idapython_攻防世界no-strings-attached_逆向之旅007

原创

Nicolas.Alan 于 2021-01-07 23:42:53 发布 209 收藏 1

提示：这篇文章里涉及到：idapython，gdb，**idapython中文手册.pdf百度云链接**

攻防世界no-strings-attached_逆向之旅007

前言

本题的摘要：
文件是linux下的elf文件，但是运行报错。有两个解题思路，第一种是用gdb在linux虚拟机下动态调试，第二种是使用ida静态分析，然后构建脚本计算出flag。

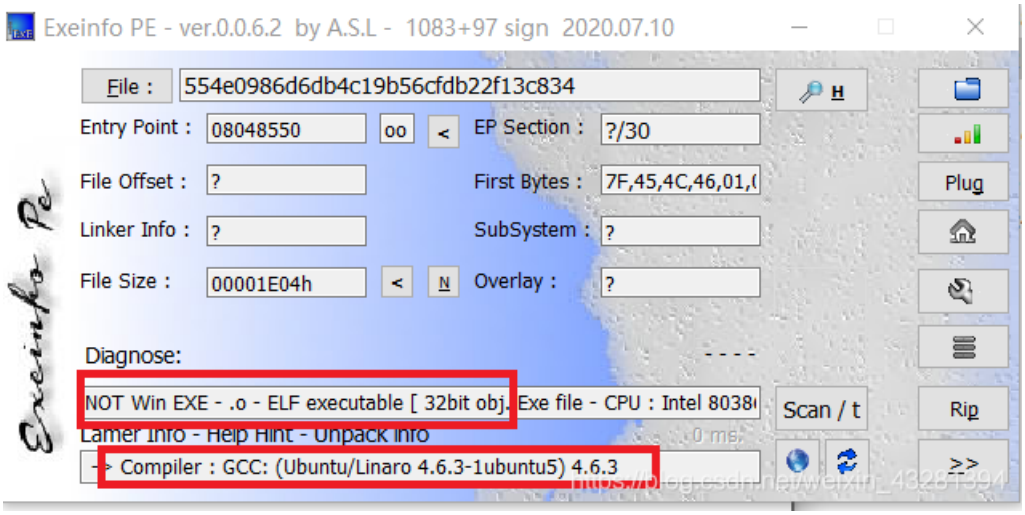
今天距离写上一篇博客有13天了，终于可以做逆向题并写博客了，前面一段时间在应对恶意代码考试及编译和反编译的考试。当然了，还有元旦假期的快乐开黑生活。

一、攻防世界no-strings-attached

在这里给出题目链接：https://adworld.xctf.org.cn/task/answer?type=reverse&number=4&grade=0&id=5080&page=1

