MWS Cup 2016
課題2: 静的解析

中津留 勇

2016/12/07

課題2 担当

中津留　勇

- SecureWorks Japan 株式会社

稲垣　俊

- セコム株式会社

課題2 の変わらぬテーマ

- 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 2016
課題2 解説

SecureWorks
A Dell Company

Tick/Daserf

- 日本国内で数年前から発生していた標的型攻撃

http://www.symantec.com/connect/blogs/tick-cyberespionage-group-zeros-japan

標的型攻撃に用いられるマルウェア、Daserf とは

Daserf はバックドア機能を有するマルウェアで、Nioupale とも呼ばれています。Daserf については、2016 年 5 月に Symantec がブログ[ii]で報告していますが、それまではセキュリティベンダによる報告はほとんどなく、このマルウェアの存在自体、広く知られているとは言えない状況でした。一方、ラックは 2013 年 1 月頃以降に対応した複数の標的型攻撃事案において Daserf を確認しており、これらの分析を続けてきました。その結果、Daserf が日本の重要インフラを標的とした攻撃者に使用され、長期間にわたって標的組織に潜伏しつつ活動している可能性が高いことが明らかになりました。

図 1 は、ラックが対応した事案において Daserf が使われた業種を分類したグラフです。グラフ外周右側の枠に含まれるのが重要インフラに属する業種[iii]で、56%と過半数を占めていることがわかります。外周左側の枠は重要インフラで利用される機器を製造する事業者で、これらを含めるとすべての事案が重要インフラに直接的、間接的に関連していることがわかります。このことから、Daserf を使う攻撃者は、少なくとも日本においては重要インフラやその関連企業をターゲットとしている可能性が高いと考えられます。

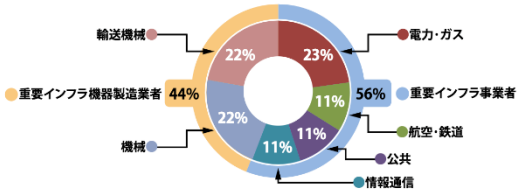


図 1 ラックが対応した標的型攻撃事案のターゲット組織

http://www.lac.co.jp/security/report/pdf/20160802_cgview_vol2_a001t.pdf

課題2

1．コード難読化

2．暗号アルゴリズム（選択式）

3．暗号化/エンコード処理

4．暗号化されたデータの復号

5．設定情報の調査

ポイント

難しく捉えない

- アセンブリ言語そのものは単純

分かりやすい情報を探す

- 文字列、定数、API

すべてのコードを読まない

- 終わりません

1. コード難読化

- 難読化が行われている箇所を探す
  - 何が「普通のコード」かを知っておく必要がある

```
004232A0                        public start
004232A0        start           proc near
004232A0 000                    push    ebp
004232A1 004                    mov     ebp, esp
004232A3 004                    add     esp, 0FFFFFFF0h
004232A6 014                    mov     eax, offset dword_423208
004232AB 014                    call    Sysinit::__linkproc__ InitExe(void *)
004232B0 014                    xor     eax, eax
004232B2 014                    push    ebp
004232B3 018                    push    offset loc_423326
004232B8 01C                    push    dword ptr fs:[eax]
004232BB 020                    mov     fs:[eax], esp
004232BE 020                    sub     esi, 7
004232C1 020                    add     eax, ecx
004232C3 020                    dec     ebx
004232C4 020                    jmp     short loc_4232E6
```

1. コード難読化

- 答えはすぐ下に

無駄なデータを挿入し、jmp 命令で飛ばしながら実行する

```
CODE:004232C3 020                       dec
CODE:004232C4 020                       jmp
CODE:004232C4       ; ---------------------
CODE:004232C6 020                       dw 0C2A1h
CODE:004232C8 020                       dd 0F303AD9Fh          9F8E6h, 1DD5589Ah, 586627
CODE:004232C8                           dd 0DF0B6BE4h, 0DF0B6E65h
CODE:004232E4 020                       db 65h, 6Eh
CODE:004232E6       ; ---------------------------------------------------------------------
CODE:004232E6
CODE:004232E6       loc_4232E6:                                     ; CODE XREF: start+2
CODE:004232E6 020                       add     ebx, 1
CODE:004232E9 020                       add     esi, 3
CODE:004232EC 020                       sub     eax, ecx
CODE:004232EE 020                       add     esi, 4
CODE:004232F1 020                       jmp     short loc_423313
CODE:004232F1       ; ---------------------------------------------------------------------
CODE:004232F3 020                       db 6Fh
CODE:004232F4 020                       dd 8263B0F7h, 0BF8C3995h, 51C1A1B9h, 0BA881E
CODE:004232F4                           dd 5D0C947Dh, 5DB96BDDh
CODE:00423310 020                       db 0DDh, 6Bh, 0B9h
CODE:00423313       ; ---------------------------------------------------------------------
CODE:00423313
CODE:00423313       loc_423313:                                     ; CODE XREF: start+5
CODE:00423313 020                       call    sub_420534
```

