《AFL模糊测试》实验报告

姓名：梁婧涵 学号：2112155 班级：1120

实验名称：AFL模糊测试

实验要求：

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

实验过程：

1. AFL的安装：1）按照默认提供的安装方法进行安装，安装完成后 afl-gcc 和 afl-fuzz 均可正常使用。

git clone https://github.com/google/AFL.git

cd AFL

make # 编译 AFL

sudo make install # 在系统中安装 AFL



1. 进入 qemu\_mode 目录，使用build\_qemu\_support.sh 脚本构建FL-Qemu。

安装成功：

[+] Build process successful!

[\*] Copying binary...

-rwxrwxr-x 1 hxn hxn 16787120 5月 24 00:09 ../afl-qemu-trace

[+] Successfully created '../afl-qemu-trace'.

[\*] Testing the build...

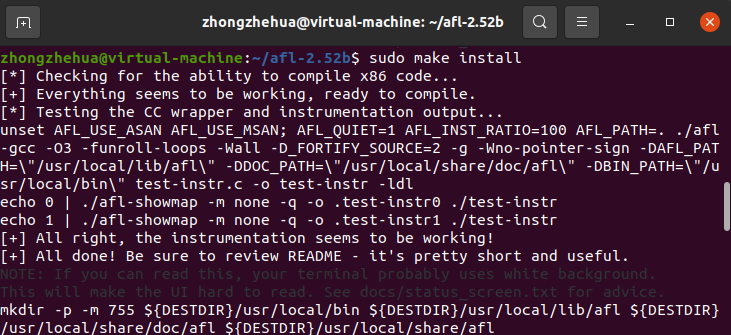
[+] Instrumentation tests passed.

[+] All set, you can now use the -Q mode in afl-fuzz!

