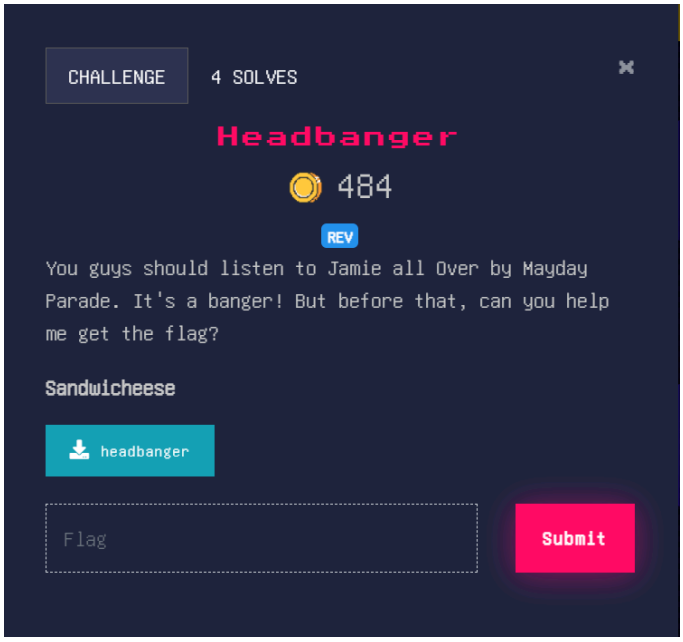
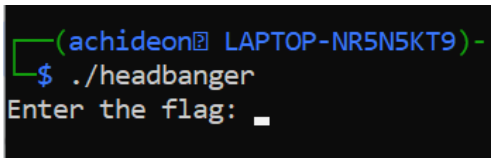
[Rev] **Headbanger** (484 points)

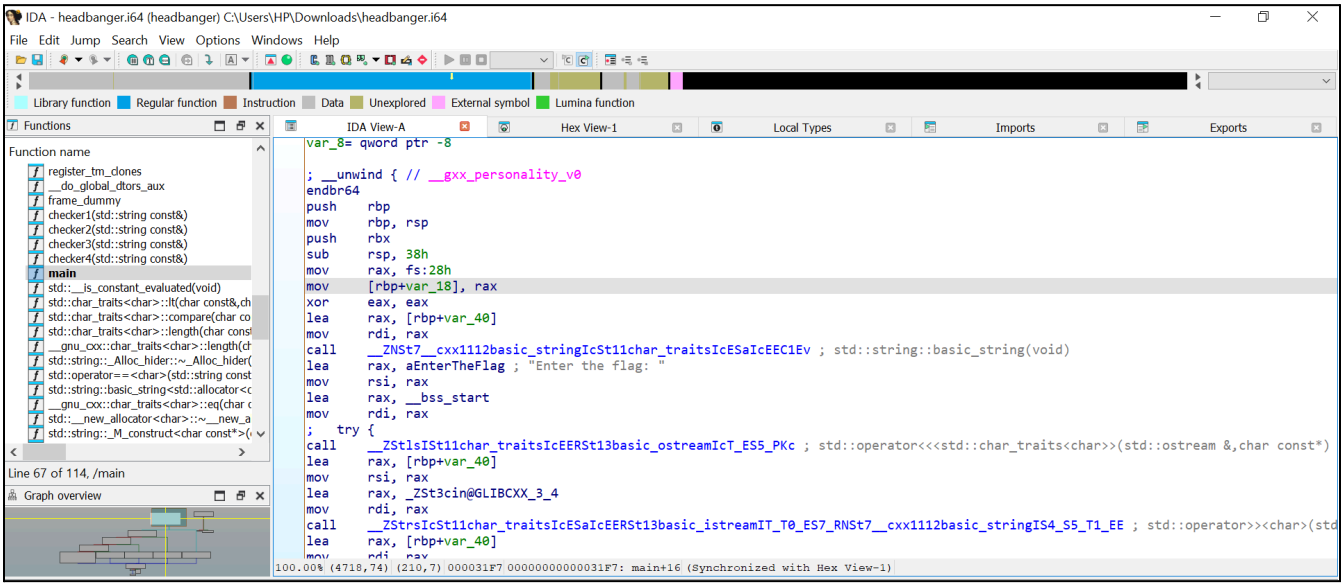
By: Achideon



Challenge dimulai dengan sebuah file binary bernama “headbanger” yang dapat di-execute



