**《软件安全》实验报告**

姓名：禹相祐 学号：2312900 班级：计算机科学与技术

**实验名称：**

AFL模糊测试

**实验要求：**

根据课本7.45章节，复现AFL在Kali的安装与应用，查阅资料理解覆盖引导和文件变异的概念和含义。

**实验过程：**

1. **Kali & AFL 的安装**

根据书一步一步来即可，此处不作展示。

1. **利用AFL实现模糊测试**
   1. **创建测试程序mytest.c**

首先创建一个demo文件夹，再在内创建一个mytest.c文件，代码如下：

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  int main(int argc, char **\*\***argv) {    char ptr[20];  **if**(argc>1){             FILE **\***fp **=** fopen(argv[1], "r");             fgets(ptr, sizeof(ptr), fp);    }  **else**{            fgets(ptr, sizeof(ptr), stdin);    }    printf("%s", ptr);  **if**(ptr[0] **==** 'd') {  **if**(ptr[1] **==** 'e') {  **if**(ptr[2] **==** 'a') {  **if**(ptr[3] **==** 'd') {  **if**(ptr[4] **==** 'b') {  **if**(ptr[5] **==** 'e') {  **if**(ptr[6] **==** 'e') {  **if**(ptr[7] **==** 'f') {                                                                    abort();                                                            }  **else**   printf("%c",ptr[7]);                                                     }  **else**    printf("%c",ptr[6]);                                            }  **else**    printf("%c",ptr[5]);  }  **else**  }  **else**  printf("%c",ptr[4]);   printf("%c",ptr[3]);   }  **else**  }  **else**  printf("%c",ptr[2]);   printf("%c",ptr[1]);   }  **else**  printf("%c",ptr[0]);  **return** 0;   } |

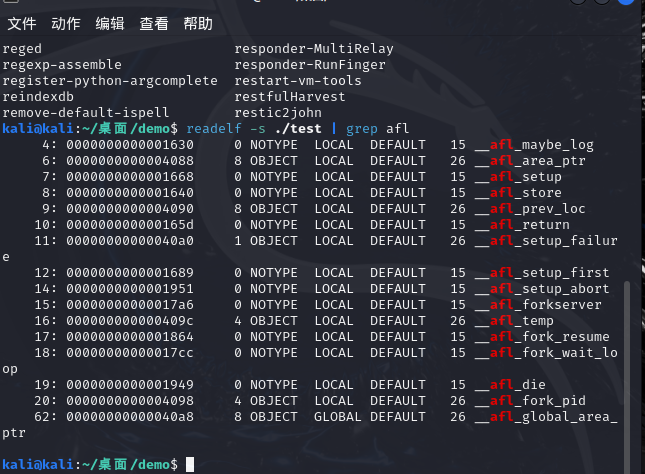
从代码可知：当输入为string = “deadbeef”时，程序会捕获到一个异常并且终止。

