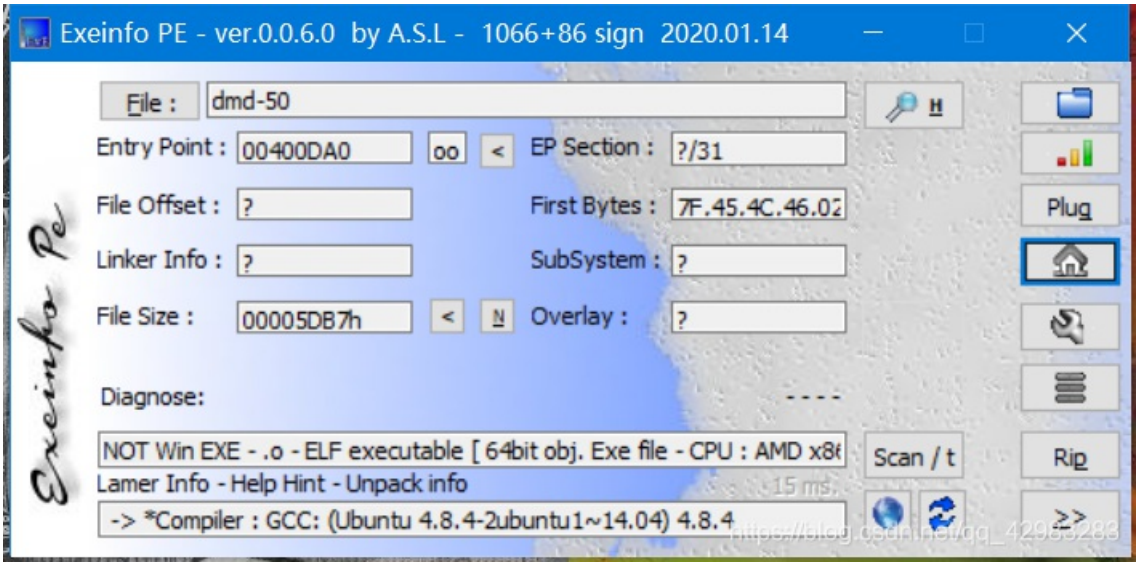
xctf攻防世界dmd-50 writeup

原创

qq_112419837 于 2020-11-12 17:34:24 发布 117 收藏

下载附件，打开：



ida64打开，得到关键字符串780438d5b6e29db0898bc4f0225935c0：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rax
  __int64 v4; // rax
  __int64 v5; // rax
  __int64 v6; // rax
  __int64 v7; // rax
  __int64 v8; // rax
  __int64 v9; // rax
  __int64 v10; // rax
  __int64 v11; // rax
  __int64 v12; // rax
```

```
__int64 v13; // rax
__int64 v14; // rax
__int64 v15; // rax
__int64 v16; // rax
__int64 v17; // rax
__int64 v18; // rax
__int64 v19; // rax
__int64 v20; // rax
__int64 v21; // rax
int result; // eax
__int64 v23; // rax
__int64 v24; // rax
__int64 v25; // rax
__int64 v26; // rax
__int64 v27; // rax
__int64 v28; // rax
__int64 v29; // rax
__int64 v30; // rax
__int64 v31; // rax
__int64 v32; // rax
__int64 v33; // rax
__int64 v34; // rax
__int64 v35; // rax
__int64 v36; // rax
__int64 v37; // rax
char v38; // [rsp+Fh] [rbp-71h]
char v39; // [rsp+10h] [rbp-70h]
char v40; // [rsp+20h] [rbp-60h]
_BYTE *v41; // [rsp+28h] [rbp-58h]
char v42; // [rsp+30h] [rbp-50h]
unsigned __int64 v43; // [rsp+68h] [rbp-18h]

