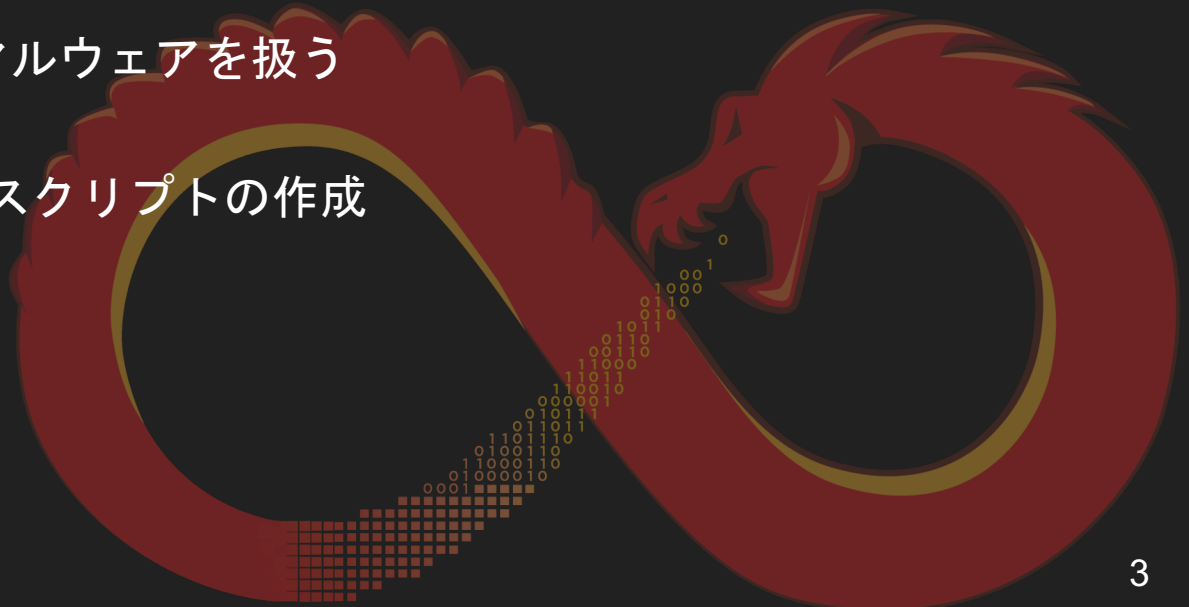
MWS Cup 2021
ポストミーティング 課題2

2021の問題担当

- 課題2主担当
  - 株式会社サイバーディフェンス研究所 中島 将太
- 問題作成委員
  - 株式会社日立製作所 石淵 一三
  - 株式会社 エヌ・エフ・ラボラトリーズ 皆川 諒
  - 株式会社 エヌ・エフ・ラボラトリーズ 齋藤 慶太
  - 学生、若手募集中！

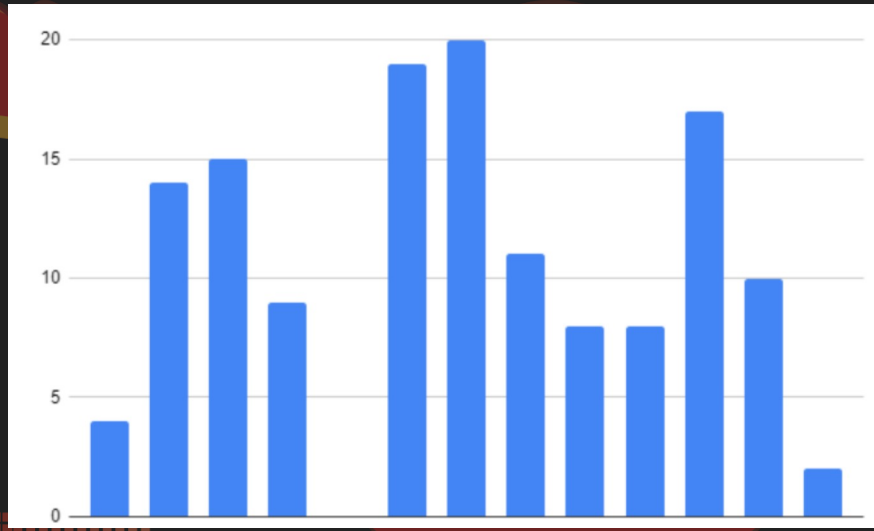
課題2のテーマ

- マルウェアを正しく理解する
  - 課題を通して解析のポイントを学習する
- 最新情報を得る
  - 最近のin-the-wildなマルウェアを扱う
- 実務に近い作業
  - 静的解析による復号スクリプトの作成



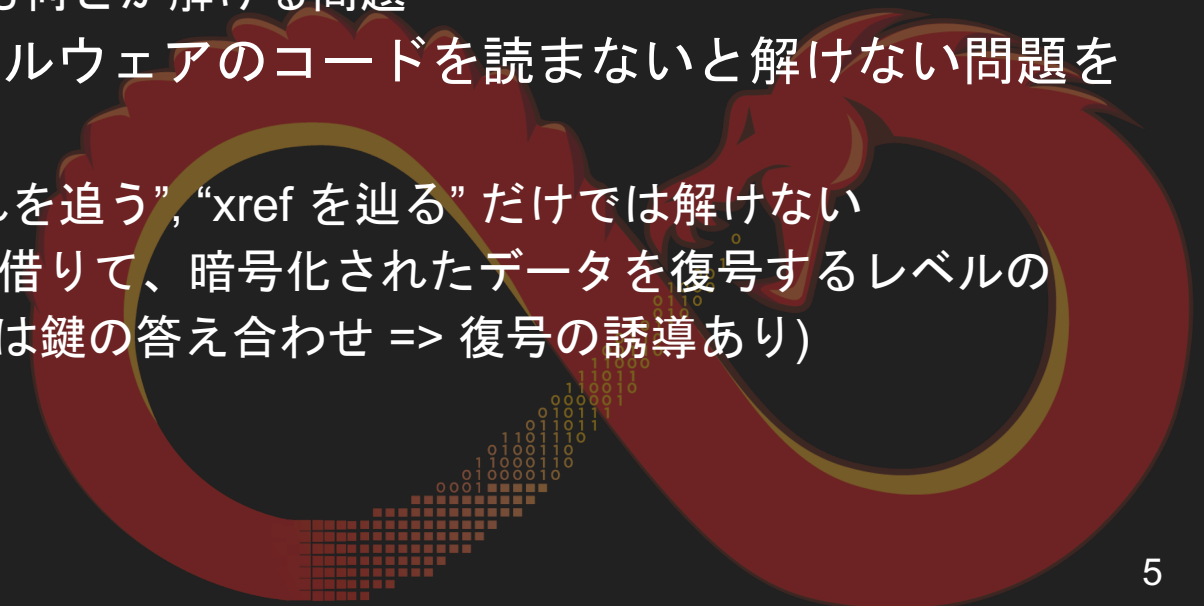
去年の反省

- Ghidra によるデコンパイラパワーの影響か問題が易化
  - 3チームが8割以上得点!
  - 1チームは満点獲得!!
- 一方で苦手なチームは非常に苦戦
  - 捨て問にされるのは避けたい
  - ボーナス問題も必要か?



