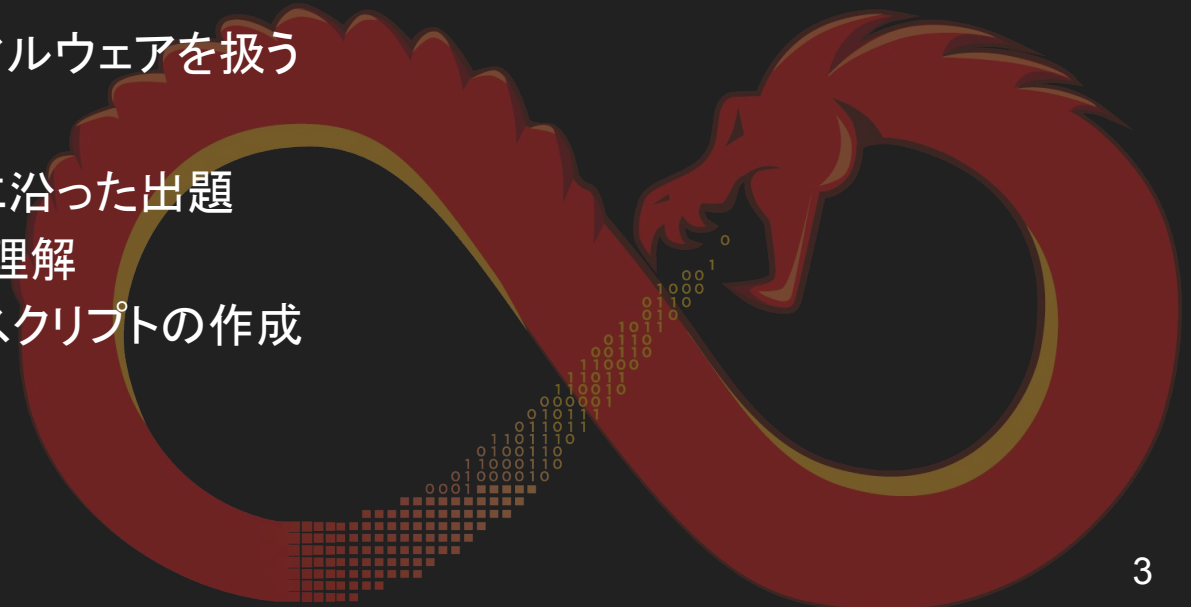
MWS Cup 2024
Static Analysis 解説

2024の問題担当

- Static Analysis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)
  - 皆川 諒 (株式会社エヌ・エフ・ラボラトリーズ)
  - 末廣 繁樹 (株式会社エヌ・エフ・ラボラトリーズ)
  - 光安 正憲 (西日本電信電話株式会社)
  - 清水 嶺 (西日本電信電話株式会社)
  - 緒方 湧己 (西日本電信電話株式会社)
  - 仲川 宜秀 (西日本電信電話株式会社)

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

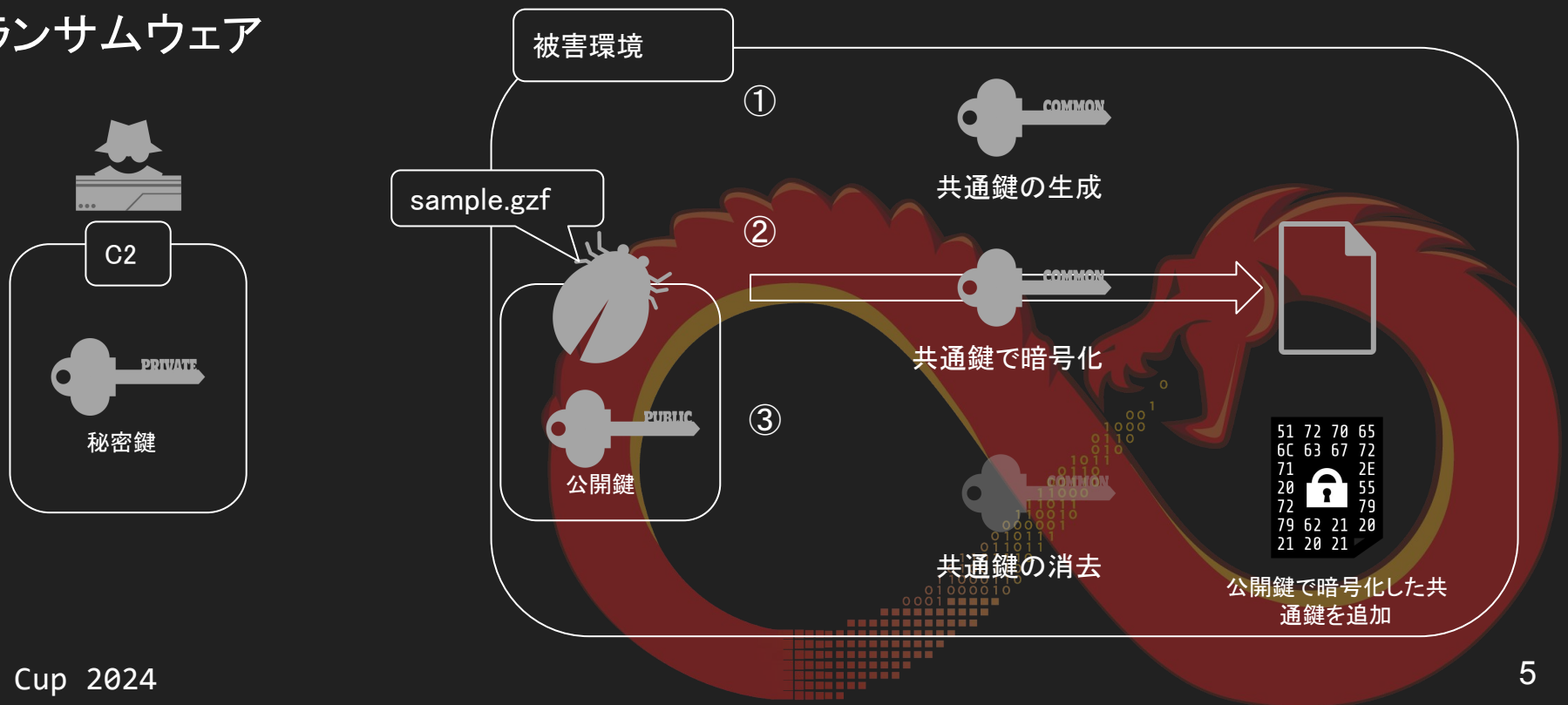


ポイント

- 積極的に変数名や型を変更する
  - 名前を付けて読みやすくしていく
  - デフォルトでは型情報がないことが多い
- デコンパイラを信用しすぎない
  - アセンブリを確認して整合性を確認する
  - 手動で修正する
- 順番に回答する必要はないので解けそうな問題から解く

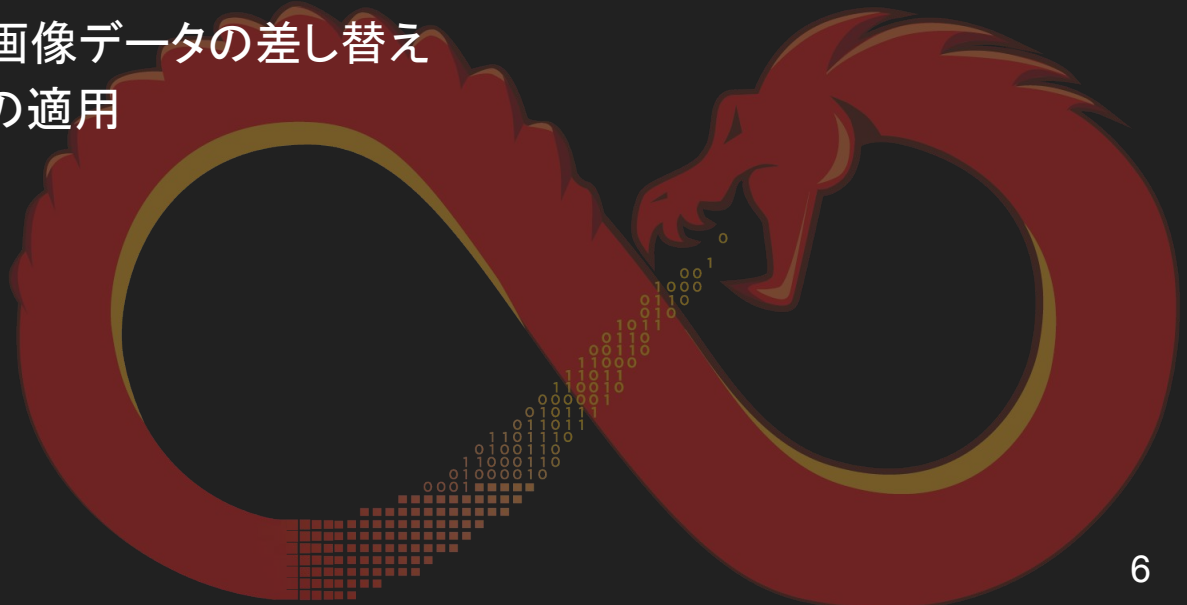
マルウェアの動作概要

ランサムウェア



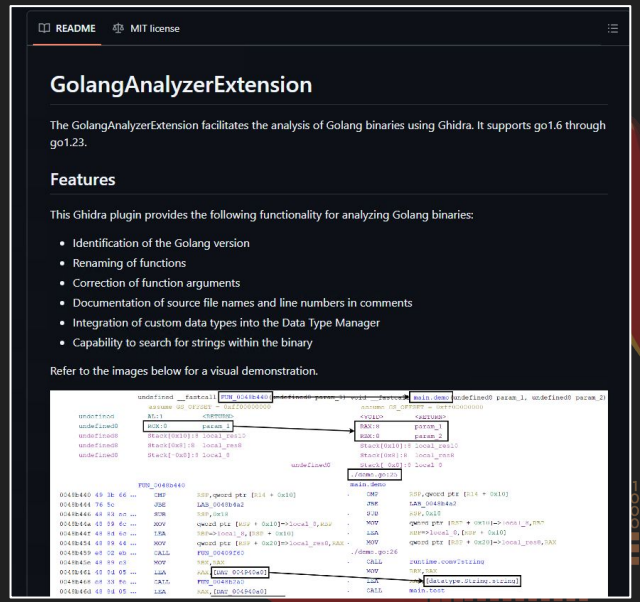
加工済みのGZF

- MWSの問題作成にあたって以下の変更を加えています
  - GolangAnalyzerExtensionのAnalyze結果の適用
  - 出題に伴う関数名と関連するメタ情報の修正
  - コード中で参照される画像データの差し替え
  - 補助のための構造体の適用



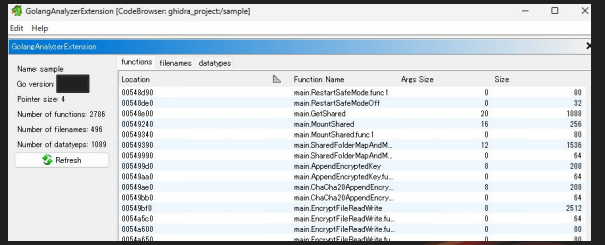
GolangAnalyzerExtension

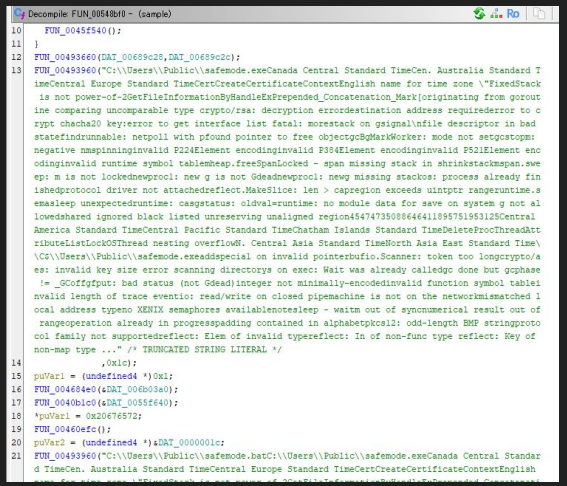
- 問題作成委員FFRIの桑原さんが開発したGo言語バイナリ解析用のGhidra Extension

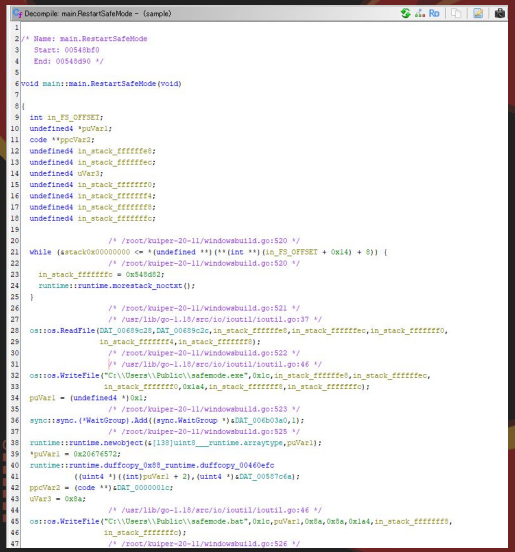


GitHub - mooncat-greenpy/Ghidra_GolangAnalyzerExtension: Analyze Golang with Ghidra

GolangAnalyzerExtension

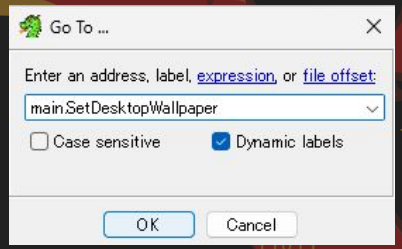






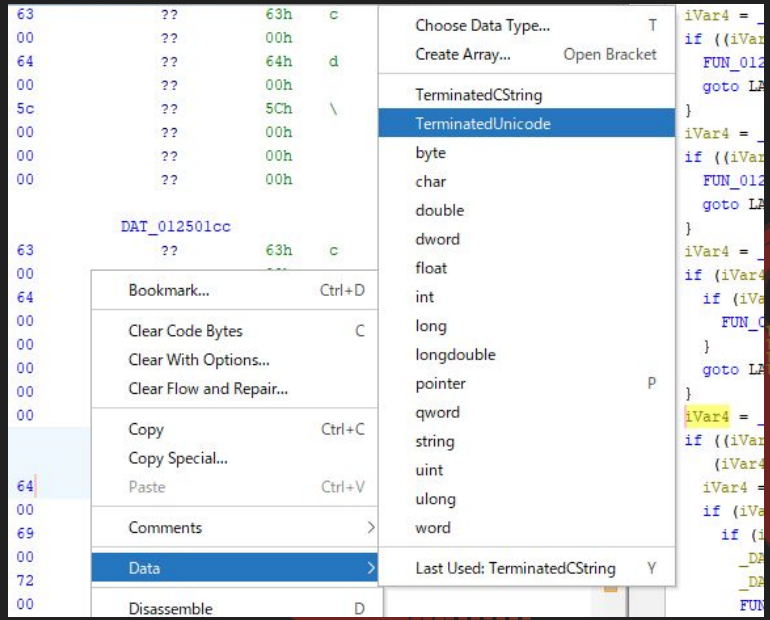
Ghidra Tips: Go To

- Gキーで開くGo Toダイアログにアドレスやlabelを入力
  - 任意の場所に簡単に移動
  - 問題ではアドレスや関数名が指定されていることが多いので必須!



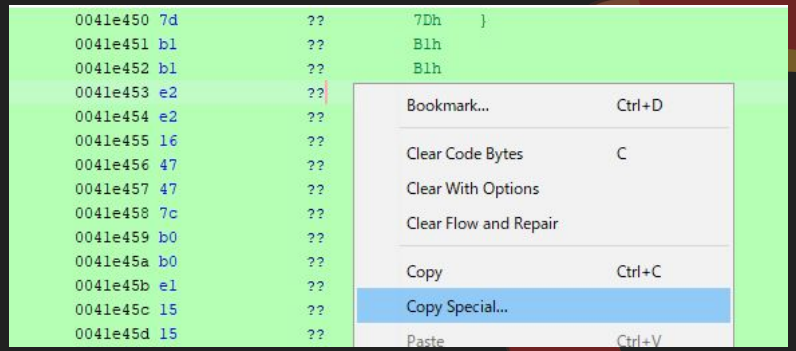
Ghidra Tips: データの定義

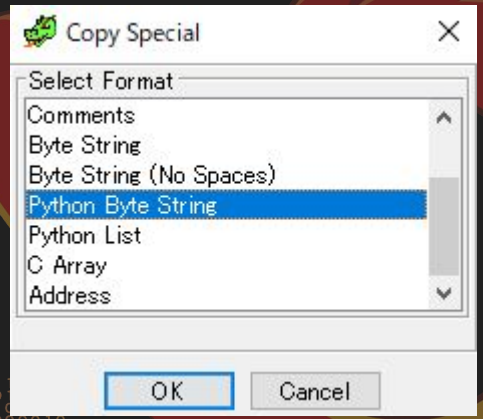
- 定義したいデータの先頭を右クリック
  - コンテキストメニューからData -> 定義する形式を選択



Ghidra Tips: データのコピー1

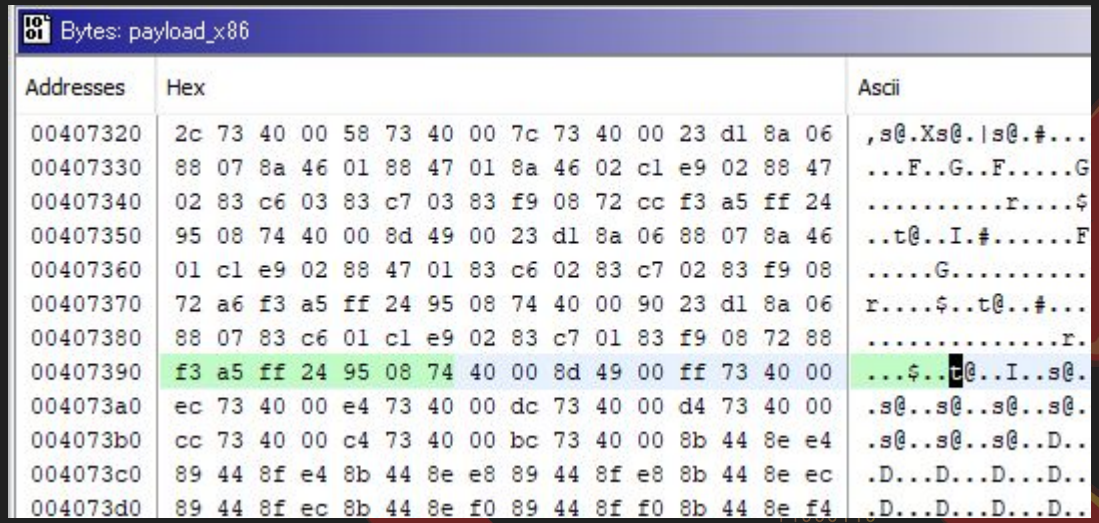
- コピーしたい範囲を選択
  - コンテキストメニューからCopy Special =>
    好きなフォーマットを選択





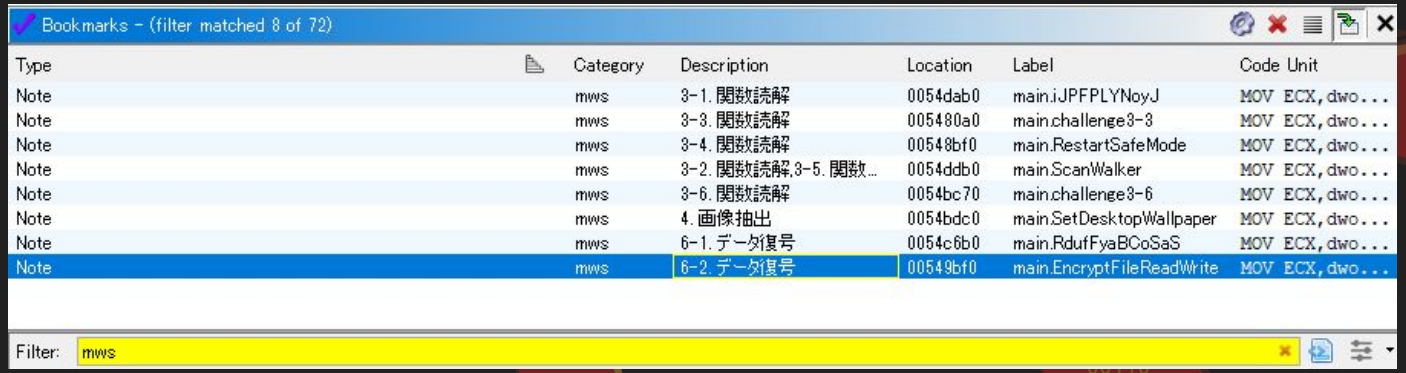
Ghidra Tips: データのコピー2

- Bytesウィンドウでコピーする



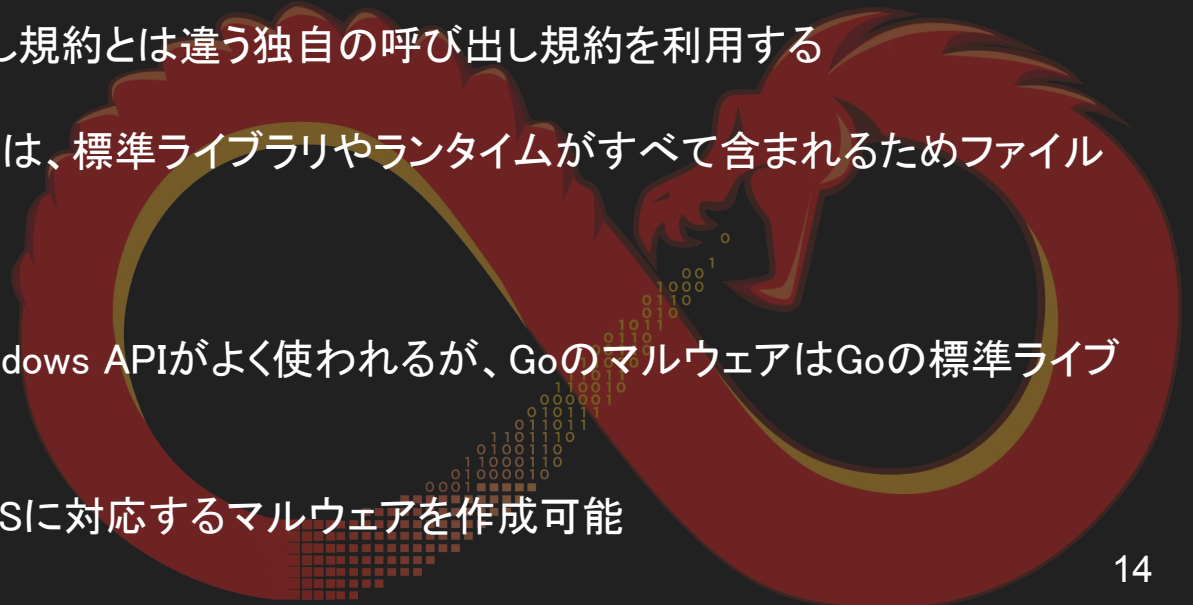
Bookmarks

問題に関連するアドレスをBookmarkとして登録済み



ヒント: Go言語のマルウェアの特徴

- 特有のデータ構造
  - Go固有の文字列フォーマットや関数名の関数情報の構造を持つ
  - ツールで対応可能
- 独自の呼び出し規約
  - C++で利用される呼び出し規約とは違う独自の呼び出し規約を利用する
- スタティックリンク
  - Go言語の実行ファイルには、標準ライブラリやランタイムがすべて含まれるためファイルサイズが大きい
  - 必要なコードだけを読む
- 標準ライブラリの使用
  - C++のマルウェアではWindows APIがよく使われるが、Goのマルウェアは Goの標準ライブラリを利用する
- クロスプラットフォーム対応
  - 1つのコードから複数のOSに対応するマルウェアを作成可能



