

# Writeup: Not Executed Branch

HACK THE PODS CTF 第2回

作問者：工藤信一郎

解説者：嶋田壮志

# 1. 説明文

タチコマからデータが送られてきた。

何やらデータは中にあるようだが、暗号化されている上に分岐に関するバグがありうまく出力できないらしい。

解析開始だイシカワ!!!

## 2. 共有されるファイル

- Not\_executed\_branch.bin
  - フラグが格納されている問題ファイル

### 3. Writeup

まずは実行してみる

```
babyblue0@babyblue0-ctf ~/Desktop/httpctf
% ./Not_executed_branch.bin
ぼくらはみんな生きている~♪
```

引数を指定してみる

```
babyblue0@babyblue0-ctf ~/Desktop/httpctf/Not_
% ./Not_executed_branch.bin AAAA BBBB CCCC
ぼくらはみんな生きている~♪
```

stringsコマンドも実行してみる

(怪しい文字列は無いため、結果省略)

### 3. Writeup

これらの結果から、このファイルについて以下のことが考えられる  
(動作から、どのようなプログラムが組まれてるかを考えるはとても大事。  
解析のフローを考えていく必要があるため。)

- バイナリ単体では動作しない  
(複数のバイナリと組み合わせて使用する)
- これ自体で完結しており、引数やパラメータを調整する必要がある
- バイナリ内の命令を直接書き換え、実行されない関数を実行する必要がある
- 乱数パラメータによって、ある時だけ違う動作をする

一通り動作を考えたのちに、リバースエンジニアリングして処理を見てみる。

### 3. Writeup

コマンド:

```
$ objdump -d -M intel ./Not_executed_branch.bin | less
```

main関数の頭数行:

```
0000000000001189 <main>:
1189: f3 0f 1e fa      endbr64
118d: 55              push    rbp
118e: 48 89 e5        mov     rbp, rsp
1191: 48 83 ec 30     sub     rsp, 0x30
1195: 64 48 8b 04 25 28 00 mov     rax, QWORD PTR fs:0x28
119c: 00 00
119e: 48 89 45 f8     mov     QWORD PTR [rbp-0x8], rax
11a2: 31 c0          xor     eax, eax
11a4: c7 45 dc 01 00 00 00 mov     DWORD PTR [rbp-0x24], 0x1
11ab: 48 b8 63 56 54 5f 5e movabs  rax, 0x5a585c5e5f545663
11b2: 5c 58 5a
11b5: 48 ba 56 7e 44 64 58 movabs  rdx, 0x4342745864447e56
11bc: 74 42 43
11bf: 48 89 45 e0     mov     QWORD PTR [rbp-0x20], rax
11c3: 48 89 55 e8     mov     QWORD PTR [rbp-0x18], rdx
11c7: c6 45 f0 52     mov     BYTE PTR [rbp-0x10], 0x52
11cb: 03 7d 4a 00     cmovbe  QWORD PTR [rbp-0x24], 0x0
```

### 3. Writeup

このWriteupの趣旨はアセンブリを読むことではないので、解読部分はスキップ。

```
11a2: 31 c0 xor eax,eax
11a4: c7 45 dc 01 00 00 00 mov DWORD PTR [rbp-0x24],0x1
11ab: 48 b8 63 56 54 5f 5e movabs rax,0x5a585c5e5f545663
11b2: 5c 58 5a
11b5: 48 ba 56 7e 44 64 58 movabs rdx,0x4342745864447e56
11bc: 74 42 43
11bf: 48 89 45 e0 mov QWORD PTR [rbp-0x20],rax
11c3: 48 89 55 e8 mov QWORD PTR [rbp-0x18],rdx
11c7: c6 45 f0 52 mov BYTE PTR [rbp-0x10],0x52
11cb: 83 7d dc 00 cmp DWORD PTR [rbp-0x24],0x0
11cf: 75 4c jne 121d <main+0x94>
11d1: c7 45 d8 00 00 00 00 mov DWORD PTR [rbp-0x28],0x0
11d8: eb 1c jmp 11f6 <main+0x6d>
```

### 3. Writeup

```
mov     DWORD PTR [rbp-0x24],0x1
```

```
cmp     DWORD PTR [rbp-0x24],0x0
```

スタック上のメモリ [rbp-0x24]は、  
「1」で初期化しているが、  
比較箇所まで書き換えられずに「0」と比較されている



つまり、絶対に実行されない部分がある



### 3. Writeup

絶対に実行されない処理を実行する方法として、以下の方法が挙げられる

- [rbp-0x24]の初期値を「1」から「0」に書き換えて実行する(静的)
- 実行時に\$IP(PC:プログラムカウンタ)を実行されない処理のアドレスに書き換える(動的)
- main関数の入り口の処理を、「jmp (cmpの次の命令)」と書き換える(静的)

どの方法でも同じ結果が期待できるが、オリジナルプログラムのセマンティクスを考慮するのであれば、一番上の方法が良さそう。  
(別に下2つのやり方を否定しているわけではない)

### 3. Writeup

[rbp0x24]の初期値を「1」から「0」に書き換えて実行する(静的)

```
$ hexedit ./Not_executed_branch.bin
```

該当のコードまで移動(命令バイト列で検索するのがおすすめ)

```
00001158  E5 74 0C 48 8B 3D A6 2E 00 00 E8 F9 FE FF FF E8 64 FF FF FF C6 05 9D 2E
00001170  00 00 01 5D C3 0F 1F 00 C3 0F 1F 80 00 00 00 00 F3 0F 1E FA E9 77 FF FF
00001188  FF F3 0F 1E FA 55 48 89 E5 48 83 EC 30 64 48 8B 04 25 28 00 00 00 48 89
000011A0  45 F8 31 C0 C7 45 DC 01 00 00 00 48 B8 63 56 54 5F 5E 5C 58 5A 48 BA 56
--- Not_executed_branch.bin --0x11A4/0x41A8-----
```

「C0 C7 45 DC 01 00 00 00」=>「C0 C7 45 DC 00 00 00 00」

```
C0 C7 45 DC 00 00 00 00
```

保存して閉じる(Ctrl+X)

### 3. Writeup

実行してみる

```
% ./Not_executed_branch.bin  
http-ctf{TachikomaIsSoCute}%
```

### 3. Writeup

flag: `http-ctf{TachikomalsSoCute}`

。。。タチコマって誰や

調べてみた。

### 3. Writeup

これは、、、  
作問者の工藤くんに聞きたいですね(白目)



## 4. おまけ

この方法でも試してみる。

- 実行時に\$IP(PC:プログラムカウンタ)を実行されない処理のアドレスに書き換える(動的)

GDBを起動

```
$ gdb ./Not_executed_branch.bin
```

## 4. おまけ

`$ disas main`で変数比較部分を見つけてから、  
`b *addr`でブレークポイントを設置する

```
gdb-peda$ b *main+70  
Breakpoint 1 at 0x11cf
```

rで実行する

ブレークポイントで停止したら、  
jne命令でジャンプしないように次の命令へPCを書き換える

```
gdb-peda$ jump *main+72
```

## 4. おまけ

フラグが表示される。

```
Continuing at 0x555555551d1.  
http-ctf{TachikomaIsSoCute}[Inferior 1 (process 4271) exited normally]  
Warning: not running
```