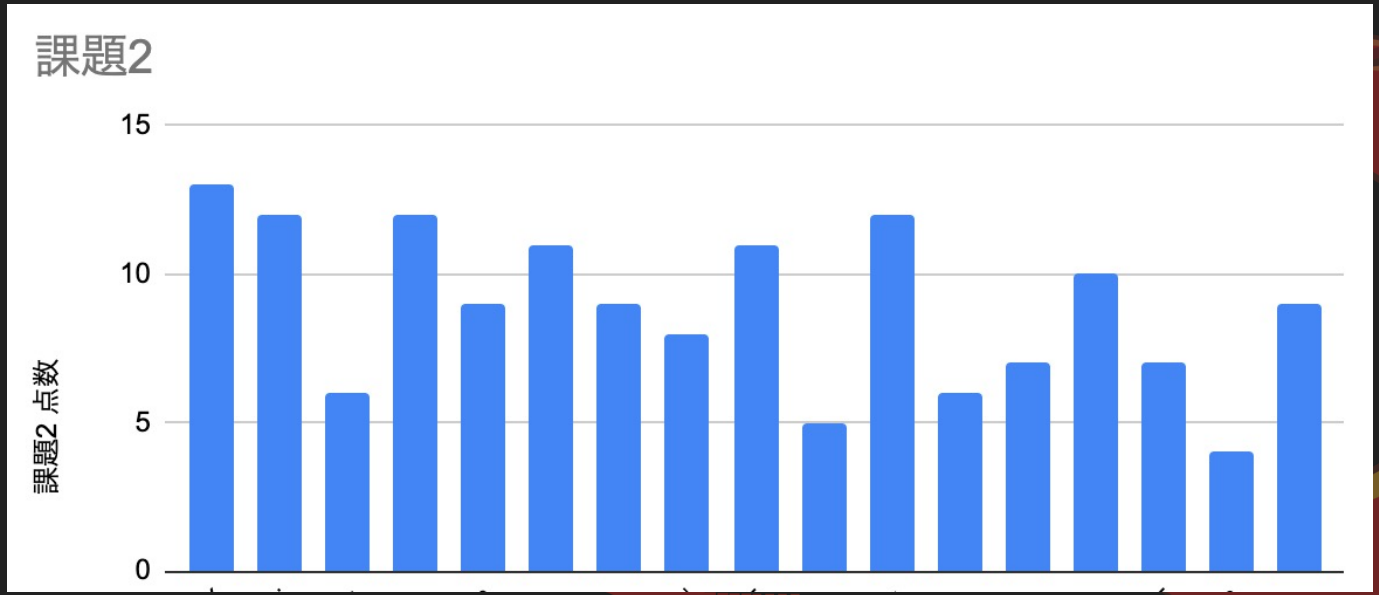
今年の方針

- ググれば解ける, 該当の関数をザッと見ただけで解けるような
  サービス問題を導入
  - 当日 Ghidra 初見でも何とか解ける問題
- 一方で、しっかりとマルウェアのコードを読まないと解けない問題を
  昨年よりも増加
  - “Windows APIの流れを追う”, “xref を辿る” だけでは解けない
  - デコンパイラの力を借りて、暗号化されたデータを復号するレベルの
    問題を数問用意(1つは鍵の答え合わせ => 復号の誘導あり)



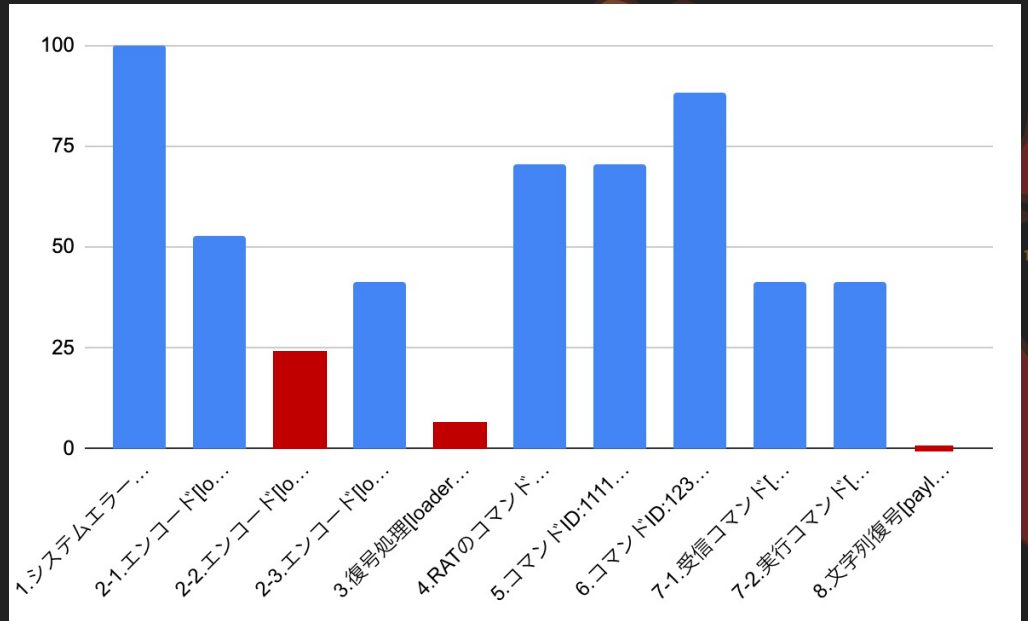
課題2 競技結果

- 結果として横這い化
- あまり差が出ない課題になったかも?

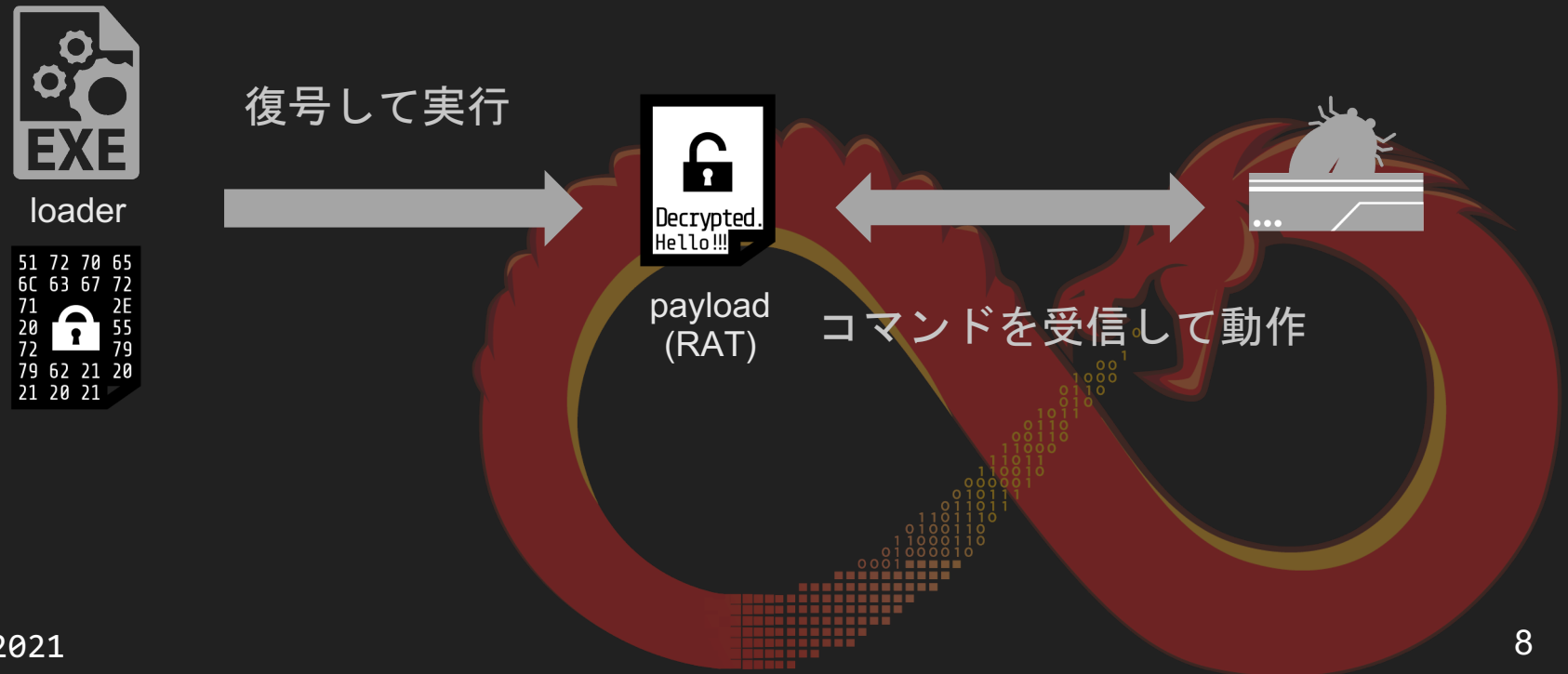


課題2 競技結果 (各課題別)