ヒント: Go言語特有のデータ構造

Go言語特有のデータ構造をいくつか紹介する

- 文字列
  - 文字列へのポインタ、文字列長で構成される

```
struct string {
    char* str;
    int len;
}
```

Go言語で以下のように記載されている

```
var testStr string = "test"

func testString(s string) *string {
...
}
```

- スライス（可変長の配列のようなもの）
  - 配列へのポインタ、配列の長さ、メモリ確保された配列の長さ

```
struct slice {
    void* array;
    int len;
    int cap;
}
```

Go言語で以下の第1引数のように記載されている

```
func testSlice(s []int) {
…
}
```

ヒント: Go言語独自の呼び出し規約

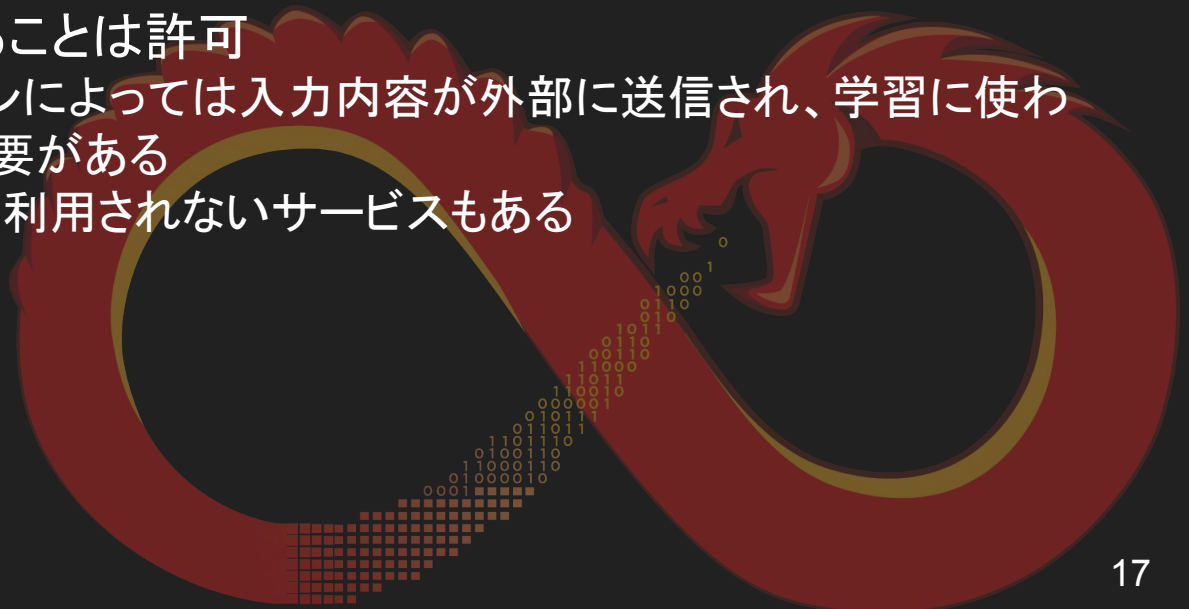
Go言語独自の呼び出し規約について紹介する

- 以下の図に関数呼び出し時の引数・戻り値が格納される場所を載せる
  - Go言語のバージョンやアーキテクチャにより変化
  - 例外もあるため注意すること

	64bit	32bit
Go 1.17以上	引数: rax, rbx, rcx, rdi, rsi, r8, r9, r10, r11, スタックの先頭から 戻り値: 引数と同じ（raxから利用）	左下と同じ
Go 1.17より前	引数: スタックの先頭から 戻り値: 引数と同じ（引数で利用してないもののみ利用）	左と同じ

ヒント: GoogleやAIの活用

- ファイルを直接VirusTotalなどの外部サービスにアップロードすることは禁止
- ただし、コードの断片や文字列をGoogle検索、Chat GPTなどのGenerative AIのサービスに投稿することは許可
  - 実務で扱う際は、プランによっては入力内容が外部に送信され、学習に使われることを考慮する必要がある
  - 有料プランでは学習に利用されないサービスもある



問題一覧

1-1. Goメタ情報 1	1-2. Goメタ情報 1	2-1. オプション 1	4. 画像抽出 1
2-2. オプション 2	3-1. 関数読解 2	3-2. 関数読解 2	3-3. 関数読解 2
3-4. 関数読解 2	3-5. 関数読解 2	3-6. 関数読解 2	5. 暗号鍵 2
6-1. データ復号 2	6-2. データ復号 3		

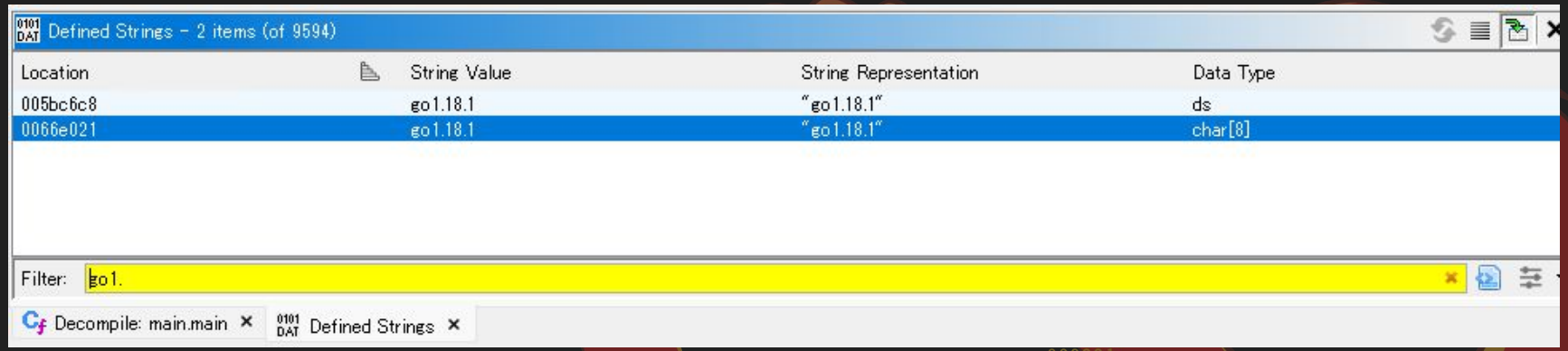
競技中はここまで

1-1. Goメタ情報


1-1. Goメタ情報
1
Static Analysis はマルウェアの静的解析に関する問題です。
問題ファイル sample.gzf をダウンロードし、Ghidra で解析して以降の問いに回答してください。
このマルウェアは、Go言語で開発されている。開発されたバージョンを解答せよ。 バージョンが1.23.0の場合は go1.23.0 のフォーマットで提出すること。
sample.gzf
Flag
Submit

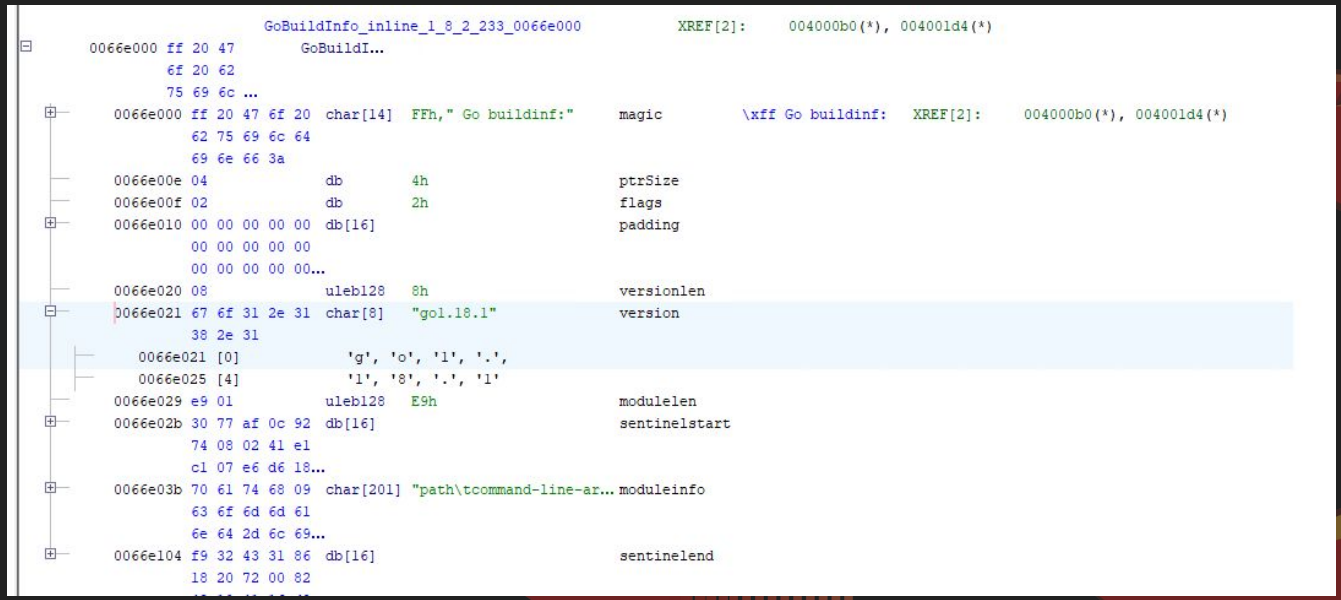
1-1. Goメタ情報

- Defined Stringsを探すとGoのバージョン情報らしき文字列が見つかる



1-1. Goメタ情報

- Go言語のプログラムはメタ情報としてバージョン情報を持っている



1-1. Goメタ情報

The Answer is: go1.18.1

1-2. Goメタ情報


1-2. Goメタ情報
1
メイン関数の開始アドレスを16進数の8桁数字で答えよ。例えば、アドレスが0x0100abcdの場合は0100abcdと回答すること。
Flag
Submit

1-2. Goメタ情報

- Go言語のプログラムのメイン関数はmain.mainのシンボル情報を持つ

```
                    ****************************************************************
                    * Name: main.main                                              *
                    * Start: 005511b0                                              *
                    * End: 00551260                                                *
                    ****************************************************************
                    void abi0 main.main(void)
    void              <VOID>          <RETURN>
    undefined4        Stack[-0x4]:4   local_4                          XREF[2]:     005511d2(W),
                                                                                    005511e1(W)
    undefined4        Stack[-0x8]:4   local_8                          XREF[2]:     005511ce(W),
                                                                                    005511f0(W)
    undefined1        Stack[-0x9]:1   local_9                          XREF[5]:     005511d6(W),
                                                                                    005511e5(W),
                                                                                    005511f4(W),
                                                                                    0055122b(W),
                                                                                    00551235(W)
    undefined4        Stack[-0x10]:4 local_10                          XREF[1]:     00551202(W)
    undefined4        Stack[-0x14]:4 local_14                          XREF[3]:     005511ff(*),
                                                                                    00551215(*),
                                                                                    00551223(*)
                    /root/kuiper-20-11/windowsbuild.go:1793
                    main::main.main                                  XREF[5]:     runtime.main:00436eaf(c),
                                                                                  00551251(j), 005905e4(*),
                                                                                  0063dee0(*), 0066d1e4(*)
    005511b0 64 8b 0d       MOV         ECX,dword ptr FS:[0x14]
             14 00 00 00
    005511b7 8b 89 00       MOV         ECX,dword ptr [ECX]
             00 00 00
```

1-2. Goメタ情報

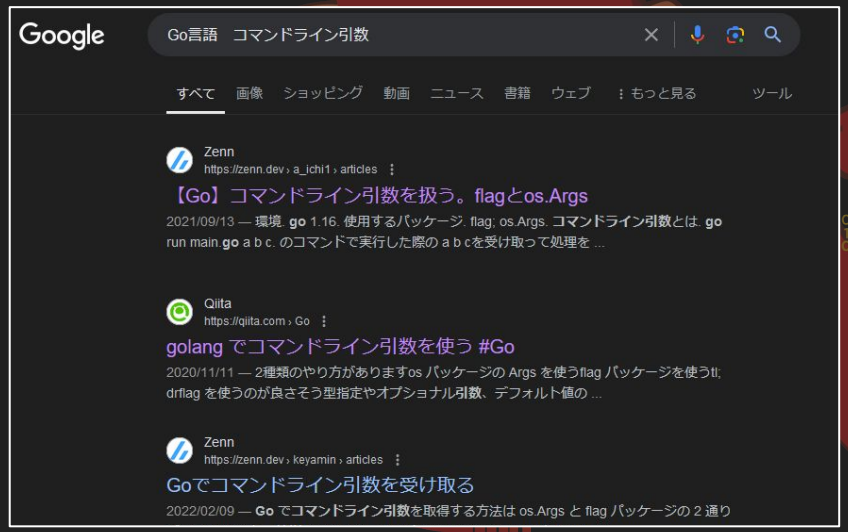
The Answer is: 005511b0

2-1. オプション


2-1. オプション
1
このマルウェアは、いくつかのコマンドライン引数が設定されている。マルウェアの開発者はこの引数についてヘルプを実装している。この機能を実装するのに利用しているパッケージ名を答えよ。
/usr/lib/go-1.23/src/math/abs.go を使っている場合は math というフォーマットで回答すること。
Flag
Submit

2-1. オプション

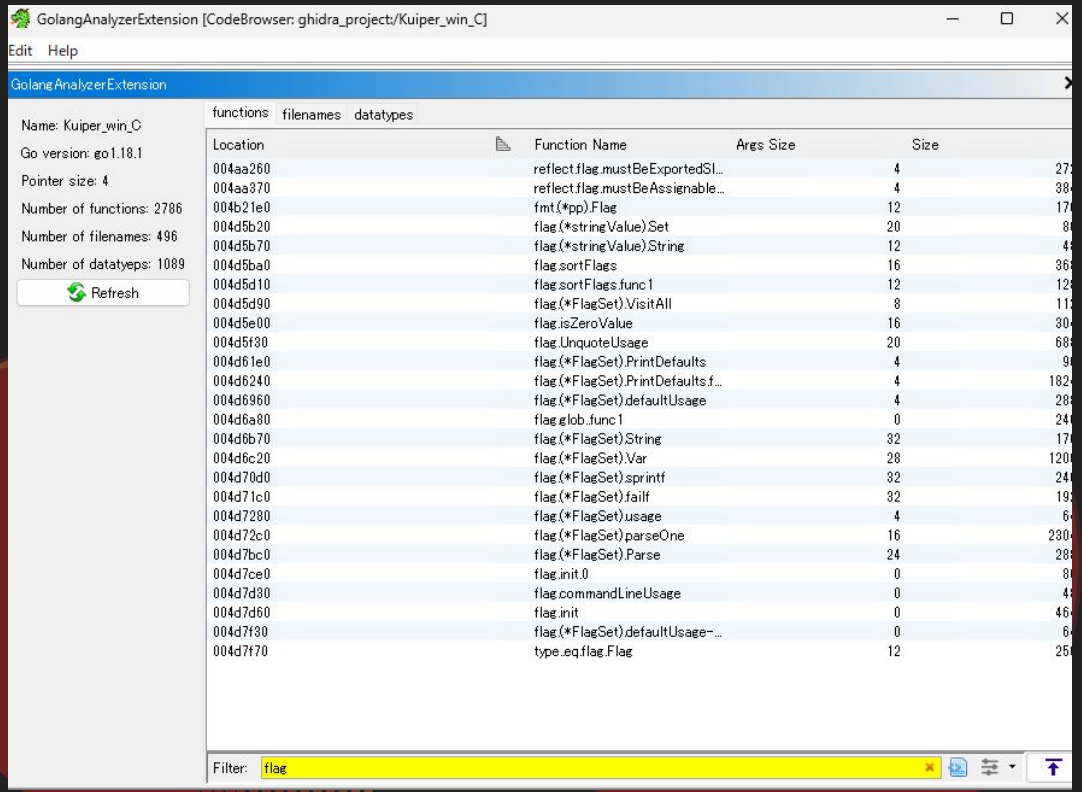
- Go言語 コマンドライン引数などでぐぐってみる
- 2パターンの実装があることがわかる
  - os パッケージの Args を使う方法
  - flag パッケージを使う方法



2-1. オプション

- Ghidraで調べる
  - flagが使われている

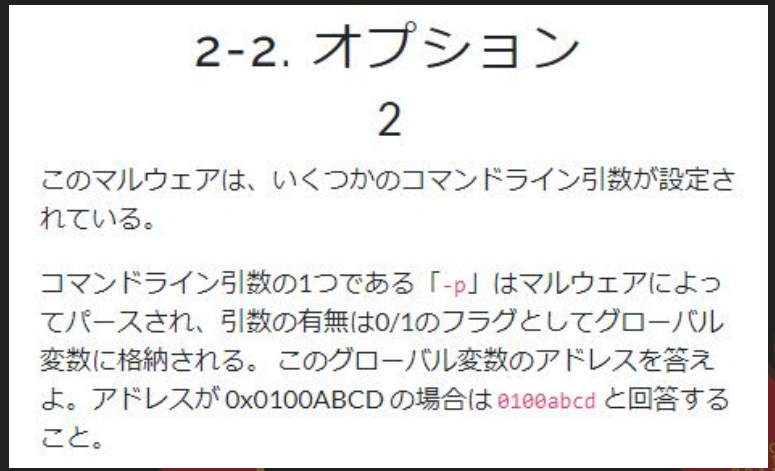
```
                    /usr/lib/go-1.18/src/flag/flag.go:773
                    flag::flag.(*FlagSet).String                XREF[14]:
004d6b70 64 8b 0d       MOV         ECX,dword ptr FS:[0x14]
         14 00 00 00
004d6b77 8b 89 00       MOV         ECX,dword ptr [ECX]
         00 00 00
004d6b7d 3b 61 08       CMP         ESP,dword ptr [ECX + 0x8]
```



2-1. オプション

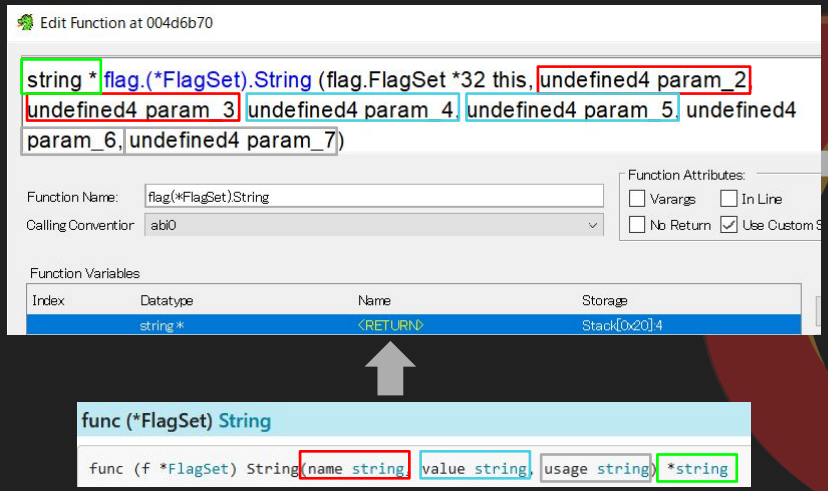
The Answer is: flag

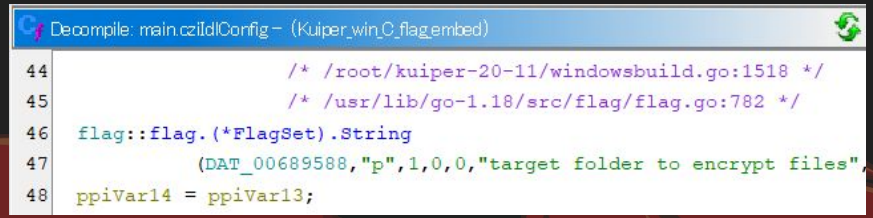
2–2. オプション

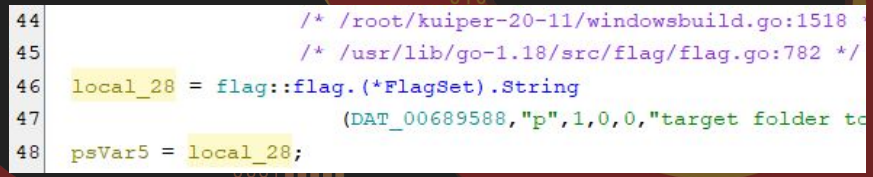

2-2. オプション
2
このマルウェアは、いくつかのコマンドライン引数が設定されている。
コマンドライン引数の1つである「-p」はマルウェアによってパースされ、引数の有無は0/1のフラグとしてグローバル変数に格納される。 このグローバル変数のアドレスを答えよ。アドレスが 0x0100ABCD の場合は 0100abcd と回答すること。

