Statistical Deobfuscation of Android Applications

This work presents a new approach for deobfuscating An-droid APKs based on probabilistic learning of large codebases (termed“Big Code”). The key idea is to learn a prob-abilistic model over thousands of non-obfuscated Androidapplications and to use this probabilistic model to deob-fuscate new, unseen Android APKs. The concrete focusof the paper is on reversing layout obfuscation, a populartransformation which renames key program elements suchas classes, packages and methods, thus making it difficult tounderstand what the program does.Concretely, the paper: (i) phrases the layout deobfusca-tion problem of Android APKs as structured prediction ina probabilistic graphical model, (ii) instantiates this modelwith a rich set of features and constraints that capture theAndroid setting, ensuring both semantic equivalence andhigh prediction accuracy, and (iii) shows how to leveragepowerful inference and learning algorithms to achieve over-all precision and scalability of the probabilistic predictions.We implemented our approach in a tool called DeGuardand used it to: (i) reverse the layout obfuscation performedby the popular ProGuard system on benign, open-source ap-plications, (ii) predict third-party libraries imported by be-nign APKs (also obfuscated by ProGuard), and (iii) renameobfuscated program elements of Android malware. The ex-perimental results indicate that DeGuard is practically ef-fective: it recovers 79.1% of the program element namesobfuscated with ProGuard, it predicts third-party librarieswith accuracy of 91.3%, and it reveals string decoders andclasses that handle sensitive data in Android malware.

This paper presents a new approach for deobfuscatingAndroid applications based on probabilistic models. Ourapproach uses large amounts of existing Android programsavailable in public repositories (referred to as “Big Code”)to learn a powerful probabilistic model which captures keyfeatures of non-obfuscated Android programs. It then usesPermission 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 citationon the first page. Copyrights for components of this work owned by others than theauthor(s) must be honored. Abstracting with credit is permitted. To copy otherwise, orrepublish, to post on servers or to redistribute to lists, requires prior specific permissionand/or a fee. Request permissions from permissions@acm.org.this probabilistic model to suggest a (statistically likely) de-obfuscation of new, obfuscated Android applications. Ourapproach enables a variety of security applications. For in-stance, our system successfully deobfuscates Android APKsproduced by ProGuard [6], the most popular obfuscationtool for Android applications.Focus: Layout Deobfuscation.The focus of this paper is on reversing layout obfuscationof Android APKs. While general obfuscation can includeother transformations (e.g., changes to the program’s datarepresentation or control-flow [25]), layout obfuscation re-mains a key part of virtually all obfuscation tools. In layoutobfuscation, the names of program elements that carry keysemantic information are replaced with other (short) identi-fiers with no semantic meaning. Examples of such elementsare comments, variable, method and class names. Renamingthese program elements makes it much harder for humansto read and understand what the program does and is use-ful in a variety of security scenarios including protection ofintellectual property.Benefits and Challenges.Among others, reversing layout obfuscation for AndroidAPKs has various benefits including: (i) it makes it easierfor security analysts to inspect Android applications obfus-cated with ProGuard, (ii) it identifies third-party librariesembedded in Android APKs, and (iii) it enables one to au-tomatically search for certain identifiers in the code.However, reversing layout obfuscation is a hard problem.The reason is that once the original names are removed fromthe application and replaced with short meaningless identi-fiers, there is little hope in recovering the original names bysimply inspecting the application alone, in isolation.Probabilistic Learning from “Big Code”.To address challenges that are difficult to solve by consid-ering the program in isolation, the last couple of years haveseen an emerging interest in new kinds of statistical toolswhich learn probabilistic models from “Big Code” and thenuse these models to provide likely solutions to tasks thatare difficult to solve otherwise. Examples of such tasks in-clude machine translation between programming languages[18], statistical code synthesis [32, 30], and predicting namesand types in source code [31, 9]. Interestingly, due to theirunique capabilities, some of these probabilistic systems havequickly become popular in the developer community [31].This Work: Android Deobfuscation via “Big Code”.Motivated by these advances, we present a new approachfor reversing Android layout obfuscation by learning fromthousands of readily available, non-obfuscated Android ap-plications. Technically, our approach works by phrasing theproblem of predicting identifier names (e.g., class names,method names, etc.) renamed by layout obfuscation asstructured prediction with probabilistic graphical models.In particular, we leverage Conditional Random Fields (CRFs)[23], a powerful model widely used in various areas includ-ing computer vision and natural language processing. To ourknowledge, this is the first time probabilistic graphical mod-els learned from “Big Code” have been applied to address acore security challenge. Using our approach we present atool called DeGuard, and show that it can automaticallyreverse layout obfuscation of Android APKs as performedby ProGuard with high precision.

