



南開大學
Nankai University

计算机学院
软件安全实验报告

实验九：Angr 应用示例实验

姓名：林盛森

学号：2312631

专业：计算机科学与技术

2025 年 5 月 18 日

目录

1 实验名称	2
2 实验要求	2
3 实验过程	2
3.1 安装 python3 和 angr	2
3.2 复现 sym-write 示例的两种 angr 求解方法	4
3.2.1 第一种方法	4
3.2.2 第二种方法	7
4 心得体会	8
4.1 如何使用 angr	8
4.2 如何通过 angr 解决一些实际问题	9

1 实验名称

Angr 应用示例实验

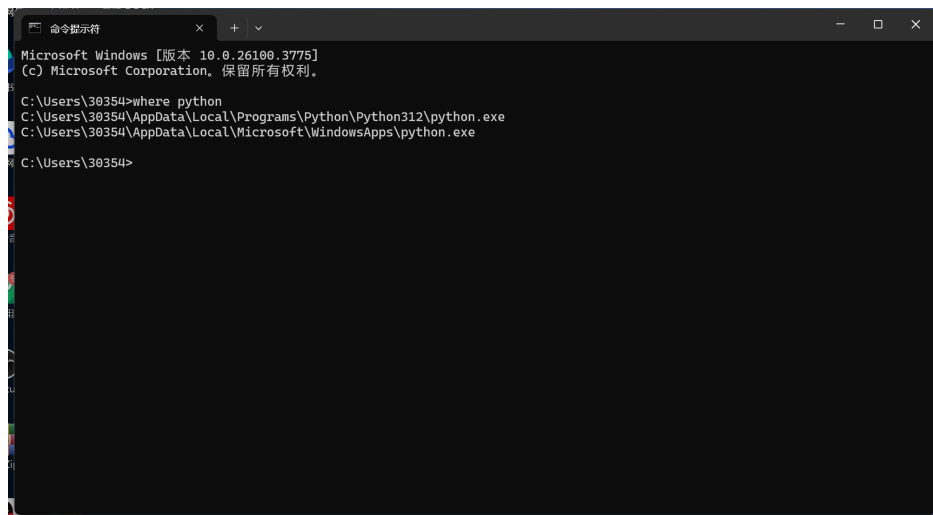
2 实验要求

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。

3 实验过程

3.1 安装 python3 和 angr

首先要进行 python3 的安装，在终端中输入命令 `where python` 可以看到我们电脑中已经安装了 python3.12。



```
命令提示符
Microsoft Windows [版本 10.0.26100.3775]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\30354>where python
C:\Users\30354\AppData\Local\Programs\Python\Python312\python.exe
C:\Users\30354\AppData\Local\Microsoft\WindowsApps\python.exe

C:\Users\30354>
```

