



# MWSCup2016 課題3解説

株式会社 F F R I  
<http://www.ffri.jp>

## 課題作成者

- 大月勇人 (NTTセキュアプラットフォーム研究所)
- 森下知哉 (NTTセキュリティ・ジャパン株式会社)
- 村上純一 (株式会社FFRI)

## 課題3 / 概要及び狙い

### ■ 概要

FFRI Datasetを題材としたマルウェア動的解析ログの分析

### ■ 狙い

(課題3-1)マルウェア動的解析ログの構造理解、目視・手動分析

(課題3-2)構造理解した上での大量データの自動処理

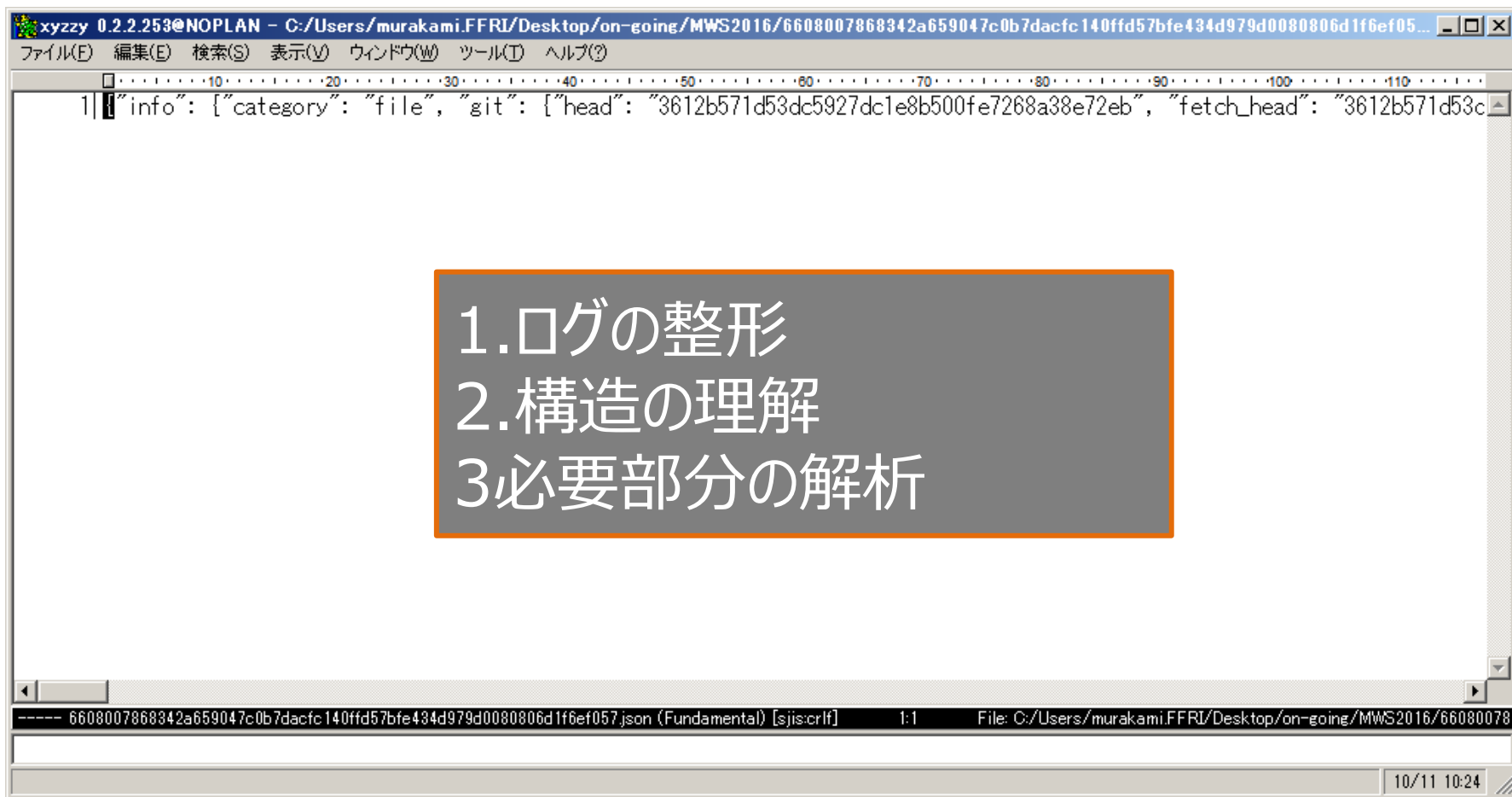
## 課題3 / 課題3-1

- FFRI Dataset中の下記の検体ログに関して  
他プロセスへのコードインジェクションと考えられる挙動を解析

```
(1) bd2953e64396acb23d9d484a58ccb9a95d12b5d897a399448a39fb170fa34ad8  
(2) 5096dda2bf508cd6932484e862325fc67802985e6c8b0af22ea6a4d49fd3b820  
(3) 15d65de0fd47a3deb5fb390050283d81b56235b2a77ac9ae97d71714cff7f76
```