1. コード難読化

- API 呼び出しの難読化も

2. 暗号アルゴリズム

- AES, DES, 3DES, RC4, TEA?

```
CODE:004198B0 120                 jl      loc_419955
CODE:004198B6 120                 inc     eax
CODE:004198B7 120                 mov     [ebp+var_18], eax
CODE:004198BA 120                 mov     [ebp+var_10], 0
CODE:004198C1
CODE:004198C1     loc_4198C1:                                 ; CODE XREF: sub_41
CODE:004198C1 120                 inc     [ebp+var_1A]
CODE:004198C4 120                 xor     eax, eax
CODE:004198C6 120                 mov     al, [ebp+var_1A]
CODE:004198C9 120                 mov     al, [ebp+eax+var_11A]
CODE:004198D0 120                 mov     [ebp+var_11], al
CODE:004198D3 120                 mov     al, [ebp+var_11]
CODE:004198D6 120                 add     [ebp+var_19], al
CODE:004198D9 120                 xor     eax, eax
CODE:004198DB 120                 mov     al, [ebp+var_19]
CODE:004198DE 120                 mov     al, [ebp+eax+var_11A]
CODE:004198E5 120                 xor     edx, edx
CODE:004198E7 120                 mov     dl, [ebp+var_1A]
CODE:004198EA 120                 mov     [ebp+edx+var_11A], al
CODE:004198F1 120                 xor     eax, eax
CODE:004198F3 120                 mov     al, [ebp+var_19]
CODE:004198F6 120                 mov     dl, [ebp+var_11]
```

2. 暗号アルゴリズム

各選択肢の調査

- AES: SPN構造のブロック暗号
- DES、3DES: Feistel構造のブロック暗号
- RC4: ストリーム暗号
- TEA: Feistel構造のブロック暗号

アセンブリ言語を読み特徴を抽出

- sub_41978C を呼んでから、XOR のループ
- AES などで使用される定数を参照していない
- 1バイトずつ XOR していてストリームっぽい

2. 暗号アルゴリズム

- sub_41978C

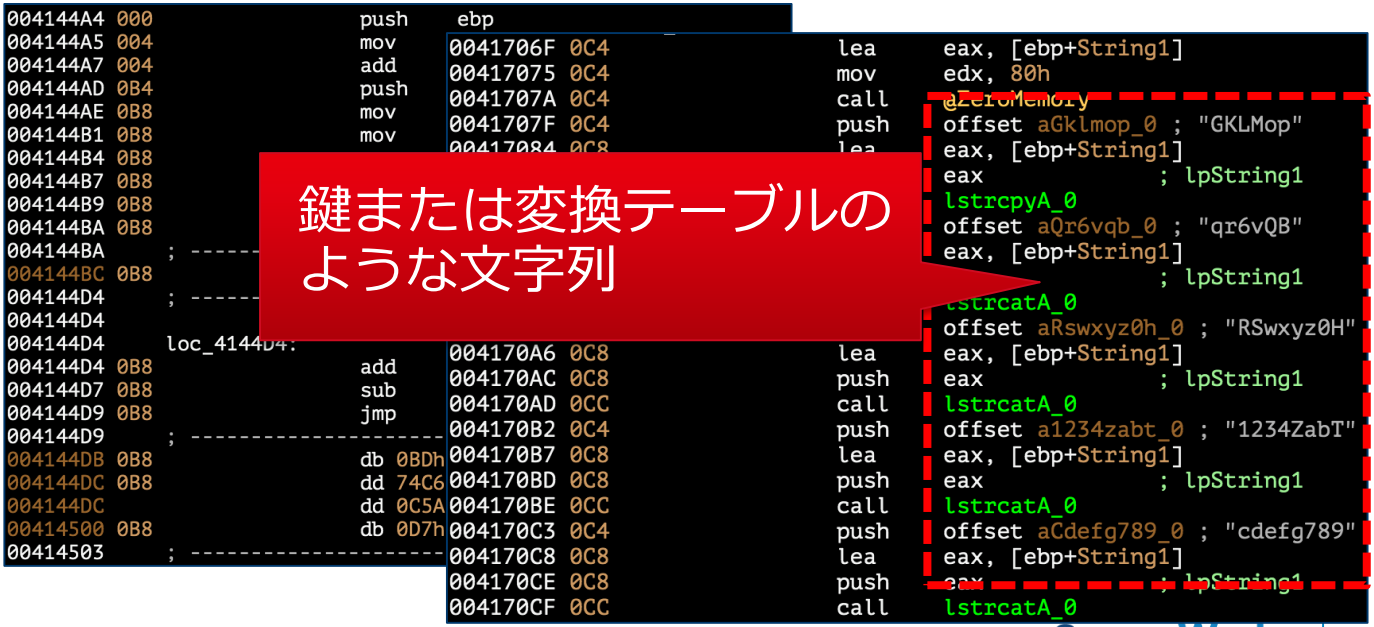
0xFF までを配置した配列を作成している

```
CODE:004197A3 018
CODE:004197A6 018                                                    ], 0
CODE:004197AD 018
CODE:004197B0 018                                                    ], 0
CODE:004197B7 018                    mov     [ebp+                   ], 0
CODE:004197BB
CODE:004197BB                loc_4197BB:                                    ; CODE
CODE:004197BB 018                    xor     eax, eax
CODE:004197BD 018                    mov     al, [ebp+var_A]
CODE:004197C0 018                    mov     edx, [ebp+var_4]
CODE:004197C3 018                    mov     cl, [ebp+var_A]
CODE:004197C6 018                    mov     [edx+eax], cl
CODE:004197C9 018                    inc     [ebp+var_A]
CODE:004197CC 018                    cmp     [ebp+var_A], 0
CODE:004197D0 018                    jnz     short loc_4197BB
```

