《Angr 求解方法》实验报告

姓名：汤清云 学号：2013536 班级： 1075

实验名称：

Angr 求解方法。

实验要求：

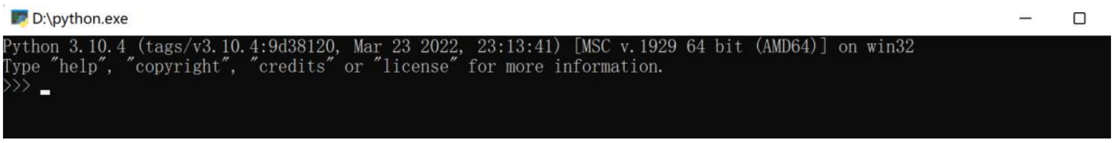
学习使用 Angr 完成两种方式的对问题的求解。

就如何使用 Angr 和如何使用 Angr 进行实际问题求解做出思考。

实验过程：

1. 安装 Angr：

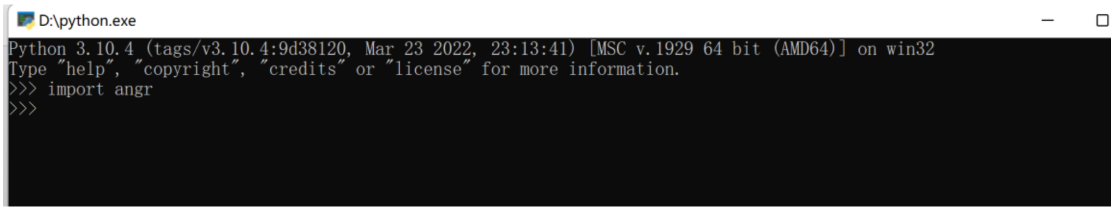
   在 windows 系统下安装 python3（此处我安装了 python 最新版），添加 python 环境变量。使用 win+R 输入 python 如图：



```
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

   在控制台输入 pip 语句安装 angr：

   语句为：pip install angr



```
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import angr
>>>
```

2. 分析代码：

   原代码片段为：

```
#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;
}
```

C++中的<<为左移，1<<i 即将 i 左移一位。&即按位与，for 循环即为将 u 的每一位都取出来比较，如果为 0，则 bit[0]++，否则就 bit[1]++。显然只有当 u 的二进制

中 1 的个数与 0 的个数相同时才能胜利。

使用 python 求解，第一种方法代码为：

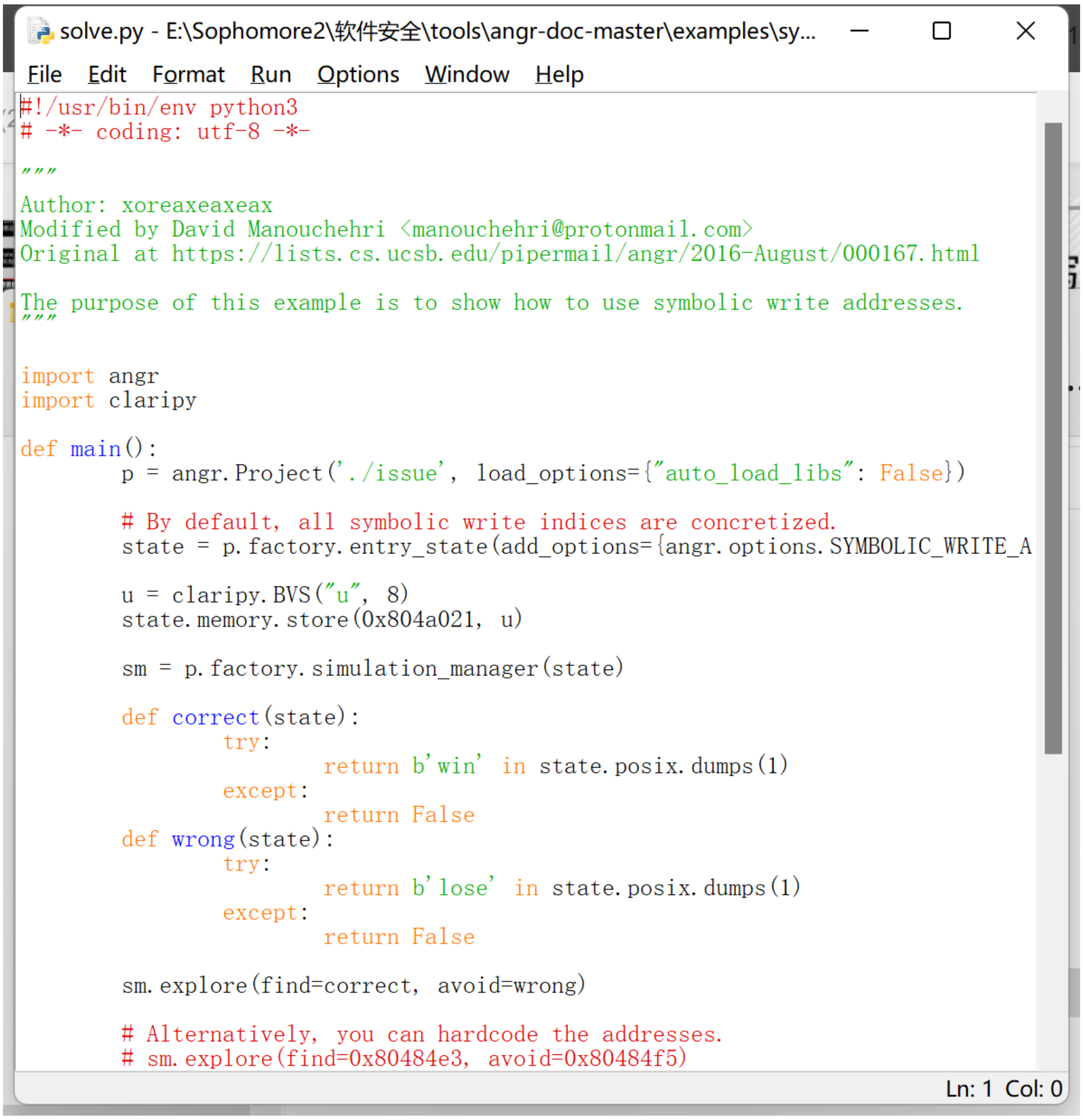
```
import angr
import claripy

