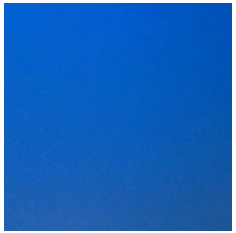
Beginners CTF 2020の Reversing問題を解いてみよう

sei0o @ SECCON Beginners Live!
2020/10/18

今日の予定

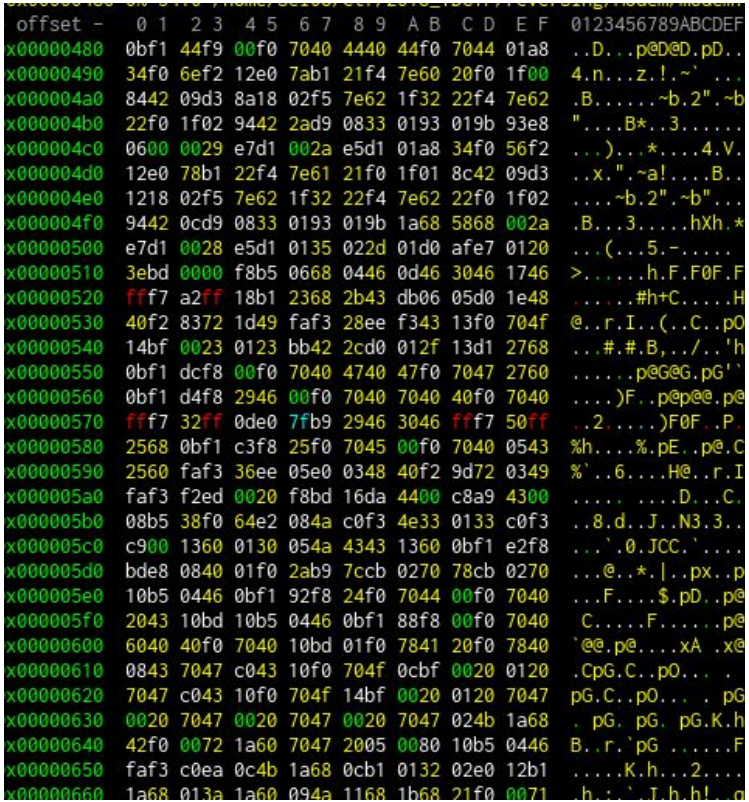
- Reversingとは何か？
- Beginners CTF 2020 でのReversing問題の概観
- mask (Ghidra) の解説
- yakisoba (angr) の解説

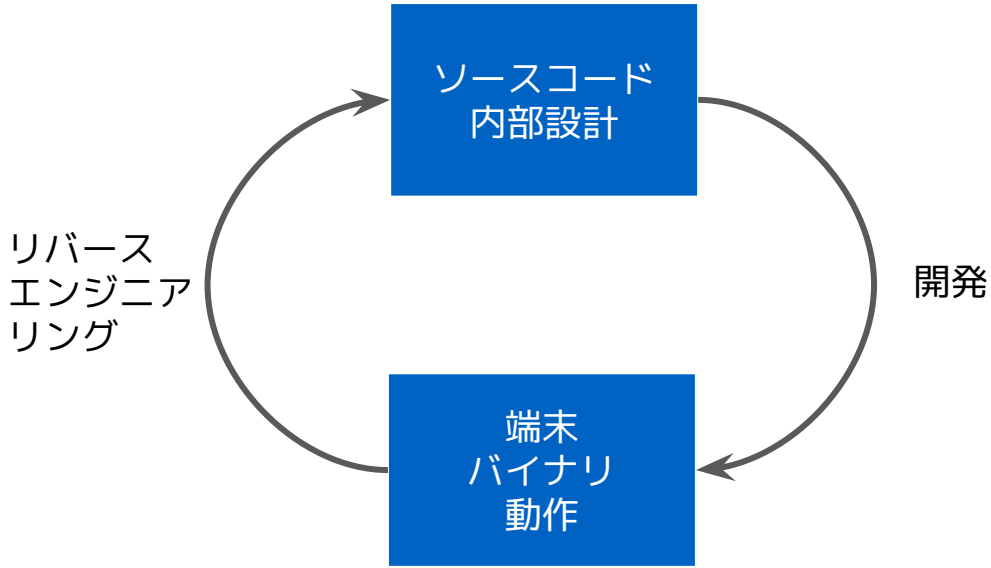
問題リポジトリ: https://github.com/SECCON/Beginners_CTF_2020