2-2. オプション

- flag.(*FlagSet).Stringの呼び出し元を辿ると以下が見つかる
  - 第2引数「"p"」がある
- 関数の型を修正する
  - 戻り値を追加







https://pkg.go.dev/flag#FlagSet.String

2-2. オプション

- psVar5を辿ると文字列を比較している処理がある
- psVar5の長さが0でないなら0066e1c0に0が格納される
  - 初期値は1

```
188                     /* /root/kuiper-20-11/windowsbuild.go:1560 */
189   iVar2 = psVar5->len;
190   if (iVar2 != 0) {
191                     /* /root/kuiper-20-11/windowsbuild.go:1561 */
192     puVar12 = psVar5->str;
193     gostr_..len = iVar2;
194     if (DAT_006b0450 != 0) {
195       runtime::runtime.gcWriteBarrier();
196       puVar12 = gostr_..str;
197     }
198     gostr_..str = puVar12;
199                     /* /root/kuiper-20-11/windowsbuild.go:1562 */
200     DAT_0066e1c0 = 0;
201   }
```

```
                    DAT_0066e1c0
0066e1c0 01             undefine...  01h
```

2-2. オプション

The Answer is: 0066e1c0

3-1. 関数読解

3-1. 関数読解

2

関数 `main.iJPFPLYNoyJ` の動作を確認し、アドレス `0x00685318` に格納されている `3082020～` で始まる文字列の使用用途、及びどのようなデータかについて最も適している記述を下記から選択してください。

- ○ ファイルを暗号化したあとの脅迫文の文章を16進数文字エンコードしたもの
- ○ ファイルの暗号化するための秘密鍵を16進数文字エンコードしたもの
- ○ ファイルを暗号化するための公開鍵を16進数文字エンコードしたもの
- ○ プロセスを停止させるためのPowershellコマンドを16進数文字エンコードしたもの
- ○ システム情報を取得するためのPowershellコマンドを16進数文字エンコードしたもの
- ○ 暗号化鍵、IVを暗号化するための秘密鍵を16進数文字エンコードしたもの
- ○ 暗号化鍵、IVを暗号化するための公開鍵を16進数文字エンコードしたもの

3-1. 関数読解

3082020～で始まる文字列の追跡は下記の通り。

- 関数main.IaVwUafTBkIOtで処理
- その結果を関数main.yjvmRItNyCuXEAQBwPで処理
- その結果をもとに、引数1，2を使用して、crypto/rsa.EncryptOAEPで処理

crypto/rsa.EncryptOAEP関数のソースから、3082020～で始まる文字列が**公開鍵**、引数1，2がRSA-OAEPで暗号化対象のデータとわかる

https://github.com/golang/go/blob/master/src/crypto/rsa/rsa.go

func EncryptOAEP(hash hash.Hash, random io.Reader, pub *PublicKey, msg []byte, label []byte)

公開鍵

暗号化対象メッセージ

関数main.iJPFPLYNoyJ

```
void __thiscall
main::main.iJPFPLYNoyJ
          (undefined4 param_1,undefined4 param_2,undefined4 param_3,undefined4 param_4,
           undefined4 param_5,undefined4 param_6,undefined4 param_7)

{
  int in_FS_OFFSET;
  undefined4 in_stack_fffffa0;
  int in_stack_fffffa4;
  int iVar1;
  undefined4 in_stack_fffffa8;
  undefined4 in_stack_fffffac;
  undefined4 in_stack_fffffc4;
  undefined4 in_stack_fffffc8;
  undefined4 in_stack_fffffcc;
  int in_stack_fffffd0;
  undefined4 in_stack_fffffd4;
  undefined local_28 [32];
  undefined4 local_8;
  int local_4;

  while (&stack0x00000000 <= *(undefined **)(**(int **)(in_FS_OFFSET + 0x14) + 8)) {
    local_4 = 0x54dc48;
    runtime::runtime.morestack_noctxt();
  }
  main.IaVwUafTBkIOt(gostr_3082020a0282020100cb797a0f776f60.str,
                     gostr_3082020a0282020100cb797a0f776f60.len,in_stack_fffffa0,in_stack_fffffa4)
  ;
  iVar1 = in_stack_fffffa4;
  main.yjvmRItNyCuXEAQBwP
            (in_stack_fffffa0,in_stack_fffffa4,in_stack_fffffa0,in_stack_fffffa4,
             in_stack_fffffa8);
  if (iVar1 == 0) {
    local_8 = in_stack_fffffa0;
    runtime::runtime.newobject
              (&crypto/sha256::crypto/sha256.digest___runtime.structtype,in_stack_fffffa4);
    local_4 = in_stack_fffffa4;
    crypto/sha256::crypto/sha256.(*digest).Reset(in_stack_fffffa4);
    runtime::runtime.stringtoslicebyte
              (local_28,param_1,param_2,iVar1,in_stack_fffffa8,in_stack_fffffac);
    crypto/rsa::crypto/rsa.EncryptOAEP
              (&*sha256.digest___implements___hash.Hash___runtime.itab,local_4,DAT_00689a80,
               DAT_00689a84,local_8,iVar1,in_stack_fffffa8,in_stack_fffffac,0,0,0,
               in_stack_fffffc4,in_stack_fffffc8,in_stack_fffffcc,in_stack_fffffd0,
               in_stack_fffffd4);
    if (in_stack_fffffd0 != 0) {
      return;
    }
    return;
  }
  return;
}
```

3-1. 関数読解

関数main.IaVwUafTBkIOt
⇒不要文字の除去（実質なし）

関数main.yjvmRItNyCuXEAQBwP
⇒**16進数文字のデコード**、及び公開鍵の抽出

main.IaVwUafTBkIOt

```
22  while (&stack0x00000000 <= *(undefined **)(**(int **)(in_FS_OFFSET + 0x14) + 8)) {
23    runtime::runtime.morestack_noctxt();
24  }
25  rVar2 = strings::strings.genSplit(str.str,str.len,0,0,0,0xffffffff);
26  strLen = rVar2.len;
27  strSlice = rVar2.array;
28  certStringSlice = strSlice;
29  strSlice = (void *)0x0;
30  tmpStr = (uint8 *)0x0;
31  for (idx = 0; idx < strLen; idx = idx + 1) {
32    ppcVar1 = (char **)((int)certStringSlice + idx * 8);
33    charCertString = *ppcVar1;
34    charCertStringLen = ppcVar1[1];
35    if ((charCertStringLen != &DAT_00000001) ||
36       (((*charCertString != '\n' && (*charCertString != ' ')) && (*charCertString != '\r')))) {
37      ret = runtime::runtime.concatstring2(0,tmpStr,strSlice,charCertString,charCertStringLen);
38      strSlice = (void *)ret.len;
39      tmpStr = ret.str;
40    }
41  }
42  ret.len = (int)strSlice;
43  ret.str = tmpStr;
44  return ret;
```

関数main.yjvmRItNyCuXEAQBwP

```
27  runtime::runtime.stringtoslicebyte
28            (0,param_1,param_2,in_stack_ffffffe0,in_stack_ffffffe4,in_stack_fffffe8);
29  uVar1 = in_stack_ffffffe0;
30  uVar2 = in_stack_ffffffe0;
31  iVar3 = in_stack_ffffffe4;
32  uVar4 = in_stack_ffffffe8;
33  encoding/hex::encoding/hex.Decode
34            (in_stack_ffffffe0,in_stack_ffffffe4,in_stack_ffffffe8,in_stack_ffffffe0,
35             in_stack_ffffffe4,in_stack_ffffffe8,in_stack_fffffffec,in_stack_fffffff0,
36             in_stack_fffffff4);
37  if (in_stack_fffffffec <= in_stack_ffffffe8) {
38    crypto/x509::crypto/x509.ParsePKCS1PublicKey
39              (in_stack_ffffffe0,in_stack_fffffffec,in_stack_ffffffe8,uVar2,iVar3,uVar4);
40    if (iVar3 != 0) {
41      return;
42    }
43    return;
44  }
```

AIに聞いてみる

この関数は、文字列の分割やフィルタリング、そして文字列の再結合に関連する処理を行っているようです。以下は、各部分を解析した説明です。

---

1. 概要

関数 `IaVwUafTBkIOt` は、入力されたデータを解析・分割し、一部の不要な文字（例えば改行や空白）を除去したうえで、文字列を結合しています。これにより、入力された文字列データを整形したり、必要な情報を取り出して再構築することを目的としているようです。

関数 `main::main.yjvmRItNyCuXEAQBwP` の解説

関数の目的

この関数は、**PKCS#1形式の公開鍵をデコードし、解析する**ことを目的としています。

関数の処理手順

1. **入力の検証:** スタックポインタの範囲をチェックし、不正なアクセスを防ぐための安全策が施されています。
2. **文字列からバイト列への変換:** 入力された文字列（PKCS#1公開鍵の16進数表現）を、バイト列に変換します。
3. **16進数デコード:** 変換されたバイト列を、16進数から2進数へデコードします。
4. **PKCS#1公開鍵の解析:** デコードされたバイト列をPKCS#1形式の公開鍵として解析します。解析に失敗すると、関数は終了します。

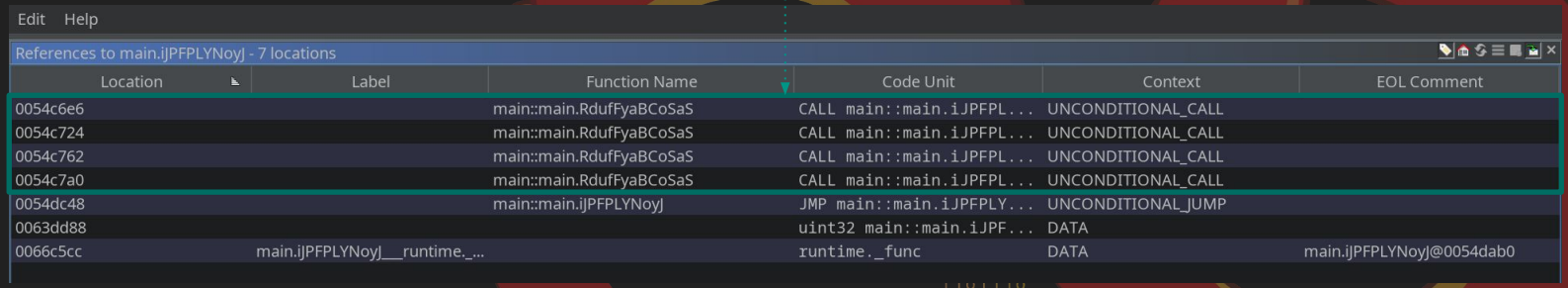
各関数について

- `runtime::runtime.stringtoslicebyte`：文字列をバイトスライスに変換します。
- `encoding/hex::encoding/hex.Decode`：16進数エンコードされたバイト列をデコードします。
- `crypto/x509::crypto/x509.ParsePKCS1PublicKey`：PKCS#1形式の公開鍵を解析します。

3-1. 関数読解

関数main.iJPFPLYNoyJにおいて、どのような情報が暗号化されているかを、param_1/param_2を追跡することで確認

Cross referenceによりmain.RdufFyaBCoSaS関数から呼び出されていることを確認



3-1. 関数読解

関数main.RdufFyaBCoSaSでグローバル変数がparam_1/param_2に格納されていることを確認。

これらのグローバル変数を追跡するとmain.EncryptFileReadWriteにて[]byteに変換後、crypto/aes.NewCipher、及びcrypto/cipher.newCFBに入力されている

それぞれのソースコードから暗号化対象となるグローバル変数が**Key/IV**とわかる。

https://github.com/golang/go/blob/master/src/crypto/aes/cipher.go

func NewCipher(key []byte)

https://github.com/golang/go/blob/master/src/crypto/cipher/cfb.go

func newCFB(block Block, iv []byte, decrypt bool)

関数main.RdufFyaBCoSaS

```
83  main.iJPFPLYNoyJ(DAT_00689c58,DAT_00689c5c,in_stack_fffffd8c,in_stack_fffffd90,in_stack_fffffd94,
84                   in_stack_fffffd98,in_stack_fffffd9c);
85  pcVar14 = in_stack_fffffd90;
86  iVar9 = in_stack_fffffd98;
87  local_50 = in_stack_fffffd9c;
88  local_4c = in_stack_fffffd8c;
89  main.iJPFPLYNoyJ(DAT_00689c60,DAT_00689c64,in_stack_fffffd8c,in_stack_fffffd90,in_stack_fffffd94,
90                   in_stack_fffffd98,in_stack_fffffd9c);
91  pcVar11 = pcVar14;
92  iVar15 = iVar9;
93  local_5c = in_stack_fffffd9c;
94  local_54 = in_stack_fffffd8c;
95  main.iJPFPLYNoyJ(DAT_00689c30,DAT_00689c34,in_stack_fffffd8c,pcVar14,in_stack_fffffd94,iVar9,
96                   in_stack_fffffd9c);
97  pcVar6 = DAT_00689c3c;
98  pcVar12 = pcVar11;
99  iVar16 = iVar15;
100 local_44 = in_stack_fffffd9c;
101 local_40 = in_stack_fffffd8c;
102 main.iJPFPLYNoyJ(DAT_00689c38,DAT_00689c3c,in_stack_fffffd8c,pcVar11,in_stack_fffffd94,iVar15,
103                  in_stack_fffffd9c);
104 pcVar13 = pcVar12;
105 iVar17 = iVar16;
106 local_58 = in_stack_fffffd9c;
107 local_48 = in_stack_fffffd8c;
```

同様の確認でこちらも別の暗号化方式のKey/IVとわかる

関数main.EncryptFileReadWrite

```
162 runtime::runtime.stringtoslicebyte(local_1d4,DAT_00689c60,DAT_00689c64,0,iVar5,in_stack_fffffd94);
163 crypto/aes::crypto/aes.NewCipher
164         (puVar14,iVar5,in_stack_fffffd94,puVar14,iVar5,in_stack_fffffd94,iVar15);
165 if (in_stack_fffffd94 != 0) {
166   local_54[0] = in_stack_fffffd94;
167   if (in_stack_fffffd94 != 0) {
168     local_54[0] = *(uint *)(in_stack_fffffd94 + 4);
169   }
170   fmt::fmt.Fprintln(&*os.File__implements__io.Writer___runtime.itab,DAT_00689698,local_54,1,1,
171                     in_stack_fffffd94,iVar15,in_stack_fffffd9c);
172   runtime::runtime.deferreturn();
173   return;
174 }
175 local_68 = iVar5;
176 local_60 = puVar14;
177 runtime::runtime.stringtoslicebyte(local_1f4,DAT_00689c58,DAT_00689c5c,puVar14,iVar5,0);
178 uVar9 = in_stack_fffffd94 & 0xffffff00;
179 crypto/cipher::crypto/cipher.newCFB
180         (local_60,local_68,puVar14,iVar5,in_stack_fffffd94,uVar9,iVar15,in_stack_fffffd9c);
```

3-1. 関数読解

The Answer is:
7. 暗号化鍵、IVを暗号化するための公開鍵を16進数文字エンコードしたもの

3–2. 関数読解

3-2. 関数読解

2

関数 `main.ScanWalker` ではファイルごとに暗号化処理を振り分ける動きをしています。 `C:\Users\mws\Desktop` 配下にある下記のファイルの中で、 `main.ScanWalker.func2` 関数を通るものをすべて選択してください。例えば、回答が h, d, a, b の4つだとわかった場合は `a,b,d,h` のようなアルファベット昇順のカンマ区切りで指定してください。

なお、本フォルダ及びファイルはマルウェアのスキャン対象になっているものとします。

- a. 250MB の aaa.vmdk
- b. 150MB の bbb.iso
- c. 650MB の ccc.txt
- d. 50kB の ddd.jpg
- e. 200MB の eee.exe
- f. 550MB の fff.mp3
- g. 10kB の ggg.inf
- h. 1GB の hhh.mp4

3-2. 関数読解

【前提知識】

関数main.ScanWalkerは関数main.ScanPathで関数path/filepath.Walkのコールバック関数として呼び出される。関数path/filepath.Walkではrootを起点として、ファイル/フォルダを探索しコールバック関数を呼び出し。

func Walk(root string, fn WalkFunc) error

コールバック関数の形は下記の通り

```
type WalkFunc
type WalkFunc func(path string, info fs.FileInfo, err error) error
```

main.ScanWalkerのparam_3/param_4でfs.FileInfoのインターフェイス（ファイル情報を取得するインターフェイス）を指定していることがわかる。

https://pkg.go.dev/path/filepath#pkg-types

関数main.ScanWalker

```
8 void main::main.ScanPath(undefined4 param_1,undefined4 param_2)
9
10 {
11   int in_FS_OFFSET;
12   uint8 *puVar1;
13   int iVar2;
14   undefined4 in_stack_fffffdc;
15   undefined4 in_stack_fffffe0;
16   string local_18;
17   runtime._type *local_c;
18   undefined4 local_8;
19   undefined **local_4;
20
21   while (&stack0x00000000 <= *(undefined **)(**(int **)(in_FS_OFFSET + 0x14) + 8)) {
22     local_4 = (undefined **)0x54e61f;
23     runtime::runtime.morestack_noctxt();
24   }
25   local_4 = (undefined **)0x0;
26   puVar1 = gostr_..str;
27   iVar2 = gostr_..len;
28   os::os.Chdir(gostr_..str,gostr_..len,in_stack_fffffdc,in_stack_fffffe0);
29   syscall::syscall.Getwd(puVar1,iVar2,in_stack_fffffdc,in_stack_fffffe0);
30   local_18 = runtime::runtime.concatstring2(0,"scanning all files on ",0x16,puVar1,iVar2);
31   if (g_isDebugLog != '\0') {
32     local_c = (runtime._type *)0x0;
33     local_8 = 0;
34     local_8 = runtime::runtime.convTstring(local_18.str,local_18.len);
35     local_c = &string___runtime._type;
36     log::log.Println(&local_c,1,1);
37   }
38   path/filepath::path/filepath.Walk(".",1,&PTR_main.ScanWalker_00590474,puVar1,iVar2);
39   if (g_arg_TargetEncryptFile == 0) {
40     local_4 = &PTR_main.StopPools_0059048c;
41   }
42   if (((g_arg_TargetEncryptFile ^ 1) & 1) != 0) {
43     main.StopPools();
44   }
45   return;
46 }
```

```
6 void main::main.ScanWalker
7                 (code *param_1,code *param_2,int param_3,undefined4 param_4,int param_5,
8                 undefined4 param_6,undefined4 param_7,undefined4 param_8)
```

path string

info fs.FileInfo

関数main.ScanWalker

3–2. 関数読解

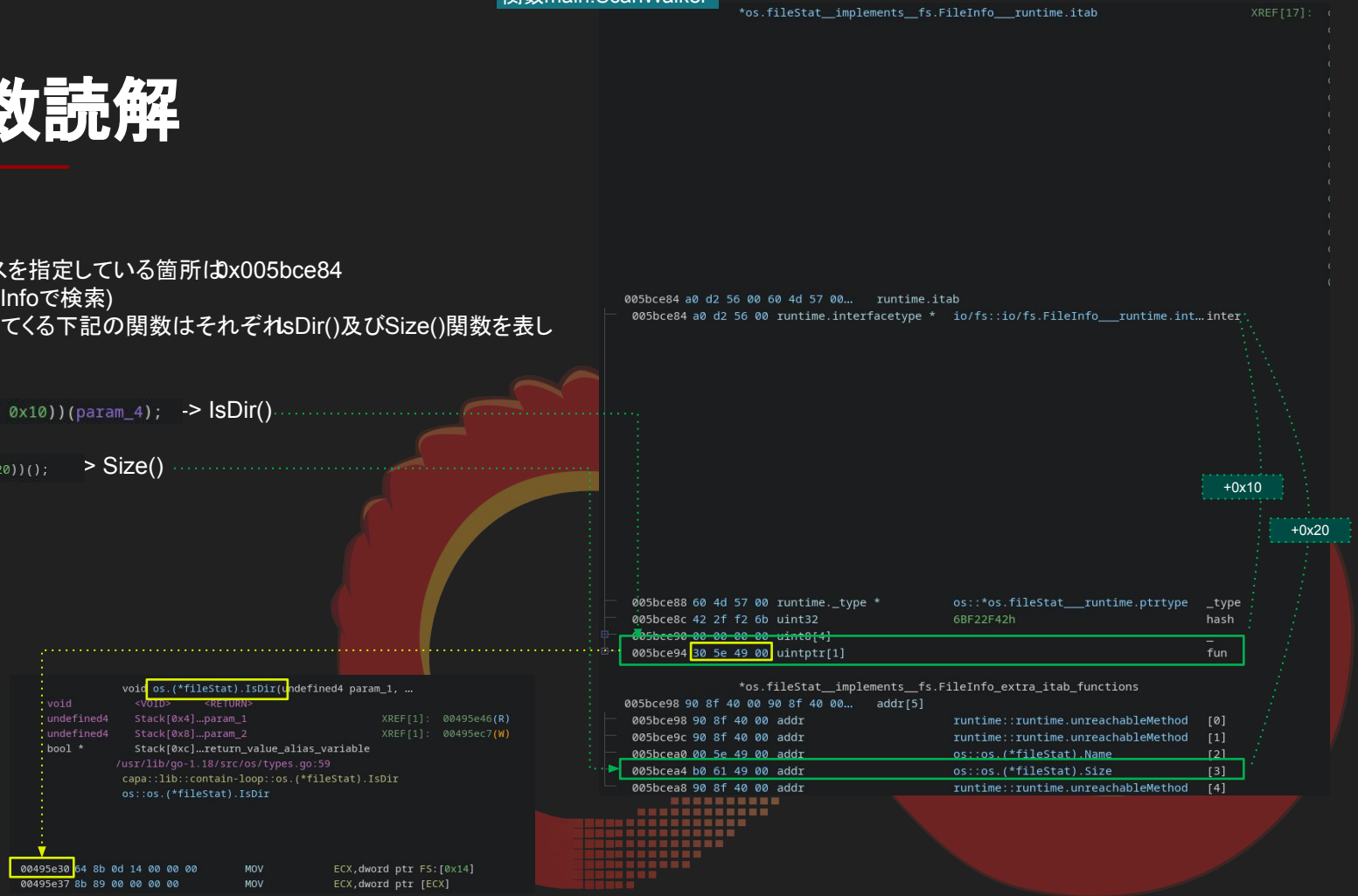
【前提知識】

fs.FileInfoのインターフェイスを指定している箇所は0x005bce84
(Comment WindowよりFileInfoで検索)
main.ScanWalkerの中で出てくる下記の関数はそれぞれIsDir()及びSize()関数を表していることがわかる。

```
69   (**(code **)(param_3 + 0x10))(param_4);
```
-> IsDir()

```
124  (**(code **)(param_3 + 0x20))();
```
> Size()

