# 《软件安全技术》综合实验

## 实验二：使用American Fuzzy Lop工具

## 作者：谭九鼎

## 学号：522024320137

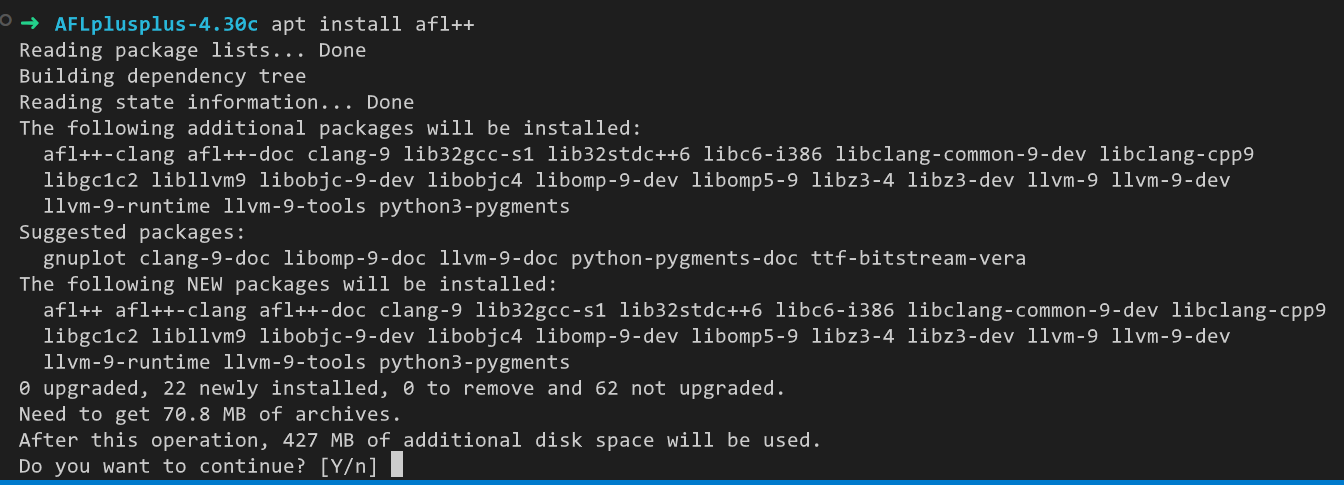
## 时间：2024年12月

官网：<https://lcamtuf.coredump.cx/afl/>

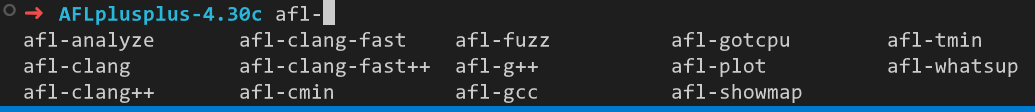
里面提到有几年没更新了，可以用社区fork：<https://aflplus.plus/> <https://github.com/AFLplusplus/AFLplusplus>

