攻防世界 srm-50

原创

别害怕我在 于 2021-08-11 16:25:36 发布 122 收藏

分类专栏： CTF逆向reverse新手

本文链接：https://blog.csdn.net/afanzcf/article/details/119610095

---

title: 攻防世界 srm-50
date: 2021年8月11日 15点20分
tags: 攻防世界
categories: 攻防世界

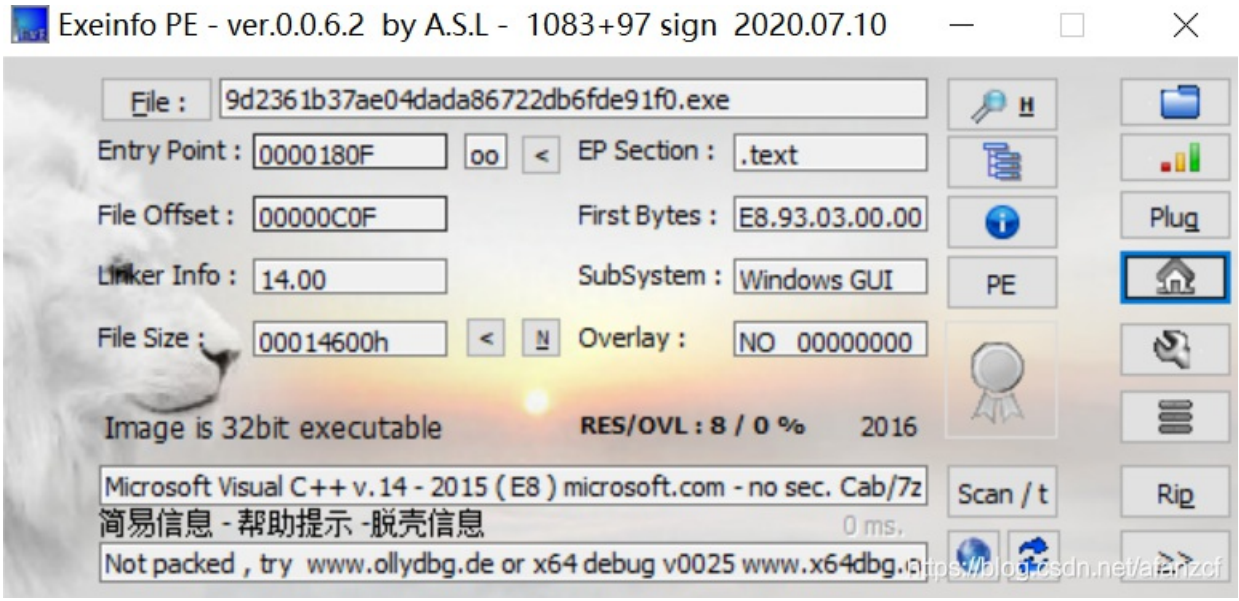
今天做题的时候，碰到了一道很有趣的题目，一个软件，邮箱注册之后，弹窗给flag。

