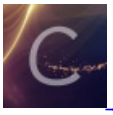
安装使用Angr符号执行来求解CTF逆向题

原创

iqiqiya 于 2018-09-23 22:22:27 发布 3379 收藏 4
分类专栏： 我的逆向之路 我的CTF之路 我的CTF进阶之路 文章标签： Angr符号执行来求解CTF逆向题 Angr符号执行 求解CTF逆向题


我的逆向之路 同时被 3 个专栏收录
108 篇文章 10 订阅
订阅专栏


我的CTF之路
92 篇文章 5 订阅
订阅专栏

我的CTF进阶之路
108 篇文章 18 订阅
订阅专栏

angr是什么　我的理解就是　可以用它来帮我们从众多的路中找到最正确的那条（太复杂的话就不行了 执行效率是个问题）

类似与z3来解方程似的(不清楚的话可以看这个怎样在win10上安装并食用Z3库来解CTF方程)

github地址https://github.com/angr/angr

angr 安装：

```
#首先安装依赖
sudo apt-get install python-dev libffi-dev build-essential virtualenvwrapper
#安装angr
mkvirtualenv angr && pip install angr
```

如果还不能import angr

那就 **sudo pip install angr**

下面有两道CTF逆向题

0x01:交互式输入(就是提示你输入 然后换行等待输入)

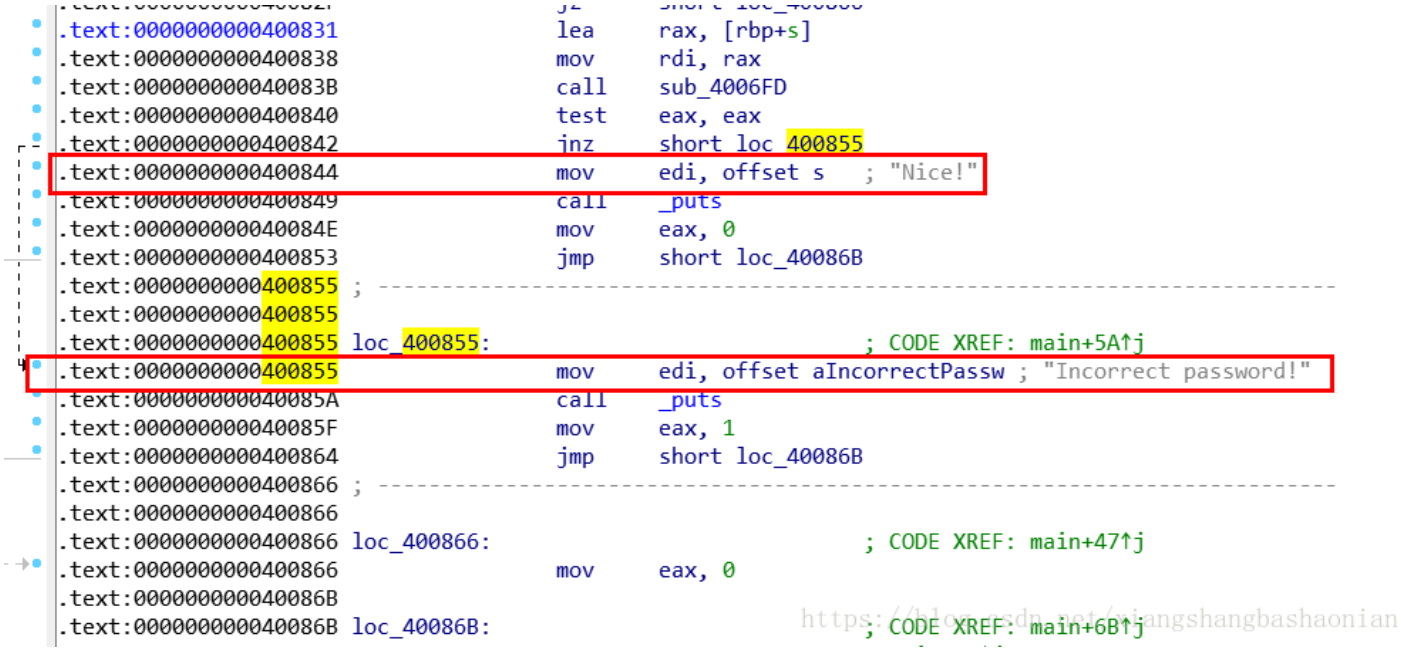
0x02:输入作为参数运行(./rev3 1234)