- サービス問題は意図通り多くのチームが正解
- しっかり**マルウェアの動作を読む必要のある問題**は正答率が低め

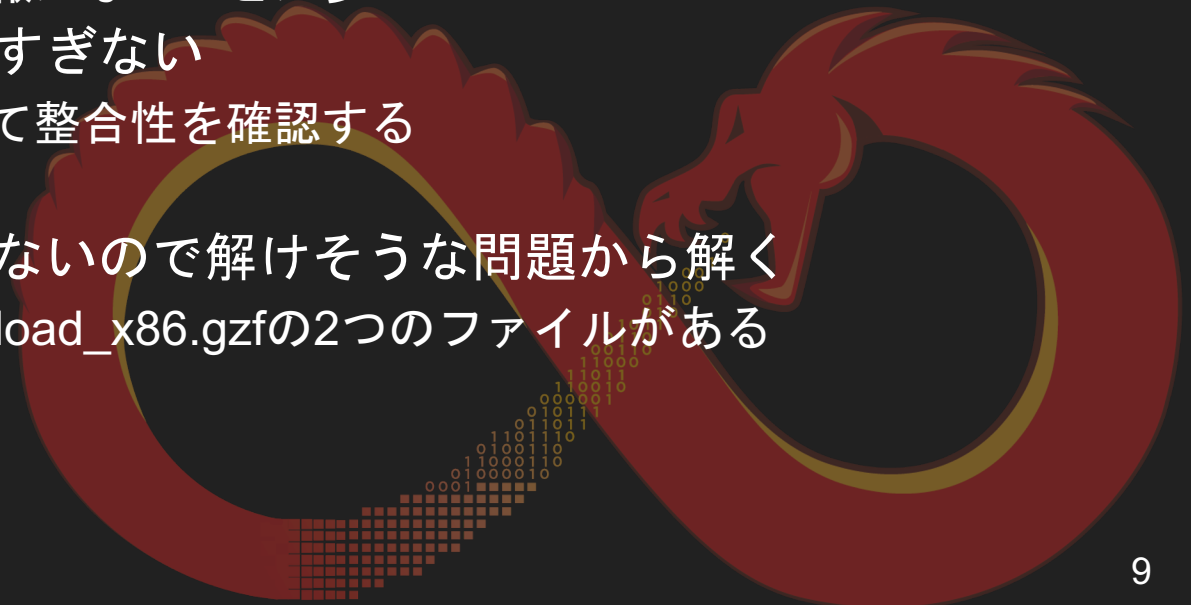


マルウェアの動作概要


EXE
loader
51 72 70 65
6C 63 67 72
71 2E
20 55
72 79
79 62 21 20
21 20 21
復号して実行
Decrypted.
Hello!!!
payload
(RAT)
コマンドを受信して動作

ポイント

- 積極的に変数名や型を変更する
  - 名前を付けて読みやすくしていく
  - デフォルトでは型情報がないことが多い
- デコンパイラを信用しすぎない
  - アセンブリを確認して整合性を確認する
  - 手動で修正する
- 順番に回答する必要はないので解けそうな問題から解く
  - loader_x86.gzfとpayload_x86.gzfの2つのファイルがある



1.システムエラーコード[loader_x86]

1.~3.は loder_x86.gzf に関する問題です。loder_x86.gzfをGhidraで解析して回答してください。

アドレス00401af3のcmp命令で比較している0xb7に対応するシステムエラーコード名(https://docs.microsoft.com/en-us/windows/win32/debug/system-error-codes--0-499-) を答えよ

※投稿回数に制限があるので注意してください。

Flag

Submit

1.システムエラーコード[loader_x86] Answer

- CreateMutex呼び出し時のエラーをGetLastErrorで取得
  - システムエラーコードが0xb7のとき1を返す
- コンテキストメニューのSet Equate(もしくはEキー)で対応する定数名に変更可能
  - ERROR_ALREADY_EXISTS
  - すでに存在するので作成できない

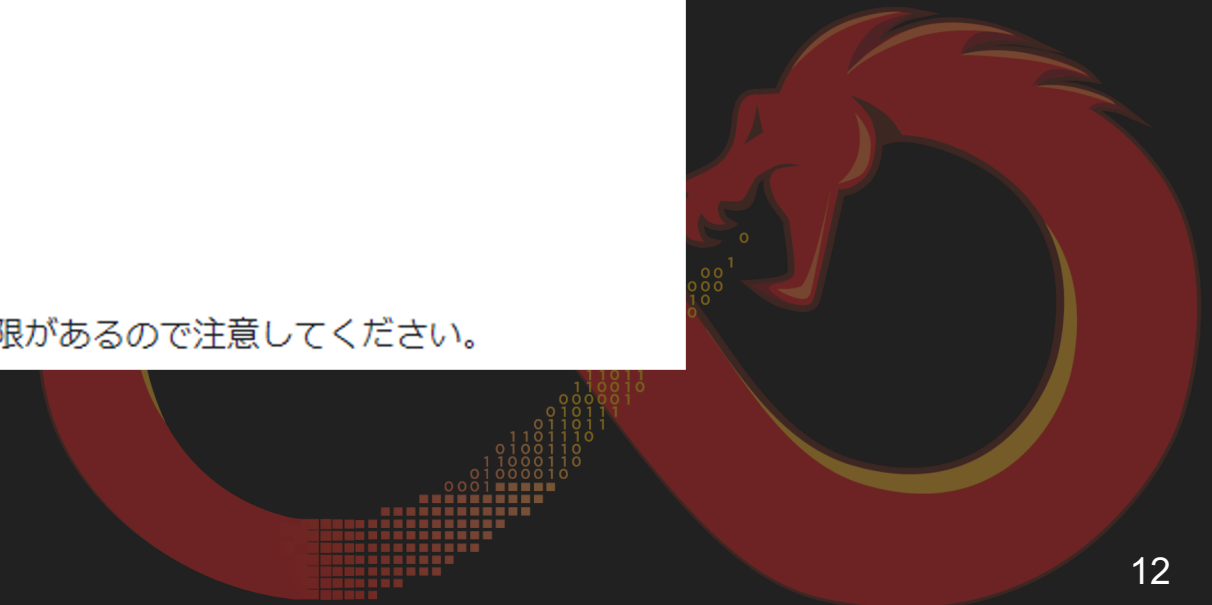
```
17  _DAT_0041c398 = 0x4a53;
18  hMutex = CreateMutexA((LPSECURITY_ATTRIBUTES)0x0,1,&DAT_0041c390);
19  uVar1 = GetLastError();
20  if (uVar1 == ERROR_ALREADY_EXISTS) {
21    return 1;
22  }
23  if (hMutex != (HANDLE)0x0) {
24    uVar1 = ReleaseMutex(hMutex);
25  }
26  return uVar1 & 0xffffff00;
```

2-1.エンコード[loader_x86]