def main():
    p = angr.Project(’./issue’, load_options={″auto_load_libs″: False}) #新建工程，导入二进制文件 issue，选择“不自动加载依赖项”

    state = p.factory.entry_state(add_options={angr.options.SYMBOLIC_WRITE_ADDRESSES})#初始化模拟程序状态的 simstate 对象 state，用以记录程序运行时的动态数据，包括内存、寄存器和符号信息等。

    u = claripy.BVS(″u″, 8)#创建符号变量 u，以 8 为 bitvector 形式存在
    state.memory.store(0x804a021, u)#将其存储到 u 的地址上。

    sm = p.factory.simulation_manager(state)#创建一个 Simulation Manager 对象，管理运行得到的状态对象

    def correct(state):
        try:
            return b’win’ in state.posix.dumps(1)#获得所有使得输出结果为 win 的标准输出
        except:
            return False
    def wrong(state):
        try:
            return b’lose’ in state.posix.dumps(1)
        except:
            return False

    sm.explore(find=correct, avoid=wrong)#使用动态符号执行，使用 explore 函数进行状态搜寻，只找出结果为 win 的输出，避免结果为 lose 的输出。

    # 也可以改成以下语句：
    # sm.explore(find=0x80484e3, avoid=0x80484f5)

    return sm.found[0].solver.eval_upto(u, 256)#约束求解，得到 state 后通过 solver 求解器求解 u 的值。
def test():
        good = set()
        for u in range(256):
            bits = [0, 0]
            for i in range(8):
                bits[u&(1<<i)!=0] += 1
            if bits[0] == bits[1]:
                good.add(u)#将能输出 win 的结果加入 good 集合中