v43 = __readfsqword(0x28u);
std::operator<<<std::char_traits<char>>(&std::cout, "Enter the valid key!\n", envp);
std::operator>><char,std::char_traits<char>>(&edata, &v42);
std::allocator<char>::allocator(&v38);
std::string::string(&v39, &v42, &v38);
md5(&v40, &v39);
v41 = (_BYTE *)std::string::c_str((std::string *)&v40);
std::string::~string((std::string *)&v40);
std::string::~string((std::string *)&v39);
std::allocator<char>::~allocator(&v38);
if ( *v41 != '7'
  || v41[1] != '8'
  || v41[2] != '0'
  || v41[3] != '4'
  || v41[4] != '3'
  || v41[5] != '8'
  || v41[6] != 'd'
  || v41[7] != '5'
  || v41[8] != 'b'
  || v41[9] != '6'
  || v41[10] != 'e'
  || v41[11] != '2'
  || v41[12] != '9'
  || v41[13] != 'd'
  || v41[14] != 'b'
  || v41[15] != '0'
  || v41[16] != '8'
  || v41[17] != '9'
```

```
    || v41[18] != '8'
    || v41[19] != 'b'
    || v41[20] != 'c'
    || v41[21] != '4'
    || v41[22] != 'f'
    || v41[23] != '0'
    || v41[24] != '2'
    || v41[25] != '2'
    || v41[26] != '5'
    || v41[27] != '9'
    || v41[28] != '3'
    || v41[29] != '5'
    || v41[30] != 'c'
    || v41[31] != '0' )
  {
    v23 = std::operator<<<std::char_traits<char>>(&std::cout, 'I');
    v24 = std::operator<<<std::char_traits<char>>(v23, 'n');
    v25 = std::operator<<<std::char_traits<char>>(v24, 'v');
    v26 = std::operator<<<std::char_traits<char>>(v25, 'a');
    v27 = std::operator<<<std::char_traits<char>>(v26, 'l');
    v28 = std::operator<<<std::char_traits<char>>(v27, 'i');
    v29 = std::operator<<<std::char_traits<char>>(v28, 'd');
    v30 = std::operator<<<std::char_traits<char>>(v29, ' ');
    v31 = std::operator<<<std::char_traits<char>>(v30, 'K');
    v32 = std::operator<<<std::char_traits<char>>(v31, 'e');
    v33 = std::operator<<<std::char_traits<char>>(v32, 'y');
    v34 = std::operator<<<std::char_traits<char>>(v33, '!');
    v35 = std::operator<<<std::char_traits<char>>(v34, ' ');
    v36 = std::operator<<<std::char_traits<char>>(v35, ':');
    v37 = std::operator<<<std::char_traits<char>>(v36, '(');
    std::ostream::operator<<(v37, &std::endl<char,std::char_traits<char>>);
    result = 0;
  }
  else
  {
    v3 = std::operator<<<std::char_traits<char>>(&std::cout, 'T');
    v4 = std::operator<<<std::char_traits<char>>(v3, 'h');
    v5 = std::operator<<<std::char_traits<char>>(v4, 'e');
    v6 = std::operator<<<std::char_traits<char>>(v5, ' ');
    v7 = std::operator<<<std::char_traits<char>>(v6, 'k');
    v8 = std::operator<<<std::char_traits<char>>(v7, 'e');
    v9 = std::operator<<<std::char_traits<char>>(v8, 'y');
    v10 = std::operator<<<std::char_traits<char>>(v9, ' ');
    v11 = std::operator<<<std::char_traits<char>>(v10, 'i');
    v12 = std::operator<<<std::char_traits<char>>(v11, 's');
    v13 = std::operator<<<std::char_traits<char>>(v12, ' ');
    v14 = std::operator<<<std::char_traits<char>>(v13, 'v');
    v15 = std::operator<<<std::char_traits<char>>(v14, 'a');
    v16 = std::operator<<<std::char_traits<char>>(v15, 'l');
    v17 = std::operator<<<std::char_traits<char>>(v16, 'i');
    v18 = std::operator<<<std::char_traits<char>>(v17, 'd');
    v19 = std::operator<<<std::char_traits<char>>(v18, ' ');
    v20 = std::operator<<<std::char_traits<char>>(v19, ':');
    v21 = std::operator<<<std::char_traits<char>>(v20, ')');
    std::ostream::operator<<(v21, &std::endl<char,std::char_traits<char>>);
    result = 0;
  }
  return result;
}
```

既然得到了输入字符串，那么远程运行，发现还是不对：

