MWS CUP 2019
課題2: 静的解析

中津留勇
石丸傑
石淵一三
中島将太
古川和祈

CyberDefense

課題2 担当

SecureWorks Japan 株式会社
中津留 勇

株式会社カスペルスキー
石丸 傑

株式会社日立製作所
石淵 一三

去年のふりかえり

問題作成に関する課題

テーマ選定の難しさ

- 自動化などを静的解析で適用することの困難さ
- より実務に近い検体から，規定時間で解答可能なレベルの設問作成

問題の出し方

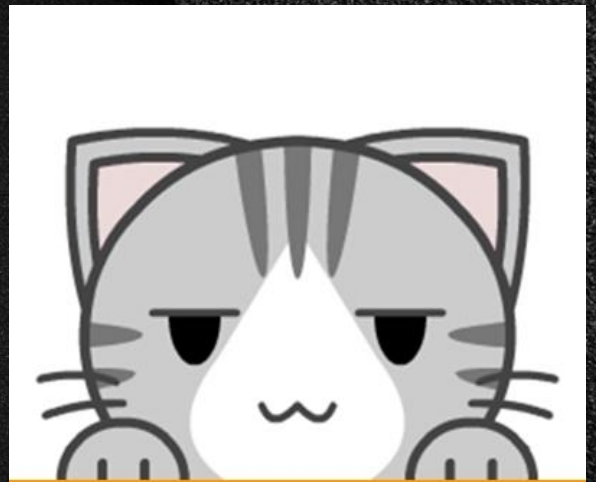
- 高配点問題の解答が二択となり，点数の偏りを生んでしまった
- 解答内容に求めるレベルを明示できていなかった

作成委員

- 例年，作成委員が海外出張でいない
  → “UN頼み”から1人参加してくれる(らしい)

課題2 新担当

サイバーディフェンス研究所
中島 将太



元UN頼みチーム
古川 和祈

課題2 変わらぬテーマ

静的解析を通じ・・・

マルウェアを正しく
理解する

最新情報を得る

実務に近い作業

MWS Cup 2019
課題2 概要

今年の検体は Datper

e38d3a7a86a72517b6ebea89cfd312db0f433385a33d87f2ec8bf83a62396bb3

JPCERT CC
Japan Computer Emergency Response Team Coordination Center
JPCERT コーディネーションセンター

JPCERT/CC Eyes

Top > "マルウエア"の一覧 > 攻撃グループTickによる日本の組織をターゲットにした攻撃活動

朝長 秀誠 (Shusei Tomonaga) 2019/02/19

攻撃グループTickによる日本の組織をターゲットにした攻撃活動

Datper

ツイート メール

以前のJPCERT/CC Eyesで攻撃グループTick[1] (BRONZE BUTLER[2]とも呼ばれる) が使用していると考えられるマルウエアDatperについて紹介しましたが、この攻撃グループに関連すると考えられる攻撃は現在も継続しています。2018年以降、JPCERT/CCにて確認している攻撃は以下の2つです。

- 標的型攻撃メールによるDatperの感染
- 資産管理ソフトウエアの脆弱性を悪用する攻撃

今回は、上記2つの攻撃から確認した新たな特徴について紹介します。

標的型攻撃メールによるDatperの感染

https://blogs.jpcert.or.jp/ja/2019/02/tick-activity.html

新たな挑戦・・・64bit

```
.text:0000000000A51A92 1208 90                   nop
.text:0000000000A51A93 1208 48 8B 8D F8 10 00 00 mov     rcx, [rbp+var_s10F8]
.text:0000000000A51A9A 1208 48 63 95 F4 10 00 00 movsxd  rdx, [rbp+var_s10F4]
.text:0000000000A51AA1 1208 4D 33 C0             xor     r8, r8
.text:0000000000A51AA4 1208 E8 87 9B 9B FF       call    sub_40B630
.text:0000000000A51AA9 1208 48 8D 8D E0 11 00 00 lea     rcx, [rbp+hObject+4] ; lpSystemTime
.text:0000000000A51AB0 1208 E8 7B E2 9C FF       call    GetLocalTime
.text:0000000000A51AB5 1208 0F B7 85 E8 11 00 00 movzx   eax, [rbp+var_s11E8]
.text:0000000000A51ABC 1208 3B 05 6E 95 0A 00    cmp     eax, cs:dword_AFB030
.text:0000000000A51AC2 1208 7C 0F                jl      short loc_A51AD3
```

The following table summarizes how parameters are passed:

Parameter type	How passed
Floating point	First 4 parameters - XMM0 through XMM3. Others passed on stack.
Integer	First 4 parameters - RCX, RDX, R8, R9. Others passed on stack.
Aggregates (8, 16, 32, or 64 bits) and __m64	First 4 parameters - RCX, RDX, R8, R9. Others passed on stack.
Aggregates (other)	By pointer. First 4 parameters passed as pointers in RCX, RDX, R8, and R9
__m128	By pointer. First 4 parameters passed as pointers in RCX, RDX, R8, and R9

https://docs.microsoft.com/en-us/cpp/build/x64-calling-convention?view=vs-2019

Windows APIの難読化解除

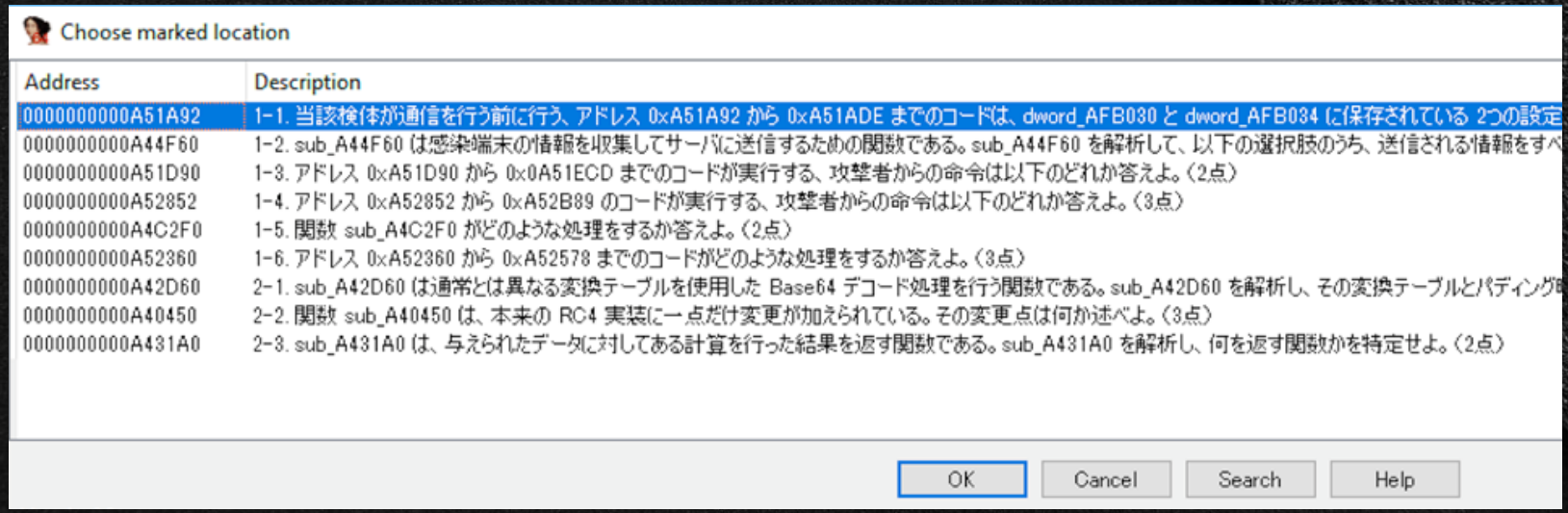
配布したIDBではAPIを解決しているが実際は難読化されている

```
.text:0000000000A432A9 098 lea     rdx, aIuUZa     ; "IU(U(za#\\;;"
.text:0000000000A432B0 098 call    sub_A41140
.text:0000000000A432B5 098 mov     rcx, [rbp+var_s78]
.text:0000000000A432B9 098 call    sub_410590
.text:0000000000A432BE 098 mov     rcx, rax        ; lpLibFileName
.text:0000000000A432C1 098 call    LoadLibraryA_0
.text:0000000000A432C6 098 mov     rbx, rax
.text:0000000000A432C9 098 lea     rcx, [rbp+var_s70]
.text:0000000000A432CD 098 lea     rdx, aAz3ZasZJ  ; "?(az3(zas:z(J"
.text:0000000000A432D4 098 call    sub_A41140
.text:0000000000A432D9 098 mov     rcx, [rbp+var_s70]
.text:0000000000A432DD 098 call    sub_410590
.text:0000000000A432E2 098 mov     rcx, rbx        ; hModule
.text:0000000000A432E5 098 mov     rdx, rax        ; lpProcName
.text:0000000000A432E8 098 call    GetProcAddress_1
.text:0000000000A432ED 098 mov     cs:InternetOpenA, rax
.text:0000000000A432F4 098 lea     rcx, [rbp+var_s68]
.text:0000000000A432F8 098 lea     rdx, aAz3ZaFZAj ; "?(az3(za}F((z,aJ"
.text:0000000000A432FF 098 call    sub_A41140
.text:0000000000A43304 098 mov     rcx, [rbp+var_s68]
.text:0000000000A43308 098 call    sub_410590
.text:0000000000A4330D 098 mov     rcx, rbx        ; hModule
.text:0000000000A43310 098 mov     rdx, rax        ; lpProcName
.text:0000000000A43313 098 call    GetProcAddress_1
.text:0000000000A43318 098 mov     cs:InternetConnectA, rax
```

IDAのブックマーク機能

- ALT + M -> ブックマーク登録
- CTRL + M -> ブックマークリスト

配布した.i64にブックマークを設定しました



MWS Cup 2019
課題2 解答例

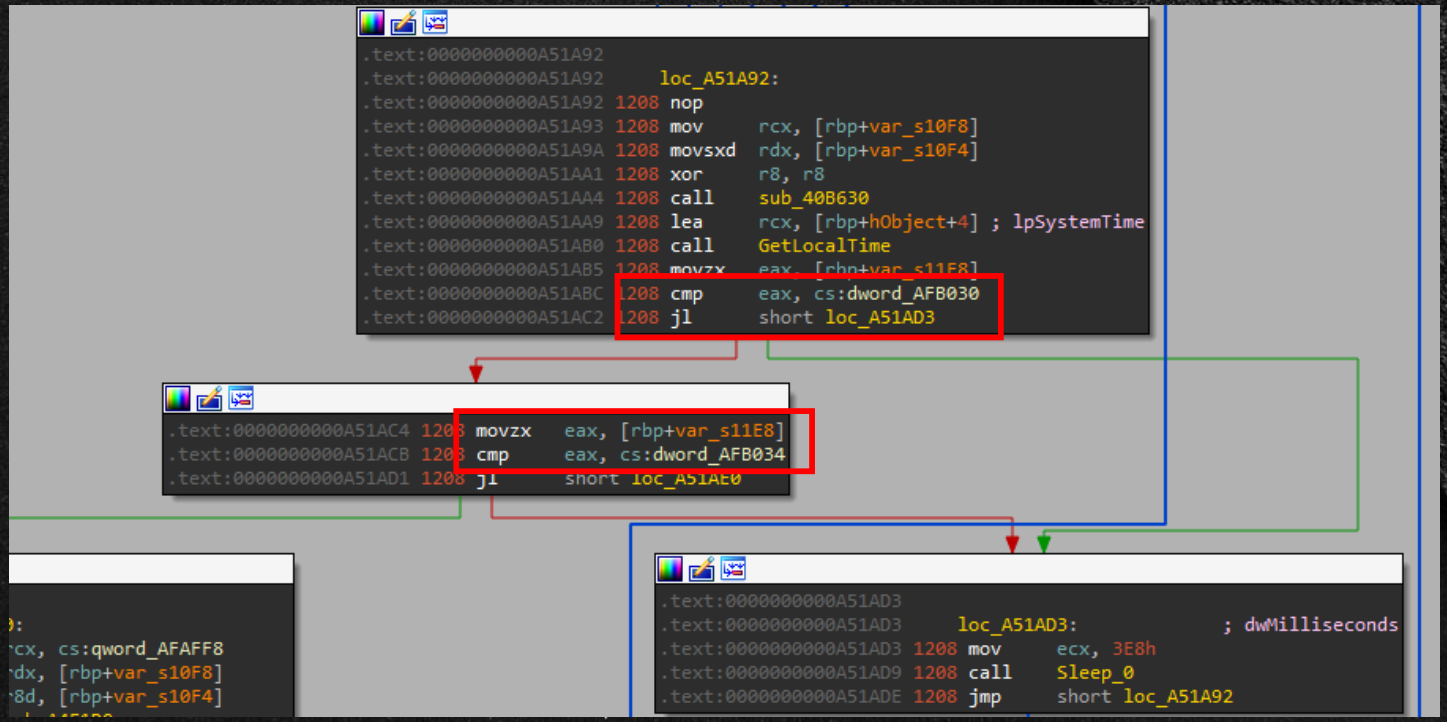
設問の意図

実務における解析を行う目的となりえる要素を設問として出題

1. 機能の特定
   - RAT である当該検体がどのような命令を実行可能なのか解析せよ
2. 暗号とエンコーディング
   - 当該検体の暗号やエンコーディングに関連した処理について設問に答えよ
3. マルウェアの特定
   - これまでの回答やその他の解析結果を元に当該検体の名称（Emdivi、Daserf、Oni、Plead 等）を答えよ

1-1 マルウェアの設定を理解する

1-1. 当該検体が通信を行う前に行う、アドレス 0xA51A92 から 0xA51ADE までのコードは、dword_AFB030 と dword_AFB034 に保存されている 2つの設定情報を使ってある処理を行う。このコードがどのような処理をするか説明せよ。（3点）



1-1 直前で呼ばれているAPIについて調べる

```
.text:0000000000A51A92
.text:0000000000A51A92     loc_A51A92:
.text:0000000000A51A92 1208 nop
.text:0000000000A51A93 1208 mov     rcx, [rbp+var_s10F8]
.text:0000000000A51A9A 1208 movsxd  rdx, [rbp+var_s10F4]
.text:0000000000A51AA1 1208 xor     r8, r8
.text:0000000000A51AA4 1208 call    sub_40B630
.text:0000000000A51AA9 1208 lea     rcx, [rbp+hObject+4] ; lpSystemTime
.text:0000000000A51AB0 1208 call    GetLocalTime
.text:0000000000A51AB5 1208 movzx   eax, [rbp+var_s11E8]
.text:0000000000A51ABC 1208 cmp     eax, cs:dword_AFB030
.text:0000000000A51AC2 1208 jl      short loc_A51AD3
```

GetLocalTime function

2018/12/05 • 読了までの所要時間: 2 分

Retrieves the current local date and time.