Android应用程序的统计反混淆

这项工作提出了一种基于大规模代码库概率学习（称为“大代码”）的反破坏An-droid APK的新方法。关键的想法是在数千个非模糊的Android应用程序中学习一个概率模型，并使用这个概率模型去消除新的，看不见的Android APK。本文的具体焦点是颠倒布局混淆，这是一种对重要程序要素（例如类，包和方法）进行重命名的流行变换，因此难以理解程序的作用。具体而言，本文：（i）短语布局消除问题（ii）通过丰富的特征和约束来实例化这个模型，这些特征和约束捕获Android设置，确保语义等价性和高预测准确性，以及（iii）展示如何利用强大的推理和学习算法来实现概率预测的总体精度和可扩展性。我们在一个名为DeGuard的工具中实现了我们的方法，并用它来：（i）逆转由流行的ProGuard系统在良性的开源应用程序中执行的布局混淆，（ii）预测通过be-nign APK（也由ProGuard混淆）导入的第三方库，以及（iii）重新命名的程序el Android恶意软件。实验结果表明，DeGuard实际上效果很好：它恢复了使用ProGuard进行扫描的程序元素名称的79.1％，它预测了第三方库的准确率为91.3％，并揭示了用于处理Android恶意软件中敏感数据的字符串解码器和类。

本文提出了一种基于概率模型去混淆Android应用程序的新方法。 Ourapproach使用公共存储库中的大量现有Android程序（称为“Big Code”）来学习捕获非混淆Android程序的关键特性的强大概率模型。然后，使用权限制作全部或部分作品的数字或硬拷贝供个人或课堂使用，只要复制品不是为了利润或商业利益而制作或发行的，并且副本在第一页中包含本通知和完整引用文件，则授予其免费使用。必须尊重他人拥有的作品组成部分的版权。允许用信用抽象。要另外复制或重新发布，要在服务器上发布或重新发布到列表，需要事先具体许可和/或收费。从permissions@acm.org请求权限。这个概率模型建议一个（统计上可能的）新的混淆的Android应用程序的去混淆。 Ourapproach支持各种安全应用。举例来说，我们的系统已经成功解除了由ProGuard [6]制作的Android APK的反混淆，这是Android应用中最受欢迎的混淆工具。聚焦：布局反混淆。本文的重点是反转Android APK的布局混淆。虽然通用混淆可能包含其他转换（例如，对程序的数据表示或控制流程[25]的更改），但布局混淆重新实现了几乎所有混淆工具的关键部分。在布局混淆中，携带关键语义信息的程序元素的名称被替换为其他（短）标识符，而没有语义含义。这些元素的例子是评论，变量，方法和类名。重新命名这些程序元素使得人们更难阅读和理解该程序的功能，并且在包括知识产权保护在内的各种安全情景中是有用的。益处和挑战。其他方面，颠倒AndroidAPK的布局混淆具有多种益处，包括:( i）它使安全分析师更容易检查与ProGuard混淆的Android应用程序，（ii）识别嵌入在Android APK中的第三方图书馆，以及（iii）它使人们能够自动搜索代码中的某些标识符。然而，颠倒布局混淆是一个难题。原因在于，一旦将原始名称从应用程序中删除并替换为无意义的简短标识符，通过单独地单独检查应用程序而孤立地恢复原始名称是没有希望的。概率学习“大规模”。为了解决孤立考虑项目难以解决的挑战，近几年来，人们开始关注新型统计工具，它们从“大代码”中学习概率模型，然后使用这些模型为难以解决的任务提供可能的解决方案。这些任务的例子包括编程语言之间的机器翻译[18]，统计代码综合[32,30]以及预测源代码中的名称和类型[31,9]。有趣的是，由于其独特的功能，这些概率系统中的一些已经在开发者社区中很流行[31]。本作品：通过“大代码”进行Android反混淆。受这些进步的影响，我们提出了一种通过学习反转Android布局混淆的新方法从成千上万的现成的，非模糊的Android应用程序。从技术上讲，我们的方法通过解决预测标识符名称（例如，类名称，方法名称等）的问题，通过布局混淆结构预测与概率图形模型重新命名。特别是，我们利用条件随机场（CRF）[23]，这是一个强大的模型，广泛应用于各个领域，包括计算机视觉和自然语言处理。据我们所知，这是第一次从“大码”中学到的概率图形模型已经被应用于解决安全挑战。使用我们的方法，我们提供了一个名为DeGuard的atool，并表明它可以自动反向处理由ProGuard执行的Android APK布局混淆，并且具有高精度。