        res = main()
        assert set(res) == good

    if __name__ == ’__main__’:
    print(repr(main()))
```

第二种方法代码为：

```
    import angr
    import claripy

    def hook_demo(state):
            state.regs.eax=0
    p   =   angr.Project(’./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_A
DDRESSES″})
    u = claripy.BVS(″u″, 8)
    state.memory.store(0x804a021, u)
    sm = p.factory.simulation_manager(state)
    sm.explore(find=0x080484DB)#此处指定分支语句条件,因为我们已经确定
唯一路径
    st=sm.found[0]
print(repr(st.solver.eval(u)))#eval（u）只输出一个结果
```

3. 实验验证：

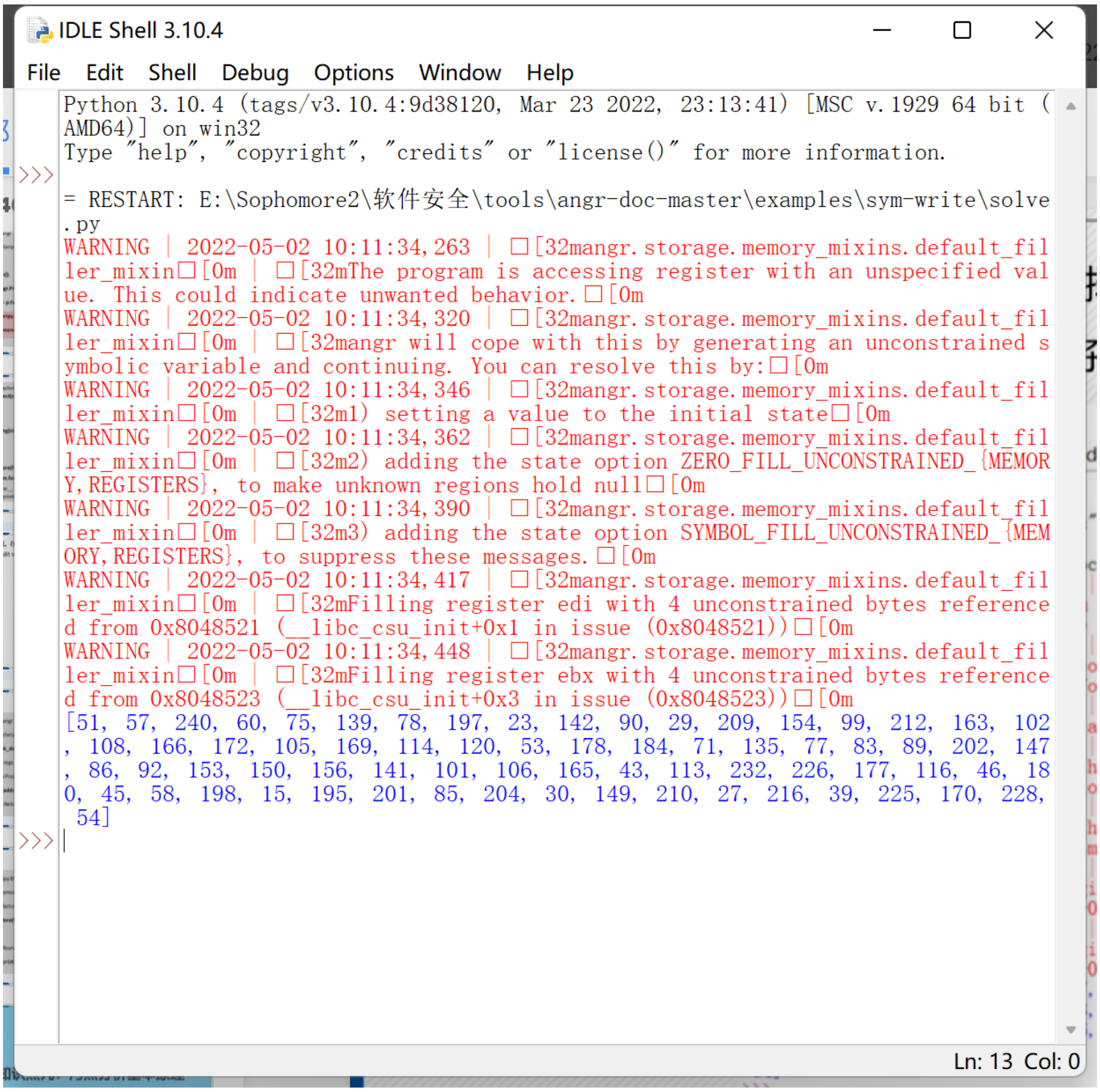
第一种方法：右键点击 solve.py 文件，选择 Edit with IDLE，可弹出以下对话框