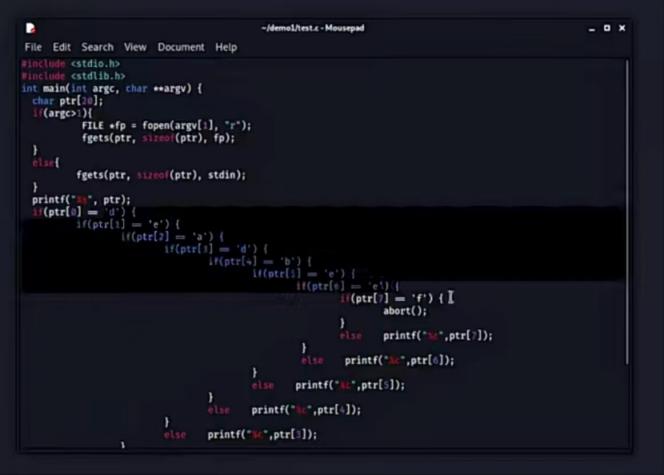
1. 下载目标程序集

wget https://ftp.gnu.org/gnu/coreutils/coreutils-9.1.tar.gz

tar -zxvf coreutils-9.1.tar.gz



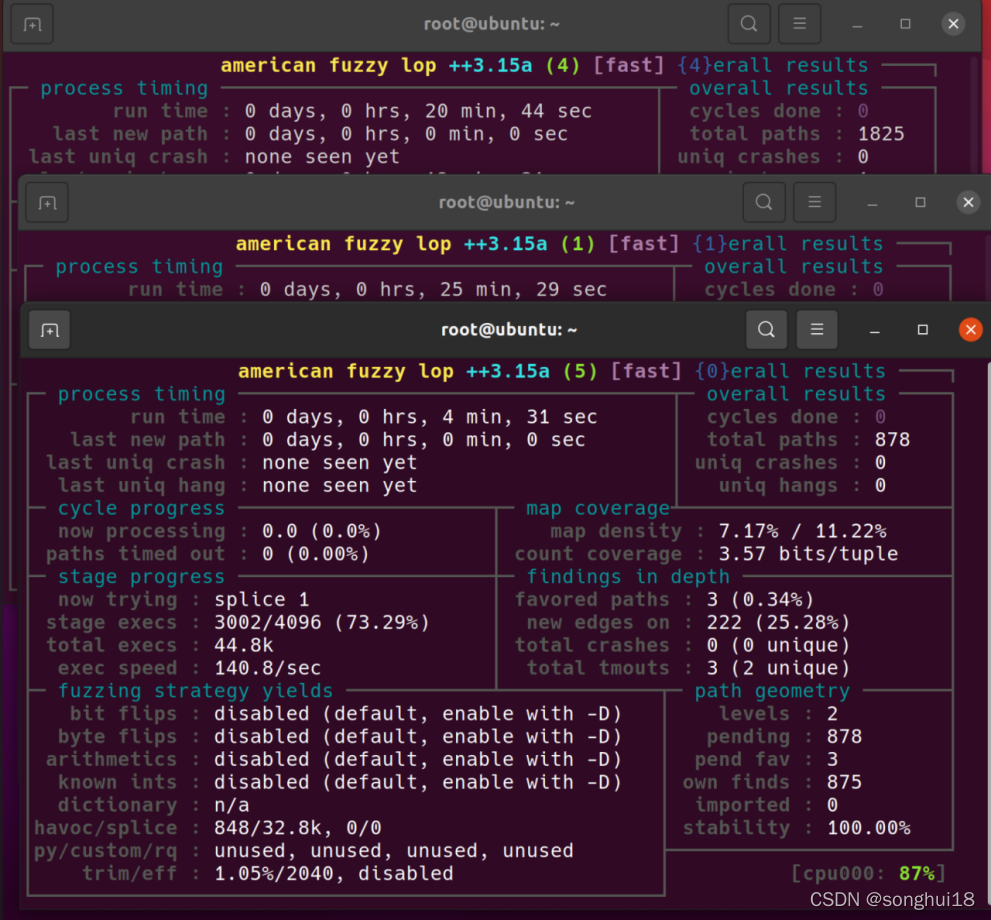
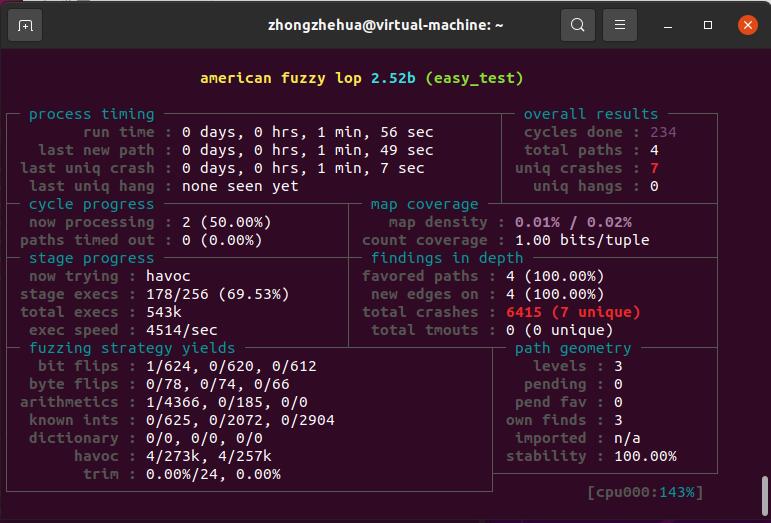
1. AFL的应用：
2. 插桩编译：用c语言编写一个测试用例，设置一些BUG和漏洞



对该源文件（easy\_test.c）进行插桩编译



1. fuzz：从stdin读取输入的目标程序，给程序设置testcase\_dir和findings\_dir两个文件夹，以及在testcase\_dir中存放一些测试用例testcase。



3.覆盖引导和文件变异的概念和定义：

覆盖引导和文件变异是软件测试中常用的两种技术，用于提高测试用例的质量和覆盖率。

覆盖引导是一种动态测试技术，它通过在程序执行时对代码进行覆盖率分析，自动生成具有高覆盖率的测试用例。具体来说，覆盖引导将程序执行分成多个步骤，并对每个步骤记录代码的覆盖情况，然后根据这些记录生成新的测试用例。这些新的测试用例通常能够覆盖程序中的更多代码分支和路径，从而提高测试的覆盖率和效果。

文件变异是一种静态测试技术，它通过对程序代码的副本进行人为的修改（即变异），生成具有不同行为的代码版本，然后使用这些代码版本进行测试。具体来说，文件变异将程序代码进行变异，比如删除、修改或添加一些语句或条件，从而生成不同的代码版本。然后，测试人员可以使用这些不同的代码版本进行测试，以检查测试用例的效果和发现程序中可能存在的缺陷。文件变异通常能够提高测试用例的质量和发现率，同时也能够评估测试套件的有效性和健壮性。

心得体会：

AFL（American Fuzzy Lop）通过记录输入样本的代码覆盖率，不断对输入进行变异，从而达到更高的代码覆盖率。AFL 采用新型的编译时插桩和遗传算法自动发现新的测试用例，这些用例会触发目标二进制文件中的新内部状态，改善了模糊测试的代码覆盖范围。

从源码编译程序时进行插桩，以记录代码覆盖率（Code Coverage）；

选择一些输入文件，作为初始测试集加入输入队列（queue）；

将队列中的文件按一定的策略进行变异；

如果经过变异文件更新了覆盖范围，则将其保留添加到队列中；

上述过程会一直循环进行，期间触发了crash的文件会被记录下来。它封装了一个GCC/CLang编译器，用于对被测代码重新编译的过程中进行插桩。插桩完毕后，AFL fuzz就可以给其编译过的代码输入不同的参数参数，跟踪被测代码的执行路径，并判定对输入的变异能否触发新的已知或未知执行路径。

流程如下：先是用afl-gcc编译源代码afl\_test.c，然后以testcase.txt为输入，启动afl-fuzz程序，将testcase作为程序的输入执行程序，afl会在这个testcase的基础上进行自动变异输入，使得程序产生crash。