攻防世界高手进阶区的一道题 srm-50

自我分析过程是自己从拿到题目，开始一步一步的分析过程，记录了自己的不足之处，以及思考不到位，下次希望能有所提高。

自我分析

1、PE



2、IDApro分析

（1）还是先shift + F12 查看字符串窗口

字符串太多。没耐心寻找，直接拖到了最后面。

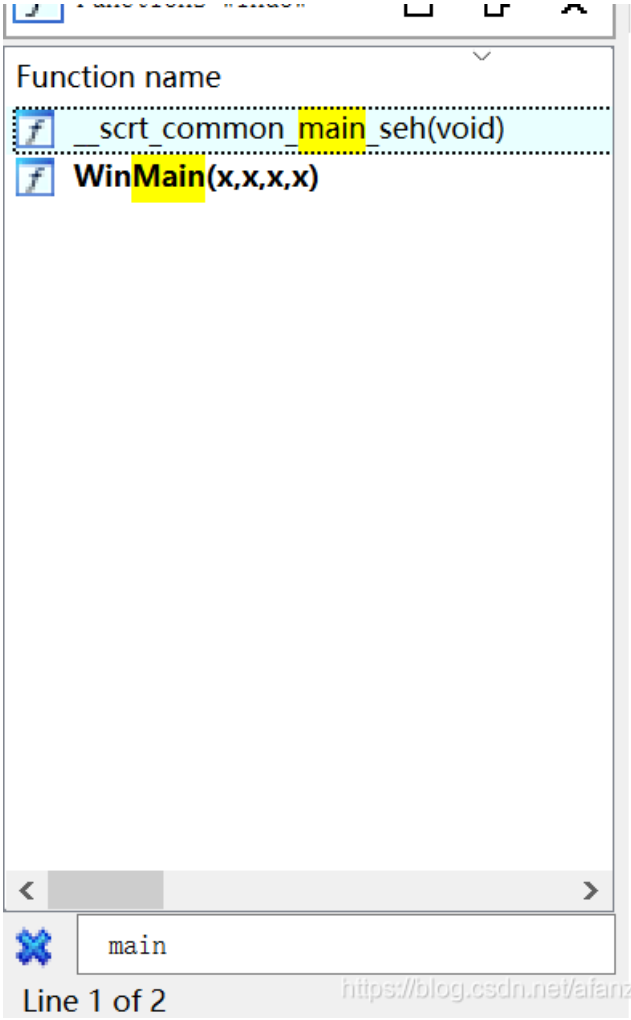
.rdata:00411...	0000000D	C	ADVAPI32.dll
.data:004121...	0000001B	C	
.data:004122...	0000001B	C	abcdefghijklmnopqrstuvwxyz
.data:004122...	0000001B	C	ABCDEFGHIJKLMNOPQRSTUVWXYZ
.data:004124...	00000009	C	
.data:004124...	0000000A	C	
.data:004124...	0000001B	C	abcdefghijklmnopqrstuvwxyz
.data:004125...	0000001B	C	ABCDEFGHIJKLMNOPQRSTUVWXYZ

看到几个abcdef26个字母，感觉有可疑之处。

跟进之后，发现就是几个字母。

（2）找主main函数

既然字符串窗口没有找到有用信息，那么跳转主main函数去寻找。

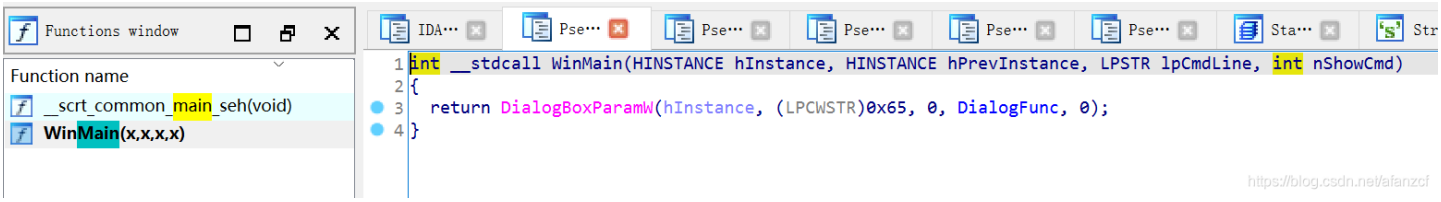

Function name
__scrt_common_main_seh(void)
WinMain(x,x,x,x)
main
Line 1 of 2
https://blog.csdn.net/afanzcf

