pwnable.kr codemap writeup / OD 脚本解题

原创

attack_eg 于 2017-09-12 16:49:19 发布 304 收藏

分类专栏： hack之路 文章标签： OllyDBG pwnable writeup ODBGScript

本文链接：https://blog.csdn.net/attack_eg/article/details/77947031

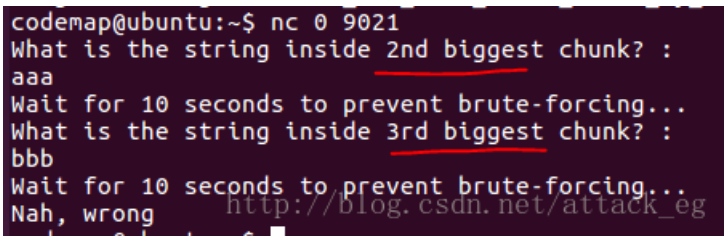
核心考察点

- IDA静态分析能力
- OD脚本编写能力

> - 推荐OD脚本编辑器loveboom大大写的OllyScript Editor（下载地址http://tools.pediy.com/windows/Debuggers/debug_tools/ollyscipt_editor/OSCEditor.rar）。里面包含很多常用命令的英文帮助。中文帮助可参见http://blog.csdn.net/whatday/article/details/8553391。
> - 另看雪上有丰富的OD脚本实例。（http://tools.pediy.com/windows/debuggers_od_script.htm）

问题

> 运行*nc 0 9021*,程序要求输入第二大内存块中的字串和第三大内存块中的字串。



> 本地运行codemap,程序声称生成了1000个随机大小的堆块，每个堆块中存放一个随机的字符串，并且输出了最大长度字符串。

```
C:\Users\dave\Desktop>codemap.exe
I will make 1000 heap chunks with random size
each heap chunk has a random string
press enter to start the memory allocation

the allcated memory size of biggest chunk is 99879 byte
the string inside that chunk is X12nM7yCJcu0x5u
log in to pwnable.kr and anwer some question to get flag.
```


解题过程

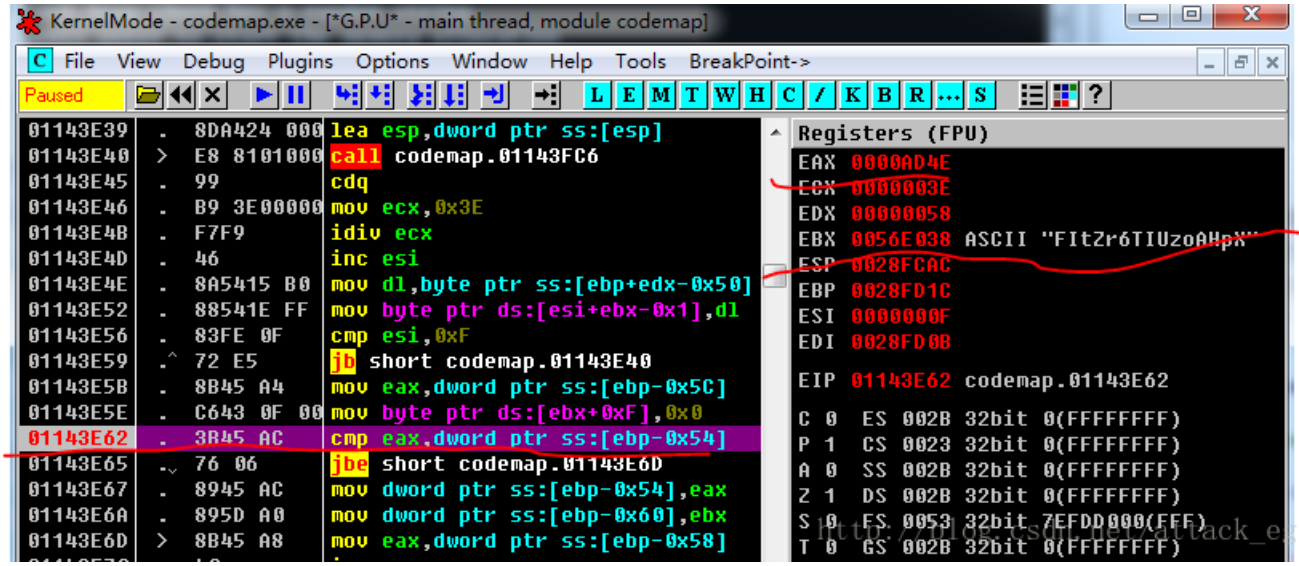
> IDA打开codemap，找到提示的0x403E65地址处代码。

```
text:00403E56                   cmp     esi, 0Fh
text:00403E59                   jb      short loc_403E40
text:00403E5B                   mov     eax, [ebp+CurrentSize]
text:00403E5E                   mov     byte ptr [ebx+0Fh], 0
text:00403E62                   cmp     eax, [ebp+biggestChunkSize]
text:00403E65                   jbe     short loc_403E6D
text:00403E67                   mov     [ebp+biggestChunkSize], eax
text:00403E6A                   mov     [ebp+biggestChunkStr], ebx
```

