

MWS Cup 作問側の話



このプレゼンについて

- MWSCupの作問意図(参加者に学び取ってほしいこと)や苦労について共有
- どんな感じで作問しているか、その面白さやメリットをお伝えして、作問協力したい人の後押しになれば



作問の大まかな流れ

DFIR&Offensive security、表層解析、静的解析ともおおよそ下記の流れで作問を実施

- ~6月: プレミーティング前後あたりで、メンバの募集・アサイン
- 6月: プレミーティング実施
- 7月: 初顔合わせ(大体オンライン)および問題案の方向性相談
- 8月~9月: 問題案骨子の作成、必要なデータの取得・作成
- 9月末~10月頭: 問題完成、ポータルサイトの作成
- 10月中旬: 問題文、回答の決定、ポータルサイトへの掲載、回答解説資料の作成、競技者への周知
- 10月下旬: 競技本番

- 今回は私が参加した静的解析での作問を例にお伝えしたいと思います

2025の問題担当

- 主担当
 - 中島 将太（株式会社サイバーディフェンス研究所）
- 問題作成委員
 - 桑原 翼（株式会社FFRIセキュリティ）
 - 末廣 繁樹（株式会社エヌ・エフ・ラボラトリーズ）
 - 山田 裕彌（株式会社ラック）
 - 川越 謙宏（工学院大学）
 - 仲川 宜秀（NTT西日本株式会社）
 - 二瓶 凌輔（NTT西日本株式会社）
 - 緒方 淳己（NTT西日本株式会社）
 - 根津 泰之（NTT西日本株式会社）

静的解析課題のテーマ

- マルウェアを正しく理解する
 - 課題を通して解析のポイントを学習する
- 最新情報を得る
 - 最近のin-the-wildなマルウェアを扱う
- 実務に近い作業
 - マルウェアのトレンドに沿った出題
 - マルウェアのコードの理解
 - 静的解析による復号スクリプトの作成
 - 最新の技術を活用して効率よく解析する(LLMの利用など)

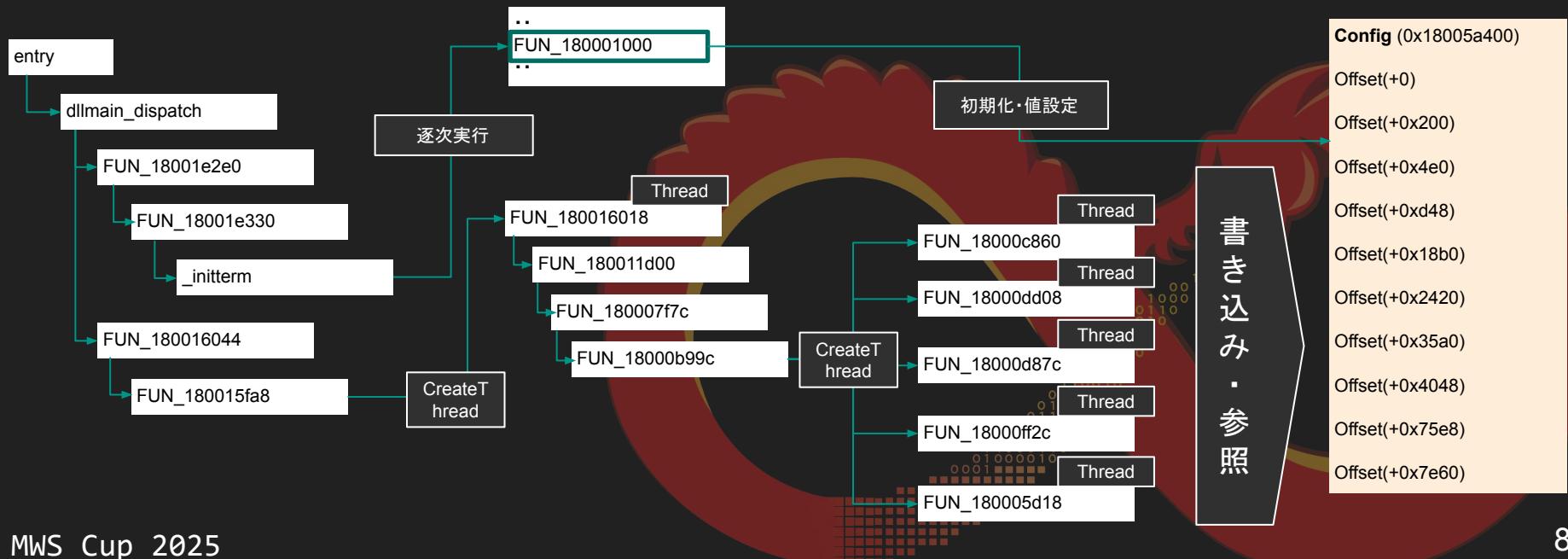
今回のマルウェア

- APT-C-60に利用されたRAT(DLL)
(ファミリ名 : SpyGlace)
- 2024年にESET/JPCERT等で
解説記事あり
- API難読化、通信暗号化、Config情報
を用いた動きなど、作問向け

The screenshot shows a news article from JPCERT/CC Eyes. The title is "正規サービスを悪用した攻撃グループAPT-C-60による攻撃". The author is listed as "亀井 智矢(Tomoya Kamei)". The date is "2024/11/26". The article content discusses an attack by the APT-C-60 group using legitimate services. It includes a section titled "マルウェア感染までの流れ" with bullet points: "マルウェア感染までの流れ", "ダウンドローダーの分析", "バックドアの分析", and "同種のマルウェアを使用したキャンペーン".

マルウェアの動きの概要

- Config情報を作成しそれらをもとに複数のThreadを起動、C2通信やコマンド実行を実施



問題一覧と作問意図

静的解析で対象となるAPI難読化読解、Config読解、通信で使用されている暗号の復号、IoC抽出（Mutex）・ファミリ特定まで幅広く出題。

ファイル配布 0	1-2. 文字列の難読化 2	1-3. APIの構造体 2	1-4. コマンド1 2
1-5. コマンド2 2	2-1. Config読解 2	2-2. Config読解 2	3-1. 通信データの符号化・暗号化 2
4-1. 文字列加工 2	1-1. 動的APIアドレス解決 3	3-2. 通信データの符号化・暗号化 3	4-2. ファミリ名 3

