

验,将最后的结果统计成正确率、漏报率和误报率,统计结果见表 3 和表 4.由于 DroidLeaks 的检测结果中误报率均为 0,所以没有呈现.

Table 3 Detecting results using DroidLeaks

方法	正确率(%)	漏报率(%)
FindBugs	6.25	93.75
Fortify	87.50	12.50
本文方法	93.75	6.25

Table 4 Detecting results using Defects-bench

方法	正确率(%)	漏报率(%)	误报率(%)
FindBugs	38.03	84.65	1.41
Fortify	59.15	39.44	0.00
本文方法	61.97	35.21	1.41

评估中准确率包括 3 部分:正确率、漏报率和误报率.正确率是指测试用例的确存在着资源泄漏问题而工具也检测出来该问题(true positive)和测试用例没有资源泄漏的问题工具也没有检测出来问题(true negative)个数的和;漏报是指测试用例存在资源问题但是检测工具并没有检测出来该问题(false negative);误报是指测试用例不存在资源泄漏问题但是方法检测出来存在问题(false positive).从表中可以看出:在两个测试用例集中,本文方法表现出比 FindBugs 和 Fortify 更高的正确率、较低的误报及漏报率.这是由于本文应用了过程间路径敏感的数据流分析和别名分析.FindBugs 漏报率较高的原因是其不支持方法间的资源泄漏检测.

为了验证本文实现的资源泄漏检测算法针对大型项目的实际检测效果,分别使用本文所提出的算法和 Fortify 对开源项目 JExcelApi 进行检测,其中,Fortify 检出 7 个资源泄漏,本文方法检出 11 个资源泄漏.经人工验证,Fortify 和本文方法同时检出 6 个资源泄漏,且分别有 1 个误报.本文方法检出数远高于 Fortify,具有较低的误报率和漏报率.

4.3 增量检测分析

在进行增量检测时,本文对召回率的定义是:针对每个全局检测中检测出来的问题,假设每种方法为当前编辑方法,从每种方法开始进行增量检测,看是否能够再次检测到这些问题.本文对大型测试用例进行时间评估和召回率评估,对小型测试用例只进行召回率的评估.在本实验中,设置使用指向分析的阈值 K 为 200.

图 8~图 11 分别表示 Tomcat、Weka、Freeplane 和 JExcelApi 在增量检测中每种方法完成检测的时间散点图分布.横坐标表示入口方法序号,纵坐标则表示从该方法开始进行增量检测完成检测的时间.

