VUzzer - Application-aware Evolutionary Fuzzing阅读报告

1、研究背景

现有的模糊测试工具大多是可扩展的。但是鉴于大多数软件的复杂性和程序大小，大多数可扩展的模糊器存在的问题是其检测到的是表层易于检测的漏洞，对于深层次的、复杂的漏洞，很少有模糊测试工具能够做到高效检测。例如AFL等的应用不可感知的模糊测试工具，它们难以发现那些藏在深处难以发掘的漏洞。而结合AFL和符号执行的自动漏洞挖掘技术Driller虽然能够探索更深层次的路径，但是基于符号执行消耗大量运行空间的方式使其难以扩展，且现实情况中程序的复杂性足以将符号执行引入路径爆炸的问题。

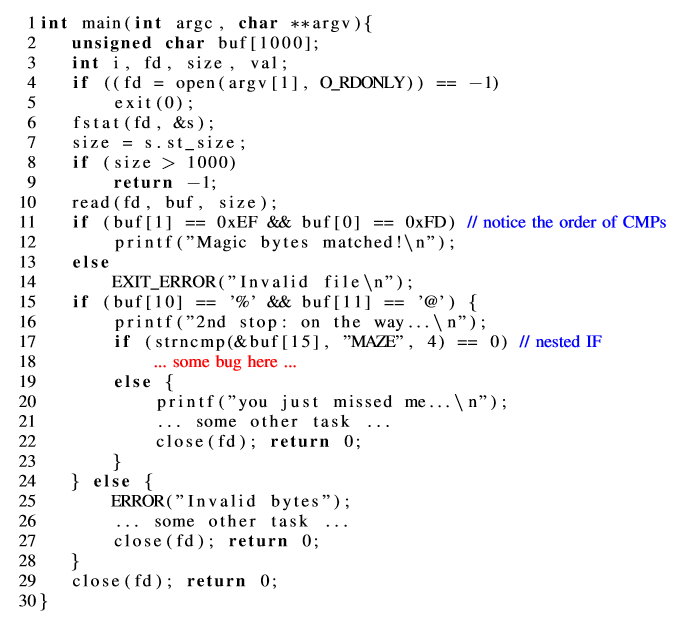
VUzzer使用应用感知的“智能”变异模糊测试的策略，不需要任何关于应用程序或输入格式的先验知识，利用基于静态和动态分析的控制和数据流特性来推断程序的基本属性，通过结果反馈和优化输入的生成过程以产生更少但是更优秀的测试输入。其具有可扩展性，有效加快挖掘速度，增加挖掘深度。

1. 关键点

应用感知、静态分析、动态污点分析

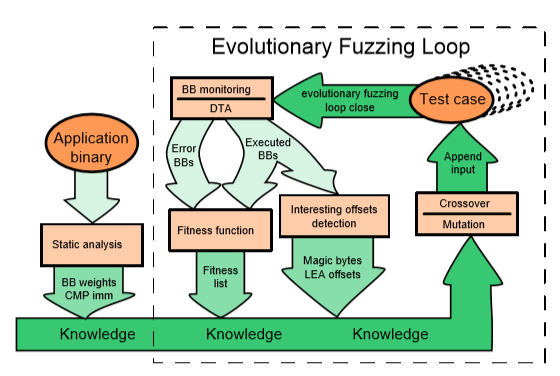
1. VUzzer解决的问题

VUzzer主要解决现有模糊器存在的三个问题：



1. magic bytes的比较问题：AFL采用的是盲变异的策略，必须完全猜中FDEF，才能进入深层次的代码；AFL必须正确找到buf[4]和buf[5]的偏移位置来进行变异；AFL是基于代码覆盖率的模糊测试工具，如果没有正确进入到if分支，而是进入else分支，那么对于AFL而言是发现了新的路径，那么AFL则会在else分支中持续探索，而不会在意if分支的内容，有可能浪费大量时间和运算资源。
2. 深层漏洞的触发问题：AFL一旦没能完全猜中FDEF，会优先进入else分支，生成遍历else分支的输入。因此，任何隐藏在嵌套的if中的漏洞，AFL都将难以发现。
3. 错误处理代码的解决：如果采用结合AFL和符号执行技术的自动漏洞挖掘技术Driller来解决AFL的上述问题，Driller将通过依次否定路径条件来寻找新的路径，那么就有可能否定第11行中的条件，这可能会导致模糊测试工具去探索一些错误处理代码。

4、VUzzer实现方法



1）静态分析阶段：

①找出基本块，计算块权重并将其存入LBB（错误处理代码块权重为负）【解决错误处理代码问题】

②扫描代码中的cmp指令相关的立即数，存入Limm中

2）动态分析阶段：（每循环执行一次为一代。）

①对种子文件（测试者提供的基础输入）进行DTA（动态污点分析），捕获输入中的共同特征，找到magic bytes的位置和错误处理块，以免新的测试输入陷于错误处理块中。

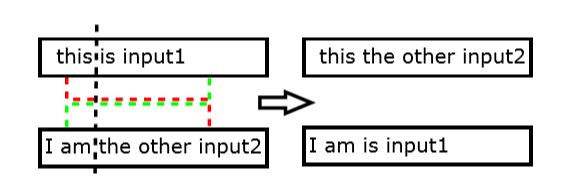
②输入执行，生成执行块的跟踪。如果发现以前未出现的基本块，则对新的基本块输入用例进行DTA分析，追踪输入的值在程序中运行的状态，监视数据流的特征，推断输入的结构。

