《漏洞利用及渗透测试基础》实验报告

姓名：齐明杰　　学号：2113997　　班级： 信安 2 班

实验名称：

AFL 模糊测试实验

实验要求：

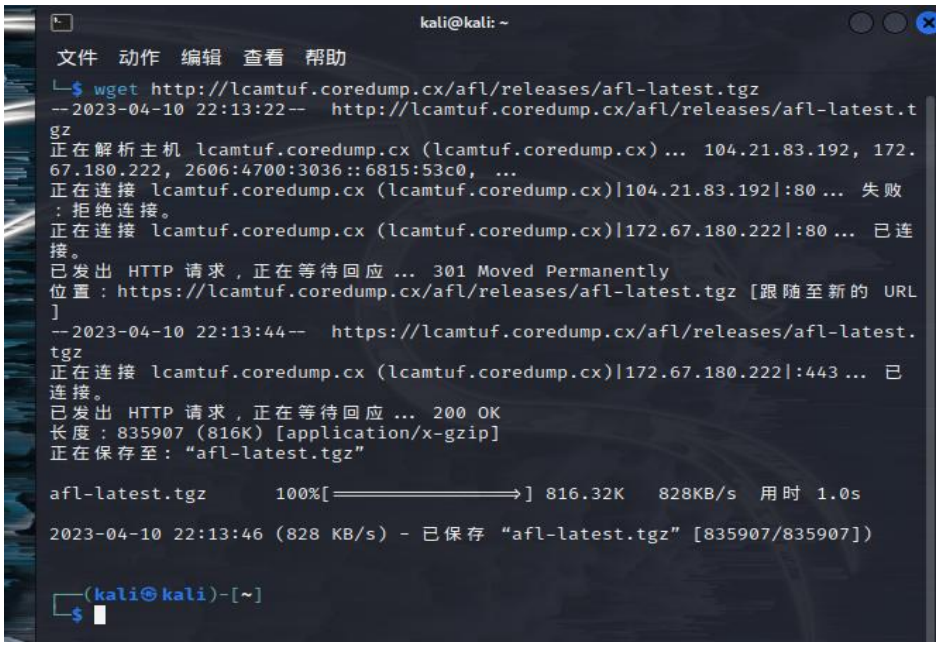
根据课本 7.4.5 章节，复现 AFL 在 KALI 下的安装、应用查阅资料理解覆盖引导和文件变异的概念和含义。