图 3.1: Enter Caption

然后利用命令行 `pip install angr` 安装 angr。

```
命令提示符
Downloading unique_log_filter-0.1.0-py3-none-any.whl (1.5 kB)
Downloading gitdb-4.0.12-py3-none-any.whl (62 kB)
Downloading pyelftools-0.32-py3-none-any.whl (188 kB)
Downloading bitstring-4.3.1-py3-none-any.whl (71 kB)
Downloading cart-1.2.3-py2.py3-none-any.whl (10 kB)
Downloading pefile-2024.8.26-py3-none-any.whl (74 kB)
Downloading pydot-4.0.0-py3-none-any.whl (37 kB)
Downloading bitarray-3.4.0-cp312-cp312-win_amd64.whl (139 kB)
Downloading smmap-5.0.2-py3-none-any.whl (24 kB)
Downloading pycryptodome-3.22.0-cp37-abi3-win_amd64.whl (1.8 MB)
1.8/1.8 MB 14.1 MB/s eta 0:00:00
Building wheels for collected packages: multiplexer
Building wheel for multiplexer (setup.py) ... done
Created wheel for multiplexer: filename=multiplexer-0.9-py3-none-any.whl size=3827 sha256=47aa9fbade47429144e4883d7fc5320de49bfb02b8eefb3fb4b7748bc23d795
Stored in directory: c:\users\30354\appdata\local\pip\cache\wheels\76\f6\59\d8cf8137916b6e9632c3d4a211d4382523eb18c654d3317b9b
Successfully built multiplexer
Installing collected packages: z3-solver, sortedcontainers, pyelftools, multiplexer, bitarray, unique-log-filter, smmap, pydot, pydumblle, pycryptodome, pefile, cxheaderparser, capstone, cachetools, bitstring, archinfo, ailment, pyvex, pyformlang, gitdb, claripy, cart, GitPython, cle, angr
Successfully installed GitPython-3.1.44 ailment-9.2.154 angr-9.2.154 archinfo-9.2.154 bitarray-3.4.0 bitstring-4.3.1 cac
hetools-5.5.2 capstone-5.0.3 cart-1.2.3 claripy-9.2.154 cle-9.2.154 cxheaderparser-1.5.0 gitdb-4.0.12 multiplexer-0.9 p
efile-2024.8.26 pycryptodome-3.22.0 pydumblle-0.0.1 pydot-4.0.0 pyelftools-0.32 pyformlang-1.0.11 pyvex-9.2.154 smmap-5
.0.2 sortedcontainers-2.4.0 unique-log-filter-0.1.0 z3-solver-4.13.0.0

[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\30354>
```

图 3.2: Enter Caption

然后我们在 python 中输入 `import angr` 语句并运行来看 angr 是否安装成功，没有报错说明安装成功。

```
命令提示符 - python
Downloading cart-1.2.3-py2.py3-none-any.whl (10 kB)
Downloading pefile-2024.8.26-py3-none-any.whl (74 kB)
Downloading pydot-4.0.0-py3-none-any.whl (37 kB)
Downloading bitarray-3.4.0-cp312-cp312-win_amd64.whl (139 kB)
Downloading smmap-5.0.2-py3-none-any.whl (24 kB)
Downloading pycryptodome-3.22.0-cp37-abi3-win_amd64.whl (1.8 MB)
1.8/1.8 MB 14.1 MB/s eta 0:00:00
Building wheels for collected packages: multiplexer
Building wheel for multiplexer (setup.py) ... done
Created wheel for multiplexer: filename=multiplexer-0.9-py3-none-any.whl size=3827 sha256=47aa9fbade47429144e4883d7fc5320de49bfb02b8eefb3fb4b7748bc23d795
Stored in directory: c:\users\30354\appdata\local\pip\cache\wheels\76\f6\59\d8cf8137916b6e9632c3d4a211d4382523eb18c654d3317b9b
Successfully built multiplexer
Installing collected packages: z3-solver, sortedcontainers, pyelftools, multiplexer, bitarray, unique-log-filter, smmap, pydot, pydumblle, pycryptodome, pefile, cxheaderparser, capstone, cachetools, bitstring, archinfo, ailment, pyvex, pyformlang, gitdb, claripy, cart, GitPython, cle, angr
Successfully installed GitPython-3.1.44 ailment-9.2.154 angr-9.2.154 archinfo-9.2.154 bitarray-3.4.0 bitstring-4.3.1 cac
hetools-5.5.2 capstone-5.0.3 cart-1.2.3 claripy-9.2.154 cle-9.2.154 cxheaderparser-1.5.0 gitdb-4.0.12 multiplexer-0.9 p
efile-2024.8.26 pycryptodome-3.22.0 pydumblle-0.0.1 pydot-4.0.0 pyelftools-0.32 pyformlang-1.0.11 pyvex-9.2.154 smmap-5
.0.2 sortedcontainers-2.4.0 unique-log-filter-0.1.0 z3-solver-4.13.0.0

[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\30354>python
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import angr
>>>
```