> 在反编译代码中我们可以看到程序的随机种子是固定的，因此产生的伪随机字串以及伪随机大小每次程序运行都是一定的。（多运行两次codemap也能通过最大size的大小和字串发现）

```
  CurrentSize = v9 % 0x186A0;
  Random_String = (char *)malloc(v9 % 0x186A0);
  if ( v10 >= 0x10 )
  {
    qmemcpy(&v18, "abcdefghijklmnopqrstubwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890", 0x3Fu);
    len_str = 0;
    do
      Random_String[++len_str - 1] = *(&v18 + rand() % 62);
    while ( len_str < 0xF );
    Random_String[15] = 0;
    if ( CurrentSize > biggestChunkSize )
    {
      biggestChunkSize = CurrentSize;
      biggestChunkStr = Random_String;
    }
  }
  if ( ++cnt >= 0x3E8 )
    break;
  srand(cnt);
}
```

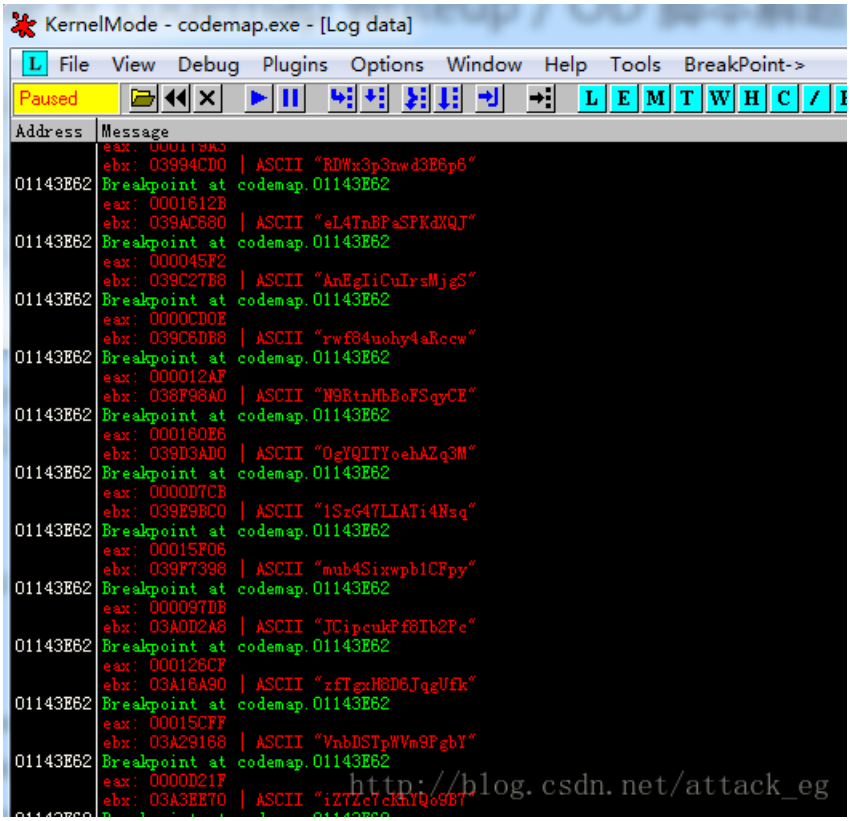

> 在OD中对提示的地址下断（程序启用了ASLR，在相对同样地址下断即可）。运行发现eax与ebx即为随机内存size和随机字串（IDA中静态分析也能获知）。（实际下断在前一条指令，效果一样，因为此时eax与ebx值已指向内存size和随机字串）



因为有1000次分配，人工不现实，直接想到了OD的脚本插件ODBGScript。（第一次写的OD脚本，虽然最后很简单，还是费了好大劲QwQ）

```
bp 1013E62 //(根据载入地址变化)
loop:
    run
    log eax
    log ebx
    jmp loop
```