* 1. **使用linux的编译器加速模糊测试**

首先，我们在命令行输入: afl-gcc -o test mytest.c 进行编译，如图：

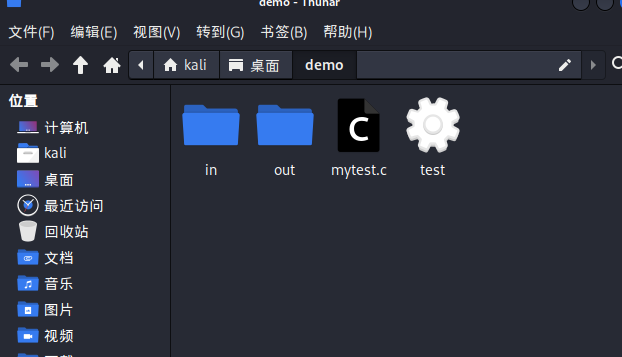


然后输入: readelf -s ./test | grep afl 来验证插桩符号，如图：

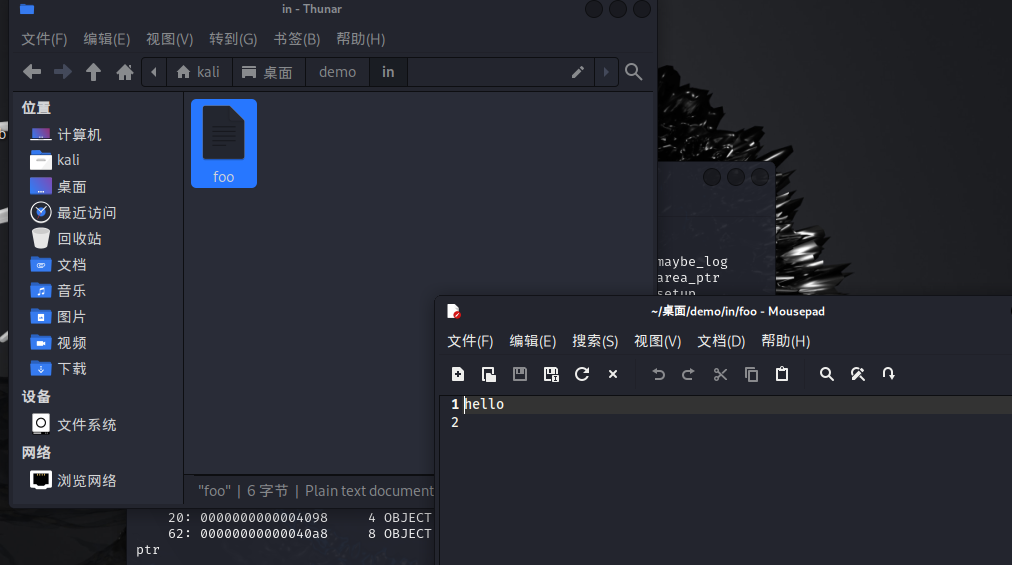


* 1. **在demo文件夹内创建in和out文件夹实现输入输出**

通过命令： mkdir in out 创建in 和 out文件夹，如图：

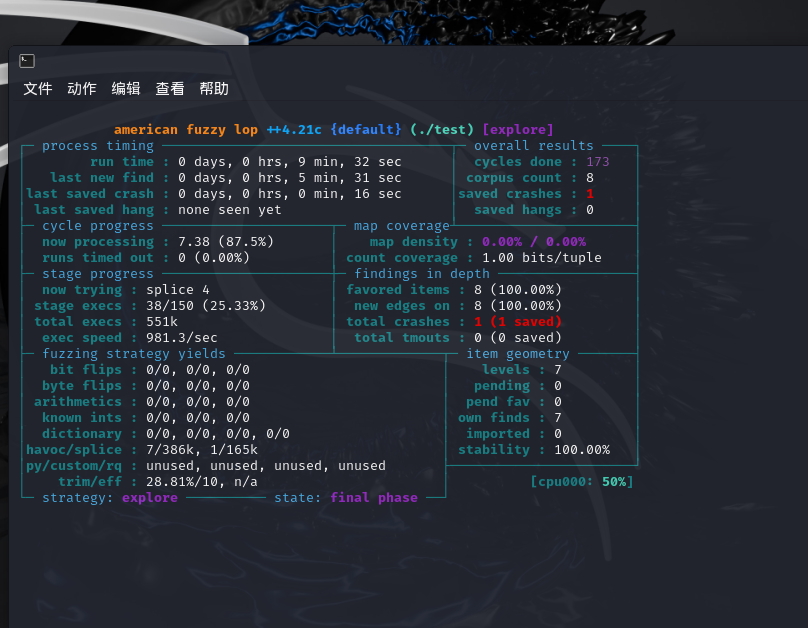


然后往in文件夹内创建一个包含”hello”的文件。通过命令： echo hello> in /foo 来实现在in文件夹内创建一个名字为foo、内容为string=“hello”的文件,如图：

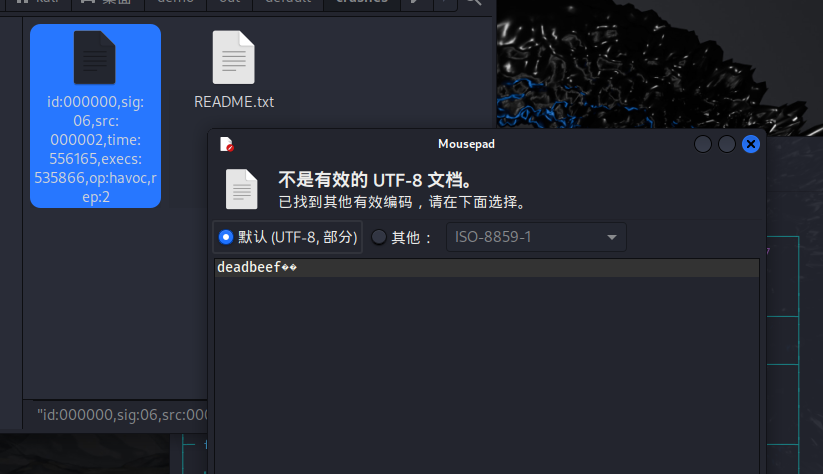


* 1. **开始模糊测试**

通过命令： afl-fuzz -i in -o out -- ./test @@ 启动模糊测试，并等待测试产生了一个crash后如图：



产生的crash可以在out文件夹内找到,如图：

****

一样为“deadbeef”，至此结束。

**心得体会：**

通过此次实验，让我对AFL的工作流程有了更深的认识：

1. 对c语言程序编译后进行插桩，以记录代码的覆盖率；
2. 选择输入文件作为最初始的测试集加入输入队列；
3. 按照一定的规则调整输入，若某次的输入更改了覆盖率，就放进输入队列；
4. 一直进行直到至少有一个crash产生。