solve.py - E:\Sophomore2\软件安全\tools\angr-doc-master\examples\sy...
File Edit Format Run Options Window Help
Ln: 1 Col: 0

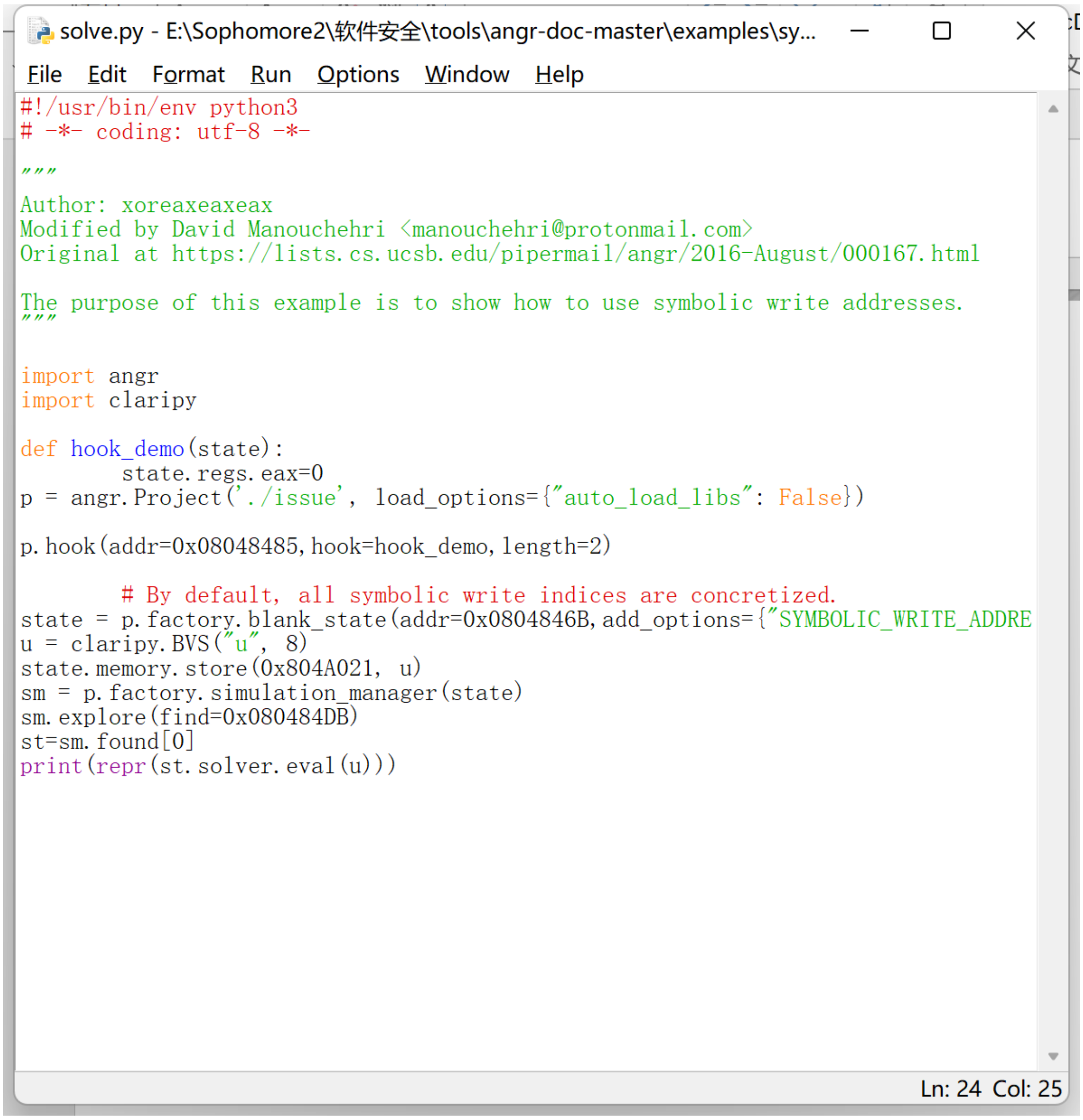
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Author: xoreaxeaxeax
Modified by David Manouchehri <manouchehri@protonmail.com>
Original at https://lists.cs.ucsb.edu/pipermail/angr/2016-August/000167.html

The purpose of this example is to show how to use symbolic write addresses.
"""

import angr
import claripy

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_A

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

	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)
```

