《漏洞利用及渗透测试基础》实验报告

姓名：申宗尚 学号：2213924 班级：信息安全

**实验名称：**

Angr应用示例

**实验要求：**

根据课本8.4.3章节，复现sym-write示例的两种angr求解方法，并就如何使用angr以及怎么解决一些实际问题做一些探讨。

**实验过程：**

1. 首先，先进行 py3.12 的安装和 path 配置和 angr 在环境下的安装
2. 在下面的代码中，为了找出达到 win 的 u 值，将 u 符号化，以具体的数值作为输入执行程序代码，在程序实际执行路径的基础上，用符号执行技术对路径进行分析，提取路径的约束表达式，再约束求解。

文本, 信件

描述已自动生成

3.然后，首先分析solve.py的代码：

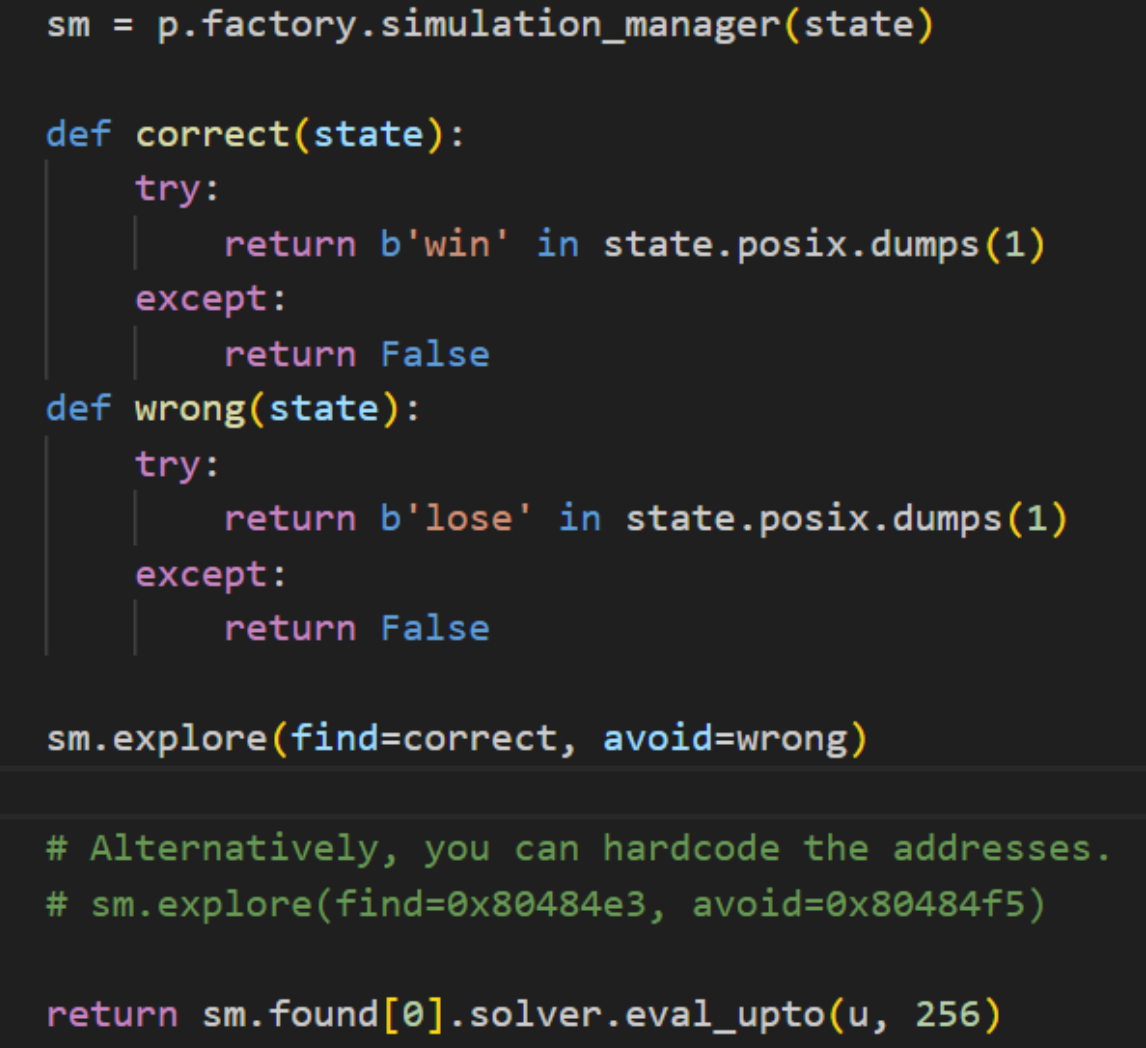


首先，这段代码创建了angr项目：使用angr.Project创建一个名为p的项目，并指定要分析的二进制文件./issue。设置load\_options为{"auto\_load\_libs": False}，表示不自动加载库文件。