FUN_00401810では文字列がエンコードされている。そのエンコード方式を以下から選択せよ

- ○RC4
- ○XOR
- ○AES
- ○XXTEA

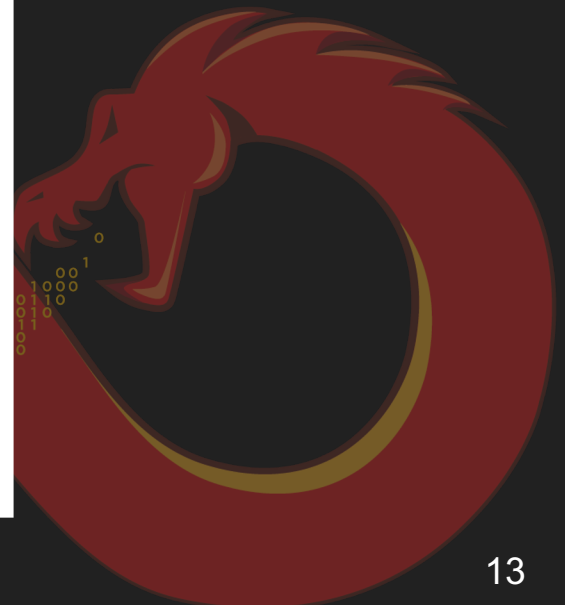
※投稿回数に制限があるので注意してください。



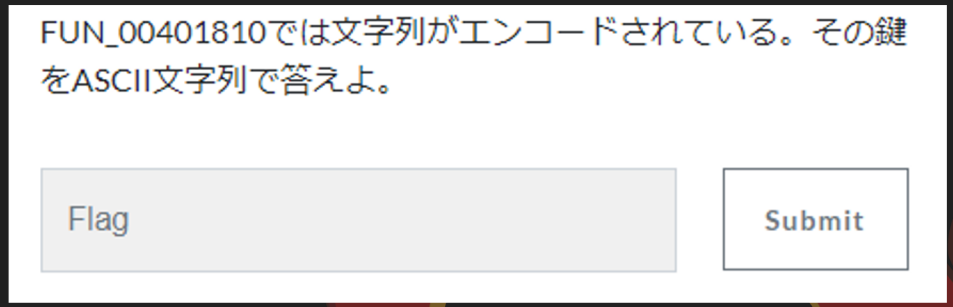
2-1.エンコード[loader_x86] Answer

- XORでデコードするループ

```
35  i = 0;
36  j = 0;
37  do {
38    k = key + i;
39    i = i + 1;
40    (&dec_str)[j] = *k ^ enc_str[j];
41    if (i == 0xf) {
42      _DAT_0041c210 = iVar2 + iVar1;
43      i = 0;
44    }
45    j = j + 1;
46  } while (j < 0xb);
47  pHVar3 = GetModuleHandleA(&dec_str);
```



2-2.エンコード[loader_x86]



2-2.エンコード[loader_x86] Answer

- Keyに利用されるデータの代入を辿る
- 16バイトとデータをASCIIに変換
  - 鍵は15バイトでローテート(実装ミス?)
  - k@W#C$DERb*F^3A

```
00401834 66 0f 6f ...   MOVDQA    XMM0,xmmword ptr [DAT_00416bd0]          = 6Bh    k
0040183c 68 d0 07 ...   PUSH      0x7d0
00401841 6a 00          PUSH      0x0
00401843 68 90 c3 ...   PUSH      dec_str                                  = ??
00401848 c7 05 44 ...   MOV       dword ptr [DAT_0041a944],0x3a            = ??
00401852 c7 45 ec ...   MOV       dword ptr [EBP + enc_str[0]],0x463e2218
00401859 c7 45 f0 ...   MOV       dword ptr [EBP + enc_str[4]],0x6b284827
00401860 c7 45 f4 ...   MOV       dword ptr [EBP + enc_str[8]],0x460e36
00401867 c7 05 74 ...   MOV       dword ptr [DAT_0041c174],0x42            = ??
00401871 f3 0f 7f ...   MOVDQU    xmmword ptr [EBP + key[0]],XMM0
00401876 c6 45 e4 00    MOV       byte ptr [EBP + local_20],0x0
0040187a e8 91 34 ...   CALL      FUN_00404d10                             uint * FUN_0
```

```
s_k@W#C$DERb*F^3A)_00416bd0                    XREF[3]:   FUN_00401810:00401834(R),
                                                          FUN_00401810:004018c1(R),
                                                          FUN_00401810:0040194f(R)
    00416bd0 6b 40 57 ...   char[16]   "k@W#C$DERb*F^3A)"
      00416bd0 [0]            'k', '@', 'W', '#'
      00416bd4 [4]            'C', '$', 'D', 'E'
      00416bd8 [8]            'R', 'b', '*', 'F'
      00416bdc [12]           '^', '3', 'A', ')'
```

2-3.エンコード[loader_x86]

FUN_00401810の処理を以下から選択せよ。

○ファイルの収集

○コンフィグの復号

○WinAPIのロード

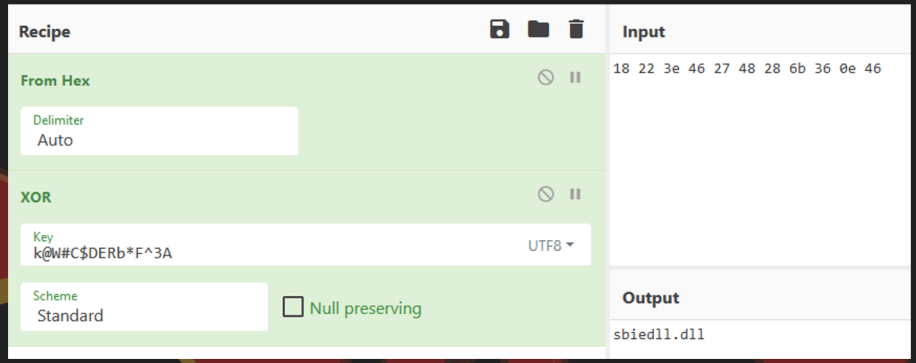
○アンチ解析機能

※投稿回数に制限があるので注意してください。

2-3.エンコード[loader_x86] Answer

- デコードした文字列

DLL名	サンドボックス
sbiedll.dll	Sandboxie
api_log.dll	CWSandbox
dir_watch.dll	CWSandbox



- サンドボックス固有のDLLを確認
  - アンチ解析機能

```
96  pHVar5 = GetModuleHandleA(&dec_str);
97  if (((pHVar3 == (HMODULE)0x0) && (pHVar4 == (HMODULE)0x0)) && (pHVar5 == (HMODULE)0x0)) {
98    return 1;
99  }
```

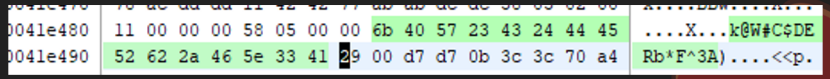
3.復号処理[loader_x86]

