《漏洞利用及渗透测试基础》实验报告

姓名：蒋薇　　学号：2110957　班级：张健老师班

实验名称：

angr 实验

实验要求：

复现 sys-write 示例的两种 angr 求解方法,就如何使用 angr 以及怎么解决一些实际问题做一些探讨

实验过程：

安装：安装 python,pip install angr

sys-write.txt 文件

```
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
#include <stdio.h>

char u=0;
int main(void)
{
        int i, bits[2]={0,0};
        for (i=0; i<8; i++) {
                bits[(u&(1<<i))!=0]++;
        }
        if (bits[0]==bits[1]) {
                printf("you win!");
        }
        else {
                printf("you lose!");
        }
        return 0;
}
```

方法一：

```
import angr
import claripy

def main():
#新建一个工程，导入二进制文件，选项是不自动加载依赖项，不会自动载入依赖的库
p = angr.Project( ‘./issue’,load_options = {“auto_load_libs”:False})

#初始化模拟程序状态的 SimState 对象 state,该对象包含了程序内存、寄存器、符号信息等模拟运行时的动态数据
#blank_state():可通过给定参数 addr 的值指定程序起始运行地址
#entry_state():指定程序在初始运行时的状态，默认从入口点执行
#add_options 获取一个独立的选项来添加到某个 state 中
#SYMBOLIC_WRITE_ADDRESSES 允许通过具体化策略处理符号地址的写操作
state=p.factory.entry_state(add_options={angr.options.SYMBOLIC_WRITE_ADDRESSES})

#创建一个符号变量，这个符号变量以 8 位 bitvector 形式存在，名称为 u
u = claripy.BVS(“u”,8)
#把符号变量保存到指定地址中，这个地址是二进制文件中.bss 段 u 的地址
```

```
    state.memory.store(0x804a021,u)

    #创建一个 Simulation Manager 对象，对象和状态有关
    sm = p.factory.simulation_manager(state)

    #使用 explore 函数进行状态搜素，检查输出字符是 win 还是 lose
    #state.posix.dumps(1)获得所有标准输出
    #state.posix.dumps(0)获得所有标准输入
    def correct(state):
        try:
           return b’win’ in state.posix.dumps(1)
        except:
           return False
    def wrong(state):
        try:
            return b’lose’ in state.posix.dumps(1)
        except:
    #进行符号执行得到想要的状态，得到满足 correct 条件且不满足 wrong 条件的 state
    sm.explore(find = correct,avoid = wrong)

    #也可直接通过地址进行定位
    #sm.explore(find = 0x80484e3,avoid = 0x80484f5)

    #获得到 state 之后，通过 solver 求解器，求解 u 的值
    #eval_upto(e,n,cast_to= None,**kwargs)求解一个表达式指定个可能的求解个数 e-表达式 n-所需解决方案的数量
    #eval(e,**kwargs)评估一个表达式以获得任何可能的解决方案 e-表达式
    #eval_one(e,**kwargs)求解表达式以获得唯一可能的解决方案 e-表达式
    return sm.found[0].solver.eval_upto(u,256)

    if __name__ == ‘__main__’
    #repr()函数将 object 对象转化为 string 类型
    print(repr(main())

    #!;usr/bin/env python
    #coding=utr-8
    import angr
    import claripy

    def hook_demo(state):
        state.regs.eax = 0

    p.angr.Project(“./issue”,load_options = {“auto_load_libs”:False})
    #hook 函数：addr 为待 hook 的地址
```

#hook 为 hook 的处理函数，在执行到 addr 时，执行这个函数，把当前 state 对象作为参数传递过去

#length 为待 hook 指令的长度，在执行完 hook 函数以后，angr 需要根据 length 来跳过这条指令，执行下一条指令

#hook 0x08048485 处的指令（xor eax，eax），等价于 eax 设置为 0

#hook 不会改变函数逻辑，只是更换实现方式，提升符号执行速度

p.hook(addr=0x08048485,hook = hook_demo,length = 2)

state=p.factory.blank_state(addr=0x0804846B,add_options={“SYMBOLIC_WRITE_ADDRESSES”})

u = claripy.BVS(“u”,8)

u 的地址，ida 分析可得

（此处根据实际操作过程，留下具体操作步骤、附加一些自己的理解，即可）

心得体会

angr 是用 Python 编写的二进制分析框架，可用于探索二进制文件并查找漏洞。下面是一些 angr 的常见用法：

1、加载二进制文件

```
import angr

binary = './example'
proj = angr.Project(binary, auto_load_libs=False)
```

2、配置 angr 进行符号执行

```
state = proj.factory.entry_state()
simgr = proj.factory.simulation_manager(state)
```

3、执行符号执行

```
simgr.run()
```

4、检查状态

```
if simgr.deadended:
    print(simgr.deadended)
```

5、获取执行结果

```
print(simgr.deadended[0].posix.dumps(0))
```

这是一个简单的示例，它使用 angr 来查找二进制文件中的“正确”输入。在实践中，angr 可以用于执行各种二进制分析任务，如漏洞研究、恶意软件分析、加密算法分析等。