To retrieve the current date and time in Coordinated Universal Time (UTC) format, use the GetSystemTime function.

Syntax

C++ コピー

```
void GetLocalTime(
  LPSYSTEMTIME lpSystemTime
);
```

SYSTEMTIME structure

2018/12/05 • 読了までの所要時間: 2 分

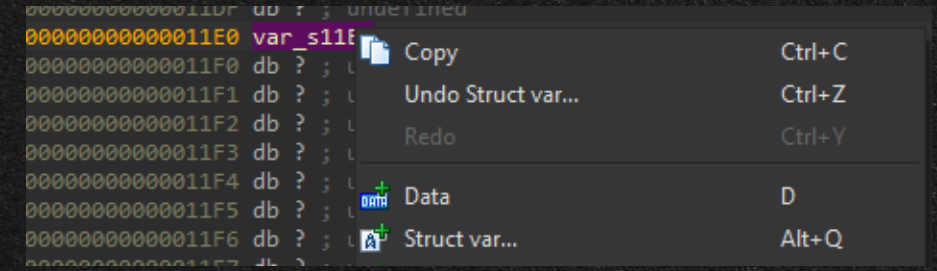
Specifies a date and time, using individual members for the month, day, year, weekday, hour, minute, second, and millisecond. The time is either in coordinated universal time (UTC) or local time, depending on the function that is being called.

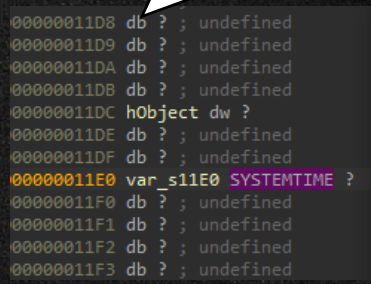
Syntax

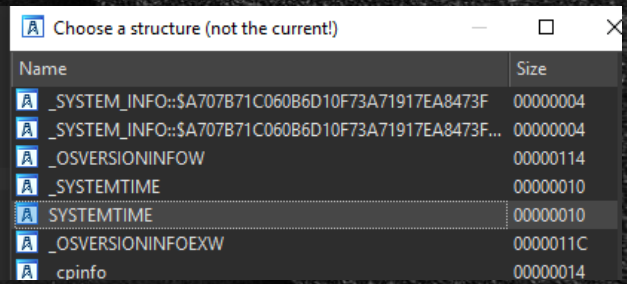
C++ コピー

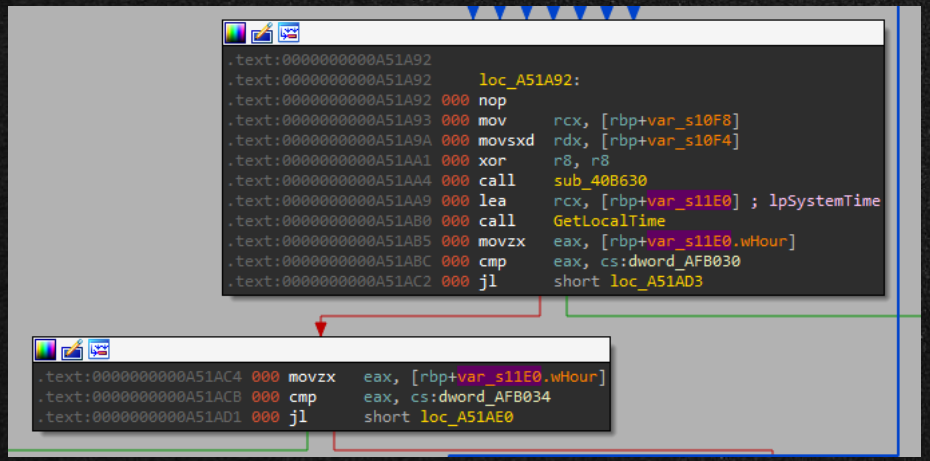
```
typedef struct _SYSTEMTIME {
  WORD wYear;
  WORD wMonth;
  WORD wDayOfWeek;
  WORD wDay;
  WORD wHour;
  WORD wMinute;
  WORD wSecond;
  WORD wMilliseconds;
} SYSTEMTIME, *PSYSTEMTIME, *LPSYSTEMTIME;
```

1-1 構造体の適用！


00000000000011E0 var_s11E
Copy Ctrl+C
Undo Struct var... Ctrl+Z
Redo Ctrl+Y
Data D
Struct var... Alt+Q

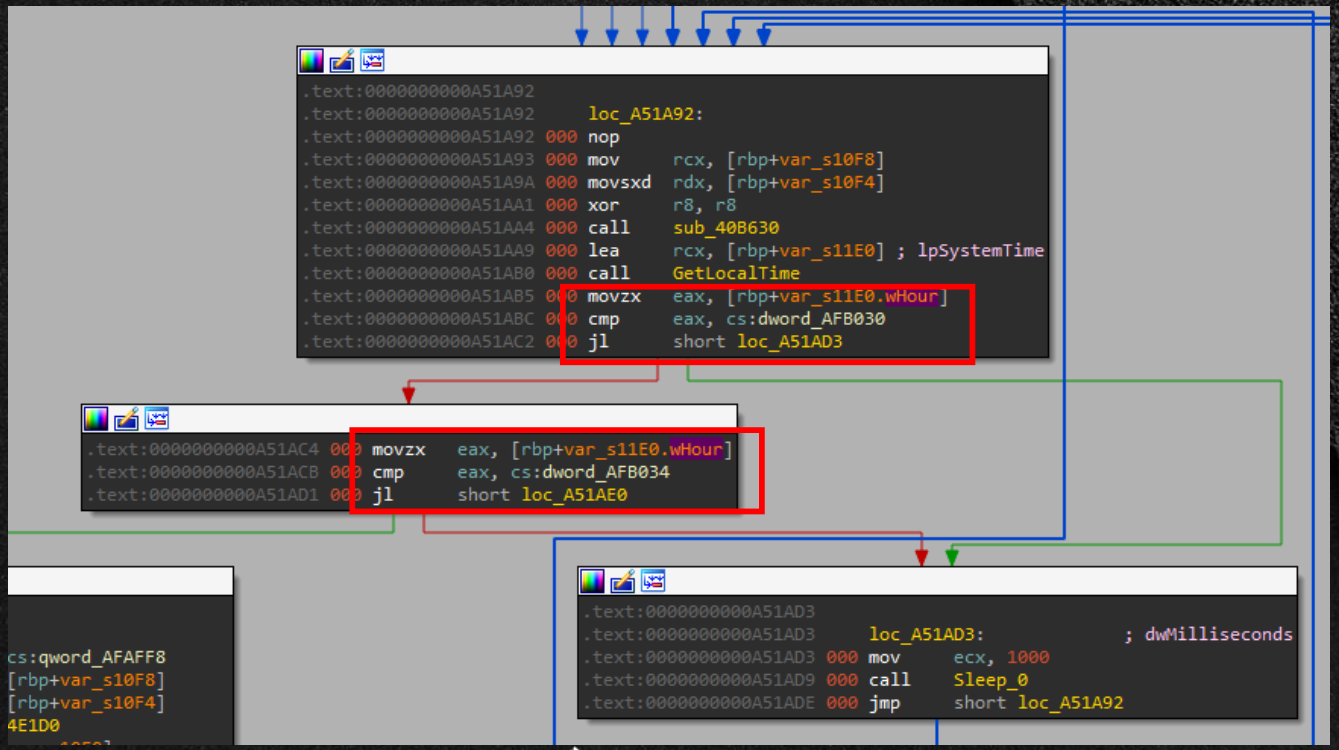

00000011D8 db ? ; undefined
00000011D9 db ? ; undefined
00000011DA db ? ; undefined
00000011DB db ? ; undefined
00000011DC hObject dw ?
00000011DE db ? ; undefined
00000011DF db ? ; undefined
00000011E0 var_s11E0 SYSTEMTIME ?
00000011F0 db ? ; undefined
00000011F1 db ? ; undefined
00000011F2 db ? ; undefined
00000011F3 db ? ; undefined


Choose a structure (not the current!)
Name
Size
_SYSTEM_INFO::$A707B71C060B6D10F73A71917EA8473F 00000004
_SYSTEM_INFO::$A707B71C060B6D10F73A71917EA8473F... 00000004
_OSVERSIONINFOW 00000114
_SYSTEMTIME 00000010
SYSTEMTIME 00000010
_OSVERSIONINFOEXW 0000011C
cpinfo 00000014


.text:0000000000A51A92
.text:0000000000A51A92 loc_A51A92:
.text:0000000000A51A92 000 nop
.text:0000000000A51A93 000 mov rcx, [rbp+var_s10F8]
.text:0000000000A51A9A 000 movsxd rdx, [rbp+var_s10F4]
.text:0000000000A51AA1 000 xor r8, r8
.text:0000000000A51AA4 000 call sub_40B630
.text:0000000000A51AA9 000 lea rcx, [rbp+var_s11E0] ; lpSystemTime
.text:0000000000A51AB0 000 call GetLocalTime
.text:0000000000A51AB5 000 movzx eax, [rbp+var_s11E0.wHour]
.text:0000000000A51ABC 000 cmp eax, cs:dword_AFB030
.text:0000000000A51AC2 000 jl short loc_A51AD3
.text:0000000000A51AC4 000 movzx eax, [rbp+var_s11E0.wHour]
.text:0000000000A51ACB 000 cmp eax, cs:dword_AFB034
.text:0000000000A51AD1 000 jl short loc_A51AE0

1-1 解答

現在時刻が dword_AFB030 より後で dword_AFB034 より前のとき loc_A51AE0 の処理へ進み、それ以外の時刻では1000ミリ秒停止したこの比較処理を繰り返す



1-2 マルウェアの挙動を理解する

1-2. sub_A44F60 は感染端末の情報を収集してサーバに送信するための関数である。sub_A44F60 を解析して、以下の選択肢のうち、送信される情報をすべて選択せよ。（2点）

- [ ] コンピュータ名
- [ ] ユーザ名
- [ ] OS バージョン
- [ ] IP アドレス
- [ ] MAC アドレス
- [ ] キーボード配列

1-2 WinAPIから収集情報を確認

```
.text:0000000000A44F85 EB8 mov     edx, 0D8h ; 'Ø'
.text:0000000000A44F8B EB8 xor     r8, r8
.text:0000000000A44F8E EB8 call    sub_40B630
.text:0000000000A44F93 EB8 call    cs:GetSystemDefaultLangID
.text:0000000000A44F99 EB8 movzx   eax, ax
.text:0000000000A44F9C EB8 mov     [rbx+54h], eax
.text:0000000000A44F9F EB8 mov     [rbp+nSize], 400h
.text:0000000000A44FA9 EB8 lea     rcx, [rbp+Buffer]
.text:0000000000A44FB0 EB8 mov     edx, 400h
.text:0000000000A44FB6 EB8 xor     r8, r8
.text:0000000000A44FB9 EB8 call    sub_40B630
.text:0000000000A44FBE EB8 lea     rcx, [rbp+Buffer] ; lpBuffer
.text:0000000000A44FC5 EB8 lea     rdx, [rbp+nSize] ; nSize
.text:0000000000A44FCC EB8 call    cs:GetComputerNameA
.text:0000000000A44FD2 EB8 mov     [rbp+nSize], 400h
.text:0000000000A44FDC EB8 lea     rcx, [rbp+var_sC8]
.text:0000000000A44FE3 EB8 mov     edx, 800h
.text:0000000000A44FE9 EB8 xor     r8, r8
.text:0000000000A44FEC EB8 call    sub_40B630
.text:0000000000A44FF1 EB8 lea     rcx, [rbp+var_sC8] ; lpBuffer
.text:0000000000A44FF8 EB8 lea     rdx, [rbp+nSize] ; nSize
.text:0000000000A44FFF EB8 call    cs:GetComputerNameW
.text:0000000000A45005 EB8 lea     rcx, [rbx+V
.text:0000000000A45009 EB8 lea     rdx, [rbp+var_sC8]
.text:0000000000A45010 EB8 call    sub_A3EE00
.text:0000000000A45015 EB8 lea     rcx, [rbp+VersionInformation]
.text:0000000000A45019 EB8 mov     edx, 9Ch
.text:0000000000A4501F EB8 xor     r8, r8
.text:0000000000A45022 EB8 call    sub_40B630
.text:0000000000A45027 EB8 mov     [rbp+VersionInformation.dwOSVersionInfoSize], 9Ch
.text:0000000000A4502E EB8 lea     rcx, [rbp+VersionInformation] ; lpVersionInformation
.text:0000000000A45032 EB8 call    cs:GetVersionExA
.text:0000000000A45038 EB8 test    eax, eax
.text:0000000000A45053     loc_A45053:                 ; name
.text:0000000000A45053 EB8 lea     rcx, [rbp+Buffer]
.text:0000000000A4505A EB8 call    cs:gethostbyname
.text:0000000000A45060 EB8 mov     rsi, rax
```

システムロケールの言語ID

コンピュータ名(Ascii)

コンピュータ名(Wide)

OSバージョン

IPアドレス

1-2 解答

1-2. sub_A44F60 は感染端末の情報を収集してサーバに送信するための関数である。sub_A44F60 を解析して、以下の選択肢のうち、送信される情報をすべて選択せよ。（2点）

- [x] コンピュータ名
- [ ] ユーザ名
- [x] OS バージョン
- [x] IP アドレス
- [ ] MAC アドレス
- [ ] キーボード配列

1-3 マルウェアの挙動を理解する

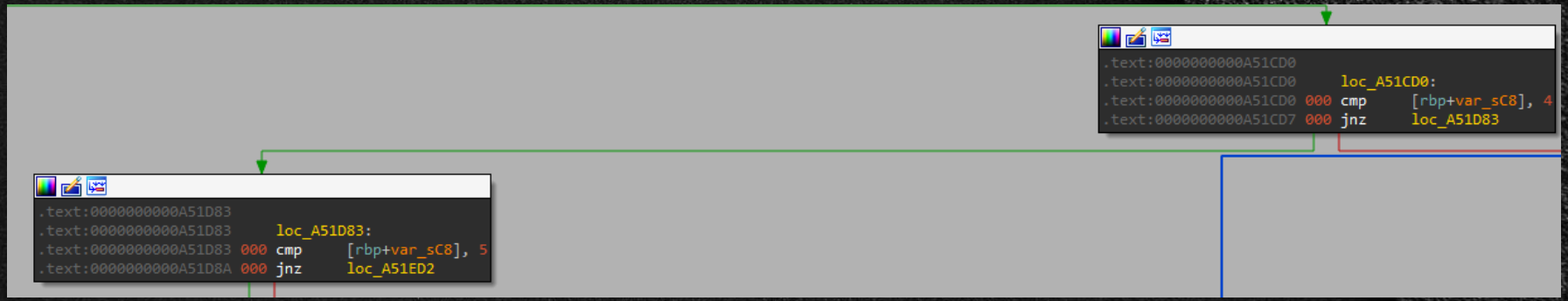
1-3. アドレス 0xA51D90 から 0x0A51ECD までのコードが実行する、攻撃者からの命令は以下のどれか答えよ。（2点）