GZFに反映済み



3–2. 関数読解

関数main.ScanWalkerにてmain.ScanWalker.func2が実行されるためには、main.ScanWalker.func2実行までの関数内の分岐を確認する必要がある。

関数main.ScanWalker

```
155  if (bVar2) {
156    ppcVar16 = (code **)0x1;
157    sync::sync.(*WaitGroup).Add(&DAT_006b03a0,1);
158    runtime::runtime.newobject(&DAT_00565220,ppcVar16);
159    *ppcVar16 = main.ScanWalker.func1;
160    ppcVar16[2] = param_2;
161    if (DAT_006b0450 == 0) {
162      ppcVar16[1] = param_1;
163    }
164    else {
165      runtime::runtime.gcWriteBarrier();
166    }
167    runtime::runtime.newproc(ppcVar16);
168    iVar9 = DAT_006854cc;
169    puVar7 = PTR_DAT_006854c8;
170    puVar5 = PTR_DAT_006854c8;
171    if ((DAT_006854d0 < DAT_006854cc + 1U) &&
172       (puVar7 = in_stack_ffffffd8, iVar9 = in_stack_ffffffdc,
173       runtime::runtime.growslice
174                 (&string___runtime._type,PTR_DAT_006854c8,DAT_006854cc,DAT_006854d0,
175                  DAT_006854cc + 1U,in_stack_ffffffd8,in_stack_ffffffdc,in_stack_fffffe0),
176       DAT_006854d0 = in_stack_fffffe0, puVar5 = puVar7, DAT_006b0450 != 0)) {
177      runtime::runtime.gcWriteBarrier();
178      puVar5 = PTR_DAT_006854c8;
179    }
180    PTR_DAT_006854c8 = puVar5;
181    DAT_006854cc = iVar9 + 1;
182    *(code **)(puVar7 + iVar9 * 8 + 4) = param_2;
183    if (DAT_006b0450 == 0) {
184      *(code **)(puVar7 + iVar9 * 8) = param_1;
185    }
186    else {
187      runtime::runtime.gcWriteBarrier();
188    }
189    time::time.Sleep(5000000,0);
190    return;
191  }
192  ppcVar16 = (code **)0x1;
193  sync::sync.(*WaitGroup).Add(&DAT_006b03a0,1);
194  runtime::runtime.newobject(&DAT_00565280,ppcVar16);
195  *ppcVar16 = main.ScanWalker.func2;
196  ppcVar16[2] = param_2;
197  if (DAT_006b0450 == 0) {
198    ppcVar16[1] = param_1;
199  }
200  else {
201    runtime::runtime.gcWriteBarrier();
202  }
203  runtime::runtime.newproc(ppcVar16);
```

main.ScanWalker.func2
が実行される個所

3-2. 関数読解

関数main.ScanWalkerにてmain.ScanWalker.func2が実行されるためには、関数内の分岐で問題ないかを確認

①パスに対象の文字が含まれていないこと
②対象の拡張子が含まれていないこと

関数main.ScanWalker

```
89  main.CheckFolder(param_1,param_2,puVar17);
90  iVar4 = slice[39]_006854d8.len;
91  pvVar3 = slice[39]_006854d8.array;
92  if ((char)puVar17 != '\0') {
93    return;
94  }
95  for (iVar6 = 0; pcVar15 = pcVar14, iVar6 < iVar4; iVar6 = iVar6 + 1) {
96    ppcVar16 = (code **)((int)pvVar3 + iVar6 * 8);
97    pcVar15 = *ppcVar16;
98    pcVar12 = param_2;
99    do {
100     pcVar12 = pcVar12 + -1;
101     if ((int)pcVar12 < 0) {
102 LAB_0054e070:
103       local_c = (code *)0x0;
104       pcVar8 = (code *)0x0;
105       goto LAB_0054e07a;
106     }
107     if (param_2 <= pcVar12) {
108                   /* WARNING: Subroutine does not return */
109       runtime::runtime.panicIndex(pcVar10,pcVar14);
110     }
111     cVar1 = param_1[(int)pcVar12];
112     if ((cVar1 == (code)0x5c) || (cVar1 == (code)0x2f)) goto LAB_0054e070;
113   } while (cVar1 != (code)0x2e);
114   pcVar8 = param_2 + -(int)pcVar12;
115   local_c = param_1 + ((uint)pcVar12 & -(int)pcVar8 >> 0x1f);
116 LAB_0054e07a:
117   if ((pcVar8 == ppcVar16[1]) &&
118      (runtime::runtime.memequal(local_c,pcVar15,pcVar8,(char)iVar9), pcVar10 = local_c,
119      pcVar14 = pcVar15, (char)iVar9 != '\0')) break;
120 }
121 if (iVar6 < iVar4) {
122   return;
123 }
```

①パスに対象の文字が含まれている場合は除外

②対象の拡張子が含まれている場合は除外

フォルダの場合は除外

3–2. 関数読解

関数main.ScanWalkerにてmain.ScanWalker.func2が実行されるためには、関数内の分岐で問題ないかを確認

①パスに対象の文字が含まれていないこと
②対象の拡張子が含まれていないこと
③bVar2=falseであること(2つの条件あり)

関数main.ScanWalker

```
144 if ((((iVar9 == 4) && ((*piVar11 == 0x6c71732e || (*piVar11 == 0x7478742e)))) ||
145     ((iVar9 == 3 && ((*(short *)piVar11 == 0x642e && (*(char *)((int)piVar11 + 2) == 'b')))))) ||
146    ((iVar9 == 5 && ((*piVar11 == 0x6f736a2e && (*(char *)(piVar11 + 1) == 'n')))))) {
147   bVar2 = false;
148 }
149 else if (DAT_0066e1e8 + -10 < (int)pcVar15) {
150   bVar2 = true;
151 }
152 else {
153   bVar2 = false;
154 }
155 if (bVar2) {
156   ppcVar16 = (code **)0x1;
157   sync::sync.(*WaitGroup).Add(&DAT_006b03a0,1);
158   runtime::runtime.newobject(&DAT_00565220,ppcVar16);
159   *ppcVar16 = main.ScanWalker.func1;
160   ppcVar16[2] = param_2;
161   if (DAT_006b0450 == 0) {
162     ppcVar16[1] = param_1;
163   }
164   else {
165     runtime::runtime.gcWriteBarrier();
166   }
167   runtime::runtime.newproc(ppcVar16);
168   iVar9 = DAT_006854cc;
169   puVar7 = PTR_DAT_006854c8;
170   puVar5 = PTR_DAT_006854c8;
171   if ((DAT_006854d0 < DAT_006854cc + 1U) &&
172      (puVar7 = in_stack_fffffd8, iVar9 = in_stack_fffffdc,
173      runtime::runtime.growslice
174                (&string___runtime._type,PTR_DAT_006854c8,DAT_006854cc,DAT_006854d0,
175                 DAT_006854cc + 1U,in_stack_fffffd8,in_stack_fffffdc,in_stack_fffffe0),
176      DAT_006854d0 = in_stack_fffffe0, puVar5 = puVar7, DAT_006b0450 != 0)) {
177     runtime::runtime.gcWriteBarrier();
178     puVar5 = PTR_DAT_006854c8;
179   }
180   PTR_DAT_006854c8 = puVar5;
181   DAT_006854cc = iVar9 + 1;
182   *(code **)(puVar7 + iVar9 * 8 + 4) = param_2;
183   if (DAT_006b0450 == 0) {
184     *(code **)(puVar7 + iVar9 * 8) = param_1;
185   }
186   else {
187     runtime::runtime.gcWriteBarrier();
188   }
189   time::time.Sleep(5000000,0);
190   return;
191 }
192 ppcVar16 = (code **)0x1;
193 sync::sync.(*WaitGroup).Add(&DAT_006b03a0,1);
194 runtime::runtime.newobject(&DAT_00565280,ppcVar16);
195 *ppcVar16 = main.ScanWalker.func2;
196 ppcVar16[2] = param_2;
197 if (DAT_006b0450 == 0) {
198   ppcVar16[1] = param_1;
199 }
200 else {
201   runtime::runtime.gcWriteBarrier();
202 }
203 runtime::runtime.newproc(ppcVar16);
```

③bVar2=trueの場合は除外

3–2. 関数読解

①パスに対象の文字が含まれていないこと

関数main.CheckFolderを確認。文字列リストが含まれているかを判定

関数main.CheckFolder

```
6 void __thiscall main::main.CheckFolder(undefined4 param_1,undefined4 param_2,undefined4 param_3)
7
8 {
9   char **ppcVar1;
10  char *pcVar2;
11  char *pcVar3;
12  void *pvVar4;
13  int iVar5;
14  int iVar6;
15  int in_FS_OFFSET;
16  int in_stack_fffffff8;
17
18  while (iVar5 = slice[30]_006854f8.len, pvVar4 = slice[30]_006854f8.array,
19        &stack0x00000000 <= *(undefined **)(*(int **)(in_FS_OFFSET + 0x14) + 8)) {
20    runtime::runtime.morestack_noctxt();
21  }
22  iVar6 = 0;
23  while( true ) {
24    if (iVar5 <= iVar6) {
25      return;
26    }
27    ppcVar1 = (char **)((int)pvVar4 + iVar6 * 8);
28    pcVar2 = ppcVar1[1];
29    pcVar3 = *ppcVar1;
30    strings::strings.Index(param_1,param_2,pcVar3,pcVar2,in_stack_fffffff8);
31    if (((-1 < in_stack_fffffff8) && (pcVar2 != (char *)0x0)) &&
32       ((pcVar2 != (char *)0x1 || (*pcVar3 != ' ')))) break;
33    iVar6 = iVar6 + 1;
34  }
35  return;
36 }
```

Path内の文字列判定

```
                    slice[30]_006854f8
006854f8 00 7e 68 00 1e 00 00 00…   runtime.s…
006854f8 00 7e 68 00 void *         DAT_00687e00        array
006854fc 1e 00 00 00 int            1Eh                 len
00685500 1e 00 00 00 int            1Eh                 cap

                    DAT_00687e00                        XREF[6]:  main.CheckFolder:0054dd1f…
                                                                  main.CheckFolder:0054dd48…
                                                                  main.CheckFolder:0054dd52…
                                                                  main.init:005514a7(W),
                                                                  main.init:005514af(*),
                                                                  006854f8(*)
00687e00 00 00 00 00        undefined4  00000000h

                    DAT_00687e04                        XREF[2]:  main.CheckFolder:0054dd4b…
                                                                  main.init:00551497(W)
00687e04 00 00 00 00        undefined4  00000000h

                    goss_Program_Files_(xor86)_687e08
                    gostr_Program_Files_(xor86)
00687e08 c5 f8 57 00 15 00 00 00    string

                    goss_boot_687e10
                    gostr_boot
00687e10 2c b5 57 00 04 00 00 00    string

                    goss_efi_687e18
                    gostr_efi
00687e18 23 b3 57 00 03 00 00 00    string

                    goss_Windows_687e20
                    gostr_Windows
00687e20 df bc 57 00 07 00 00 00    string

                    goss_sources_687e28
                    gostr_sources
00687e28 80 bd 57 00 07 00 00 00    string

                    goss_Default_687e30
                    gostr_Default
00687e30 96 bb 57 00 07 00 00 00    string

                    goss_Public_687e38
                    gostr_Public
```

```
Program Files (xor86)
boot
efi
Windows
sources
Default
```

【文字列リスト】
Program Files (xor86), boot, efi, Windows, sources, Default, Public, Open, sources64, Temp, System32, AppData, SysWOW64, BOOTNXT, support, upgrade, DumpStack.log.tmp, pagefile.sys, swapfile.sys, desktop.ini, ntuser.dat, thumbs, System Volume Information, $Recycle.Bin, perflogs, App Data, appdata, ProgramData, Recovery,

3-2. 関数読解

①パスに対象の文字が含まれていないこと

関数main.CheckFolderを確認。文字列リストが含まれているかを判定

関数main.CheckFolder

```
6 void main::main.CheckFolder(undefined4 param_1,undefined4 param_2,undefined4 param_3)
7
8 {
9   char **ppcVar1;
10  char *pcVar2;
11  char *pcVar3;
```

```
0054dd90 74 9f           JZ          LAB_0054dd31

                         /root/kuiper-20-11/windowsbuild.go:1230
                         LAB_0054dd92                                XREF[1]:    0054dd87(j)
0054dd92 c6 44 24        MOV         byte ptr [ESP + param_3],0x1
         34 01
0054dd97 83 c4 28        ADD         ESP,0x28
0054dd9a c3              RET

                         /root/kuiper-20-11/windowsbuild.go:1250
                         LAB_0054dd9b                                XREF[1]:    0054dd42(j)
0054dd9b c6 44 24        MOV         byte ptr [ESP + param_3],0x0
         34 00
0054dda0 83 c4 28        ADD         ESP,0x28
0054dda3 c3              RET
```

関数main.ScanWalker

```
87  main.CheckFolder(param_1,param_2,puVar6);
88  iVar8 = slice[39]_006854d8.len;
89  pvVar3 = slice[39]_006854d8.array;
90  if ((char)puVar6 != '\0') {
91    return;
92  }
```

3–2. 関数読解

②対象の拡張子が含まれていないこと

拡張子リストをもとに含まれていないことを判定

関数main.ScanWalker

```
90  iVar4 = slice[39]_006854d8.len;
91  pvVar3 = slice[39]_006854d8.array;
92  if ((char)puVar17 != '\0') {
93    return;
94  }
95  for (iVar6 = 0; pcVar15 = pcVar14, iVar6 < iVar4; iVar6 = iVar6 + 1) {
96    ppcVar16 = (code **)((int)pvVar3 + iVar6 * 8);
97    pcVar15 = *ppcVar16;
98    pcVar12 = param_2;
99    do {
100     pcVar12 = pcVar12 + -1;
101     if ((int)pcVar12 < 0) {
102LAB_0054e070:
103       local_c = (code *)0x0;
104       pcVar8 = (code *)0x0;
105       goto LAB_0054e07a;
106     }
107     if (param_2 <= pcVar12) {
108                 /* WARNING: Subroutine does not return */
109       runtime::runtime.panicIndex(pcVar10,pcVar14);
110     }
111     cVar1 = param_1[(int)pcVar12];
112     if ((cVar1 == (code)0x5c) || (cVar1 == (code)0x2f)) goto LAB_0054e070;
113   } while (cVar1 != (code)0x2e);
114   pcVar8 = param_2 + -(int)pcVar12;
115   local_c = param_1 + ((uint)pcVar12 & -(int)pcVar8 >> 0x1f);
116LAB_0054e07a:
117   if ((pcVar8 == ppcVar16[1]) &&
118      (runtime::runtime.memequal(local_c,pcVar15,pcVar8,(char)iVar9), pcVar10 = local_c,
119      pcVar14 = pcVar15, (char)iVar9 != '\0')) break;
120 }
121 if (iVar6 < iVar4) {
122   return;
123 }
```

拡張子の判定

拡張子が該当した場合は関数終了

```
006854d8 00 7f 68 00 27 00 00 00…   runtime.s…
006854d8 00 7f 68 00 void *          gostr_.ini                array
006854dc 27 00 00 00 int             27h                       len
006854e0 27 00 00 00 int             27h                       cap
```

```
00687f00 04 b4 57 00 04 00 00 00    string
00687f00 04 b4 57 00 uint8 *        s_.ini_0057b404           str        .ini        XREF[4]:  main.ScanWalker:0054df3e(…
                                                                                             main.ScanWalker:0054e031(…
                                                                                             main.ScanWalker:0054e034(…
                                                                                             006854d8(*)
00687f04 04 00 00 00 int            4h                        len                    XREF[1]:  main.ScanWalker:0054e03a(…

                 goss_.key_687f08
                 gostr_.key
00687f08 14 b4 57 00 04 00 00 00    string                                                  .key

                 goss_.cert_687f10
                 gostr_.cert
00687f10 62 b6 57 00 05 00 00 00    string                                                  .cert

                 goss_.private_key_687f18
                 gostr_.private_key
00687f18 06 ce 57 00 0c 00 00 00    string                                                  .private_key

                 goss_.man_687f20
                 gostr_.man
00687f20 1c b4 57 00 04 00 00 00    string                                                  .man

                 goss_.cfg_687f28
                 gostr_.cfg
00687f28 d8 b3 57 00 04 00 00 00    string                                                  .cfg

                 goss_.DATA_687f30
                 gostr_.DATA
00687f30 44 b6 57 00 05 00 00 00    string                                                  .DATA

                 goss_.inf_687f38
                 gostr_.inf
00687f38 00 b4 57 00 04 00 00 00    string                                                  .inf
```

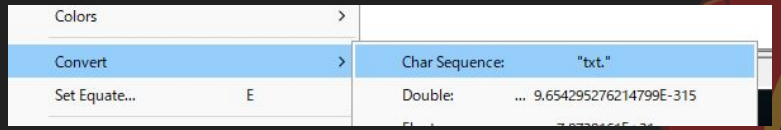
拡張子リスト
ini,key,cert,private_key,man,cfg,DATA,inf,LOG2,LOG,LOG1,ttf,regtrans-ms,searchms,desktop,Desktop,tmp,DAT,TMP,so,dll,exe,go,msi,html,bak,dat,blf,386,lnk,bat,cmd,sys,crlk,bin,elf,rtf,com

3-2. 関数読解

③bVar2=falseであること(2つの条件あり)

1つ目の条件は下記の拡張子であること

.txt
.db
.json
.sql



関数main.ScanWalker

.sql .txt .jso .d

```
144  if ((((iVar9 == 4) && ((*piVar11 == 0x6c71732e || (*piVar11 == 0x7478742e)))) ||
145      ((iVar9 == 3 && ((*(short *)piVar11 == 0x642e && (*(char *)((int)piVar11 + 2) == 'b')))))) ||
146     ((iVar9 == 5 && ((*piVar11 == 0x6f736a2e && (*(char *)(piVar11 + 1) == 'n')))))) {
147    bVar2 = false;
148  }
149  else if (DAT_0066e1e8 + -10 < (int)pcVar15) {
150    bVar2 = true;
151  }
152  else {
153    bVar2 = false;
154  }
155  if (bVar2) {
```

3–2. 関数読解

③bVar2=falseであること（2つの条件あり）

2つ目の条件はファイルのサイズが0xbd00000-10（198,180,854）以下であること

関数main.ScanWalker

```
144  if ((((iVar9 == 4) && ((*piVar11 == 0x6c71732e || (*piVar11 == 0x7478742e)))) ||
145      ((iVar9 == 3 && ((*(short *)piVar11 == 0x642e && (*(char *)((int)piVar11 + 2) == 'b')))))) ||
146     ((iVar9 == 5 && ((*piVar11 == 0x6f736a2e && (*(char *)(piVar11 + 1) == 'n')))))) {
147    bVar2 = false;
148  }
149  else if (DAT_0066e1e8 + -10 < (int)pcVar15) {
150    bVar2 = true;
151  }
152  else {
153    bVar2 = false;
154  }
155  if (bVar2) {
```

pcVer15については、Disassemblerをもとに追跡する必要あり

```
DAT_0066e1e8
0066e1e8 00 00 d0 0b        undefined4   0BD00000h
```