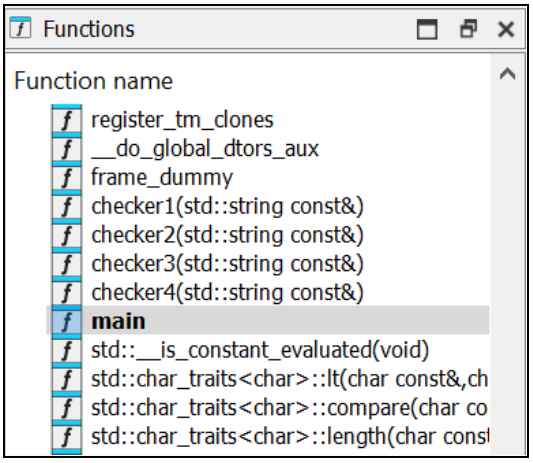
Agar dapat memahami konten di dalam binary file, maka file tersebut dibuka menggunakan disassembler IDA.



Untungnya file tersebut memiliki function main sehingga algoritma utama di dalam headbanger dapat dibaca secara langsung. Apabila function main di-decompile maka dapat terlihat kode yang menjadi challenge bagi flag ini.

```
14
15  v14 = __readfsqword(0x28u);
16  std::string::basic_string(v13, argv, envp);
17  std::operator<<<std::char_traits<char>>(&_bss_start, "Enter the flag: ");
18  std::operator>><char>(&std::cin, v13);
19  if ( checker1((__int64)v13) )
20  {
21    v3 = std::operator<<<std::char_traits<char>>(&_bss_start, "Checker 1 passed");
22    std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);
23    if ( (unsigned __int8)checker2((__int64)v13) )
24    {
25      v4 = std::operator<<<std::char_traits<char>>(&_bss_start, "Checker 2 passed");
26      std::ostream::operator<<(v4, &std::endl<char,std::char_traits<char>>);
27      if ( (unsigned __int8)checker3((__int64)v13) )
28      {
29        v5 = std::operator<<<std::char_traits<char>>(&_bss_start, "Checker 3 passed");
30        std::ostream::operator<<(v5, &std::endl<char,std::char_traits<char>>);
31        if ( (unsigned __int8)checker4(v13) )
32        {
33          v6 = std::operator<<<std::char_traits<char>>(&_bss_start, "Checker 4 passed");
34          std::ostream::operator<<(v6, &std::endl<char,std::char_traits<char>>);
35          v7 = std::operator<<<std::char_traits<char>>(&_bss_start, "Congratulations! You found the flag: ");
36          v8 = std::operator<<<char>(v7, v13);
37        }
```

Sekilas terlihat sejumlah command bahasa C++, pada line 18 terdapat command “cin” untuk menginput seperti yang muncul jika file headbanger dijalankan. Di line yang sama, terlihat jika input pengguna disimpan dalam variabel v13. Variabel tersebut kemudian dijalankan melalui fungsi checker1, checker2, checker3, serta checker4 sebelum akhirnya flag akhir didapatkan.