FUN_00401ca0とFUN_00402710は暗号化されたペイロードを読み込んで復号する処理である。 復号処理を解析して、次の文字列を復号せよ。

> KC85RDFFMDA+A14vMV0yNAQyQCxAIRoCA1MqMVIISg==

3.復号処理[loader_x86] Answer 1/4

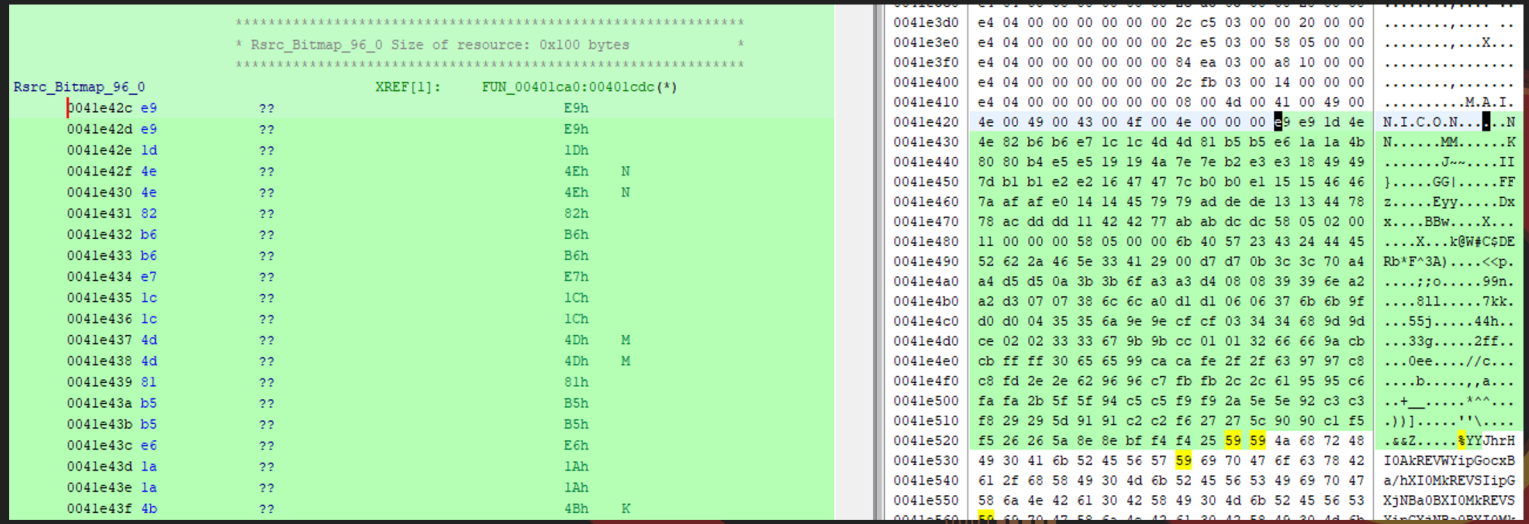
- リソース領域のデータをロード
  - オフセット0x5c-0x6bのデータをコピー
- 鍵は2.のXORと同じ




```
30  pHVar6 = FindResourceA((HMODULE)0x0,(LPCSTR)0x96,(LPCSTR)0x2);
31  if (pHVar6 == (HRSRC)0x0) {
32    return (LPVOID)0x0;
33  }
34  SizeofResource((HMODULE)0x0,pHVar6);
35  pvVar7 = LoadResource((HMODULE)0x0,pHVar6);
36  resource = LockResource(pvVar7);
37  _DAT_0041c204 = DAT_0041c084 ^ DAT_0041c2c0;
38  hWnd = GetModuleHandleA((LPCSTR)0x0);
39  hdc = GetDC((HWND)hWnd);
40  GetBkColor(hdc);
41  _DAT_0041b1d0 = DAT_0041b030 + _DAT_0041b04c;
42  uVar3 = *(undefined4 *)((int)resource + 0x60);
43  uVar4 = *(undefined4 *)((int)resource + 100);
44  uVar5 = *(undefined4 *)((int)resource + 0x68);
45  dwSize = *(SIZE_T *)((int)resource + 0x50);
46  uVar1 = *(uint *)((int)resource + 0x54);
47  sVar2 = *(size_t *)((int)resource + 0x58);
48  param_2->field_0x0 = *(undefined4 *)((int)resource + 0x5c);
49  param_2->field_0x4 = uVar3;
50  param_2->field_0x8 = uVar4;
51  param_2->field_0xc = uVar5;
```

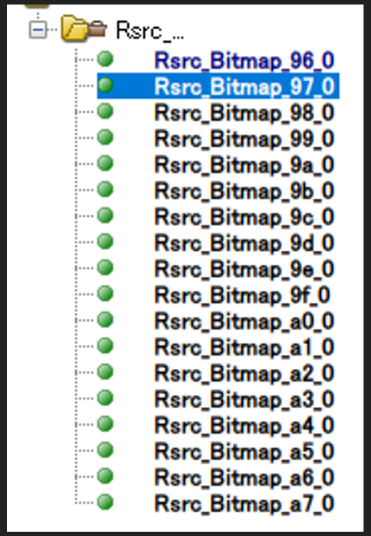
3.復号処理[loader_x86] Answer 2/4

- リソース領域のデータを確認
  - Rsrc_Bitmap_96_0



3.復号処理[loader_x86] Answer 3/4

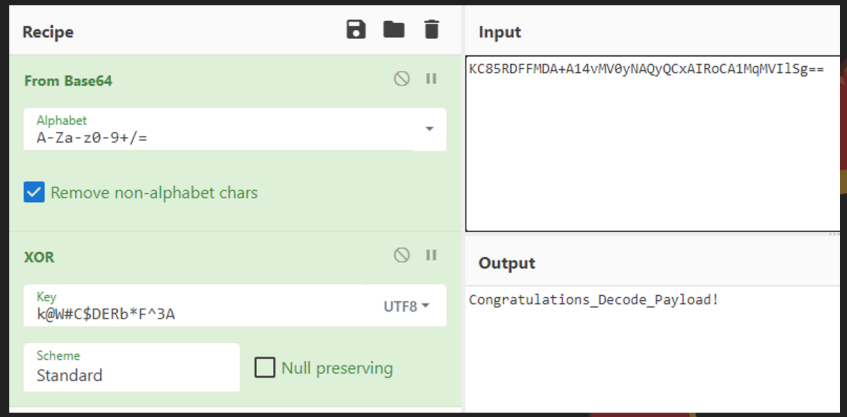
- 復号データもリソース領域からロード
  - 0x96のコンフィグにリソース数を保持




```
56  if (uVar1 != 0) {
57    do {
58      lpName = lpName + 1;
59      pHVar6 = FindResourceA((HMODULE)0x0,lpName,(LPCSTR)0x2);
60      if (pHVar6 == (HRSRC)0x0) {
61        VirtualFree(pvVar8,dwSize,0x8000);
62        _DAT_0041bff8 = 0x3f;
63        _DAT_0041c174 = 0x42;
64        FUN_00404d10((uint *)&dec_str,0,2000);
65        _dec_str = 0x61706d49;
66        _DAT_0041c394 = 0x7463;
67        CreateFontA(0x30,0,0,0,0,0,1,0,1,8,0,5,2,&dec_str);
68        return (LPVOID)0x0;
69      }
70      _DAT_0041b268 = 0x34;
71      SizeofResource((HMODULE)0x0,pHVar6);
72      pvVar7 = LoadResource((HMODULE)0x0,pHVar6);
73      _Src = LockResource(pvVar7);
74      _Size = 0x2000;
75      if (uVar1 - uVar9 == 1) {
76        _Size = sVar2;
77      }
78      FID_conflict:_memcpy(local_34,_Src,_Size);
79      _DAT_0041c374 = DAT_0041c058 ^ DAT_0041c0ac;
80      FileTimeToSystemTime(&local_20,(LPSYSTEMTIME)&local_18);
81      uVar9 = uVar9 + 1;
82      _DAT_0041b318 = DAT_0041b200 ^ _DAT_0041b23c;
83      local_34 = (void *)((int)local_34 + 0x2000);
84    } while (uVar9 < uVar1);
85  }
```

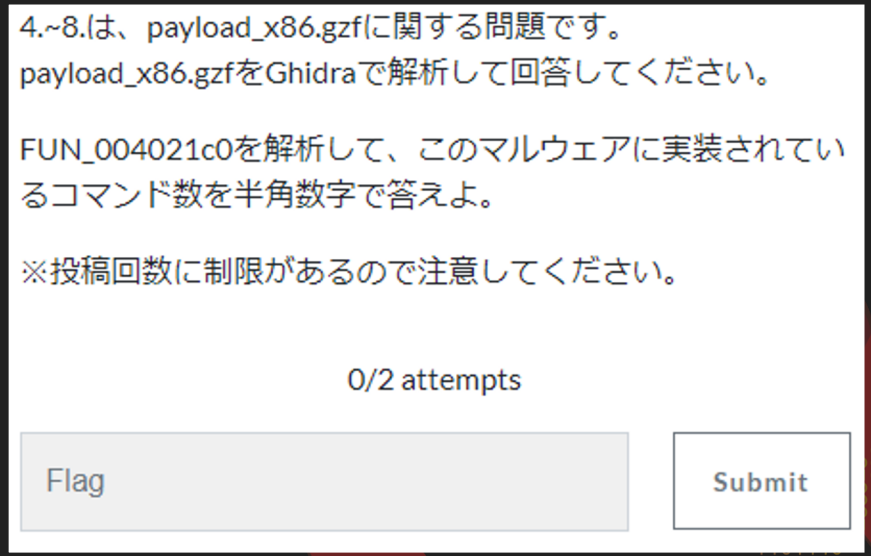
3.復号処理[loader_x86] Answer 4/4

- Base64とXORでデコード
  - 鍵は15バイトでローテートする




```
17  _Src = b64_decode(param_1,param_3,(int *)&size);
18  if (size != 0) {
19    _DAT_0041bcb8 = DAT_0041bc48 - _DAT_0041bee8;
20    _DAT_0041bfb0 = DAT_0041c014 & _DAT_0041clec;
21    KillTimer((HWND)0x0,1);
22    FID_conflict:_memcpy(param_2,_Src,size);
23    FID_conflict:_free(_Src);
24    i = 0;
25    offset = 0;
26    if (0 < (int)size) {
27      do {
28        xor_key = key + i;
29        i = i + 1;
30        param_2[offset] = param_2[offset] ^ *xor_key;
31        if (i == 0xf) {
32          i = 0;
33        }
34        offset = offset + 1;
35      } while (offset < (int)size);
36    }
37    return size;
```

