discovRE: Efficient Cross-Architecture Identification of Bugs in Binary Code

MAR 9TH, 2016

论文下载

Abstract & Introdutcion

背景与SP'15类似(Cross-Architecture Bug Search in Binary Executables)

- 源码bug检测难以满足需求，因为软件闭源和编译环境问题
- 当前二进制相似代码检测的问题：
  - 动态方法难以跨架构
  - 基于语义特征的方法效率低下

Approach

- 为了提高效率，只能静态分析
- 分两步走，先用简单的特征向量缩小比较范围，而后用复杂的图同构比较方法进一步比较

Coarse Grain

- 特征向量包括：

Feature	sd(values)	values	avg.cor	sd(cor)
Arithmetic Instr.	39.483	623	0.907	0.109
Function Calls	22.980	273	0.983	0.073
Logic Instr.	49.607	625	0.953	0.067
Redirections	40.104	556	0.978	0.066
Transfer Instr.	163.443	1,635	0.961	0.075
Local Vars.	2.78E6	890	0.983	0.099
Basic Blocks	48.194	619	0.978	0.067
scc	25.078	389	0.942	0.128
Edges	76.932	835	0.979	0.066
Incoming Calls	46.608	261	0.975	0.086
Instr.	295.408	2,447	0.970	0.069
Parameters	2.157	38	0.720	0.228

TABLE II: Robustness of numeric features. The selected features are highlighted.

- 平均相关度越高，标准差越低，说明特征越鲁棒
- 用KNN算法计算距离

Fine Grain

- 最大子图同构匹配
- 接下来是图论的问题；（原本看看图的表示方法觉得很高大上，刚才才发现就是{节点, 边}）
- （子图同构匹配是经典问题了，貌似最近有风声说复杂度要降低，反正这里还是当NP完全的）
- 已有方法复杂度高，没法破，限定循环数量折中

Evaluation

Correctness

- 将openssl 1.0.1e在不同编译环境、架构上编译，得到1293个函数
- 随机选取函数，在代码库里搜索，重复100w次(?!)
- 准确率高达93.93%

跨架构漏洞查找

- 几乎照搬SP'15的实验
- 结果从效率和效果双重打压对方