- ○ 感染端末への任意のファイル送信
- ○ C2 サーバへの任意のファイル送信
- ○ 任意のコマンド実行
- ○ 任意のプロセスの終了
- ○ プロセス一覧の取得
- ○ マルウェアのインストール
- ○ マルウェアのアップデート
- ○ マルウェアのアンインストール

コマンド解析のための補足

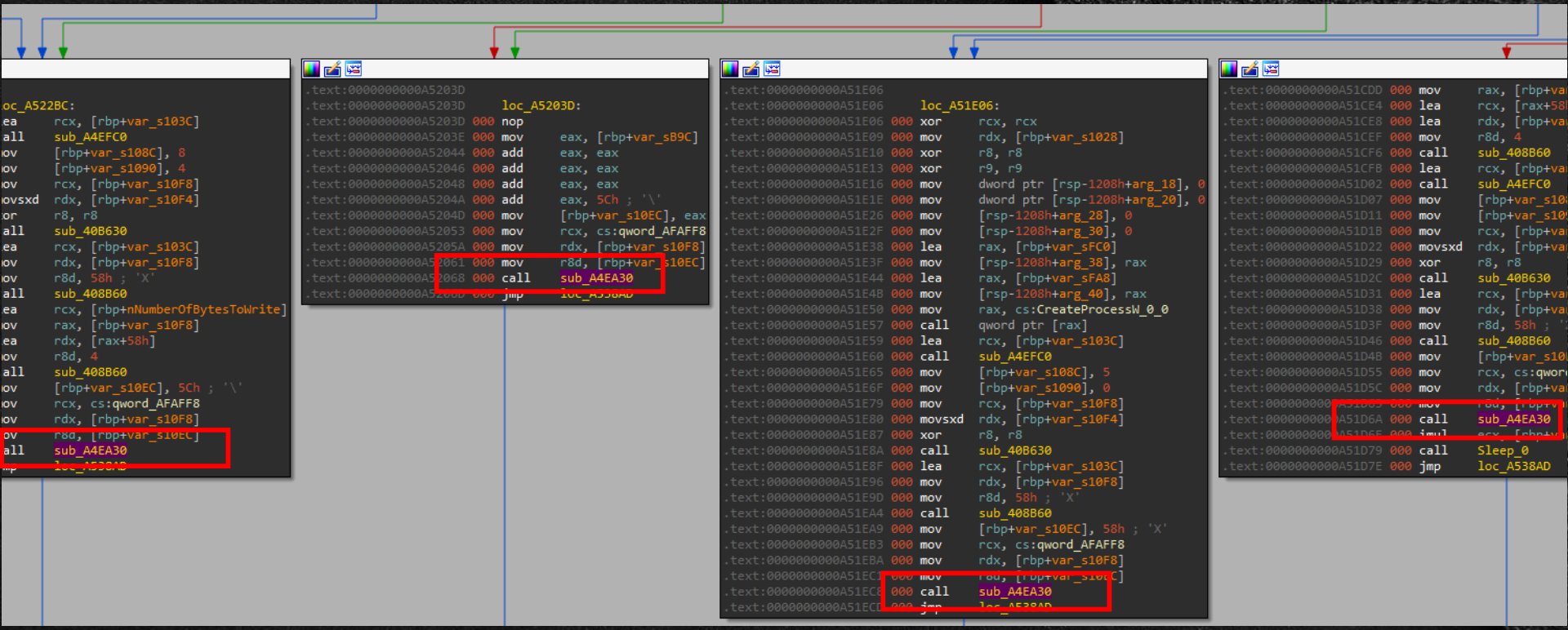
特定の変数[rdp+ver_sC8]の値をハードコードされた数値と比較する分岐処理がある

- RATのコマンド分岐処理
- 分岐後の処理が問題として出題されている



コマンド解析のための補足

各分岐の最後に呼び出されているsub_A4EA30を確認する



コマンド解析のための補足

sub_A4EA30から呼び出される処理に通信系の処理がある

- コマンド実行後、C2へ結果を送信している処理と推測する

```
.text:0000000000A43808     loc_A43808:             ; hConnect
.text:0000000000A43808 2AB8 mov     rcx, rsi
.text:0000000000A4380B 2AB8 lea     rdx, aGet       ; lpszVerb
.text:0000000000A43812 2AB8 mov     r8, [rbp+lpszObjectName] ; lpszObjectName
.text:0000000000A43816 2AB8 lea     r9, szVersion   ; "HTTP/1.1"
.text:0000000000A4381D 2AB8 mov     rax, [rbp+lpszReferrer]
.text:0000000000A43824 2AB8 mov     qword ptr [rsp+dwFlags], rax ; lpszReferrer
.text:0000000000A43829 2AB8 mov     qword ptr [rsp+dwService], 0 ; lplpszAcceptTypes
.text:0000000000A43832 2AB8 mov     [rsp+var_s30], 80000100h ; dwFlags
.text:0000000000A4383A 2AB8 mov     [rsp+dwContext], 0 ; dwContext
.text:0000000000A43843 2AB8 call    cs:HttpOpenRequestA
.text:0000000000A43849 2AB8 mov     rdi, rax
.text:0000000000A4384C
.text:0000000000A4384C     loc_A4384C:
.text:0000000000A4384C 2AB8 test    rdi, rdi
.text:0000000000A4384F 2AB8 jnz     short loc_A43871
.text:0000000000A43851 2AB8 mov     rcx, rbx        ; hInternet
.text:0000000000A43854 2AB8 call    cs:InternetCloseHandle
.text:0000000000A4385A 2AB8 mov     rcx, rsi        ; hInternet
.text:0000000000A4385D 2AB8 call    cs:InternetCloseHandle
.text:0000000000A43863 2AB8 mov     eax, 0FFFFFFFDh
.text:0000000000A43869 2AB8 mov     [rbp+var_s64], eax
.text:0000000000A4386C 2AB8 jmp     loc_A43C19
.text:0000000000A43871
.text:0000000000A43871     loc_A43871:             ; lpString
.text:0000000000A43871 2AB8 lea     rcx, [rbp+String1]
.text:0000000000A43878 2AB8 call    lstrlenA
.text:0000000000A4387D 2AB8 mov     rcx, rdi        ; hRequest
.text:0000000000A43880 2AB8 lea     rdx, [rbp+String1] ; lpszHeaders
.text:0000000000A43887 2AB8 mov     r8d, eax        ; dwHeadersLength
.text:0000000000A4388A 2AB8 mov     r9d, 20000000h ; dwModifiers
.text:0000000000A43891 2AB8 call    cs:HttpAddRequestHeadersA
.text:0000000000A43897 2AB8 cmp     cs:dword_AD9CC0, 0
.text:0000000000A4389E 2AB8 jz      short loc_A43912
.text:0000000000A438A0 2AB8 lea     rcx, Buffer     ; lpString
```

1-3 呼び出されるアドレスを確認する

```
0000000000A51E06                        loc_A51E06:             ; LPCWSTR lpApplicationName
0000000000A51E06                        _lpApplicationName = rcx
0000000000A51E06 48 33 C9               xor     _lpApplicationName, _lpApplicationName
0000000000A51E09                        _lpCommandLine = rdx
0000000000A51E09 48 8B 95 28 10 00 00   mov     _lpCommandLine, [rbp+lpCommandLine] ; LPWSTR lpCommandLine
0000000000A51E10 4D 33 C0               xor     r8, r8
0000000000A51E13 4D 33 C9               xor     r9, r9
0000000000A51E16 C7 44 24 20 00 00 00 00 mov    [rsp+dwCreationDisposition], 0
0000000000A51E1E C7 44 24 28 00 00 00 00 mov    [rsp+dwFlagsAndAttributes], 0
0000000000A51E26 48 C7 44 24 30 00 00 00 00 mov [rsp+hTemplateFile], 0
0000000000A51E2F 48 C7 44 24 38 00 00 00 00 mov [rsp+var_s38], 0
0000000000A51E38 48 8D 85 C0 0F 00 00   lea     rax, [rbp+var_sFC0]
0000000000A51E3F 48 89 44 24 40         mov     [rsp+var_s40], rax
0000000000A51E44 48 8D 85 A8 0F 00 00   lea     rax, [rbp+var_sFA8]
0000000000A51E4B 48 89 44 24 48         mov     [rsp+var_s48], rax
0000000000A51E50                        CreateProcessW = rax
0000000000A51E50 48 8B 05 D1 99 08 00   mov     CreateProcessW, cs:CreateProcessW_0_0
0000000000A51E57 FF 10                  call    qword ptr [CreateProcessW]
0000000000A51E59 48 8D 8D 3C 10 00 00   lea     rcx, [rbp+var_s103C]
```

```
.data:0000000000ADB818 off_ADB818 dq offset qword_AE54C0
.data:0000000000ADB820 off_ADB820 dq offset CreateConsoleScreenBuffer
.data:0000000000ADB828 off_ADB828 dq offset CreateProcessW_0
.data:0000000000ADB830 off_ADB830 dq offset unk_A65030
.data:0000000000ADB838 off_ADB838 dq offset off_421B70
```

lpCommandLine
の代入元

```
0000000000A51DA9 E8 B2 6D 9B FF         call    sub_408360
0000000000A51DAE 48 8B 85 F8 10 00 00   mov     rax, [rbp+var_s10F8]
0000000000A51DB5 48 8D 40 59            lea     rax, [rax+59h]
0000000000A51DB9 48 89 85 28 10 00 00   mov     [rbp+lpCommandLine], rax
0000000000A51DC0 48 8D 8D C0 0F 00 00   lea     rcx, [rbp+var_sFC0]
0000000000A51DC7 C7 C2 68 00 00 00      mov     edx, 68h ; 'h'
0000000000A51DCD 4D 33 C0               xor     r8, r8
0000000000A51DD0 E8 5B 98 9B FF         call    sub_40B630
```

CreateProcessWを使う処理であること、第2引数である
lpCommandLine（rdx）を参照し、値の代入元を確認してみると
削除やインストールを予測出来る文字列が見つからないことから推測

1-3 解答

1-3. アドレス 0xA51D90 から 0x0A51ECD までのコードが実行する、攻撃者からの命令は以下のどれか答えよ。（2点）

- ○ 感染端末への任意のファイル送信
- ○ C2 サーバへの任意のファイル送信
- ● 任意のコマンド実行
- ○ 任意のプロセスの終了
- ○ プロセス一覧の取得
- ○ マルウェアのインストール
- ○ マルウェアのアップデート
- ○ マルウェアのアンインストール

1-4 マルウェアの挙動を理解する

1-4. アドレス 0xA52852 から 0xA52B89 のコードが実行する、攻撃者からの命令は以下のどれか答えよ。（3点）

- ○ 感染端末への任意のファイル送信
- ○ C2 サーバへの任意のファイル送信
- ○ 任意のコマンド実行
- ○ 任意のプロセスの終了
- ○ プロセス一覧の取得
- ○ マルウェアのインストール
- ○ マルウェアのアップデート
- ○ マルウェアのアンインストール

1-4 API呼び出しの流れを理解

```
A5290B      loc_A5290B:
A5290B 000  mov     rcx, [rbp+var_sE8]
A52912 000  call    sub_410920
A52917 000  mov     rcx, rax        ; lpFileName
A5291A 000  mov     edx, 80000000h  ; dwDesiredAccess
A52920 000  mov     r8d, 1          ; dwShareMode
A52927 000  xor     r9, r9          ; lpSecurityAttributes
A5292A 000  mov     dword ptr [rsp-1208h+arg_18], 3 ; dwCreationDisposition
A52932 000  mov     dword ptr [rsp-1208h+arg_20], 80h ; dwFlagsAndAttributes
A5293A 000  mov     [rsp-1208h+arg_28], 0 ; hTemplateFile
A52943 000  call    CreateFileW_0
A52948 000  mov     dword ptr [rbp+hFile], eax
A5294E 000  mov     eax, dword ptr [rbp+hFile]
A52954 000  cmp     rax, 0FFFFFFFFFFFFFFFFh
A52958 000  jnz     loc_A52A2C
```

```
0A52A2C
0A52A2C      loc_A52A2C:                ; hFile
0A52A2C 000  mov     ecx, dword ptr [rbp+hFile]
0A52A32 000  xor     rdx, rdx           ; lpFileSizeHigh
0A52A35 000  call    GetFileSize
0A52A3A 000  mov     [rbp+nNumberOfBytesToRead], eax
0A52A40 000  mov     eax, [rbp+nNumberOfBytesToRead]
0A52A46 000  mov     [rbp+nNumberOfBytesToWrite], eax
```

```
.text:0000000000A52A62 000 add     eax, [rbp+nNumberOfBytesToRead]
.text:0000000000A52A68 000 add     eax, 5Ch ; '\'
.text:0000000000A52A6B 000 mov     [rbp+var_s10EC], eax
.text:0000000000A52A71 000 mov     eax, [rbp+var_s10EC]
.text:0000000000A52A77 000 sub     eax, 58h ; 'X'
.text:0000000000A52A7A 000 mov     [rbp+var_s1090], eax
.text:0000000000A52A80 000 mov     ecx, 40h ; '@'   ; uFlags
.text:0000000000A52A86 000 movsxd  rdx, [rbp+var_s10EC] ; dwBytes
.text:0000000000A52A8D 000 call    GlobalAlloc
.text:0000000000A52A92 000 mov     [rbp+lpRootPathName], rax
.text:0000000000A52A99 000 lea     rcx, [rbp+var_s103C]
.text:0000000000A52AA0 000 mov     rdx, [rbp+lpRootPathName]
.text:0000000000A52AA7 000 mov     r8d, 58h ; 'X'
```

```
00A52B25 000 mov     ecx, dword ptr [rbp+hFile] ; hFile
00A52B2B 000 mov     rdx, [rbp+lpRootPathName]
00A52B32 000 add     eax, eax
00A52B34 000 movsxd  rax, eax
00A52B37 000 lea     rdx, [rdx+rax+5Ch] ; lpBuffer
00A52B3C 000 mov     r8d, [rbp+nNumberOfBytesToRead] ; nNumberOfBytesToRead
00A52B43 000 lea     r9, [rbp+NumberOfBytesRead] ; lpNumberOfBytesRead
00A52B4A 000 mov     [rsp-1208h+arg_18], 0 ; lpOverlapped
00A52B53 000 call    ReadFile
```