图 3.3: Enter Caption

然后我们去 github 上 angr 官方手册把实验样例下载下来，并进行解压如下图所示。

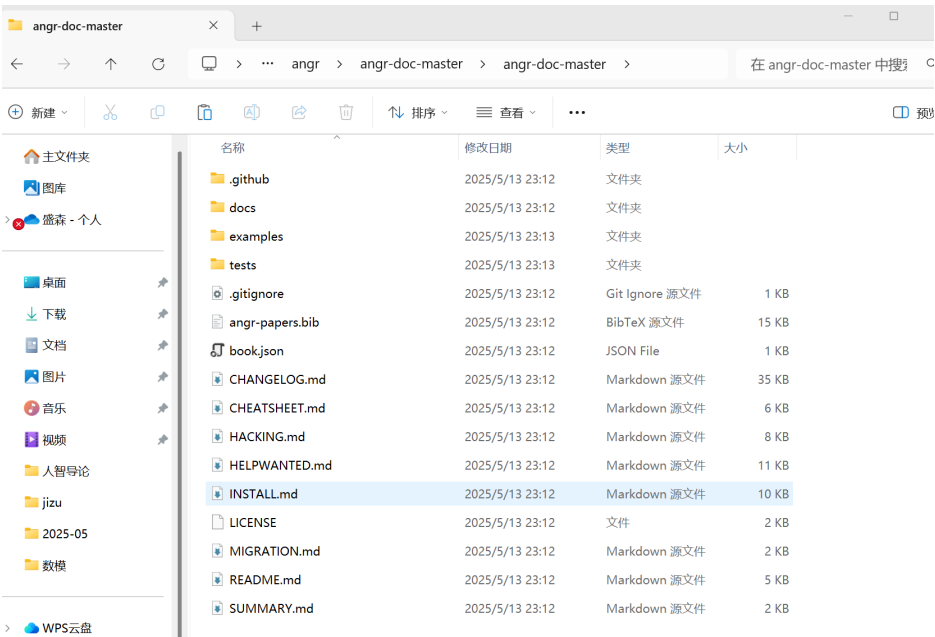


图 3.4: Enter Caption

其中有各类 examples，展示了 Angr 的用法。

3.2 复现 sym-write 示例的两种 angr 求解方法

3.2.1 第一种方法

issue.c:

```
1 #include <stdio.h>
2
3 char u=0;
4 int main(void)
5 {
6     int i, bits[2]={0,0};
7     for (i=0; i<8; i++) {
8         bits[(u&(1<<i))!=0]++;
9     }
10    if (bits[0]==bits[1]) {
11        printf("you win!");
12    }
13    else {
14        printf("you lose!");
15    }
16    return 0;
17 }
```

solve1.c

```
1 import angr
2 import claripy
```

```
3
4 def main():
5     p = angr.Project('./issue', load_options={"auto_load_libs": False})
6
7     # By default, all symbolic write indices are concretized.
8     state = p.factory.entry_state(add_options={angr.options.SYMBOLIC_WRITE_ADDRESSES})
9
10    u = claripy.BVS("u", 8)
11    state.memory.store(0x804a021, u)
12
13    sm = p.factory.simulation_manager(state)
14
15    def correct(state):
16        try:
17            return b'win' in state.posix.dumps(1)
18        except:
19            return False
20    def wrong(state):
21        try:
22            return b'lose' in state.posix.dumps(1)
23        except:
24            return False
25
26    sm.explore(find=correct, avoid=wrong)
27
28    # Alternatively, you can hardcode the addresses.
29    # sm.explore(find=0x80484e3, avoid=0x80484f5)
30
31    return sm.found[0].solver.eval_upto(u, 256)
32
33
34 def test():
35     good = set()
36     for u in range(256):
37         bits = [0, 0]
38         for i in range(8):
39             bits[u & (1 << i) != 0] += 1
40         if bits[0] == bits[1]:
41             good.add(u)
42
43     res = main()
44     assert set(res) == good
45
46 if __name__ == '__main__':
47     print(repr(main()))
```