注：eax与ebx的值已及提示信息将输出到log窗口。在log窗口提前右击选择”log to file”，将输出到文件。（log窗口缓存不够）



接下来就用python脚本处理，对内存size大小排序，找到第二大和第三大内存块存储的字串就好了。（python脚本处理时，发现OD脚本只获取到999个数据！！差一个最后也不知道咋回事，还好1/500的概率没撞上QwQ）

```
import re
f = open("resss.txt","r") #log重定向的文件
re1 = re.compile(r'[0-9A-Fa-f]{8}')
re2 = re.compile(r'[0-9a-zA-Z]{15}')
lis = []
sizeLis = []
for line in f.readlines():
    if "eax" in line:
        res = re1.findall(line)
        sizeLis.append(int(res[0],16))
        #print res
        lis.append(int(res[0],16))
    elif "ebx" in line:
        res = re2.findall(line)
        #print res
        #print lis
        lis = lis + res
#print len(lis)
#print len(sizeLis)
sizeLis.sort()
secondBigSiz = sizeLis[-2]
thirdBigSiz = sizeLis[-3]
print "second Biggest size string:",lis[lis.index(secondBigSiz)+1]
print "third Biggest size string:",lis[lis.index(thirdBigSiz)+1]
```

程序运行结果：

```
In [31]: %run "C:/U...ckMap/sol_codemap.py"
second Biggest size string: roKBk...GMUKrMb
third Biggest size string: 2ckbn...csMA2s
```


> 在远程服务器上提交即获得flag：

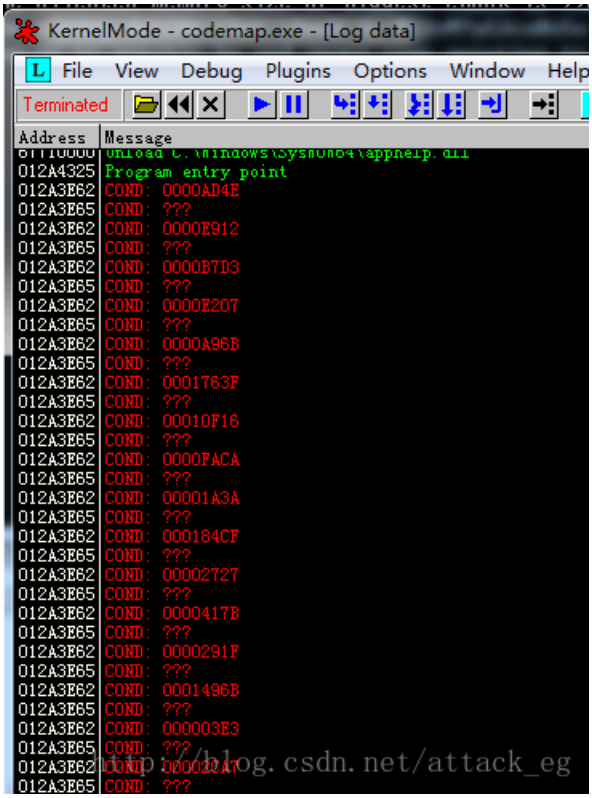
```
codemap@ubuntu:~$ nc 0 9021
What is the string inside 2nd biggest chunk? :
roKBk...GMUKrMb
Wait for 10 seconds to prevent brute-forcing...
What is the string inside 3rd biggest chunk? :
2ckbn...csMA2s
Wait for 10 seconds to prevent brute-forcing...
Congratz! flag : select_eax_from_...._desc_limit_20
```


总结

- 解题的过程并非一帆风顺，初始准备用bpl来获取0x403E65处eax的值以及ebx指向的字串：

```
bpl 403e62,"eax"
bpl 403e65,"string [ebx]"
run
```

> *string [ebx]* 表达式得到的值是???，地址解析失败。单点调试的时候是能够获取到字串值的，脚本连续运行可能导致ebx值迅速变化取值错误。（这是我的猜测）
> bpl命令是利用设置条件记录断点（*shift+F4*）。此处eax值在后面与第二个脚本的结果相同，取值正确。
> 可能这是OD的一个待完善的地方。大神可留言帮助我理解。



- 解题后在网上搜题解，才发现自己的想法并不是标准解题思路。网上大多数解题思路是使用IDA脚本（IDC或IDAPython）。也看到一种比较另类的解法，利用vs2015的追踪堆分配情况的功能来解题（https://www.52pojie.cn/forum.php?mod=viewthread&tid=593201）。