反编译第一个函数。

```
int __usercall __scrt_common_main_seh@<eax>(i
{
  _DWORD *v3; // eax
  _DWORD *v4; // esi
  _tls_callback_type *v5; // eax
  _tls_callback_type *v6; // esi
  CHAR *v7; // eax
  char *v8; // esi
  int v9; // [esp-14h] [ebp-48h]
  char v10; // [esp+10h] [ebp-24h]

  if ( __scrt_initialize_crt(1) )
    goto LABEL_3;
  do
  {
    __scrt_fastfail(a1, a2, 7u);
LABEL_3:
    LOBYTE(a1) = 0;
    v10 = __scrt_acquire_startup_lock();
  }
  while ( dword_412ADC == 1 );
  if ( dword_412ADC )
  {
    LOBYTE(a1) = 1;
  }
  else
  {
    dword_412ADC = 1;
    if ( _initterm_e((_PIFV *)&dword_40C140,
      return 255;
    _initterm((_PVFV *)&First, (_PVFV *)&Last
    dword_412ADC = 2;
  }
  __scrt_release_startup_lock(v10);
  v3 = (_DWORD *)sub_401CAF();
  v4 = v3;
  if ( *v3 && (unsigned __int8)__scrt_is_nonw
    ((void (__thiscall *)(_DWORD, _DWORD, int
  v5 = (_tls_callback_type *)sub_401CB5();
  v6 = v5;
  if ( *v5 && (unsigned __int8)__scrt_is_nonw
    _register_thread_local_exe_atexit_callbac
  unknown_libname_1(0);
  v9 = (unsigned __int16)__scrt_get_show_wind
  v7 = (CHAR *)unknown_libname_8();
  v8 = (char *)WinMain((HINSTANCE)0x400000, 0
  unknown_libname_3(0);
  if ( !__scrt_is_managed_app() )
    _loaddll(v8);
  if ( !(_BYTE)a1 )
    _cexit();
```


并没有发现什么有用信息。



点开第二个函数，把注意都放在了粉红色字符上。

跟进之后，没有得到结果。

```
.idata:0040C124 ; HWND __stdcall GetDlgItem(HWND hDlg, int nIDDlgItem)
.idata:0040C124                 extrn GetDlgItem:dword  ; CODE XREF: DialogFunc+312↑p
.idata:0040C124                                         ; DATA XREF: DialogFunc+312↑r
.idata:0040C128 ; INT_PTR __stdcall DialogBoxParamW(HINSTANCE hInstance, LPCWSTR lpTemplateName, HWN
.idata:0040C128                 extrn DialogBoxParamW:dword
.idata:0040C128                                         ; CODE XREF: WinMain(x,x,x,x)+11↑p
.idata:0040C128                                         ; DATA XREF: WinMain(x,x,x,x)+11↑r
.idata:0040C12C
```