http://o0i.es/c4blive.pdf

Reversing, Binary, RE, rev, …





http://o0i.es/c4blive.pdf

Beginners CTF 2020

問題	難易度	正答チーム数 (得点)
mask	Beginner	全1070チームのうち 354チーム（62pts)
yakisoba	Easy	144 (156pts)
ghost	Medium	68 (279pts)
siblangs	Medium	37（363pts)
sneaky	Hard	23 (410pts)

http://o0i.es/c4blive.pdf

maskの解説

```
$ file ./mask
./mask: ELF 64-bit LSB pie executable, x86-64, ... dynamically linked, ...
not stripped

$ ./mask
Usage: ./mask [FLAG]

$ ./mask aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Putting on masks...
aaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaa
Wrong FLAG. Try again.

$ ./mask abcdefghijklmnopqrstuvw
Putting on masks...
a`adede`a`adedepqpqtutu
abc`abchijkhijk`abc`abc
Wrong FLAG. Try again.
```

http://o0i.es/c4blive.pdf

気合で解く

- 2つの文字列からどうにかこうにかして（？）推測
- abcdefghijk…を与えて、対応を見る

以降は気合で解かない（Reversing感のある）方法を紹介します

http://o0i.es/c4blive.pdf

```
$ objdump -M intel -d ./mask
```

```
0000000000001179 <main>:
    1179:       55                      push   rbp
    117a:       48 89 e5                mov    rbp,rsp
    117d:       48 81 ec f0 00 00 00    sub    rsp,0xf0
    1184:       89 bd 1c ff ff ff       mov    DWORD PTR [rbp-0xe4],edi
    118a:       48 89 b5 10 ff ff ff    mov    QWORD PTR [rbp-0xf0],rsi
    1191:       64 48 8b 04 25 28 00    mov    rax,QWORD PTR fs:0x28
    1198:       00 00
    119a:       48 89 45 f8             mov    QWORD PTR [rbp-0x8],rax
    119e:       31 c0                   xor    eax,eax
    11a0:       83 bd 1c ff ff ff 01    cmp    DWORD PTR [rbp-0xe4],0x1
    11a7:       75 16                   jne    11bf <main+0x46>
    11a9:       48 8d 3d 54 0e 00 00    lea    rdi,[rip+0xe54]        # 2004 <_IO_stdin_used+0x4>
    11b0:       e8 8b fe ff ff          call   1040 <puts@plt>
    11b5:       b8 00 00 00 00          mov    eax,0x0
    11ba:       e9 2f 01 00 00          jmp    12ee <main+0x175>
    11bf:       48 8b 85 10 ff ff ff    mov    rax,QWORD PTR [rbp-0xf0]
    11c6:       48 83 c0 08             add    rax,0x8
    11ca:       48 8b 10                mov    rdx,QWORD PTR [rax]
    11cd:       48 8d 85 30 ff ff ff    lea    rax,[rbp-0xd0]
    11d4:       48 89 d6                mov    rsi,rdx
    11d7:       48 89 c7                mov    rdi,rax
    11da:       e8 51 fe ff ff          call   1030 <strcpy@plt>
    11df:       48 8d 85 30 ff ff ff    lea    rax,[rbp-0xd0]
    11e6:       48 89 c7                mov    rdi,rax
    11e9:       e8 62 fe ff ff          call   1050 <strlen@plt>
    11ee:       89 85 2c ff ff ff       mov    DWORD PTR [rbp-0xd4],eax
    11f4:       48 8d 3d 1e 0e 00 00    lea    rdi,[rip+0xe1e]        # 2019 <_IO_stdin_used+0x19>
```

