

CFG 与符号执行

符号执行概述：

符号执行技术使用符号值代替数字值执行程序，得到的变量的值是由输入变量的符号值和常量组成的表达式。符号执行技术首先由 King 在 1976 年提出，经过三十多年的发展，现在仍然被广泛研究，它在软件测试和程序验证中发挥着重要作用。符号执行是一种重要的形式化方法和静态分析技术，它使用数学和逻辑首先定义一些基本概念。程序的路径

(path) 是程序的一个语句序列，这个语句序列包括程序的一些顺序的代码片段，代码片段之间的连接是由于分支语句导致的控制转移。一个路径是可行的 (feasible)，是指存在程序输入变量的至少一组值，如果以这组值作为输入，程序将沿着这条路径执行。否则，路径就是不可行的 (infeasible)。路径条件 (path condition, PC) 是针对一个路径的，它是一个关于程序输入变量的符号值的约束，一组输入值使得程序沿着这条路径执行当且仅当这组输入值满足这条路径的路径条件。

angr 框架介绍：

在二进制代码中寻找并且利用漏洞是一项非常具有挑战性的工作，它的挑战性主要在于人工很难直观的看出二进制代码中的数据结构、控制流信息等。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，方便后续的安全研究人员的开发。---它能够进行动态的符号执行分析（如，KLEE 和 Mayhem），也能够进行多种静态分析。

演示示例(test.c)：

```
char *sneaky = "SOSNEAKY";
int authenticate(char *username, char *password)
{
    char stored_pw[9];
    stored_pw[8] = 0;
    int pwfile;
    if (strcmp(password, sneaky) == 0) return 1;
    pwfile = open(username, O_RDONLY);
    read(pwfile, stored_pw, 8);
    if (strcmp(password, stored_pw) == 0) return 1;
    return 0;
}
```

```

int accepted()
{
    printf("Welcome to the admin console, trusted user!");
}
int rejected()
{
    printf("Go away!");
    exit(1);
}

int main(int argc, char **argv)
{
    char username[9];
    char password[9];
    int authed;
    username[8] = 0;
    password[8] = 0;
    printf("Username: n");
    read(0, username, 8);
    read(0, &authed, 1);
    printf("Password: n");
    read(0, password, 8);
    read(0, &authed, 1);
    authed = authenticate(username, password);
    if (authed) accepted();
    else rejected();
}

```

样例程序的功能就是让你输入用户名和密码，然后 `authenticate` 函数会进行检验，如果失败就显示 `Go away`，反之就显示认证成功。

angr 脚本(solver.py)(解释见代码注释):

```
import angr

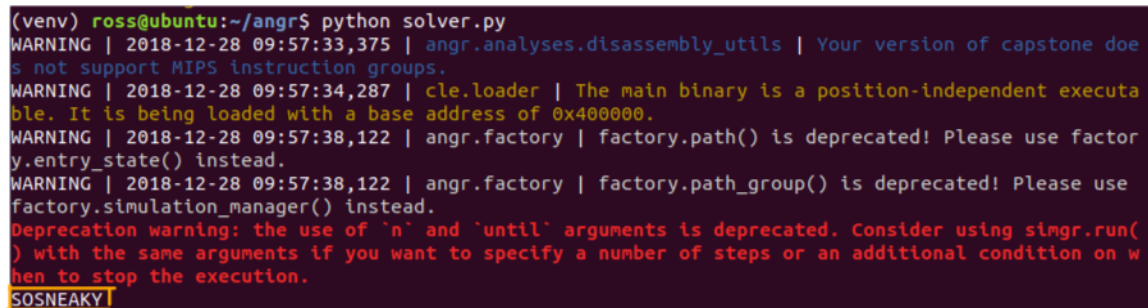
def basic_symbolic_execution():
    p = angr.Project('test')
    # 新建一个angr的工程, 括号中是目标二进制程序的路径
    state = p.factory.entry_state()
    # 接着新建一个SimState的对象
    path = p.factory.path(state)
    # 使用factory.path这个容器获取state的起点path对象
    # 根据前面获取的函数入口点的path对象, 利用path_group容器获取沿着path开端下面将会执行的path列表
    pathgroup = p.factory.path_group(path)
    # 接下来就让pathgroup对象一直执行下去, 直到执行到可选择的路径个数大于一个, 即产生选择分支的时候, 再停止。
    pathgroup.step(until=lambda lpg: len(lpg.active) > 1)
    # 对应在上述的简单程序中authenticate函数的 if (strcmp(password, sneaky) == 0)这个条件判断语句

    input_0 = pathgroup.active[0].state.posix.dumps(0)
    # dump出所有分支的内容, 看看哪个答案应该是最可能的
    input_1 = pathgroup.active[1].state.posix.dumps(0)

    if 'SOSNEAKY' in input_0:
        return input_0
    else:
        return input_1

def test():
    pass
if __name__ == '__main__':
    print basic_symbolic_execution()
```