ファイルを読み込む処理であること、
そのあとC2に通信を行うことから推測

1-4 解答

1-4. アドレス 0xA52852 から 0xA52B89 のコードが実行する、攻撃者からの命令は以下のどれか答えよ。（3点）

- ○ 感染端末への任意のファイル送信
- ● C2 サーバへの任意のファイル送信
- ○ 任意のコマンド実行
- ○ 任意のプロセスの終了
- ○ プロセス一覧の取得
- ○ マルウェアのインストール
- ○ マルウェアのアップデート
- ○ マルウェアのアンインストール

1-5 マルウェアの挙動を理解する

1-5. 関数 sub_A4C2F0 がどのような処理をするか答えよ。（2点）

```
0A4CC2E 478 lea     rcx, [rbp+String1]
0A4CC32 478 lea     rdx, aShel      ; "shel"
0A4CC39 478 call    sub_A3EDD0
0A4CC3E 478 lea     rcx, [rbp+String1] ; lpString1
0A4CC42 478 lea     rdx, aL32D      ; "l32.d"
0A4CC49 478 call    lstrcatA
0A4CC4E 478 lea     rcx, [rbp+String1] ; lpString1
0A4CC52 478 lea     rdx, aLl        ; "ll"
0A4CC59 478 call    lstrcatA
0A4CC5E 478 lea     rcx, [rbp+String1] ; lpLibFileName
0A4CC62 478 call    LoadLibraryA_0
0A4CC67 478 mov     rsi, rax
0A4CC6A 478 lea     rcx, [rbp+String1]
0A4CC6E 478 lea     rdx, aShfil     ; "SHFil"
0A4CC75 478 call    sub_A3EDD0
0A4CC7A 478 lea     rcx, [rbp+String1] ; lpString1
0A4CC7E 478 lea     rdx, aEoper     ; "eOper"
0A4CC85 478 call    lstrcatA
0A4CC8A 478 lea     rcx, [rbp+String1] ; lpString1
0A4CC8E 478 lea     rdx, aAtionw    ; "ationW"
0A4CC95 478 call    lstrcatA
0A4CC9A 478 mov     rcx, rsi        ; hModule
0A4CC9D 478 lea     rdx, [rbp+String1] ; lpProcName
0A4CCA1 478 call    GetProcAddress_1
0A4CCA6 478 mov     rsi, rax
0A4CCA9 478 lea     rcx, [rbp+var_s428]
0A4CCB0 478 mov     edx, 38h ; '8'
0A4CCB6 478 xor     r8, r8
0A4CCB9 478 call    sub_40B630
0A4CCBE 478 mov     [rbp+var_s430], 3
0A4CCC8 478 mov     [rbp+var_s448], 14h
0A4CCD1 478 mov     [rbp+var_s438], rbx
0A4CCD8 478 lea     rcx, [rbp+var_s428]
0A4CCDF 478 call    rsi
```

1-5 GetProcAddress

```
0A4CC2E 478 lea     rcx, [rbp+String1]
0A4CC32 478 lea     rdx, aShel      ; "shel"
0A4CC39 478 call    sub_A3EDD0
0A4CC3E 478 lea     rcx, [rbp+String1] ; lpString1
0A4CC42 478 lea     rdx, aL32D      ; "l32.d"
0A4CC49 478 call    lstrcatA
0A4CC4E 478 lea     rcx, [rbp+String1] ; lpString1
0A4CC52 478 lea     rdx, aLl        ; "ll"
0A4CC59 478 call    lstrcatA
0A4CC5E 478 lea     rcx, [rbp+String1] ; lpLibFileName
0A4CC62 478 call    LoadLibraryA_0
0A4CC67 478 mov     rsi, rax
0A4CC6A 478 lea     rcx, [rbp+String1]
0A4CC6E 478 lea     rdx, aShfil     ; "SHFil"
0A4CC75 478 call    sub_A3EDD0
0A4CC7A 478 lea     rcx, [rbp+String1] ; lpString1
0A4CC7E 478 lea     rdx, aEoper     ; "eOper"
0A4CC85 478 call    lstrcatA
0A4CC8A 478 lea     rcx, [rbp+String1] ; lpString1
0A4CC8E 478 lea     rdx, aAtionw    ; "ationW"
0A4CC95 478 call    lstrcatA
0A4CC9A 478 mov     rcx, rsi        ; hModule
0A4CC9D 478 lea     rdx, [rbp+String1] ; lpProcName
0A4CCA1 478 call    GetProcAddress_1
0A4CCA6 478 mov     rsi, rax
0A4CCA9 478 lea     rcx, [rbp+var_s428]
0A4CCB0 478 mov     edx, 38h ; '8'
0A4CCB6 478 xor     r8, r8
0A4CCB9 478 call    sub_40B630
0A4CCBE 478 mov     [rbp+var_s430], 3
0A4CCC8 478 mov     [rbp+var_s448], 14h
0A4CCD1 478 mov     [rbp+var_s438], rbx
0A4CCD8 478 lea     rcx, [rbp+var_s428]
0A4CCDF 478 call    rsi
```

GetProcAddressを呼び出してAPIのアドレスを解決

- shell32.dll
- SHFileOperationA

GetProcAddress function

Retrieves the address of an exported function or variable from the specified dynamic-link library (DLL).

Syntax

```
FARPROC WINAPI GetProcAddress(
  _In_ HMODULE hModule,
  _In_ LPCSTR  lpProcName
);
```

Parameters

hModule [in]

A handle to the DLL module that contains the function or variable. The LoadLibrary, LoadLibraryEx, LoadPackagedLibrary, or GetModuleHandle function returns this handle.

The **GetProcAddress** function does not retrieve addresses from modules that were loaded using the **LOAD_LIBRARY_AS_DATAFILE** flag. For more information, see LoadLibraryEx.

lpProcName [in]

The function or variable name, or the function's ordinal value. If this parameter is an ordinal value, it must be in the low-order word; the high-order word must be zero.

Return value

If the function succeeds, the return value is the address of the exported function or variable.

If the function fails, the return value is NULL. To get extended error information, call GetLastError.

1-5 呼び出されるAPIを確認する

SHFileOperationA function

12/05/2018 • 5 minutes to read

Copies, moves, renames, or deletes a file system object. This function has been replaced in Windows Vista by IFileOperation.

Syntax

C++ Copy

```
int SHFileOperationA(
  LPSHFILEOPSTRUCTA lpFileOp
);
```

Parameters

`lpFileOp`

Type: **LPSHFILEOPSTRUCT**

A pointer to an SHFILEOPSTRUCT structure that contains information this function needs to carry out the specified operation. This parameter must contain a valid value that is not **NULL**. You are responsible for validating the value. If you do not validate it, you will experience unexpected results.

Return Value

Type: **int**

Returns zero if successful; otherwise nonzero. Applications normally should simply check for zero or nonzero.

SHFILEOPSTRUCTA structure

2018/12/05 • 読了までの所要時間: 7 分

Contains information that the SHFileOperation function uses to perform file operations.

> **Note** As of Windows Vista, the use of the **IFileOperation** interface is recommended over this function.

Syntax

C++ コピー

```
typedef struct _SHFILEOPSTRUCTA {
  HWND         hwnd;
  UINT         wFunc;
  PCZZSTR      pFrom;
  PCZZSTR      pTo;
  FILEOP_FLAGS fFlags;
  BOOL         fAnyOperationsAborted;
  LPVOID       hNameMappings;
  PCSTR        lpszProgressTitle;
} SHFILEOPSTRUCTA, *LPSHFILEOPSTRUCTA;
```

1-5 解答

wFuncにFO_DELETE、fFlagsにFOF_SILENTとFOF_NOCONFIRMATIONを指定してSHFileOperationAを呼び出すことによりダイアログを出さずにファイルを削除する

```
.text:0000000000A4CCB6 478 xor     r8, r8
.text:0000000000A4CCB9 478 call    sub_40B630
.text:0000000000A4CCBE 478 mov     [rbp+SHFILEOPSTRUCTA.wFunc], 3
.text:0000000000A4CCC8 478 mov     word ptr [rbp+SHFILEOPSTRUCTA.fFlags], 14h
.text:0000000000A4CCD1 478 mov     [rbp+SHFILEOPSTRUCTA.pFrom], rbx
.text:0000000000A4CCD8 478 lea     rcx, [rbp+SHFILEOPSTRUCTA]
.text:0000000000A4CCDF 478 call    rsi             ; SHFileOperationA
.text:0000000000A4CCE1 478 test    eax, eax
```

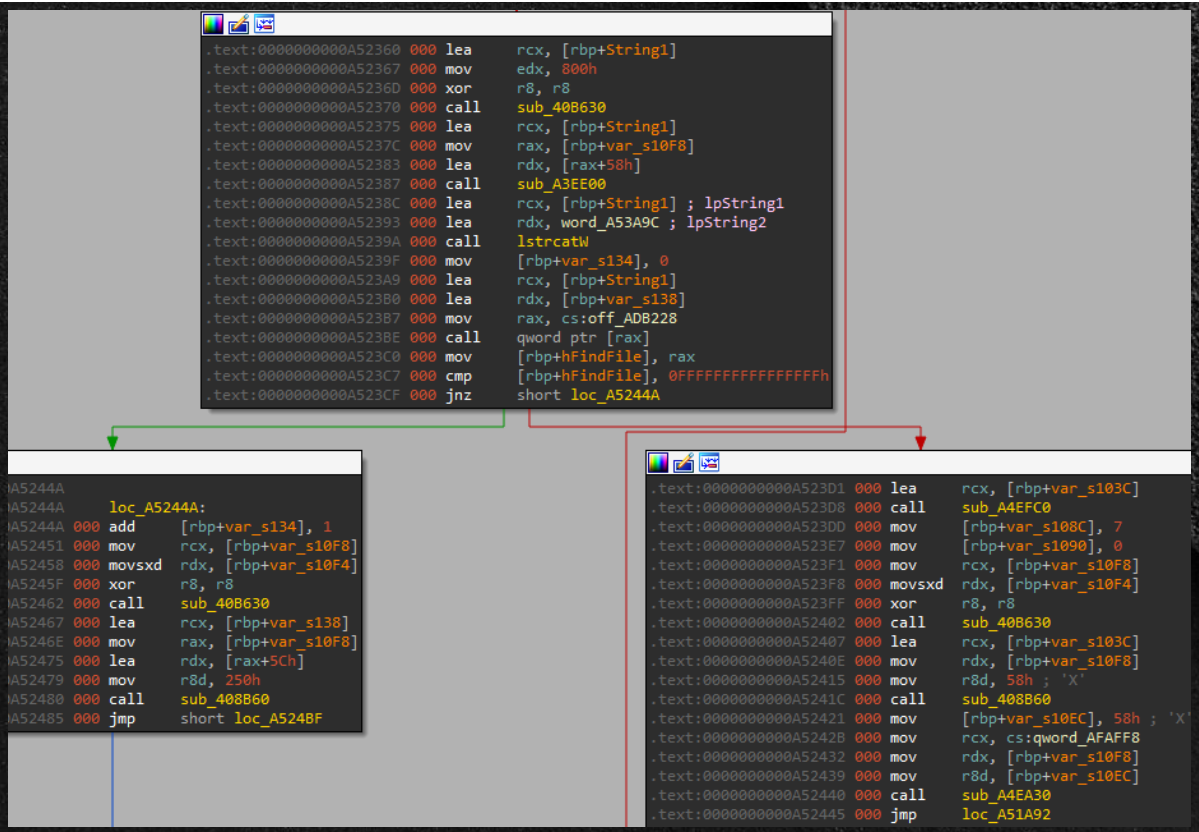
```
164  /****************************************
165  * SHFileOperation
166  */
167  #define FO_MOVE           0x0001
168  #define FO_COPY           0x0002
169  #define FO_DELETE         0x0003
170  #define FO_RENAME         0x0004
```

https://github.com/wine-mirror/wine/blob/master/include/shellapi.h

```
172  #define FOF_MULTIDESTFILES         0x0001
173  #define FOF_CONFIRMMOUSE           0x0002
174  #define FOF_SILENT                 0x0004
175  #define FOF_RENAMEONCOLLISION      0x0008
176  #define FOF_NOCONFIRMATION         0x0010
177  #define FOF_WANTMAPPINGHANDLE      0x0020
178  #define FOF_ALLOWUNDO              0x0040
179  #define FOF_FILESONLY              0x0080
180  #define FOF_SIMPLEPROGRESS         0x0100
181  #define FOF_NOCONFIRMMKDIR         0x0200
182  #define FOF_NOERRORUI              0x0400
183  #define FOF_NOCOPYSECURITYATTRIBS  0x0800
184  #define FOF_NORECURSION            0x1000  /* don't do recursion into directories */
185  #define FOF_NO_CONNECTED_ELEMENTS  0x2000  /* don't do connected files */
186  #define FOF_WANTNUKEWARNING        0x4000  /* during delete operation, warn if delete instead
187                                                of recycling (even if FOF_NOCONFIRMATION) */
188  #define FOF_NORECURSEREPARSE       0x8000  /* don't do recursion into reparse points */
189  #define FOF_NO_UI                  (FOF_SILENT | FOF_NOCONFIRMATION | FOF_NOERRORUI | FOF_NOCONFIRMMKDIR)
```

1-6 マルウェアの挙動を理解する

1-6. アドレス 0xA52360 から 0xA52578 までのコードがどのような処理をするか答えよ。（3点）



1-6 呼び出されるアドレスを確認する

```
.text:0000000000A5238C 000 lea     rcx, [rbp+String1] ; lpString1
.text:0000000000A52393 000 lea     rdx, word_A53A9C ; lpString2
.text:0000000000A5239A 000 call    lstrcatW
.text:0000000000A5239F 000 mov     [rbp+var_s134], 0
.text:0000000000A523A9 000 lea     rcx, [rbp+String1]
.text:0000000000A523B0 000 lea     rdx, [rbp+var_s138]
.text:0000000000A523B7 000 mov     rax, cs:off_ADB228
.text:0000000000A523BE 000 call    qword ptr [rax]
.text:0000000000A523C0 000 mov     [rbp+hFindFile], rax
```

```
000ADB228     off_ADB228 dq offset FindFirstFileW_1
000ADB230     off_ADB230 dq offset off_420DE0
```

```
.text:0000000000A524BF
.text:0000000000A524BF     loc_A524BF:
.text:0000000000A524BF 000 mov     rcx, [rbp+hFindFile]
.text:0000000000A524C6 000 lea     rdx, [rbp+var_s138]
.text:0000000000A524CD 000 mov     rax, cs:off_ADAD68
.text:0000000000A524D4 000 call    qword ptr [rax]
.text:0000000000A524D6 000 test    eax, eax
.text:0000000000A524D8 000 setnz   al
.text:0000000000A524DB 000 cmp     al, 1
.text:0000000000A524DE 000 jz      short loc_A52487
```

```
.data:0000000000ADAD58     off_ADAD58 dq offset byte_AFA838
.data:0000000000ADAD60     off_ADAD60 dq offset off_420F00
.data:0000000000ADAD68     off_ADAD68 dq offset FindNextFileW
.data:0000000000ADAD70     off_ADAD70 dq offset off_420DA0
.data:0000000000ADAD78     off_ADAD78 dq offset qword_A7AD20
.data:0000000000ADAD80     off_ADAD80 dq offset byte_AE5F84
```

1-6 呼び出されるAPIを確認する

FindFirstFileW function

12/05/2018 • 5 minutes to read

Searches a directory for a file or subdirectory with a name that matches a specific name (or partial name if wildcards are used).