問題ジャンル

- API難読化解除
 - 1-1
 - 1-2
 - 1-3
- コマンド解析
 - 1-4
 - 1-5
- コンフィグ解析
 - 2-1
 - 2-2

- 復号系
 - 3-1
 - 3-2
- アトリビューション
 - 4-1
 - 4-2

問題一覧と作問意図

API難読化解除

- 1-1. 動的APIアドレス解決
- 1-2. 文字列の難読化
- 1-3. APIの構造体

マルウェアが解析妨害を行う際によく使用されるAPI難読化や動的アドレス解決はよく使用されており、それを理解することは解析者にとって重要

3

▲0 ▾0

今回提供されたマルウェアは、静的解析妨害のため、IAT(Import Address Table)を用いないAPIの呼び出しを行っている。1-1.txtをダウンロードして、カッコ(1)~(14)に当てはまるものを選択肢から選んで回答せよ。

(1)~(14)に当てはまる全角カタカナをコンマ(,)ですべてつなげた形式で回答すること。例えば、(1)ア(2)イ(3)ウ(4)エ(5)オ(6)カ(7)キ(8)ク(9)ケ(10)コ(11)サ(12)シ(13)ス(14)セの場合、解答はア,イ,ウ,エ,オ,カ,キ,ク,ケ,コ,サ,シ,ス,セとなる

2

▲0 ▾0

このマルウェアは解析妨害のために、Windows APIを含むマルウェアの動作に必要な文字列をエンコードして保持している。実行時には、関数FUN_180016408によってエンコードされた文字列を、保持するエンコード済み文字列と照合することで、使用する文字列を特定する。

とある文字列のエンコード結果が以下の文字列である。元の文字列を答えよ。

6R;cU57c76WYc7LG2F7

0/3 attempts

IATを使用せずにAPIの関数を実行するためには、DLLがロードされる場所を示すベースアドレスをもとに、使用したいAPIの実行アドレスを取得する処理が必要となる。

まず、DLLのベースアドレスを取得するために、プログラムにロードされているDLL情報を確認する。DLLのベースアドレスは、(1)構造体を利用して特定でき、本マルウェアでは関数(2)において以下①~④のような流れで取得している。

- ① (1) 構造体のオフセット(3)の(4)よりPEB_LDR_DATA構造体へのポインターを取得する
- ② PEB_LDR_DATA構造体のオフセット(5)の(6)によりDLL情報が格納された構造体の双向リストを取得する
- ③ ②で取得した双向リストはLDR_DATA_TABLE_ENTRY構造体のオフセット0x10のInMemoryOrderLinksを指しており、そこから0x48を加算したオフセット0x58の(7)と対象のDLLが一致するかを確認する。本マルウェアでは、関数(2)の第二引数が対象のDLLの種類を表しており、第二引数が4のときのDLLは(8)である。そして(7)と対象のDLLが一致すると、LDR_DATA_TABLE_ENTRY構造体のオフセット0x10に0x20を加算したオフセット0x30より(9)を取得する
- ④ 今回のマルウェアでは関数FUN_180017690において、③で取得したDLLベースアドレスをコンフィグ情報としてある領域に格納しており、(8)の場合の格納アドレスは(10)である。

次に、これまでの処理で取得したDLLのベースアドレスをもとに、DLLに含まれるAPIの実行アドレスを取得する。(8)のDLLでは関数(11)においてこの処理を以下の⑥~⑨のような流れで取得している。

- ⑤ DLLのDOSヘッダのオフセット(12)よりPEヘッダの先頭アドレスを特定する
- ⑥ PEヘッダの先頭からオフセット0x88にあるIMAGE_EXPORT_DIRECTORY構造体を取得する
- ⑦ ⑥で取得した構造体のオフセット0x18、0x20を参照することでそれぞれIMAGE_EXPORT_DIRECTORYの要素である(13)、(14)を取得する
- ⑧ ⑦で取得したAPIの関数名リストとAPIの関数名を暗号化した文字列とを比較し一致した場合は、AddressOfNameOrdinalsを取得し関数名リストの先頭からAddressOfNameOrdinals分移動した位置のAPIのアドレスを取得しDLLのベースアドレスを加算することでAPIアドレスを取得する。
- ⑨ ⑧で取得したAPIアドレスはコンフィグ情報として格納する

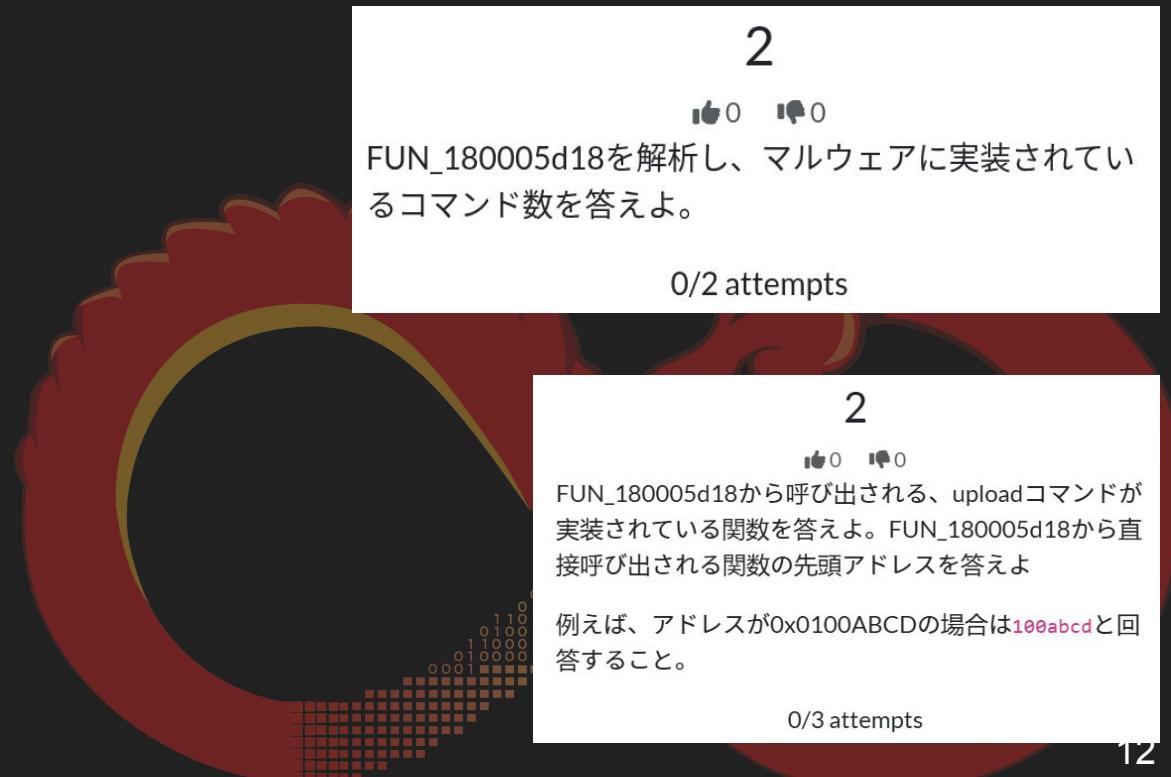
問題一覧と作問意図

コマンド解析

1-4. コマンド

1-5. コマンド2

RATマルウェアが攻撃者に対してどのような機能を提供するかを知ることは、被害範囲の特定につながる。RAT—C2サーバ間のコマンドを解析することで理解を深める



2

FUN_180005d18を解析し、マルウェアに実装されているコマンド数を答えよ。

0/2 attempts

2

FUN_180005d18から呼び出される、uploadコマンドが実装されている関数を答えよ。FUN_180005d18から直接呼び出される関数の先頭アドレスを答えよ