暗号アルゴリズムは RC4

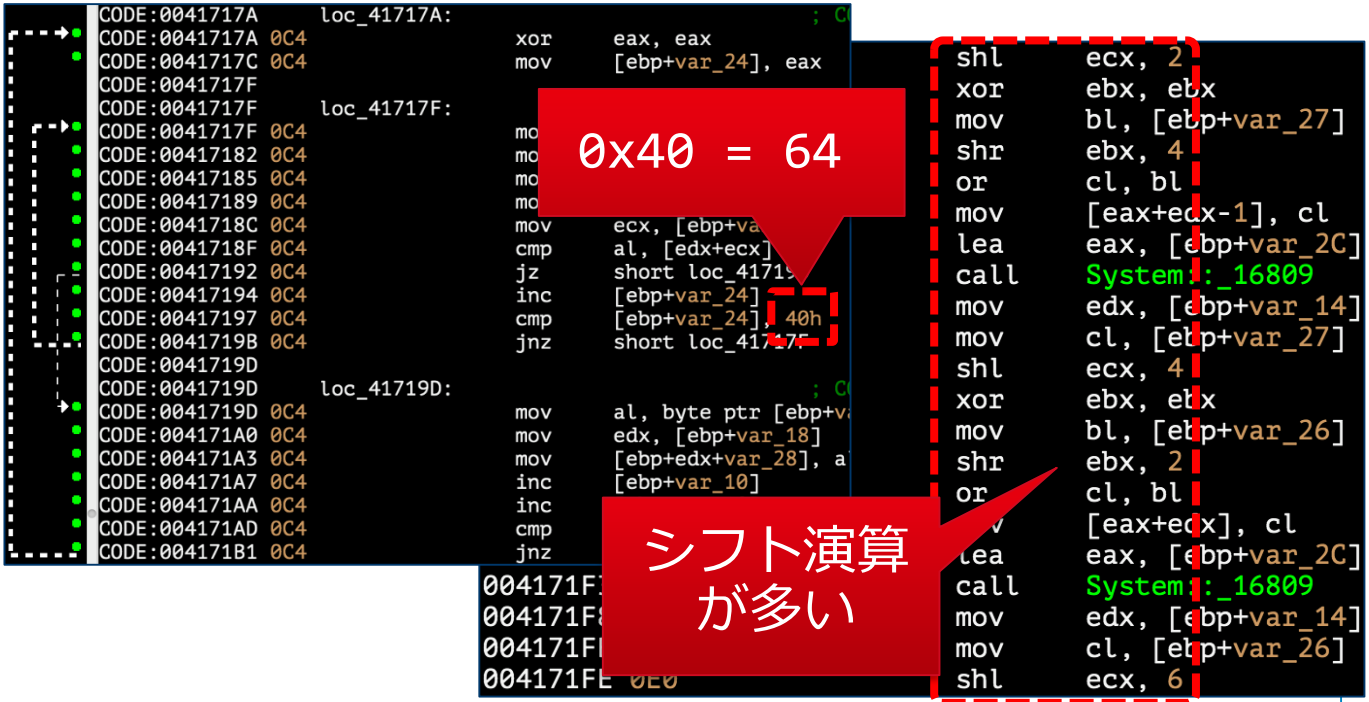
3. 暗号化/エンコード処理

- 暗号なのかエンコードなのか、そして鍵や変換テーブルは何か



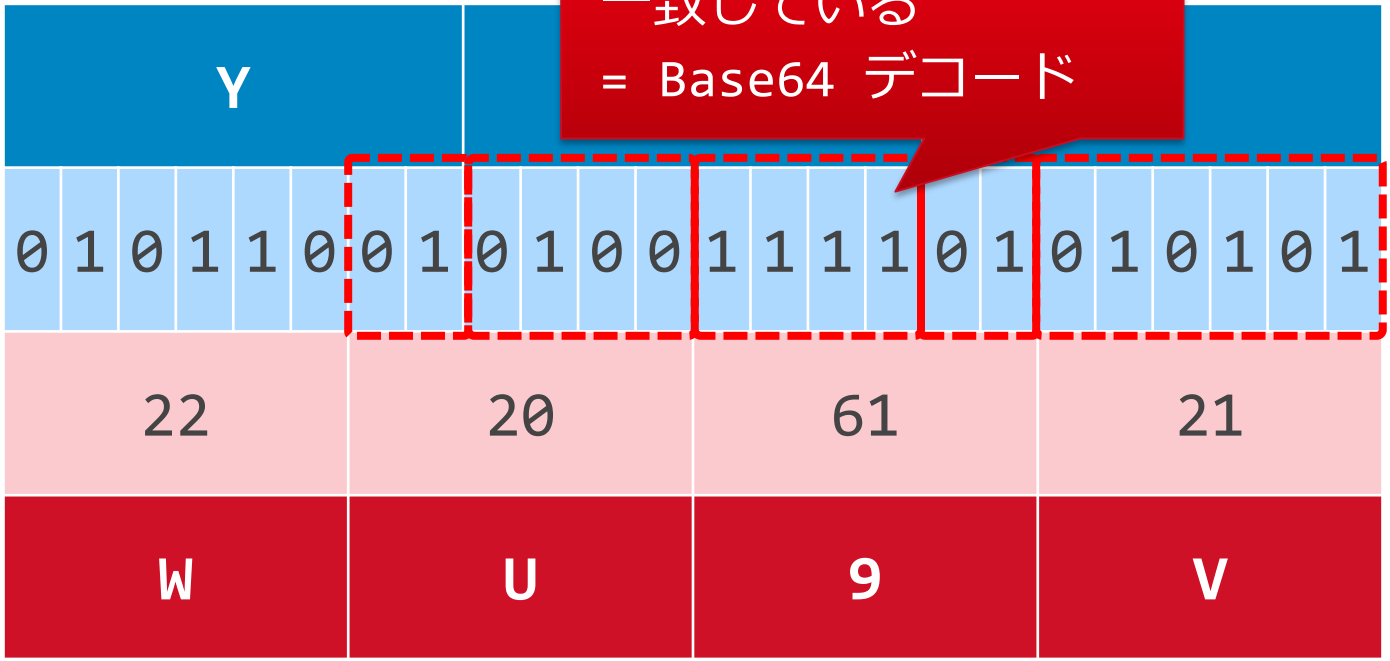
3. 暗号化/エンコード処理

- 具体的な処理を「さらっと」読み解く



3. 暗号化/エンコード処理

- Base64



3. 暗号化/エンコード処理

- 具体的な処理を「さらっと」読み解く

```
0041721A 0C4                    mov     edx, [ebp+var_4]
0041721D 0C4                    mov     eax, offset dword_417308
00417222 0C4                    call    __linkproc__ LStrPos
00417227 0C4                    mov
0041722A 0C4                    cmp
0041722E 0C4                    jz
00417230 0C4                    mov
00417233 0C4                    sub
00417236 0C4                    inc
00417237 0C4                    mov     [ebp+var_10], eax
00417300                                dd FFFFFFFFh, 1
00417308        dword_417308    dd 21h
```