Mari memulai dengan fungsi checker1,

```
1  _BOOL8 __fastcall checker1(__int64 a1)
2  {
3    char v1; // bl
4    _BOOL4 v2; // r12d
5    _BYTE v4[40]; // [rsp+10h] [rbp-40h] BYREF
6    unsigned __int64 v5; // [rsp+38h] [rbp-18h]
7
8    v5 = __readfsqword(0x28u);
9    v1 = 0;
10   v2 = 0;
11   if ( std::string::length(a1) == 68 )
12   {
13     std::string::substr(v4, a1, 0LL, 7LL);
14     v1 = 1;
15     if ( (unsigned __int8)std::operator==<char>(v4, "CTFITB{") )
16     {
17       if ( *(_BYTE *)std::string::back(a1) == 125 )
18         v2 = 1;
19     }
20   }
21   if ( v1 )
22     std::string::~string(v4);
23   return v2;
24 }
```

Fungsi checker1 memiliki satu parameter yaitu **a1** yang berfungsi sebagai variabel string yang menjadi input dari pengguna. Pada line 11 terdapat command ‘if’ yang membandingkan panjang input dengan 68 character. Pada line 13 diambil 7 character substring awal dari variabel **a1** kemudian disimpan ke dalam variabel **v4**.

Variabel **v4** lalu dibandingkan lagi dengan *CTFITB{* pada line 15. Akhirnya, karakter terakhir pada **a1** dibandingkan dengan karakter dengan ascii code 125 yaitu *}*. Apabila **a1** memenuhi seluruh perbandingan tersebut, maka variabel v2 yang di-return akan menjadi 1.

Maka dapat disimpulkan, string yang dimasukkan harus sepanjang 68 karakter, diawali dengan *CTFITB{* dan diakhiri dengan *}*.

```
┌──(achideon LAPTOP-NR5N5KT9)-[~/CTF/ITB/Reverse Engineering]
└─$ ./headbanger
Enter the flag: CTFITB{**********************************************************}
Checker 1 passed
Checker 2 failed
```

Kemudian dilanjutkan dengan fungsi checker2

```
14  v11 = __readfsqword(0x28u);
15  std::string::substr(innerflag, flag, 7LL, 67LL);
16  v8 = &v5;
17  std::string::basic_string<std::allocator<char>>(key, "DUS1C_Uk47K447Yb3Dov", &v5);
18  std::__new_allocator<char>::~__new_allocator(&v5);
19  for ( i = 0; ; i += 3 )
20  {
21    a = i;
22    if ( a >= std::string::length(innerflag) - 1 )
23      break;
24    v6 = *(_BYTE *)std::string::operator[](innerflag, i);
25    if ( v6 <= 64 || v6 > 90 )
26    {
27      if ( v6 > 96 && v6 <= 122 )
28        v6 = (v6 - 90) % 26 + 97;
29    }
30    else
31    {
32      v6 = (v6 - 58) % 26 + 65;
33    }
34    v1 = (_BYTE *)std::string::operator[](key, i / 3);
35    if ( v6 != *v1 )
36    {
37      v2 = 0;
38      goto LABEL_13;
39    }
40  }
41  v2 = 1;
42LABEL_13:
43  std::string::~string(key);
44  std::string::~string(innerflag);
45  return v2;
```

Sejumlah variabel awal dinamai ulang untuk memudahkan pembacaan. Variabel innerflag adalah isi dari flag yang dicari atau string yang berada di dalam kurung { }, terlihat dari command substring dari char ke-7 sampai char ke-67.

Terdapat sebuah for loop dengan iterasi i yang naik kelipatan 3, dan berakhir ketika i melewati panjang string innerflag. Line ke-24 mengambil char ke-i dari string innerflag, memasukkannya ke dalam variabel v6, kemudian memprosesnya ke bawah. Sehingga pada line ke-35 variabel v6 yang sudah diproses tersebut dibandingkan dengan variabel v1 yang berasal dari char ke-i/3 dari string key.

