实验报告

课程名称：软件测试

实验名称：KLEE 实验

专业班级：软件工程 3 班

学　　号：1612880

姓　　名：潘忠杰

2018 年 12 月 30 日

实验一

实验名称	KLEE 实验		
实验地点	5 区机房	实验时间	2018.12

实验目的和要求

使用 KLEE 测试软件对含有缺陷的 C 语言代码进行缺陷测试。

实验环境

Ubuntu18.04+docker+klee

实验过程

1. 前往 docker 官方网站下载并安装 docker；
2. 在命令行中使用 sudo apt-get install klee 安装 klee；
3. 在 docker 创建一个永久性容器用于进行代码测试；
4. 首先在容器中进入 klee_src 文件夹，之后进入 examples 文件夹，找到 tutorial1 对应的文件进行测试，将生成的 klee_last 文件夹目录截图

```
klee@e2857d66e018:~/klee_src/examples/get_sign$ klee get_sign.bc
KLEE: output directory is "/home/klee/klee_src/examples/get_sign/klee-out-0"
KLEE: Using STP solver backend

KLEE: done: total instructions = 31
KLEE: done: completed paths = 3
KLEE: done: generated tests = 3
klee@e2857d66e018:~/klee_src/examples/get_sign$ ls klee-last/
assembly.ll  messages.txt  run.stats         test000002.ktest  warnings.txt
info         run.istats    test000001.ktest  test000003.ktest
```

5. 其次在容器中找到 regexp 文件夹，将内部文件进行测试，将生成的 klee_last 文件夹目录截图

```
klee@e2857d66e018:~/klee_src/examples/regexp$ ls
Regexp.bc  Regexp.c  klee-last  klee-out-0  notes.txt
klee@e2857d66e018:~/klee_src/examples/regexp$ ls klee-last/
assembly.ll        test000003.ktest    test000008.ktest    test000013.ktest
info               test000004.ktest    test000009.kquery   test000014.ktest
messages.txt       test000005.ktest    test000009.ktest    test000015.ktest
run.istats         test000006.ktest    test000009.ptr.err  test000016.ktest
run.stats          test000007.kquery   test000010.ktest    warnings.txt
test000001.ktest   test000007.ktest    test000011.ktest
test000002.ktest   test000007.ptr.err  test000012.ktest
```

6. 之后在本地编写一个具有五种缺陷的 C 语言代码，使用 docker cp 指令将文件复制进入 docker 容器，对代码进行测试，得到五个错误，错误截图如下

```
klee@e2857d66e018:~/klee_src/examples/mytest$ klee code.bc
KLEE: output directory is "/home/klee/klee_src/examples/mytest/klee-out
-18"
KLEE: Using STP solver backend
WARNING: this target does not support the llvm.stacksave intrinsic.
KLEE: WARNING: undefined reference to function: test5
KLEE: NOTE: found huge malloc, returning 0
KLEE: ERROR: /home/klee/klee_src/examples/mytest/code.c:17: concretized
 symbolic size
KLEE: NOTE: now ignoring this error at this location
KLEE: ERROR: /home/klee/klee_src/examples/mytest/code.c:19: memory erro
r: out of bound pointer
KLEE: NOTE: now ignoring this error at this location
KLEE: ERROR: /home/klee/klee_src/examples/mytest/code.c:25: divide by z
ero
KLEE: NOTE: now ignoring this error at this location
KLEE: WARNING ONCE: calling external: test5() at /home/klee/klee_src/ex
amples/mytest/code.c:30
KLEE: ERROR: /home/klee/klee_src/examples/mytest/code.c:29: failed exte
rnal call: test5
KLEE: NOTE: now ignoring this error at this location

KLEE: done: total instructions = 90
KLEE: done: completed paths = 5
KLEE: done: generated tests = 4
```

7. 找到 klee_last 文件夹目录，并且将所有错误文件保存

```
Error: concretized symbolic size
File: /home/klee/klee_src/examples/mytest/code.c
Line: 17
assembly.ll line: 37
Stack:
        #000000037 in test2 (x) at /home/klee/klee_src/examples/mytest/code.c:17
        #100000108 in test (err=23709168) at /home/klee/klee_src/examples/mytest/code.c:34
        #200000129 in main () at /home/klee/klee_src/examples/mytest/code.c:44
Info:
  size expr: (Mul w64 4
                    (ZExt w64 (ReadLSB w32 4 err)))
  concretization : 0
  unbound example: 4
```

```
Error: memory error: out of bound pointer
File: /home/klee/klee_src/examples/mytest/code.c
Line: 19
assembly.ll line: 52
Stack:
        #000000052 in test2 (x) at /home/klee/klee_src/examples/mytest/code.c:19
        #100000108 in test (err=23709168) at /home/klee/klee_src/examples/mytest/code.c:34
        #200000129 in main () at /home/klee/klee_src/examples/mytest/code.c:44
Info:
        address: 0
        next: object at 23699568 of size 4
                MO7[4] allocated at global:.str
```

```
Error: failed external call: test5
File: /home/klee/klee_src/examples/mytest/code.c
Line: 29
assembly.ll line: 91
Stack:
        #000000091 in test4 (x) at /home/klee/klee_src/examples/mytest/code.c:29
        #100000116 in test (err=23709168) at /home/klee/klee_src/examples/mytest/code.c:36
        #200000129 in main () at /home/klee/klee_src/examples/mytest/code.c:44
```

```
Error: divide by zero
File: /home/klee/klee_src/examples/mytest/code.c
Line: 25
assembly.ll line: 79
Stack:
        #000000141 in klee_div_zero_check (z) at /home/klee/klee_src/runtime/Intrinsic/klee_div_zero_check.c:14
        #100000079 in test3 (x) at /home/klee/klee_src/examples/mytest/code.c:25
        #200000112 in test (err=23709168) at /home/klee/klee_src/examples/mytest/code.c:35
        #300000129 in main () at /home/klee/klee_src/examples/mytest/code.c:44
```

心得体会
通过本次实验，我对 klee 软件有了初步的了解，并且对一些基本用法有所掌握，成功测试了待测程序，加深了对于程序缺陷的理解和认识。

附：源程序

```
#include <klee/klee.h>


struct
{
    int id[3];
    char name[5];
    int money[6];
    char type[3];
}base[2];


void test1(int x) {
    int *ptr;
```

```
    if (ptr == &x) {
        x = 1;
    }
}

void test2(int x) {
    int a[x];
    for(int i = 0; i <= x; i++) {
        a[i] = i;
    }
}

void test3(int x) {
    int b = 1, c;
    c = b / x;
}

void test4(int x) {
    int b = 2;
    x = b;
    test5();
}
```

```
void test(int* err) {
    test1(err[0]);
    test2(err[1]);
    test3(err[2]);
    test4(err[3]);
}

#define SIZE 4

int main() {
    int err[SIZE];
    klee_make_symbolic(err, sizeof err, "err");
    test(err);
    return 0;
}
```