binutilsパッケージが必要です

http://o0i.es/c4blive.pdf

プログラムの解析

- Ghidra = SRE
  - 日本語での記事も出回ってきました
- ここでは逆コンパイラとして使ってみる

http://o0i.es/c4blive.pdf

Decompile: main - (mask)

```
1
2  undefined8 main(int iParm1,long lParm2)
3
4  {
5    int iVar1;
6    size_t sVar2;
7    long in_FS_OFFSET;
8    int local_e0;
9    byte local_d8 [64];
10   byte local_98 [64];
11   byte local_58 [72];
12   long local_10;
13
14   local_10 = *(long *)(in_FS_OFFSET + 0x28);
15   if (iParm1 == 1) {
16     puts("Usage: ./mask [FLAG]");
17   }
18   else {
19     strcpy((char *)local_d8,*(char **)(lParm2 + 8));
20     sVar2 = strlen((char *)local_d8);
21     iVar1 = (int)sVar2;
22     puts("Putting on masks...");
23     local_e0 = 0;
24     while (local_e0 < iVar1) {
25       local_98[(long)local_e0] = local_d8[(long)local_e0] & 0x75;
26       local_58[(long)local_e0] = local_d8[(long)local_e0] & 0xeb;
27       local_e0 = local_e0 + 1;
28     }
29     local_98[(long)iVar1] = 0;
30     local_58[(long)iVar1] = 0;
31     puts((char *)local_98);
32     puts((char *)local_58);
33     iVar1 = strcmp((char *)local_98,"atd4`qdedtUpetepqeUdaaeUeaqau");
34     if ((iVar1 == 0) &&
35        (iVar1 = strcmp((char *)local_58,"c`b bk`kj`KbababcaKbacaKiacki"), iVar1 == 0)) {
36       puts("Correct! Submit your FLAG.");
37     }
38     else {
39       printf("Wrong FLAG. Try again :)");
```

c4blive.pdf

Decompile: main - (mask)

```
1
2  undefined8 main(int iParm1,long lParm2)
3
4  {
5    int iVar1;
6    size_t sVar2;
7    long in_FS_OFFSET;
8    int local_e0;
9    byte local_d8 [64];
10   byte local_98 [64];
11   byte local_58 [72];
12   long local_10;
13
```

- 変数ずらずら
- `int main(int argc, char **argv)` と比較
- `iParm1 = argc, lParm2 = argv`
  - クリック→Lでリネーム

http://o0i.es/c4blive.pdf

- 引数のチェック
- 見たことがある文字列

```
14    local_10 = *(long *)(in_FS_OFFSET + 0x28);
15    if (argc == 1) {
16      puts("Usage: ./mask [FLAG]");
17    }
18    else {
19      strcpy((char *)local_d8,*(char **)(argv + 8)
```

http://o0i.es/c4blive.pdf

```
else {
  strcpy((char *)local_d8,*(char **)(argv + 8));
  sVar2 = strlen((char *)local_d8);
  iVar1 = (int)sVar2;
  puts("Putting on masks...");
  local_e0 = 0;
  while (local_e0 < iVar1) {
    local_98[(long)local_e0] = local_d8[(long)local_e0] & 0x75;
    local_58[(long)local_e0] = local_d8[(long)local_e0] & 0xeb;
    local_e0 = local_e0 + 1;
  }
  local_98[(long)iVar1] = 0;
  local_58[(long)iVar1] = 0;
  puts((char *)local_98);
  puts((char *)local_58);
```

argv[1]

- 見づらい
- 配列要素へのアクセスはアドレスの足し算

- local_d8 → argstr
- iVar1 → len

http://o0i.es/c4blive.pdf

```
else {
  strcpy((char *)argstr,*(char **)(argv + 8));
  sVar2 = strlen((char *)argstr);
  len = (int)sVar2;
  puts("Putting on masks...");
  local_e0 = 0;
  while (local_e0 < len) {
    local_98[(long)local_e0] = argstr[(long)local_e0] & 0x75;
    local_58[(long)local_e0] = argstr[(long)local_e0] & 0xeb;
    local_e0 = local_e0 + 1;
  }
  local_98[(long)len] = 0;
  local_58[(long)len] = 0;
  puts((char *)local_98);
  puts((char *)local_58);
```