运行截图:



```
(venv) ross@ubuntu:~/angr$ python solver.py
WARNING | 2018-12-28 09:57:33,375 | angr.analyses.disassembly_utils | Your version of capstone does not support MIPS instruction groups.
WARNING | 2018-12-28 09:57:34,287 | cle.loader | The main binary is a position-independent executable. It is being loaded with a base address of 0x400000.
WARNING | 2018-12-28 09:57:38,122 | angr.factory | factory.path() is deprecated! Please use factory.entry_state() instead.
WARNING | 2018-12-28 09:57:38,122 | angr.factory | factory.path_group() is deprecated! Please use factory.simulation_manager() instead.
Deprecation warning: the use of 'n' and 'until' arguments is deprecated. Consider using simgr.run() with the same arguments if you want to specify a number of steps or an additional condition on when to stop the execution.
SOSNEAKY
```

CFG 生成:

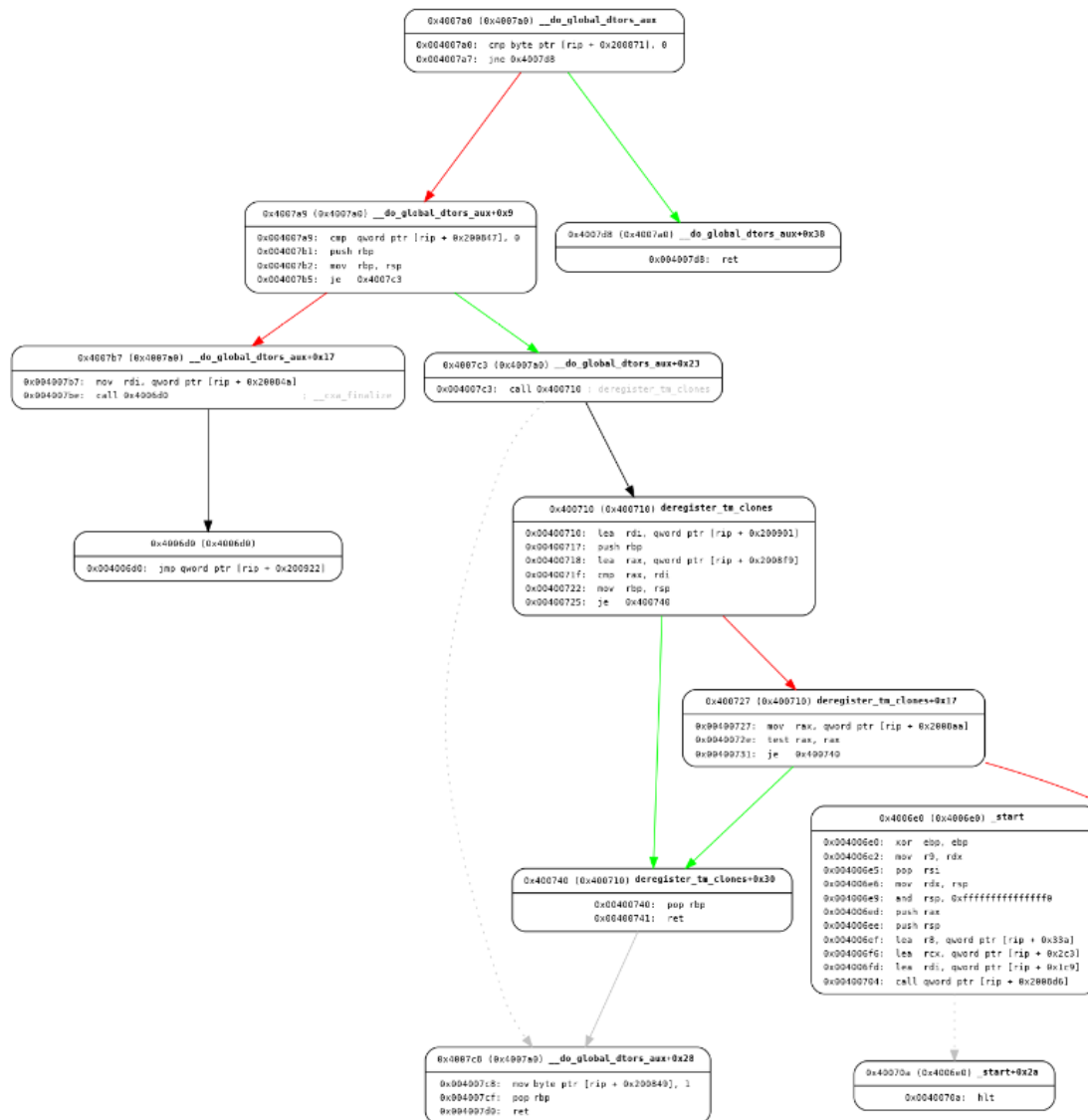
利用 angr-utils, cfg-explorer 库等, 可以根据可执行程序生成 CFG 图, 以 angr-utils 为例代码如下:

```

import angr
from angrutils import *
binary = "./test"
b=angr.Project(binary, load_options={'auto_load_libs':False})
cfg=b.analyses.CFG()
plot_cfg(cfg, "cfgfast", asminst=True, remove_imports=True, remove_path_terminator=True)

```

运行结果如图所示(局部图, 由于图片太大, 所以截取部分以作展示):



总结:

通过本次实验了解了符号执行的相关内容, 对 angr 库和如何根据可执行文件生成 CFG 也有了一定的了解。