To specify additional attributes to use in a search, use the FindFirstFileEx function.

To perform this operation as a transacted operation, use the FindFirstFileTransacted function.

Syntax

C++ Copy

```
HANDLE FindFirstFileW(
  LPCWSTR            lpFileName,
  LPWIN32_FIND_DATAW lpFindFileData
);
```

Parameters

`lpFileName`

The directory or path, and the file name. The file name can include wildcard characters, for example, an asterisk (*) or a question mark (?).

This parameter should not be **NULL**, an invalid string (for example, an empty string or a string that is missing the terminating null character), or end in a trailing backslash ().

If the string ends with a wildcard, period (.), or directory name, the user must have access permissions to the root and all subdirectories on the path.

In the ANSI version of this function, the name is limited to **MAX_PATH** characters. To extend this limit to 32,767 wide characters, call the Unicode version of the function and prepend "\?" to the path. For more information, see Naming a File.

Tip Starting in Windows 10, version 1607, for the unicode version of this function (**FindFirstFileW**), you can opt-in to remove the **MAX_PATH** character limitation without prepending "\\?\". See the "Maximum Path Limitation" section of **Naming Files, Paths, and Namespaces** for details.

`lpFindFileData`

A pointer to the WIN32_FIND_DATA structure that receives information about a found file or directory.

FindNextFileW function

12/05/2018 • 2 minutes to read

Continues a file search from a previous call to the FindFirstFile, FindFirstFileEx, or FindFirstFileTransacted functions.

Syntax

C++ Copy

```
BOOL FindNextFileW(
  HANDLE             hFindFile,
  LPWIN32_FIND_DATAW lpFindFileData
);
```

Parameters

`hFindFile`

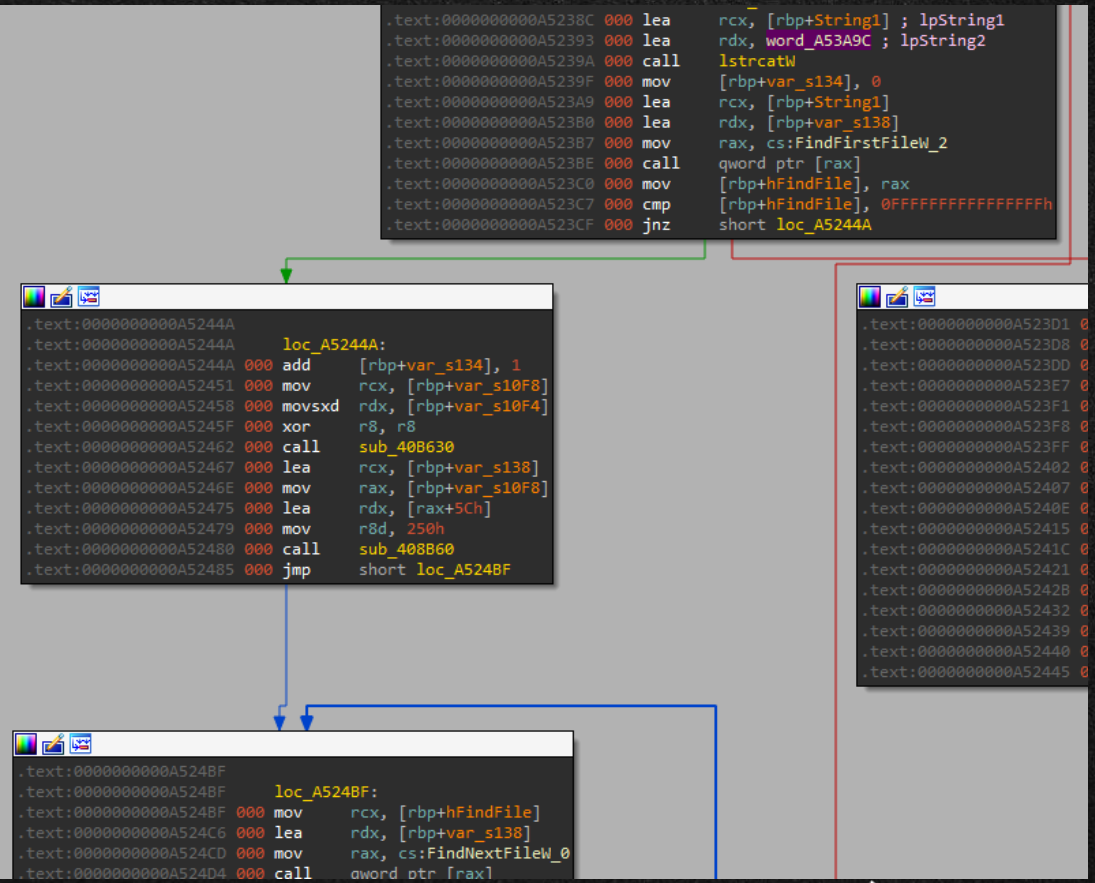
The search handle returned by a previous call to the FindFirstFile or FindFirstFileEx function.

`lpFindFileData`

A pointer to the WIN32_FIND_DATA structure that receives information about the found file or subdirectory.

berDefense

1-6 解答



ワイルドカードを指定して
FindFirstFileWとFindNextFileWを
呼び出してファイルの一覧を取得
する

2-1 アルゴリズムを理解する

2-1. sub_A42D60 は通常とは異なる変換テーブルを使用した Base64 デコード処理を行う関数である。sub_A42D60 を解析し、その変換テーブルとパディング時に使用される記号（通常は "=" が使用される）を答えよ。（3点）

```
.text:0000000000A42D60     ; __unwind { // sub_40D9A0
.text:0000000000A42D60 000 push    rbp
.text:0000000000A42D61 008 push    r14
.text:0000000000A42D63 010 push    r13
.text:0000000000A42D65 018 push    rdi
.text:0000000000A42D66 020 push    rsi
.text:0000000000A42D67 028 push    rbx
.text:0000000000A42D68 030 sub     rsp, 0C8h
.text:0000000000A42D6F 0F8 mov     rbp, rsp
.text:0000000000A42D72 0F8 mov     [rbp+var_s20], rdx
.text:0000000000A42D76 0F8 mov     [rbp+var_sB0], 0
.text:0000000000A42D81 0F8 mov     [rbp+arg_0], rcx
.text:0000000000A42D88 0F8 mov     rcx, [rbp+arg_0]
.text:0000000000A42D8F 0F8 call    sub_40F510
.text:0000000000A42D94 0F8 nop
.text:0000000000A42D95 0F8 lea     rcx, [rbp+var_s30]
.text:0000000000A42D99 0F8 mov     edx, 80h
.text:0000000000A42D9F 0F8 xor     r8, r8
.text:0000000000A42DA2 0F8 call    sub_40B630
.text:0000000000A42DA7 0F8 lea     rcx, [rbp+var_s30]
.text:0000000000A42DAB 0F8 lea     rdx, aPzl       ; "PzL"
.text:0000000000A42DB2 0F8 call    sub_A3EDD0
.text:0000000000A42DB7 0F8 lea     rcx, [rbp+var_s30]
.text:0000000000A42DBB 0F8 lea     rdx, aS3w       ; "S3w"
.text:0000000000A42DC2 0F8 call    sub_A3EE30
.text:0000000000A42DC7 0F8 lea     rcx, [rbp+var_s30]
.text:0000000000A42DCB 0F8 lea     rdx, aLhe       ; "lhe"
.text:0000000000A42DD2 0F8 call    sub_A3EE30
.text:0000000000A42DD7 0F8 lea     rcx, [rbp+var_s30]
.text:0000000000A42DDB 0F8 lea     rdx, aMan       ; "mAn"
.text:0000000000A42DE2 0F8 call    sub_A3EE30
.text:0000000000A42DE7 0F8 lea     rcx, [rbp+var_s30]
.text:0000000000A42DEB 0F8 lea     rdx, aFxs       ; "fxs"
.text:0000000000A42DF2 0F8 call    sub_A3EE30
.text:0000000000A42DF7 0F8 lea     rcx, [rbp+var_s30]
.text:0000000000A42DFB 0F8 lea     rdx, aUign      ; "UiGN"
```

2-1 解答

変換テーブル: PzLS3wlhemAnfxsUiGNOgT5R6aH9^`KyD8Cocd04EFIvWBQJZXpj1qtu2rkb7VMY!
パディングに使用する文字は変換テーブル最後の"!"

```
.text:0000000000A43141      algn_A43141:
.text:0000000000A43141      align 4
.text:0000000000A43144      aPzl db 'PzL',0
.text:0000000000A43148      aS3w db 'S3w',0
.text:0000000000A4314C      aLhe db 'lhe',0
.text:0000000000A43150      aMan db 'mAn',0
.text:0000000000A43154      aFxs db 'fxs',0
.text:0000000000A43158      aUign db 'UiGN',0
.text:0000000000A4315D      aOgt db 'OgT',0
.text:0000000000A43161      a5r6 db '5R6',0
.text:0000000000A43165      aAh9 db 'aH9',0
.text:0000000000A43169      aK_5 db '^`K',0
.text:0000000000A4316D      aYd8 db 'yD8',0
.text:0000000000A43171      aCoc db 'Coc',0
.text:0000000000A43175      aD04 db 'd04',0
.text:0000000000A43179      aEfi db 'EFI',0
.text:0000000000A4317D      aVwb db 'vWB',0
.text:0000000000A43181      aQjz db 'QJZ',0
.text:0000000000A43185      aXpj db 'Xpj',0
.text:0000000000A43189      a1qt db '1qt',0
.text:0000000000A4318D      aU2r db 'u2r',0
.text:0000000000A43191      aKb7 db 'kb7',0
.text:0000000000A43195      aVmy db 'VMY',0
.text:0000000000A43199      asc_A43199 db '!',0
.text:0000000000A4319B      align 20h
.text:0000000000A431A0
```

PzLS3wlhemAnfxsUiGNOgT5R6aH9^`KyD
8Cocd04EFIvWBQJZXpj1qtu2rkb7VMY!

2-2 アルゴリズムを理解する

2-2. 関数 sub_A40450 は、本来の RC4 実装に一点だけ変更が加えられている。その変更点は何か述べよ。（3点）

```
.text:0000000000A40450      sub_A40450 proc near
.text:0000000000A40450
.text:0000000000A40450      var_s2E= byte ptr  2Eh
.text:0000000000A40450      var_s12E= byte ptr  12Eh
.text:0000000000A40450      var_s12F= byte ptr  12Fh
.text:0000000000A40450
.text:0000000000A40450 000  push    rbp
.text:0000000000A40451 008  push    rdi
.text:0000000000A40452 010  push    rsi
.text:0000000000A40453 018  push    rbx
.text:0000000000A40454 020  sub     rsp, 138h
.text:0000000000A4045B 158  mov     rbp, rsp
.text:0000000000A4045E 158  mov     rbx, rcx
.text:0000000000A40461 158  mov     rsi, rdx
.text:0000000000A40464 158  mov     edi, r8d
.text:0000000000A40467 158  mov     rcx, rbp
.text:0000000000A4046A 158  lea     rdx, [rbp+var_s2E]
.text:0000000000A4046E 158  mov     r8, r9
.text:0000000000A40471 158  call    sub_A3EEA0
.text:0000000000A40476      ; junk
.text:0000000000A40582 158  xor     rcx, rcx
.text:0000000000A40585 158  mov     r9d, edi
.text:0000000000A40588 158  sub     r9d, 1
.text:0000000000A4058C 158  cmp     ecx, r9d
.text:0000000000A4058F 158  jg      loc_A41118
.text:0000000000A40595 158  add     r9d, 1
.text:0000000000A40599
.text:0000000000A40599      loc_A40599:
.text:0000000000A40599 158  lea     rax, [rbp+var_s12E]
.text:0000000000A405A0 158  add     byte ptr [rax], 1
.text:0000000000A405A3      ; junk
.text:0000000000A406D6 158  movzx   rax, [rbp+var_s12E]
.text:0000000000A406DE 158  movzx   r8, byte ptr [rax+rbp+2Eh]
.text:0000000000A406E4      ; junk
```

2-2 RC4

伝統と格式のRC4問題！
問題のマルウェアファミリを変更しても課題2では毎年出題されている

Daserf

Oni

Plead

2-2 開催中に提示したヒント

```
__int64 __fastcall sub_A40450(__int64 a1, __int64 a2, int a3, __int64 a4)
{
  __int64 v4; // rbx
  __int64 v5; // rsi
  int v6; // edi
  int v7; // ecx
  char v8; // r8
  __int64 _0; // [rsp+0h] [rbp+0h]
  char vars2E[256]; // [rsp+2Eh] [rbp+2Eh]
  unsigned __int8 vars12E; // [rsp+12Eh] [rbp+12Eh]
  unsigned __int8 vars12F; // [rsp+12Fh] [rbp+12Fh]

  v4 = a1;
  v5 = a2;
  v6 = a3;
  sub_A3EEA0((__int64)&_0, (__int64)vars2E, a4);
  v7 = 0;
  if ( v6 - 1 >= 0 )
  {
    do
    {
      v8 = vars2E[++vars12E];
      vars12F += v8;
      vars2E[vars12E] = vars2E[vars12F];
      vars2E[vars12F] = v8;
      *(_BYTE *)(v5 + v7) = vars2E[(unsigned __int8)(vars2E[vars12E] + v8)] ^ *(_BYTE *)(v4 + v7);
      ++v7;
    }
    while ( v7 != v6 );
  }
  return sub_A40430(&_0, vars2E);
}
```

```
v3 = 0;
if ( a3 )
  v3 = *(_DWORD *)(a3 - 4);
*(_BYTE *)(a2 + 256) = 0;
*(_BYTE *)(a2 + 257) = 0;
v4 = 0;
do
{
  *(_BYTE *)(a2 + v4) = v4 + 5;
  ++v4;
}
while ( v4 );
v5 = 0;
v6 = 0;
v7 = 0;
do
{
  if ( v6 >= v3 )
    v8 = 0;
  else
    v8 = *(_BYTE *)(a3 + v6);
  if ( ++v6 >= v3 )
    v6 = 0;
  v5 += v8 + *(_BYTE *)(a2 + v7);
  v9 = *(_BYTE *)(a2 + v7);
  *(_BYTE *)(a2 + v7) = *(_BYTE *)(a2 + v5);
  result = v5;
  *(_BYTE *)(a2 + v5) = v9;
  ++v7;
}
while ( v7 );
return result;
}
```

