述方法中完成更进一步精度提升的.我们将在后续的工作中,将多源技术结合机器学习技术及更多语义信息来探索高精度的恶意软件分类方法.

7 总结与未来的工作

本文以当前污点分析检测 Android 应用隐私泄露的误报率高为出发点,针对恶意软件和良性软件之间的多源绑定发生特性差异来提高检测精度.我们提出了一种多源绑定发生的静态污点分析技术:在精度上,该技术具有上下文敏感、流敏感、域敏感等特性,并且可以有效地区分出分支互斥路径的情况;在效率上,提出一种按需的高效实现方法,降低了多次的多源问题计算的开销.我们的实验数据表明:即使在一次污点分析结果上进行多次多源问题分析,我们的执行时间也在可以接受的范围之内(初始阶段开销为 19.7%,进一步的多源分析时间平均为 0.3s).随后验证了多源绑定发生技术在隐私泄露检测问题中的应用,发现当前良性软件和恶性软件的多源绑定发生特性确实具有较大的差异.随后,我们验证了多源技术与简单方法相比的精度提升(减少多源对 41.1%).最后提出了一种智能手机隐私泄露结果风险评级标准,应用此评级结果,可以进一步改善用户的检测的效率.

在未来的工作中,我们将把该技术与其他应用技术相结合,以提供更高的恶意软件检测精度.

- 首先,该技术可以结合更多语义信息,基于语义信息的恶意软件检测方法可以避免代码混淆技术带来的威胁,而多源绑定发生技术为更多语义信息支持提供可能性,一些有效的语义信息包括 Android 生命周期、特殊的回调函数、关键的 API 使用、GUI 信息特征、特殊的分支触发条件等.例如,我们可以提取污点分析路径中的关键 API 作为语义信息基础,探索多个绑定发生源之间的 API 调用序列之间的关系.又如,我们可以探索在一个特殊的 Button 触发下,有哪些多源被绑定发生.如果这些源绑定发生行为与 Button 本身语义违反,则可以报告相关问题.
- 其次,我们将探索更有效的统计分析方法.目前,利用该方法在检测 Android 安全问题上有很大的应用空间.例如,MudFlow 尝试将待分析程序与良性软件间的差异作为特征,进一步利用 SVM 分类器进行检测.DroidADDMiner[8]利用了 FlowDroid 提取 API 数据之间的依赖关系作为特征向量进行恶意软件检测.多源方法在上述方法中的应用需要更有效的统计方法支持.另外一个值得研究方向是探索应用市场中不同的目录类别下的应用程序中多源问题的使用情况,因为相同类别下的 APK 往往会有类似的使用结果,探索相关的多源特性违反情况可以提供保证软件质量和软件安全的特征.

此外,多源绑定的污点分析技术还可以用来对别名分析进行优化.非别名的污点传播是跟随程序的执行而传播的,然而对于别名的传播往往不能通过直接的计算可以得到.例如,FlowDroid 尝试在遇到堆变量赋值计算别名时启动一个后向的别名分析求解器.此时,如果后向的别名传播和正向的变量在分支互斥的路径下,则会产生误报.例如,如图 13 所示,*a* 和 *b* 别名的前提是 IF 语句执行了第 5 行的分支,而实际的污点传播则是通过第 7 行进行的.此时,我们可以利用多源绑定的技术对该问题进行精度提升(设置 *a* 为另一个污点源,判断 *a* 是否与 source 绑定发生).我们坚信多源技术是一类重要基础性的技术,未来将被更多程序分析领域应用.

```
1  void foo(){
2   A a = new A();  A b = ...
3    String p = source();  String q= ...
4   if (random()){
5      b = a;
6   } else{
7      q = p;
8   }
9    b. f = q;
10   sink(a.f);
11}
```

Fig.13 An example of alias analysis optimization

图 13 别名分析优化示例

References:

[1] McAfee. Mobile threat report. 2016. http://www.mcafee.com/us/resources/reports/rp-mobile-threat-report-2016.pdf