0x21 = !
Base64 のパディング

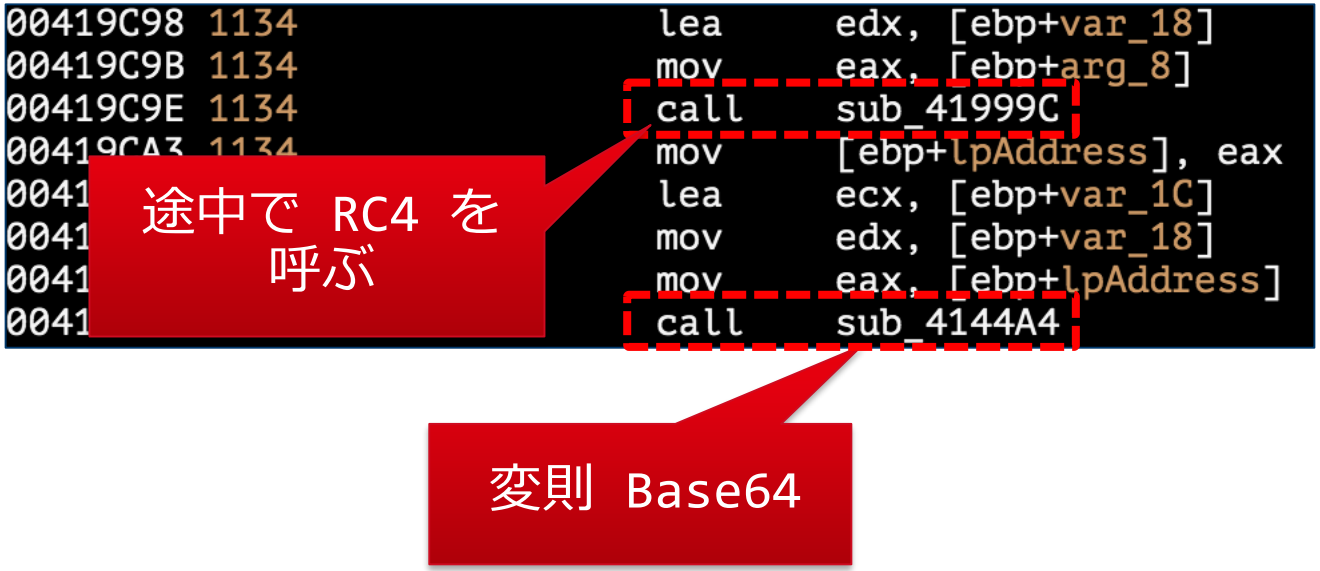
変則 Base64 デコード
変換テーブル:
GKLMopqr6vQBRSwxyz0H1234ZabTcdefg789^/CDAENOPUIVWYJhijklXstu5mnF!