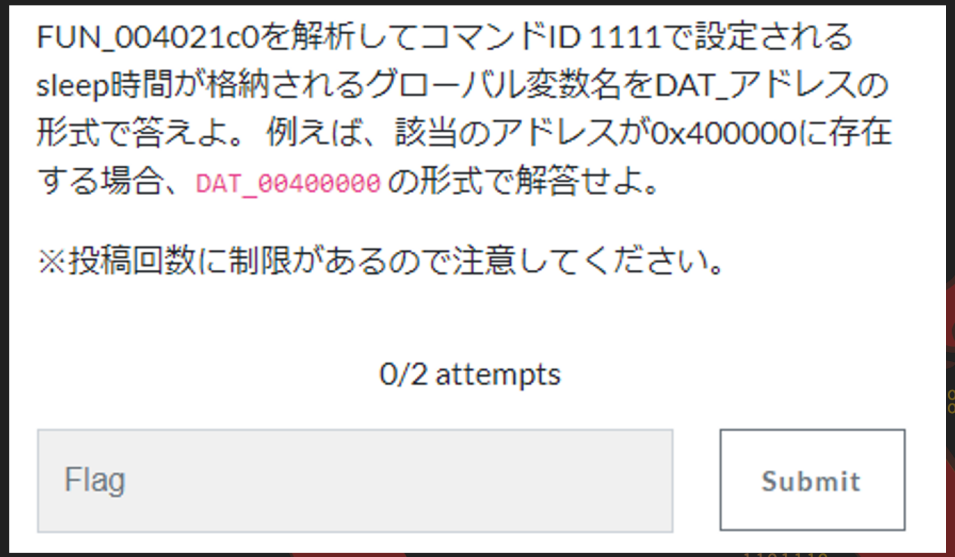
4.RATのコマンド数[payload_x86]



4.RATのコマンド数[payload_x86]

- 実装されているコマンドは8つ
  - 8877
  - 3333
  - 1111
  - 1234
  - 4444
  - 8888
  - 9876
  - 9999

5.コマンドID:1111[payload_x86]



5.コマンドID:1111[payload_x86]

- コマンド1111内のグローバル変数の参照を追う

```
101         if (cmdid == 1111) {
102           if ((int)pcVar11 < 10) {
103             local_10 = 0;
104             local_c = 0;
105             FID_conflict:_memcpy(&local_10,_Memory + 1,(size_t)(pcVar11 + -4));
106             cmdid = FUN_004036fb((byte *)&local_10);
107             if (cmdid < 20002) {
108               local_55c = 0;
109               DAT_004191d8 = cmdid;
```

```
DAT_004191d8                                    XREF[2]:   FUN_00401450:004014c2(R),
                                                           FUN_004021c0:00402347(W)
       004191d8 05 00 00 00     undefined4  00000005h
```

```
40                 }
41                 Sleep(DAT_004191d8 * 1000);
42               } while (uVar4 != 0);
43               WSACleanup();
```

6.コマンドID:1234[payload_x86]

FUN_004021c0を解析し、コマンド 1234 の機能を答えよ

- ○ファイルを削除するバッチファイルを実行する
- ○C2 サーバから受信したデータをPowerShell経由で実行する
- ○C2 サーバから受信したデータをファイルとして保存して実行する
- ○C2 サーバから受信したデータを別スレッドで実行する

※投稿回数に制限があるので注意してください。

6.コマンドID:1234[payload_x86]

- C2 サーバから受信したデータを別スレッドで実行する

```
122         if ((cmdid == 1234) &&
123            (lpStartAddress = (uint *)_malloc((size_t)pcVar11), lpStartAddress != (uint *)0x0))
124           FUN_00409f90(lpStartAddress,0,(uint)pcVar11);
125           FID_conflict:_memcpy(lpStartAddress,_Memory + 1,(size_t)(pcVar11 + -4));
126           local_564[1] = (uint *)0x0;
127           pvVar7 = CreateThread((LPSECURITY_ATTRIBUTES)0x0,0,
128                                 (LPTHREAD_START_ROUTINE)lpStartAddress,(LPVOID)0x0,0,
129                                 (LPDWORD)(local_564 + 1));
130           if (pvVar7 == (HANDLE)0x0) goto LAB_00402737;
131           WaitForSingleObject(pvVar7,5000);
```

7-1.受信コマンド[payload_x86]

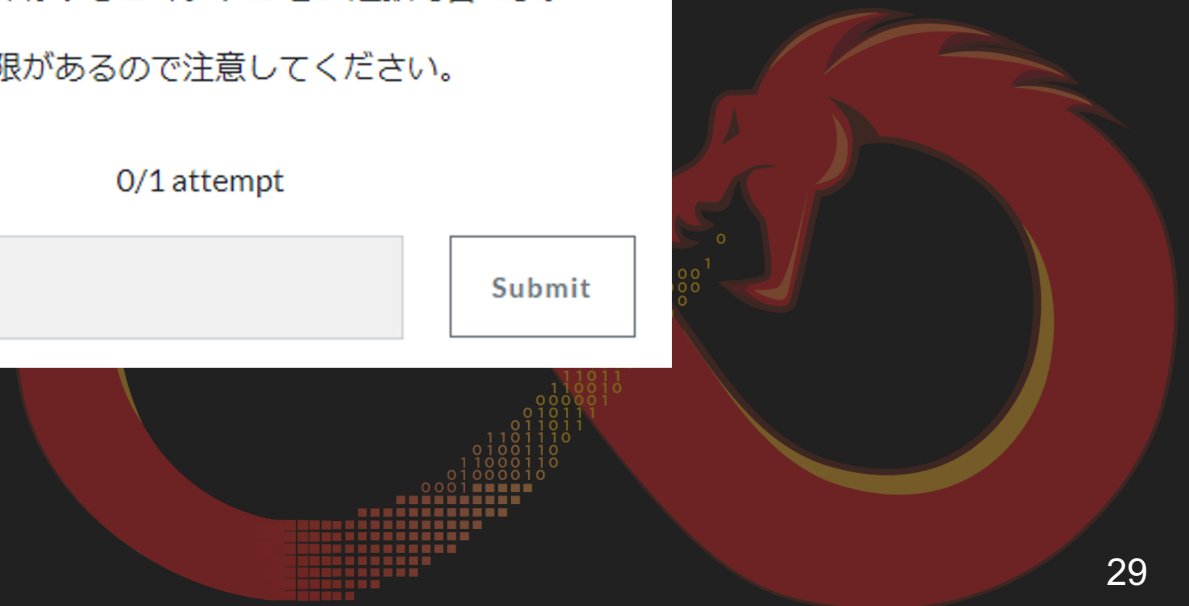
FUN_004021c0を解析してC2サーバから受信したデータをファイルとして保存するコマンドIDを10進数で答えよ。