[2] Livshits VB, Lam MS. Finding security vulnerabilities in Java applications with static analysis. In: Proc. of the Conf. on Usenix Security Symp. USENIX Association, 2005. 262−266. https://www.usenix.org/legacy/event/sec05/tech/full_papers/livshits/livshits_html/

[3] Sabelfeld A, Myers AC. Language-based information-flow security. IEEE Journal on Selected Areas in Communications, 2003, 21(1):5−19. [doi: 10.1109/JSAC.2002.806121]

[4] Li L, Bissyandé TF, Papadakis M, Rasthofer S, Bartel A, Octeau D. Static analysis of Android apps: A systematic literature review. In: Proc. of the Information & Software Technology. 2017. 67−95. http://orbilu.uni.lu/handle/10993/26879

[5] Avdiienko V, Kuznetsov K, Gorla A, Zeller A, Arzt S, Rasthofer S, Bodden E. Mining apps for abnormal usage of sensitive data. In: Proc. of the 37th Int'l Conf. on Software Engineering (ICSE), Vol.1. IEEE Press, 2015. 426−436. [doi: 10.1109/ICSE.2015.61]

[6] Feng Y, Anand S, Dillig I, Aiken A. Apposcopy: Semantics-based detection of android malware through static analysis. In: Proc. of the 22nd ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. ACM Press, 2014. 576−587. [doi: 10.1145/2635868.2635869]

[7] Pan X, Wang X, Duan Y, Wang X, Yin H. Dark hazard: Learning-based, large-scale discovery of hidden sensitive operations in Android apps. In: Proc. of the NDSS. 2017. http://www.cs.ucr.edu/~heng/pubs/ndss2017.pdf

[8] Li Y, Shen T, Sun X, Pan X, Mao B. Detection, classification and characterization of Android malware using API data dependency. In: Proc. of the Int'l Conf. on Security and Privacy in Communication Systems. Cham: Springer-Verlag, 2015. 23−40. [doi: 10.1007/978-3-319-28865-92]

[9] Aho AV, Sethi R, Ullman JD. Compilers, Principles, Techniques. Boston: Addison Wesley, 1986.

[10] Reps T, Horwitz S, Sagiv M. Precise interprocedural dataflow analysis via graph reachability. In: Proc. of the 22nd ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. ACM Press, 1995. 49−61. [doi: 10.1145/199448.199462]

[11] Reps T. Program analysis via graph reachability. Information and Software Technology, 1998,40(11):701−726. [doi: 10.1016/S0950-5849(98)00093-7]

[12] Arzt S, Rasthofer S, Fritz C, Bodden E, Bartel A, Klein J, Le Traon Y, Octeau D, McDaniel P. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. ACM SIGPLAN Notices, 2014,49(6):259−269. [doi: 10.1145/2594291.2594299]

[13] Lam P, Bodden E, Lhoták O, Hendren L. The Soot framework for Java program analysis: A retrospective. In: Proc. of the Cetus Users and Compiler Infastructure Workshop (CETUS 2011), Vol.15. 2011. [doi: 10.1.1.221.5311]

[14] Rasthofer S, Arzt S, Bodden E. A machine-learning approach for classifying and categorizing Android sources and sinks. In: Proc. of the Network and Distributed System Security Symp. (NDSS). 2014. [doi: 10.14722/ndss.2014.23039]

[15] Arzt S, Bodden E. StubDroid: Automatic inference of precise data-flow summaries for the Android framework. In: Proc. of the 38th Int'l Conf. on Software Engineering. ACM Press, 2016. 725−735. [doi: 10.1145/2884781.2884816]

[16] Google play. https://play.google.com/store

[17] Zhou Y, Jiang X. Dissecting Android malware: Characterization and evolution. In: Proc. of the 2012 IEEE Symp. on Security and Privacy (SP). IEEE, 2012. 95−109. [doi: 10.1109/SP.2012.16]

[18] Fritz C, Arzt S, Rasthofer S, Bodden E, Bartel A, Klein J, Le Traon Y, Octeau D, McDaniel P. Highly precise taint analysis for Android applications. Technical Report, TUD-CS-2013-0113, EC SPRIDE, 2013. http://www.bodden.de/pubs/TUD-CS-2013-0113.pdf