4. 暗号化されたデータの復号

- Daserf が実行したコマンドの結果が暗号化される方式と同じ方式で暗号化した文字列の復号

4. 暗号化されたデータの復号

- デコードスクリプトの実装
  - 問2、3 の解析結果 + α で復号可能



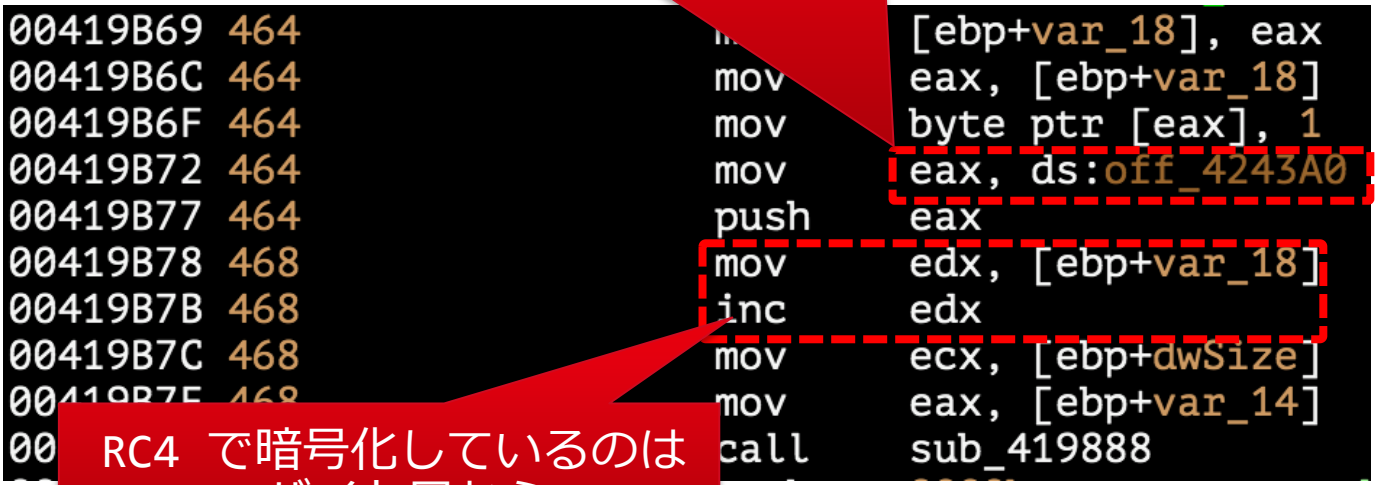
4. 暗号化されたデータの復号

- sub_41999C の前半

4. 暗号化されたデータの復号

- sub_41999C の後半

RC4鍵:
bn32gkj324kgd43<dkh6Jdhghsd?dgh;y4dsgserd



RC4 で暗号化しているのは
2バイト目から

4. 暗号化されたデータの復号

変則 Base64 デコード

2バイト目以降を RC4 復号

復号したデータを LZNT1 で展開

4. 暗号化されたデータの復号

```
In [1]: %paste
import string, base64, lznt1, re, struct
from Crypto.Cipher import ARC4
standard_b64table = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/='
daserf19_b64table = 'GKLMopqr6vQBRSwxyz0H1234ZabTcdefg789^/CDAENOPUIVWYJhijklXstu5mnF!'
rc4key = 'bn32gkj324kgd43<dkh6Jdhghsd?dgh;y4dsgserd'

def custom_b64decode(s, custom_table):
    s = s.translate(string.maketrans(custom_table, standard_b64table))
    return base64.b64decode(s)

def decrypt(data):
    enc = custom_b64decode(data, daserf19_b64table)
    rc4 = ARC4.new(rc4key)
    dec = rc4.decrypt(enc[1:])
    out = lznt1.dCompressBuf(dec)
    return out
## -- End pasted text --

In [2]:
decrypt("GeB57VGaerHM0y8Wg73sZE^gpfmCpDPAYUBCIU32ahwKDKje/^BNlnav4u9hs^Gwo9GQq7L3XXs6AK
oHPvVF/DPSGJC8")
Out[2]: "Congratulations! Now you can decrypt Daserf's GET request!"
```