分析到这里的时候，分析不动了（其实还是自己思路没打开）

观看大哥的wp之后

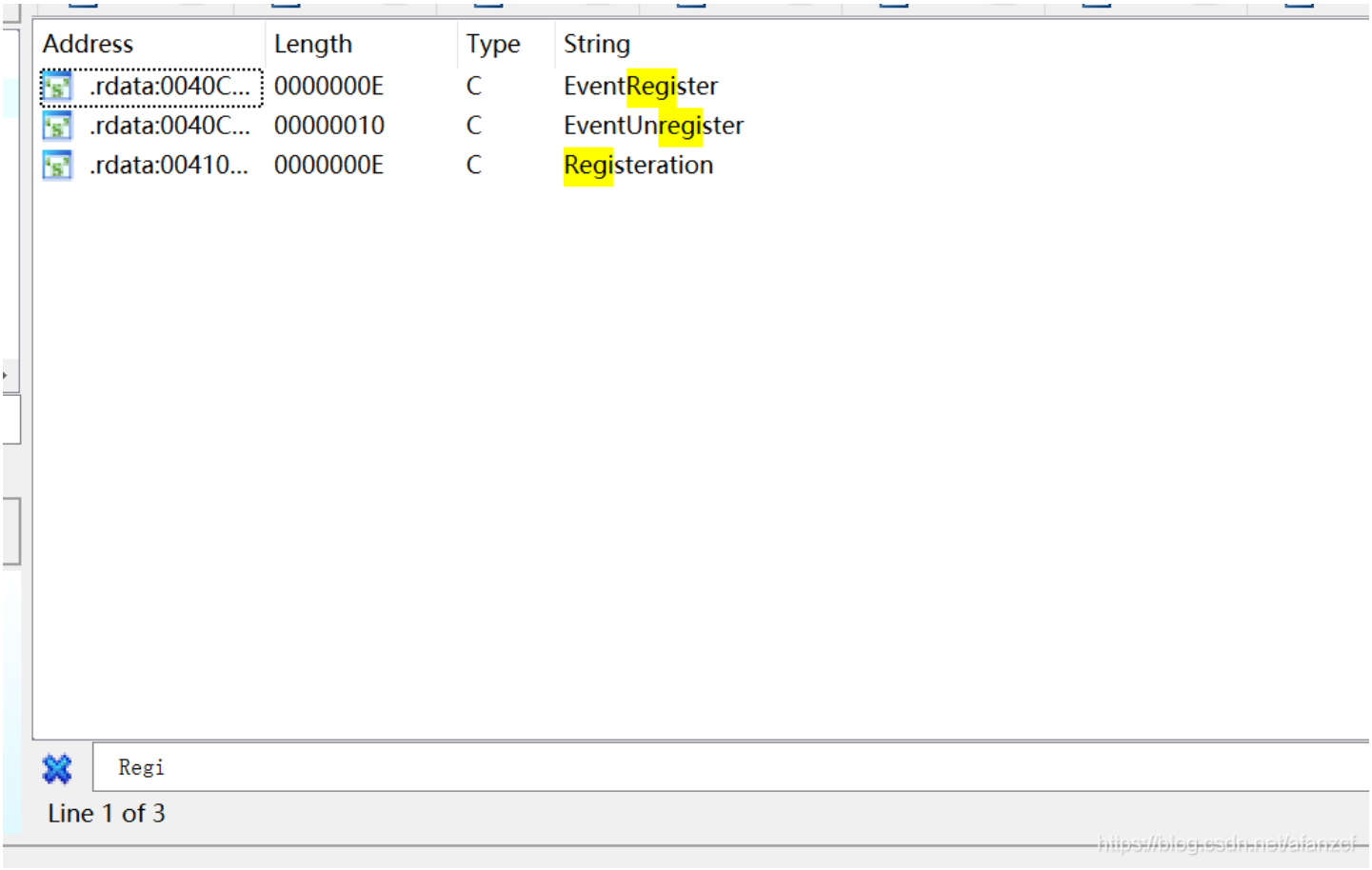
发现问题所在。

这里突然有所感悟！！！

在自己打开程序，瞎乱输入之后，其实弹出来了一个弹窗。

但是并没有在字符串窗口搜索这个字符串。

我们在字符串窗口搜索一下。



可以看到在，最后，发现了这个字符串。我们跟进去。

```
.rdata:00410A60 asc_410A60      db '.',0                ; DATA XREF: DialogFunc+EA↑o
.rdata:00410A60                                         ; DialogFunc+102↑o
.rdata:00410A62                 align 4
.rdata:00410A64 ; const CHAR Caption[]
.rdata:00410A64 Caption         db 'Registeration',0    ; DATA XREF: DialogFunc+2B8↑o
.rdata:00410A72                 align 10h
.rdata:00410A80 xmmword_410A80  xmmword 2067616C662072756F590A2173736563h
.rdata:00410A80                                         ; DATA XREF: DialogFunc+15C↑r
.rdata:00410A90 xmmword_410A90  xmmword 637553206E6F697461727473696676552h
.rdata:00410A90                                         ; DATA XREF: DialogFunc+14B↑r
.rdata:00410AA0 xmmword_410AA0  xmmword 696166206E6F697461727473696676552h
.rdata:00410AA0                                         ; DATA XREF: DialogFunc+130↑r
.rdata:00410AB0 ; Debug Directory entries
.rdata:00410AB0                 dd 0                    ; Characteristics
.rdata:00410AB4                 dd 56B31802h            ; TimeDateStamp: Thu Feb 04 09:21:06 2016
.rdata:00410AB8                 dw 0                    ; MajorVersion
.rdata:00410ABA                 dw 0                    ; MinorVersion
.rdata:00410ABC                 dd 2                    ; Type: IMAGE_DEBUG_TYPE_CODEVIEW
.rdata:00410AC0                 dd 65h                  ; SizeOfData
```