例えば、アドレスが0x0100ABCDの場合は100abcdと回答すること。

0/3 attempts

問題一覧と作問意図

コンフィグ解析

2-1. Config読み解

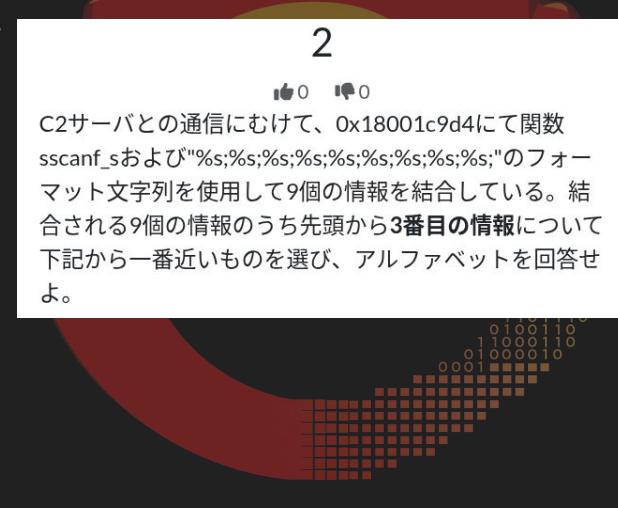
2-2. Config読み解

RATマルウェアがC2サーバに送付する情報をたどり、どのような情報を収集・Config情報として保有しているか・C2サーバに送信するかは被害特定やIoCの作成に有用。

2

0 0

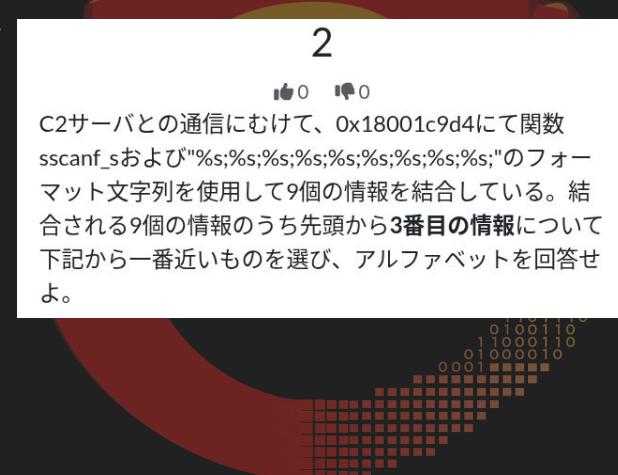
C2サーバとの通信にむけて、0x18001c9d4で関数 `sscanf_s` および "%s;%s;%s;%s;%s;%s;%s;%s;" のフォーマット文字列を使用して9個の情報を結合している。結合される9個の情報のうち先頭から**2番目の情報**について下記から一番近いものを選び、アルファベットを回答せよ。



2

0 0

C2サーバとの通信にむけて、0x18001c9d4にて関数 `sscanf_s` および "%s;%s;%s;%s;%s;%s;%s;%s;%s;" のフォーマット文字列を使用して9個の情報を結合している。結合される9個の情報のうち先頭から**3番目の情報**について下記から一番近いものを選び、アルファベットを回答せよ。



【選択肢】

- a. クライアントのインターフェイス名
- b. クライアントのプロセッサの名称
- c. クライアントOSのインストール言語
- d. クライアントのコンピュータ名
- e. クライアントのCPUアーキテクチャ
- f. クライアントのユーザ名
- g. クライアントのグローバルIPアドレス
- h. クライアントのインストール日時
- i. マルウェアのバージョン
- j. クライアントのOSのプロダクト名とバージョン
- k. クライアントのマシンの製造元
- l. クライアントのRecent フォルダ情報
- m. クライアントのautoexec フォルダ情報
- n. クライアントのmodulepath情報
- o. C2通信のtoken情報
- p. C2通信用のユーザ名
- q. C2サーバのグローバルIPアドレス
- r. C2サーバのAPIエンドポイント
- s. C2サーバ向けのコマンド文字列

0/2 attempts

問題一覧と作問意図

復号系

- 3-1. 通信データの符号化・暗号化
- 3-2. 通信データの符号化・暗号化

RATマルウェアがC2サーバに送付する情報をどのように暗号化しているかについて解析することで、どのようなデータが送信されているか被害特定につなげる

- a. RC4
- b. RC5
- c. RC6
- d. AES-128
- e. AES-256
- f. DES
- g. TEA
- h. XXTEA
- i. VEST
- j. Blowfish

0/1 attempt

2

1 0 1 0

このマルウェアは通信のデータを符号化・暗号化している。FUN_180019c24関数では、これらの処理に必要な初期化をおこなっている。この処理に関連する通信データの暗号化を解析して、以下の選択肢からどのアルゴリズムをカスタムしたものか下記から選び、アルファベットで答えよ。

3

1 0 1 0

このマルウェアは通信のデータを符号化・暗号化している。FUN_180019c24関数では、これらの処理に必要な初期化をおこなっている。この処理に関連する通信データの符号化・暗号化を解析して、以下のデータを復号せよ。

U5hwD1ouNDUV7fk1PueeMcWIH829sydPQij/i7numExoSF1
OAwrH3/A+scU=

0/3 attempts

問題一覧と作問意図

アトリビューション

4-1. 文字列加工

4-2. ファミリ名

マルウェア解析を通してIoC情報を作る元ネタや、OSINTに関わる情報について理解を深める。これらの情報をもとに過去事例を調査することで効率よく解析を行うことができる。



