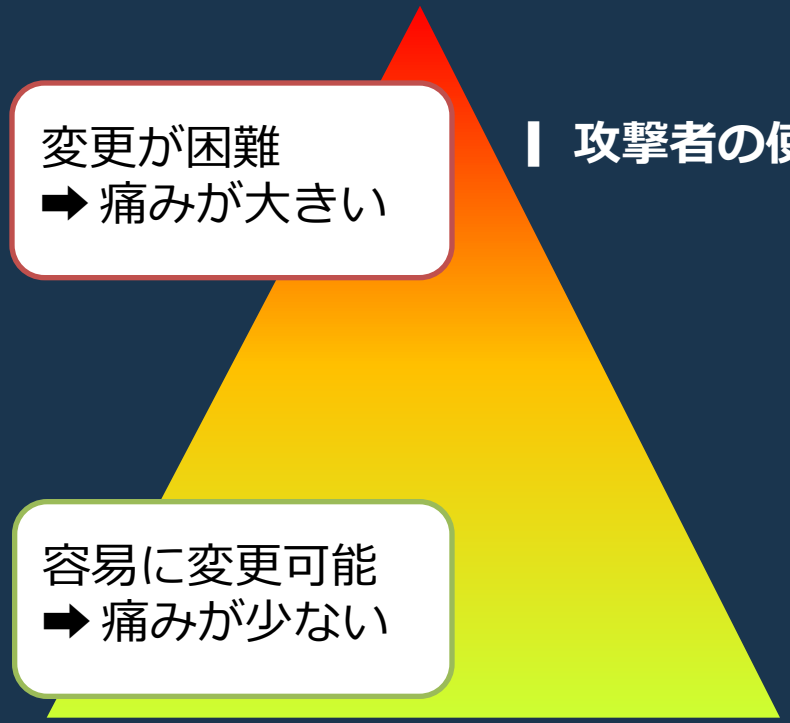
```
sub    rsp, 48h
and    [rsp+48h+var_18], 0
and    [rsp+48h+var_20], 0
mov    r8d, 3 ; _QWORD
lea    rcx, aConout ; "CONOUT$"
xor    r9d, r9d ; _QWORD
mov    edx, 40000000h ; _QWORD
mov    dword ptr [rsp+48h+var_28], r8d ;
_QWORD
call   cs:kernel32_CreateFileW
mov    cs:qword_1800B5EE0, rax
add    rsp, 48h
retn
```

YARAルール (yara32 xxx.yara PID)

```
rule MNC_APT_2016_17_WINNTI
{
 meta:
  malware_family = "WINNTI"
  actro = "WINNTI"
  last_modified = "2017-01"
  description = "Hex top of OpenProc_CreateThread / Strings on Memory"
  rev = 1
  weight = 100
 strings:
  $hex1 = { 4C 89 44 24 18 4C 89 4C 24 20 55 53 56 57 48 8B EC 48 83 EC 58 48 83 65 C8 00 48 8B }
  $hex2 = { DA 33 D2 49 8B F0 48 8B F9 44 8D 42 28 48 8D 4D D0 E8 ?? F? FF FF 48 85 F6 75 15 E8 }
  $str1 = "CONOUT$" wide
  $str2 = "A at L %d"
 condition:
  all of them
}
```

まとめ

APTをとらえる



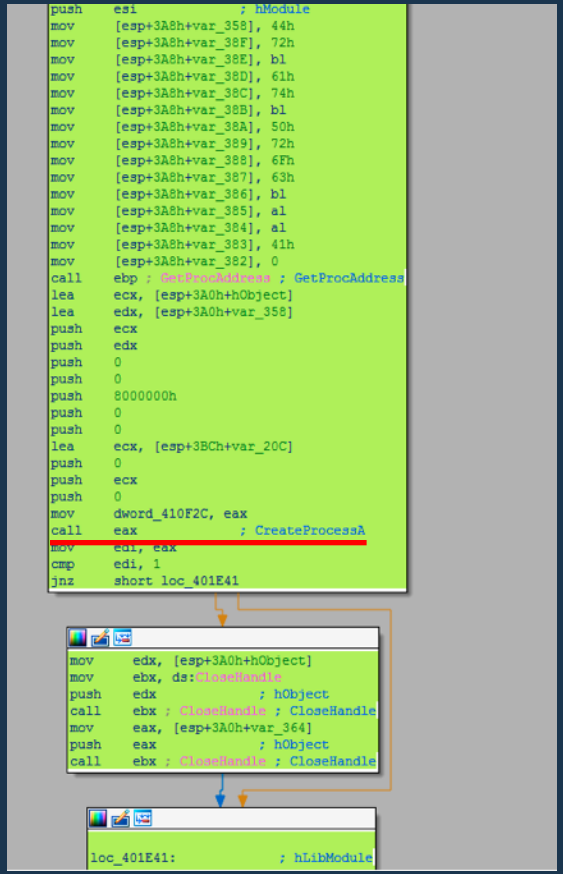
攻撃者における“痛み”のピラミッド [10]