2-2 解答

オープンソースのARC4の
ソースコードと比べてみる

```
29  static int arc4_set_key(struct crypto_tfm *tfm, const u8 *in_key,
30                          unsigned int key_len)
31  {
32          struct arc4_ctx *ctx = crypto_tfm_ctx(tfm);
33          int i, j = 0, k = 0;
34
35          ctx->x = 1;
36          ctx->y = 0;
37
38          for (i = 0; i < 256; i++)
39                  ctx->S[i] = i;
40
41          for (i = 0; i < 256; i++) {
42                  u32 a = ctx->S[i];
43                  j = (j + in_key[k] + a) & 0xff;
44                  ctx->S[i] = ctx->S[j];
45                  ctx->S[j] = a;
46                  if (++k >= key_len)
47                          k = 0;
48          }
49
50          return 0;
51  }
```

https://elixir.bootlin.com/linux/v4.1/source/crypto/arc4.c

```
v3 = 0;
if ( a3 )
  v3 = *(_DWORD *)(a3 - 4);
*(_BYTE *)(a2 + 256) = 0;
*(_BYTE *)(a2 + 257) = 0;
v4 = 0;
do
{
  *(_BYTE *)(a2 + v4) = v4 + 5;
  ++v4;
}
while ( v4 );
v5 = 0;
v6 = 0;
v7 = 0;
do
{
  if ( v6 >= v3 )
    v8 = 0;
  else
    v8 = *(_BYTE *)(a3 + v6);
  if ( ++v6 >= v3 )
    v6 = 0;
  v5 += v8 + *(_BYTE *)(a2 + v7);
  v9 = *(_BYTE *)(a2 + v7);
  *(_BYTE *)(a2 + v7) = *(_BYTE *)(a2 + v5);
  result = v5;
  *(_BYTE *)(a2 + v5) = v9;
  ++v7;
}
while ( v7 );
return result;
}
```

2-2 解答

- 配列の初期化時に値を5ずらした配列を作っている

通常のRC4 [0, 1 … 255]

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61,
 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91
, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140,
 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164
, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 18
8, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 2
12, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235,
236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255]
```

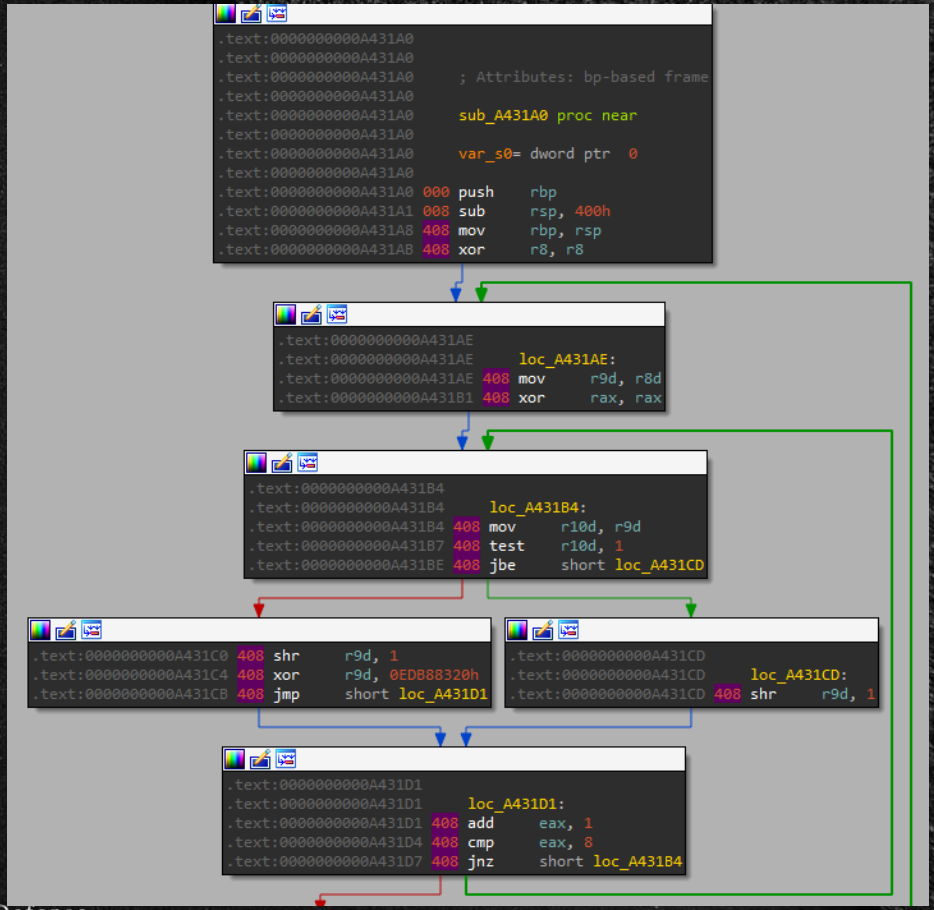
問題のカスタムRC 4 [5, 6 … 255, 0 … 4]

```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65
, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 9
5, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119,
120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167
, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 19
1, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 2
15, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238,
239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 0, 1, 2, 3, 4]
```

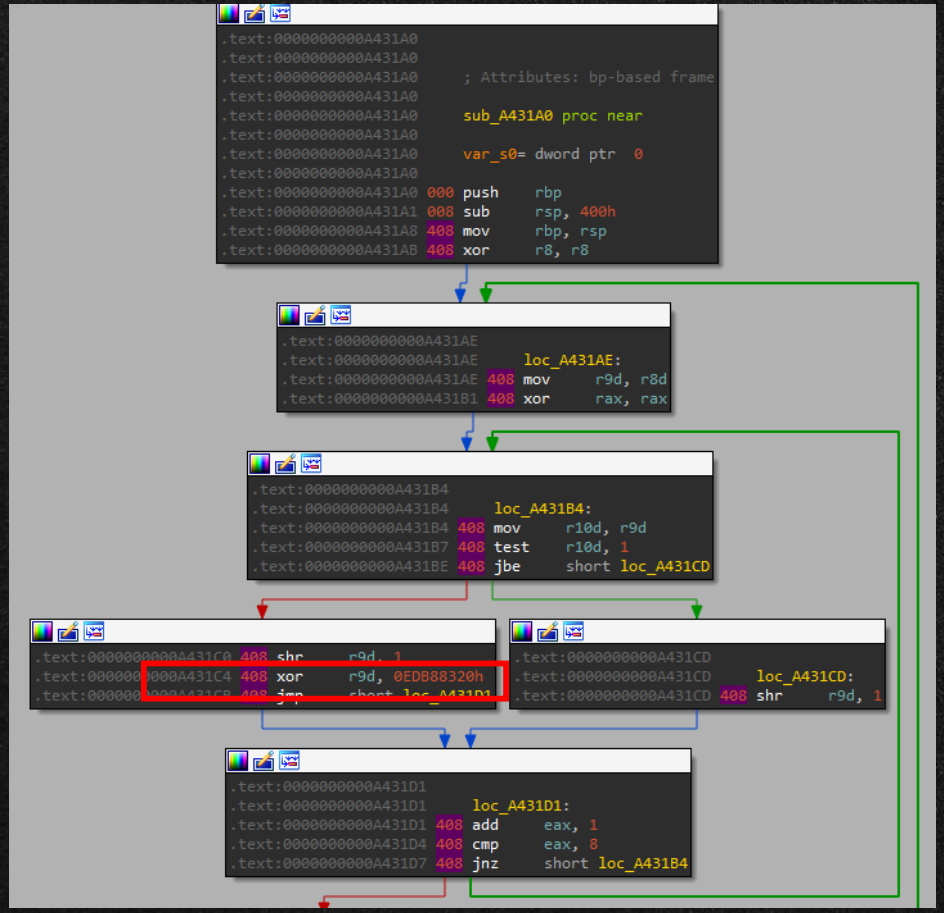
2-3 アルゴリズムを理解する

2-3. sub_A431A0 は、与えられたデータに対してある計算を行った結果を返す関数である。sub_A431A0 を解析し、何を返す関数かを特定せよ。（2点）

- ○ MD5 ハッシュ値
- ○ SHA-1 ハッシュ値
- ○ SHA-256 ハッシュ値
- ○ CRC32 チェックサム

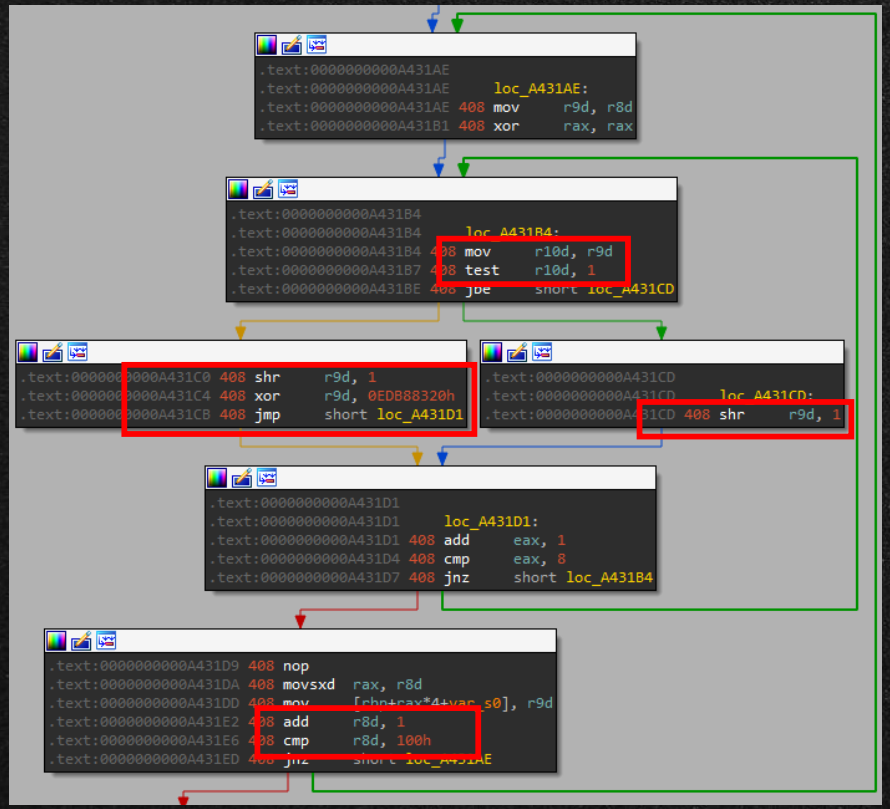


2-3 固定値



怪しい固定値のxorに注目

2-3 アルゴリズムの確認



実装例 [編集]

CRC-32 [編集]

C言語での実装例。RFC 1952 (gzip) や RFC 2083 (PNG) の末尾にも実装例が載っている。zlib などにも含まれている。

```
uint32_t crc_table[256];

void make_crc_table(void) {
    for (uint32_t i = 0; i < 256; i++) {
        uint32_t c = i;
        for (int j = 0; j < 8; j++) {
            c = (c & 1) ? (0xEDB88320 ^ (c >> 1)) : (c >> 1);
        }
        crc_table[i] = c;
    }
}

uint32_t crc32(uint8_t *buf, size_t len) {
    uint32_t c = 0xFFFFFFFF;
    for (size_t i = 0; i < len; i++) {
        c = crc_table[(c ^ buf[i]) & 0xFF] ^ (c >> 8);
    }
    return c ^ 0xFFFFFFFF;
}

int main(void)
{
    make_crc_table();
    uint32_t crc = crc32();

    return 0;
}
```

2-3 解答

2-3. sub_A431A0 は、与えられたデータに対してある計算を行った結果を返す関数である。sub_A431A0 を解析し、何を返す関数かを特定せよ。（2点）

- ○ MD5 ハッシュ値
- ○ SHA-1 ハッシュ値
- ○ SHA-256 ハッシュ値
- ● CRC32 チェックサム

3-1 マルウェアを特定する

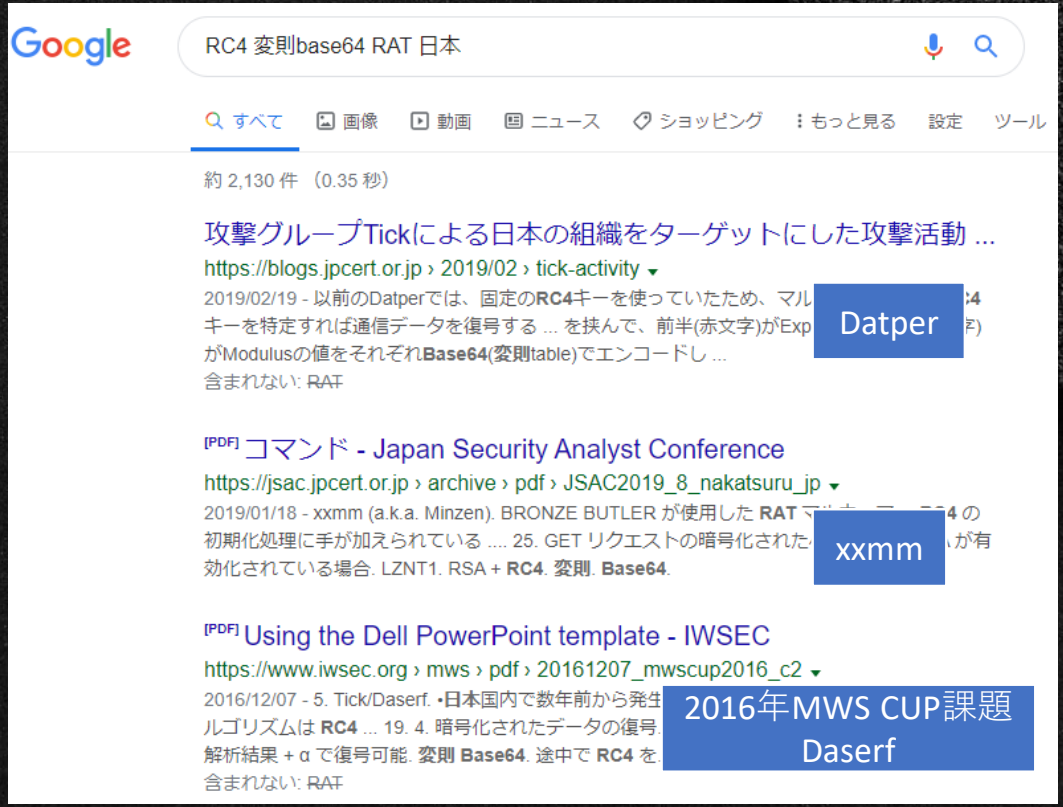
3-1. これまでの回答やその他の解析結果を元に、当該検体の名称（Emdivi、Daserf、Oni、Plead 等）を答えよ。（2点）

出題意図

- マルウェアの特徴を基に公開情報を活用できるか

3-1 OSINT(Google)

これまでの解析結果を基にOSINTして候補を絞る



3-1 APIデコード処理に注目する 1

APIのデコード処理