③适应度计算。为执行权重较高基本块的输入提供高分，同时对相应路径进行优先排序。【解决深层漏洞的触发问题】

④遗传算子和新的输入生成。

3）突变阶段：【解决magic bytes的比较问题】

通过交叉和变异两种策略来实现，



①随机修改非magic bytes的位置（在动态污点分析阶段确定其位置）

②对于确定类型为数字的位置将其修改为特殊数字，例如 0, MAX\_UINT、负数等等。

③为所有受污染的cmp父母输入指令,如果op1≠op2的值，用步骤2 中的op2（即magic bytes的真实值）替换字符串中的污染值或者有一个固定的概率的值取代污染字节的随机序列。

1. VUzzer实现细节

静态分析：VUzzer实现了IDA中的静态分析(常数串提取和基本块重计算)。分析是使用Python在IDAPython上编写的。

动态分析：VUzzer在Pin动态分析框架的基础上实现了两种动态分析(基本块跟踪和DTA)。对于基本块跟踪，实现了一个指针工具（pin tool）来记录在执行过程中遇到的每个基本块及其频率，指针可以根据需要有选择地跟踪由某些库执行的基本块。DTA实现是基于由Stamatogiannakis等人提出的DataTracker，而DataTracker又是基于LibDFT。由于LibDFT只能处理32位应用程序，所以当前的VUzzer原型只能用于模糊32位应用程序。不过，支持64位的VUzzer正在实现。

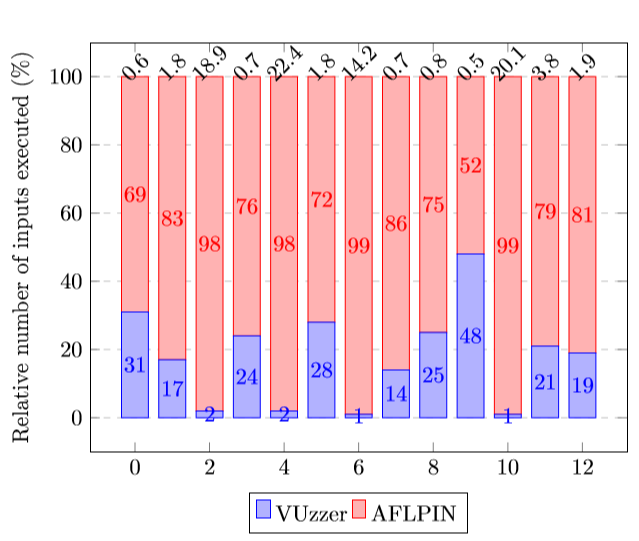
1. 实验评估

分别在三组数据集上进行VUzzer性能的评估，主要评估的是与现有的一些模糊测试工具相比（eg：AFL），VUzzer如何用更少的输入来找到隐藏在更深层的漏洞，而不是像AFL那样去产生更快的输入执行。

实验环境：配备32位的2核Intel CPU和4G内存的Ubuntu 14.04 LTS系统上进行了实验，该系统。

1. DARPA CGC Dataset

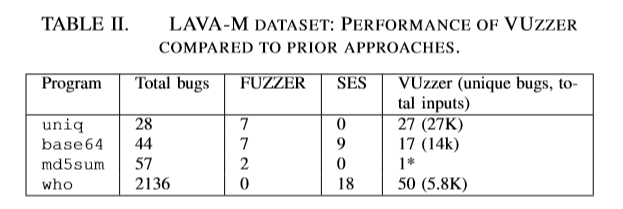
对于DARPA CGC数据集，所提供的环境是一个VM，它有一个名为DECREE的自定义操作系统。



每个CGC二进制文件执行的输入的相对数量，其中VUzzer和AFLPIN都发现崩溃时，VUzzer的执行输入更少，证明了VUzzer能够用更少的输入来发现更多的漏洞。

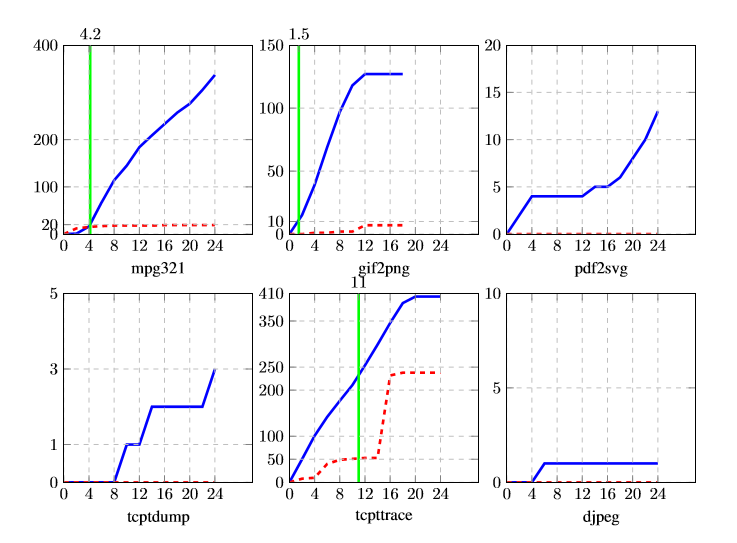
1. LAVA Dataset

对LAVA测试集进行了测试，在相同时间内，VUzzer能比其他fuzzer找到更多的崩溃。



1. Various Applications (VA) Dataset

在其他真实应用程序的测试中，VUzzer同样显示出更好的工作效率。（蓝色实线为VUzzer，红色虚线为AFL）



1. 总结

VUzzer通过静态分析和动态污点分析技术增强了fuzzer的效率，解决了magic bytes比较、深层漏洞挖掘、错误代码处理等问题，但动态污点分析技术还是在一定程度上影响了整体工作效率。