中国科学院软件研究所学术年会’2019
暨计算机科学国家重点实验室开放周

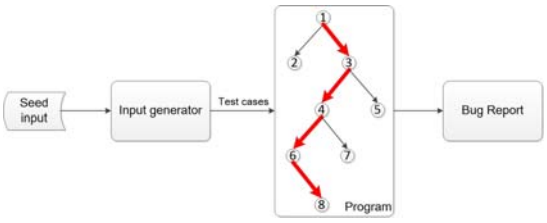
学术论文

Fuzzing Program Logic Deeply Hidden in Binary Program Stages

Yanhao Wang, Chua Zheng Leong, Yuwei Liu, **Purui Su** (purui@iscas.ac.cn), Zhenkai Liang

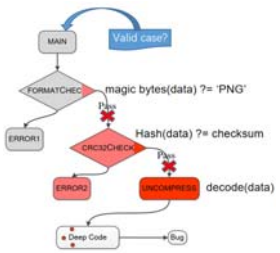
SANER 2019 : IEEE International Conference on Software Analysis, Evolution and Reengineering

模糊测试技术



- 模糊测试通过随机构造测试用例测试程序行为逻辑
- 其是漏洞挖掘实战中的利器，方案简单且效率高

模糊测试缺陷



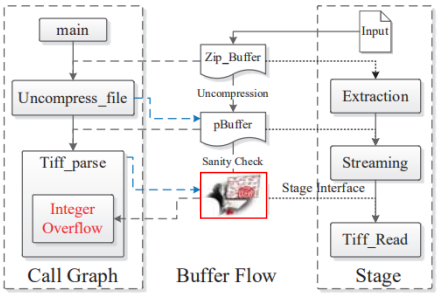
- 测试用例构造随机，无法有效绕过浅层处理逻辑测试程序深层代码

当前解决方案缺陷

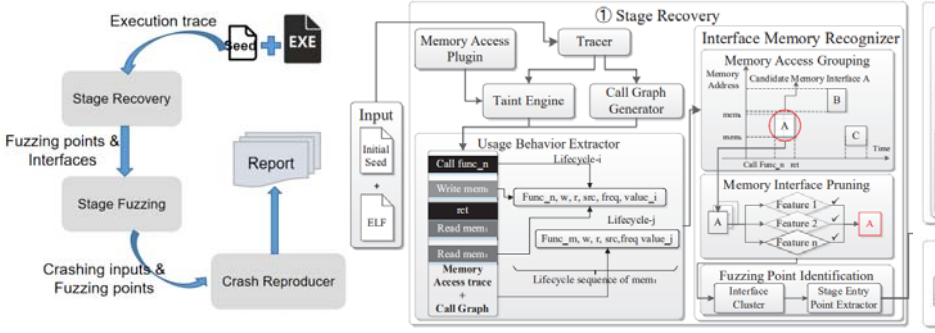
- 混合测试方法（基于符号执行等技术）
  - 负载高&扩展性差
  - 路径爆炸问题
  - 对于复杂路径约束依然无法有效求解
- 细粒度测试方法（函数或API级测试）
  - 测试目标选择问题
  - 测试用例构造问题
  - 缺乏软件真实运行环境导致的误报和漏报问题

方案设计概述

- 复杂程序设计思想
  - 基于组件的设计开发可以保证代码的可重用性和维护性
  - 程序设计者将程序划分为组件集合，各组件间功能独立，且相互之间通过数据接口进行信息交互
- 程序逻辑执行阶段
  - 静态识别程序组件是困难问题，因此我们在程序动态运行时检测与组件属性相似的程序逻辑执行阶段
  - 逻辑执行阶段的功能相对独立，其以内存数据接口作为自身的逻辑输入
- 基于程序逻辑执行阶段识别的定向测试方案
  - 我们使用数据流及控制流分析方法识别程序执行过程中创建的逻辑执行阶段的内存接口，并使用内存模糊测试技术、通过在程序运行时动态修改内存接口中的数据对各逻辑执行阶段进行定向测试，对于测试中发现的崩溃我们提取路径约束并进行验证。



系统结构



- 程序逻辑处理阶段内存接口识别
  - 基于数据流及控制流方法定位程序逻辑处理阶段的逻辑内存输入接口
- 程序逻辑处理阶段定向测试
  - 基于内存模糊测试方法为逻辑处理阶段的测试提供程序真实运行环境
  - 结合系统特性提供高效的测试方法
- 程序崩溃验证
  - 分析崩溃是否为误报

覆盖率结果

TABLE I. INSTRUCTION COVERAGE. STF AND AFL REPRESENT THE NUMBER OF INSTRUCTIONS ONLY COVERED BY STAGEFUZZER AND AFL. COM REPRESENTS THE INSTRUCTIONS COVERED BY BOTH FUZZERS. AFL+COM REPRESENTS ALL THE INSTRUCTIONS COVERED BY AFL.