第一种方法运行结果

我们将程序在终端中进行编译运行。使用命令行 `python solve.py`。结果如下所示：

```

C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.26100.3775]
(c) Microsoft Corporation. 保留所有权利。

D:\工具包\angr\angr-doc-master\angr-doc-master\examples\sym-write>python solve.py
WARNING | 2025-05-13 23:15:16,294 | angr.storage.memory_mixins.default_filler_mixin | The program is accessing register
with an unspecified value. This could indicate unwanted behavior.
WARNING | 2025-05-13 23:15:16,295 | angr.storage.memory_mixins.default_filler_mixin | angr will cope with this by gener
ating an unconstrained symbolic variable and continuing. You can resolve this by:
WARNING | 2025-05-13 23:15:16,295 | angr.storage.memory_mixins.default_filler_mixin | 1) setting a value to the initial
state
WARNING | 2025-05-13 23:15:16,295 | angr.storage.memory_mixins.default_filler_mixin | 2) adding the state option ZERO_F
ILL_UNCONSTRAINED_(MEMORY,REGISTERS), to make unknown regions hold null
WARNING | 2025-05-13 23:15:16,295 | angr.storage.memory_mixins.default_filler_mixin | 3) adding the state option SYMBOL
_FILL_UNCONSTRAINED_(MEMORY,REGISTERS), to suppress these messages.
WARNING | 2025-05-13 23:15:16,295 | angr.storage.memory_mixins.default_filler_mixin | Filling register edi with 4 uncon
strained bytes referenced from 0x8048521 (__libc_csu_init+0x1 in issue (0x8048521))
WARNING | 2025-05-13 23:15:16,296 | angr.storage.memory_mixins.default_filler_mixin | Filling register ebx with 4 uncon
strained bytes referenced from 0x8048523 (__libc_csu_init+0x3 in issue (0x8048523))
[51, 57, 60, 240, 75, 139, 78, 197, 23, 142, 90, 29, 209, 154, 212, 99, 163, 182, 188, 166, 172, 185, 169, 114, 53, 120,
225, 184, 178, 71, 135, 77, 83, 89, 141, 147, 153, 92, 86, 150, 156, 202, 101, 106, 165, 43, 226, 113, 46, 177, 116, 23
2, 180, 58, 198, 15, 201, 195, 85, 204, 30, 149, 210, 27, 216, 39, 45, 170, 228, 54]

D:\工具包\angr\angr-doc-master\angr-doc-master\examples\sym-write>

```

图 3.5: Enter Caption

对第一种方法的分析

这段程序利用了符号执行工具 angr 和符号表达式库 claripy，目的是通过符号执行找到某个输入字节 `u`，使得程序输出中包含“win”字样，且避免出现包含“lose”的输出。然后对比用一个纯 Python 的方法计算符合条件的字节，验证符号执行的结果。

在上述 Angr 示例中，几个关键步骤如下：

(1) 新建一个 Angr 工程，并且载入二进制文件。auto_load_libs 设置为 false，将不会自动载入依赖的库，默认情况下设置为 false。如果设置为 true，转入库函数执行，有可能给符号执行带来不必要的麻烦。

(2) 初始化一个模拟程序状态的 SimState 对象 state（使用函数 entry_state()），该对象包含了程序的内存、寄存器、文件系统数据、符号信息等等模拟运行时动态变化的数据。此外，也可以使用函数 blank_state() 初始化模拟程序状态的对象 state，在该函数里可通过给定参数 addr 的值指定程序起始运行地址。

(3) 将要求解的变量符号化，注意这里符号化后的变量存在二进制文件的存储区。

(4) 创建模拟管理器（Simulation Managers）进行程序执行管理。初始化的 state 可以经过模拟执行得到一系列的 states，模拟管理器 sm 的作用就是对这些 states 进行管理。