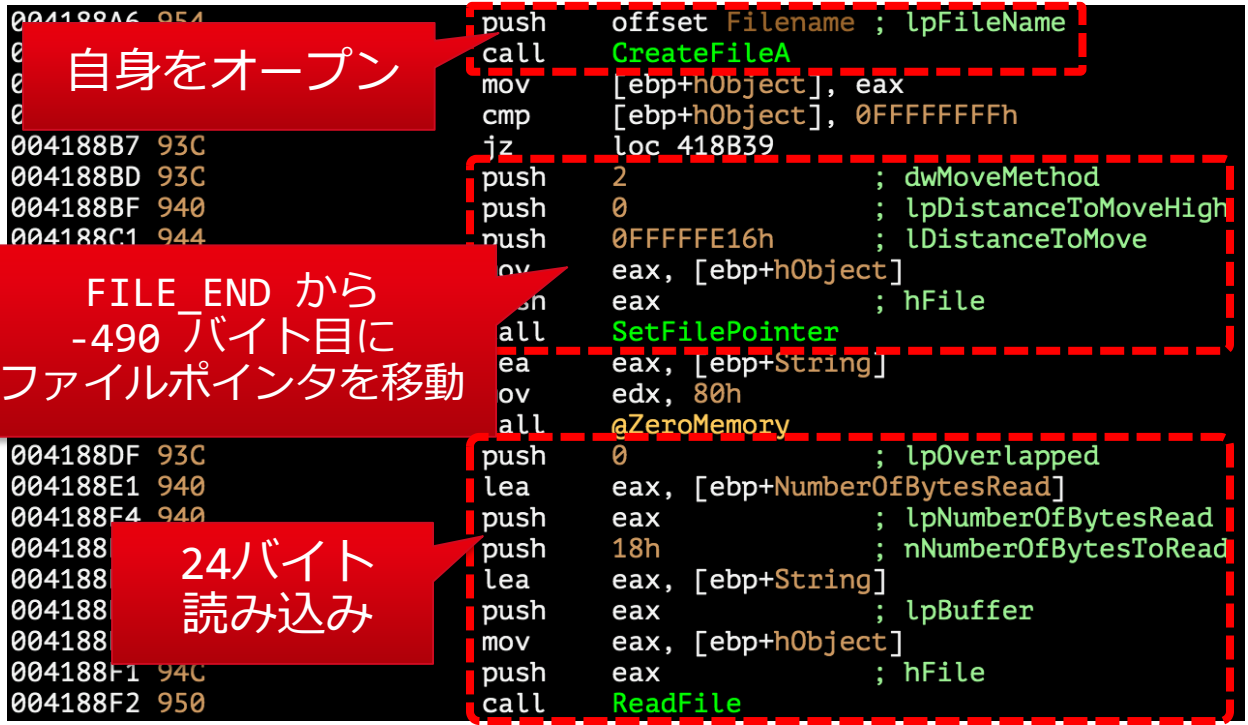
5. 設定情報の調査

- 格納先、暗号/エンコード方式

```
004188AB 968                call    CreateFileA
004188B0 94C                mov     [ebp+hFile], eax
004188B3 94C                cmp     [ebp+hFile], 0FFFFFFFFh
004188B7 94C                jz      loc_418B39
004188BD 94C                push    2                   ; dwMoveMethod
004188BF 950                push    0                   ; lpDistanceToMoveHigh
004188C1 954                push    0FFFFFE16h          ; lDistanceToMove
004188C6 958                mov     eax, [ebp+hFile]
004188C9 958                push    eax                 ; hFile
004188CA 95C                call    SetFilePointer
004188CF 94C                lea     eax, [ebp+Buffer] ; void *
004188D5 94C                mov     edx, 80h            ; unsigned int
004188DA 94C                call    Windows::ZeroMemory(void *,uint)
004188DF 94C                push    0                   ; lpOverlapped
004188E1 950                lea     eax, [ebp+NumberOfBytesRead]
004188E4 950                push    eax                 ; lpNumberOfBytesRead
004188E5 954                push    18h                 ; nNumberOfBytesToRead
004188E7 958                lea     eax, [ebp+Buffer]
004188ED 958                push    eax                 ; lpBuffer
004188EE 95C                mov     eax, [ebp+hFile]
```