- local_e0 → i （ループ変数）
- local_98,58 → str1, str2

```
$ ./mask abcdefghijklmnopqrstuvw
Putting on masks...
a`adede`a`adedepqpqtutu ← str1
abc`abchijkhijk`abc`abc ← str2
Wrong FLAG. Try again.
```

http://o0i.es/c4blive.pdf

```
strcpy((char *)argstr,*(char **)(argv + 8));
sVar3 = strlen((char *)argstr);
len = (int)sVar3;
puts("Putting on masks...");
i = 0;
while (i < len) {
  str1[(long)i] = argstr[(long)i] & 0x75;
  str2[(long)i] = argstr[(long)i] & 0xeb;
  i = i + 1;
}
str1[(long)len] = 0;
str2[(long)len] = 0;
puts((char *)str1);
puts((char *)str2);
```

- ANDを取っている（ビット演算）
  - ‘A’ = 0x41
- 0xは16進数であることを示す

```
$ ./mask abcdefghijklmnopqrstuvw
Putting on masks...
a`adede`a`adedepqpqtutu ← str1
abc`abchijkhijk`abc`abc ← str2
Wrong FLAG. Try again.
```

http://o0i.es/c4blive.pdf

```
puts((char *)str2);
iVar2 = strcmp((char *)str1,"atd4`qdedtUpetepqeUdaaeUeaqau");
if ((iVar2 == 0) && (iVar2 = strcmp((char *)str2,"c`b bk`kj`KbababcaKbacaKiacki"), iVar2 == 0))
{
  puts("Correct! Submit your FLAG.");
}
else {
  printf("Wrong FLAG. Try again :)");
}
```

- 引数の文字列が等しいとき、`strcmp`は`0`を返す
- `if (strcmp(str1, …) && strcmp(str2, …) )` と同じこと

こうなってほしい→

```
$ ./mask ctf4b{なにか}
Putting on masks…
atd4`qdedtUpetepqeUdaaeUeaqau ← str1
c`b bk`kj`KbababcaKbacaKiacki ← str2
Correct! Submit your FLAG.
```

http://o0i.es/c4blive.pdf

ここまでの整理

- `str1[i] = argstr[i] & 0x75 = FLAG[i] & 0x75`
- `str2[i] = argstr[i] & 0xeb = FLAG[i] & 0xeb`
- 正しい`FLAG`を与えると、`str1,2`は埋め込まれた文字列に一致する
- `FLAG[i]`を求めたい

こうなってほしい→

```
$ ./mask ctf4b{なにか} ← argstr == FLAG
Putting on masks…
atd4`qdedtUpetepqeUdaaeUeaqau ← str1
c`b bk`kj`KbababcaKbacaKiacki ← str2
Correct! Submit your FLAG.
```

http://o0i.es/c4blive.pdf

```
str1[(long)i] = argstr[(long)i] & 0x75;
str2[(long)i] = argstr[(long)i] & 0xeb;
```

- 0x75 = 0b01110101 (2進数)
- 0xeb = 0b11101011

みんな大好き AAAAAAAA...
= 0x4141414141...

例えば "c" (0x63 = 0b01100011) とANDをとると

```
    0x63 = 01100011          0x63 = 01100011
AND 0x75 = 01110101      AND 0xeb = 11101011
           01100001 → str1          01100011  →  str2
```

しかし、今は "c" にあたる文字がわからない ⇒ ANDを逆算したい！

- FLAG[i] & 0x75 == str1[i]
- FLAG[i] & 0xeb == str2[i]
- わからん & わかる(定数) == わかる(出力)