二者均没有官方发布二进制，只有从源码编译的教程。

但debian仓库维护了预编译的包可以使用：sudo apt update && sudo apt install afl++



安装完成后获得的二进制：



也可以用docker：<https://hub.docker.com/r/aflplusplus/aflplusplus>

从源码编译的尝试：

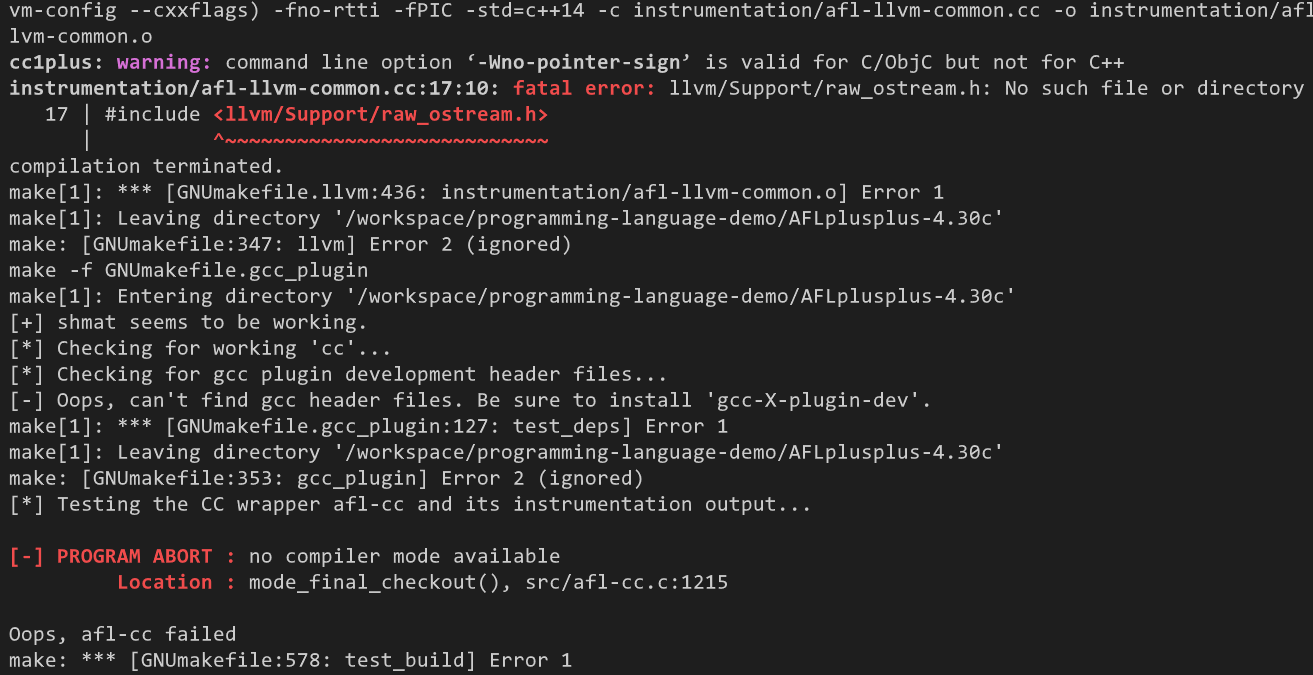
下载：https://github.com/AFLplusplus/AFLplusplus/archive/refs/tags/v4.30c.tar.gz

解压：tar xf AFLplusplus-4.30c.tar.gz

编译：

cd AFLplusplus-4.30c

STATIC=1 make all



错误挺多的，难以解决，不继续探究。

教程：

<https://lcamtuf.coredump.cx/afl/QuickStartGuide.txt>

<https://github.com/alex-maleno/Fuzzing-Module>

<https://github.com/antonio-morales/Fuzzing101>

选取第二个仓库的exercise1测试，是一个单文件的simple\_crash.cpp，稍微修改了一下：

#include <iostream>

using namespace std;

int main() {

   string str;

   cout << "enter input string: ";

   getline(cin, str);

   cout << str << endl << str [0] << endl;

    if(str[0] == 0 || str[str.length() - 1] == 0)

        abort();

    int count = 0;

    char prev\_num = 'x';

    while (count != str.length() - 1) {

        char c = str[count];

        if(c >= 48 && c <= 57) {

            if(c == prev\_num + 1) {

                abort();

            }

            prev\_num = c;

        }

        count++;

    }

}

使用AI解读：

1. 通过getline从用户输入读取一整行文本并存储到字符串str中

2. 输出完整的输入字符串以及输入字符串的第一个字符

3. 边界条件检查：检查字符串首字符 `str[0]` 或尾字符 `str[str.length() - 1]` 是否为0

- 潜在问题：

- `str[0]` 是字符串的第一个字符，而不是 ASCII 值 `0`。所以此检查判断的是字符 `'0'` 是否存在，这可能是作者的失误

- 如果用户输入是空字符串（`str.length() == 0`），访问 `str[0]` 或 `str[str.length() - 1]` 会导致未定义行为（UB）

4. 循环检查字符串内容

- 如果当前字符 `c` 是数字（ASCII 值在 `48` 到 `57`，即 `'0'` 到 `'9'`），会进一步检查

- 若 `c` 是前一个数字字符的递增值（`prev\_num + 1`），直接调用 `abort()`

- 否则将 `c` 更新为 `prev\_num`

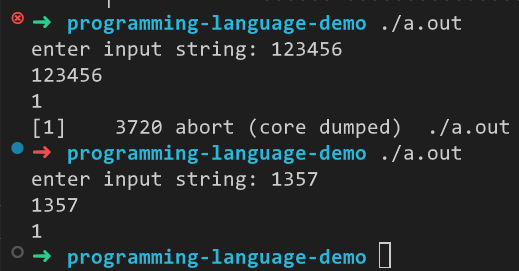
- 潜在问题：

- 如果输入字符串是空字符串，`str.length() - 1` 会产生未定义行为

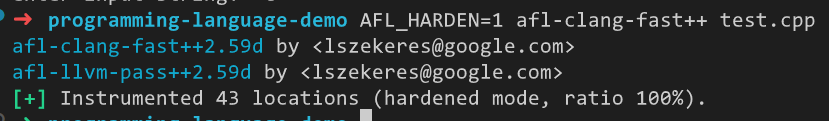
- 当字符串长度为 1 时，`count != str.length() - 1` 条件可能错误地导致漏检

