

# 《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。

```
#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;
}
```

3. 然后，首先分析 solve.py 的代码：

```
def main():
    p = angr.Project('./issue', load_options={"auto_load_libs": False})

    # By default, all symbolic write indices are concretized.
    state = p.factory.entry_state(add_options={angr.options.SYMBOLIC_WRITE_ADDRESSES})

    u = claripy.BVS("u", 8)
    state.memory.store(0x804a021, u)
```

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。

```
sm = p.factory.simulation_manager(state)

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:
        return False

sm.explore(find=correct, avoid=wrong)

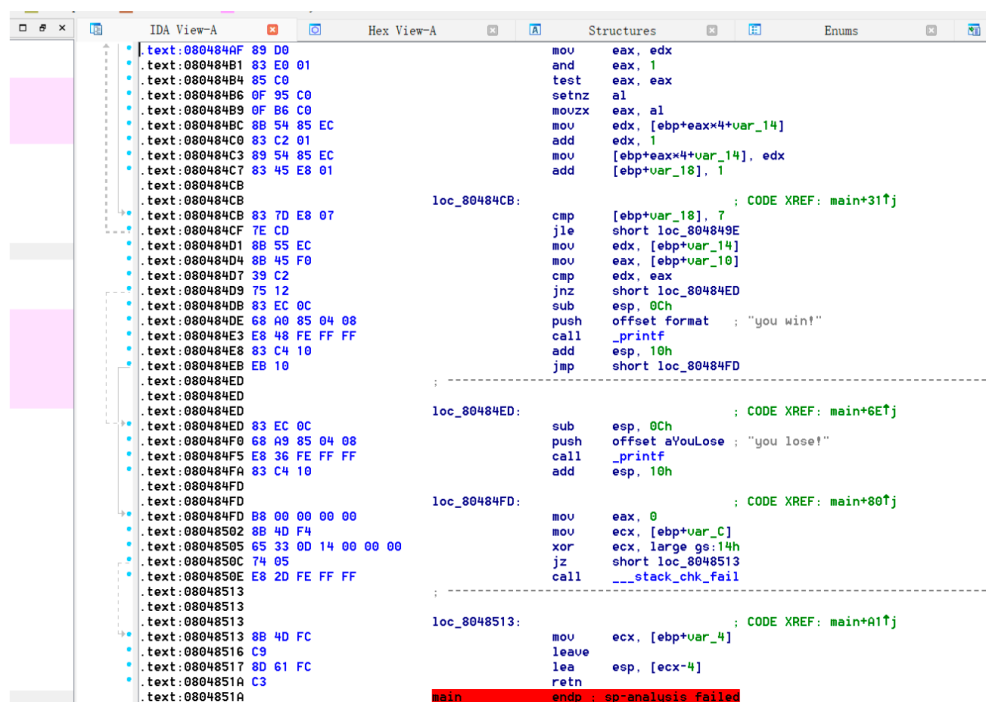
# Alternatively, you can hardcode the addresses.
# sm.explore(find=0x80484e3, avoid=0x80484f5)

return sm.found[0].solver.eval_upto(u, 256)
```

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 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：



```
IDA View-A Hex View-A Structures Enums
[.text:080484AF 89 D0      mov     eax, edx
[.text:080484B1 83 E0 01    and     eax, 1
[.text:080484B4 85 C0      test    eax, eax
[.text:080484B6 0F 95 C0    setnz   al
[.text:080484B8 0F B6 C0    movzx   eax, al
[.text:080484BC 8B 54 85 EC mov     edx, [ebp+eax*4+var_14]
[.text:080484C0 83 C2 01    add     edx, 1
[.text:080484C3 8B 54 85 EC mov     [ebp+eax*4+var_14], edx
[.text:080484C7 83 45 E8 01 add     [ebp+var_18], 1
[.text:080484CB
loc_80484CB:
[.text:080484CB 83 7D E8 07 cmp     [ebp+var_18], 7
[.text:080484CF 7E CD      jle     short loc_804849E
[.text:080484D1 8B 55 EC    mov     edx, [ebp+var_14]
[.text:080484D4 8B 45 F0    mov     eax, [ebp+var_10]
[.text:080484D7 39 C2      cmp     edx, eax
[.text:080484D9 75 12      jnz     short loc_80484ED
[.text:080484DB 83 EC 0C    sub     esp, 0Ch
[.text:080484DE 68 A0 85 04 08 push    offset aYouWin ; "you win!"
[.text:080484E3 E8 48 FE FF FF call    _printf
[.text:080484E8 83 C4 10    add     esp, 10h
[.text:080484EB EB 10      jmp     short loc_80484FD
[.text:080484ED
loc_80484ED:
[.text:080484ED 83 EC 0C    sub     esp, 0Ch
[.text:080484F0 68 A9 85 04 08 push    offset aYouLose ; "you lose!"
[.text:080484F5 E8 36 FE FF FF call    _printf
[.text:080484FA 83 C4 10    add     esp, 10h
[.text:080484FD
loc_80484FD:
[.text:080484FD B8 00 00 00 00 mov     eax, 0
[.text:08048502 8B 4D F4    mov     ecx, [ebp+var_C]
[.text:08048505 65 33 00 14 00 00 xor     ecx, large gs:14h
[.text:0804850C 74 05      jz      short loc_8048513
[.text:0804850E E8 2D FE FF FF call    __stack_chk_fail
[.text:08048513
loc_8048513:
[.text:08048513 mov     ecx, [ebp+var_4]
[.text:08048516 C9         leave   esp, [ecx-4]
[.text:08048517 8D 61 FC    lea     esp, [ecx-4]
[.text:0804851A C3         retn
main      endp ; sp-analysis failed
```