Namun apa yang dilakukan oleh program dari line 25 sampai line 34? Program memproses variabel v6 apabila berada pada rentang 97 sampai 122 dan juga pada rentang 65 sampai 90. Apabila melihat dari tabel ascii,

```
cook@pop-os:~$ ascii -d
   0 NUL    16 DLE    32      48 0    64 @    80 P    96 `   112 p
   1 SOH    17 DC1    33 !    49 1    65 A    81 Q    97 a   113 q
   2 STX    18 DC2    34 "    50 2    66 B    82 R    98 b   114 r
   3 ETX    19 DC3    35 #    51 3    67 C    83 S    99 c   115 s
   4 EOT    20 DC4    36 $    52 4    68 D    84 T   100 d   116 t
   5 ENQ    21 NAK    37 %    53 5    69 E    85 U   101 e   117 u
   6 ACK    22 SYN    38 &    54 6    70 F    86 V   102 f   118 v
   7 BEL    23 ETB    39 '    55 7    71 G    87 W   103 g   119 w
   8 BS     24 CAN    40 (    56 8    72 H    88 X   104 h   120 x
   9 HT     25 EM     41 )    57 9    73 I    89 Y   105 i   121 y
  10 LF     26 SUB    42 *    58 :    74 J    90 Z   106 j   122 z
  11 VT     27 ESC    43 +    59 ;    75 K    91 [   107 k   123 {
  12 FF     28 FS     44 ,    60 <    76 L    92 \   108 l   124 |
  13 CR     29 GS     45 -    61 =    77 M    93 ]   109 m   125 }
  14 SO     30 RS     46 .    62 >    78 N    94 ^   110 n   126 ~
  15 SI     31 US     47 /    63 ?    79 O    95 _   111 o   127 DEL
```

Artinya variabel **v6** hanya akan diproses apabila berupa huruf kecil atau besar, maka sisanya tidak akan diproses. Apa proses yang dilakukan? Line 28 menunjukkan variabel **v6** dikurangi 90, di-modulo 26, lalu ditambah 97, sedangkan line 32 menunjukkan variabel **v6** dikurangi 58, di-modulo 26, lalu ditambah 65.

Proses tersebut merupakan caesar cipher sebesar 7 ke kanan, mengubah string key dari DUS1C_Uk47K447Yb3Dov menjadi WNL1V_Nd47D447Ru3Who. Variabel **v6** merupakan char ke-i yaitu kelipatan 3 dari innerflag, sedangkan variabel **v1** merupakan char ke-i/3 dari string key, artinya setiap char dari string yang baru saja dicipher disimpan dengan jarak 3 char antara satu sama lainnya.

CTFITB{W**N**L**1**V**_**N**d**4**7**D**4**4**7**R**u**3**W**h**o**}

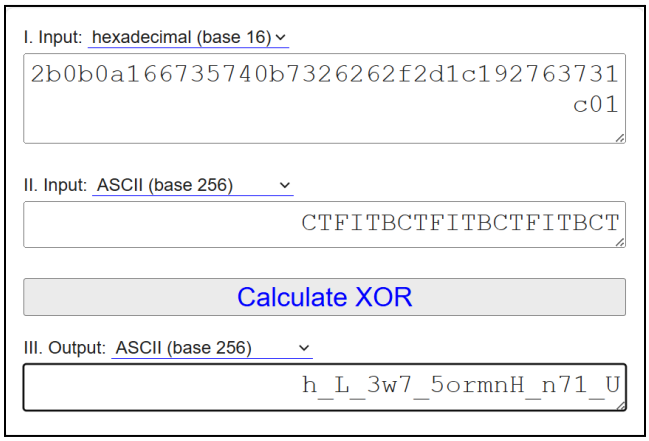
```
┌──(achideon LAPTOP-NR5N5KT9)-[~/CTF/ITB/Reverse Engineering]
└─$ ./headbanger
Enter the flag: CTFITB{W**N**L**1**V**_**N**d**4**7**D**4**4**7**R**u**3**W**h**o**}
Checker 1 passed
Checker 2 passed
Checker 3 failed
```