※投稿回数に制限があるので注意してください。

0/1 attempt

Flag

Submit

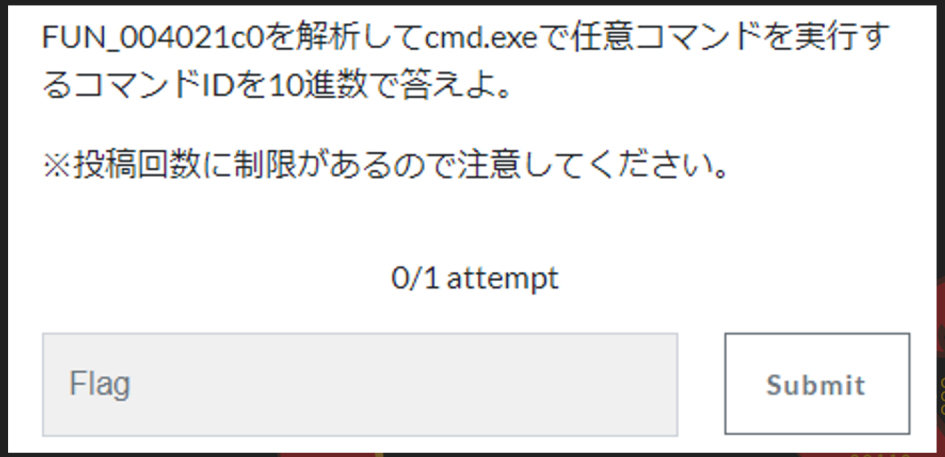


7-1.受信コマンド[payload_x86]

- 8877
  - ファイルをオープン
  - 受信データをファイルに保存

```
64        pFVar9 = _fopen(&local_15c,"wb");
65        if (pFVar9 != (FILE *)0x0) {
66          puVar10 = _Memory;
67          do {
68            cVar2 = *(char *)puVar10;
69            puVar10 = (undefined4 *)((int)puVar10 + 1);
70          } while (cVar2 != '\0');
71          local_564[1] = (uint *)((int)_Memory + 1);
72          puVar13 = _Memory;
73          do {
74            cVar2 = *(char *)puVar13;
75            puVar13 = (undefined4 *)((int)puVar13 + 1);
76          } while (cVar2 != '\0');
77          FUN_00403ad5((ushort *)((int)_Memory + (int)((int)puVar13 + (1 - (int)local_564[1]))),1,
78                       (uint)(pcVar11 + (-1 - ((int)puVar10 - (int)((int)_Memory + 1)))),pFVar9);
79          _fclose(pFVar9);
```

7-2.実行コマンド[payload_x86]

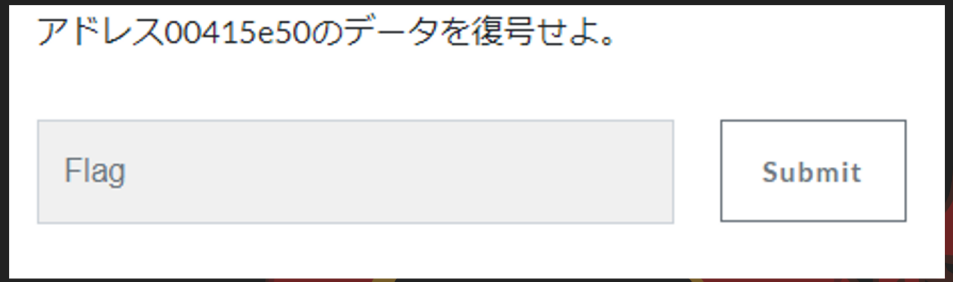


7-2.実行コマンド[payload_x86]

- 9999
  - パイプを使ってcmd.exeで任意のコマンド実行

```
40  FUN_0040320f((char *)local_418,(byte *)"%s\\cmd.exe /c %s");
41  puVar3 = (uint *)_malloc(0x400);
42  if (puVar3 == (uint *)0x0) {
43    FUN_00403200(local_14 ^ (uint)&stack0xfffffa58,extraout_DL,in_stack_fffffa58);
44    return;
45  }
46  FUN_00409f90(puVar3,0,0x400);
47  local_584 = 1;
48  local_58c = 0xc;
49  CreatePipe(&local_598,&local_590,(LPSECURITY_ATTRIBUTES)&local_58c,1000);
50  FUN_00409f90(&local_568.cb,0,0x44);
51  local_568.hStdOutput = local_590;
52  local_568.hStdError = local_590;
53  local_568.wShowWindow = 0;
54  local_580 = (_PROCESS_INFORMATION)ZEXT816(0);
55  local_568.cb = 0x44;
56  local_568.dwFlags = 0x101;
57  BVar4 = CreateProcessA((LPCSTR)0x0,(LPSTR)local_418,(LPSECURITY_ATTRIBUTES)0x0,
58                         (LPSECURITY_ATTRIBUTES)0x0,1,0,(LPVOID)0x0,(LPCSTR)0x0,
59                         (LPSTARTUPINFOA)&local_568,(LPPROCESS_INFORMATION)&local_580);
60  if (BVar4 != 0) {
61    CloseHandle(local_590);
62    Sleep(500);
63    local_594 = (uint *)_malloc(0x100000);
64    if (local_594 != (uint *)0x0) {
65      FUN_00409f90(local_594,0,0x100000);
66      iVar6 = 0;
67      iVar5 = ReadFile(local_598,puVar3,1000,&local_59c,(LPOVERLAPPED)0x0);
```

8.文字列復号[payload_x86]



8.文字列復号[payload_x86] 1/3

- データの参照を追う

```
s_bYR+jw2oi3a79/wVS2DG6ts12vPN+FIT_00415e50      XREF[1]:     FUN_004012a0:00401354(*)
        00415e50 62 59 52 ...     ds          "bYR+jw2oi3a79/wVS2DG6ts12vPN+FITVgY5uO3Vk/LE38C2sR...
```

- 2つの関数で復号
  - Base64
  - カスタムXOR

```
33      _Memory = (char *)FUN_004030d0((byte *)
34                                     "bYR+jw2oi3a79/wVS2DG6ts12vPN+FITVgY5uO3Vk/LE38C2sRbqzxkJKWT2",
35                                     0x3c,(int *)&stack0xfffffe60);
36      FUN_00402ec0((int)_Memory,(byte *)_Memory,iVar3);
```

8.文字列復号[payload_x86] 2/3

- カスタムXORの実装

```
void __fastcall FUN_00402ec0(int param_1,byte *param_2,int param_3)

{
  byte bVar1;
  uint uVar2;
  byte bVar3;
  uint uVar4;
  byte *pbVar5;

  bVar3 = 0x84;
  uVar4 = 0x9a9a2c2;
  uVar2 = 0x57219043;
  if (0 < param_3) {
    pbVar5 = param_2;
    do {
      bVar1 = (byte)uVar2;
      *pbVar5 = pbVar5[param_1 - (int)param_2] ^ (byte)uVar4 ^ bVar1 ^ bVar3;
      bVar3 = (bVar1 ^ bVar3) & (byte)uVar4 ^ bVar1 & bVar3;
      uVar4 = uVar4 >> 8 | ((uVar4 * 8 ^ uVar4) & 0x7f8) << 0x14;
      uVar2 = uVar2 >> 8 | (((uVar2 * 2 ^ uVar2) << 4 ^ uVar2) & 0xffffff80 ^ uVar2 << 7) << 0x11;
      param_3 = param_3 + -1;
      pbVar5 = pbVar5 + 1;
    } while (param_3 != 0);
  }
  return;
```