用g++手动编译运行测试：

输入不连续字符时不会崩溃，输入顺序字符或不输入时会abort



使用afl编译：AFL\_HARDEN=1 afl-clang-fast++ test.cpp



创建种子：

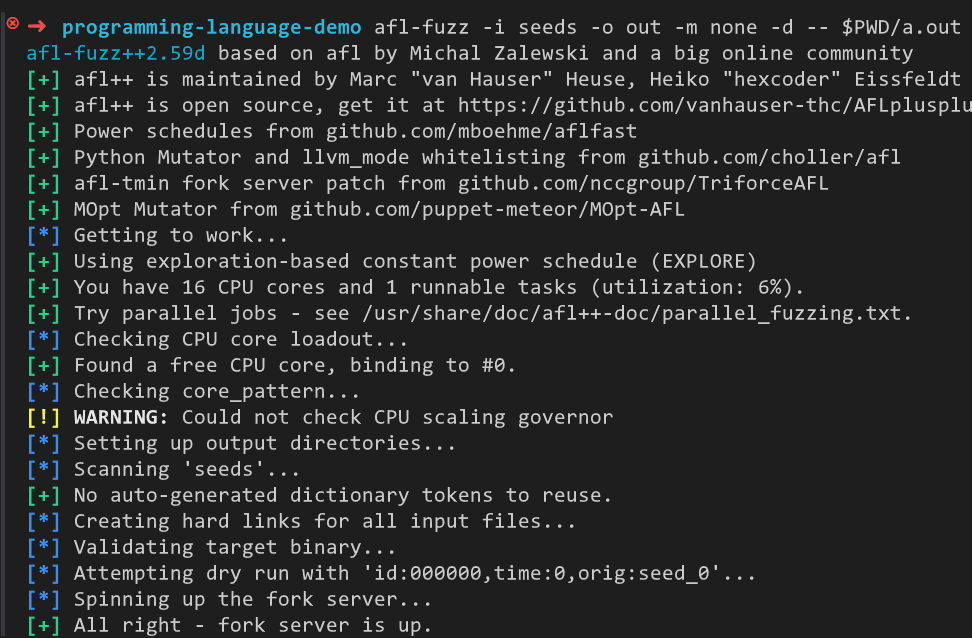
mkdir seeds && cd seeds

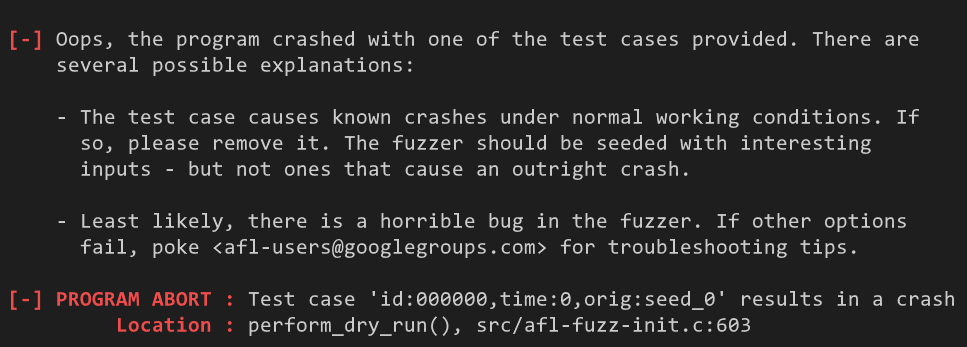
for i in {0..4}; do dd if=/dev/urandom of=seed\_$i bs=64 count=10; done

cd ..

运行测试：

afl-fuzz -i seeds -o out -m none -d -- $PWD/a.out





根据教程，“一旦 AFL++ 的 UI 中显示至少 1 个崩溃，请按 Ctrl + C 退出 AFL++。您可以通过遍历 out/default/crashes 目录来找到导致程序崩溃的输入”。但实际上此程序直接退出了，而且out下也没有default目录，crashes里也为空。