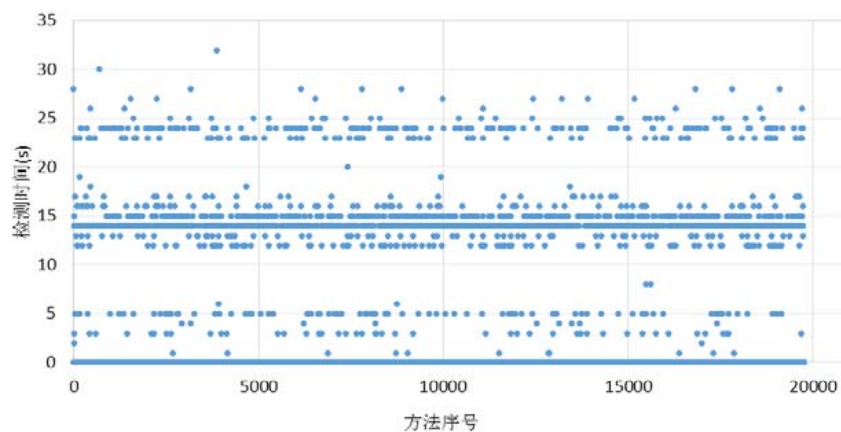


Fig.8 Tomcat's time distribution of incremental detection

图 8 Tomcat 中方法增量检测时间分布

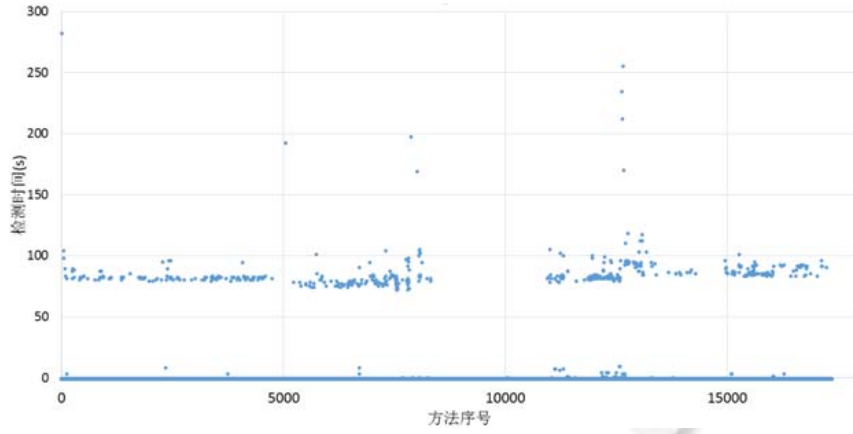


Fig.9 Weka's time distribution of incremental detection
图9 Weka 中方法增量检测时间分布

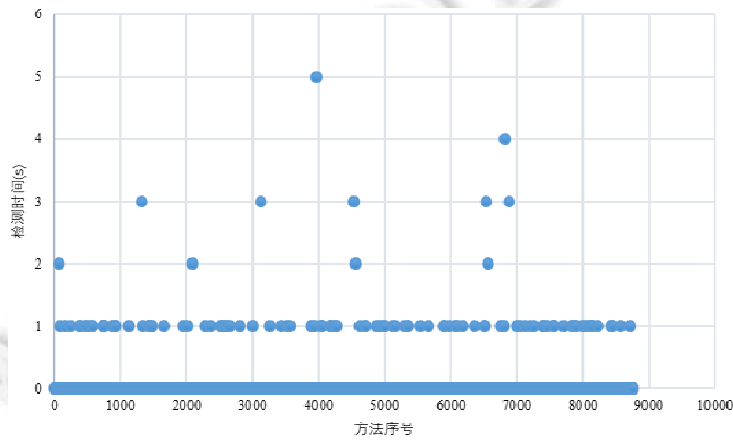


Fig.10 Freeplane's time distribution of incremental detection
图10 Freeplane 中方法增量检测时间分布

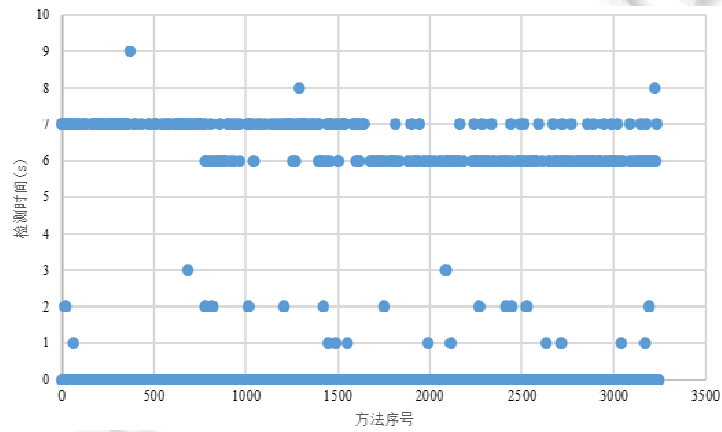


Fig.11 JExcelApi's time distribution of incremental detection
图11 JExcelApi 中方法增量检测时间分布

图 12(a)~图 12(d)分别表示 Tomcat、Weka、Freeplane、JExcelApi 在增量检测中每种方法完成检测的时间分布.