攻撃者の使うRATの不易な部分（メモリ）

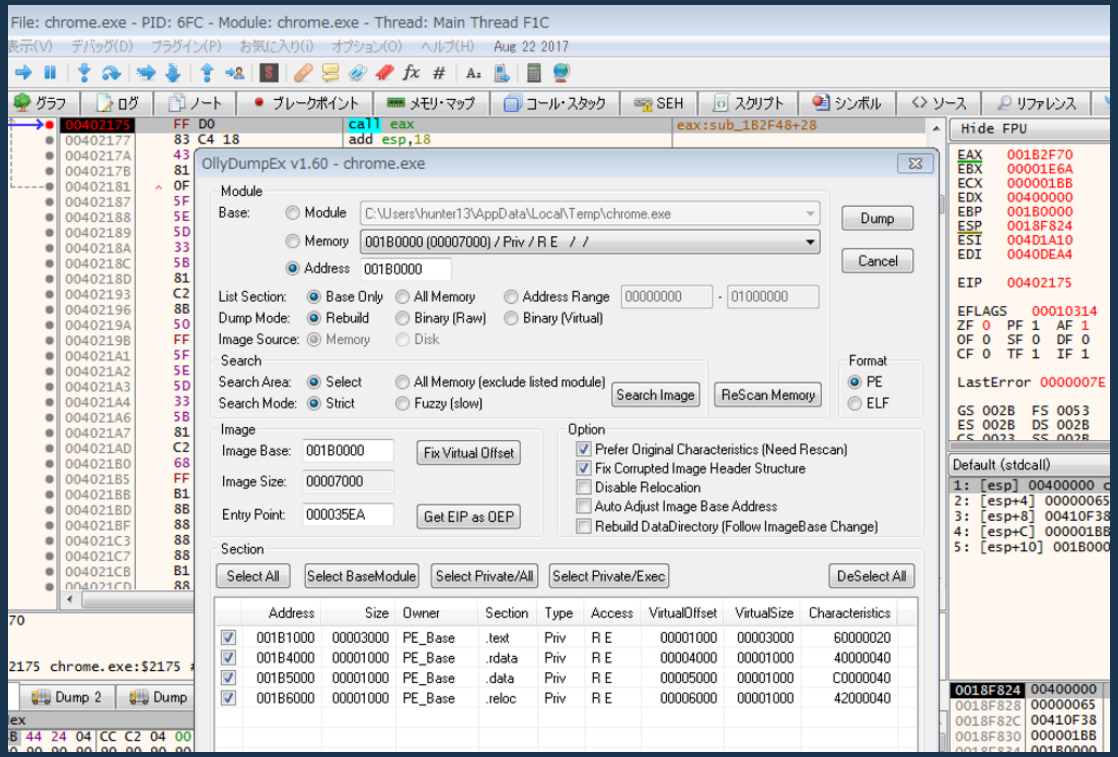
Appendix

Tools Tips：MazeWalker

- Intel社が公開しているPINを利用
- コードトレース
- 動的情報をidbに反映。

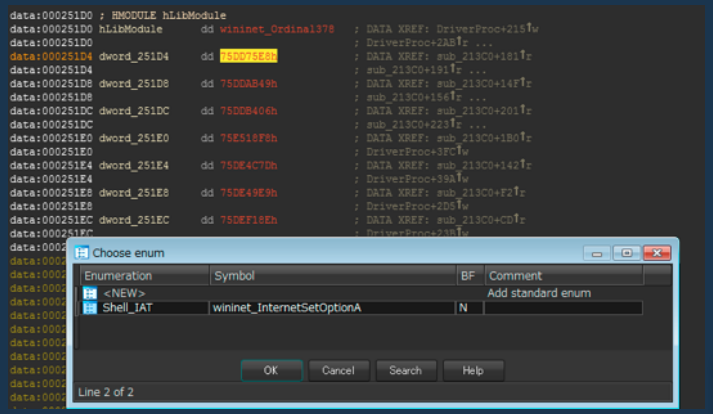


Tool Tips: x64dbg - OllyDump



Tools Tips: Dynamic Load Library IDBへ反映

```
data:000251D0 ; HMODULE hLibModule
data:000251D0 hLibModule      dd 75DC0000h          ; DATA XREF: DriverProc+215↑w
data:000251D0                                       ; DriverProc+2AB↑r ...
data:000251D4 dword_251D4     dd 75DD75E8h          ; DATA XREF: sub_213C0+181↑r
data:000251D4                                       ; sub_213C0+191↑r ...
data:000251D8 dword_251D8     dd 75DDAB49h          ; DATA XREF: sub_213C0+14F↑r
data:000251D8                                       ; sub_213C0+156↑r ...
data:000251DC dword_251DC     dd 75DDB406h          ; DATA XREF: sub_213C0+201↑r
data:000251DC                                       ; sub_213C0+223↑r ...
data:000251E0 dword_251E0     dd 75E518F8h          ; DATA XREF: sub_213C0+1B0↑r
data:000251E0                                       ; DriverProc+3FC↑w
data:000251E4 dword_251E4     dd 75DE4C7Dh          ; DATA XREF: sub_213C0+142↑r
data:000251E4                                       ; DriverProc+39A↑w
data:000251E8 dword_251E8     dd 75DE49E9h          ; DATA XREF: sub_213C0+F2↑r
data:000251E8                                       ; DriverProc+2D5↑w
data:000251EC dword_251EC     dd 75DEF18Eh          ; DATA XREF: sub_213C0+CD↑r
data:000251EC                                       ; DriverProc+23B↑w
```