```
0A41178 C68 lea     rcx, [rbp+String1] ; lpString1
0A4117F C68 lea     rdx, String2       ; "5j2Cn"
0A41186 C68 call    lstrcpyA
0A4118B C68 lea     rcx, [rbp+var_s40] ; lpString1
0A4118F C68 lea     rdx, aX4hgxh5      ; "x4hgxh5"
0A41196 C68 call    lstrcpyA
0A4119B C68 lea     rcx, [rbp+String1] ; lpString1
0A411A2 C68 lea     rdx, off_A414F2 ; lpString2
0A411A9 C68 call    lstrcatA
0A411AE C68 lea     rcx, [rbp+String1] ; lpString1
0A411B5 C68 lea     rdx, asc_A414F6 ; "^@$"
0A411BC C68 call    lstrcatA
0A411C1 C68 lea     rcx, [rbp+String1] ; lpString1
0A411C8 C68 lea     rdx, asc_A414FA ; "*(]{}"
0A411CF C68 call    lstrcatA
0A411D4 C68 lea     rcx, [rbp+var_s40] ; lpString1
0A411D8 C68 lea     rdx, aLvnr57       ; "lvnr57"
0A411DF C68 call    lstrcatA
0A411E4 C68 lea     rcx, [rbp+String1] ; lpString1
0A411EB C68 lea     rdx, aMxK3d        ; "|+mX k3D"
0A411F2 C68 call    lstrcatA
0A411F7 C68 lea     rcx, [rbp+String1] ; lpString1
0A411FE C68 lea     rdx, aKLgc         ; "K'LGc!"
0A41205 C68 call    lstrcatA
0A4120A C68 lea     rcx, [rbp+String1] ; lpString1
0A41211 C68 lea     rdx, aHhnpgz       ; "hHNPgZ"
0A41218 C68 call    lstrcatA
0A4121D C68 lea     rcx, [rbp+String1] ; lpString1
0A41224 C68 lea     rdx, aZ0t8         ; ",z0T8"
0A4122B C68 call    lstrcatA
0A41230 C68 lea     rcx, [rbp+var_s40] ; lpString1
0A41234 C68 lea     rdx, aCng2hs       ; "cng2Hs"
```

生成された文字列を用いて

5j2Cnx`^@$*(]{}|+mX k3DK'LGc!hHNPgZ,z0T8_sRU7)<>"[lBpdfI#%bu;yt-YeoW?4vAMQVa.6qJi:=wFO9&/1ESr

Daserf (SHA256: 01d681c51ad0c7c3d4b320973c61c28a353624ac665fd390553b364d17911f46)の亜種では、単純な置換暗号用の同じ置換
ールでも非常によく似たテーブルを見つけました。文字列はこれらの脅威に固有のものなので、このグループに関係する開発者がこれらの
とコードが共有されているため、Cyphortによって報告された攻撃はTickにつながりがあった可能性があります。以下のテーブルは、それぞ
定されました。

Minzen (SHA256:26727d139b593486237b975e7bdf93a8148c52d5fb48d5fe540a634a16a6ba82):

plain text = "5j2Cnx`^@$*(]{}|+mX k3DK'LGc!hHNPgZ,z0T8_sRU7)<>"[lBpdfI#%bu;yt-YeoW?4vAMQVa.6qJi:=wFO9&/1ESr"

cipher text = "KhL9V1ds5Z"QnfNC&Fb8xGr-()<>[]{}|+THce;0%7O!iz#W DE6qS?aw./BJlk,yUPjgl`^@$*tumYA'p2RoX=v_:M43"

Datper (SHA256: 7d70d659c421b50604ce3e0a1bf423ab7e54b9df361360933bac3bb852a31849):

[begin code]plain text = "KhL9V1ds5Z"QnfNC&Fb8xGr-()<>[]{}|+THce;0%7O!iz#W DE6qS?aw./BJlk,yUPjgl\`^@$*tumYA'p2RoX=v_:M43"

cipher text = "5j2Cnx\`^@$*(]{}|+mX k3DK'LGc!hHNPgZ,z0T8_sRU7)<>"[lBpdfI#%bu;yt-YeoW?4vAMQVa.6qJi:=wFO9&/1ESr" [end code]

https://www.paloaltonetworks.jp/company/in-the-news/2017/tick-continues-cyber-espionage-attacks

Minzen(XXMM) か Datper である可能性が高い

3-1 APIデコード処理に注目する 2

エンコードされたAPIの文字列を用いて

aa_get_proc_address_routine
aa_get_proc_address_routine_0