http://o0i.es/c4blive.pdf

ANDの性質

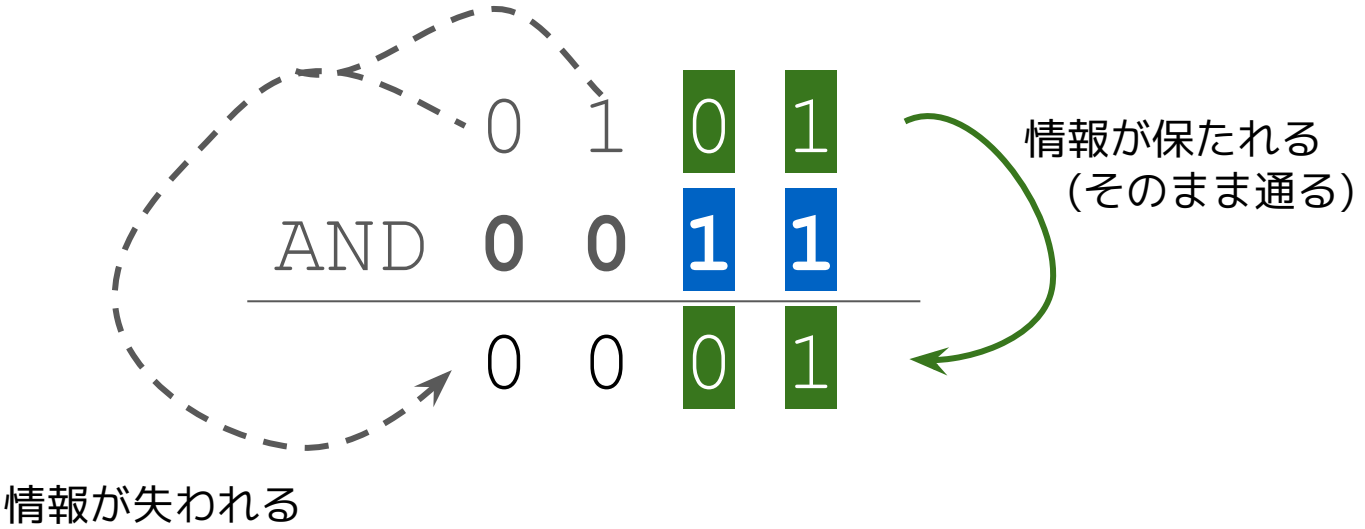
上 AND 下 :「上=1かつ下=1」のとき1

```
      0 1 0 1
AND   0 0 1 1
-------------
      0 0 0 1
```

- ビット演算（OR, AND, XOR）の知識はCTFに限らず一般のプログラミングにおいても重要
- if文で使うAND, OR (`&&`, `||`)とは別物です

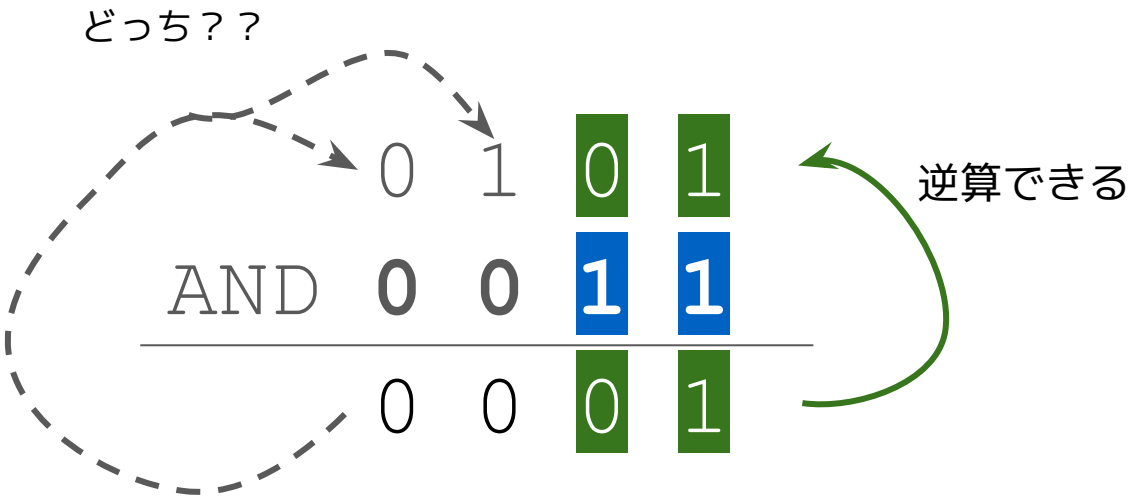
http://o0i.es/c4blive.pdf

マスク演算:「上=1かつ下=1」以外に、下側をマスクと見る



http://o0i.es/c4blive.pdf

逆にいえば…



http://o0i.es/c4blive.pdf

未知の数 (0b????????) に対してANDを取ると…

左と右の逆算できた部分を並べて...