然后，使用p.factory.entry\_state创建一个初始状态，添加选项angr.options.SYMBOLIC\_WRITE\_ADDRESSES，这会让所有符号写入的地址具体化。

最后，创造符号变量，使用claripy.BVS("u", 8)创建一个名为u、大小为8位的符号比特向量。并将符号变量u存储到内存地址0x804a021。

进行程序插桩完成后便可查看输出文件得知所用到的指令数。

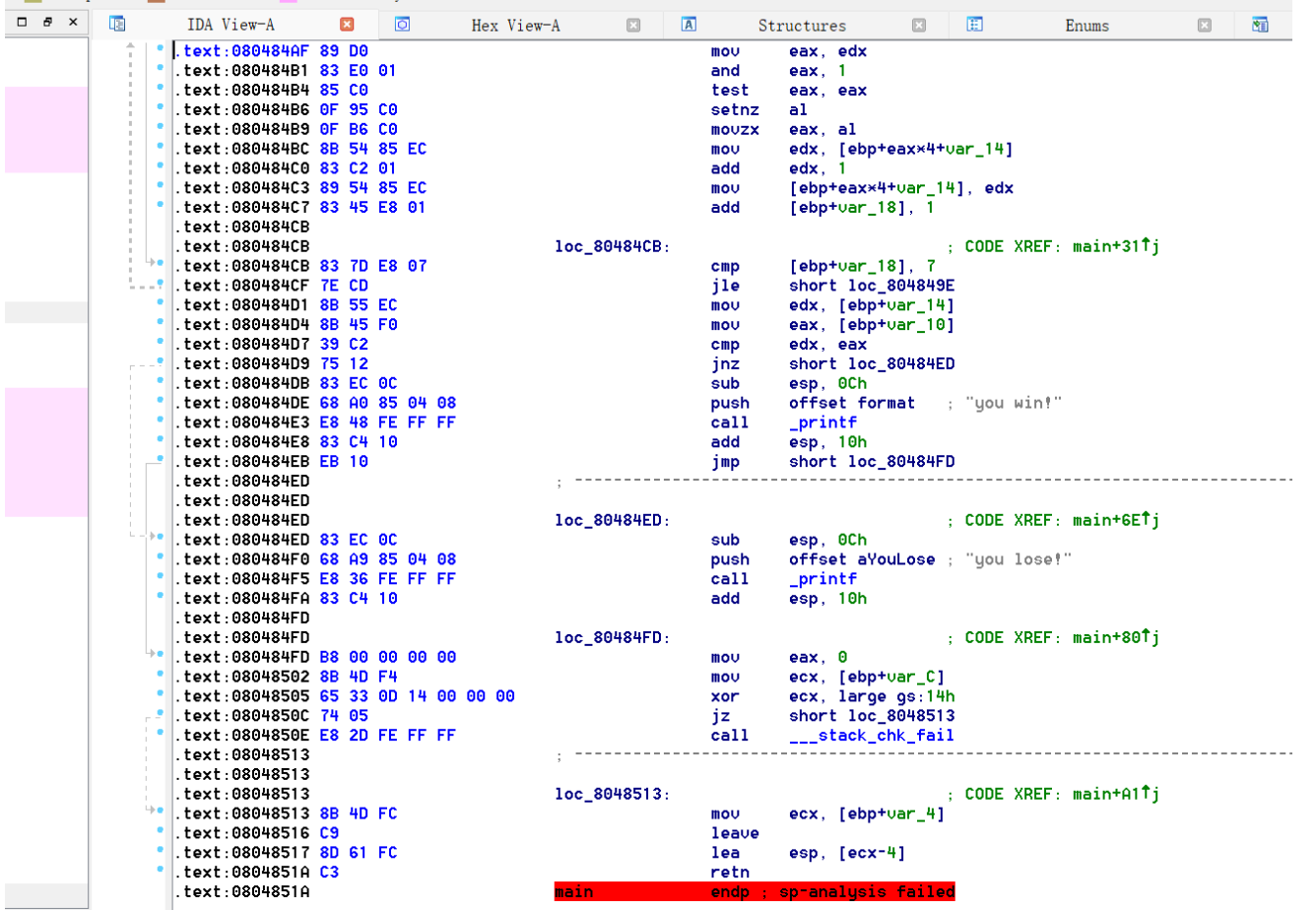


然后，创建一个 Simulation Manager 对象，管理运行得到的状态对象，定义函数：

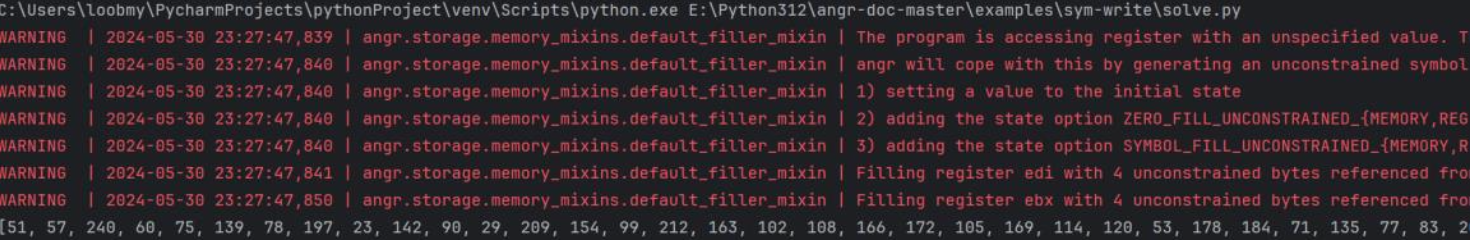
state.posix.dumps(1)获得所有标准输出，动态符号执行&得到想要的状态→使用 explor，函数进行状态搜寻。也可以写成：sm.explore(find=0x80484e3, avoid=0x80484f5)，约束求解，获得到 state 之后，通过 solver 求解器，求解 u 的值，256 表示求解出 256 种结果。

5.将其导入idapro进行反汇编，可以对应上这里的 sm.explore（） 。

得到以下结果：



6. 第一种方法直接运行 solve结果：



51, 57, 240, 60, 75, 139, 78, 197, 23, 142, 90, 29, 209, 154, 99, 212, 163, 102,

108, 166, 172, 105, 169, 114, 120, 53, 178, 184, 71, 135, 77, 83, 202, 89, 147,