实验过程：

一、 安装 AFL

1. 首先使用命令下载安装包：

wget http://lcamtuf.coredump.cx/afl/releases/afl-latest.tgz

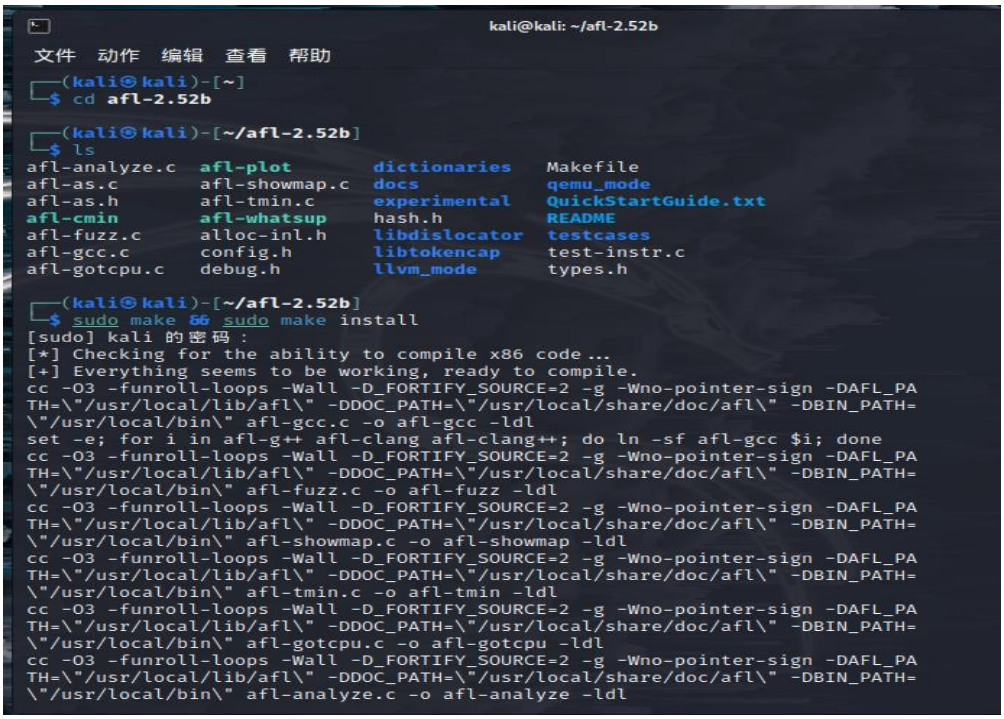


2. 解压命令：tar xvf afl-latest.tgz

3. 进入目标文件夹并且编译 AFL：

cd afl-2.52b

sudo make && sudo make install



```
文件  动作  编辑  查看  帮助
PATH=\"/usr/local/bin\" test-instr.c -o test-instr -ldl
echo 0 | ./afl-showmap -m none -q -o .test-instr0 ./test-instr
echo 1 | ./afl-showmap -m none -q -o .test-instr1 ./test-instr
[+] All right, the instrumentation seems to be working!
[+] LLVM users: see llvm_mode/README.llvm for a faster alternative to afl-gcc
.
[+] All done! Be sure to review README - it's pretty short and useful.


mkdir -p -m 755 ${DESTDIR}/usr/local/bin ${DESTDIR}/usr/local/lib/afl ${DESTD
IR}/usr/local/share/doc/afl ${DESTDIR}/usr/local/share/afl
rm -f ${DESTDIR}/usr/local/bin/afl-plot.sh
install -m 755 afl-gcc afl-fuzz afl-showmap afl-tmin afl-gotcpu afl-analyze a
fl-plot afl-cmin afl-whatsup ${DESTDIR}/usr/local/bin
rm -f ${DESTDIR}/usr/local/bin/afl-as
if [ -f afl-qemu-trace ]; then install -m 755 afl-qemu-trace ${DESTDIR}/usr/l
ocal/bin; fi
if [ -f afl-clang-fast -a -f afl-llvm-pass.so -a -f afl-llvm-rt.o ]; then set
 -e; install -m 755 afl-clang-fast ${DESTDIR}/usr/local/bin; ln -sf afl-clang
-fast ${DESTDIR}/usr/local/bin/afl-clang-fast++; install -m 755 afl-llvm-pass
.so afl-llvm-rt.o ${DESTDIR}/usr/local/lib/afl; fi
if [ -f afl-llvm-rt-32.o ]; then set -e; install -m 755 afl-llvm-rt-32.o ${DE
STDIR}/usr/local/lib/afl; fi
if [ -f afl-llvm-rt-64.o ]; then set -e; install -m 755 afl-llvm-rt-64.o ${DE
STDIR}/usr/local/lib/afl; fi
set -e; for i in afl-g++ afl-clang afl-clang++; do ln -sf afl-gcc ${DESTDIR}/
usr/local/bin/$i; done
install -m 755 afl-as ${DESTDIR}/usr/local/lib/afl
ln -sf afl-as ${DESTDIR}/usr/local/lib/afl/as
install -m 644 docs/README docs/ChangeLog docs/*.txt ${DESTDIR}/usr/local/sha
re/doc/afl
cp -r testcases/ ${DESTDIR}/usr/local/share/afl
cp -r dictionaries/ ${DESTDIR}/usr/local/share/afl

┌──(kali㉿kali)-[~/afl-2.52b]
└─$
```