(5) 进行符号执行得到想要的状态，得到想要的状态。上述程序所表达的状态就是，符号执行后，源程序里打印出的字符串里包含 win 字符串，而没有包含 lose 字符串。在这里，状态被定义为两个函数，通过符号执行得到的输出 state.posix.dumps(1) 中是否包含 win 或者 lose 的字符串来完成定义。

注意：这里也可以用 find=0x80484e3, avoid=0x80484f5 来代替，即通过符号执行是否到达特定代码区的地址。使用 IDA 反汇编可知 0x80484e3 是 printf(“you win!”) 对应的汇编语句；0x80484f5 则是 printf(“you lose!”) 对应的汇编语句。

(6) 获得到 state 之后，通过 solver 求解器，求解 `u` 的值。

这里多个函数可以使用，eval_upto(e, n, cast_to=None, **kwargs) 求解一个表达式多个可能的求解方案，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；eval_one(e, **kwargs) 求解表达式以获得唯一可能的解决方案。

(7) 在 test 函数中，用 Python 纯穷举的方式验证 main() 的符号执行结果是否正确。对 0 255 每个字节 `u`：会去统计该字节中 bit 位为 0 和 1 的数量，如果 0 和 1 数量相等（即 4 个 0 和 4 个 1），则加入集合 good。接着调用 main() 获得符号执行返回的结果。然后断言两个集合是否完全相同，保

证符号执行正确地找到了所有符合“0 和 1 位数相等”的字节。

3.2.2 第二种方法

solve2.py

```

1 import angr
2 import claripy
3
4 def hook_demo(state):
5     state.regs.eax = 0
6
7 p = angr.Project("./issue", load_options={"auto_load_libs": False})
8 # hook函数: addr为待hook的地址
9 #
10 # hook为hook的处理函数, 在执行到addr时, 会执行这个函数, 同时把当前的state对象作为参数传递过去
11 # length 为待hook指令的长度, 在执行完 hook 函数以后, angr 需要根据 length
12 # 来跳过这条指令, 执行下一条指令
13 # hook 0x08048485处的指令 (xor eax,eax), 等价于将eax设置为0
14 # hook并不会改变函数逻辑, 只是更换实现方式, 提升符号执行速度
15 p.hook(addr=0x08048485, hook=hook_demo, length=2)
16 state = p.factory.blank_state(addr=0x0804846B,
17                               add_options={"SYMBOLIC_WRITE_ADDRESSES"})
18 u = claripy.BVS("u", 8)
19 state.memory.store(0x0804A021, u)
20 sm = p.factory.simulation_manager(state)
21 sm.explore(find=0x080484DB)
22 st = sm.found[0]
23
24 print(repr(st.solver.eval(u)))

```

第二种方法运行结果

我们同样将程序在终端中进行编译运行。使用命令行 `python solve2.py`。结果如下所示：

```

Windows PowerShell
版权所有 (C) Microsoft Corporation. 保留所有权利。

安装最新的 PowerShell, 了解新功能和改进! https://aka.ms/PSWindows

PS D:\工具包\angr\angr-doc-master\angr-doc-master\examples\sym-write> python solve2.py
WARNING | 2025-05-13 23:17:25,313 | angr.storage.memory_mixins.default_filler_mixin | The program is accessing memory w
ith an unspecified value. This could indicate unwanted behavior.
WARNING | 2025-05-13 23:17:25,313 | angr.storage.memory_mixins.default_filler_mixin | angr will cope with this by gener
ating an unconstrained symbolic variable and continuing. You can resolve this by:
WARNING | 2025-05-13 23:17:25,313 | angr.storage.memory_mixins.default_filler_mixin | 1) setting a value to the initial
state
WARNING | 2025-05-13 23:17:25,313 | angr.storage.memory_mixins.default_filler_mixin | 2) adding the state option ZERO_F
ILL_UNCONSTRAINED_(MEMORY,REGISTERS), to make unknown regions hold null
WARNING | 2025-05-13 23:17:25,313 | angr.storage.memory_mixins.default_filler_mixin | 3) adding the state option SYMBOL
_FILL_UNCONSTRAINED_(MEMORY,REGISTERS), to suppress these messages.
WARNING | 2025-05-13 23:17:25,313 | angr.storage.memory_mixins.default_filler_mixin | Filling memory at 0x7fff0000 with
4 unconstrained bytes referenced from 0x8048472 (main+0x7 in issue (0x8048472))
WARNING | 2025-05-13 23:17:25,314 | angr.storage.memory_mixins.default_filler_mixin | Filling register ebp with 4 uncon
strained bytes referenced from 0x8048475 (main+0xa in issue (0x8048475))
226
PS D:\工具包\angr\angr-doc-master\angr-doc-master\examples\sym-write>