もとのbitを、黄色と緑の少なくとも一方が知っている→もとの数がわかる

http://o0i.es/c4blive.pdf

例えば “c” (0x63 = 0b01100011) とANDをとると

左と右の逆算できた部分を並べて...

?110?0?1
011?0?11
01100011 = “c” が得られた

http://o0i.es/c4blive.pdf

もうちょっと変形

```
 ?110?0?1
 011?0?11
--------
 01100011 = "c"
```

このとき、?に0を入れてみると、

```
  01100001
( ) 01100011
--------
  01100011 = "c"
```

http://o0i.es/c4blive.pdf

「0とORすると変化しない」性質を利用して、

、

```
  01100001
OR 01100011
----------
  01100011 = "c"
```

情報が保たれる

```
  0 1 0 1
OR 0 0 1 1
---------
  0 1 1 1
```

http://o0i.es/c4blive.pdf

解法まとめ

```
  01100001  ← str1[i] … 埋め込まれた値
OR 01100011  ← str2[i]
-----------
  01100011  → FLAG[i]
```

- これを各文字に対して行う（str1,2の長さはFLAGの長さ）

こうなってほしい→

```
$ ./mask ctf4b{なにか} ← FLAG
Putting on masks…
atd4`qdedtUpetepqeUdaaeUeaqau ← str1
c`b bk`kj`KbababcaKbacaKiacki ← str2
Correct! Submit your FLAG.
```

http://o0i.es/c4blive.pdf

解答例

```
str1 = "atd4`qdedtUpetepqeUdaaeUeaqau"
str2 = "c`b bk`kj`KbababcaKbacaKiacki"

flag = ""
for i in range(len(str1)):
    flag += chr(ord(str1[i]) | ord(str2[i]))

print(flag)
```

```
str_a = "atd4`qdedtUpetepqeUdaaeUeaqau"
str_b = "c`b bk`kj`KbababcaKbacaKiacki"

flag = str_a.chars.zip(str_b.chars).map do |(a, b)|
  (a.ord | b.ord).chr
end.join

puts flag
```

http://o0i.es/c4blive.pdf

なにこれ？

```
if (lVar1 == *(long *)(in_FS_OFFSET + 0x28)) {
  return 0;
}
                    /* WARNING: Subroutine does not return */
__stack_chk_fail();
```

- SSP (Canary) に関連するコード
- pwnをするとわかります
  - もちろん、もとのソースコードにこんな記述はない

http://o0i.es/c4blive.pdf

yakisoba

```
$ file ./yakisoba
./yakisoba: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV),
dynamically linked, ... stripped

$  ./yakisoba
FLAG: ctf4b{aaaaaaaaaaaaaaa
Wrong!
```

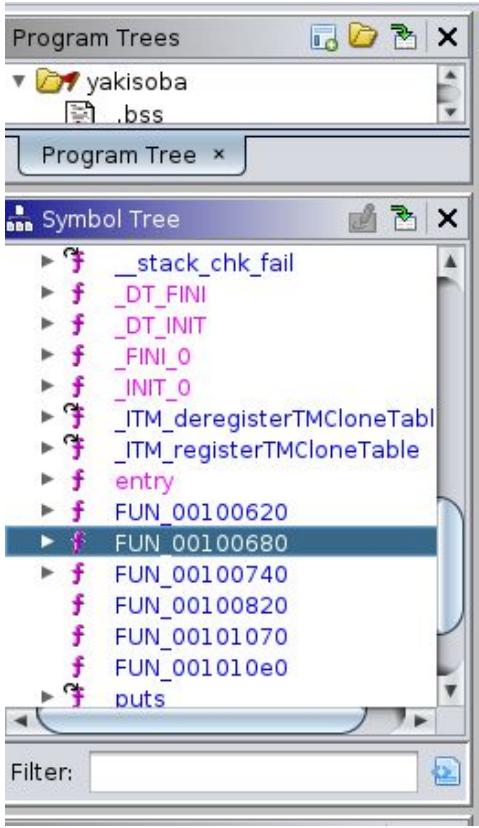
```
$ file ./mask
./mask: ... not stripped

$ nm ./mask
... 0000000000001179 T main
                     U puts@@GLIBC_2.2.5 ...
$ nm ./yakisoba
nm: yakisoba: no symbols
```