- Load File > C Header
- Add Enum
- Type m

アドレス	アドレス	コメント
000251BC	00000020	
000251C0	00000040	
000251C4	00000000	
000251C8	00000000	
000251CC	00000000	
000251D0	75DC0000	wininet.Ordinal378
000251D4	75DD75E8	wininet.InternetSetOptionA
000251D8	75DDAB49	wininet.InternetCloseHandle
000251DC	75DDB406	wininet.InternetReadFile
000251E0	75E518F8	wininet.HttpSendRequestA
000251E4	75DE4C7D	wininet.HttpOpenRequestA
000251E8	75DE49E9	wininet.InternetConnectA
000251EC	75DEF18E	wininet.InternetOpenA
000251F0	00000000	

```
000251D0  75DC0000  wininet.Ordinal378
000251D4  75DD75E8  wininet.InternetSetOptionA
000251D8  75DDAB49  wininet.InternetCloseHandle
000251DC  75DDB406  wininet.InternetReadFile
000251E0  75E518F8  wininet.HttpSendRequestA
000251E4  75DE4C7D  wininet.HttpOpenRequestA
000251E8  75DE49E9  wininet.InternetConnectA
000251EC  75DEF18E  wininet.InternetOpenA
```

x64dbgのインポートアドレス領域を
テキストファイルに
保存

```
enum Shell_IAT {
wininet_Ordinal378 = 0x75DC0000,
wininet_InternetSetOptionA = 0x75DD75E8,
wininet_InternetCloseHandle = 0x75DDAB49,
wininet_InternetReadFile = 0x75DDB406,
wininet_HttpSendRequestA = 0x75E518F8,
wininet_HttpOpenRequestA = 0x75DE4C7D,
wininet_InternetConnectA = 0x75DE49E9,
wininet_InternetOpenA = 0x75DEF18E,
};
```

pythonでenum定義のC header
ファイルに
変換

Tools Tips: Dynamic Load Library IDBへ反映

```
import sys


def main():
    # API Dictionary List
    api_list = [
        {'api': 'dummpy', 'address': '0x0000'},
        {'api': 'dummy2', 'address': '0x0001'}
    ]

    # header File Open
    wf = open('shell_iat.h', 'w')
    enum_start = 'enum Shell_IAT {\n'
    enum_end = '};\n'
    wf.write(enum_start)

    for line in open(sys.argv[1]).readlines():
        f_dup = False
        address_table = line[:-1].split(' ')
        caller_address = address_table[0]
        # skip no address line


        api_address = '0x' + address_table[1]
        api_name = address_table[2].replace('.', '_')

        for api_x in api_list:
            if api_name == api_x['api']:
                f_dup = True
                break

        # No duplicate API
        if f_dup == False:
            new_dict = {'api': api_name, 'address': api_address}
            api_list.append(new_dict)
            api_map = api_name + ' = ' + api_address + ',' + '\n'
            wf.write(api_map)

    wf.write(enum_end)
    wf.close()

if __name__ == '__main__':
    main()
```

Tools

- x64dbg https://x64dbg.com
  - OllydumpEx
  - xAnalyzer
- MazeWalker https://github.com/0xPhoeniX/MazeWalker
- FLARE https://github.com/fireeye/flare-ida
- FileInsight https://www.mcafee.com/jp/downloads/free-tools/fileinsight.aspx
- BinDiff https://www.zynamics.com/bindiff.html
- Process Hacker http://processhacker.sourceforge.net/

Reference

1. http://blog.trendmicro.co.jp/archives/5293
2. https://www2.fireeye.com/WEB-2014RPTJPM-Trends2014.html
3. https://www.arbornetworks.com/blog/asert/wp-content/uploads/2014/06/ASERT-Threat-Intelligence-Brief-2014-07-Illuminating-Etumbot-APT.pdf
4. https://www.lac.co.jp/lacwatch/people/20160126_000306.html
5. https://kasperskycontenthub.com/wp-content/uploads/sites/43/vlpdfs/winnti-more-than-just-a-game-130410.pdf
6. http://blog.trendmicro.co.jp/archives/11809
7. https://hitcon.org/2016/pacific/0composition/pdf/1201/1201%20R2%201610%20winnti%20polymorphism.pdf
8. https://attack.mitre.org/wiki/Groups
9. https://github.com/0xhajic/
10. http://rvasec.com/slides/2014/Bianco_Pyramid%20of%20Pain.pdf