EAXが該当のデータが格納されているため、EAXを追跡

```
0054e17a 8b 15 e8 e1 66 00    MOV     EDX,dword ptr [DAT_0066e1e8]
0054e180 83 c2 f6             ADD     EDX,-0xa
0054e183 39 d0                CMP     EAX,EDX
0054e185 7f 04                JG      LAB_0054e18b
```

```
0054e0c0 7c 21                JL      LAB_0054e0e3
                 /root/kuiper-20-11/windowsbuild.go:1313
0054e0c2 8b 44 24 48          MOV     EAX,dword ptr [ESP + 0x48]=>param_3
0054e0c6 8b 40 20             MOV     EAX,dword ptr [EAX + 0x20]
                 /root/kuiper-20-11/windowsbuild.go:1296
0054e0c9 8b 4c 24 4c          MOV     ECX,dword ptr [ESP + 0x4c]=>param_4
0054e0cd 89 0c 24             MOV     dword ptr [ESP]=>local_3c,ECX
                 /root/kuiper-20-11/windowsbuild.go:1313
0054e0d0 ff d0                CALL    EAX
0054e0d2 8b 44 24 04          MOV     EAX,dword ptr [ESP + 0x4]=>local_38
```

```
                 void os.(*fileStat).Size(undefined4 param_1, u…
      void           <VOID>      <RETURN>
      undefined4     Stack[0x4]…param_1                    XREF[1]:  004961c2(R)
      undefined4     Stack[0x8]…param_2                    XREF[1]:  004961d6(W)
      undefined4     Stack[0xc]…param_3                    XREF[1]:  004961d2(W)
                 /usr/lib/go-1.18/src/os/types_windows.go:108
                  capa::lib::contain-loop::os.(*fileStat).Size
                  os::os.(*fileStat).Size                  XREF[6]:  004961e0(j), 00574dec(*),
                                                                     00574df0(*), 005bcea4(*),
                                                                     0063bf30(*), 0065b2dc(*)
004961b0 64 8b 0d 14 00 00 00   MOV     ECX,dword ptr FS:[0x14]
004961b7 8b 89 00 00 00 00      MOV     ECX,dword ptr [ECX]
004961bd 3b 61 08               CMP     ESP,dword ptr [ECX + 0x8]
004961c0 76 19                  JBE     LAB_004961db
                 /usr/lib/go-1.18/src/os/types_windows.go:109
004961c2 8b 44 24 04            MOV     EAX,dword ptr [ESP + 0x4]=>param_1
004961c6 8b 48 28               MOV     ECX,dword ptr [EAX + 0x28]
004961c9 83 c1 00               ADD     ECX,0x0
004961cc 8b 40 24               MOV     EAX,dword ptr [EAX + 0x24]
004961cf 83 d0 00               ADC     EAX,0x0
004961d2 89 44 24 0c            MOV     dword ptr [ESP + 0xc]=>param_3,EAX
004961d6 89 4c 24 08            MOV     dword ptr [ESP + 0x8]=>param_2,ECX
004961da c3                     RET
                 /usr/lib/go-1.18/src/os/types_windows.go:108
                  LAB_004961db                             XREF[1]:  004961c0(j)
004961db e8 60 93 fc ff         CALL    runtime::runtime.morestack_noctxt
004961e0 eb ce                  JMP     os::os.(*fileStat).Size
004961e2 cc                     ??      CCh
004961e3 cc                     ??      CCh
```

引数8byte*1

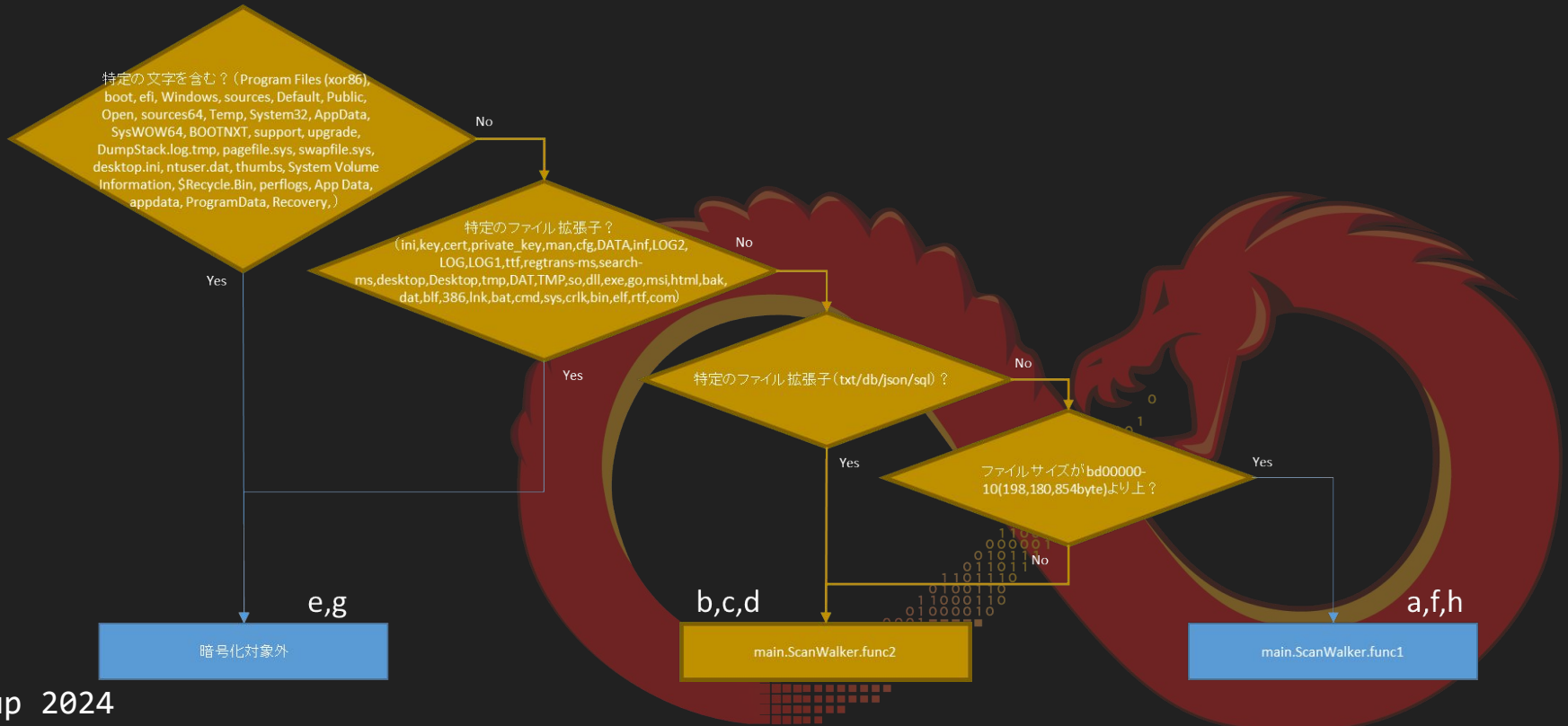
返り値8byte*2

Size()は引数を1つ返り値を2つ取る

関数(**(code **)(param_3 + 0x20))();
-> Size()の実行結果と判明

```
69   (**(code **)(param_3 + Size() (param_4);
```

3-2. 関数読解 Part 1

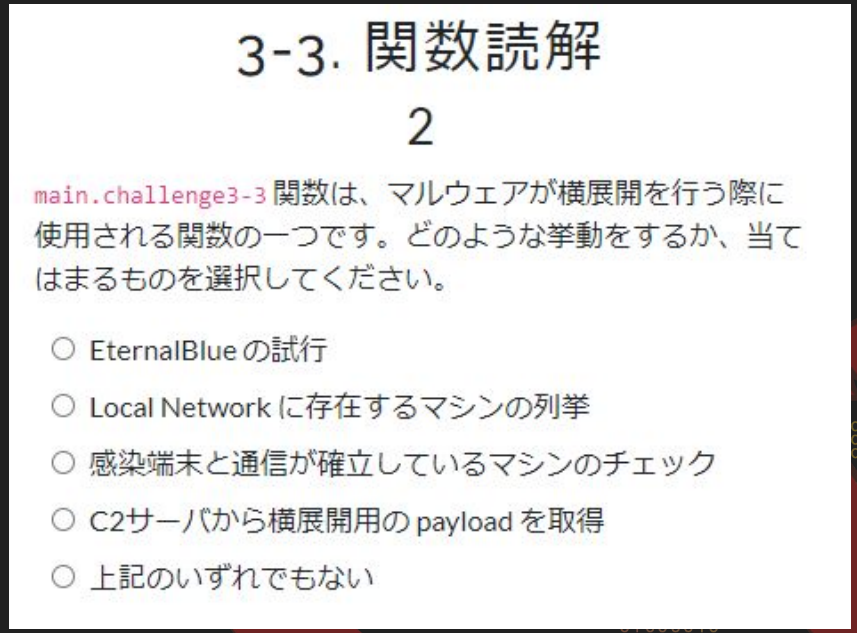


3-2. 関数読解

The Answer is:

b,c,d

3-3. 関数読解


3-3. 関数読解
2
main.challenge3-3 関数は、マルウェアが横展開を行う際に使用される関数の一つです。どのような挙動をするか、当てはまるものを選択してください。
EternalBlue の試行
Local Network に存在するマシンの列挙
感染端末と通信が確立しているマシンのチェック
C2サーバから横展開用の payload を取得
上記のいずれでもない

3-3. 関数読解

- Go の標準関数から流れを大まかに確認する
- 文字列の再結合
  - 入力文字列を genSplit で “.” の文字列単位に分割し、分割した文字列の前半3つの文字列と “.” を concatstrings で再結合

☞ **IPv4アドレスを作ろうとしてそう??**

```
42  char *local_8;
43  undefined4 local_4;
44
45  while (&local_20 <= *(uint4 **)(**(int **)(in_FS_OFFSET + 0x14) + 8)) {
46    local_4 = 0x54848c;
47    runtime::runtime.morestack_noctxt();
48  }
49  puVar7 = (uint4 *)0x0;
50  puVar8 = (uint4 *)0xffffffff;
51  strings::strings.genSplit
52            (param_1,param_2,".",1,0,0xffffffff,in_stack_ffffff78,in_stack_ffffff7c,
53             in_stack_ffffff80);
54  if (3 < (int)in_stack_ffffff7c) {
55    local_48 = in_stack_ffffff78;
56    runtime::runtime.duffzero_0x0_0x30_runtime.duffzero_00460b34(
57    local_30 = *local_48;
58    local_2c = local_48[1];
59    local_24 = 1;
60    local_28 = ".";
61    local_20 = local_48[2];
62    local_1c = local_48[3];
63    local_14 = 1;
64    local_18 = ".";
65    local_10 = local_48[4];
66    local_c = local_48[5];
67    local_4 = 1;
68    local_8 = ".";
69    puVar6 = (uint4 *)&DAT_00000006;
70    runtime::runtime.concatstrings(local_68,&local_30,6,6,puVar7,puVar8);
71    puVar2 = (uint4 *)0x0;
```

```
struct string {
    char* str;
    int len;
}
```

Go の文字列は上記のような文字列へのポインタとサイズの構造体として扱われます

3-3. 関数読解

- IPアドレスを列挙
  - 0 から 255 までのループカウンタを先ほどの文字列と結合
- smbの疎通確認
  - IsSmbPortOpen
  - smb のポートが空いているIPアドレスを探索している

```
puVar2 = (uint4 *)0x0;
puVar3 = (uint4 *)0x0;
                /* /root/kuiper-20-11/windowsbuild.go:446 */
local_3c = &DAT_006b02d0;
puVar9 = puVar8;
local_40 = puVar7;
                /* /root/kuiper-20-11/windowsbuild.go:445 */
                /* /root/kuiper-20-11/windowsbuild.go:453 */
for (iVar1 = 0; iVar1 < 0x100; iVar1 = iVar1 + 1) ← 0 ~ 255 までのループカウンタ
 strconv::strconv.FormatInt(iVar1,iVar1 >> 0x1f,10,puVar6,puVar7);
                /* /root/kuiper-20-11/windowsbuild.go:452 */
                /* /root/kuiper-20-11/windowsbuild.go:454 */
 runtime::runtime.concatstring2(0,local_40,puVar8,puVar6,puVar7,puVar9,in_stack_ffffff78);
 puVar6 = puVar9;
 puVar4 = (uint4 *)&DAT_00000009;
```

snip.

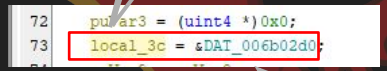
```
if (((param_2 != in_stack_ffffff78) ||
    (puVar5 = in_stack_ffffff78,
    runtime::runtime.memequal(local_44,param_1,in_stack_ffffff78,(char)puVar6),
    (char)puVar6 == '\0')) &&
   (main.IsSmbPortOpen(local_44,in_stack_ffffff78,puVar5), (char)puVar5 != '\0')) {
  puVar5 = (uint4 *)&DAT_00000009;
  puVar6 = local_44;
  puVar7 = in_stack_ffffff78;
                /* /root/kuiper-20-11/windowsbuild.go:454 */
                /* /root/kuiper-20-11/windowsbuild.go:459 */
  runtime::runtime.concatstring2(0,"smb open ",9,local_44,in_stack_ffffff78,puVar9,puVar10);
                /* /root/kuiper-20-11/windowsbuild.go:1106 */
```

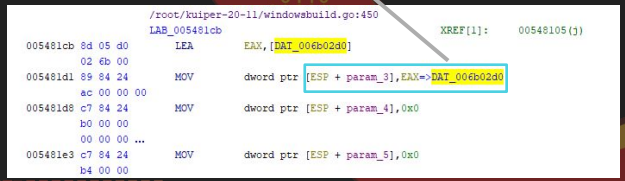
自分と同じIPアドレスならskip

違う場合は smb への疎通確認

3-3. 関数読解

- 疎通ができたIPアドレスは引数に追加される
- その他特筆すべき機能なし





3-3. 関数読解

- 最後に、読び出し元である main.cziIdlAll 関数を確認
- challenge3-3 で収集された IPアドレスが横展開のための関数に渡っていることが確認できる

☞ **Local Network の smb 探索だけを行う関数！！**

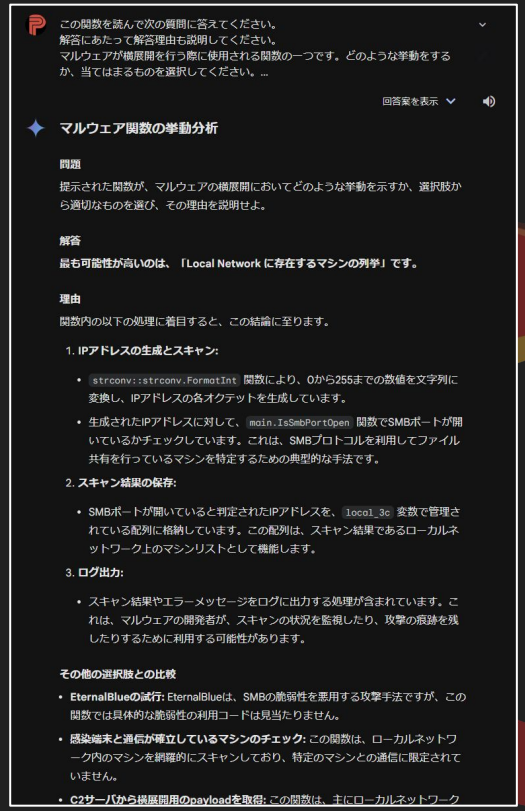
```
else {
  puVar3 = gostr_auto.str;
  iVar6 = gostr_auto.len;
  main.challenge4((main *)gostr_auto.str);
  DAT_0068551c = local_38;
  DAT_00685520 = local_34;
  pprVar2 = (runtime.
  if (DAT_006b0450 !=
    runtime::runtime.
    pprVar2 = (runtim
  }
}
PTR_DAT_00685518 = (undefined *)pprVar2;
if (DAT_0066e1c1 != '\0') {
  local_18 = (runtime._type *)0x0;
  local_14 = (code *)0x0;
  runtime::runtime.copyTstring((runtime *)&DAT_00582ac3);
  local_18 = &string_
  local_14 = pcVar9;
  log::log.Println(&l
}
main.StartProcessKill
pcVar9 = DAT_00685520;
main.StartRemoteAutoSpreadAndCopy((main *)PTR_DAT_00685518);
ppcVar7 = (code **)0x1;
```

```
LAB_00550f1e                                    XREF[2]:    00550ed9(j), 0
    MOV     dword ptr [ESP]=>local_44,EAX=>s_auto_0057b510   = "auto"
4   MOV     dword ptr [ESP + local_40],ECX
    CALL    main::main.challenge3-3                          void main.c
```

引数のアドレスが同じ!!

```
MOV     dword ptr [ESP]=>local_44,EAX
MOV     dword ptr [ESP + local_40],ECX
MOV     dword ptr [ESP + local_3c],EDX
CALL    main::main.StartRemoteAutoSpreadAndCopy
```

AIに聞いてみる



この関数を読んで次の質問に答えてください。
解答にあたって解答理由も説明してください。
マルウェアが横展開を行う際に使用される関数の一つです。どのような挙動をするか、当てはまるものを選択してください。...

回答案を表示

マルウェア関数の挙動分析

問題

提示された関数が、マルウェアの横展開においてどのような挙動を示すか、選択肢から適切なものを選び、その理由を説明せよ。

解答

最も可能性が高いのは、「Local Network に存在するマシンの列挙」です。

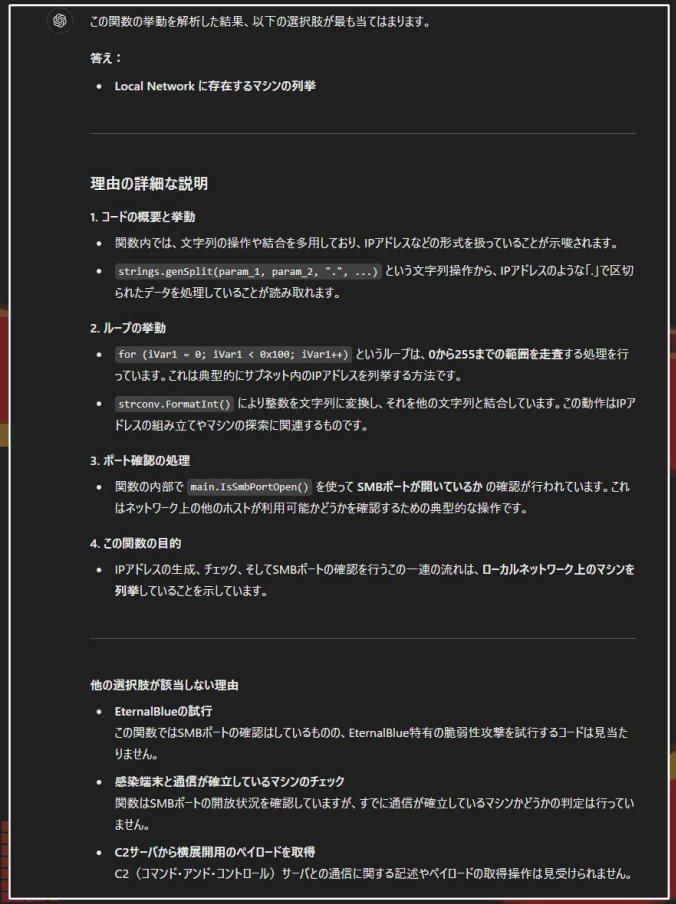
理由

関数内の以下の処理に着目すると、この結論に至ります。

1. **IPアドレスの生成とスキャン:**
   - `strconv::strconv.FormatInt` 関数により、0から255までの数値を文字列に変換し、IPアドレスの各オクテットを生成しています。
   - 生成されたIPアドレスに対して、`main.IsSmbPortOpen` 関数でSMBポートが開いているかチェックしています。これは、SMBプロトコルを利用してファイル共有を行っているマシンを特定するための典型的な手法です。
2. **スキャン結果の保存:**
   - SMBポートが開いていると判定されたIPアドレスを、`local_3c` 変数で管理されている配列に格納しています。この配列は、スキャン結果であるローカルネットワーク上のマシンリストとして機能します。
3. **ログ出力:**
   - スキャン結果やエラーメッセージをログに出力する処理が含まれています。これは、マルウェアの開発者が、スキャンの状況を監視したり、攻撃の痕跡を残したりするために利用する可能性があります。

その他の選択肢との比較

- **EternalBlueの試行:** EternalBlueは、SMBの脆弱性を悪用する攻撃手法ですが、この関数では具体的な脆弱性の利用コードは見当たりません。
- **感染端末と通信が確立しているマシンのチェック:** この関数は、ローカルネットワーク内のマシンを網羅的にスキャンしており、特定のマシンとの通信に限定されていません。
- **C2サーバから横展開用のpayloadを取得:** この関数は、主にローカルネットワーク



この関数の挙動を解析した結果、以下の選択肢が最も当てはまります。

答え：

- **Local Network に存在するマシンの列挙**

理由の詳細な説明

1. コードの概要と挙動

- 関数内では、文字列の操作や結合を多用しており、IPアドレスなどの形式を扱っていることが示唆されます。
- `strings.genSplit(param_1, param_2, ".", ...)` という文字列操作から、IPアドレスのような「.」で区切られたデータを処理していることが読み取れます。

2. ループの挙動

- `for (iVar1 = 0; iVar1 < 0x100; iVar1++)` というループは、**0から255までの範囲を走査**する処理を行っています。これは典型的にサブネット内のIPアドレスを列挙する方法です。
- `strconv.FormatInt()` により整数を文字列に変換し、それを他の文字列と結合しています。この動作はIPアドレスの組み立てやマシンの探索に関連するものです。

3. ポート確認の処理

- 関数の内部で `main.IsSmbPortOpen()` を使って **SMBポートが開いているか** の確認が行われています。これはネットワーク上の他のホストが利用可能かどうかを確認するための典型的な操作です。

4. この関数の目的

- IPアドレスの生成、チェック、そしてSMBポートの確認を行うこの一連の流れは、**ローカルネットワーク上のマシンを列挙**していることを示しています。

他の選択肢が該当しない理由

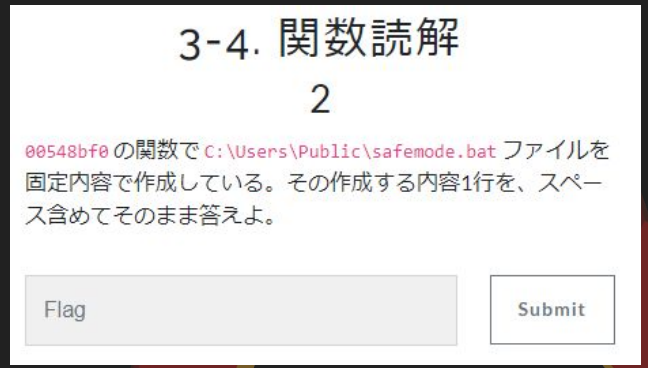
- **EternalBlueの試行**
  この関数ではSMBポートの確認はしているものの、EternalBlue特有の脆弱性攻撃を試行するコードは見当たりません。
- **感染端末と通信が確立しているマシンのチェック**
  関数はSMBポートの開放状況を確認していますが、すでに通信が確立しているマシンかどうかの判定は行っていません。
- **C2サーバから横展開用のペイロードを取得**
  C2（コマンド・アンド・コントロール）サーバとの通信に関する記述やペイロードの取得操作は見受けられません。