2

0x180042c00 に、UTF16-LE文字列 K31610KI09834PG75459K が存在する。その文字列は加工され、特定の用途で使用される。使用用途を次の4種類のうちから選び、実際に使用される加工結果と合わせて、アンダースコアで区切って答えよ。

- a. 通信を復号するための鍵
- b. 二重起動を防ぐための一意な値
- c. 暗号化された関数名
- d. 設定情報を保存するためのファイル名

例えば、用途がa、加工結果の文字列が Example である場合は a_Example と答えよ。

0/2 attempts

3

これまでの解析結果を踏まえて、このマルウェアを使用する攻撃グループとマルウェアファミリの名称を答えよ。

例えば攻撃グループがAでマルウェアファミリがXの場合は A_X と回答すること

0/3 attempts

Flag

Submit

作問経緯

8/5～: 対象の検体を各自で2～3個選定

8/27(第1回Mtg): 検体の方向性の確認(Rust検体第一候補、次点API難読化検体)

9/6: Rust検体断念、次点検体のfpspy/SpyGlaceのどちらにするかを引き続き確認

9/16: 検体確定(SpyGlace)、以降問題案の作成(問題案17問ほど)

9/26(第2回Mtg): 問題案の確認、選別(問題の統廃合により11問)

10/16: 問題をほぼFix、CTFdにアップロード完了

10/20: LLMでのトライアル回答実施

10/27: ぎりぎりまで資料修正、チェック、トラブル対応

10/28: 本番



検体選定

- 当時は下記の考え方で検体を持ち寄り
 - 昨年度はGoのRansomware検体。今年度はRustマルウェアも増えてきていることからRustでRansomware以外の検体を中心に調査
- 下記の理由によりRustを断念。問題難易度や問題個所の観点からSpyGlaceを選定
 - 持ち寄ったRust検体を動かす際にDecompilerが非常に遅く、Rust固有の処理みたいのがなかったため、問題として微妙と判断
 - API難読化の問題の要望もあったので、これらを含むSpyGlaceを選定

作問経緯

- 作問はGoogleDocsに記載し、コメントをしあう、またコミュニケーションはDiscordで行うスタイル
 - 作問ごとにLLMを想定した対策を実施。

S Cup 2025

作問経緯(仲川の場合)

- Config読解の2問を主に担当(2人×1時間くらいで想定。)
- LLM対策としてAPI難読化を説いたうえで、複数の関数にまたがって解析を行う必要がある問題(Middleレベル)
- 問題の回答の確認について、Isolation環境を準備し、デバッガを用いて動作を動的に確認実施。
暗号等については、実際にC2サーバ・プロトコル模擬を実施し動作を確認

```
File Edit View Bookmarks Settings Help
logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

# 2. Custom Handler with logging
class MyHTTPSHandler(http.server.SimpleHTTPRequestHandler):
    def do_GET(self):
        logging.info("GET request,\nPath: %s\nHeaders:\n%s\n", str(self.path), str(self.headers))
        #self.wfile.write("GET request for {}".format(self.path).encode('utf-8'))
        super().do_GET() # Call the parent's do_GET for default behavior

    def do_POST(self):
        content_length = int(self.headers['Content-Length']) # <-- Gets the size of data
        post_data = self.rfile.read(content_length) # <-- Gets the data itself
        logging.info("POST request,\nPath: %s\nHeaders:\n%s\nBody:\n%s\n",
                    str(self.path), str(self.headers), post_data.decode('utf-8'))

        self.send_response(200)
        self.end_headers()
        #self.wfile.write(b'additiveEo4KegF2ITgfuA==')

        self.wfile.write(b'additiveEZ8XfAhxJzxI7KEeZqk=')
        #self.send_response(200)
        #self.end_headers()
        #self.wfile.write("POST request for {}".format(self.path).encode('utf-8'))

# 3. Server setup
if __name__ == "__main__":
    host = "0.0.0.0"
    port = 443 # Default HTTPS port

    ssl_keylogfile = "/tmp/tls_keys_20250216-001.log"
    os.environ["SSLKEYLOGFILE"] = ssl_keylogfile
    # Create an SSL context
    context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
    #ssl_context.set_ciphers('PSK')
```

2-2. Config読解(回答の一例)

2

1 0 1 0

C2サーバとの通信にむけて、0x18001c9d4にて関数
sscanf_sおよび"%s;%s;%s;%s;%s;%s;%s;%s;%s;"のフォー
マット文字列を使用して9個の情報を結合している。結
合される9個の情報のうち先頭から**3番目の情報**について
下記から一番近いものを選び、アルファベットを回答せ
よ。

【選択肢】

- a. クライアントのインターフェイス名
- b. クライアントのプロセッサの名称
- c. クライアントOSのインストール言語
- d. クライアントのコンピュータ名
- e. クライアントのCPUアーキテクチャ
- f. クライアントのユーザ名
- g. クライアントのグローバルIPアドレス
- h. クライアントのインストール日時
- i. マルウェアのバージョン
- j. クライアントのOSのプロダクト名とバージョン
- k. クライアントのマシンの製造元
- l. クライアントのRecent フォルダ情報
- m. クライアントのautoexec フォルダ情報
- n. クライアントのmodulepath情報
- o. C2通信のtoken情報
- p. C2通信用のユーザ名
- q. C2サーバのグローバルIPアドレス
- r. C2サーバのAPIエンドポイント
- s. C2サーバ向けのコマンド文字列

0/2 attempts

2-2. Config読解

```
164 ppppcVar8 = local_178;
165 if (0xf < local_160) {
166     ppppcVar8 = (char ****)local_178[0];
167 }
168 ppppcVar12 = local_258;
169 if (0xf < local_240) {
170     ppppcVar12 = (char ****)local_258[0];
171 }
172 ppppcVar14 = local_238;
173 if (0xf < local_220) {
174     ppppcVar14 = (char ****)local_238[0];
175 }
176 ppppcVar15 = local_218;
177 if (0xf < local_200) {
178     ppppcVar15 = (char ****)local_218[0];
179 }
180 ppppcVar17 = local_1f8;
181 if (0xf < local_1e0) {
182     ppppcVar17 = (char ****)local_1f8[0];
183 }
184 ppppcVar16 = local_1d8;
185 if (0xf < local_1c0) {
186     ppppcVar16 = (char ****)local_1d8[0];
187 }
188 plVar3 = param_3;
189 if (0x1 < (ulonglong)param_3[3]) {
190     plVar3 = (longlong *)param_3;
191 }
192 ppppppuVar7 = local_1b8;
193 if (0xf < local_1a0) {
194     ppppppuVar7 = (undefined8 *****)local_1b8[0];
195 }
196 ppppppuVar13 = local_198;
197 if (0xf < local_180) {
198     ppppppuVar13 = (undefined8 *****)local_198[0];
199 }
200 sscanf_s(local_158,"%s;%s;%s;%s;%s;%s;%s;%s:",ppppppuVar13,ppppppuVar7,(char *)plVar3
201     ,(char *)ppppcVar16,(char *)ppppcVar17,(char *)pppcVar15,(char *)pppcVar14,
202     ,(char *)pppcVar12,(char *)pppcVar8);
203 pCVar4 = FUN_180013ffc(param_1,local_158);
```