之后，我们可以在/usr/local/bin 路径下找到 AFL：

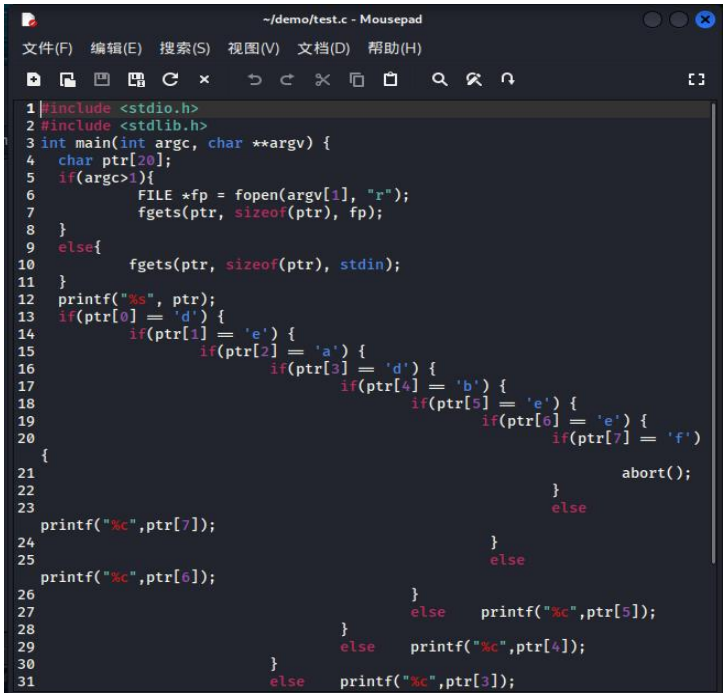
```
cp -r dictionaries/ ${DESTDIR}/usr/local/share/afl

┌──(kali㉿kali)-[~/afl-2.52b]
└─$ ls /usr/local/bin

afl-analyze  afl-clang++  afl-fuzz  afl-gcc     afl-plot     afl-tmin     mount-shared-folders
afl-clang    afl-cmin     afl-g++   afl-gotcpu  afl-showmap  afl-whatsup  restart-vm-tools
```

```
┌──(kali㉿kali)-[~/afl-2.52b]
└─$ ls /usr/local/bin/afl*
/usr/local/bin/afl-analyze  /usr/local/bin/afl-fuzz    /usr/local/bin/afl-plot
/usr/local/bin/afl-clang    /usr/local/bin/afl-g++     /usr/local/bin/afl-showmap
/usr/local/bin/afl-clang++  /usr/local/bin/afl-gcc     /usr/local/bin/afl-tmin
/usr/local/bin/afl-cmin     /usr/local/bin/afl-gotcpu  /usr/local/bin/afl-whatsup
```

二、 AFL 测试

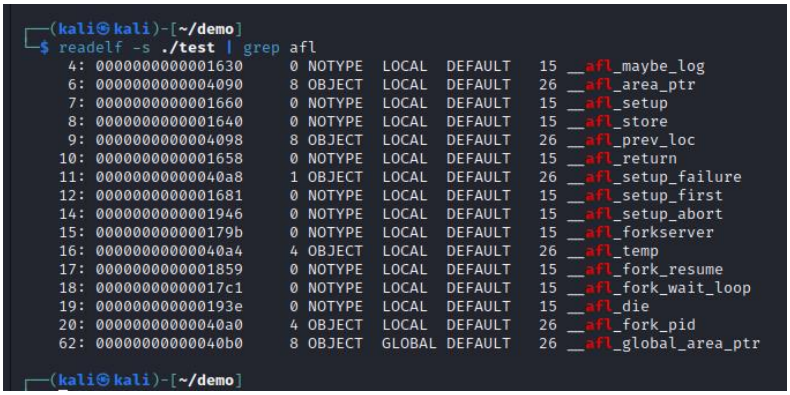
1. 创建本次实验的程序

新建文件夹 demo，并创建本次实验的程序 test.c，该代码编译后得到的程序如果被传入“deadbeef”则会终止，如果传入其他字符会原样输出：



接下来使用 afl 的编译器编译，输入命令：afl-gcc -o test test.c，结果如下：

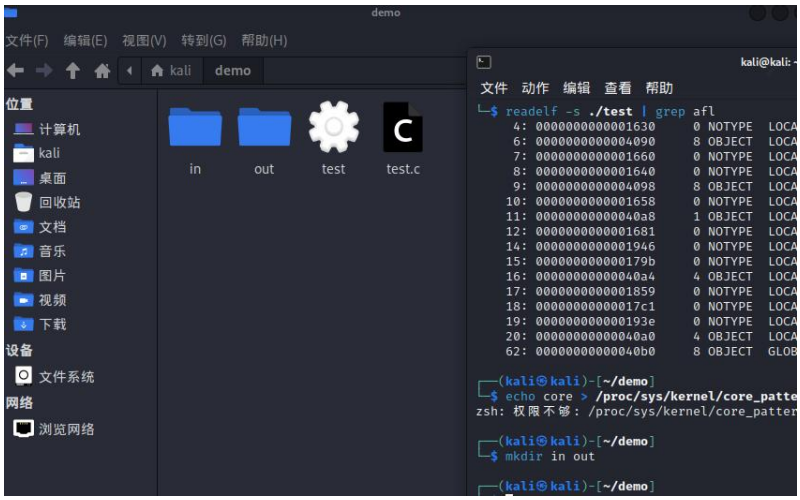
编译后会有插桩符号，使用如下命令验证：readelf -s ./test | grep afl，结果：