3-3. 関数読解

The Answer is: 2

Local Network に存在するマシンの列挙

3-4. 関数読解


3-4. 関数読解
2
00548bf0 の関数で C:\Users\Public\safemode.bat ファイルを固定内容で作成している。その作成する内容1行を、スペース含めてそのまま答えよ。
Flag
Submit

3-4. 関数読解

- os::os.WriteFileでC:\Users\Public\safemode.batファイルを
  puVar1内容の0x8a=138バイトで作成している
- puVar1が指す領域の先頭4バイトへ”reg “の4文字を書き込む
- puVar1が指す領域の3バイト目以降へ、0x00587c6aからコピーする
  （コピーサイズのアライメントの都合か
  3～4バイト目へは2回書き込まれている）

https://pkg.go.dev/os#WriteFile

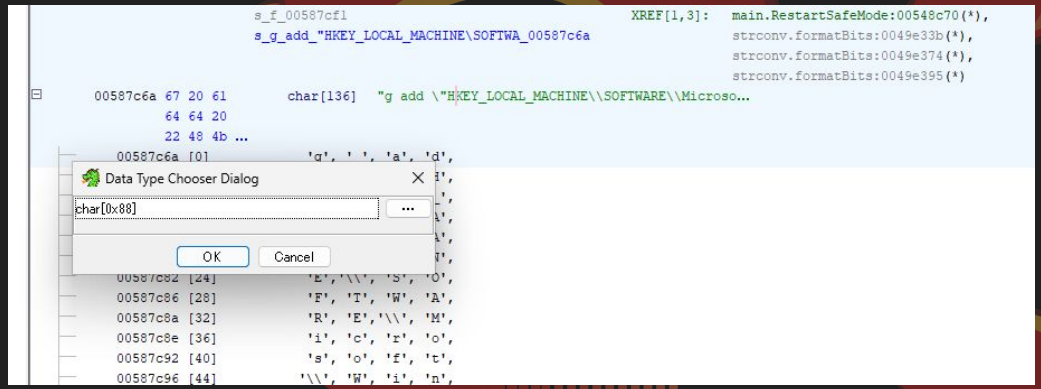
func WriteFile

```
func WriteFile(name string, data []byte, perm FileMode) error
```

```
                    /* ASCII: "reg " */
*puVar1 = 0x20676572;
runtime::runtime.duffcopy_0x88_runtime.duffcopy_0x88_runtime.duffcopy_00460efc
          ((uint4 *)((int)puVar1 + 2),(uint4 *)&DAT_00587c6a);
ppcVar2 = (code **)&DAT_0000001c;
uVar3 = 0x8a;
                    /* /usr/lib/go-1.18/src/io/ioutil/ioutil.go:46 */
os::os.WriteFile("C:\\Users\\Public\\safemode.bat",0x1c,puVar1,0x8a,0x8a,0x1a4,in_stack_ffffffc8,
                 in_stack_fffffffc);
```

3-4. 関数読解

- 3バイト目以降として使われる
  0x00587c6aからの0x88(136)バイト分を確認する
- Data -> Chose Data Type
  - char[0x88]を指定



3-4. 関数読解

The Answer is:
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /t REG_SZ /v Shell /d "C:\Users\Public\safemode.exe" /f

3-4. 関数読解 – 参考

- 公開情報の解析記事に、実行コマンドとして次の記載がある

T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder Example command: - reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersionWinlogon" /t REG_SZ /v Shell /d "C:\Users\Public\safemode.exe" /f

- レジストリパスのCurrentVersionとWinLogonの間のバックスラッシュが抜けている
- このように公開情報が誤っている場合もあるため注意

https://www.trellix.com/blogs/research/the-evolution-of-the-kuiper-ransomware/

3-5. 関数読解


3-5. 関数読解
2
0054ddb0 の関数(main::main.ScanWalker)には README_TO_DECRYPT.txt ファイルを作成する処理がある。作成内容は、バイナリ中にハードコードされた特定の文字列を、変換した結果が元になっている。そのハードコードされた文字列の先頭8文字を答えよ。
例えばハードコードされた「FooBarBaz」文字列を変換して README_TO_DECRYPT.txt ファイル内容を作成している場合は、FooBarBa と答えよ。
Flag
Submit

3-5. 関数読解

- README_TO_DECRYPT.txtファイルは、
  0054dffbで呼び出しているos::os.WriteFileで作成される。

```
                /* "README_TO_DECRYPT.txt"のファイルパスを作成 */
runtime::runtime.concatstring3
          (0,param_1,param_2,"\\",1,gostr_README_TO_DECRYPT.txt.str,
           gostr_README_TO_DECRYPT.txt.len,in_stack_fffffe0,in_stack_ffffffe4);
                /* "README_TO_DECRYPT.txt"内容を書き込み */
os::os.WriteFile(in_stack_fffffe0,in_stack_ffffffe4,PTR_DAT_00685508,DAT_0068550c,DAT_00685510,
                 0x180,iVar9,in_stack_fffffe0);
```

func WriteFile

```
func WriteFile(name string, data []byte, perm FileMode) error
```

https://pkg.go.dev/os#WriteFile

3-5. 関数読解

- data引数に対応するDAT_0068550cとDAT_00685510の書き込み場所を調べると、0054e768付近であると分かる

```
                    DAT_0068550c                                    XREF[3]:     main.ScanWalker:0054dfc6(R),
                                                                                 main.cziIdlNote:0054e768(W),
                                                                                 main.RenameAllFiles:0054e83a(R)
0068550c 00 00 00 00     undefined4  00000000h
```

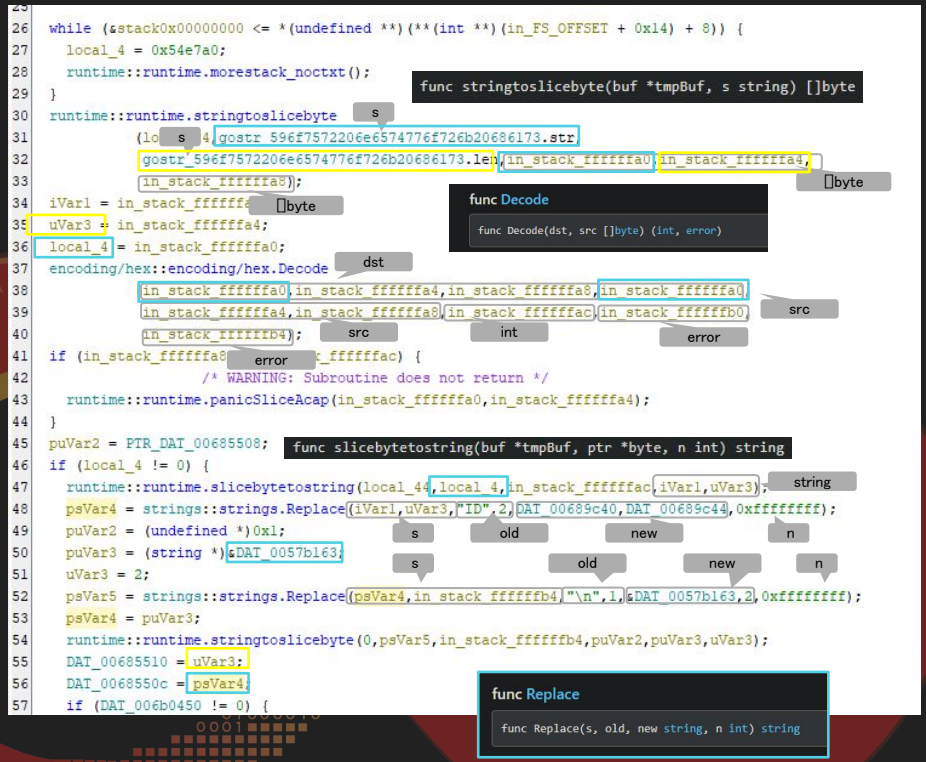
```
                    DAT_00685510                                    XREF[3]:     main.ScanWalker:0054dfcc(R),
                                                                                 main.cziIdlNote:0054e76e(W),
                                                                                 main.RenameAllFiles:0054e840(R)
00685510 00 00 00 00     undefined4  00000000h
00685514 00              ??          00h
```

```
                    /root/kuiper-20-11/windowsbuild.go:1366
0054e757 e8 74 f5        CALL        runtime::runtime.stringtoslicebyte
         ef ff
0054e75c 8b 44 24 0c     MOV         EAX,dword ptr [ESP + local_60]
0054e760 8b 4c 24 10     MOV         ECX,dword ptr [ESP + local_5c]
0054e764 8b 54 24 14     MOV         EDX,dword ptr [ESP + local_58]
0054e768 89 0d 0c        MOV         dword ptr [DAT_0068550c],ECX
         55 68 00
0054e76e 89 15 10        MOV         dword ptr [DAT_00685510],EDX
         55 68 00
```

```
53    psVar4 = puVar3;
54    runtime::runtime.stringtoslicebyte(0,psVar5,in_stack_ffffffb4,puVar2,puVar3,uVar3);
55    DAT_00685510 = uVar3;
56    DAT_0068550c = psVar4;
```

3-5. 関数読解

- 周辺の処理を調査
  - 006852f8の文字列をHexDecode
  - IDや改行コードを置き換え
  - DAT_0068550cとDAT_00685510を使って書き込み



3-5. 関数読解

- 006852f8の文字列をHexDecodeしてIDや改行コードを置き換えたものがREADME_TO_DECRYPT.txt内容になる
- バイナリ中にハードコードされた特定の文字列は006852f8の文字列のことだと判断できる
- 006852f8の文字列
  - データは005888c8に存在する
- 005888c8
  - 596f7572から始まるハードコードされた文字列が見つかる

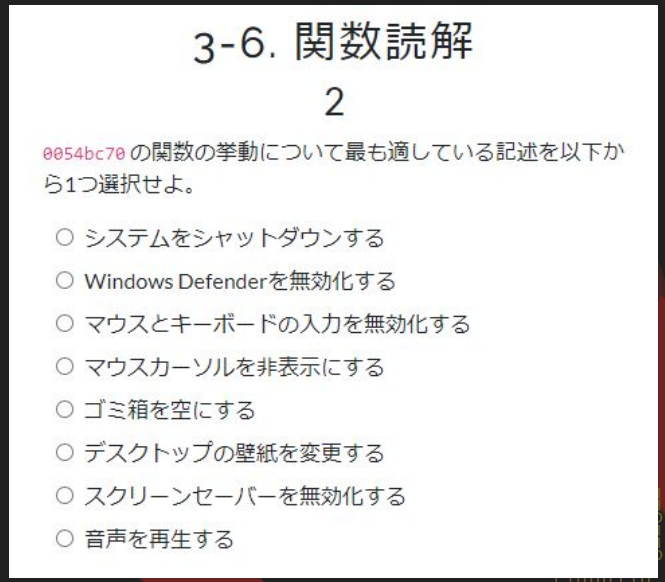
```
006852f8 c8 88 58        string
         00 f2 0b
         00 00
   006852f8 c8 88 58 00     uint8 *    s_596f7572206e6574776f... str
   006852fc f2 0b 00 00     int        BF2h                     len
```

```
005888c8 35 39 36        char[3058]  "596f7572206e6574776f726b20686173206265656e206...
```

3-5. 関数読解

The Answer is: 596f7572

3-6. 関数読解


3-6. 関数読解
2
0054bc70 の関数の挙動について最も適している記述を以下から1つ選択せよ。
○ システムをシャットダウンする
○ Windows Defenderを無効化する
○ マウスとキーボードの入力を無効化する
○ マウスカーソルを非表示にする
○ ゴミ箱を空にする
○ デスクトップの壁紙を変更する
○ スクリーンセーバーを無効化する
○ 音声を再生する

3-6. 関数読解

- syscall.LoadDLLでuser32.dllを読み込んでいる
- syscall.(*DLL).FindProcでSystemParametersInfoW関数のアドレスを取得している

```
22  while (&stack0x00000000 <= *(undefined **)(**(int **)(in_FS_OFFSET + 0x14) + 8)) {
23    local_4 = 0x54bdba;
24    runtime::runtime.morestack_noctxt();
25  }
26  syscall::syscall.LoadDLL("user32.dll",10,in_stack_fffffff8,in_stack_fffffffc,in_stack_ffffffe0);
27  if (in_stack_fffffffc == (syscall.Proc *)0x0) {
28    iVar2 = 0x15;
29    syscall::syscall.(*DLL).FindProc
30              (in_stack_fffffff8,"SystemParametersInfoW",0x15,0,in_stack_ffffffe0,in_stack_ffffffe4)
31    ;
```

3-6. 関数読解

- syscall.(*Proc).Callで取得したSystemParametersInfoW関数を呼び出している
  - 第1引数に取得または設定するアクションを指定する
  - 0x0014である場合はSPI_SETDESKWALLPAPER

```
C++

BOOL SystemParametersInfoW(
  [in]      UINT  uiAction,
  [in]      UINT  uiParam,
  [in, out] PVOID pvParam,
  [in]      UINT  fWinIni
);
```

```
SPI_SETDESKWALLPAPER
0x0014
```

https://learn.microsoft.com/ja-jp/windows/win32/api/winuser/nf-winuser-systemparametersinfow

```
*puVar1 = 0x14;
puVar1[1] = 0;
puVar1[2] = uVar2;
puVar1[3] = 2;
                /* /root/kuiper-20-11/windowsbuild.go:884 */
                /* /root/kuiper-20-11/windowsbuild.go:889 */
uVar2 = 4;
syscall::syscall.(*Proc).Call
          (in_stack_ffffffdc,puVar1,4,4,in_stack_fffffff0,in_stack_fffffff4,in_stack_fffffff8,
           in_stack_fffffffc);
```

AIに聞いてみる

この関数の挙動から適切な記述を選択するなら：

正解：() デスクトップの壁紙を変更する

理由

この関数は、次のような操作を行っています：

1. `user32.dll` **のロード**

```cpp
syscall::syscall.LoadDLL("user32.dll", ...);
```

   - `user32.dll` は、Windowsのユーザーインターフェイス関連の機能を提供するライブラリです。
   - このライブラリは、壁紙の変更、スクリーンセーバー設定、入力制御などの操作をサポートします。

2. `SystemParametersInfoW` **APIの取得**

```cpp
syscall::syscall.(*DLL).FindProc(in_stack_ffffffd8, "SystemParametersInfoW", ...);
```

   - `SystemParametersInfoW` は、Windows APIで、システム設定（特にUI関連）を変更するための関数です。
   - このAPIを使用すると、**デスクトップの壁紙**の変更や、スクリーンセーバー、マウスカーソルの表示/非表示といった設定を行えます。

3. **壁紙変更の可能性**

   - `SystemParametersInfoW` でサポートされる操作の中には、壁紙の変更が含まれています。
   - 関数が `user32.dll` と `SystemParametersInfoW` を使っていることから、**壁紙変更**が最も適している選択肢です。

0054bc70 の関数の挙動について最も適している記述を以下から1つ選択せよ。

* () システムをシャットダウンする
* () Windows Defenderを無効化する...

回答案を表示

関数 `main::main.challenge3-6` の挙動分析

問題

提示された関数 `main::main.challenge3-6` の最も適切な記述を以下の選択肢から選び、その理由を説明せよ。

解答

最も可能性が高いのは、「デスクトップの壁紙を変更する」です。

理由

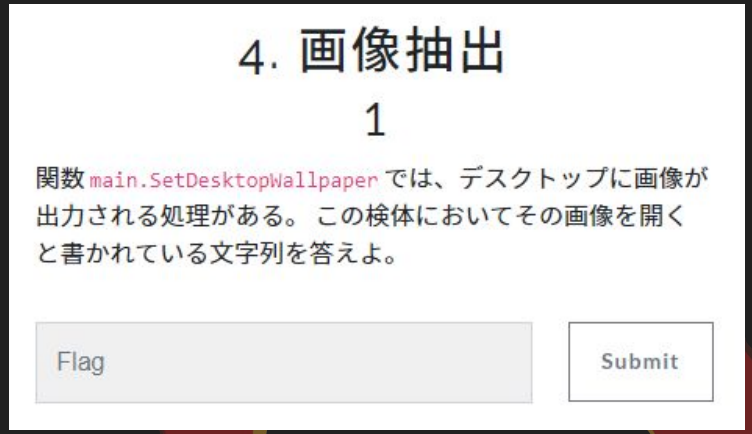
関数内の以下の処理に着目すると、この結論に至ります。

- **user32.dll の読み込み:** Windows API を扱うためのダイナミックリンクライブラリである user32.dll を読み込んでいます。
- **SystemParametersInfoW 関数の呼び出し:** この関数は、システムのパラメータを設定するために使用されます。引数によって、様々な設定が可能です。
- **引数 param_1 と param_2:** 関数に渡されるこれらの引数は、新しい壁紙のパスを指定していると考えられます。
- **SystemParametersInfoW 関数の第2引数:** この引数は、設定するパラメータの種類を指定します。今回の場合、SPI_SETDESKWALLPAPER (0x14) が設定されていることから、デスクトップの壁紙を変更する意図が明確です。

3-6. 関数読解

The Answer is:
デスクトップの壁紙を変更する

4. 画像抽出


4. 画像抽出
1
関数 main.SetDesktopWallpaper では、デスクトップに画像が出力される処理がある。 この検体においてその画像を開くと書かれている文字列を答えよ。
Flag
Submit

4. 画像抽出

関数main.SetDesktopWallpaperの中で、以下の関数が呼び出されている。

- main.LoadDiskImage
  - 画像を読み込む関数はmain.LoadDiskImage
- main.ChangeDesktopWallPaper(challenge3-6)

```
7  void main::main.SetDesktopWallpaper(void)
8
9  {
10   int in_FS_OFFSET;
11   undefined4 in_stack_fffffff8;
12   undefined4 in_stack_fffffffc;
13
14                     /* /root/kuiper-20-11/windowsbuild.go:895 */
15   while (&stack0x00000000 <= *(undefined **)(**(int **)(in_FS_OFFSET + 0x14) + 8)) {
16                     /* /root/kuiper-20-11/windowsbuild.go:895 */
17     in_stack_fffffffc = 0x54be3d;
18     runtime::runtime.morestack_noctxt();
19   }
20                     /* /root/kuiper-20-11/windowsbuild.go:899 */
21   main.LoadDiskImage();
22                     /* /root/kuiper-20-11/windowsbuild.go:900 */
23   runtime::runtime.concatstring3
24             (0,DAT_00689c68,DAT_00689c6c,"\\",1,gostr_image.jpg.str,gostr_image.jpg.len,
25              in_stack_fffffff8,in_stack_fffffffc);
26   main.ChangeDesktopWallPaper(in_stack_fffffff8,in_stack_fffffffc);
27                     /* /root/kuiper-20-11/windowsbuild.go:901 */
28   return;
29 }
```

4. 画像抽出

- main.LoadDiskImageを確認する
- main.DecodeHexStringが呼び出される
- 16進数の文字列gostr_ffd8ffe000104a46494600010101006O.strをバイトデータに変換後、os.WriteFileで書き出すことで画像ファイルを生成している

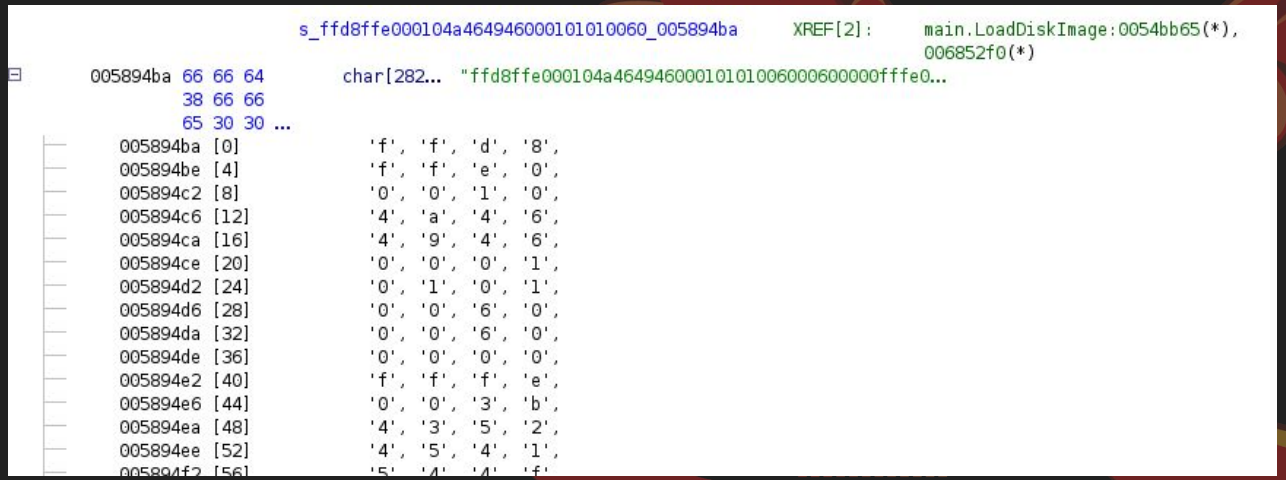
```
28  main.DecodeHexString
29            (gostr_ffd8ffe000104a46494600010101006O.str,gostr_ffd8ffe000104a46494600010101006O.len,
30             in_stack_ffffffd0,in_stack_ffffffd4);
31  runtime::runtime.stringtoslicebyte
32            (0,in_stack_ffffffd0,in_stack_ffffffd4,in_stack_ffffffd4,in_stack_ffffffd8,
33             in_stack_ffffffdc);
34  iVar1 = gostr_image.jpg.len;
35  runtime::runtime.concatstring3
36            (0,DAT_00689c68,DAT_00689c6c,"\\",1,gostr_image.jpg.str,gostr_image.jpg.len,
37             in_stack_fffffffe4,in_stack_fffffffe8);
38                    /* /usr/lib/go-1.18/src/io/ioutil/ioutil.go:46 */
39  os::os.WriteFile(in_stack_fffffffe4,in_stack_fffffffe8,in_stack_ffffffd4,in_stack_ffffffd8,
40                   in_stack_ffffffdc,0x1a4,iVar1,in_stack_fffffffe4);
```