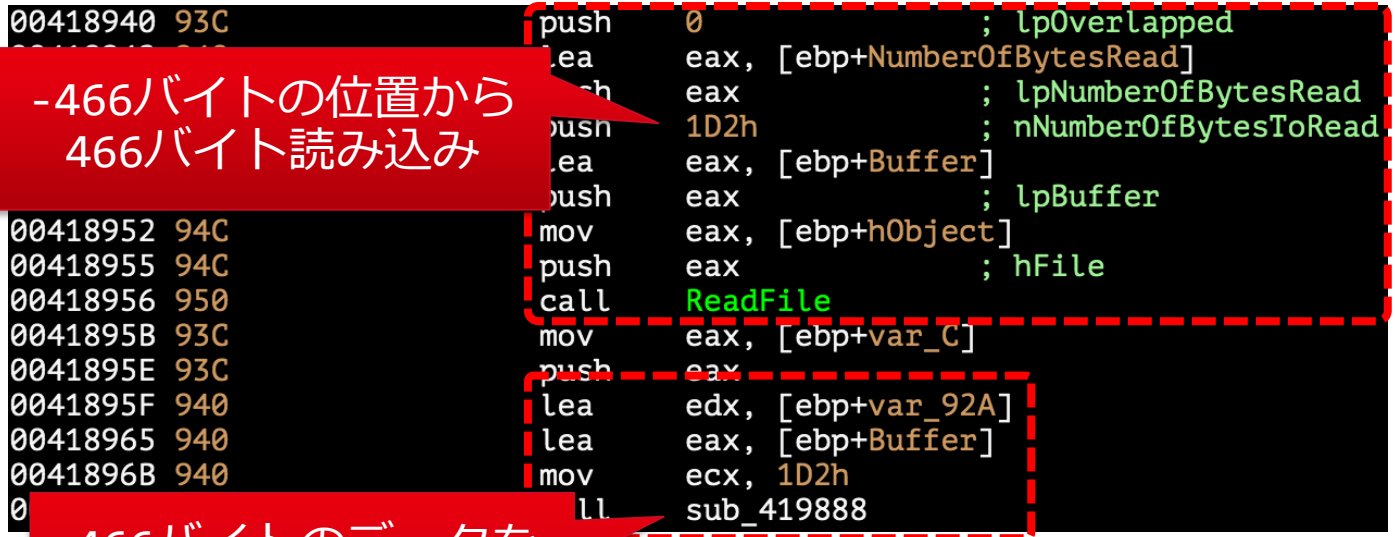
5. 設定情報の調査

- データの読み込み 1



5. 設定情報の調査

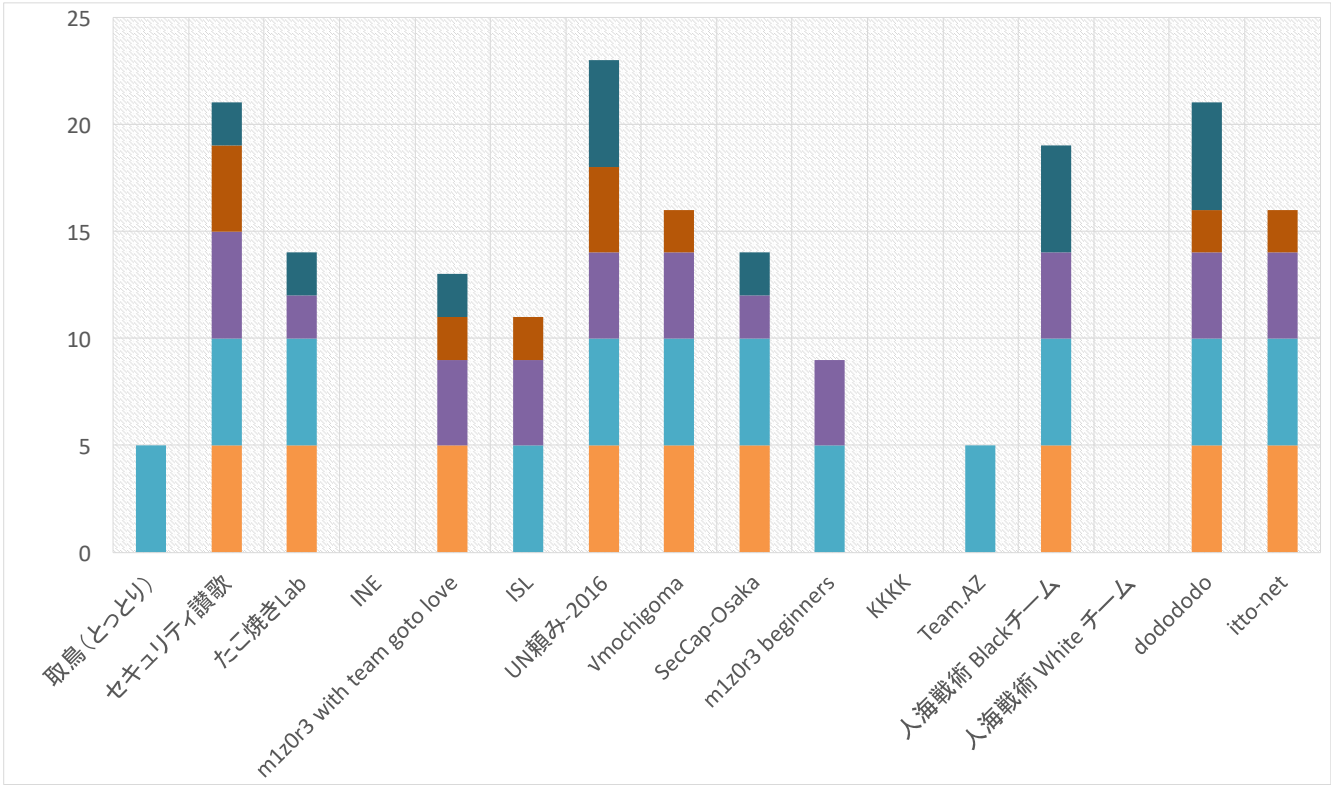
- データの読み込み 2



5. 設定情報の調査

- ファイル末尾に設定データを保持

採点結果


25
20
15
10
5
0
取鳥(とっとり)
セキュリティ讃歌
たこ焼きLab
INE
m1z0r3 with team goto love
ISL
UN頼み-2016
Vmochigoma
SecCap-Osaka
m1z0r3 beginners
KKKK
Team.AZ
人海戦術 Blackチーム
人海戦術 Whiteチーム
dododo
itto-net

来年度に向けて

SecureWorks
A Dell Company

自分たちの課題

0 か 1 か問題の解決

- ある程度解消できたがまだまだ

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
- 現状のやり方では現場でリアルなマルウェアを見ている人が必須

作成委員不足

- みんな静的解析やろう！
- セキュリティ賛歌・UN頼みあたりが良さそう

みんなの課題

- 静的解析の知見を溜めよう

Web

- セキュリティ・キャンプ全国大会2015でのマルウエア分析講義(2015-09-10)
  https://www.jpcert.or.jp/magazine/acreport-seccamp.html
- リバースエンジニアリング入門 - @IT
  http://www.atmarkit.co.jp/ait/series/2614/
- Edomae 2015 - マルウェアを解析してみよう
  http://www.slideshare.net/SatoshiMimura/edomae-2015

書籍

- Practical Malware Analysis | No Starch Press
  https://www.nostarch.com/malware
- O'Reilly Japan - アナライジング・マルウェア
  https://www.oreilly.co.jp/books/9784873114552/

過去問

- ML に投げてるので、それを入手する
- データセットに入っている資料を読む


Thank you!
SecureWorks
A Dell Company