## 6. 第一种方法直接运行 solve 结果:

```
..Users\loobmy\PycharmProjects\pythonProject\venv\Scripts\python.exe E:\Python312\angr-doc-master\examples\sym-write\solve.py
WARNING | 2024-05-30 23:27:47,839 | angr.storage.memory_mixins.default_filler_mixin | The program is accessing register with an unspecified value. T
WARNING | 2024-05-30 23:27:47,840 | angr.storage.memory_mixins.default_filler_mixin | angr will cope with this by generating an unconstrained symbol
WARNING | 2024-05-30 23:27:47,840 | angr.storage.memory_mixins.default_filler_mixin | 1) setting a value to the initial state
WARNING | 2024-05-30 23:27:47,840 | angr.storage.memory_mixins.default_filler_mixin | 2) adding the state option ZERO_FILL_UNCONSTRAINED_(MEMORY,REG
WARNING | 2024-05-30 23:27:47,840 | angr.storage.memory_mixins.default_filler_mixin | 3) adding the state option SYMBOL_FILL_UNCONSTRAINED_(MEMORY,R
WARNING | 2024-05-30 23:27:47,841 | angr.storage.memory_mixins.default_filler_mixin | Filling register edi with 4 unconstrained bytes referenced fro
WARNING | 2024-05-30 23:27:47,850 | angr.storage.memory_mixins.default_filler_mixin | Filling register ebx with 4 unconstrained bytes referenced fro
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, 2
```

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:

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

def main():
    p = angr.Project(thing='./issue', load_options={"auto_load_libs": False})
    p.hook(addr=0x08048485, hook=hook_demo, length=2)
    # By default, all symbolic write indices are concretized.
    state = p.factory.blank_state(addr=0x0804846B, add_options={"SYMBOLIC_WRITE_ADDRESSES"})

    u = claripy.BVS(name="u", size=8)
    state.memory.store(addr=0x804a021, u)

    sm = p.factory.simulation_manager(state)
    sm.explore(find=0x080484DB)
    st = sm.found[0]
    print(repr(st.solver.eval(u)))
```

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:

```
C:\Users\loobmy\PycharmProjects\pythonProject\venv\Scripts\python.exe E:\Python312\angr-doc-master\examples\sym-w
WARNING | 2024-05-30 23:44:48,107 | angr.storage.memory_mixins.default_filler_mixin | The program is accessing m
WARNING | 2024-05-30 23:44:48,107 | angr.storage.memory_mixins.default_filler_mixin | angr will cope with this b
WARNING | 2024-05-30 23:44:48,107 | angr.storage.memory_mixins.default_filler_mixin | 1) setting a value to the
WARNING | 2024-05-30 23:44:48,107 | angr.storage.memory_mixins.default_filler_mixin | 2) adding the state option
WARNING | 2024-05-30 23:44:48,107 | angr.storage.memory_mixins.default_filler_mixin | 3) adding the state option
WARNING | 2024-05-30 23:44:48,107 | angr.storage.memory_mixins.default_filler_mixin | Filling memory at 0x7fff00
WARNING | 2024-05-30 23:44:48,107 | angr.storage.memory_mixins.default_filler_mixin | Filling register ebp with
83
```

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。