4. 画像抽出

gostr_ffd8ffe000104a4649460001010100 60の内容は
s_ffd8ffe000104a4649460001010100 60_005894baで定義されたアドレスに格納される

4. 画像抽出

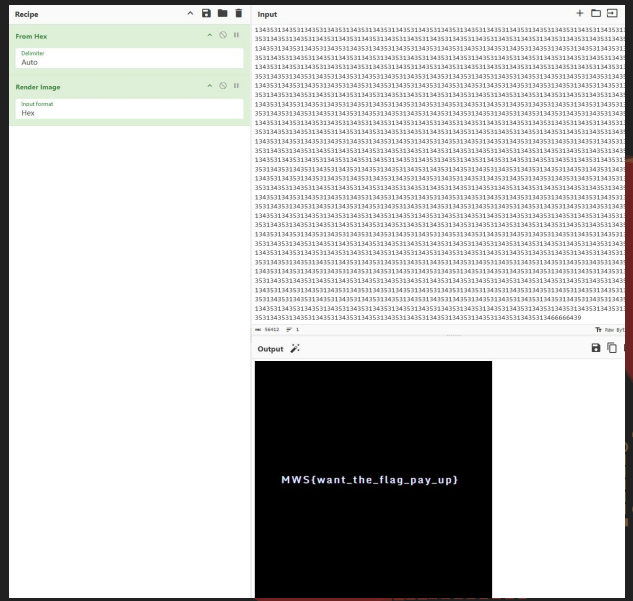
- 005894ba-00592e6に格納されている0x6E2Eh(28206)バイトの16進数列が画像ファイルの元データとなる
  - ※ffd8から始まりffd9で終わる形式はjpegフォーマット



4. 画像抽出

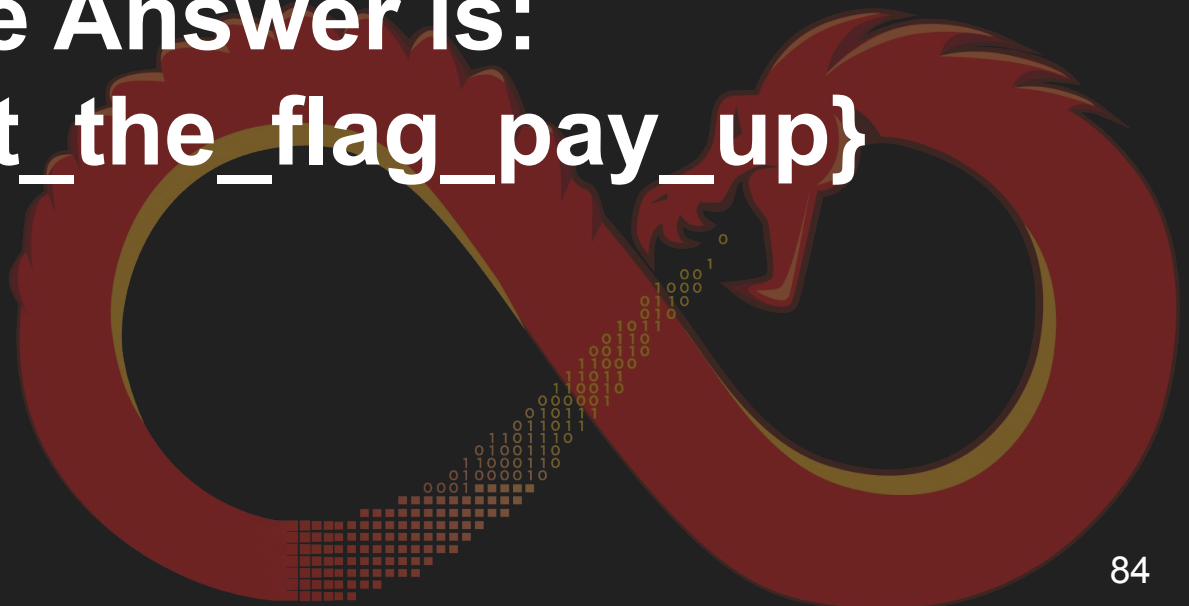
データをコピーし、画像に変換すればフラグとなる画像が出力

https://gchq.github.io/CyberChef/#recipe=From_Hex('Auto')Render_Image('Hex')



4. 画像抽出

The Answer is:
MWS{want_the_flag_pay_up}



5. 暗号鍵

5. 暗号鍵

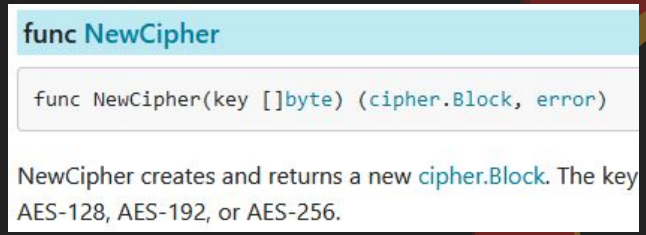
2

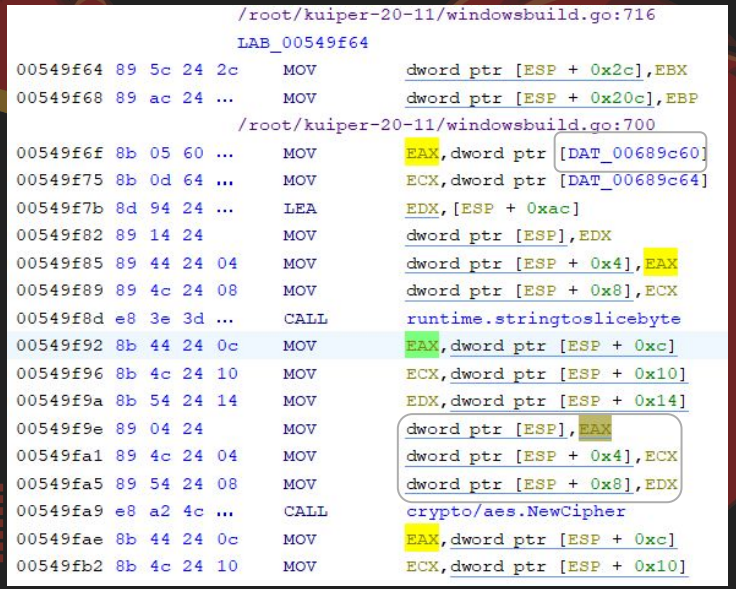
このマルウェアはAESの鍵を生成し、ファイルを暗号化している。生成されたAESの鍵はグローバル変数に格納されている。 生成された鍵をグローバル変数に格納する関数の先頭アドレスを答えよ。

例えば、アドレスが 0x0100ABCD の場合は `0100abcd` と回答すること。

5. 暗号鍵

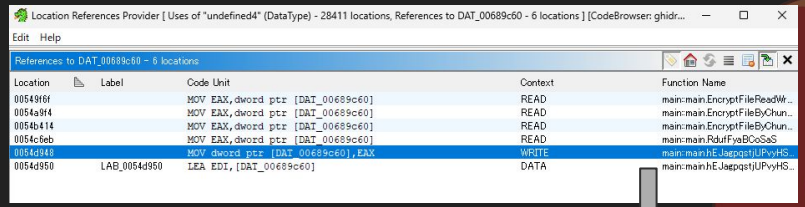
- aes.NewCipherを調べるとmain.EncryptFileReadWriteが見つかる
  - 00689c60の文字列をバイト列に変換
  - バイト列をcrypto/aes.NewCipherの第1引数に指定
    - https://pkg.go.dev/crypto/aes#NewCipher

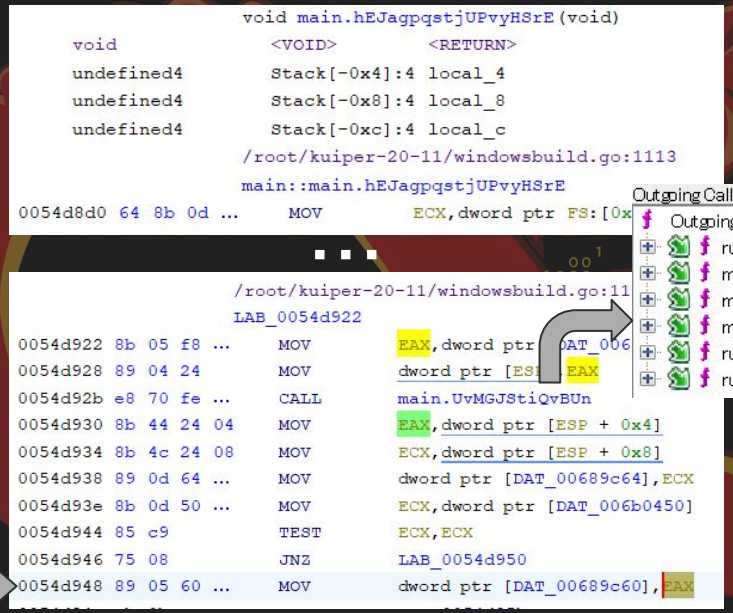




5. 暗号鍵

- 00689c60を辿る
  - main.UvMGJStinOvBUnの戻り値をmain.hEJagpqstjUPvyHSrE(0054d8d0)内で格納
    - 乱数生成している





5. 暗号鍵

The Answer is: 0054d8d0

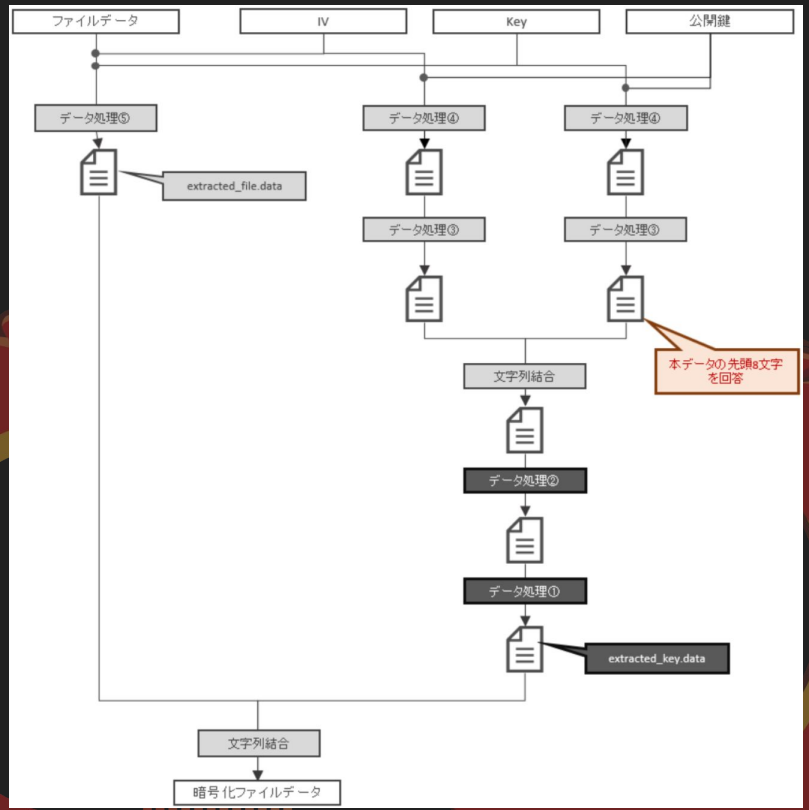
6-1. データ復号

6-1. データ復号

2

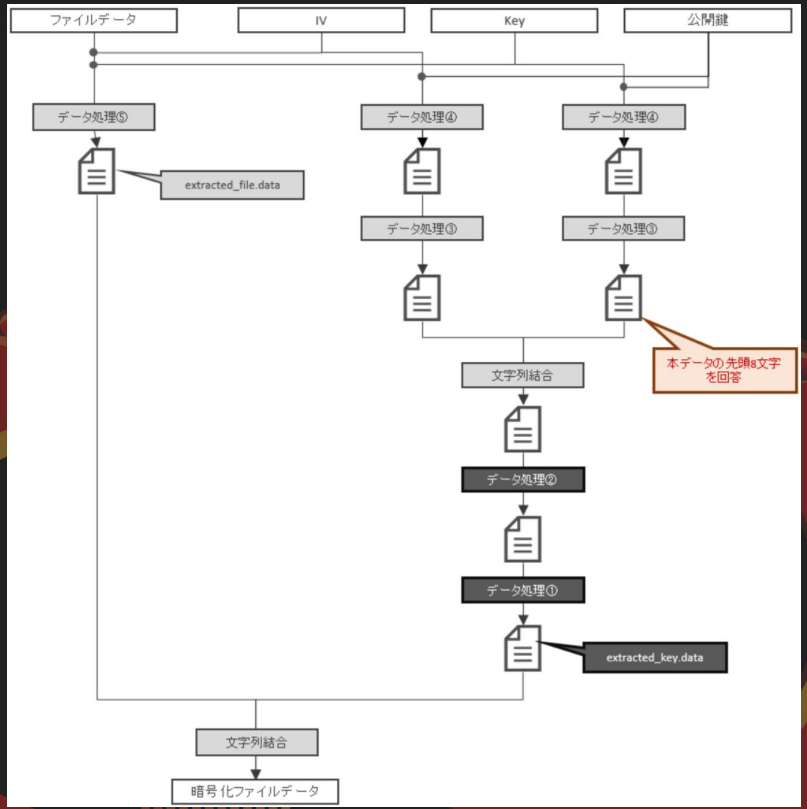
対象ファイル（flag.txt）が暗号化される際に、下記の図のようにIV/Keyを用いてファイルデータを暗号化します。またIV/Keyもデータ処理が施され、ファイルの末尾に付与されます。

関数main.RdufFyaBCoSaSの動作を確認し、データ処理①/データ処理②で何が行われているかを解析し、添付のextracted_key.dataを用いて赤吹き出しで示されたデータの先頭8文字を回答してください。



6-1. データ復号

データ処理①と②を関数main.RdufFyaBCoSaSを確認して特定する

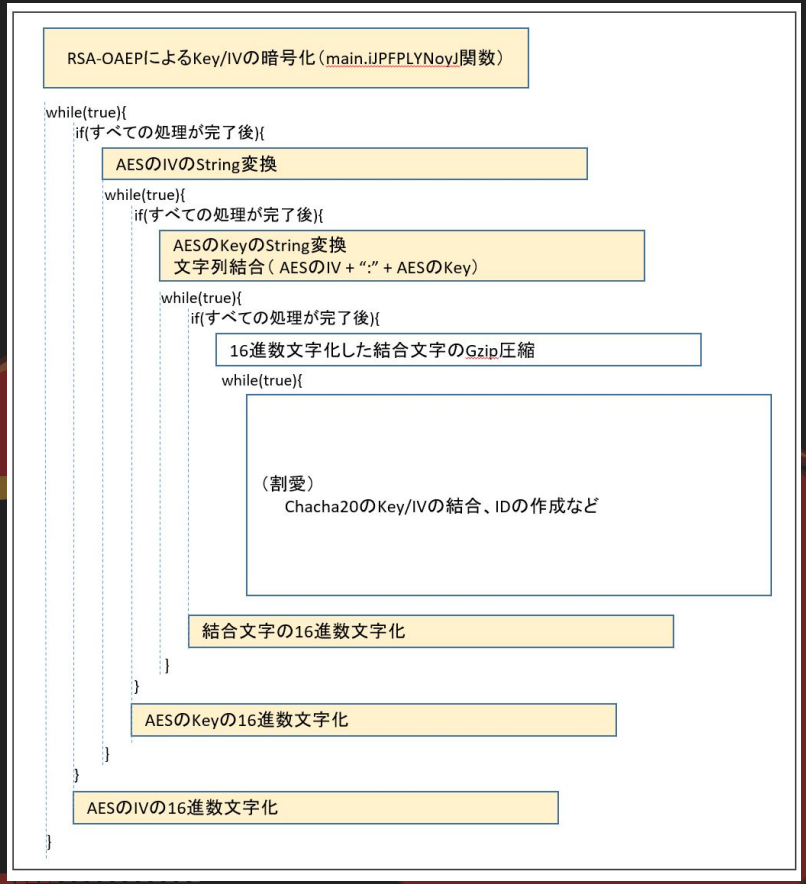


6-1. データ復号

関数5の概要は右の通り

・RSA-OAEPによるkey/IVの暗号化
・AESのIVの16進数文字化
・AESのKeyの16進数文字化
・文字列結合
・結合した文字列の16進数化
・Gzip圧縮

関数main.RdufFyaBCoSaS



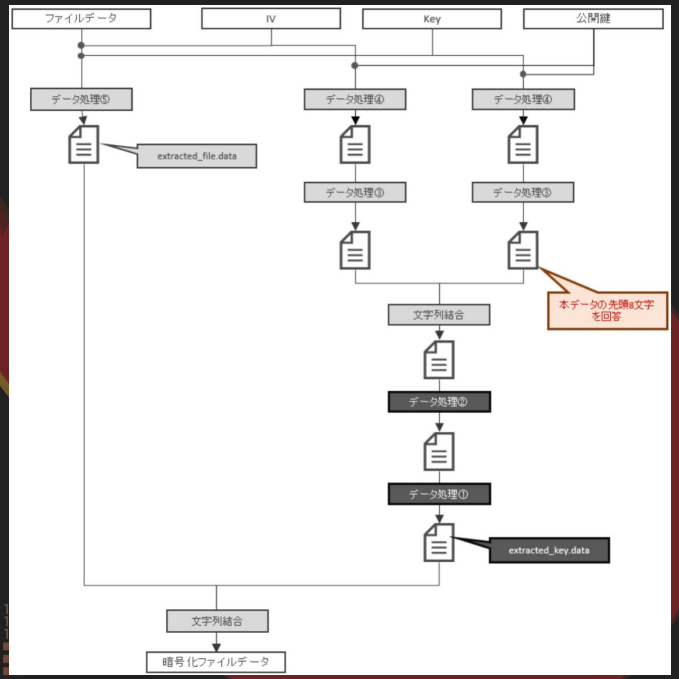
6-1. データ復号

- データ処理①
  - extracted_key.dataの先頭を見ると”1E 8B 09 00”となっている
  - gzip形式

extracted_key.data

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	1F	8B	08	00	00	00	00	00	FF	44	97	8D	91	B4	27		.‹.. .....ÿD—.‘´'
00000010	0C	83	5B	02	04	E2	A5	9C	F4	DF	44	C6	8F	BC	F7	65	.ƒ[..â¥œôßDÆ.¼÷e
00000020	26	97	EC	2E	3F	B6	2C	C9	46	5B	4F	CB	B2	F4	E9	68	&—ì.?¶,ÉF[OË²ôéh
00000030	DB	92	8E	AD	67	69	58	3A	3A	7A	FA	F4	64	5B	CB	C7	Û’Ž.giX::zúôd[ËÇ
00000040	47	F2	D4	F0	B6	BC	BD	55	7F	97	97	86	6B	C5	75	AD	GòÔð¶¼½U.——†kÅu.
00000050	1F	7A	3E	BA	9E	5E	FA	64	2F	4B	53	CB	66	FF	F5	92	.z>ºž^úd/KSËfÿõ’
00000060	B5	BD	74	39	73	69	EA	B3	BD	B4	F4	58	57	67	1F	73	µ½t9siê³½´ôXWg.s
00000070	A5	B7	6E	DD	AE	A5	A3	A1	95	A8	BC	B5	B4	F5	55	FC	¥·nÝ®¥£¡•¨¼µ´õUü
00000080	AE	78	36	27	1E	4F	2D	89	8C	B6	96	67	E7	34	34	2B	®x6'.O-‰Œ¶–gç44+

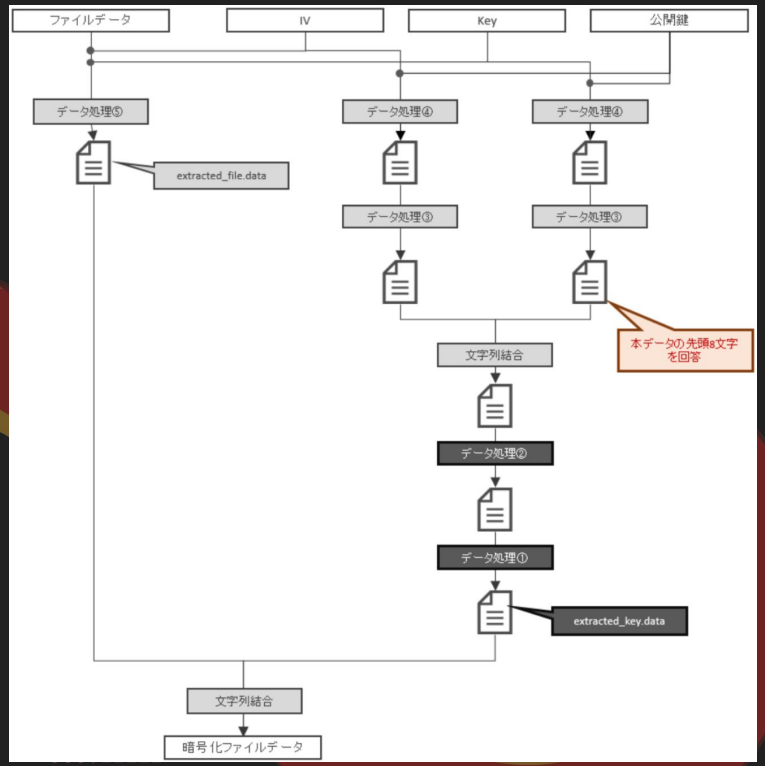
```
277                   iVar9 = 0;
278                   while( true ) {
279                     if (iVar17 <= iVar9) {
280                       runtime::runtime.slicebytetostring(local_1b4,pcVar14,uVar2,pcVar14,iVar15)
281                       ;
282                       main.GzipCompress(pcVar14,iVar15,uVar2,pcVar14,iVar15);
283                       runtime::runtime.slicebytetostring(0,uVar2,pcVar14,pcVar14,iVar15);
284                       DAT_00689c54 = iVar15;
285                       pcVar11 = pcVar14;
286                       iVar9 = DAT_00689c54;
287                       if (DAT_006b0450 != 0) {
288                         runtime::runtime.gcWriteBarrier();
289                         pcVar11 = DAT_00689c50;
```