2. 创建测试用例

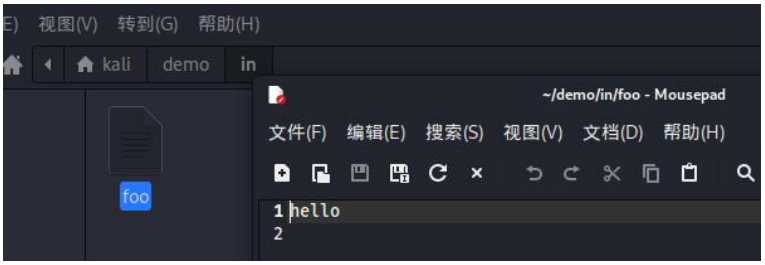
首先，创建两个文件夹 in 和 out，分别存储模糊测试所需的输入和输出相关的内容。

命令：mkdir in out，结果如下图：



然后，在输入文件夹中创建一个包含字符串“hello”的文件。

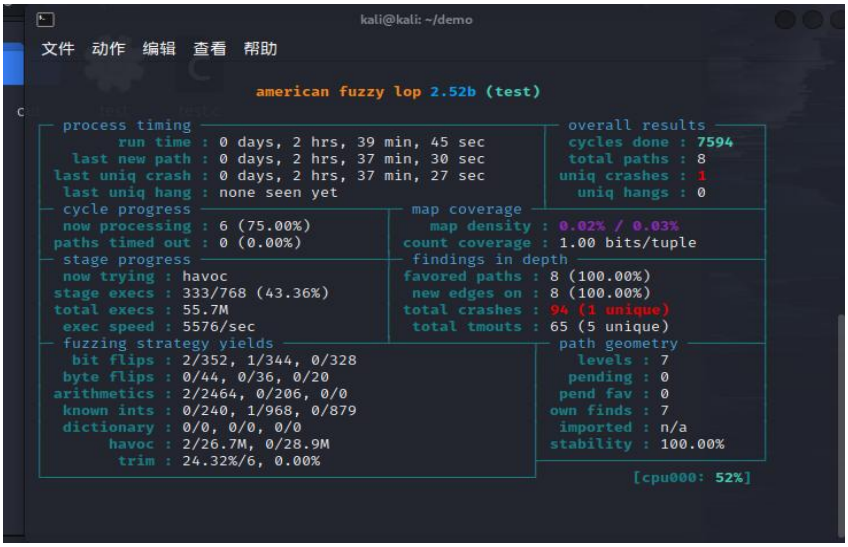
命令：echo hello> in/foo，结果如下图：



3. 启动模糊测试

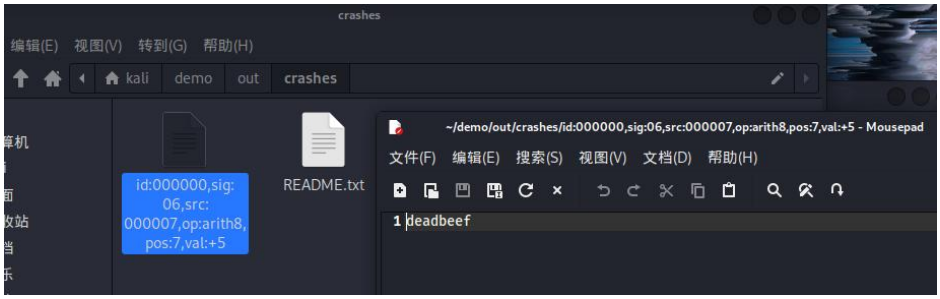
运行如下命令，开始启动模糊测试：

命令：afl-fuzz -i in -o out -- ./test @@，运行一段时间后，结果如下图：



4. 分析 crash

在 out 文件夹下的 crashes 子文件夹里面是我们产生 crash 的样例，如下图所示：



可见，这正是使我们代码产生崩溃的输入。

心得体会：

通过这次实验，我深刻认识到 AFL 是一款非常强大的模糊测试工具。相比于传统的手动测试，AFL 可以自动化地处理大量的测试用例，并通过覆盖率的记录和调整，增加发现漏洞的概率。同时，AFL 的安装和使用也非常简便，不需要复杂的配置和编程知识。这使得 AFL 在安全领域广受欢迎。总之，这次实验让我对 AFL 有了更深刻的理解和认识，也让我更加热爱信息安全这个领域。