86, 153, 92, 150, 156, 106, 101, 141, 165, 43, 113, 232, 226, 177, 116, 46, 180,

45, 58, 198, 15, 201, 195, 85, 204, 30, 149, 210, 27, 216, 39, 225, 170, 228,

54

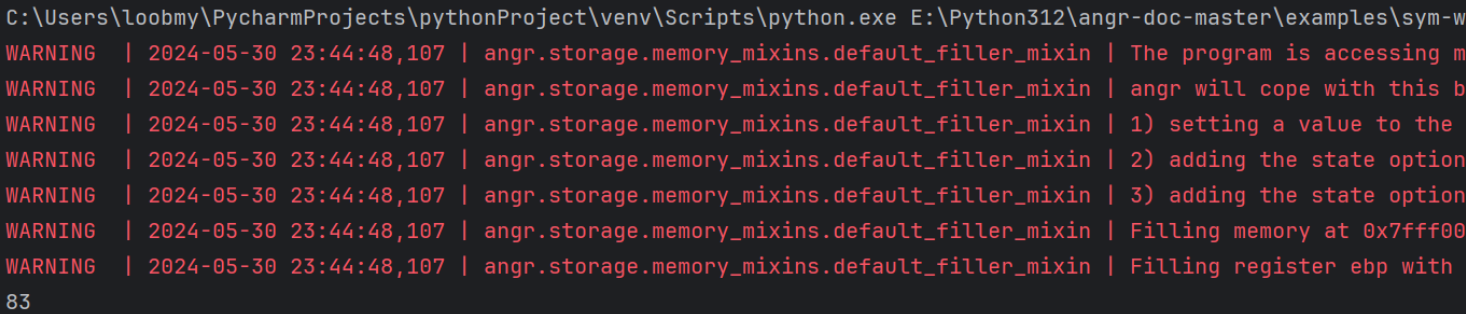
将该值验证，正确

7. 第二种hook：进行代码增加如下：



0x08048485处指令为 xor eax, eax, hook一个函数，指令长度为 2，实际并没有带来任何变化，仅为 Hook演示， SimState对象的调用的函数改为 blank\_state，并将 addr参数设置为被检测文件的程序入口地址，这等价于使用 entry\_state 创建 SimState 对象，在sm.explore中只给定一个条件，因为是分支语句，已经足以确定唯一路径。

8. 结果如下：



验证正确，实验成功。

**心得体会：**

在本次实验中，我对angr的使用有了更深刻的认识。

其次，在代码分析和实现过程中，我深入理解了angr的工作原理和符号执行的基本概念。通过具体化符号写入地址、创建符号变量、存储符号变量到内存地址等操作，我对符号执行技术有了更全面的认识，掌握了如何通过符号执行技术来进行路径分析和约束求解。

本次实验尝试了两种求解方法：直接运行solve结果和使用hook函数。两个过程各有其优缺点。直接运行solve结果能够快速求解并获得多种可能结果，而使用hook函数则提供了一种灵活的方法，可以在特定指令处进行干预和调试。