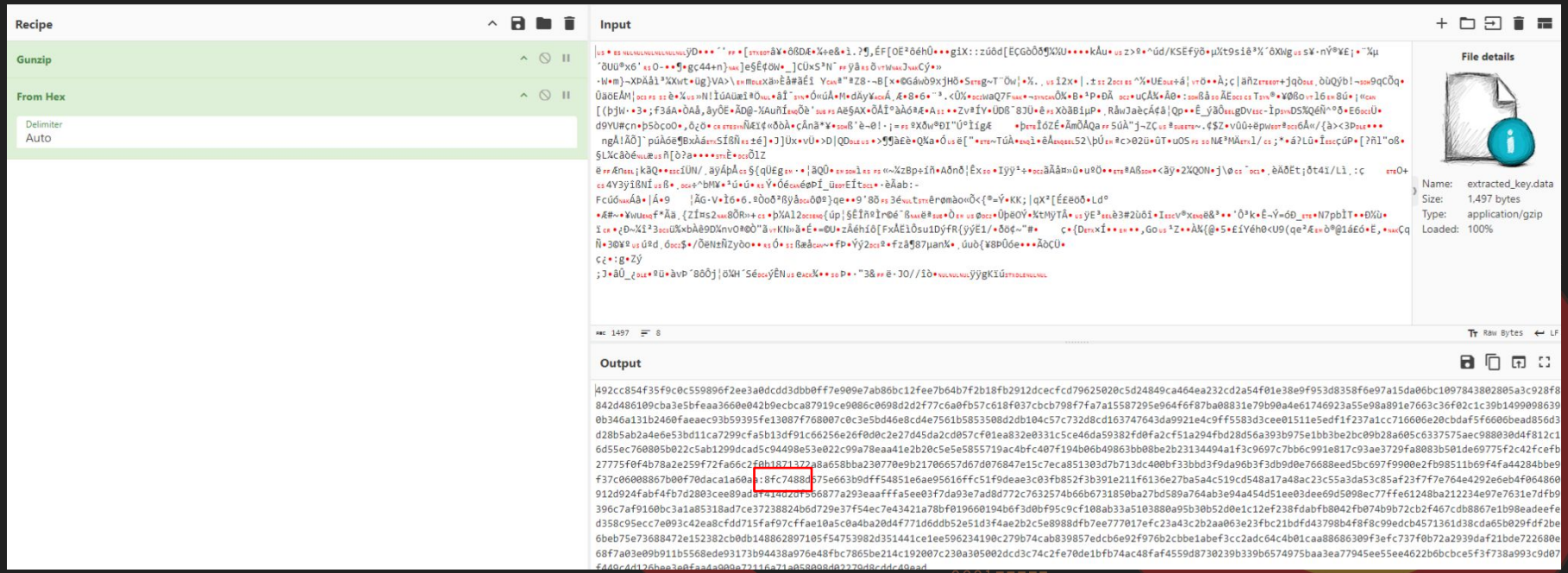
6-1. データ復号

- データ処理②
  - slicebytetostringによる16進数文字化
- concatstringで変換したIVと":"とKEYを結合

```
210     iVar9 = 0;
211     while( true ) {
212       if ((int)pcVar14 <= iVar9) {
213         runtime::runtime.slicebytetostring(local_b4,pcVar13,uVar2,pcVar13,iVar15);
214         pcVar6 = gos_:_57b07b;
215         iVar16 = 1;
216         runtime::runtime.concatstring3
217                   (local_f4,local_6c,in_stack_fffffd94,gos_:_57b07b,1,pcVar13,iVar15,
218                    in_stack_fffffda0,in_stack_fffffda4);
219         runtime::runtime.stringtoslicebyte
220                   (local_d4,in_stack_fffffda0,in_stack_fffffda4,pcVar6,iVar16,pcVar13);
221         prVar4 = &uint8___runtime._type;
```



6-1. データ復号


Recipe
Gunzip
From Hex
Delimiter
Auto
Input
File details
Name: extracted_key.data
Size: 1,497 bytes
Type: application/gzip
Loaded: 100%
Output

6-1. データ復号

The Answer is:

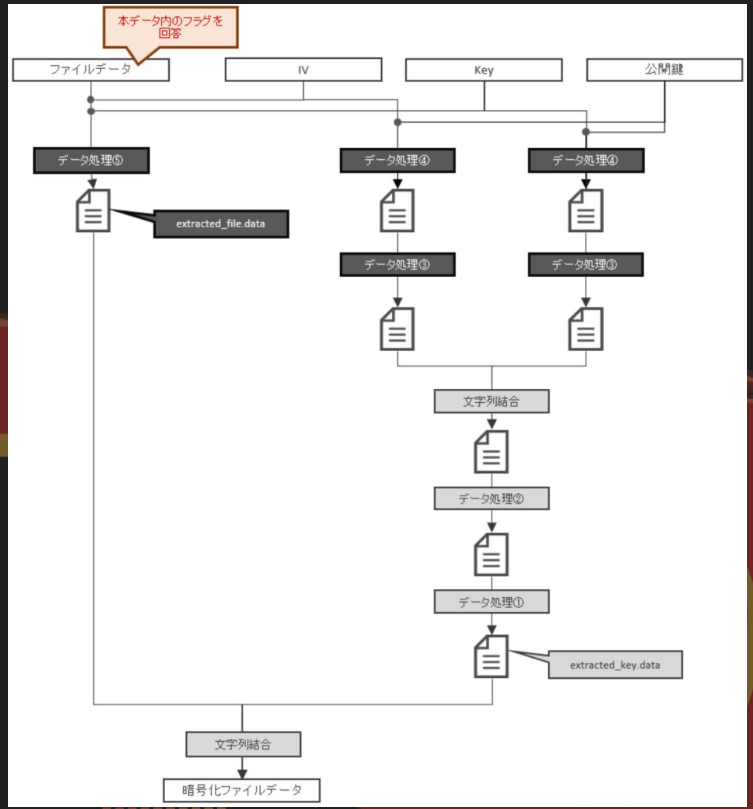
8fc7488d

6-2. データ復号

6-2. データ復号

3

対象ファイル（flag.txt）が暗号化される際に、下記の図の通り関数main.EncryptFileReadWriteにて、データ処理が施されています。この関数の動作を追跡しデータ処理③～⑤を特定した上で、添付のextracted_file.dataを復号化、その中にあるフラグを回答してください。



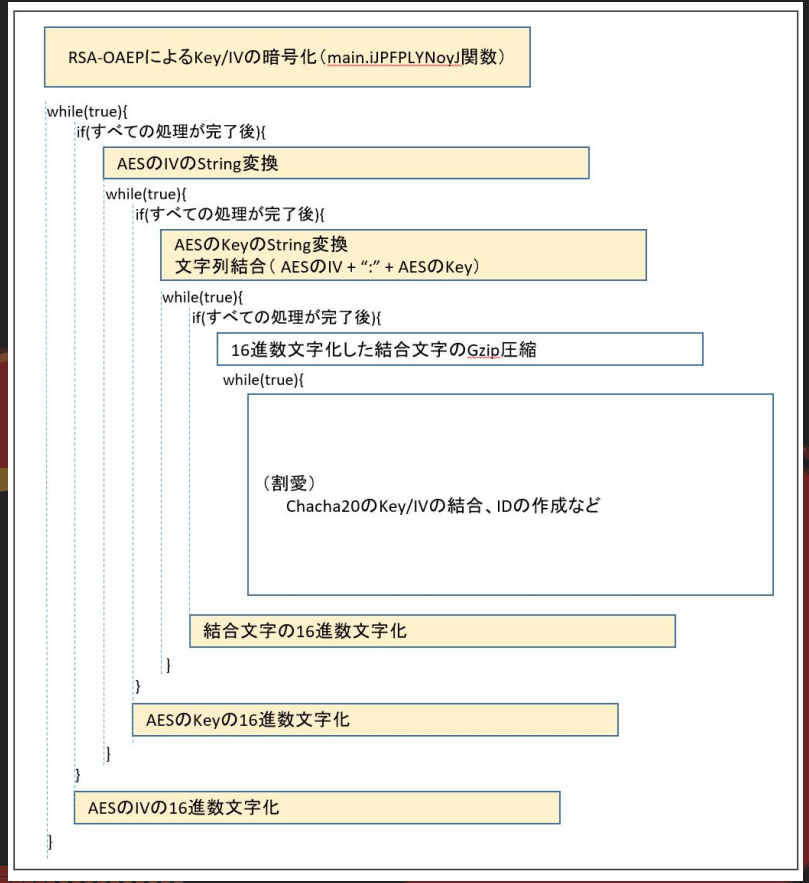
6-2. データ復号

関数main.RdufFyaBCoSaSの概要は右の通り

・RSA-OAEPによるkey/IVの暗号化
・AESのIVの16進数文字化
・AESのKeyの16進数文字化
・文字列結合
・結合した文字列の16進数化
・Gzip圧縮

上記の処理を実装してAESのKeyを復元するスクリプトを作成する

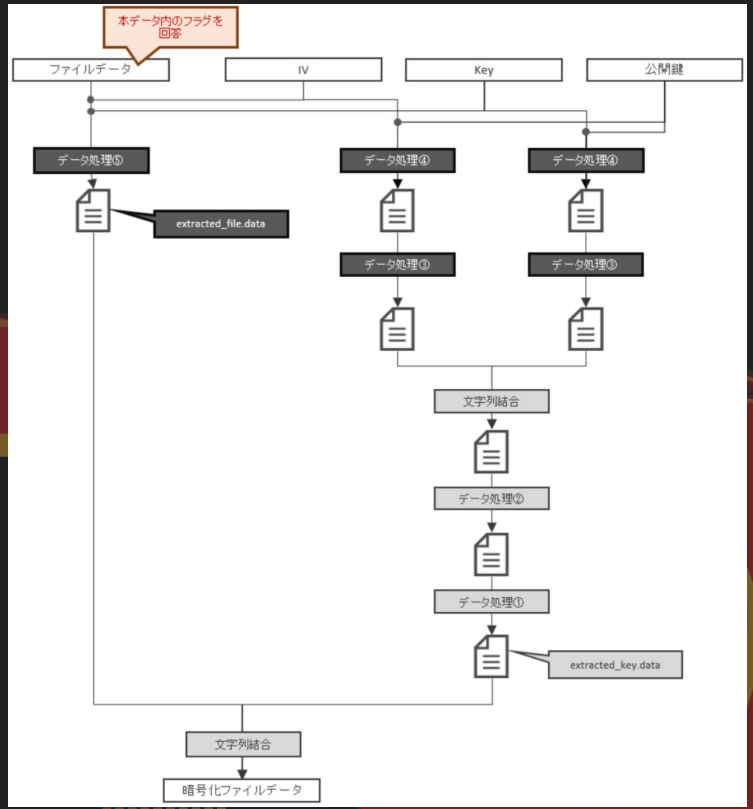
関数main.RdufFyaBCoSaS



6-2. データ復号

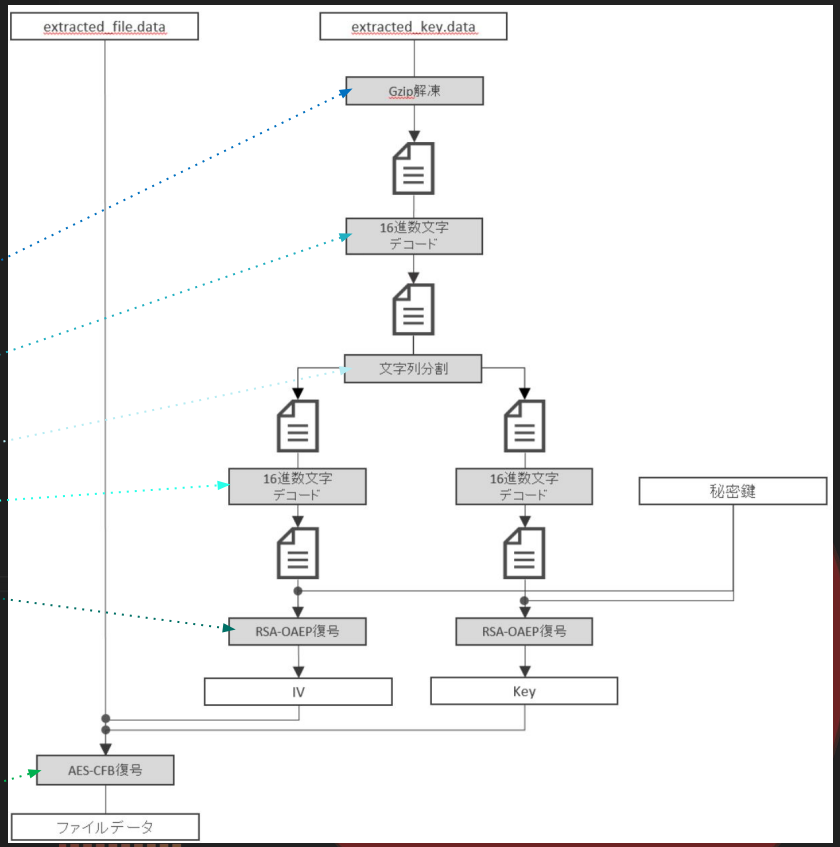
問題文でAESによる暗号化と特定されており、関数解析3-1より、aes.NewCipher/cipher.NewCFB関数を使用していることから、データ処理④はAES-CFBモードということがわかる。

データ復号6-1より、Key/IVが判明しているので、これらを用いて復号処理を実施。



復号スクリプト例（Python3）

```python
import argparse
import binascii
import base64
import re
import os
import gzip
import glob

from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA
from Crypto.Hash import SHA256
from Crypto.Cipher import AES


def decrypt_OAEP(cipher_data, private_key):

    imported_private_key = RSA.importKey(open(private_key).read())
    cipher = PKCS1_OAEP.new(imported_private_key, hashAlgo=SHA256)

    decrypted_data = cipher.decrypt(cipher_data)

    print(f"[decrypt_OAEP] Decrypt: {binascii.hexlify(decrypted_data)}")

    return decrypted_data


def decrypt_AES_CFB(cipher_data, iv, key):
    print("[decrypt_AES_CFB] Decrypt as RSA ...")
    cipher = AES.new(key, AES.MODE_CFB, iv=iv, segment_size=128)
    decrypted_data = cipher.decrypt(cipher_data)

    return decrypted_data


def decode_key(extracted_key_data, private_key):
    print("[decode_key] Decrypt Key/IV ...")

    param = binascii.unhexlify(gzip.decompress(extracted_key_data)).decode("utf8").split(":")

    RSA_iv = decrypt_OAEP(binascii.unhexlify(param[0]), private_key)
    RSA_key = decrypt_OAEP(binascii.unhexlify(param[1]), private_key)

    return RSA_iv,RSA_key


if __name__ == "__main__":

    parser = argparse.ArgumentParser(description='Decode encrypted file by kuiper ransamware.')
    parser.add_argument('-ef','--extracted_file', help='extracted file data')
    parser.add_argument('-ek','--extracted_key', help='extracted key data')
    parser.add_argument('-p','--private_key', help='private key file')

    args = parser.parse_args()

    with open(args.extracted_key, "rb") as f:
        extracted_key_data = f.read()
    iv,key = decode_key(extracted_key_data, args.private_key)

    print(f"key: {key}")

    with open(args.extracted_file, "rb") as f:
        extracted_file_data = f.read()
    decrypted_file_data = decrypt_AES_CFB(extracted_file_data, iv, key)

    print(f"file_data: {decrypted_file_data}")
```



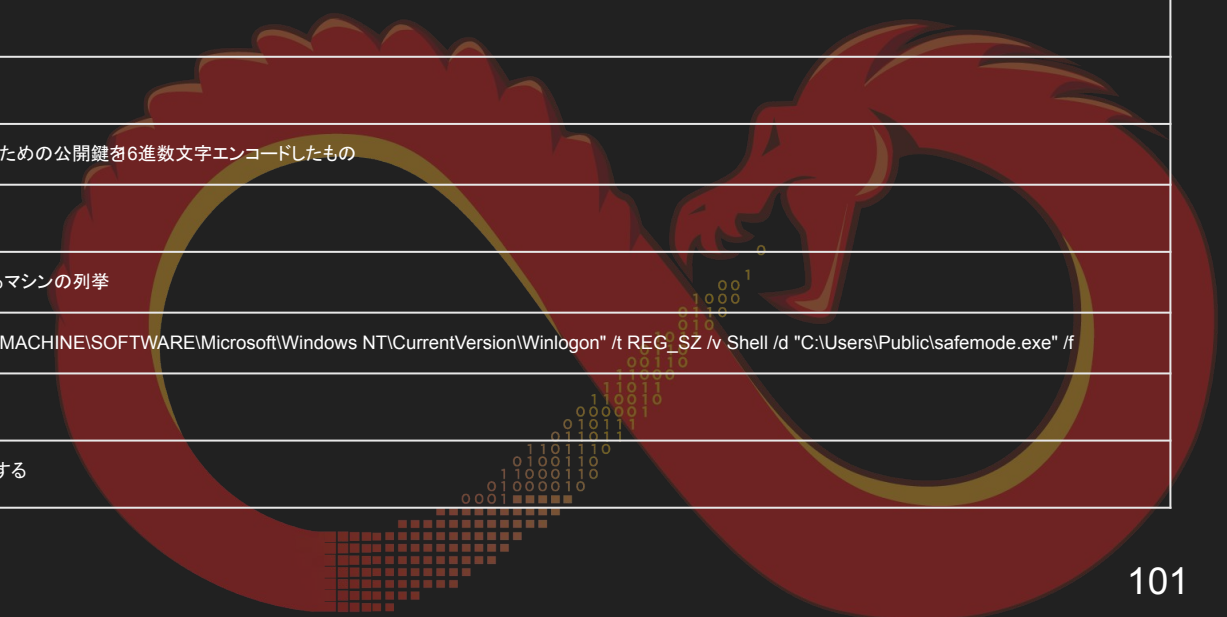
6-2. データ復号

The Answer is:

MWS{fb9acf8aea87f456d4ec629515c8e482ff260c5656ec4fa70ab6720cee83c086}

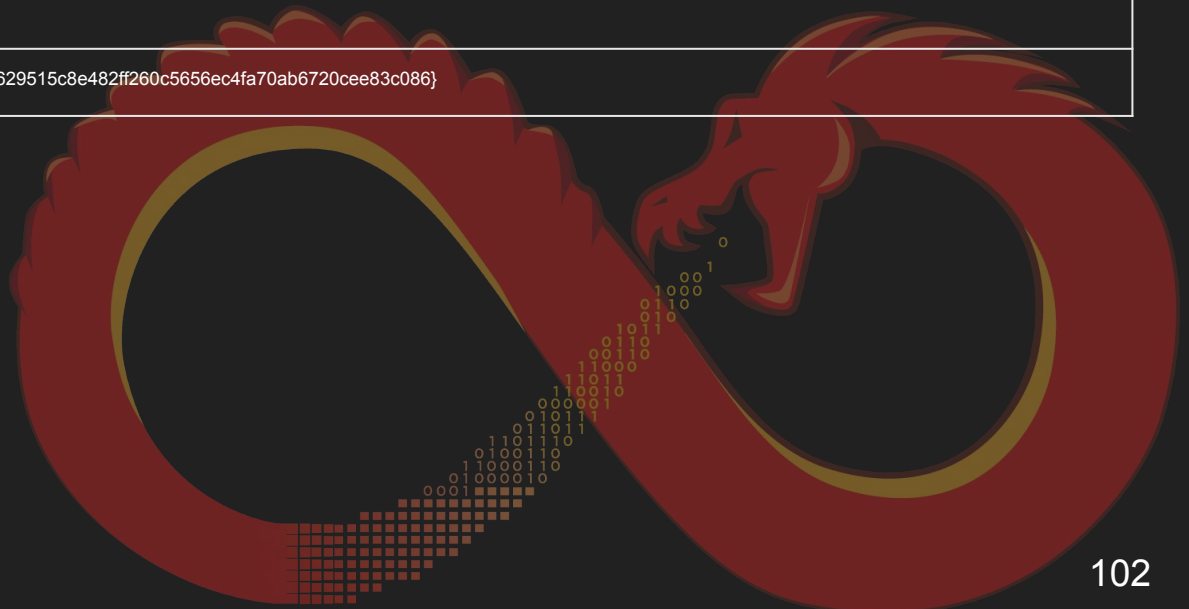
解答まとめ

1-1. Goメタ情報	go1.18.1
1-2. Goメタ情報	005511b0
2-1. オプション	flag
2-2. オプション	0066e1c0
3-1. 関数読解	暗号化鍵、IVを暗号化するための公開鍵を6進数文字エンコードしたもの
3-2. 関数読解	b,c,d
3-3. 関数読解	Local Network に存在するマシンの列挙
3-4. 関数読解	reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /t REG_SZ /v Shell /d "C:\Users\Public\safemode.exe" /f
3-5. 関数読解	596f7572
3-6. 関数読解	デスクトップの壁紙を変更する



解答まとめ

4. 画像抽出	MWS{want_the_flag_pay_up}
5. 暗号鍵	0054d8d0
6-1. データ復号	8fc7488d
6-2. データ復号	MWS{fb9acf8aea87f456d4ec629515c8e482ff260c5656ec4fa70ab6720cee83c086}



ご協力お願いします

- 来年度以降の継続のためにもみなさんのフィードバックが重要です！