- 2-1から引き続き3つ目のデータ(第5引数)を辿る
- param_3の情報を格納していることがわかる



2-2. Config読解

- param_3の情報をトレースすると、親関数の中にあるFUN_180012438を経由しFUN_180018734の第3引数となっている

```
159 local_1f0 = 0;
160 FID_conflict::_Construct_lv_contents(local_208,(undefined8 *)local_78);
161 local_1e0[0] = 0;
162 local_1d0 = 0;
163 local_1c8 = 0;
164 std::basic_string<>::_Construct_lv_contents((basic_string<> * local_1e0 & local_118)+
165 pbVar6 = (basic_string<> *)
166     FUN_18001c6fc(param_1 + 0xd48,(basic_string<> *)local_1b8 local_1e0
167         (basic_string<> *)local_208);
168 pbVar6 = (basic_string<> *)(param_1 + 0x80b0);
```

```
66 FUN_180018f0(param_1 + 0x1b0,(basic_string<> *)local_208);
67 FUN_180012438(param_1,&local_118)+
68 lVar2 = param_1 + 0x1048;
```

```
2 void FUN_180012438(longlong param_1,basic_string<> *param_2)
3 {
4     code *pcVar1;
5     LPVOID pvVar2;
6     undefined1 auStack_68 [32];
7     undefined4 local_48;
8     basic_string<> *local_40;
9     LPVOID local_38 [2];
10    undefined8 local_28;
11    ulonglong local_20;
12    ulonglong local_18;
13
14    local_18 = DAT_1800590b8 ^ (ulonglong)auStack_68;
15    *(undefined8 *)param_2 = 0;
16    *(undefined8 *)param_2 + 0x10 = 0;
17    *(undefined8 *)param_2 + 0x18 = 0xf;
18    *(undefined8 *)param_2 + 0x20 = 0x0;
19    *param_2 = (basic_string<>*)0x0;
20    local_48 = 1;
21    local_28 = 0;
22    local_20 = 0xf;
23    local_38[0] = (LPVOID)0x0;
24    local_40 = param_2;
25    FUN_180018734(param_1 + 0x18b0,local_3d,param_2);
26    if (0xf < local_20) {
27        pvVar2 = local_38[0];
28        if ((0xffff < local_20 + 1) &&
29            (pvVar2 = *(LPVOID *)((longlong)local_38[0] + -8),
30             0x1f < (ulonglong)((longlong)local_38[0] + (-8 - (longlong)pvVar2))) {
31            FUN_1800254ac();
32            pcVar1 = (code *)swi(3);
33            (*pcVar1)();
34            return;
35        }
36        thunk_FUN_18002585c(pvVar2);
37    }
38    FUN_18001dc10(local_18 ^ (ulonglong)auStack_68);
39    return;
40}
```

```

27 local_48 = DAT_1800590b8 ^ (ulonglong)auStack_b8;
28 iVar5 = param_1 + 0x308;
29 iVar2 = FUN_180016620(lVar5);
30 iVar1 = (**(code **)(iVar2 + 0x148))(local_70);
31 if (iVar1 == 0) goto LAB_180018824;
32 iVar2 = param_1 + 0x308;
33 if (local_60 == 0) {
34     iVar3 = FUN_180016620(lVar2);
35     local_58 = 0;
36     local_60 = 0;
37     uVar9 = 0;
38 }
39 else {
40     iVar3 = FUN_180016620(lVar2);
41     uVar9 = 3;
42 }
43 local_98 = (ulonglong)local_98 << 0x20;
44 iVar3 = (**(code **)(iVar3 + 0x130))
45     ("Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/5.0)",uVar9,
46     local_60,local_58);
47 if (iVar3 == 0) goto LAB_180018824;
48 iVar4 = FUN_180016620(lVar5);
49 plVar10 = (longlong *)param_1 + 0x228; アクセス先のドメイン0x18005a400+18b0+0x228に格納)
50 if (7 < *(ulonglong *)param_1 + 0x240) {
51     plVar10 = (longlong *)*plVar10;
52 }
53 iVar4 = (**(code **)(iVar4 + 0x138))(iVar3,plVar10,0,0); WinHttpConnect
54 pauvar8 = (undefined1 (*) [10])ex;
55 if (iVar4 != 0) {
56     iVar6 = FUN_180016620(lVar2);
57     local_88 = CONCAT44(local_88,_4_4_,0x800100);
58     local_90 = 0;
59     local_98 = 0;
60     iVar6 = (**(code **)(iVar6 + 0x140))(iVar4,&DAT_180043698,0,0); WinHttpOpenRequest
61     if (iVar6 == 0) {
62         iVar5 = FUN_180016620(lVar2);
63         (**(code **)(iVar5 + 0x180))(iVar4);
64     }
65     else {
66         local_50[0] = 0x3300;
67         iVar7 = FUN_180016620(lVar2);
68         iVar1 = (**(code **)(iVar7 + 0x150))(iVar6,0x1f,local_50); WinHttpSetOption
69         if (iVar1 == 0) {
70             iVar5 = FUN_180016620(lVar2);
71             (**(code **)(iVar5 + 0x180))(iVar6);
72         }
73     }
74     iVar7 = FUN_180016620(lVar2);
75     local_88 = 0;
76     local_90 = local_90 & 0xfffffffff00000000;
77     local_98 = local_98 & 0xfffffffff00000000;
78     iVar1 = (**(code **)(iVar7 + 0x158))(iVar6,0,0,0); WinHttpSendRequest
79     if (iVar1 != 0) {
80         iVar7 = FUN_180016620(lVar2);
81         iVar1 = (**(code **)(iVar7 + 0x168))(iVar6,0); WinHttpReceiveResponse