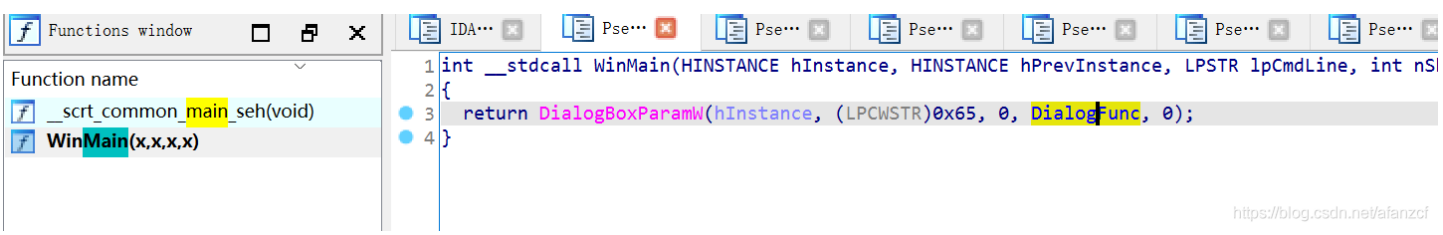
交叉引用。再反编译。

```
INT_PTR __stdcall DialogFunc(HWND hDlg, UINT a2, WPARAM a3, LPARAM a4)
{
  HMODULE v5; // eax
  HICON v6; // eax
  HMODULE v7; // eax
  HWND v8; // eax
  HCURSOR v9; // [esp-4h] [ebp-34Ch]
  CHAR String[256]; // [esp+8h] [ebp-340h] BYREF
  CHAR v11[256]; // [esp+108h] [ebp-240h] BYREF
  CHAR Text[256]; // [esp+208h] [ebp-140h] BYREF
  char Source[60]; // [esp+308h] [ebp-40h] BYREF

  if ( a2 == 16 )
  {
    EndDialog(hDlg, 0);
    return 0;
  }
  if ( a2 == 272 )
  {
    v5 = GetModuleHandleW(0);
    v6 = LoadIconW(v5, (LPCWSTR)0x67);
    SetClassLongA(hDlg, -14, (LONG)v6);
    v7 = GetModuleHandleW(0);
    v9 = LoadCursorW(v7, (LPCWSTR)0x66);
    v8 = GetDlgItem(hDlg, 1);
    SetClassLongA(v8, -12, (LONG)v9);
    return 1;
  }
  if ( a2 != 273 || (unsigned __int16)a3 != 1 )
    return 0;
  memset(String, (unsigned __int16)a3 - 1, sizeof(String));
  memset(v11, 0, sizeof(v11));                  // 作用是将某一块内存中的内容全部设置为指定的值， 这个函数通常为新申请的内存做初始化工作。
  memset(Text, 0, sizeof(Text));
  GetDlgItemTextA(hDlg, 1001, String, 256);
  GetDlgItemTextA(hDlg, 1002, v11, 256);        // 上面的一大串代码是在初始化
  if ( strstr(String, "@") && strstr(String, ".") && strstr(String, ".")[1] && strstr(String, "@")[1] != '.' )//
                                                // strstr作用是返回字符串中首次出现子串的地址
  {
    strcpy(&Source[36], "Registration failure.");
    strcpy(Source, "Registration Success!\nYour flag is:");
    if ( strlen(v11) == 16
      && v11[0] == 'C'
      && v11[15] == 'X'
      && v11[1] == 'Z'
      && v11[14] == 'A'
      && v11[2] == '9'
      && v11[13] == 'b'
      && v11[3] == 'd'
      && v11[12] == '7'
      && v11[4] == 'm'
      && v11[11] == 'G'
      && v11[5] == 'q'
      && v11[10] == '9'
      && v11[6] == '4'
      && v11[9] == 'g'
      && v11[7] == 'c'
      && v11[8] == '8' )
    {
      strcpy_s(Text, 0x100u, Source);
      strcat_s(Text, 0x100u, v11);
    }
    else
    {
      strcpy_s(Text, 0x100u, &Source[36]);
    }
  }
  else
  {
    strcpy_s(Text, 0x100u, "Your E-mail address in not valid.");
  }
  MessageBoxA(hDlg, Text, "Registeration", 0x40u);
  return 1;
```


