AFLplusplusnet <https://github.com/zyingp/AFLplusplusnet>的原理是通过网络真实发送种子到待测试服务器程序，同时为了优化速度，没有每次都新起一个服务器实例，不仅如此，还复用原来的网络连接减少连接开销，原因是其实服务器本来就可以同时支持多个连接且一个连接中常常可以支持收发多个包。AFLplusplusnet为了防止一段时间服务器连接出现问题（比如mosquitto可能一个连接上会收到一个长度字段很大的包，如字段位100多M，这样它就会一直不停地在该连接上读包内容希望读完那100M字节），可以设置处理一定input后重连。但是还没有做处理一定input后服务器也重启。

可以查看commit看我的修改历史<https://github.com/zyingp/AFLplusplusnet/commits/main> 看看我是怎么改的。

Visualstudio文件夹下有aflplusplus.sln 可以直接调试（安装好visual studio2019以上 和wsl1以上）

1. 在AFLplusplusnet下

采用make source-only编译（有nyx-mode错误可忽略）

2. 准备好用AFLplusplusnet或者AFLplusplus4.00c插桩好的网络程序如mosquitto或者dnsmasq，可以同时采用AddressSantizer插桩以利于发现bug，准备好相应的种子文件。

3. 进行fuzz：

/media/iotteam/hd/zyp/AFLplusplusnet/afl-fuzz -P 0:10001 -i testcase\_dir\_mqtt -o finding\_dir -- ./mosquitto -p 10001

其中-P 的第一个参数是表示协议（0表示TCP协议， 1表示是UDP协议），第二个参数是端口号，这里是10001，其它参数与一般fuzz一样，如testcase\_dir\_dns则是种子文件夹。

一些文字和图案

中度可信度描述已自动生成

这里可以同时用gdb监视被测试的程序，以防止出现崩溃但是不好复现。

在另外一个窗口采用命令ps -ef | grep mosq

绿色的钟表

描述已自动生成

会出现两个mosquito，找到那个父pid是另一个的，那个就是被测试的实例，如这里是1768889. 执行

sudo gdb -ex "attach 1768889"

同时在gdb中记得输入c （即continue）让程序继续跑，因为刚刚attach会暂停程序：

图形用户界面, 文本

描述已自动生成

另外可以通过-H调节参数

/media/iotteam/hd/zyp/AFLplusplusnet/afl-fuzz -P 0:10001 -H 1000:1000:10:1:1000 -i testcase\_dir\_mqtt -o finding\_dir -- ./mosquitto -p 10001

-H 后面的5个参数分别表示：

u32 reconnect\_num = 1000; // 多少input重连一次

u32 socket\_timeout\_usecs = 1000; // 连接的等待时间 in us

u32 wait\_after\_send\_one\_usecs = 10; // 发送完一个种子中一块内容后的等待时间in us

u32 wait\_after\_sendusecs = 1; // 发送完一个种子所有内容后的等待时间in us

u32 new\_start\_server\_waitusecs = 1000; // 重启服务器后的等待时间 in us

测试dnsmasq也是类似的：

/media/iotteam/hd/zyp/AFLplusplusnet/afl-fuzz -P 1:10001 -i testcase\_dir\_dns -o finding\_dir -- ./dnsmasq -p 10001 -d

其中-P参数的1表示是UDP协议，10001表示端口号，testcase\_dir\_dns则是种子文件夹。

一些文字和图案

中度可信度描述已自动生成

曾英佩

2023-5-22