```

FUN_180018734の解析から、
0x18005a400+18b0+0x228の文字列が
HTTPのアクセス先であり、
GETアクセスのレスポンスが第3引数に格
納されることがわかる

```

82 iVar1 != 0) { WinHttpQueryDataAvailable
83     local_78 = 0;
84     iVar7 = FUN_180016620(lVar5);
85     iVar1 = (**(code **)(iVar7 + 0x170))(iVar6);
86     if (iVar1 == 0) {
87         pauVar8 = (undefined1 (*) [16])operator_new((ulonglong)(local_78 + 1));
88         FUN_180021470(pauVar8,0,(ulonglong)(local_78 + 1));
89     }
90     if (local_78 == 0) goto LAB_180018824; WinHttpReadData
91     iVar2 = FUN_180016620(lVar2);
92     iVar1 = (**(code **)(iVar2 + 0x170))(iVar6,pauVar8,local_78,&local_74);
93     if (iVar1 != 0) {
94         (*pauVar8)[local_74] = 0;
95         std::basic_string<>::assign(param_3,(char )pauVar8);
96     }
97     if (pauVar8 != (undefined1 (*) [16])0x0) {
98         FUN_18002585c(pauVar8);
99     }
100 }
101 iVar2 = FUN_180016620(lVar5);
102 (**(code **)(iVar2 + 0x180))(iVar6);
103 iVar2 = FUN_180016620(lVar5);
104 (**(code **)(iVar2 + 0x180))(iVar4);
105 iVar5 = FUN_180016620(lVar5);
106 (**(code **)(iVar5 + 0x180))(iVar3);
107 goto LAB_180018824;
108 }
109 LAB_180018929:
110     iVar5 = FUN_180016620(lVar2);
111     (**(code **)(iVar5 + 0x180))(iVar6);
112 }
113 iVar5 = FUN_180016620(param_1 + 0x308);
114     (**(code **)(iVar5 + 0x180))(iVar4);
115 }
116 }
117 iVar5 = FUN_180016620(param_1 + 0x308);
118 (**(code **)(iVar5 + 0x180))(iVar3);
119 LAB_180018824:
120     FUN_18001dc10(local_48 ^ (ulonglong)auStack_b8);
121     return;
122 }

```

レスポンスデータ
第3引数にコピー

```

1 undefined1 (*) [16] FUN_180017e34(undefined1 (*param_1) [16])
2
3
4 {
5     longlong *plVar1;
6     longlong *plVar2;
7     longlong *plVar3;
8     longlong *plVar4;
9     longlong *plVar5;
10    longlong *plVar6;
11
12    FUN_18001308c(param_1);
13    *(undefined8 *)(&param_1[0x20] + 8) = 0;
14    *(undefined8 *)(&param_1[0x21] + 8) = 0;
15    *(undefined8 *)(&param_1[0x22] = 7;
16    *(undefined2 *)(&param_1[0x20] + 8) = 0;
17    plVar1 = (longlong *)(&param_1[0x22] + 8);
18    plVar1 = 0;
19    *(undefined8 *)(&param_1[0x23] + 8) = 0;
20    *(undefined8 *)param_1[0x24] = 7;
21    *(undefined2 *)plVar1 = 0;
22    *(undefined8 *)(&param_1[0x24] + 8) = 0;
23    *(undefined8 *)(&param_1[0x25] + 8) = 0;
24    *(undefined8 *)(&param_1[0x26] = 7;
25    *(undefined2 *)(&param_1[0x24] + 8) = 0;
26    plVar2 = (longlong *)(&param_1[0x26] + 8);
27    plVar2 = 0;
28    *(undefined8 *)(&param_1[0x27] + 8) = 0;
29    *(undefined8 *)(&param_1[0x28] = 7;
30    *(undefined2 *)plVar2 = 0;
31    plVar3 = (longlong *)(&param_1[0x28] + 8);
32    plVar3 = 0;
33    *(undefined8 *)(&param_1[0x29] + 8) = 0;
34    *(undefined8 *)(&param_1[0x2a] = 7;
35    *(undefined2 *)plVar3 = 0;
36    plVar4 = (longlong *)(&param_1[0x2a] + 8);
37    plVar4 = 0;
38    *(undefined8 *)(&param_1[0x2b] + 8) = 0;
39    *(undefined8 *)(&param_1[0x2c] = 7;
40    *(undefined2 *)plVar4 = 0;
41    plVar5 = (longlong *)(&param_1[0x2c] + 8);
42    plVar5 = 0;
43    *(undefined8 *)(&param_1[0x2d] + 8) = 0;
44    *(undefined8 *)(&param_1[0x2e] = 7;
45    *(undefined2 *)plVar5 = 0;
46    plVar6 = (longlong *)(&param_1[0x2e] + 8);
47    plVar6 = 0;
48    *(undefined8 *)(&param_1[0x2f] + 8) = 0;
49    *(undefined8 *)(&param_1[0x30] = 7;
50    *(undefined2 *)plVar6 = 0;
51    FUN_180016198(undefined1 (*) [16])(&param_1[0x30] + 8);
52    FID_conflict:assign(plVar2,(undefined8 *)L"2";:443d503086:f:e4d;aea0:3ga7ga12yqdudq~"vs",0x2c);
53    FID_conflict:assign(plVar3,(undefined8 *)L"2";:443d503086:f:e4d;aea0:3ga7ga12ohwdm~"vs",0x2c);
54    FID_conflict:assign(plVar4,(undefined8 *)L"2";:443d503086:f:e4d;aea0:3ga7ga12yqdudq~"vs",0x2c);
55    FID_conflict:assign(plVar5,(undefined8 *)L"2";:443d503086:f:e4d;aea0:3ga7ga12ohwdm~"vs",0x2d);
56    FID_conflict:assign(plVar6,(undefined8 *)L"2";:443d503086:f:e4d;aea0:3ga7ga12yqdudq~"vs",0x2c); データ'sh-hshex-rg'を格納
57    FID_conflict:assign(plVar1,(undefined8 *)L"sh-hshex-rqj",0xd);
58    (*(undefined8 *)param_1[0x20] = 0;
59    return param_1;
60 }

```

+0x228のアドレス

GhidraのSearch program textにて0x228で検索、
FUN_180017e34にて、0x18005a400+18b0+0x228に
データを格納していること、およびFUN_180019018にて
復号処理をしていることを確認