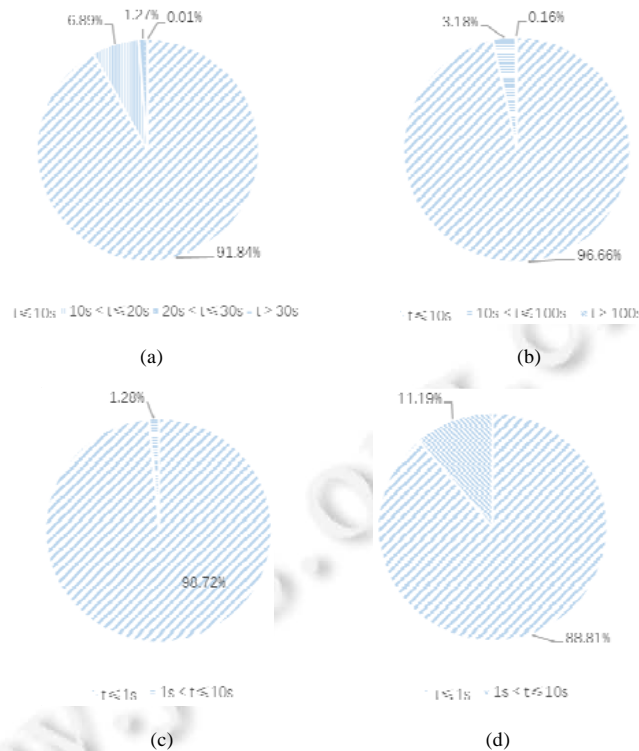


Fig.12 Time proportion distribution in incremental detection

图 12 方法增量检测时间占比分布

根据图 8~图 11,Freeplane 和 JExcelApi 中的增量方法检测基本在 10s 内出结果, Tomcat 在近 2 万种方法中, 绝大多数的方法增量检测都是在 30s 内完成, 只有极个别的方法执行时间超出 30s, 但也都控制在 35s 以内. 由于 Weka 比 Tomcat 超出了接近 20 万行代码, 导致 Weka 的部分方法检测时间超过了 100s. 但是从图 12(a)和图 12(b)的时间分布图来看, Tomcat 和 Weka 检测时间小于 10s 的方法占比仍达到 90% 以上, 因此, 大多数情况下仍可以达到即时检测和报告的效果.

表 5 给出了增量检测相对于批量全局检测的召回率情况.

Table 5 Recall ratio of different benchmarks in incremental detection

表 5 不同测试用例集在增量检测中的召回率

测试用例名	召回率(%)
DroidLeaks	100.00
Defects-bench	100.00
Tomcat	98.26
Weka	87.87
Freeplane	100.00
JExcelApi	100.00

可以看出:对于小规模测试程序,本文所提出的方法虽然实施的增量检测,仍然能够达到 100%的召回率;即使在大型项目中,Freeplane 和 JExcelApi 的召回率也是 100%,但是对于 Weka 和 Tomcat,仍然能够获得 87%以上的召回率,亦即全局批量检测中超过 87%的资源泄漏可以在增量检测中被即时检测到.召回率降低主要是由于方法调用图在增量代码分析的时构建不完整造成的.在 Java 项目中,由于多态的特性,从某一个方法出发,并不

能确定某些对象实际对应的类,所以也不能确定其对应的成员方法,在进行增量分析时,构建的方法调用图不完整,故召回率达不到 100%.因此,实际应用中可以采用增量和批量相结合的方式,既解决了检测时间的问题,也保证了较高的准确率.

5 总结

本文针对大规模代码提出了一种增量式资源泄漏分析与检测方法.该方法以用户修改的方法为入口点,进行即时方法间资源泄漏检测.传统的资源泄漏检测算法在代码修改量远小于原始代码量的情况下,通常耗费大量的时间进行冗余检测分析,极大地降低了检测效率.本文提出的针对大规模项目代码的增量式资源泄漏检测算法,是在进行资源泄漏检测的过程中,通过逐步缩小待分析资源相关方法的范围,并对未进行资源操作的相关方法进行“剪枝”,从而避免非资源相关方法的冗余分析,提高了资源泄漏检测的效率和准确性.该方法支持过程间流敏感的资源泄漏检测,在用户编辑代码的过程中,从变更的函数入手,通过资源闭包求解、指向分析过滤等多种技术手段缩小资源泄漏检测的范围,进而实现了几十万行代码的即时缺陷分析与报告.与现有的工具比较分析发现:本文所提出的方法在保证准确率的前提下,90%的增量检测实验在 10s 内完成.实验结果表明,本文所提出的方法能够满足在用户编辑程序过程中即时对缺陷进行检测和报告的实际应用需求.

References:

- [1] Wang KC, Wang TT, Su XH, Ma PJ. Key scientific issues and state-art of automatic software fault localization. *Chinese Journal of Computers*, 2015,38(11):2262–2278 (in Chinese with English abstract).
- [2] Wang T. Research and implements on static detection technology for source-oriented software vulnerability [MS. Thesis]. Wuhan: Huazhong University of Science and Technology, 2015 (in Chinese with English abstract).
- [3] Shirazi J, Pepperdine K, Hutt JT. Java performance tuning. 2010. <http://www.javaperformancetuning.com/news/news116.shtml>
- [4] Xiao Q, Gong YZ, Yang ZH, Jin DH, Wang YW. Path sensitive static defect detecting method. *Ruan Jian Xue Bao/Journal of Software*, 2010,21(2):209–217 (in Chinese with English abstract). <http://www.jos.org.cn/1000-9825/3782.htm> [doi: 10.3724/SP.J.1001.2010.03782]
- [5] Yang X, Gong YZ, Jin DH. A method of detecting resource leak based on static analysis. In: *Proc. of the 3rd National Software Test Conf. and Advanced Forum on Mobile Computing, Grid, Intelligence*. 2009. 5–9 (in Chinese with English abstract).
- [6] Ayewah N, Pugh W, Morgenthaler JD, Penix J, Zhou YQ. Using findbugs on production software. In: *Proc. of the ACM Sigplan Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2007)*. 2007. 805–806. [doi: 10.1145/1297846.1297897]
- [7] Péter R. Rice HG. Classes of recursively enumerable sets and their decision problems. *Trans. of the American Mathematical Society*, 1953,74(2):358–366. [doi: 10.1090/S0002-9947-1953-0053041-6]
- [8] Zhao YS, Gong YZ, Liu L, Xiao Q, Yang ZH. Improving the efficiency and accuracy of path-sensitive defect detecting. *Chinese Journal of Computers*, 2011,34(6):1100–1113 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.01100]
- [9] Broy M, Denert E. *Pioneers and Their Contributions to Software Engineering*. Berlin, Heidelberg: Springer-Verlag, 2001. [doi: 10.1007/978-3-642-59412-0]
- [10] Torlak E, Chandra S. Effective interprocedural resource leak detection. In: *Proc. of the 32nd ACM/IEEE Int'l Conf. on Software Engineering*. ACM Press, 2010. 535–544. [doi: 10.1145/1806799.1806876]
- [11] Do LNQ, Ali K, Livshits B, Bodden E, Smith J, Murphyhill ER. Just-in-Time static analysis. In: *Proc. of the ACM Sigsoft Int'l Symp.* ACM Press, 2016. 307–317. [doi: 10.1145/3092703.3092705]
- [12] Vallérai R, Co P, Gagnon E, Hendren L, Lam P, Sundaresan V. Soot—A Java bytecode optimization framework. In: *Proc. of the Masters Abstracts Int'l*. 2000. 214–224.
- [13] Evanno G, Regnaut S, Goudet J. Optimizing Java bytecode using the soot framework: Is it feasible. In: *Proc. of the Int'l Conf. on Compiler Construction*. Springer-Verlag, 2000. 18–34. [doi: 10.1007/3-540-46423-9_2]
- [14] Mei H, Wang QX, Zhang L, Wang J. Software analysis: A road map. *Chinese Journal of Computers*, 2009,32(9):1697–1710 (in Chinese with English abstract).

- [15] Umanee N. Shimple: And Investigation of Static Single Assignment Form [MS. Thesis]. School of Computer Science, McGill University, 2006.
- [16] Vallee-Rai R, Hendren LJ. Jimple: Simplifying Java bytecode for analyses and transformations. Technical Report, 1998-4, Sable Research Group, McGill University, 1998.
- [17] Liu YP, Wei LL, Xu C, Cheung SC. DroidLeaks: Benchmarking resource leak bugs for android applications. CoRR abs/1611.08079, 2016.

附中文参考文献:

- [1] 王克朝,王甜甜,苏小红,马培军.软件错误自动定位关键科学问题及研究进展.计算机学报,2015,38(11):2262-2278.
- [2] 王涛.面向源码的软件漏洞静态检测技术研究与实现[硕士学位论文].武汉:华中科技大学,2015.
- [4] 肖庆,宫云战,杨朝红,金大海,王雅文.一种路径敏感的静态缺陷检测方法.软件学报,2010,21(2):209-217. <http://www.jos.org.cn/1000-9825/3782.htm> [doi: 10.3724/SP.J.1001.2010.03782]
- [5] 杨绣,宫云战,金大海.一种基于静态分析的资源泄漏检测方法.见:第3届全国软件测试会议与移动计算网格、智能化高级论坛论文集.2009.5-9.
- [8] 赵云山,宫云战,刘莉,肖庆,杨朝红.提高路径敏感缺陷检测方法的效率及精度研究.计算机学报,2011,34(6):1100-1113. [doi: 10.3724/SP.J.1016.2011.01100]
- [14] 梅宏,王千祥,张路,王戟.软件分析技术进展.计算机学报,2009,32(9):1697-1710.



高志伟(1991—),男,山西阳泉人,硕士,CCF 学生会员,主要研究领域为软件分析与可靠性.



高玉金(1974—),男,博士,讲师,主要研究领域为并行程序设计,计算机体系结构.



计卫星(1980—),男,博士,副教授,CCF 专业会员,主要研究领域为计算机系统结构,并行计算与高性能计算,程序分析与优化.



廖心怡(1993—),女,硕士,CCF 学生会员,主要研究领域为高性能计算.



石剑君(1991—),女,博士生,主要研究领域为软件分析与可靠性.



罗辉(1991—),男,硕士生,CCF 学生会员,主要研究领域为静态代码检测.



王一拙(1979—),男,博士,讲师,CCF 专业会员,主要研究领域为计算机系统结构,并行编程模型.



石峰(1961—),男,博士,教授,博士生导师,主要研究领域为多核处理器体系结构,并行与分布式计算.