选择 Run-run module:



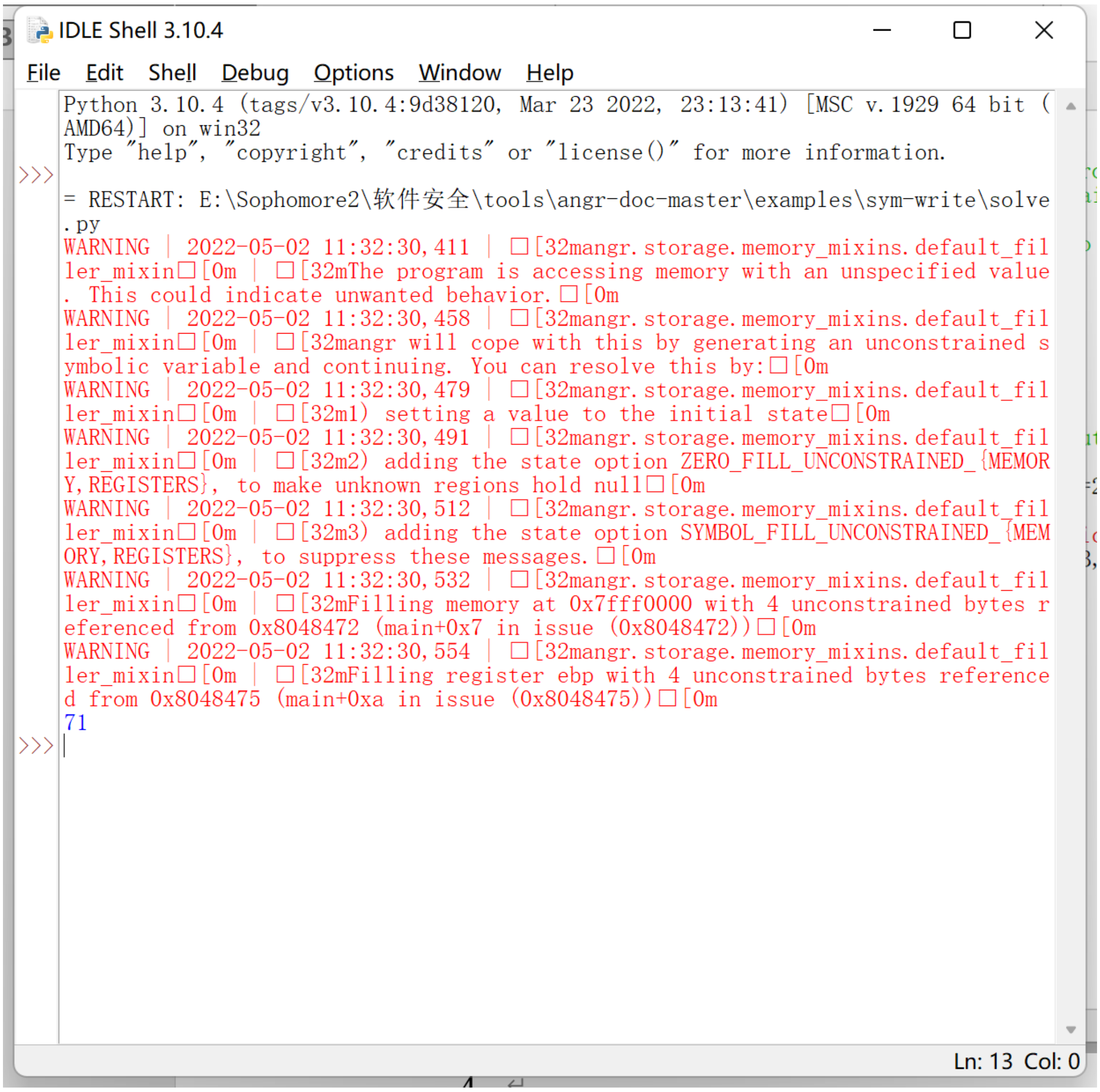
```
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: E:\Sophomore2\软件安全\tools\angr-doc-master\examples\sym-write\solve
.py
WARNING | 2022-05-02 10:11:34,263 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32mThe program is accessing register with an unspecified val
ue. This could indicate unwanted behavior.□[0m
WARNING | 2022-05-02 10:11:34,320 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32mangr will cope with this by generating an unconstrained s
ymbolic variable and continuing. You can resolve this by:□[0m
WARNING | 2022-05-02 10:11:34,346 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32m1) setting a value to the initial state□[0m
WARNING | 2022-05-02 10:11:34,362 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32m2) adding the state option ZERO_FILL_UNCONSTRAINED_{MEMOR
Y,REGISTERS}, to make unknown regions hold null□[0m
WARNING | 2022-05-02 10:11:34,390 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32m3) adding the state option SYMBOL_FILL_UNCONSTRAINED_{MEM
ORY,REGISTERS}, to suppress these messages.□[0m
WARNING | 2022-05-02 10:11:34,417 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32mFilling register edi with 4 unconstrained bytes reference
d from 0x8048521 (__libc_csu_init+0x1 in issue (0x8048521))□[0m
WARNING | 2022-05-02 10:11:34,448 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32mFilling register ebx with 4 unconstrained bytes reference
d from 0x8048523 (__libc_csu_init+0x3 in issue (0x8048523))□[0m
[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, 89, 202, 147
, 86, 92, 153, 150, 156, 141, 101, 106, 165, 43, 113, 232, 226, 177, 116, 46, 18
0, 45, 58, 198, 15, 195, 201, 85, 204, 30, 149, 210, 27, 216, 39, 225, 170, 228,
 54]
>>>
```