- 当該挙動について以下を明らかにする
  - コード挿入元プロセス情報 (PID、プロセス名)
  - コード挿入先プロセス情報 (PID、プロセス名)
  - コードの挿入方法
  - 挿入コードの実行開始方法

## 実際のFFRI Dataset(json形式)

A screenshot of a Notepad++ window. The title bar reads "xyzyy 0.2.2.253@NOPLAN - C:/Users/murakami.FFRI/Desktop/on-going/MWS2016/6608007868342a659047c0b7dacfc140ffd57bfe434d979d0080806d1f6ef05...". The menu bar includes "ファイル(F)", "編集(E)", "検索(S)", "表示(V)", "ウインドウ(W)", "ツール(T)", and "ヘルプ(?)". The status bar at the bottom shows "---- 6608007868342a659047c0b7dacfc140ffd57bfe434d979d0080806d1f6ef057.json (Fundamental) [sjis:crlf] 1:1 File: C:/Users/murakami.FFRI/Desktop/on-going/MWS2016/66080078" and the time "10/11 10:24". The main text area contains a single line of JSON: 

```
1|{"info": [{"category": "file", "git": [{"head": "3612b571d53dc5927dc1e8b500fe7268a38e72eb", "fetch_head": "3612b571d53c
```

- 1.ログの整形
- 2.構造の理解
- 3.必要部分の解析

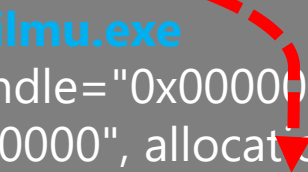
## 基本的な解法

### 1. 自分以外のプロセスに対して何らかの操作をしているAPI呼び出しを探す

- まずは"process\_handle != 0xffffffff"で呼び出されているAPIを探す
  - 0xffffffff (x64なら0xffffffffffffffff)はCurrentProcessを示すハンドル
- 見つけたAPIの呼出し元が**コード挿入元プロセス**(の候補), process\_handleが示すプロセスが**コード挿入先プロセス**(の候補)

```
1461202528.318561: 2024.1600 ilmu.exe
NtCreateUserProcess((略), thread_handle="0x0000022c",
filepath="C:¥¥WINDOWS¥¥SysWOW64¥¥explorer.exe", flags_thread=1,
process_name="", command_line="¥"C:¥¥WINDOWS¥¥SysWOW64¥¥explorer.exe¥",
(略), process_handle="0x00000230") => 0
```

```
1461202528.458561: 2024.1600 ilmu.exe
NtMapViewOfSection(section_handle="0x00000234", commit_size=0, win32_protect=64,
buffer="", base_address="0x00470000", allocation_type=0, section_offset=0,
view_size=184320, process_handle="0x00000230") => 0
```



## 基本的な解法

### 2. 見つけたAPIの中からコード挿入に相当するAPIを見つける

- 指定したプロセスのメモリへ書き込むAPI:  
WriteProcessMemory, NtWriteVirtualMemory など
- 共有メモリ領域を作成するAPI: NtMapViewOfSection など
  - 与えられている `section_handle` でAPIログを検索すると、共有メモリ領域自体の情報も取得できる

挿入先プロセスでは  
0x00470000がベースアドレス

```
1461202528.458561: 2024.1600 ilmu.exe  
NtMapViewOfSection(section_handle="0x00000234", commit_size=0, win32_protect=64,  
buffer="", base_address="0x00470000", allocation_type=0, section_offset=0,  
view_size=184320, process_handle="0x00000230") => 0
```

挿入元プロセスでは  
0x00590000がベースアドレス

```
1461202528.458561: 2024.1600 ilmu.exe  
NtMapViewOfSection(section_handle="0x00000234", commit_size=0, win32_protect=4,  
buffer="", base_address="0x00590000", allocation_type=0, section_offset=0,  
view_size=184320, process_handle="0xffffffff") => 0
```

## 基本的な解法

### 3. 挿入したコード領域を実行するスレッドを見つける

- 挿入先プロセス内に作成されたスレッドを示す **thread\_handle** に着目する
- 新規スレッドを作成するAPI:  
CreateRemoteThread, NtCreateThreadEx など
- 既存スレッドのコンテキストを変更するAPI:  
NtSetContextThread, NtQueueApcThread など
- スレッドの実行を停止・再開するAPI:  
NtSuspendThread, NtResumeThread など

プロセス生成時に  
一緒に作成されたスレッド

```
460564222.529554: 224.476 94BBBC6850981D8A576A2150B20109BE93A83D41.exe
NtCreateUserProcess((略), thread_handle="0x00000080",
filepath="C:¥¥WINDOWS¥¥SysWOW64¥¥explorer.exe", flags_thread=1,
process_name="", command_line="¥"C:¥¥WINDOWS¥¥SysWOW64¥¥explorer.exe¥"", (略),
process_handle="0x000000b0") => 0
```

```
1460564224.029554: 224.476 94BBBC6850981D8A576A2150B20109BE93A83D41.exe
NtSetContextThread(registers={eip=6580218, (略)}, thread_handle="0x00000080") => 0
```



