

《漏洞利用及渗透测试基础》实验报告

姓名：蒋薇 学号：2110957 班级：计科1班

实验名称：

AFL 模糊实验实验

实验要求：

复现 AFL 在 KALI 下的安装、应用，理解覆盖引导和文件变异的概念和含义。

实验过程：

1. 复现 AFL 在 KALI 下的安装、应用

创建文件夹 AFL, cmd 进入控制台，

安装命令：

```
sudo apt-get install afl
```

ls /usr/bin/afl, 查看安装后的可执行文件, afl-gcc, afl-fuzz...

```
udeng@hudeng-ThinkPad-X250:~/下载/AFL-2.57b$ afl-fuzz
afl-fuzz 2.57b by <lcantuf@google.com>

afl-fuzz [ options ] -- /path/to/fuzzed_app [ ... ]

required parameters:
  -i dir      - input directory with test cases
  -o dir      - output directory for fuzzer findings

execution control settings:
  -f file      - location read by the fuzzed program (stdin)
  -t msec     - timeout for each run (auto-scaled, 50-1000 ms)
  -m megs     - memory limit for child process (50 MB)
  -Q          - use binary-only instrumentation (QEMU mode)

fuzzing behavior settings:
```

2.

```
afl-gcc -o test test.c
```

生成目标 test 可执行文件

3. 插桩

```
readelf -s ./test | grep afl
```

4. 创建输入输出文件夹

输出报错信息到指定文件夹：

```
echo core> /proc/sys/kernel/core pattern
```

输入输出文件夹：

```
mkdir in out
```

种子文件：

```
afl-fuzz -i in -o out - ./test @@
```

开始测试：文件

```
echo hello> in/foo
```

findings in depth :test/crashes, 得到 crash

2. 理解覆盖引导和文件变异的概念和含义

覆盖引导, 即 通过向目标程序插桩, 统计代码覆盖, 反馈给模糊测试引擎 (fuzzer, 即模糊测试工具), 反馈信息用于变异种子, 生成更高质量的输入, 使得 fuzzer 能够用更好的输入让被测程序达到更高的代码覆盖率。

bitflip、arithmetic、interest、dictionary 是 deterministic fuzzing 过程, 属于 dumb mode(-d) 和主 fuzzer(-M) 会进行的操作; havoc、splice 与前面不同是存在随机性, 是所有 fuzz 都会进行的变异操作。文件变异是具有启发性判断的, 应注意“避免浪费, 减少消耗”的原则, 即之前变异应该尽可能产生更大的效果, 比如 eff_map 数组的设计; 同时减少不必要的资源消耗, 变异可能没啥好效果的话要及时止损。

(此处根据实际操作过程, 留下具体操作步骤、附加一些自己的理解, 即可)

心得体会:

AFL (American Fuzzy Lop) 是一款基于覆盖引导 (Coverage-guided) 的模糊测试工具, 它通过记录输入样本的代码覆盖率, 从而调整输入样本以提高覆盖率, 增加发现漏洞的概率。

- ①从源码编译程序时进行插桩, 以记录代码覆盖率 (Code Coverage);
- ②选择一些输入文件, 作为初始测试集加入输入队列 (queue);
- ③将队列中的文件按一定的策略进行“突变”;
- ④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;
- ⑤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。