

《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。

2. 利用 AFL 实现模糊测试

2.1 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);
    }
}
```

```

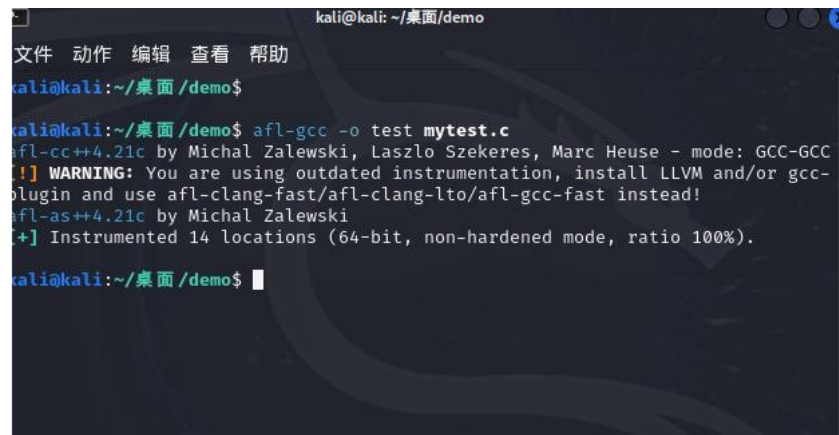
}
printf("%s", ptr);
if(ptr[0] == 'd') {
    if(ptr[1] == 'e') {
        if(ptr[2] == 'a') {
            if(ptr[3] == 'd') {
                if(ptr[4] == 'b') {
                    if(ptr[5] == 'e') {
                        if(ptr[6] == 'e') {
                            if(ptr[7] == 'f') {
                                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。

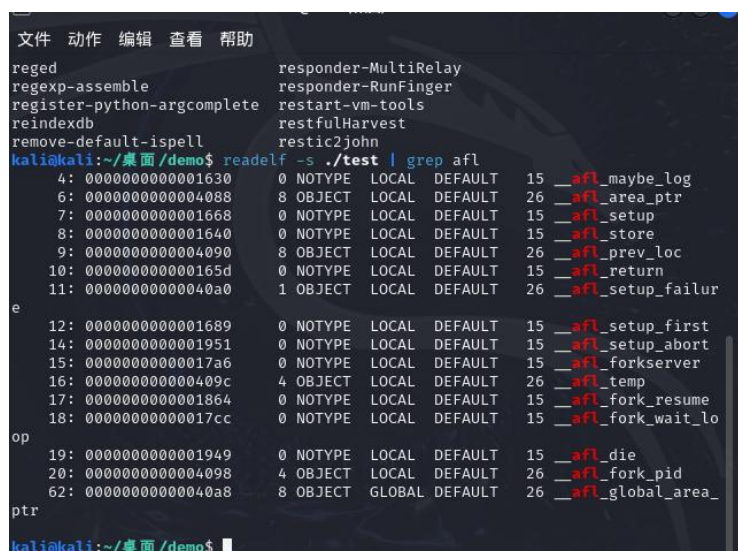
2.2 使用 linux 的编译器加速模糊测试

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



```
kali@kali: ~/桌面/demo
文件 动作 编辑 查看 帮助
kali@kali:~/桌面/demo$
kali@kali:~/桌面/demo$ afl-gcc -o test mytest.c
afl-cc++4.21c by Michal Zalewski, Laszlo Szekeres, Marc Heuse - mode: GCC-GCC
[!] WARNING: You are using outdated instrumentation, install LLVM and/or gcc-
plugin and use afl-clang-fast/afl-clang-lto/afl-gcc-fast instead!
afl-as++4.21c by Michal Zalewski
[+] Instrumented 14 locations (64-bit, non-hardened mode, ratio 100%).
kali@kali:~/桌面/demo$
```

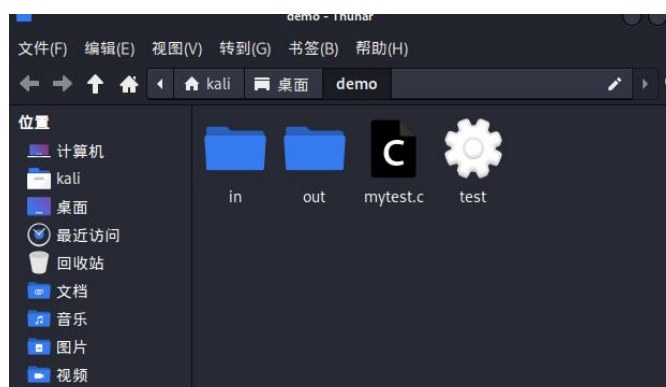
然后输入：`readelf -s ./test | grep afl` 来验证插桩符号，如图：