←シンボルの確認

http://o0i.es/c4blive.pdf

mainはどこ・・・？



Decompile: FUN_00100680 - (yakisoba)

```
1
2  ulong FUN_00100680(void)
3
4  {
5    int iVar1;
6    uint uVar2;
7    ulong uVar3;
8    long in_FS_OFFSET;
9    undefined auStack72 [40];
10   long local_20;
11
12   uVar3 = 1;
13   local_20 = *(long *)(in_FS_OFFSET + 0x28);
14   __printf_chk(1,"FLAG: ");
15   iVar1 = __isoc99_scanf(&DAT_001010fb,auStack72);
16   if (iVar1 != 0) {
17     uVar2 = FUN_00100820(auStack72);
18     uVar3 = (ulong)uVar2;
19     if (uVar2 == 0) {
20       puts("Correct!");
21     }
22     else {
23       uVar3 = 0;
24       puts("Wrong!");
25     }
26   }
```

見るべき関数は多くないので、
一個一個見て判断する

`__libc_start_main` の呼び出しから判断してもよい

```
2  void entry(undefined8 uParm1,undef
3
4  {
5    undefined8 in_stack_00000000;
6    undefined auStack8 [8];
7
8    __libc_start_main(FUN_00100680,i
9                      ,auStack8);
10   do {
```

http://o0i.es/c4blive.pdf

```
1
2  ulong FUN_00100680(void)
3
4  {
5    int iVar1;
6    uint uVar2;
7    ulong uVar3;
8    long in_FS_OFFSET;
9    undefined auStack72 [40];
10   long local_20;
11
12   uVar3 = 1;
13   local_20 = *(long *)(in_FS_OFFSET + 0x28);
14   __printf_chk(1,"FLAG: ");
15   iVar1 = __isoc99_scanf(&DAT_001010fb,auStack72);
16   if (iVar1 != 0) {
17     uVar2 = FUN_00100820(auStack72);
18     uVar3 = (ulong)uVar2;
19     if (uVar2 == 0) {
20       puts("Correct!");
21     }
22     else {
23       uVar3 = 0;
24       puts("Wrong!");
25     }
26   }
27   if (local_20 == *(long *)(in_FS_OFFSET + 0x28)) {
28     return uVar3;
29   }
30                     /* WARNING: Subroutine does not return */
31   __stack_chk_fail();
32 }
33
```

変数の宣言

処理内容

FORTIFY

```
12    uVar3 = 1;
13    local_20 = *(long *)(in_FS_OFFSET + 0x28);
14    __printf_chk(1,"FLAG: ");
15    iVar1 = __isoc99_scanf(&DAT_001010fb,auStack72);
16    if (iVar1 != 0) {
17      uVar2 = FUN_00100820(auStack72);
18      uVar3 = (ulong)uVar2;
19      if (uVar2 == 0) {
20        puts("Correct!");
21      }
22      else {
23        uVar3 = 0;
24        puts("Wrong!");
25      }
26    }
```

Listingより、"%31s"

