Signal First to solve this challenge!

Reversing - 150 points

# Soal

|  |
| --- |
| cari passwordnya dapatkan flagnya  [signal](https://ctf.idsecconf.org/download?file_key=440f0a9a29f1f25b9affb612ecc074c200f278e855c0da7b2099c8e14da47e7e&team_key=2a3d00950bcec988c6f9a00242687028ade2e180c4a337d64a8761b253b2df59) - 66709fa0e58608c25f37af97bf0f1c03 |

# Solusi

Diberikan file ELF 64-bit, not stripped. Hasil dekompilasi dengan IDA:

|  |
| --- |
| int **\_\_cdecl** main(int argc, **const** char \*\*argv, **const** char \*\*envp) {  int result; *// eax@2*  size\_t v4; *// rax@3*  unsigned **\_\_int64** v5; *// rax@3*  unsigned **\_\_int64** v6; *// rdx@3*  unsigned **\_\_int64** v7; *// rt2@3*  double v8; *// xmm0\_8@4*  **\_\_int64** (**\_\_fastcall** \*v9)(); *// [sp+20h] [bp-1B0h]@3*  **\_\_int64** v10; *// [sp+28h] [bp-1A8h]@3*  int v11; *// [sp+A8h] [bp-128h]@3*  double v12; *// [sp+1C8h] [bp-8h]@6*   **if** ( argc > 1 )  {  memset(input, 0, 0x200uLL);  strncpy(input, argv[1], 0x1FFuLL);  signal(10, usr\_1);  sigemptyset((sigset\_t \*)&v10);  v11 = 0x10000000;  v9 = fpe;  sigaction(8, (**const** **struct** sigaction \*)&v9, 0LL);  v4 = strlen(input);  v7 = 1 % (v4 - 16);  v5 = 1 / (v4 - 16);  v6 = v7;  **if** ( (v5 & 0x8000000000000000LL) != 0LL )  {  v6 = v5 & 1 | (v5 >> 1);  v8 = (double)(signed int)v6 + (double)(signed int)v6;  }  **else**  {  v8 = (double)(signed int)v5;  }  v12 = v8;  printf("checking %f**\n**", &v9, v6, v8);  printf("%s", invalid\_pass);  result = 0;  }  **else**  {  puts("usage: signal <password>");  result = 0;  }  **return** result; } |

Program merespon dua signal[[1]](#footnote-0):

* Signal 8: SIGUSR1
* Signal 10: SIGFPE

Handler untuk SIGUSR1 adalah fungsi pengecekan password:

|  |
| --- |
| void usr\_1() {  int sig; *// [sp+10h] [bp-10h]@1*  int v1; *// [sp+14h] [bp-Ch]@1*  int v2; *// [sp+1Ch] [bp-4h]@1*   sig = 2;  v1 = 12;  v2 = (int\_input ^ 0x1122334455667788LL) == password;  signal(2, incorrect\_password);  signal(12, correct\_password);  kill(0, \*(&sig + v2)); } |

Handler untuk SIGFPE adalah fungsi untuk mengubah input menjadi angka (sebagai hexadecimal) dan men-trigger SIGUSR1:

|  |
| --- |
| void \_\_noreturn fpe() {  char \*endptr; *// [sp+18h] [bp-8h]@1*   puts("Start password checking...");  int\_input = strtoll(input, &endptr, 16);  kill(0, 10);  exit(0); } |

Di main, ada statemen ini:

|  |
| --- |
| v4 = strlen(input);  v7 = 1 % (v4 - 16);  v5 = 1 / (v4 - 16); |

Lihat bahwa jika kita ingin men-trigger SIGFPE, kita harus membuat pembagian dengan nol, sehingga dari sini kita tahu bahwa v4 (panjang input) harus bernilai 16 supaya   
1 / (v4 - 16) menjadi 1 / (16 - 16).

Kemudian di fungsi pengecekan password:

|  |
| --- |
| v2 = (int\_input ^ 0x1122334455667788LL) == password; |

Input kita di-XOR dengan 0x1122334455667788 harus bernilai sama dengan variabel global password:

|  |
| --- |
| .data:0000000000601280 password dq **62A2A01426E579E**h |

Langsung saja di-XOR:

|  |
| --- |
| >>> 0x1122334455667788 ^ 0x62A2A01426E579E  1659604247215874070 |

Lihat panjang angkanya bukan 16 tapi 19. Kalau kita lihat lagi di fungsi respon SIGFPE, input angka kita harus berupa hex (diubah dengan strtoll() basis 16):

|  |
| --- |
| int\_input = strtoll(input, &endptr, 16); |

Sehingga input yang valid adalah:

|  |
| --- |
| >>> hex(1659604247215874070)  '0x1708194517082016' |

Bisa dimasukkan ke program untuk memastikan:

|  |
| --- |
| $ ./signal 1708194517082016  Start password checking...  correct password, the flag is **flag{1708194517082016}** |

1. <http://www.comptechdoc.org/os/linux/programming/linux_pgsignals.html> [↑](#footnote-ref-0)