```

图 3.6: Enter Caption

对第二种方法的分析

上述代码与前面的解法有三处区别：

(1) 采用了 hook 函数，将 0x08048485 处的长度为 2 的指令通过自定义的 hook_demo 进行替代，功能是一致的，原始 xor eax, eax 和 state.regs.eax = 0 是相同的作用，这里只是演示，可以将一些复杂的系统函数调用，比如 printf 等，可以进行 hook，提升符号执行的性能。

(2) 进行符号执行得到想要的状态，有变化，变更为 find=0x080484DB。因为源程序 win 和 lose 是互斥的，所以，只需要给定一个 find 条件即可。

(3) 最后，eval(u) 替代了原来的 eval_upto，将打印一个结果出来，而不是同一个结果。

4 心得体会

4.1 如何使用 angr

(1) 加载二进制

```
1 import angr
2 proj = angr.Project('binary_name', auto_load_libs=False)
```

auto_load_libs=False 不加载系统库，加快分析速度，避免干扰。

(2) 创建初始执行状态

```
1 state = proj.factory.entry_state()
```

从程序入口点开始执行状态。

(3) 设置符号输入

比如给程序输入一个符号变量：

```
1 import claripy
2
3 sym_arg = claripy.BVS('input', 8*length) # 比如长度为 length 字节
4 state.memory.store(address, sym_arg)
```

或者给命令行参数、stdin 赋值：

```
1 state.posix.stdin.write(sym_arg)
2 state.posix.stdin.seek(0)
```

(4) 创建 SimulationManager 进行路径探索

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

(5) 使用 explore() 寻找满足条件的路径

```
1 def is_successful(state):
2     return b'success' in state.posix.dumps(1)
3
4 def should_abort(state):
5     return b'fail' in state.posix.dumps(1)
6
```

```
7 | simgr.explore(find=is_successful, avoid=should_abort)
```

(6) 获取满足条件的输入解

```
1 | if simgr.found:
2 |     found_state = simgr.found[0]
3 |     solution = found_state.solver.eval(sym_arg, cast_to=bytes)
4 |     print(solution)
```

4.2 如何通过 angr 解决一些实际问题

1. 漏洞挖掘：

Angr 可以用于在二进制程序中发现漏洞，如缓冲区溢出、整数溢出等。通过符号执行和路径搜索，Angr 能够探索程序的不同执行路径，并识别可能存在漏洞的代码路径。

2. 逆向工程：

Angr 可用于逆向工程任务，如反编译、恶意代码分析等。通过分析程序的控制流和数据流，Angr 可以帮助理解程序的逻辑结构和算法。

3. 破解验证：

Angr 能够通过符号执行求解程序输入，从而找到绕过授权验证、破解密码或序列号的方法。

4. 自动化测试用例生成：

Angr 可以自动生成覆盖程序不同路径的测试用例，提升软件测试的覆盖率和质量。

5. 恶意软件分析：

Angr 支持动态符号执行和污点分析，帮助分析恶意软件的行为，绕过反调试技术，定位恶意代码逻辑。

6. 协议与文件格式分析：

通过符号执行探索协议状态机和文件结构，Angr 能自动生成合法的协议消息或文件输入，辅助协议逆向和文件解析。