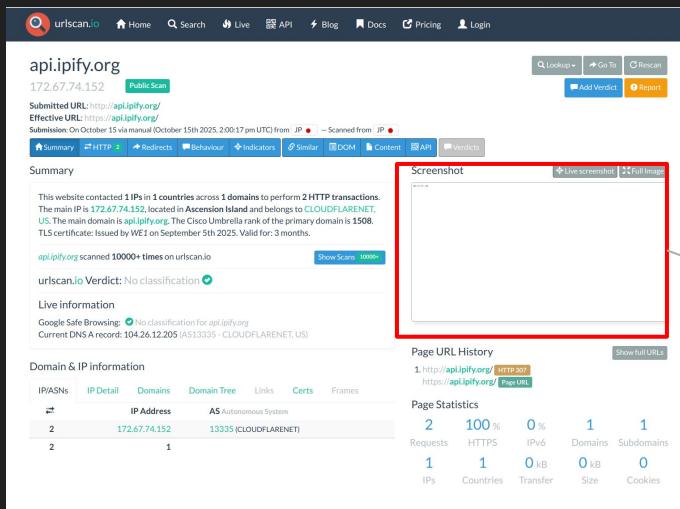
```

1 void FUN_180019018(longlong param_1,undefined8 *param_2)
2
3
4 {
5     ushort *puVar1;
6
7     puVar1 = (ushort *)(&param_1 + 0x268);
8     if (7 < *(ulonglong *)(&param_1 + 0x280)) {
9         puVar1 = *(ushort **)puVar1;
10    }
11    FUN_180014390(param_1,puVar1);
12    puVar1 = (ushort *)(&param_1 + 0x288);
13    if (7 < *(ulonglong *)(&param_1 + 0x2a0)) {
14        puVar1 = *(ushort **)puVar1;
15    }
16    FUN_180014390(param_1,puVar1);
17    puVar1 = (ushort *)(&param_1 + 0x2a8);
18    if (7 < *(ulonglong *)(&param_1 + 0x2c0)) {
19        puVar1 = *(ushort **)puVar1;
20    }
21    FUN_180014390(param_1,puVar1);
22    puVar1 = (ushort *)(&param_1 + 0x228); FUN_180014390にて xor2-1で復号
23    if (7 < *(ulonglong *)(&param_1 + 0x240)) {
24        puVar1 = *(ushort **)puVar1;
25    }
26    FUN_180014390(param_1,puVar1);
27    puVar1 = (ushort *)(&param_1 + 0x2c8);
28    if (7 < *(ulonglong *)(&param_1 + 0x2e0)) {
29        puVar1 = *(ushort **)puVar1;
30    }

```

2-2. Config読解

- 復号により得られる文字列”api.ipify.org”とは、アクセス元のグローバルIPアドレスを表示するサービス
- よって答えは「g. クライアントのグローバルIPアドレス」



urlscan.io Home Search Live API Blog Docs Pricing Login

api.ipify.org
172.67.74.152 Public Scan

Submitted URL: http://api.ipify.org/
Effective URL: https://api.ipify.org/
Submission: On October 15 via manual (October 15th 2025, 2:00:17 pm UTC) from JP - Scanned from JP

Summary: This website contacted 1 IPs in 1 countries across 1 domains to perform 2 HTTP transactions. The main IP is 172.67.74.152, located in Ascension Island and belongs to CLOUDFLARENET. US. The main domain is api.ipify.org. The Cisco Umbrella rank of the primary domain is 1508. TLS certificate: Issued by WET on September 5th 2025. Valid for: 3 months.

Live information: Google Safe Browsing: No classification for api.ipify.org. Current DNS A record: 104.26.12.205 (AS13335 - CLOUDFLAREN.NET, US).

Domain & IP information: IPASN: 172.67.74.152, IP Address: 172.67.74.152, AS: Autonomous System, IP Detail: 172.67.74.152, Domains: 1, Domain Tree: 1, Links: 1, Certs: 1, Frames: 1.

Screenshot: A screenshot of the ipify API response showing the IP address 146.70.201.224.

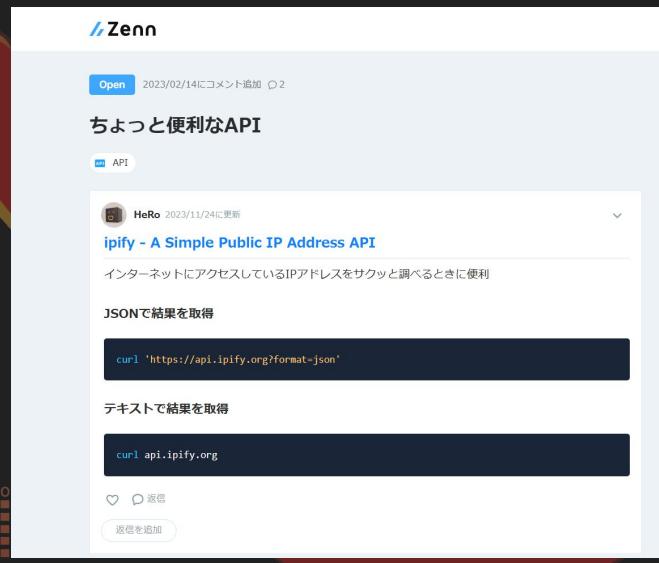
Page URL History: 1. http://api.ipify.org/ (HTTP 200), 2. https://api.ipify.org/ (Page 1/1)

Page Statistics: Requests: 2, HTTPS: 100%, IPv6: 0%, Domains: 1, Subdomains: 1. IPs: 1, Countries: 1, Transfer: 0 kB, Size: 0 kB, Cookies: 0.

urlscan確認結果

MWS Cup 2025

引用元:https://urlscan.io/



Zenn Open 2023/02/14にコメント追加

ちょっと便利なAPI

API

HeRo 2023/11/24に更新

ipify - A Simple Public IP Address API

インターネットにアクセスしているIPアドレスをサクッと調べるときに便利

JSONで結果を取得

```
curl "https://api.ipify.org?format=json"
```

テキストで結果を取得

```
curl api.ipify.org
```

25

LLMによるトライアル回答

- LLMによる回答で、Config読み解(2-1/2-2)とも解かれる。(他も結構解かれる。)



Result

	gpt-5-codex high	Claude Sonet- 4.5	Gemini-2.5- Pro	Grok-4- Fast
1-1	○	○	✗	✗
1-2	○	○	○	○
1-3	○	○	✗	○
1-4	✗	✗	✗	✗
1-5	✗	○	○	✗
2-1	○	○	✗	✗
2-2	○	✗	✗	✗
3-1	○	○	○	○
3-2	○	○	✗	✗
4-1	○	○	✗	✗
4-2	○	✗	✗	✗

LLMによるトライアル回答

- Claudeの2-1の正答は勘違いによるもの。本当の意味で解けているものではなかった。

解析結果

0x18001c9d4でsscanf_sを使用して9個の情報を結合する処理を解析しました。

2番目の情報の特定プロセス:

1. FUN_18001c6fc内のsscanf_s呼び出し

- 9個の情報が順番に処理されます

- 2番目の情報は `param_1 + 0x220` の内容

2. `param_1 + 0x220` の設定元: FUN_180017b98

- グローバル変数 `DAT_180059050` から値を読み取り
- FUN_180014390でデコード処理

3. DAT_180059050の初期化: FUN_180001050

c

```
FUN_180002a00(&DAT_180059050, 0x20, 0,  
"tvdqhg....TYHMH#####");
```

4. 文字列のデコード

- デコードアルゴリズム: `(*char XOR 2) - 1`
- "tvdqhg" → "userid"
- "TYHMH" → "UZINI"

5. 結論

- 2番目の情報は "userid" を含む文字列
- これはクライアントのユーザ名を表します

答え:f. クライアントのユーザ名

LLMによるトライアル回答

- Codexの2-1の正答は完全に正解(GetUserNameが使用されていることまで特定。)

- Analyzing suspected registry and WMI operations ^

I'm trying to decode a function interacting with the registry and WMI by inspecting function pointers and offsets, suspecting calls to Windows API like GetComputerNameW; I'm exploring how fields at offsets like +0x200 and +0x240 relate to stored data such as install language or computer name by tracing related functions and pointers.

- Analyzing aggregator's called functions

- Analyzing function pointers and data structures ^

I'm exploring how a function uses pointers at specific offsets, likely referencing Windows API calls like GetComputerNameW or GetUserNameW. To confirm which function is called, I'll investigate the data copied into these structures and correlate with known option lists, examining related functions that fill these data fields. This might help distinguish which API corresponds to what data offset.

- Analyzing function pointer usage

- Investigating function pointer structure ^

I'm tracing how a function pointer at an offset is set up within a structure, aiming to identify the API it calls. I'm planning to find initializing code that loads those function pointers, possibly from system DLLs, by looking for functions writing into related structure offsets and analyzing call sites to narrow down the data sources.

LLMによるトライアル回答

- Codexの2-2の正答は少し怪しい(HTTPで取得したデータが外部IPと決め打ち)が、大筋正解

Finished working

- Analyzing argument order in assembly call

- Tracing pointer origin for third argument ^

I'm digging into how the third string argument, passed via RAX and stored on the stack, actually points back to `param_3`—a pointer to some string structure. I want to understand what data `param_3` holds at the call and how it fits into the overall concatenation before `sprintf_s`. I'll look for function call references next to get clues about `param_3`'s real meaning.

- Analyzing parameter roles by call inspection

- Checking string at `FUN_180018734`

- Mapping data retrieval via WinINet and WinHTTP ^

I'm analyzing functions accessing structured fields, trying to connect stored strings to computer name, user name, or OS details. I'm focusing on `FUN_180018734` which seems to open a WinHTTP session using `WinHttpOpen` with a "Mozilla/5.0" user-agent, possibly to fetch the global IP or related network info. I want to piece together how these COM calls and offsets in the structure relate to data gathered for C2 communication.

- Analyzing HTTP request and response handling ^

I'm piecing together how the code constructs HTTP requests using wide strings like "GET" and "POST," then sends these requests using Windows HTTP APIs. It looks like the function fetches an HTTP response string—probably the external IP address from a remote service—which then gets stored as the third parameter passed to the command-and-control message. Next, I want to confirm what the first and fourth parameters represent by tracing their data origins, especially the fourth parameter which might relate to network interface info.

- Analyzing likely third info value

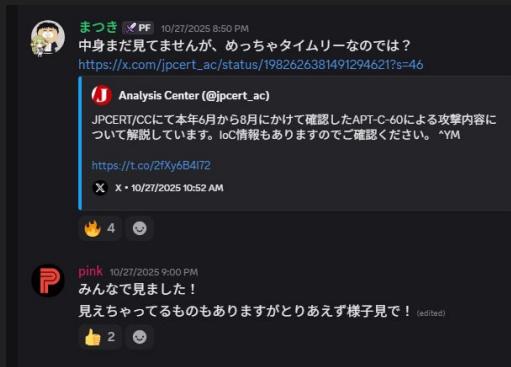
軌道修正

- 試行錯誤の結果、今回LLMを許容していることもあり、回数制限は厳しめにし、LLMの言うことを鵜呑みにするのではなく、LLMが導き出した答えの検証を含めてしっかりと回答してもらう方向で修正



最後まで気を抜けない

- 前日(10/27)にJPCERT/CCよりRATの暗号解読スクリプトが公表。
- ちょうど通信データの復号問題でも活用可能。



改変されたRC4はKSAのサイクルを増やす点やXORする値に加算を行うなどの点が通常のRC4とは異なり、次に示すPythonスクリプトでデコードすることが可能です。

```
import base64

def CustomRC4(key: bytes, data: bytes) -> bytes:
    # --- KSA ---
    S = list(range(256))
    n = 3
    for round in range(n):
        j = 0
        keylen = len(key)
        if keylen == 0:
            raise ValueError("key must be non-empty")
        for i in range(256):
            j = (j + S[i] + key[i % keylen]) & 0xFF
            S[i], S[j] = S[j], S[i]

    # --- PRGA ---
    i = j = 0
    out = []
    for b in data:
        i = (i + 1) & 0xFF
        j = (j + S[i]) & 0xFF
        k = S[S[i] + j] & 0xFF
        S[i], S[j] = S[j], S[i]
        k2 = S[(S[(i >> 3) ^ (0x20 * i)] & 0xFF) + S[((0x20 * i) ^ (j >> 3)) & 0xFF]] & 0xAA & 0xFF] + S[(S[j] + S[i]) & 0xFF]
        out.append((b ^ k ^ k2) & 0xFF)

    return bytes(out)

def decode(base64in):
    key = b"90b149c69b149c4b99c04d1dc9b940b9"
    decoded = CustomRC4(key, base64.b64decode(base64in))
    print("Result: ", decoded)
```

最後に

- 個人的に作問によって非常に多くのことを学べました(API難読化手法の理解、動的解析、C2サーバ模擬など)
- 作問メンバの皆様の努力・協力のもと、無事MWSCupを終えられたことを素晴らしいこと思います。
- この問題を真摯にチャレンジ・回答頂いた競技者の皆様に感謝の意を示します。
- 最後にこのプレゼンを見て作間に興味を持たれた方は、ぜひSlackなどでご連絡ください。