```
mov     cs:InternetOpenA, rax
lea     rcx, [rbp+var_s68]
lea     rdx, aAz3ZaFZAj ; "?(az3(za)F((z,aJ"
call    sub_A41140
mov     rcx, [rbp+var_s68]
call    sub_410590
mov     rcx, rbx        ; hModule
mov     rdx, rax        ; lpProcName
call    GetProcAddress_1
mov     cs:InternetConnectA, rax
lea     rcx, [rbp+var_s60]
lea     rdx, aZaaSZWzbZAj ; "Zaa:s:z(wzB.z`aJ"
call    sub_A41140
mov     rcx, [rbp+var_s60]
call    sub_410590
mov     rcx, rbx        ; hModule
mov     rdx, rax        ; lpProcName
call    GetProcAddress_1
mov     cs:HttpOpenRequestA, rax
lea     rcx, [rbp+var_s58]
lea     rdx, aZaaPzWzbZAj ; "Zaa:pz(¥wzB.z`aJ"
call    sub_A41140
mov     rcx, [rbp+var_s58]
call    sub_410590
mov     rcx, rbx        ; hModule
mov     rdx, rax        ; lpProcName
call    GetProcAddress_1
```

HYBRID ANALYSIS

Request Info

File Details

All Details: Off

7e3006f77d59c4b7ad9a8b6542a7d634f16246b5.rl

Filename	7e3006f77d59c4b7ad9a8b6542a7d634f16246b5.rl
Size	137KiB (139776 bytes)
Type	peexe executable
Description	PE32 executable (GUI) Intel 80386, for MS Windows
Architecture	WINDOWS
SHA256	2f6745ccebf8e1d9e3e5284a895206bbb4347cf7daa2371652423aa9b94dfd3d
Compiler/Packer	Borland Del

VIRUSTOTAL

SUMMARY DETECTION DETAILS RELATIONS

Ikarus	Trojan-Downloader.Agent
Jiangmin	Trojan.Datper.g
K7AntiVirus	Trojan (7000000f1)
K7GW	Trojan (7000000f1)
Kaspersky	HEUR:Trojan.Win32.Datper.gen
MAX	Malware (ai Score=100)
MaxSecure	Trojan.Malware.73854553.susgen

CyberDefense

3-1 コンフィグ

- 1-1で設定されていた時刻はコンフィグからロードされている
- aa_load_configの処理を解析するとコンフィグの各設定が|||で区切られていることが判明する

xrefs to dword_AFB030

Direction	Typ	Address	Text	
Up	w	aa_load_config+37B	mov	cs:dword_AFB030, eax
	r	aa_main_thread+B8C	cmp	eax, cs:dword_AFB030

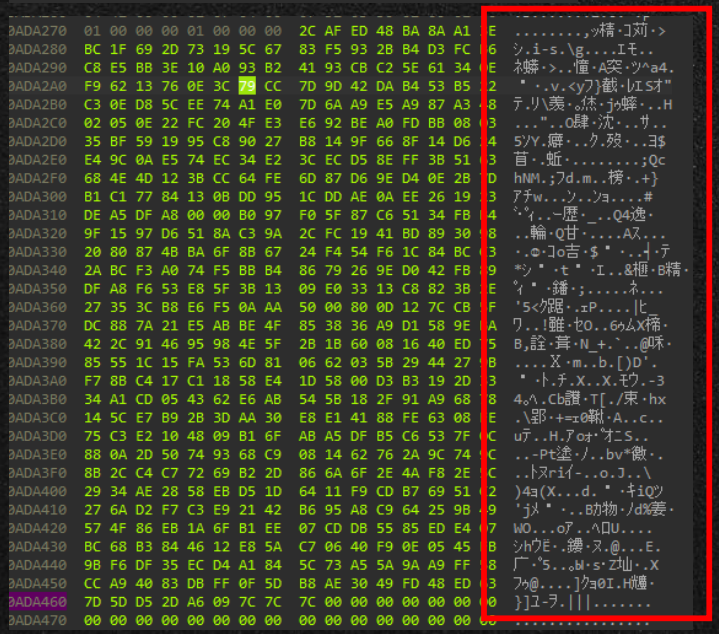
```
0A50D2D C98 call    sub_A423D0
0A50D32 C98 mov     rcx, [rbp+var_s40]
0A50D36 C98 call    sub_A42900
0A50D3B C98 mov     cs:dword_AFB030, eax
0A50D41 C98 lea     rcx, [rbp+var_s38]
0A50D45 C98 mov     rdx, [rbp+var_sC88]
0A50D4C C98 lea     r8, asc_A50F24  ; "|||"
0A50D53 C98 mov     r9d, 0Ah
0A50D5A C98 call    sub_A423D0
0A50D5F C98 mov     rcx, [rbp+var_s38]
0A50D63 C98 call    sub_A42900
0A50D68 C98 mov     cs:dword_AFB034, eax
0A50D6E C98 lea     rcx, [rbp+var_s30]
0A50D72 C98 mov     rdx, [rbp+var_sC88]
0A50D79 C98 lea     r8, asc_A50F24  ; "|||"
0A50D80 C98 mov     r9d, 0Bh
0A50D87 C98 call    sub_A423D0
0A50D8C C98 lea     rcx, qword_AFB038
0A50D93 C98 mov     rdx, [rbp+var_s30]
0A50D97 C98 call    sub_40FAB0
0A50D9C C98 lea     rcx, [rbp+var_s28]
0A50DA0 C98 mov     rdx, [rbp+var_sC88]
0A50DA7 C98 lea     r8, asc_A50F24  ; "|||"
0A50DAE C98 mov     r9d, 0Ch
```

3-1 コンフィグ

コンフィグの特徴はDatperと一致する

xrefs to unk_ADA278

Direction	Typ	Address	Text	
Up	o	aa_load_config+94	lea	rcx, unk_ADA278
Up	o	aa_load_config+BD	lea	rcx, unk_ADA278
Up	o	aa_load_config+CC	lea	rcx, unk_ADA278



日本と韓国を狙う標的型攻撃：The Bald Knight Rises

Kaspersky Labでは数多くの標的型攻撃を調査していますが、そのうちの一つが、日本と韓国を狙う非常に洗練された攻撃です。

Kaspersky Team　2018年2月2日

・Datper

XXMMとは別種のマルウェアです。ダウンローダーのWaliがダウンロードするところから、攻撃の第2段階で使われるバックドアと考えられます。また、DatperもXXMM同様にマルウェアの設定ファイルが暗号化されて内包されています。復号してみると、通信先や稼働時間等の設定が含まれており、この点においてもXXMMと共通していることが確認されています。

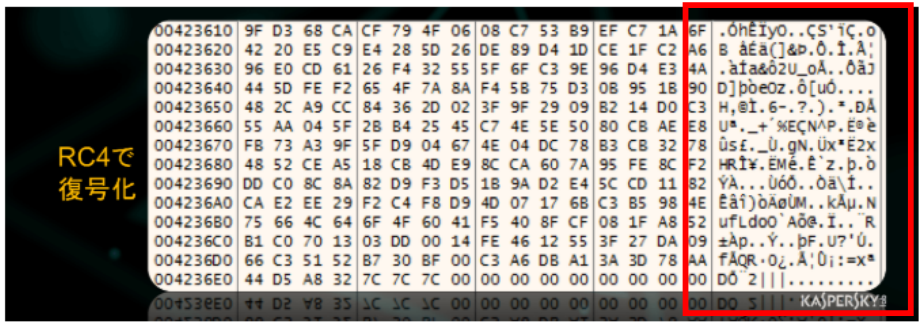


図4. 復号されたDatperの設定データ

https://blog.kaspersky.co.jp/the-bald-knight-rises-apt/19376/

3-1 コンフィグの復号

競技時間中でのコンフィグの復号は想定していないが
コンフィグの情報から検体の名称を特定することもできる

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	80	63	65	66	37	66	34	66	65	7C	7C	7C	68	74	74	70	€cef7f4fe|||http
00000010	3A	2F	2F	77	77	77	2E	61	72	6F	6D	61	74	69	63	74	://www.aromatict
00000020	72	65	65	2E	63	6F	2E	6B	72	2F	69	6E	63	6C	75	64	ree.co.kr/includ
00000030	65	2F	69	6E	64	65	78	2E	70	68	70	7C	7C	7C	32	34	e/index.php|||24
00000040	30	30	7C	7C	7C	64	34	66	79	33	79	6B	64	6B	32	64	00|||d4fy3ykdk2d
00000050	64	73	73	72	7C	7C	7C	4E	55	4C	4C	7C	7C	7C	4E	55	dssr|||NULL|||NU
00000060	4C	4C	7C	7C	7C	4E	55	4C	4C	7C	7C	7C	4E	55	4C	4C	LL|||NULL|||NULL
00000070	7C	7C	7C	30	7C	7C	7C	32	34	7C	7C	7C	4E	55	4C	4C	|||0|||24|||NULL
00000080	7C	7C	7C	4D	6F	7A	69	6C	6C	61	2F	35	2E	30	20	28	|||Mozilla/5.0 (
00000090	57	69	6E	64	6F	77	73	20	4E	54	20	36	2E	31	3B	20	Windows NT 6.1;
000000A0	57	4F	57	36	34	3B	20	54	72	69	64	65	6E	74	2F	37	WOW64; Trident/7
000000B0	2E	30	3B	20	72	76	3A	31	31	2E	30	29	20	6C	69	6B	.0; rv:11.0) lik
000000C0	65	20	47	65	63	6B	6F	7C	7C	7C	61	69	61	41	24	63	e Gecko|||aiaA$c
000000D0	73	68	30	68	35	38	38	32	41	2B	77	4E	6D	52	73	79	sh0h5882A+wNmRsy
000000E0	6B	6E	44	51	73	69	37	4C	61	36	49	54	3D	59	44	38	knDQsi7La6IT=YD8

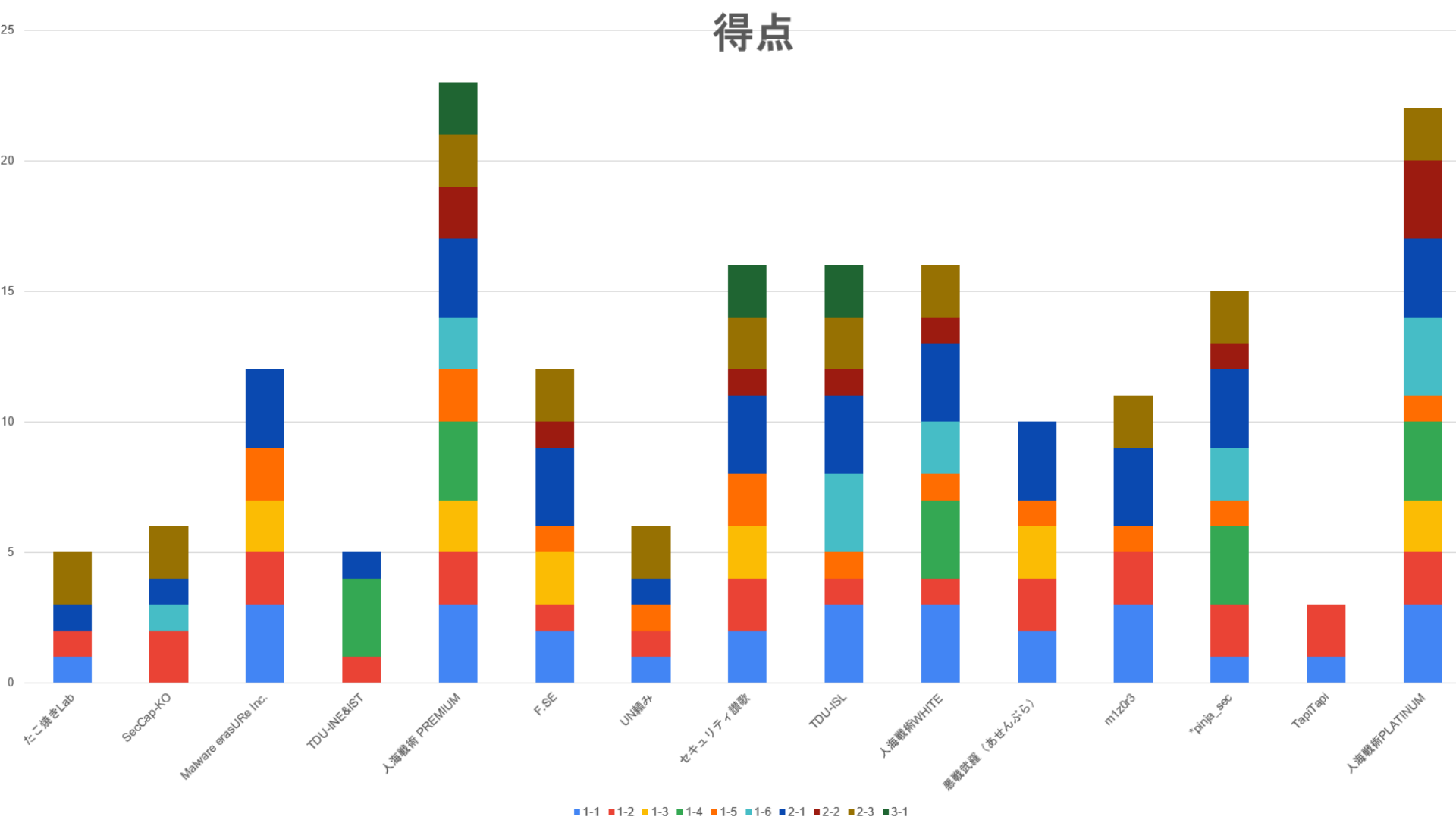
Appendix C: 通信先一覧

- www.rakutenline.com
- menu.rakutenline.com
- www.sa-guard.com
- menu.sa-guard.com
- www.han-game.com
- menu.han-game.com
- 211.233.81.242
- www.aromatictree.co.kr
- ro.thumbbay.com

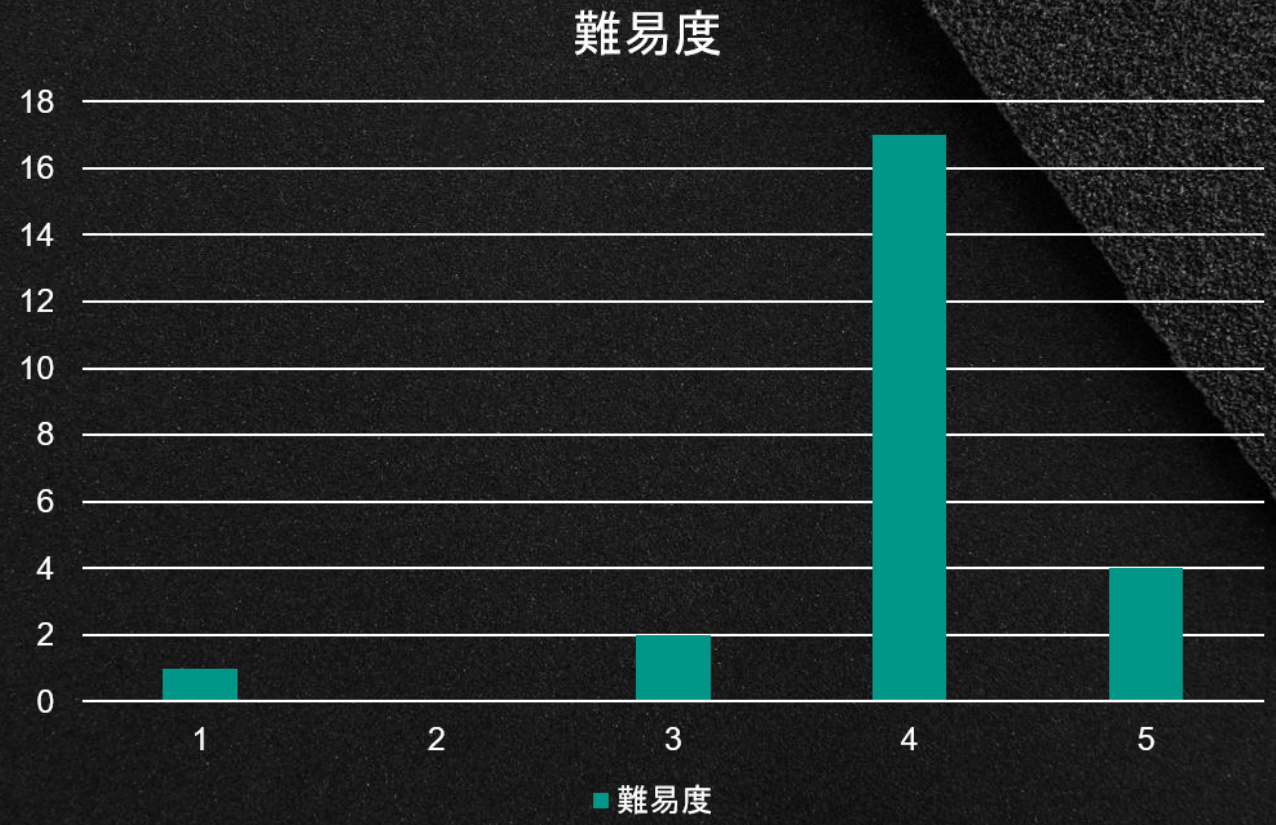
https://blogs.jpcert.or.jp/ja/2019/02/tick-activity.html

CyberDefense

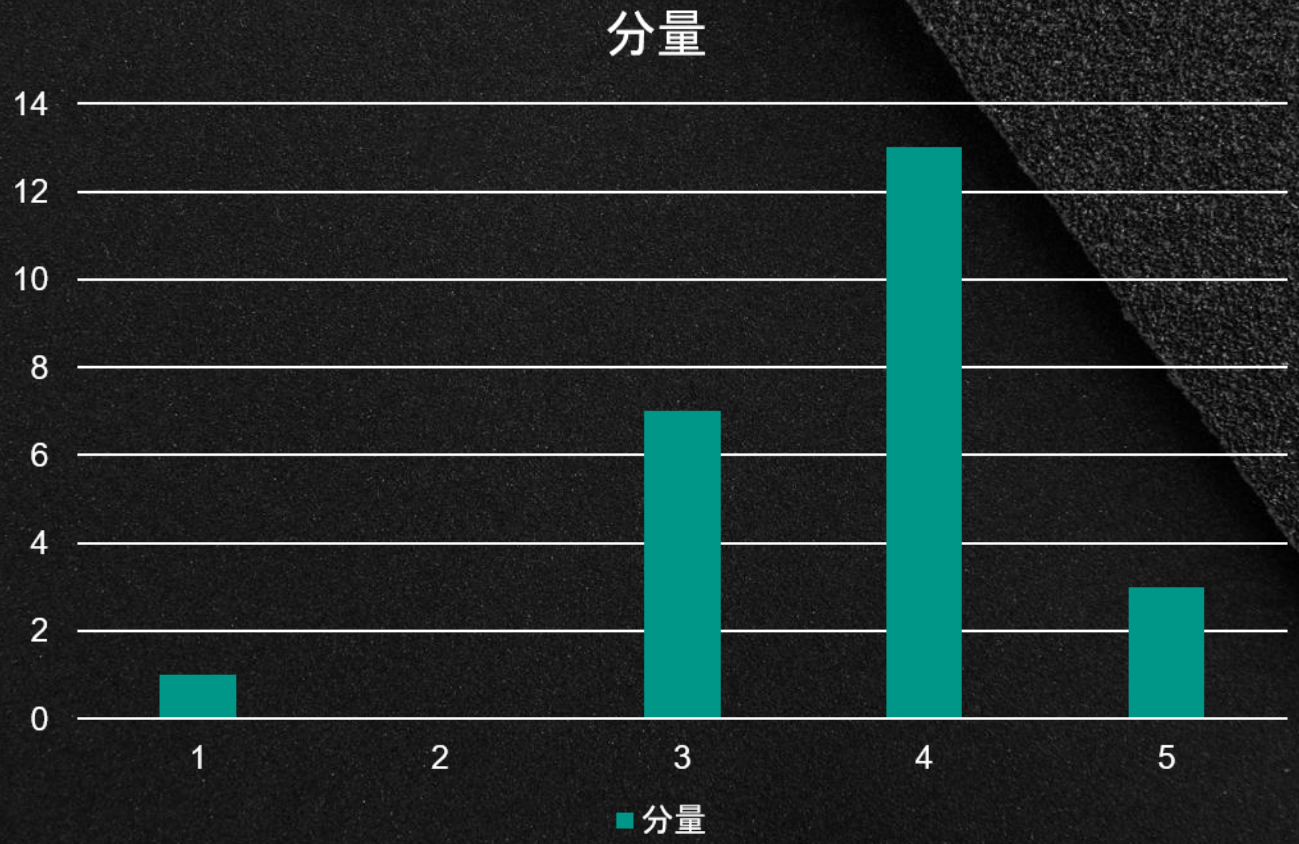
MWS Cup 2019
課題2 採点結果と振り返り


得点
25
20
15
10
5
0
たこ焼きLab
SecCap-KO
Malware erasURe Inc.
TDU-INE&IST
人海戦術 PREMIUM
F.SE
UN頼み
セキュリティ饅歌
TDU-ISL
人海戦術WHITE
悪戯武羅（あそんぶら）
m1z0r3
*ninja_sec
TapiTapi
人海戦術PLATINUM
1-1 1-2 1-3 1-4 1-5 1-6 2-1 2-2 2-3 3-1

課題2の難易度はどうでしたか？



課題2の分量はどうでしたか？



課題2として取り上げてほしい内容はありますか？

マルウェアの種類に関する意見

- Linuxのマルウェア
- Windows以外を対象とする検体の解析
- Cutwail
- ランサムウェア

問題に関する意見

- 難読化されたコードの解析
- 動的解析環境検知手法に関する内容
- 動的解析では得られない情報を取得するもの

現状でカバーできている意見

- これまで同様の実際のマルウェア
- 最新のマルウェア
- 復習したときに基本教材として使えるようなもの

課題2に対する意見、コメント

問題に関して

- マルウェアを特定する問題について、それまでの問題がどのくらい伏線になっていたのかが気になった
- 着目すべきアドレスが問題文で指定されていが、検体全体から重要な処理を探すかという設問も解いてみたい
- 分量が多かった

良かった点

- 静的解析お役立ち情報を伝えるのは良かった
- 初学者には難易度が高かったが、勉強になり楽しく取り組めた
- 課題がx64アーキテクチャであったため新鮮だった
- 各問題へのブックマークが便利だったので、今後もあると嬉しい

その他

- 昨年「今年から問題などを公開する方向」とのことで待っていたのだが、公開されていないのが残念だった

問題作成者振り返り

- 問題に関して
  - 簡単な問題として選択問題を用意したが、選択問題は部分点をあげられないのが悩ましい
    - 全体としてバランス良く分布していたので今回は良し？
  - デコンパイラ使用禁止の注意書きを入れるべきだった
  - 内容についての意見は次回の参考にします
  - 公開については、できてなくてごめんなさい
    - 来年こそは！
- その他
  - 実務解析者がチームにいる場合の考慮が必要
    - 教育という観点では実務者がチームにいて相談できるのはよい
    - 実務者は専門外の問題を解くことをルール化したい
      - その場合問題レベルも再考が必要
  - 公開されてないやんけ、とかアドバイスとか、何でも聞いてくださーい