Computational Soundness for Dalvik Bytecode

Automatically analyzing information flow within Androidapplications that rely on cryptographic operations with theircomputational security guarantees imposes formidable chal-lenges that existing approaches for understanding an app’sbehavior struggle to meet. These approaches do not distin-guish cryptographic and non-cryptographic operations, andhence do not account for cryptographic protections: f(m)is considered sensitive for a sensitive message m irrespectiveof potential secrecy properties offered by a cryptographicoperation f. These approaches consequently provide a safeapproximation of the app’s behavior, but they mistakenlyclassify a large fraction of apps as potentially insecure andconsequently yield overly pessimistic results.In this paper, we show how cryptographic operations canbe faithfully included into existing approaches for automatedapp analysis. To this end, we first show how cryptographicoperations can be expressed as symbolic abstractions withinthe comprehensive Dalvik bytecode language. These ab-stractions are accessible to automated analysis and can beconveniently added to existing app analysis tools using mi-nor changes in their semantics. Second, we show that ourabstractions are faithful by providing the first computationalsoundness result for Dalvik bytecode, i.e., the absence of at-tacks against our symbolically abstracted program entailsthe absence of any attacks against a suitable cryptographicprogram realization. We cast our computational soundnessresult in the CoSP framework, which makes the result mod-ular and composable.

Android constitutes an open-source project not only interms of source code but also in terms of the whole ecosys-tem, allowing practically everyone to program new apps andPermission to make digital or hard copies of all or part of this work for personal orclassroom use is granted without fee provided that copies are not made or distributedfor profit or commercial advantage and that copies bear this notice and the full cita-tion on the first page. Copyrights for components of this work owned by others thanACM must be honored. Abstracting with credit is permitted. To copy otherwise, or re-publish, to post on servers or to redistribute to lists, requires prior specific permissionand/or a fee. Request permissions from permissions@acm.org.CCS’16 October 24–28, 2016, Vienna, Austriac ? 2016 ACM. ISBN 978-1-4503-4139-4/16/10...$15.00DOI: http://dx.doi.org/10.1145/2976749.2978418make them publicly available in Google Play. This open na-ture of Android has facilitated a rapid pace of innovation,but it has also led to the creation and widespread deploy-ment of malicious apps [1,2]. Such apps often cause privacyviolations that leak sensitive information such as locationor the user’s address book, either as an intended function-ality or as a result of uninformed programming. In somecases such apps can even extract sensitive information fromhonest apps.A comprehensive line of research has, hence, strived torigorously analyze how apps are accessing and processingsensitive information. These approaches typically employthe concept of information flow control (IFC), i.e., certaininformation sources such as GPS position and address bookare declared to be sensitive, and certain information sinksare declared to be adversarially observable. An IFC-basedanalysis then traces the propagation of sensitive informationthrough the program, i.e., if sensitive data m is input to afunction f, then the result f(m) is considered sensitive aswell. IFC-based analyses thereby determine if informationfrom sensitive sources can ever reach an observable sink, andin that case report a privacy violation.A considerable number of apps rely on cryptographic op-erations, e.g., for encrypting sensitive information before it issent over the Internet. However, analyzing information flowwithin Android apps that rely on such cryptographic opera-tions with their computational security guarantees imposesformidable challenges that all existing approaches for auto-mated app analysis struggle to meet, e.g., [3–5]. Roughly,these approaches do not distinguish cryptographic opera-tions from other, non-cryptographic functions. Thus, thestandard information-tracing mechanism for arbitrary func-tions applies: f(m) is considered sensitive for a sensitivemessage m irrespective of potential secrecy properties of-fered by a cryptographic function f, e.g., the encryptionof a sensitive message m is still considered sensitive suchthat sending this encryption over the Internet is considereda privacy breach. These approaches consequently provide asafe approximation of the app’s behavior, but they mistak-enly classify a large fraction of apps as potentially insecureand consequently yield overly pessimistic results. While ap-proaches based on manual declassification have successfullymanaged to treat cryptographic operations and their protec-tive properties more accurately, see the section on relatedwork for more details, no concept for an accurate crypto-graphic treatment is known for automated analysis of An-droid apps.

Dalvik字节码的计算健全性

自动分析Android应用程序中的信息流，这些应用程序依靠加密操作和计算安全保证，为现有的理解应用程序行为难题的现有方法带来了巨大的挑战。这些方法不区分密码和非密码操作，并且没有考虑密码保护：f（m）被认为对敏感消息敏感，而不管密码操作f提供的潜在保密属性。这些方法因此提供了对应用程序行为的安全逼近，但是他们错误地将大部分应用程序分类为潜在不安全并且随后产生过度悲观的结果。在本文中，我们展示如何将加密操作忠实地包括到用于自动应用程序分析的现有方法中。为此，我们首先展示如何使用全面的Dalvik字节码语言将加密操作表示为符号抽象。这些ab-stractions可以进行自动分析，并且可以方便地将其添加到现有的应用程序分析工具中，而不会改变其语义。其次，我们通过为Dalvik字节码提供第一个计算音质结果来证明失序是忠实的，即，没有针对我们的符号抽象程序的攻击，导致没有针对合适的加密程序实现的任何攻击。我们在CoSP框架中施加了计算的可靠性结果，这使得结果模块化和可组合。

Android构成了一个开源项目，不仅包含源代码，还包括整个生态系统，实际上每个人都可以编写新的应用程序和权限，以便将个人或课堂使用的全部或部分作品的数字或硬拷贝制作成如果副本不是为了利润或商业利益而制作或分发的，并且副本在第一页上载有本通知和全部引用。Android的这种开放特性促进了创新的快速发展，但也导致了恶意应用程序的创建和广泛部署[1,2]。此类应用程序通常会导致泄露敏感信息（例如位置或用户通讯录）的隐私违规，无论是作为预期的功能还是由于不知情的编程。在某些情况下，这些应用程序甚至可以从最有价值的应用程序中提取敏感信息。因此，全面的研究线索极力分析应用程序访问和处理敏感信息的方式。这些方法通常采用信息流控制（IFC）的概念，即某些信息源如GPS位置和地址簿宣称是敏感的，并且某些信息汇集被宣称是敌对的。然后基于IFC的分析通过程序追踪敏感信息的传播，即如果敏感数据m被输入到函数f，则结果f（m）也被认为是敏感的。基于IFC的分析从而确定敏感信息源是否能够到达可观察汇点，并且在这种情况下报告隐私侵害。相当多的应用程序依赖于加密操作，例如，在敏感信息通过互联网发送之前对其进行加密。然而，依靠这些加密操作及其计算安全性保证的Android应用程序分析信息流，给自动配对应用程序分析的所有现有方法都难以应付，例如[3-5]带来了难以克服的挑战。粗略地说，这些方法不会将加密操作与其他非加密函数区分开来。因此，适用于任意函数的标准信息追踪机制适用于：f（m）被认为对敏感信息敏感m，而不考虑密码函数f所赋予的潜在保密特性，例如，仍然考虑加密一个敏感信息m因此通过互联网发送这种加密被认为是隐私泄露。这些方法因此提供了对应用程序行为的不安全近似，但他们错误地将大部分应用程序分类为可能不安全，并因此产生过度悲观的结果。虽然基于手动解密的方法已经成功地进行管理，以更加准确地对待密码操作及其保护属性，但更多细节请参见关于相关工作的章节，但对于自动分析An-droid，没有关于准确密码处理的概念。