AlexCTF 2017

C++ is awesome Write Up

Author : pandakim2122

Date : 2017.12.04

```
if ( argc != 2 )
{
  v3 = *argv;
  v4 = std::operator<<<std::char_traits<char>>(&std::cout, "Usage: ", a3);
  v6 = std::operator<<<std::char_traits<char>>(v4, v3, v5);
  std::operator<<<std::char_traits<char>>(v6, " flag\n", v7);
  exit(0);
}
```

이 파일은 인자를 한 개 전달받아 동작합니다.

```
std::allocator<char>::allocator(&v13, argv, a3);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&arg_str, argv[1], &v13);
std::allocator<char>::~allocator(&v13);
j = 0;
for ( i = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::begin(&arg_str); ; increment(&i) )
{
  v14 = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::end(&arg_str);
  if ( !sub_400D3D((__int64)&i, (__int64)&v14) )
    break;
  input_ch = *(unsigned __int8 *)sub_400D9A((__int64)&i);
  if ( (_BYTE)input_ch != off_6020A0[dword_6020C0[j]] )
    goto_exit((__int64)&i, (__int64)&v14, input_ch);
  ++j;
}
```

입력받은 인자(문자열)에 대해 1바이트씩 offset_6020a0[dword_6020c0[j]]와 동일한 지 검증하는데, offset_6020a0에는 특정 문자열이, dword_6020c0에는 4바이트의 배열 형태로 어떠한 인덱스 값이 존재합니다.

offset_6020a0

- L3t_ME_T3ll_Y0u_S0m3th1ng_1mp0rtant_A_{FL4G}_W0nt_b3_3X4ctly_th4t_345y_t0_c4ptur3_H0wev3r_1T_w1ll_b3_C00l_1F_Y0u_g0t_1t

dword_6020c0

- [0x24, 0x0, 0x5, 0x36, 0x65, 0x7, 0x27, 0x26, 0x2D, 0x1, 0x3, 0x0, 0x0D, 0x56, 0x1, 0x3, 0x65, 0x3, 0x2D, 0x16, 0x2, 0x15, 0x3, 0x65, 0x0, 0x29, 0x44, 0x44, 0x1, 0x44, 0x2B]

```
.data:00000000006020A0 off_6020A0      dq offset aL3tMeT3llY0uS0
.data:00000000006020A0                                         ; DATA XREF: main+D1↑r
.data:00000000006020A0                                         ; "L3t_ME_T3ll_Y0u_S0m3th1ng_1mp0rtant_A_{"...
.data:00000000006020A8                 align 20h
.data:00000000006020C0 ; int dword_6020C0[]
.data:00000000006020C0 dword_6020C0    dd 24h                  ; DATA XREF: main+DD↑r
.data:00000000006020C4                 dd 0
.data:00000000006020C8                 dd 5
.data:00000000006020CC                 dd 36h
.data:00000000006020D0                 dd 65h
.data:00000000006020D4                 dd 7
.data:00000000006020D8                 dd 27h
.data:00000000006020DC                 dd 26h
.data:00000000006020E0                 dd 2Dh
.data:00000000006020E4                 dd 1
.data:00000000006020E8                 dd 3
.data:00000000006020EC                 dd 0
.data:00000000006020F0                 dd 0Dh
.data:00000000006020F4                 dd 56h
.data:00000000006020F8                 dd 1
.data:00000000006020FC                 dd 3
.data:0000000000602100                 dd 65h
.data:0000000000602104                 dd 3
.data:0000000000602108                 dd 2Dh
.data:000000000060210C                 dd 16h
.data:0000000000602110                 dd 2
.data:0000000000602114                 dd 15h
.data:0000000000602118                 dd 3
.data:000000000060211C                 dd 65h
.data:0000000000602120                 dd 0
.data:0000000000602124                 dd 29h
.data:0000000000602128                 dd 44h
.data:000000000060212C                 dd 44h
.data:0000000000602130                 dd 1
.data:0000000000602134                 dd 44h
.data:0000000000602138                 dd 2Bh
.data:0000000000602138 _data           ends

.rodata:0000000000400E58 aL3tMeT3llY0uS0 db 'L3t_ME_T3ll_Y0u_S0m3th1ng_1mp0rtant_A_{FL4G}_W0nt_b3_3X4ctly_th4t'
.rodata:0000000000400E58                                       ; DATA XREF: .data:off_6020A0↓o
.rodata:0000000000400E58                 db '_345y_t0_c4ptur3_H0wev3r_1T_w1ll_b3_C00l_1F_Y0u_g0t_1t',0
```

```
>>> s = b"L3t_ME_T3ll_Y0u_S0m3th1ng_1mp0rtant_A_{FL4G}_W0nt_b3_3X4ctly_th4t_345y_t0_c4ptur3_H0wev3r_1T_w1ll_b3_C00l_1F_Y0u_g0t_1t"
>>> idx = [0x24, 0x0, 0x5, 0x36, 0x65, 0x7, 0x27, 0x26, 0x2D, 0x1, 0x3, 0x0, 0x0D, 0x56, 0x1, 0x3, 0x65, 0x3, 0x2D, 0x16, 0x2, 0x15
>>> res = ""
>>>
>>> for i in idx :
...     res += chr(s[i])
...
>>>
>>> print(res)
ALEXCTF{W3_L0v3_C_W1th_CL45535}
>>>
```

Offset_6020a0의 0x24번째, 0번째, 5번째, 0x36번째 … 를 조합하면 다음과 같이 답이 나옵니다.

Flag : ALEXCTF{W3_L0v3_C_W1th_CL45535}