Kemudian lanjut ke checker 3

```
20  v18 = __readfsqword(0x28u);
21  std::string::substr(v14, a1, 0LL, 6LL);
22  std::string::substr(innerflag, a1, 7LL, 67LL);
23  v13 = &v7;
24  std::string::basic_string<std::allocator<char>>(key, "2b0b0a166735740b7326262f2d1c192763731c01", &v7);
25  std::__new_allocator<char>::~__new_allocator(&v7);
26  iteration = 0;
27  v11 = 0;
28  for ( i = 0; ; i += 3 )
29  {
30    v5 = i;
31    if ( v5 >= std::string::length(innerflag) - 1 )
32      break;
33    v8 = *(_BYTE *)std::string::operator[](innerflag, i + 1);
34    v9 = *(_BYTE *)std::string::operator[](v14, iteration % 6);// CTFITB
35    v8 ^= v9;                                   // xor
36    sprintf(s, "%02x", v8);
37    v1 = s[0];
38    if ( v1 != *(_BYTE *)std::string::operator[](key, v11)
39      || (v2 = s[1], v2 != *(_BYTE *)std::string::operator[](key, v11 + 1)) )
40    {
41      v4 = 0;
42      goto LABEL_11;
43    }
44    ++iteration;
45    v11 += 2;
46  }
47  v4 = 1;
48LABEL_11:
49  std::string::~string(key);
50  std::string::~string(innerflag);
51  std::string::~string(v14);
52  return v4;
}
```

checker 3 memiliki proses kerja yang serupa dengan checker 2, namun terdapat perbedaan yaitu sekarang variabel perbandingan **v8** mengambil char ke-(i + 1) dari innerflag berarti bergeser satu dari setiap char yang ditemukan dalam checker 2. Selain itu sekarang variabel **v8** mengalami operasi xor dengan variabel **v9**. Pada line 34, variabel **v9** diisi dengan char ke-(j % 6) dari substring flag *CTFITB* dengan j yang naik 1 setiap iterasi. Jumlah karakter hex pada string key

ada 40, berarti apabila dikonversikan menjadi ascii akan memiliki 20 karakter, melalui informasi tersebut maka key untuk melakukan operasi xor adalah CTFITBCTFITBCTFITBCT



Didapat string *h_L_3w7_5ormnH_n71_U*

CTFITB{Wh*N_*LL*1_*V3*_w*N7*d_*45*7o*Dr*4m*4n*7H*R_*un*37*W1*h_*oU*}

```
┌──(achideon㉿ LAPTOP-NR5N5KT9)-[~/CTF/ITB/Reverse Engineering]
└─$ ./headbanger
Enter the flag: CTFITB{Wh*N_*LL*1_*V3*_w*N7*d_*45*7o*Dr*4m*4n*7H*R_*un*37*W1*h_*oU*}
Checker 1 passed
Checker 2 passed
Checker 3 passed
Checker 4 failed
```

Yak, tinggal checker 4 untuk mendapat flag yang dicari. Sebenarnya dari flag yang dimiliki sekarang, dapat terlihat kata-kata yang mungkin akan terbentuk.

```
46
47  v44 = __readfsqword(0x28u);
48  std::string::substr(innerflag, flag, 7LL, 67LL);
49  v1 = *(char *)std::string::operator[](innerflag, 2LL);
50  v2 = *(char *)std::string::operator[](innerflag, 20LL) * v1;
51  if ( v2 - *(char *)std::string::operator[](innerflag, 56LL) != 2512 )
52    goto LABEL_22;
53  v3 = *(char *)std::string::operator[](innerflag, 5LL);
54  v4 = *(char *)std::string::operator[](innerflag, 56LL) * v3;
55  if ( v4 - *(char *)std::string::operator[](innerflag, 53LL) != 4573 )
56    goto LABEL_22;
57  v5 = *(char *)std::string::operator[](innerflag, 8LL);
58  v6 = *(char *)std::string::operator[](innerflag, 32LL);
59  if ( v5 + v6 * *(char *)std::string::operator[](innerflag, 14LL) != 4277 )
60    goto LABEL_22;
61  v7 = *(char *)std::string::operator[](innerflag, 11LL);
62  v8 = v7 - *(char *)std::string::operator[](innerflag, 8LL);
63  if ( v8 + *(char *)std::string::operator[](innerflag, 17LL) != 8 )
64    goto LABEL_22;
65  v9 = *(char *)std::string::operator[](innerflag, 14LL);
66  v10 = *(char *)std::string::operator[](innerflag, 20LL) * v9;
67  if ( v10 - *(char *)std::string::operator[](innerflag, 17LL) != 4130 )
68    goto LABEL_22;
69  v11 = *(char *)std::string::operator[](innerflag, 17LL);
70  v12 = *(char *)std::string::operator[](innerflag, 11LL) * v11;
71  if ( v12 - *(char *)std::string::operator[](innerflag, 14LL) != 2570 )
72    goto LABEL_22;
73  v13 = *(char *)std::string::operator[](innerflag, 20LL);
74  v14 = *(char *)std::string::operator[](innerflag, 41LL);
75  if ( v13 + v14 * *(char *)std::string::operator[](innerflag, 53LL) != 2856 )
76    goto LABEL_22;
77  v15 = *(char *)std::string::operator[](innerflag, 23LL);
78  v16 = *(char *)std::string::operator[](innerflag, 20LL) * v15;
```