先占个坑　以后慢慢分析

0x01:

可以先file一下看看　看到是ELF　x64

再载入IDA x64




```
In [2]: import angr
WARNING | 2018-09-23 23:00:21,741 | angr.analyses.disassembly_utils | Your version of capstone does not sup

In [3]: proj = angr.Project("./r100",auto_load_libs=False) #貌似这个只能在根目录下  要不然最后无法输出

In [4]: state = proj.factory.entry_state() #这个我理解的就是保存入口状态

In [5]: simgr = proj.factory.simgr(state)

In [6]: simgr.explore(find=0x400844,avoid=0x400855) #find就是正确的那条路 avoid错误的
Out[6]: <SimulationManager with 1 found, 2 active, 12 avoid> #找到了一条路径

In [7]: simgr.found[0].posix.dumps(0) #Code_Talkers就是flag
Out[7]: 'Code_Talkers\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\xf5\x
```

这道题目的正常做法可以看看这个

【WhaleCTF逆向题】第二期入门语言【初级】writeup

0x02:

可以先file一下看看 看到是ELF x64

再载入IDA x64

```
.text:00000000004005F6                 mov     rdi, rax
.text:00000000004005F9                 call    verify
.text:00000000004005FE                 test    eax, eax
.text:0000000000400600                 jz      short loc_40060E
.text:0000000000400602                 mov     edi, offset aCorrectThatIsT ; "Correct! that is the secret key!"
.text:0000000000400607                 call    _puts
.text:000000000040060C                 jmp     short loc_400618
.text:000000000040060E ; ---------------------------------------------------------------------------
.text:000000000040060E
.text:000000000040060E loc_40060E:                             ; CODE XREF: main+3B↑j
.text:000000000040060E                 mov     edi, offset aIMSorryThatSTh ; "I'm sorry, that's the wrong secret key!"
.text:0000000000400613                 call    _puts
.text:0000000000400618
.text:0000000000400618 loc_400618:                             ; CODE XREF: main+47↑j
.text:0000000000400618                 mov     eax, 0
.text:000000000040061D
.text:000000000040061D locret_40061D:                          ; CODE XREF: main+24↑j
.text:000000000040061D                 leave
.text:000000000040061E                 retn
```



```
(angr) root@kali:/# python
Python 2.7.15+ (default, Aug 31 2018, 11:56:52)
[GCC 8.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import angr   #导入angr
WARNING | 2018-09-24 02:00:03,499 | angr.analyses.disassembly_utils | Your version of capstone does not sup
>>> import claripy    #导入claripy
>>> proj = angr.Project("./ais3_crackme")    #载入文件
>>> argv1 = claripy.BVS('argv1',50*8)    #B是bit 1字节=8bit  猜测输入不多于50字节 就是50*8
>>> state = proj.factory.entry_state(args=['./ais3_crackme',argv1])
>>> simgr = proj.factory.simgr(state)
>>> simgr.explore(find=0x400602,avoid=0x40060E)    #成功位置及失败位置
<SimulationManager with 1 found, 4 active, 46 avoid>
>>> print simgr.found[0].solver.eval(argv1)    #转成ascll码输出
98258555959157598537470557556759296005297673422675047953095862682125718585910497108264649432140497523405083
>>> print simgr.found[0].solver.eval(argv1,cast_to=str)    #直接输出字符
ais3{I_tak3_g00d_n0t3s}
>>>
```

可以看看：http://www.freebuf.com/sectool/143056.html

http://www.360doc.com/content/18/0307/21/31784658_735232878.shtml