8.文字列復号[payload_x86] 3/3

```python
import base64

def decode(enc):
    mask1 = 0x84
    mask2 = 0x9A9A2C2
    mask3 = 0x57219043

    for i in range(len(enc)):
        enc[i] ^= (mask1 ^ mask2 ^ mask3) & 0xFF
        mask1 = (mask1 ^ mask3) & mask2 ^ mask3 & mask1 & 0xFF
        mask2 = mask2 >> 8 | (((mask2 ^ (8 * mask2)) & 0x7F8) << 0x14)
        mask3 = mask3 >> 8 | (((((mask3 * 2) ^ mask3) * 16) ^ mask3) & 0xffffff80 ^ mask3 << 7) << 0x11 & 0xFFFFFFFF
    return(enc)

decode(bytearray(base64.b64decode('bYR+jw2oi3a79/wVS2DG6ts12vPN+FITVgY5uO3Vk/LE38C2sRbqzxkJKWT2')))
```

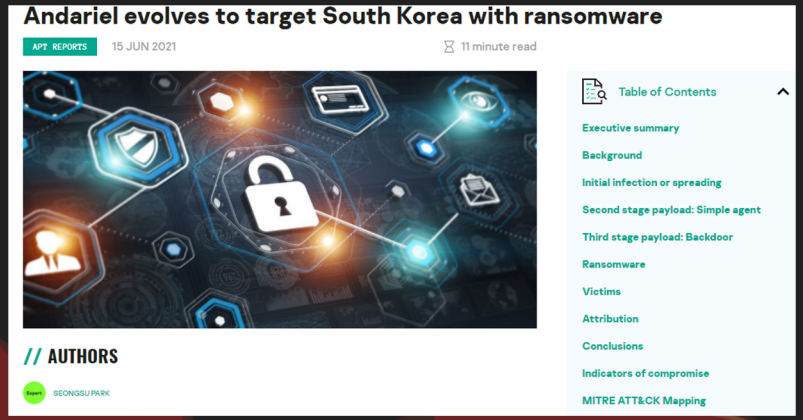
```
...
>>> decode(bytearray(base64.b64decode('bYR+jw2oi3a79/wVS2DG6ts12vPN+FITVgY5uO3Vk/LE38C2sRbqzxkJKWT2')))
bytearray(b'http://www.conkorea.com/cshop/banner/list.php')
```

解答まとめ

1. アドレス00401af3のcmp命令で比較している0xb7に対応するシステムエラーコード名を答えよ。	ERROR_ALREADY_EXISTS
2-1. FUN_00401810では文字列がエンコードされている。そのエンコード方式を以下から選択せよ。	XOR
2-2. FUN_00401810では文字列がエンコードされている。その鍵をASCII文字列で答えよ。	k@W#C$DERb*F^3A
2-3. FUN_00401810の処理を以下から選択せよ。	アンチ解析機能
3. FUN_00401ca0とFUN_00402710は暗号化されたペイロードを読み込んで復号する処理である。復号処理を解析して、次の文字列を復号せよ	Congratulations_Decode_Payload!
4. FUN_004021c0を解析して、このマルウェアに実装されているコマンド数を答えよ	8
5. FUN_004021c0を解析してコマンドID 1111で設定されるsleep時間が格納されるグローバル変数名をDAT_アドレスの形式で答えよ。	DAT_004191d8
6. FUN_004021c0を解析し、コマンド 1234 の機能を答えよ	C2 サーバから受信したデータを別スレッドで実行する
7-1. FUN_004021c0を解析してC2 サーバから受信したデータをファイルとして保存するコマンドIDを10進数で答えよ。	8877
7-2. FUN_004021c0を解析してcmd.exeで任意コマンドを実行するコマンドIDを10進数で答えよ。	9999
8. アドレス00415e50のデータを復号せよ。	http[:]//www.conkorea[.]com/cshop/banner/list[.]php

マルウェアファミリ

- 2021/04 Lazarusのサブグループ Andarielが利用するマルウェア
  - https://securelist.com/andariel-evolves-to-target-south-korea-with-ransomware/102811/
  - https://blog.malwarebytes.com/threat-intelligence/2021/04/lazarus-apt-conceals-malicious-code-within-bmp-file-to-drop-its-rat/

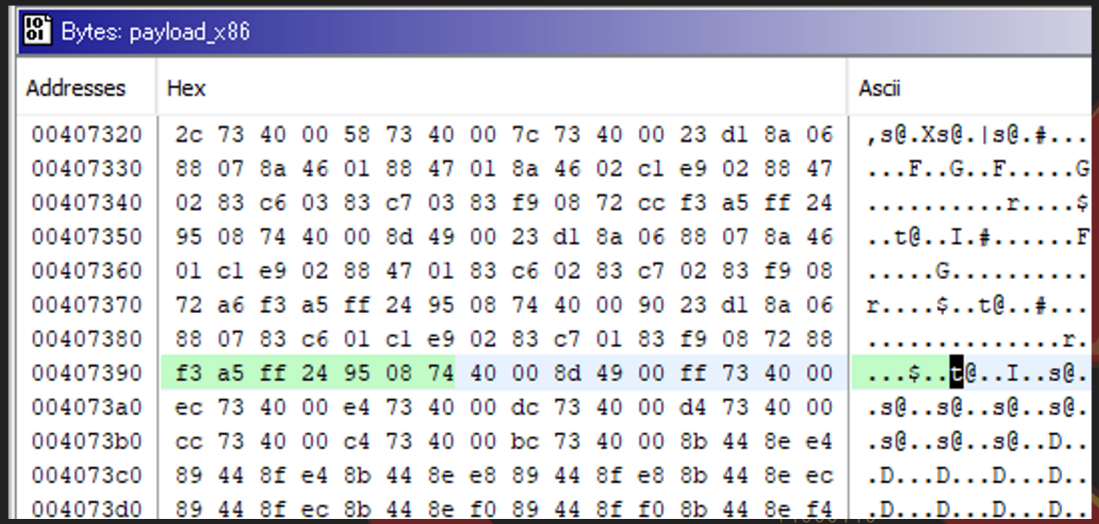


Ghidra Tips: Go To

- Gキーで開くGo Toダイアログにアドレスやlabelを入力
  - 任意の場所に簡単に移動
  - 問題ではアドレスが指定されていることが多いので必須!

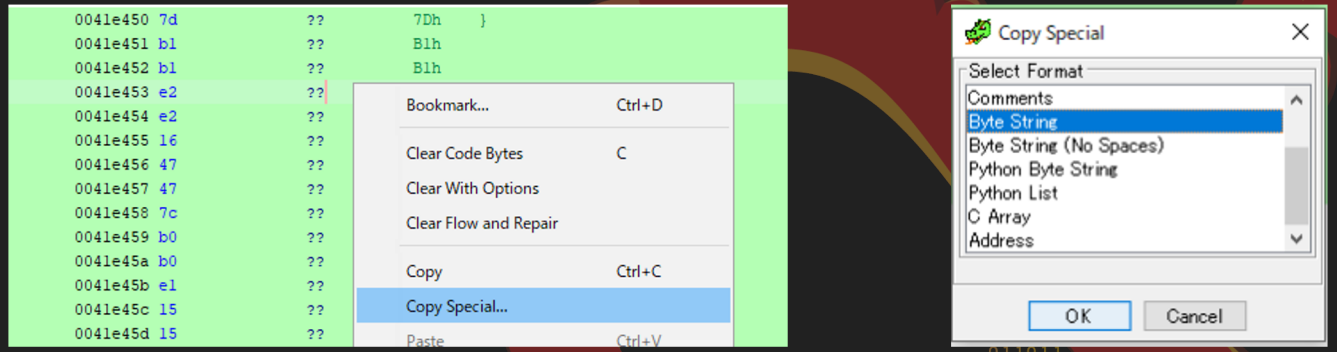
Ghidra Tips: データのコピー1

- Bytesウィンドウでコピーする



Ghidra Tips: データのコピー2

- コピーしたい範囲を選択
  - コンテキストメニューからCopy Special => Byte Stringを選択



Ghidra Help: データのコピー3

- Listingビューで見切れている場合
  - 文字列
- Defined Stringsウィンドウ
  - Copy Current Column