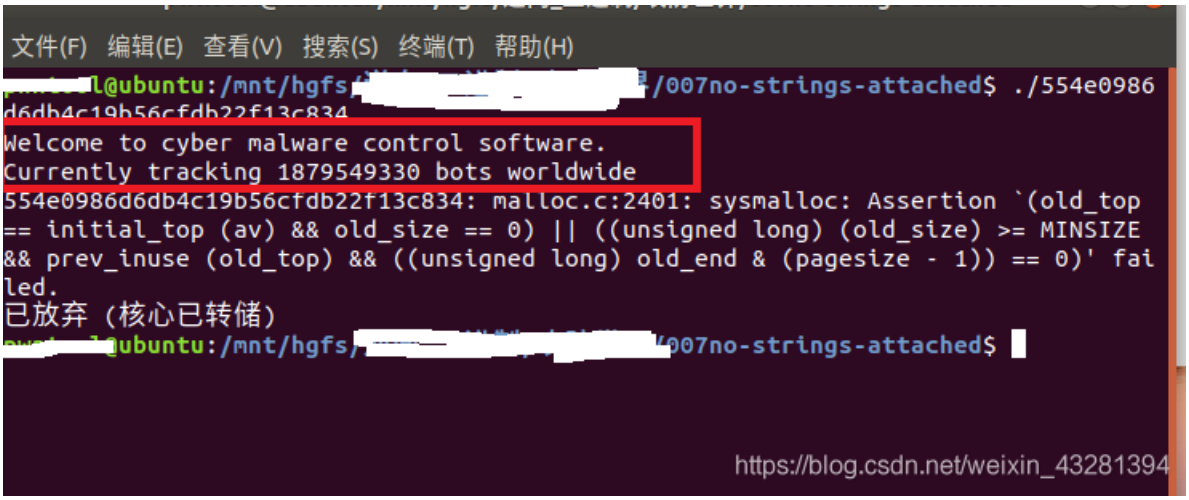
二、writeup

1.用exeinfo看一下



可得到信息：elf文件，32位，使用gcc 4.6.3编译

2.在linux下运行

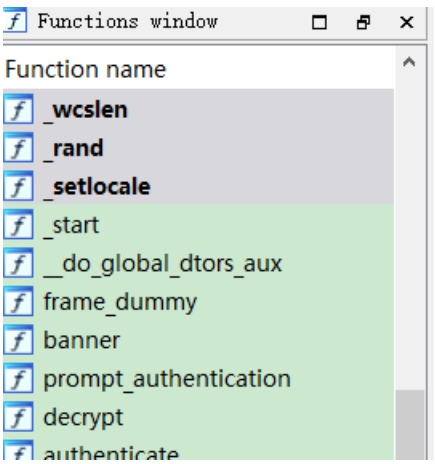


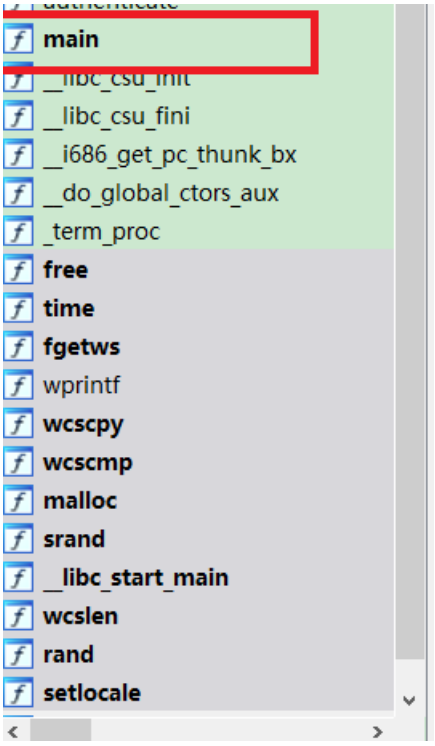
发现程序会输出两句话，然后报错。

3.分析

对于此题，我选择使用第二种思路，即使用ida静态分析，然后写脚本计算flag。对于另一种思路：使用gdb调试，我后面会写一篇使用gdb解其他题目的博客。

这个题还是一如既往的比较好找到main函数。





F5反编译

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  setlocale(6, &locale);
  banner();                                     // 输出两句话，以及产生的随机数
  prompt_authentication();                      // 让程序输出：please。。。。
  authenticate();
  return 0;
}
```


通过分析，可以确定flag是在authenicate（）中，着重分析该函数：

```
void authenticate()
{
  wchar_t ws[8192]; // [esp+1Ch] [ebp-800Ch] BYREF
  wchar_t *s2; // [esp+801Ch] [ebp-Ch]

  s2 = decrypt((wchar_t *)&s, (wchar_t *)&dword_8048A90);
  if ( fgetws(ws, 0x2000, stdin) )              // 输入的字符串存放在ws中。
  {
    ws[wcslen(ws) - 1] = 0;
    if ( !wcscmp(ws, s2) )
      wprintf((int)&unk_8048B44);               // 让程序输出：success
    else
      wprintf((int)&unk_8048BA4);               // 让程序输出：access fenied（失败）
  }
  free(s2);
}
```


分析过程见图中的注释，可以知道flag是在s2中存着。加下来对decrypt函数进行分析。