找到关键函数所在。

以上是我突发奇想得到的经验，其实在自己的上一步找main函数的时候，仔细的一点也能找到这里。



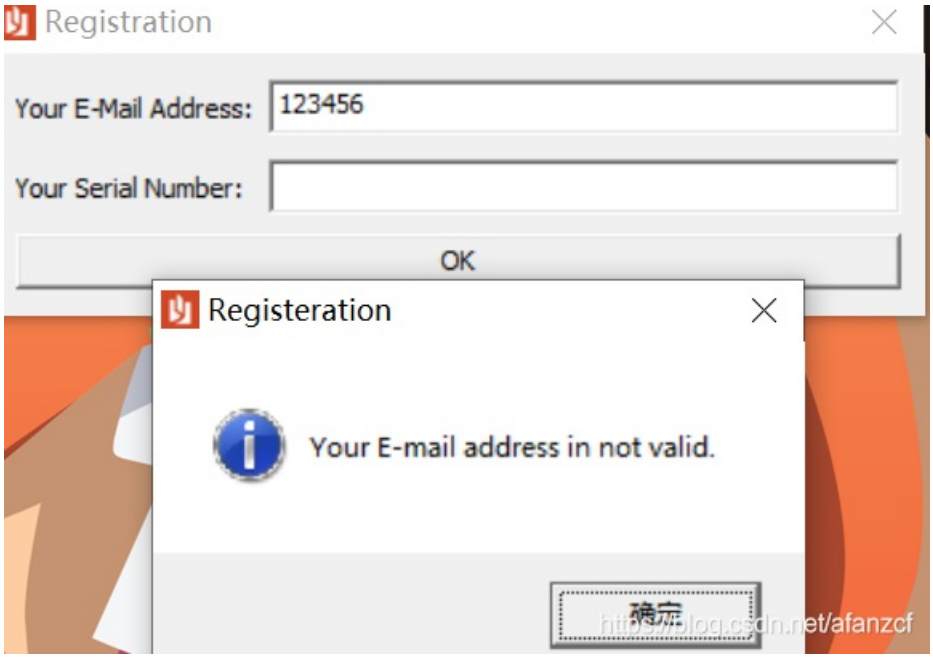
这个函数就是关键函数。

下面分析这个函数。

if语句的前面。memset函数，百度一手，

```
memset函数的作用：将某一块内存中的内容全部设置为指定的值， 这个函数通常为新申请的内存做初始化工作。
```

也就是从if语句前面，应该都是初始化，if语句中，看到@和 .应该是在判断邮箱是否正确。



我们可以看到，strcpy函数把Success这个信息放到了Source里面，也就是Source里面存的就是flag

看到Source的前面是一串字符，一开始没注意顺序，直接写了CXZA9bd7mGq94gcg

但是不对，后面才注意，这个数组有顺序，按顺序再写一遍。得到

```
flag：CZ9dmq4c8g9G7bAX
```

总结

在程序中，最关键的函数或许不是main函数，其次，在寻找关键字符串的时候，不能毫无目的的去寻找，像这个题一样，一开始是毫无目的，但是在明有错误提醒的字符串之后，还没注意到问题的严重性，还是瞎猫撞耗子的寻找，最后还没找到。还是得多分析分析啊。