Coverage-based Greybox Fuzzing as Markov Chain 阅读报告

1. 背景问题

大多数覆盖率引导的模糊测试器会率先使用相同的少数“高频”路径，并通过向低频路径靠拢来开发策略，在相同的测试次数下探索更多的路径。比较典型的就是AFL模糊测试器。但AFL中存在以下几个问题：

1. AFL的种子搜索策略效率不高

AFL采用的方法就是将能够触发漏洞或提高代码覆盖率的种子作为下一轮种子变异的基础，这些种子通过循环不断被筛选进入下一轮。但是“好的种子”可能有多个，在这个迭代的过程中，种子的选择顺序对整体模糊测试效果的影响也很大。

1. AFL的分配策略不合理

AFL的power schedule总是给予更多的资源（更高的power schedule）。给初始化种子大量的能量的确能够帮助发现更多近邻，这些近邻也会执行一些低频路径。但是执行一段时间后，变异成了更多种子，那么这些同来源的种子就会集中执行部分路径，这部分路径也就成为高频路径，此时的低频路径就难以被覆盖到。

总体而言，许多模糊测试会执行高频路径，导致大多数的测试路径是相同的，因此AFL-FAST提出了使AFL偏向低频路径的策略，即给予低频路径更多机会，使fuzz更加公平，以便在相同的模糊测试量下探索更多的路径。

1. 马尔科夫链

马尔科夫链的核心思想就是下一状态的概率分布只能由当前状态决定，在时间序列中它前面的事件均与之无关。

1. AFLFast中针对AFL的改进

s(i)表示种子之前从队列T中选到的次数。

f(i)表示执行状态i的生成输入的数量。

能量函数p(i)是关于s(i)和f(i)的函数。

f(i)/n是最大似然估计。

1. 搜索策略

本文认为一个好的基于灰盒测试的fuzzer应该给low-frequency paths高优先级。​

策略有两个标准：

s(i)小的优先级高：s(i)表示种子之前从队列中选到的次数。

f(i)小的优先级高：f(i)表示执行状态为i的生成输入的能量。

1. 能量分配策略

本文提出的能量分配策略是首先赋值低能量，然后根据选择的次数增加能量。同时，能量分配策略赋予状态的能量与状态固有分布的密度成反比。若状态i存在于low-density region，fuzzer可以分配更多能量来找这个区域中状态i的neighborhood。

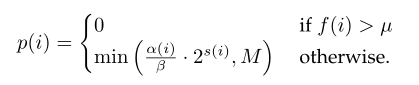
* 1. exploitation-based constant schedule(EXPLOIT)：

其中是算法中assignEnergy的实现。AFL根据执行时间，块转换覆盖率，种子的构造时间来计算。

* 1. exploration-based constant schedule(EXPLORE)：和EXPLOIT差不多：

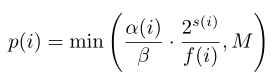
，其中。

* 1. Cut-Off Exponential(COE)：



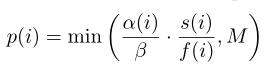
其中保持模糊测试的原始判断，是个常数，值较低,是执行已发现路径的模糊平均数。，其中是发现的路径集合。也就是说，为高频路径，即使从模糊其他种子接收大量模糊，也被视为低优先级，直到它们再次低于平均值才模糊。常数M提供了每次模糊迭代生成的输入数量的上限。

* 1. Exponential sche dule(FAST)：

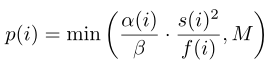


输入队列中选择的输入次数越多，执行路径i就越少，所以放在指数上，选的越多，就赋予低频路径越多的能量值，让其多变异。

* 1. Linear schedule(LINEAR)：



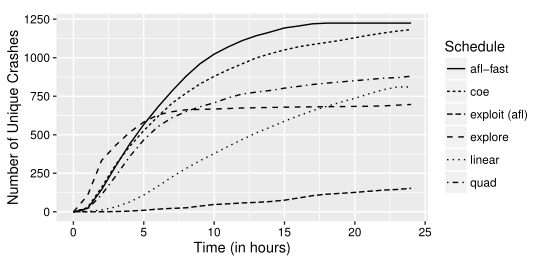
* 1. Quadratic schedule(QUAD)：



对于每个种子输入fuzz的次数，在AFL原有计算结果的基础上，再乘以一个以指数增长的系数，然后设定一个最大值M，如果超过了这个最大值，进行截断。

通过上述的策略，AFLFast智能地控制种子fuzz次数，从而使队列调度给予低频路径更多机会，使fuzz更加公平，转向可能隐藏漏洞的路径。

1. 实验评估
2. 能量分配策略的比较

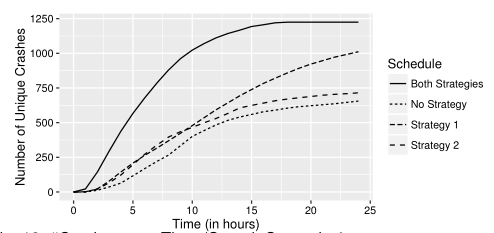


AFL-Fast实现的能量分配策略表现好于其他。

cut-off exponential schedule (coe)效果略差于AFL-Fast。

24小时的实验后，AFL-Fast的能量分配策略发现的unique crashes比其他三种（linear，quad，explore）多50%。

1. 搜索策略的比较



两种策略的结合比各自使用策略效果更好。前12小时其他策略表现相似，24小时后策略1（改变了选favorite的策略）比策略2更有效。策略2（改变了从队列中选取测试输入的顺序）似乎效果不明显。用这个策略和不使用这个策略的AFLFAST效果相似。总之，AFLFAST集成两个策略后找到的unique crashes是原生AFLFAST或集成策略1找到unique crashes数量的2倍。

1. 总结

AFL-Fast是2016年的工作，现在看来是比较旧的工作了。但是其中提出的根据路径执行频率实行能量分配调度问题在一定程度上提高了模糊测试的速度。不过，AFL-Fast不足之处也在于提高了速度，但没有表明其测试的效果也能得到提升。