```
[1] Accepting connection from 192.168.163.1 ...
Enter the valid key!
780438d5b6e29db0898bc4f0225935c0
Invalid Key! :(
```

设置断点，在判断前：

```
0  md5(&v40, &v39);
1  v41 = (_BYTE *)std::string::c_str((std::string *)&v40);
2  std::string::~string((std::string *)&v40);
3  std::string::~string((std::string *)&v39);
4  std::allocator<char>::~allocator(&v38);
5  if ( *v41 != '7'
6    || v41[1] != '8'
7    || v41[2] != '0'
8    || v41[3] != '4'
00000EC0 main:48 (400EC0)
```


发现v41的值已经不符合判断条件：

```
[stack]:00007FFE5DBD8FD5 db    0
[stack]:00007FFE5DBD8FD6 db    0
[stack]:00007FFE5DBD8FD7 db    0
[stack]:00007FFE5DBD8FD8 db  38h ; 8
[stack]:00007FFE5DBD8FD9 db  77h ; w
[stack]:00007FFE5DBD8FDA db  55h ; U
[stack]:00007FFE5DBD8FDB db    2
[stack]:00007FFE5DBD8FDC db    0
[stack]:00007FFE5DBD8FDD db    0
[stack]:00007FFE5DBD8FDE db    0
[stack]:00007FFE5DBD8FDF db    0
[stack]:00007FFE5DBD8FE0 db  37h ; 7
[stack]:00007FFE5DBD8FE1 db  38h ; 8
[stack]:00007FFE5DBD8FE2 db  30h ; 0
[stack]:00007FFE5DBD8FE3 db  34h ; 4
[stack]:00007FFE5DBD8FE4 db  33h ; 3
[stack]:00007FFE5DBD8FE5 db  38h ; 8
[stack]:00007FFE5DBD8FE6 db  64h ; d
[stack]:00007FFE5DBD8FE7 db  35h ; 5
UNKNOWN 00007FFE5DBD8FD8: [stack]:00007FFE5DBD8FD8 (Synchronized with RIP)
```

ex View-1

发现v41起始值是8WU,后面才是7804那个字符串。

仔细分析前面的代码，v42接收了输入，然后把值赋给了v39，v39经过md5以后赋给v40，之后赋给v41：

```
v43 = __readfsqword(0x28u);
std::operator<<<std::char_traits<char>>(&std::cout, "Enter the valid key!\n", envp);
std::operator>><char,std::char_traits<char>>(&edata, &v42);
std::allocator<char>::allocator(&v38);
std::string::string(&v39, &v42, &v38);
md5(&v40, &v39);
v41 = (_BYTE *)std::string::c_str((std::string *)&v40);
std::string::~string((std::string *)&v40);
std::string::~string((std::string *)&v39);
std::allocator<char>::~allocator(&v38);
if ( *v41 != 55
  || v41[1] != 56
  || v41[2] != 48
  || v41[3] != 52
```

然后用工具解密780438d5b6e29db0898bc4f0225935c0：

v40那里存的是md5的地址：

```
std::operator<<<std::char_traits<char>>(&std::cout, "Enter the valid key!\n", envp);
 std::operator>><char,std::char_traits<char>>(&edata, &v42); #v42接收输入
 std::allocator<char>::allocator(&v38);
 std::string::string(&v39, &v42, &v38);#v39存入输入的地址
 md5((MD5 *)&v40, (const std::string *)&v39);#输入经过md5以后存入v40地址
 v41 = std::string::c_str((std::string *)&v40);
 std::string::~string((std::string *)&v40);
 std::string::~string((std::string *)&v39);
 std::allocator<char>::~allocator(&v38);
```