## 各検体の挙動概要

### ■ 検体1

- explorer.exe (WOW64) 起動 (NtCreateUserProcess)
- 共有メモリ領域作成 (NtMapViewOfSection)
- 共有領域をエントリポイントとするスレッドを作成 (NtCreateThreadEx)

### ■ 検体2

- explorer.exe (WOW64) 起動 (NtCreateUserProcess)
- 共有メモリ領域作成 (NtMapViewOfSection)
- メモリ書き込み (WriteProcessMemory)
- 待機処理らしきもの  
(NtDelayExecution→NtSuspendThread→NtGetContextThread  
→NtResumeThread→NtDelayExecution→…)
- 挿入領域を実行開始するようなコンテキスト変更(NtSetContextThread)後,  
実行再開 (NtResumeThread)

### ■ 検体3

- svchost.exe (WOW64) 起動 (CreateProcessInternalW)
- 共有メモリ領域作成 (NtMapViewOfSection)
- APC追加 (NtQueueApcThread) 後, 実行開始 (NtResumeThread)

## 課題3 / 課題 3-2

- 問題内容

- FFRI Dataset 2016 未収録の2検体分のログ(LOCKY)を提供
- FFRI Dataset 2016 のログから指定した50検体分のログを対象とし、APIコールログから上記検体ログの亜種と考えられるものを根拠と共に回答

- 問題の狙い

- マルウェアの特徴を捉える力の向上
- 捉えた特徴を多数の特徴を大量のデータから処理する仕組みの検討  
→ 特定のマルウェアファミリーに対する解析の効率化

## 評価のポイント (配点:13点)

- 正しい亜種を抽出できているか(1個につき加点1.5)
- 亜種ではないログを対象としていないか(1個につき減点-1)
- 特徴を捉えてるAPIコールログ(1個につき加点1)

## APIコール順からみられるマルウェアの特徴

	CryptAcquireContextA
	CryptCreateHash
	CryptHashData
①	RegCreateKeyExA
	RegQueryValueExA ×3
	GetTempPathW
	～中略～
	GetSystemMetrics
	LdrGetDllHandle
②	CryptAcquireContextA
	CryptCreateHash
	CryptHashData
	InternetCrackUrlA

## 亜種と考えられるマルウェアの類似点①

### ■ 提供検体

2つともHKEY\_CURRENT\_USER¥¥SOFTWARE¥配下に特徴的なレジストリを保持

```
"api": "RegQueryValueExA", "regkey":  
"HKEY_CURRENT_USER¥¥SOFTWARE¥¥66VNx456¥¥RSJ4BICd2I"  
"HKEY_CURRENT_USER¥¥SOFTWARE¥¥66VNx456¥¥RM1q748M0PW00"  
"HKEY_CURRENT_USER¥¥SOFTWARE¥¥66VNx456¥¥u2A7Z1bl56k57"
```

### ■ FFRI Dataset 2016中の初期化Lockyの場合(#)

```
"api" : "RegQueryValueExA", "regkey" :  
"HKEY_CURRENT_USER¥¥SOFTWARE¥¥Locky¥¥id"  
"HKEY_CURRENT_USER¥¥SOFTWARE¥¥Locky¥¥pubkey"  
"HKEY_CURRENT_USER¥¥SOFTWARE¥¥Locky¥¥paytext"
```

# 002f8158966ab89ef6e0c33bc79708653002af90c5f7685154813b4856169b54.json

## 亜種と考えられるマルウェアの類似点②

- 外部に対する通信挙動において“CryptHashData” に自身のOS情報を含む
- APIコールの順序、上記引数の特徴からURLのファイル名部分の変化を確認

### ■ 提供検体

```
“api”: “CryptCreateHash”, “algorithm_identifier”: “CALG_MD5”  
“api”: “CryptHashData”, “buffer”: “id=9E4AC6B8E6954B8A&act=getkey&  
    affid=13&lang=ja&corp=0&serv=0&os=Windows+8&sp=0&x64=1”  
“api”: “InternetCrackUrlA”, “regkey”: “http://xxx.xxx.xxx.xxx/submit.php”
```

### ■ FFRI Dataset検体の一部

```
“api”: “CryptCreateHash”, “algorithm_identifier”: “CALG_MD5”  
“api”: “CryptHashData”, “buffer”: “id=9E4AC6B8E6954B8A&act=getkey&  
    affid=13&lang=ja&corp=0&serv=0&os=Windows+8&sp=0&x64=1”  
“api”: “InternetCrackUrlA”, “regkey”: “http://xxx.xxx.xxx.xxx/main.php”
```