Program	AFL	STF	COM	STF−AFL / AFL+COM
convert	3057	8431	61755	8.29%
fax2ps	213	5133	29640	16.48%
html2text	0	8025	17542	45.75%
objdump	0	3175	53186	5.97%
pngfix	220	7526	27194	26.65%
tiff2bw	587	5620	35769	13.84%
tiff2pdf	468	5569	49705	10.17%
tiffcp	521	3903	34971	9.53%
tiffcrop	5959	6150	38049	0.43%
tiffinfo	0	10792	21311	50.64%
wvRTF	9217	17646	68751	10.81%
wvWare	16798	30236	76239	14.44%
xmllint	1197	3459	74529	2.99%
xsltproc	59	17629	52246	33.59%
AVG	2735	9521	45777	17.83%

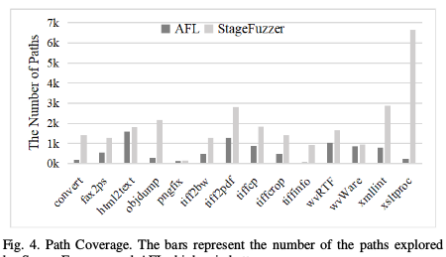


Fig. 4. Path Coverage. The bars represent the number of the paths explored by STAGEFUZZER and AFL; higher is better.

- StageFuzzer的指令及路径覆盖率比AFL更高

内存接口识别结果验证

TABLE II. RESULT OF STAGE IDENTIFICATION. TRACE SIZE REPRESENTS THE LENGTH OF AN EXECUTION TRACE. TIME REPRESENTS THE TIME CONSUMPTION OF THE STAGE IDENTIFICATION. NUMBER OF STAGES REPRESENTS THE CANDIDATE STAGES THAT STAGEFUZZER REPORTED. VERIFIED REPRESENTS THE CANDIDATE STAGE WAS VERIFIED BASED ON THE ABOVE MEASURES.

Program	Trace Size	Time (sec)	Number of Stages	Verified Stages	Copied Buffer	Processed Buffer	Original Buffer
convert	10747904	154	8	7	4	3	1
fax2ps	458752	16	4	4	3	0	1
html2text	2424832	46	5	5	4	0	1
objdump	11665408	197	3	3	1	1	1
pngfix	18087936	322	6	6	3	2	1
tiff2bw	1835008	45	6	5	5	0	1
tiff2pdf	3866624	94	7	7	6	0	1
tiffcp	2621440	64	4	4	3	0	1
tiffcrop	1245184	34	6	6	5	0	1
tiffinfo	33292288	444	5	5	4	0	1
wvRTF	21364736	385	8	8	7	0	1
wvWare	44957696	716	8	6	7	0	1
xmllint	82378752	1068	4	4	3	0	1
xsltproc	2031616	49	5	5	3	1	1
SUM	236978176	3634	79	75	58	7	14

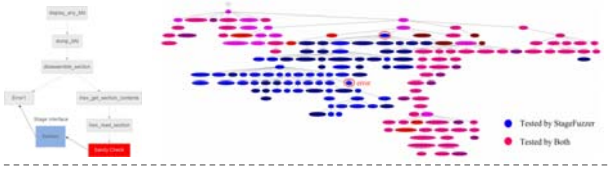
漏洞检测

TABLE III. CRASHES DETECTED BY STAGEFUZZER AND AFL.

Program	AFL	STF #Report	STF #Trac	STF %FN	Program	AFL	STF #Report	STF #Trac	STF %FN
wvRTF	6	11	8	27%	tiff2pdf	0	0	0	0
wvWare	4	6	4	33%	tiff2ps	0	0	0	0
objdump	0	1	1	0	tiff2rgba	0	0	0	0
convert	0	0	0	0	tiffcp	0	0	0	0
fax2ps	0	0	0	0	tiffcrop	0	0	0	0
html2text	0	0	0	0	tiffinfo	0	0	0	0
pngfix	0	0	0	0	xmllint	0	0	0	0
tiff2bw	0	0	0	0	xsltproc	0	0	0	0

- StageFuzzer发现的程序崩溃数量比AFL更多

实例分析



- 测试程序运行指令objdump -mi386:i386 -D
- AFL被校验和逻辑所阻塞，只能测试文件格式分析及验证代码，如红色节点所示
- StageFuzzer识别出了文件内容处理阶段的逻辑内存接口，从而实现了对该部分代码的有效测试，如蓝色节点所示