```
// 143ah ,1401h
wchar_t *__cdecl decrypt(wchar_t *s, wchar_t *a2)
{
```

```
{
  size_t v2; // eax
  signed int v4; // [esp+1Ch] [ebp-1Ch]
  signed int i; // [esp+20h] [ebp-18h]
  signed int v6; // [esp+24h] [ebp-14h]
  signed int v7; // [esp+28h] [ebp-10h]
  wchar_t *dest; // [esp+2Ch] [ebp-Ch]

  v6 = wcslen(s);
  v7 = wcslen(a2);
  v2 = wcslen(s);
  dest = (wchar_t *)malloc(v2 + 1);
  wcscpy(dest, s);
  while ( v4 < v6 )
  {
    for ( i = 0; i < v7 && v4 < v6; ++i )
      dest[v4++] -= a2[i];
  }
  return dest;
}
```


该函数所描述的步骤即为计算flag的方法，接下来编写脚本来计算flag。
首先使用idapytohn获得参数s，dword_8048A90的值，这两个参数的类型为wchar_t数组，wchar_t：

> wchar_t是C/C++的字符数据类型，是一种扩展的字符存储方式。 在Windows下，wchar_t占2个字节（byte）；
> 在Linux下，wchar_t占4个字节。
> wchar_t类型主要用在国际化程序的实现中，但它不等同于Unicode编码。Unicode编码的字符一般以wchar_t类型存储。

这个程序是linux下的，所以wchar_t是dd类型（double word ）占4字节。接下来使用idapython将s和dword_8048A90提取出来：
（注意！！！ 我用的是idapro7.5，使用其他版本的ida，脚本会不一样的，如何编写idapython脚本呢，我这有**idapython中文手册.pdf**：
百度云链接：链接：https://pan.baidu.com/s/1B0qNeQSQHeWPaUZqJfYMFQ
提取码：0dkl

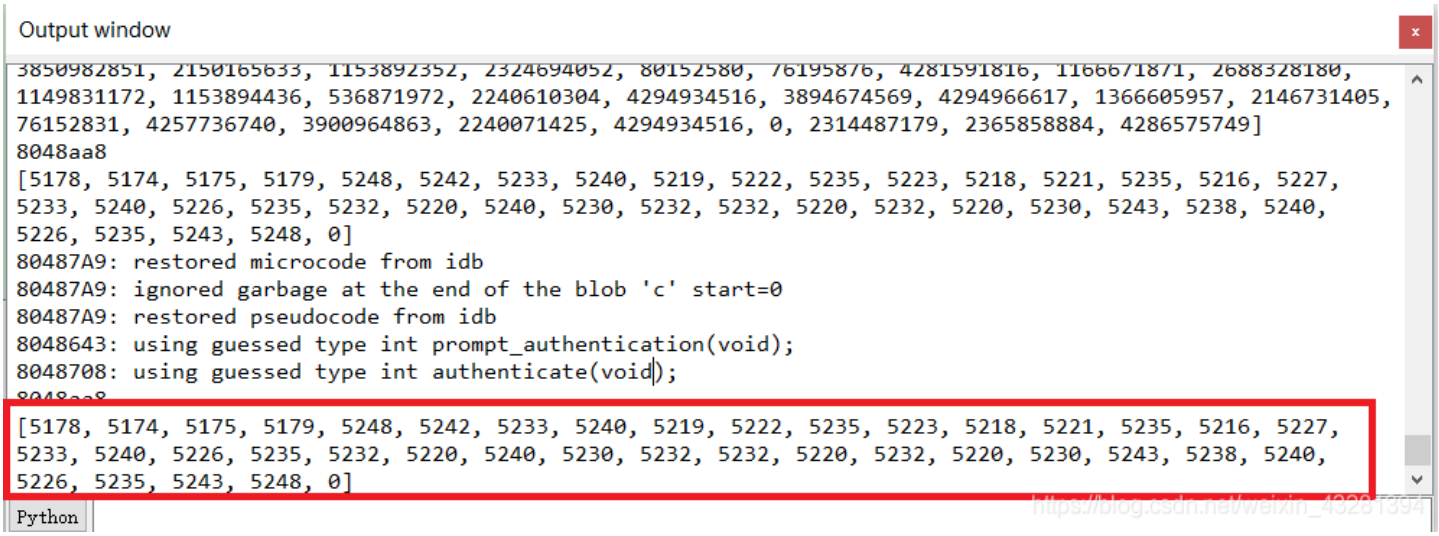
）
提取参数s:

```
from ida_bytes import get_dword as dd
addr = here()
print("%x"%addr)
arr2 =[]
for i in range(39):
    arr2.append(dd(addr+4*i))
print(arr2)
```

运行该脚本时，将鼠标点击s，然后再运行，这样的话addr的值就是s的初始地址了。至于39是怎么算出出来的，一个dd占4个字节，s总长度是0x8048B44 - 0x8048BAA8 = 0x9c字节,换算为十进制是156字节，然后156/4 = 39，即s有39个元素，每一个元素占4个字节。

PS：如何运行该脚本呢？

shift+f2,然后scripting lanuage选择Python，然后编写脚本，编写好之后，点击run即可看到运行结果



得到：在这里插入代码片

```
s = [5178, 5174, 5175, 5179, 5248, 5242, 5233, 5240, 5219, 5222, 5235, 5223, 5218, 5221, 5235, 5216, 5227, 5233,
 5240, 5226, 5235, 5232, 5220, 5240, 5230, 5232, 5232, 5220, 5232, 5220, 5230, 5243, 5238, 5240, 5226, 5235, 524
3, 5248,0]
```

同理：

第二个参数：

```
dword_8048A90= [5121, 5122, 5123, 5124, 5125,0]
```

接下来编写脚本：

```
arg1 = [5178, 5174, 5175, 5179, 5248, 5242, 5233, 5240, 5219, 5222, 5235, 5223, 5218, 5221, 5235, 5216, 5227, 52
33, 5240, 5226, 5235, 5232, 5220, 5240, 5230, 5232, 5232, 5220, 5232, 5220, 5230, 5243, 5238, 5240, 5226, 5235,
5243, 5248,0]
arg2 = [5121, 5122, 5123, 5124, 5125,0]
v6 = len(arg1)-1
v7 = len(arg2)-1
v2 = v6
dest = arg1
v4 = 0
while(v4<v6):
    i=0
    while(i<v7 and v4<v6):
        dest[v4]=dest[v4]-arg2[i]
        i=i+1
        v4=v4+1
flag=""
for i in dest:
    try:
        flag=flag+chr(i)
    except:
        pass

print(flag)
```

运行结果：

```
D:\Pycharmprojects\006exp\Scripts\python.exe D:/Pycharmprojects/006exp/007exp.py
9447{you_are_an_international_mystery}

Process finished with exit code 0
```

注意：为啥v6，v7要len（）-1呢？
因为wcslen函数：

> 在头文件<wchar.h>中定义size_t wcslen（const wchar_t * str）;(1)（自C95以来）size_t wcsnlen_s（const wchar_t * str，size_t strsz）;(2)（自C11以来）
> 1）返回宽字符串的长度，即**在终止空宽字符之前的非空宽字符数**。

所以wcslen_s的返回值没有算上数组中最后的0，要减1 。

PS：

> 1.有错误的地方，请私信我，感谢指正。
> 2.或有不懂的地方，请私信我，我会及时回复。
> 3.文章中涉及到的所有工具，有的在我之前写的文章里有百度云链接，或者你可以私信我，我邮件或者百度云分享给你。我们一起进步，一起加油。
> 4.如果你觉得这篇文章对你用，请点个赞，鼓励一下我吧，谢谢！

三、总结

(1)db定义*斜体样式*字节类型变量，一个字节数据占1个字节单元，读完一个，偏移量加1
dw定义字类型变量，一个字数据占2个字节单元，读完一个，偏移量加2
dd定义双字类型变量，一个双字数据占4个字节单元，读完一个，偏移量加4

(2)wcslen
在头文件<wchar.h>中定义
size_t wcslen（const wchar_t * str）;(1)（自C95以来）
size_t wcsnlen_s（const wchar_t * str，size_t strsz）;(2)（自C11以来）
1）返回宽字符串的长度，即在终止空宽字符之前的非空宽字符数。

(3)idapython的使用
idapython中文手册.pdf：
百度云链接：链接：https://pan.baidu.com/s/1B0qNeQSQHeWPaUZqJfYMFQ
提取码：0dkl

(4).坚持！坚持！ 坚持！