第二种方法：


solve.py - E:\Sophomore2\软件安全\tools\angr-doc-master\examples\sy...
File
Edit
Format
Run
Options
Window
Help
Ln: 24 Col: 25

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Author: xoreaxeaxeax
Modified by David Manouchehri <manouchehri@protonmail.com>
Original at https://lists.cs.ucsb.edu/pipermail/angr/2016-August/000167.html

The purpose of this example is to show how to use symbolic write addresses.
"""


import angr
import claripy

def hook_demo(state):
        state.regs.eax=0
p = angr.Project('./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_ADDRE
u = claripy.BVS("u", 8)
state.memory.store(0x804A021, u)
sm = p.factory.simulation_manager(state)
sm.explore(find=0x080484DB)
st=sm.found[0]
print(repr(st.solver.eval(u)))
```

运行结果为：



```
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: E:\Sophomore2\软件安全\tools\angr-doc-master\examples\sym-write\solve
.py
WARNING | 2022-05-02 11:32:30,411 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32mThe program is accessing memory with an unspecified value
. This could indicate unwanted behavior.□[0m
WARNING | 2022-05-02 11:32:30,458 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32mangr will cope with this by generating an unconstrained s
ymbolic variable and continuing. You can resolve this by:□[0m
WARNING | 2022-05-02 11:32:30,479 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32m1) setting a value to the initial state□[0m
WARNING | 2022-05-02 11:32:30,491 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32m2) adding the state option ZERO_FILL_UNCONSTRAINED_{MEMOR
Y,REGISTERS}, to make unknown regions hold null□[0m
WARNING | 2022-05-02 11:32:30,512 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32m3) adding the state option SYMBOL_FILL_UNCONSTRAINED_{MEM
ORY,REGISTERS}, to suppress these messages.□[0m
WARNING | 2022-05-02 11:32:30,532 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32mFilling memory at 0x7fff0000 with 4 unconstrained bytes r
eferenced from 0x8048472 (main+0x7 in issue (0x8048472))□[0m
WARNING | 2022-05-02 11:32:30,554 | □[32mangr.storage.memory_mixins.default_fil
ler_mixin□[0m | □[32mFilling register ebp with 4 unconstrained bytes reference
d from 0x8048475 (main+0xa in issue (0x8048475))□[0m
71
>>>
```

心得体会：

Angr 是一个基于 python 的二进制漏洞分析框架，它将以前多种分析技术集成进来，既能够执行动态的符号执行分析，也能够进行多种动态分析。

如何使用 Angr：可以从 Angr 的 example 中分析其运行代码逻辑，学习代码编写思路，参考官方网站所给出的指导手册进行 Angr 的编写使用。在 CSDN 网站上也有很多人已经总结了较为完善的基于 python 的 Angr 编写，可以进行参考。

如何使用 Angr 解决实际问题：可以在 Angr 之上开发以下工作：如 angrop：rop 链自动化生成器；Patcherex：二进制文件自动化 patch 引擎；Driller：用符号执行增强 AFL 的下一代 fuzzer 等。