こうなってほしい→

```
$ ./yakisoba
FLAG: ctf4b{なにか}
Correct!
```

http://o0i.es/c4blive.pdf

分岐が多くて長い

0x100820 (FUN_00100820) を見てみると…

```
589             return 0xe0;
590           }
591           if (bVar1 == (
592             return 0x39;
593           }
594         }
595         return 0xc;
596       }
597     }
598     return uVar2;
599   }
600
```

```
}
if (bVar1 == 0xe0) {
  return 0x87;
}
if (bVar1 != 0x70) {
  return 0xb8;
}
bVar1 = pbParm1[0x13];
if (bVar1 != 0x70) {
  if (bVar1 < 0x71) {
    if (bVar1 == 0x29) {
      return 0xac;
    }
    if (bVar1 != 0x5a) {
      return 0xcc;
    }
    return 0x23;
  }
  if (bVar1 == 0xee) {
    return 0xa0;
  }
  if (bVar1 == 0xf7) {
    return 0x31;
  }
  if (bVar1 != 0x86) {
    return 0xcc;
  }
  return 0xcd;
}
bVar1 = pbParm1[0x14];
if (bVar1 != 0x33) {
  if (b
    if
      return 0x17;
```

http://o0i.es/c4blive.pdf

激おこangr丸٩(ꈍᴗˬᴗ)۶

- `FUN_00100820`を読むのは心が折れる
- angrを使おう
- シンボリック実行 (Symbolic Execution) を通じて探索
  - 大量の条件分岐
- 期待する結果を入れると、それを引き起こす入力を探してくれる

http://o0i.es/c4blive.pdf

解答例 CTFするぞより

```
...import angrとか...

flag = claripy.BVS("flag", 8 * 0x20)
p = angr.Project("../files/yakisoba",
load_options={"auto_load_libs": False})
state = p.factory.entry_state( stdin=flag)
simgr = p.factory.simulation_manager(state)

simgr.explore( find=0x4006d2, avoid=0x4006f7)

try:
    found = simgr.found[0]
    print(found.solver.eval(flag, cast_to=bytes))
except IndexError:
    print("Not Found")
```

1. 入力を定義
   - 標準入力
   - 長さは？使える文字は？

2. 目標を指定して解析を実行
   - どこを通ってほしい？
   - どこを通らないでほしい？

http://o0i.es/c4blive.pdf

```
flag = claripy.BVS("flag", 8 * 0x20)
p = angr.Project("../files/yakisoba",
load_options={"auto_load_libs": False})
state = p.factory.entry_state( stdin=flag)
```

- BVS = Bit Vector Symbol, シンボル名とサイズを指定
- State (実行の状態) はエントリポイントにセット
- stdin = 標準入力にflagをセット

```
0x20 = 32byte
scanf(”%31s", ...);
```
サイズは適当でも動く

http://o0i.es/c4blive.pdf

```
simgr.explore(find=0x4006d2, avoid=0x4006f7)
```

```
12    uVar3 = 1;
13    local_20 = *(long *)(in_FS_OFFSET + 0x28);
14    __printf_chk(1,"FLAG: ");
15    iVar1 = __isoc99_scanf(&DAT_001010fb,auStack72);
16    if (iVar1 != 0) {
17      uVar2 = FUN_00100820(auStack72);
18      uVar3 = (ulong)uVar2;
19      if (uVar2 == 0) {
20        puts("Correct!");
21      }
22      else {
23        uVar3 = 0;
24        puts("Wrong!");
25      }
26    }
```

```
001006d0 75 25           JNZ        LAB_001006f7
001006d2 48 8d 3d        LEA        RDI,[s_Correct!_00101107]
         2e 0a 00 00
001006d9 e8 52 ff        CALL       puts
         ff ff
```

- findに通ってほしいアドレス，avoidに避けてほしいアドレス

```
$ ./yakisoba
FLAG: ctf4b{なにか}
Correct!
```

http://o0i.es/c4blive.pdf

```
$ python3 solve.py

...なんやらかんやら...

b'ctf4b{sp4gh3tt1_r1pp3r1n0}\x00\xdb\xdb\xdb\xdb\x
00'
```

```
$ ./yakisoba
FLAG: ctf4b{sp4gh3tt1_r1pp3r1n0}
Correct!
```

http://o0i.es/c4blive.pdf

ありがとうございました

- Reversingの問題はすべて作問者によるwriteupが公開されています
  - CTFするぞ, sei0opub
- 疑問点はアンケート, @sei0o, sei0o@live.jp までお願いします

http://o0i.es/c4blive.pdf