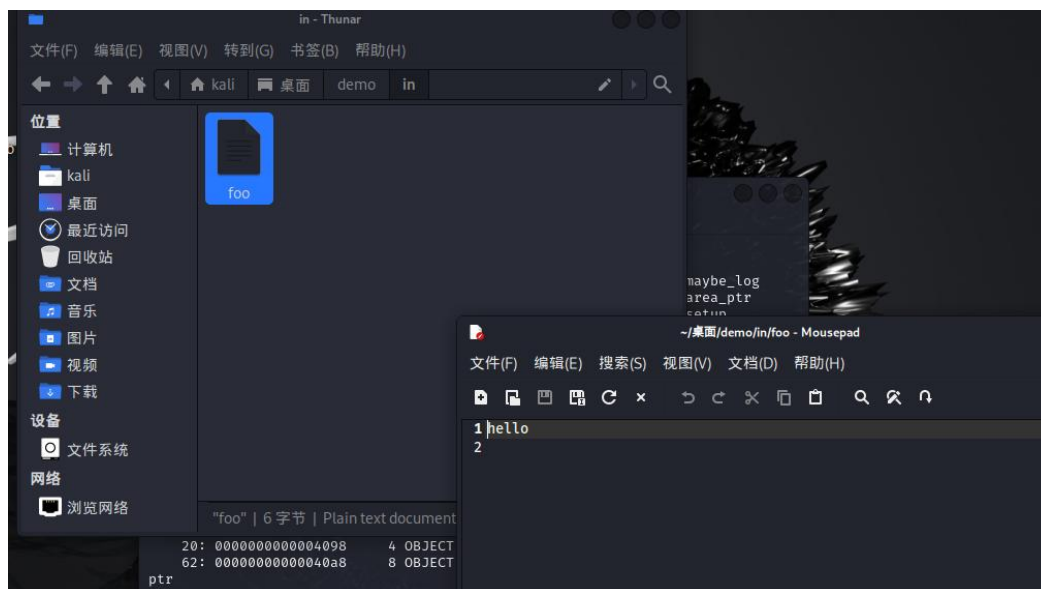
```
文件 动作 编辑 查看 帮助
reged responder-MultiRelay
regexp-assemble responder-RunFinger
register-python-argcomplete restart-vm-tools
reindexdb restfulHarvest
remove-default-ispell restic2john
kali@kali:~/桌面/demo$ readelf -s ./test | grep afl
4: 0000000000001630 0 NOTYPE LOCAL DEFAULT 15 __afl_maybe_log
6: 0000000000004088 8 OBJECT LOCAL DEFAULT 26 __afl_area_ptr
7: 0000000000001668 0 NOTYPE LOCAL DEFAULT 15 __afl_setup
8: 0000000000001640 0 NOTYPE LOCAL DEFAULT 15 __afl_store
9: 0000000000004090 8 OBJECT LOCAL DEFAULT 26 __afl_prev_loc
10: 000000000000165d 0 NOTYPE LOCAL DEFAULT 15 __afl_return
11: 00000000000040a0 1 OBJECT LOCAL DEFAULT 26 __afl_setup_failur
12: 0000000000001689 0 NOTYPE LOCAL DEFAULT 15 __afl_setup_first
14: 0000000000001951 0 NOTYPE LOCAL DEFAULT 15 __afl_setup_abort
15: 00000000000017a6 0 NOTYPE LOCAL DEFAULT 15 __afl_forkserver
16: 000000000000409c 4 OBJECT LOCAL DEFAULT 26 __afl_temp
17: 0000000000001864 0 NOTYPE LOCAL DEFAULT 15 __afl_fork_resume
18: 00000000000017cc 0 NOTYPE LOCAL DEFAULT 15 __afl_fork_wait_lo
19: 0000000000001949 0 NOTYPE LOCAL DEFAULT 15 __afl_die
20: 0000000000004098 4 OBJECT LOCAL DEFAULT 26 __afl_fork_pid
62: 00000000000040a8 8 OBJECT GLOBAL DEFAULT 26 __afl_global_area_
ptr
kali@kali:~/桌面/demo$
```

2.3 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:

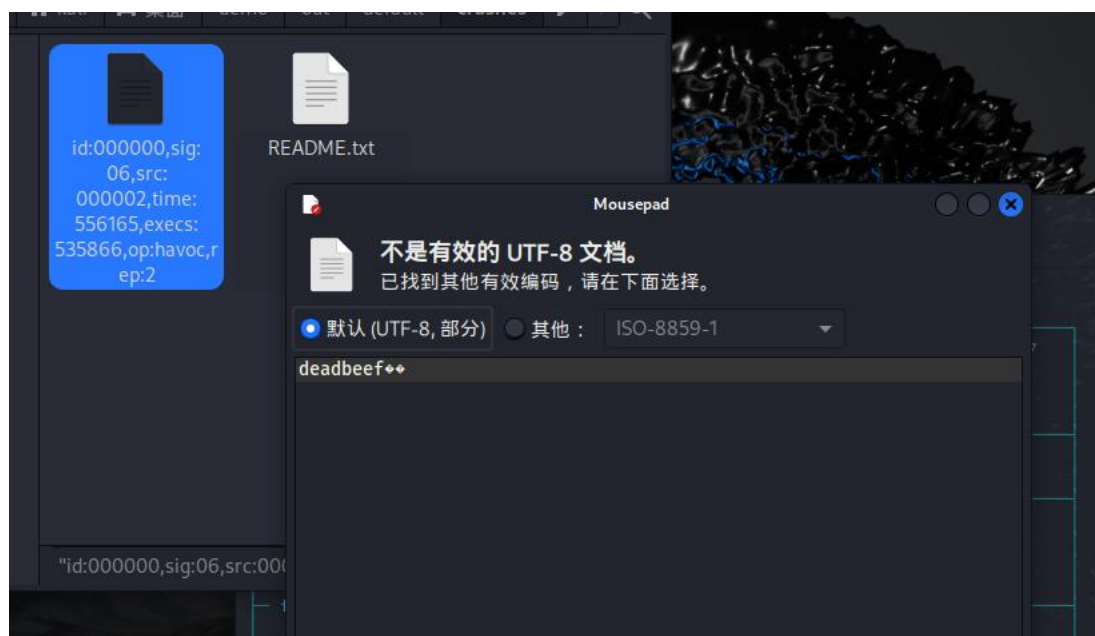


2.4 开始模糊测试

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。