[19] Lerch J, Hermann B, Bodden E, Mezini M. FlowTwist: Efficient context-sensitive inside-out taint analysis for large codebases. In: Proc. of the 22nd ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. ACM Press, 2014. 98−108. [doi: 10.1145/2635868.2635878]

[20] http://www.anzhi.com/applist.html

[21] http://virusshare.com

[22] Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Proc. of the 20th Int'l Conf. on Very Large Data Bases (VLDB'94), Vol.1215. 1994. 487−499. [doi: 10.1.1.100.247]

[23] Crandall JR, Chong FT. Minos: Control data attack prevention orthogonal to memory model. In: Proc. of the 37th Int'l Symp. on Microarchitecture (MICRO-37). IEEE, 2004. 221−232. [doi: 10.1109/MICRO.2004.26]

[24] Zhu Y, Jung J, Song D, Kohno T, Wetherall D. Privacy scope: A precise information flow tracking system for finding application leaks. Technical Report, EECS-2009-145, Berkeley: University of California, 2009.

[25] Clause J, Li W, Orso A. DYTAN: A generic dynamic taint analysis framework. In: Proc. of the 2007 Int'l Symp. on Software Testing and Analysis. ACM Press, 2007. 196−206. [doi: 10.1145/1273463.1273490]

[26] Luk CK, Cohn R, Muth R, Patil H, Klauser A, Lowney G, Wallace S, Reddi VJ, Hazelwood K. Pin: Building customized program analysis tools with dynamic instrumentation. ACM SIGPLAN Notices, 2005,40(6):190−200. [doi: 10.1145/1064978.1065034]

[27] Tripp O, Pistoia M, Fink SJ, Sridharan M, Weisman O. TAJ: Effective taint analysis of Web applications. ACM SIGPLAN Notices, 2009,44(6):87−97. [doi: 10.1145/1542476.1542486]

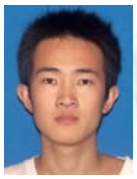
[28] Enck W, Gilbert P, Han S, Tendulkar V, Chun BG, Cox LP, Jung J, McDaniel P, Sheth AN. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans. on Computer Systems, 2014,32(2):393−407. [doi: 10.1145/2619091]

[29] Lu L, Li Z, Wu Z, Lee W, Jiang G. Chex: Statically vetting Android apps for component hijacking vulnerabilities. In: Proc. of the 2012 ACM Conf. on Computer and Communications Security. ACM Press, 2012. 229−240. [doi: 10.1145/2382196.2382223]

[30] Gordon MI, Kim D, Perkins JH, Gilham L, Nguyen N, Rinard MC. Information flow analysis of Android applications in DroidSafe. In: Proc. of the NDSS 2015. 2015. [doi: 10.14722/ndss.2015.23089]

[31] Li L, Bartel A, Bissyandé TF, Klein J, Le Traon Y, Arzt S, Rasthofer S, Bodden E, Octeau D, McDaniel P. Iccta: Detecting inter-component privacy leaks in Android apps. In: Proc. of the 37th Int'l Conf. on Software Engineering, Vol.1. IEEE Press, 2015. 280−291. [doi: 10.1109/ICSE.2015.48]

[32] Octeau D, Luchaup D, Dering M, Jha S, McDaniel P. Composite constant propagation: Application to Android inter-component communication analysis. In: Proc. of the 37th Int'l Conf. on Software Engineering, Vol.1. IEEE Press, 2015. 77−88. [doi: 10.1109/ICSE.2015.30]



王蕾(1989－),男,吉林白山人,博士,主要研究领域为程序分析,软件安全.

李炼(1977－),男,博士,研究员,博士生导师,CCF 专业会员,主要研究领域为程序分析,编译,自动测试,软件安全.

周卿(1987－),男,博士,主要研究领域为静态多线程程序分析.

冯晓兵(1969－),男,博士,研究员,博士生导师,CCF 杰出会员,主要研究领域为编程模型,编译优化.

何冬杰(1992－),男,硕士,主要研究领域为 Android 安全,静态分析,经验研究.