架构

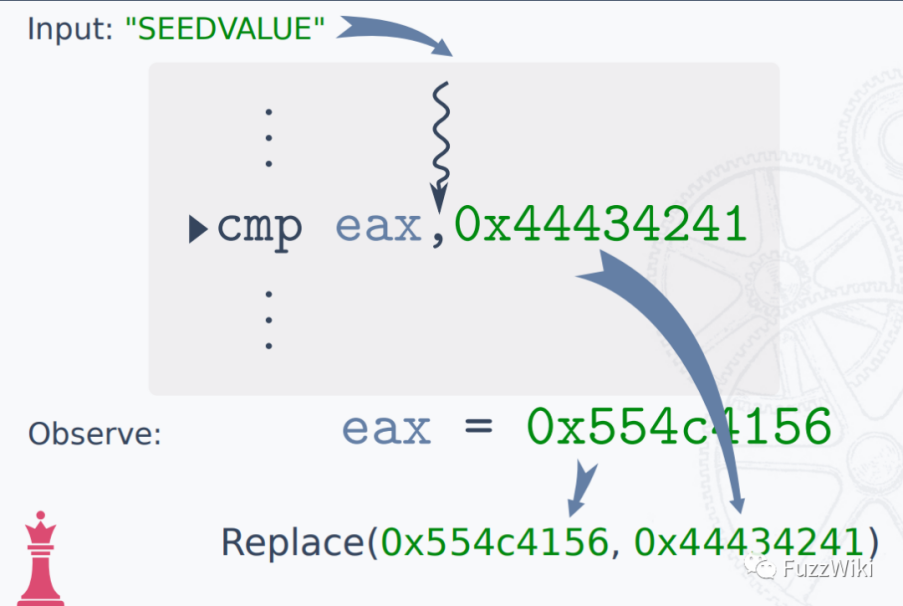
大体架构同afl++。

与afl的不同在于，封装更多，把很多afl的全局变量封装到一个结构体里。把afl的afl-fuzz.c拆分成了多个文件。

主要源代码在src、include、instrumentation文件夹里。其他文件夹的内容可以不用管。

解决的问题

覆盖率方面使用redqueen机制来进入check分支，获得更高的覆盖率。使用cmplog插桩记录分支条件的比较情况，从而生成一些变异规则用于变异。



为了进一步提高变异效率，在变异时采用truzz的方式，通过bitflip识别关键字节，如果bitflip路径变化导致路径显着缩短（具体表现为边数的减小），则该转换可能与验证检查有关，因此对这些字节变异大概率会导致进程提前退出，从而降低fuzz效率。所以在变异到关键字节时，以小概率保留变异，大概率跳过变异。这样既可以提高有效执行次数，也不会漏掉一些类似checksum导致的漏洞。

引导机制方面使用ngram的机制，即局部路径的方式来进行引导，可以探索更细粒度的覆盖率。局部路径是指相邻的2条边组成的路径。采用afl的边覆盖率类似的方式，做2条边的异或，作为局部路径，并使用局部路径来替代原有的边，更新机制和afl边覆盖类似。

调度策略采用truzz的方式。Fuzz过程中动态监控新边数（每一轮fuzz\_one，做一次），如果在种子队列里面发现有走过更多新边的种子，则优先做fuzz，新边数越多，越优先。这种策略认为出现大量新边表示走到了一个复杂的功能点，所以可以更快的探索更有价值的种子。

对比结果

[Feishu Sheet](https://anban.feishu.cn/sheets/shtcnTQra8dJEfeL7giNLdqE5Gf)

编译：

环境：LLVM-13

wget https://apt.llvm.org/llvm.sh

chmod +x llvm.sh

./llvm.sh 13

Make

使用方法：

1. 引入环境变量export AFL\_LLVM\_CMPLOG=1，通过afl-clang-fast编译出redqueen插桩的二进制文件。打开这个环境变量必须用LLVM编译
2. 引入环境变量export AFL\_LLVM\_INSTRUMENT=NGRAM-2，通过afl-clang-fast编译出ngram插桩的二进制文件。
3. 开始fuzz：afl-fuzz -D不跳过确定性变异阶段 -i 输入文件夹 -c redqueen插桩的二进制文件 -o输出文件夹 ngram插桩的二进制文件 fuzz参数。

如

afl-fuzz -D -i fuzzer\_input -c ./lava-install-cmp/bin/uniq -o output\_cmp\_2\_gram0602 -- ./lava-install-ngram-0602/bin/uniq @@

修改的code

改动的地方都有my change的标记。

afl-fuzz.h 加入了truzz相关的全局变量

Afl-fuzz-one.c 加入了一个fuzz\_one\_truzz函数

Afl-fuzz.c加入了truzz相关的数据初始化、工具函数以及fuzz循环时的队列的选择策略

不足，可以规避的缺点，改进的方向。

可以加入一个环境变量控制Afl-fuzz-one.c 中的truzz\_sig变量，从而控制是否开启truzz模块。Afl-fuzz.c中的种子队列选择也可以通过环境变量来做控制。

由于确定性变异阶段往往效果不好，而truzz的算法依赖于确定性变异的bitflip 8/8阶段，因此可以做一个选项，在确定性变异阶段只做bitflip 8/8的变异来完成truzz的功能。这样可以提高效率。