Checker 4 nguli banget mas

Checker 4 sebenarnya terlihat mustahil untuk dilakukan apabila tidak ada pengetahuan awal mengenai flagnya, namun karena sejumlah kata dapat terbentuk di dalam flag sebenarnya sejumlah char dapat dikira-kira.

Contohnya pada line 49 hingga 52

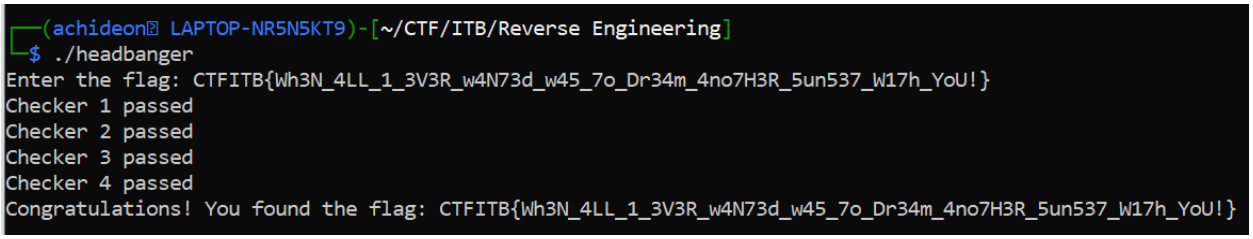
```
49  v1 = *(char *)std::string::operator[](innerflag, 2LL);
50  v2 = *(char *)std::string::operator[](innerflag, 20LL) * v1;
51  if ( v2 - *(char *)std::string::operator[](innerflag, 56LL) != 2512 )
52    goto LABEL_22;
```

v1 diambil dari innerflag[2] sedangkan **v2** diambil dari innerflag[20] dikalikan dengan **v1**. Melihat kondisi flag saat ini, dapat diperkirakan innerflag[2] bisa berupa karakter *e, E,* atau *3* karena membentuk kata ‘when’. Hal yang sama terjadi pada innerflag[20] dapat berupa karakter *e, E,* atau *3* karena karakter di sekitarnya membentuk kata ‘wanted’.

Pada line 51, variabel **v2** dikurangi dengan innerflag[56] menghasilkan angka 2512. Karena ascii table hanya memiliki nilai maksimum 127, maka dapat diperkirakan variabel **v2** berada di antara 2385 sampai 2639. Dari kemungkinan karakter pada innerflag[2] dan innerflag[20], hanya apabila kedua karakter tersebut adalah 3, variabel **v2** menjadi valid. Karena kode ascii dari angka 3 adalah 51, dan 51*51 = 2601, cocok untuk menjadi variabel **v2**. Dari variabel **v2** dapat ditemukan juga innerflag[56] dengan 2601 - 2512 = 89 yaitu karakter *Y*.

Menggunakan prinsip serupa, setiap line dengan command if diperiksa untuk mendapatkan char yang tersisa dari flag. (yep nguli banget 😭)

Didapatkanlah flag akhir yaitu,



FLAG:
CTFITB{Wh3N_4LL_1_3V3R_w4N73d_w45_7o_Dr34m_4no7H3R_5un537_W17h_YoU!}