Call Me Back! Attacks on System Server and System Apps

in Android through Synchronous Callback

Android is the most commonly used mobile device opera-tion system. The core of Android, the System Server (SS),is a multi-threaded process that provides most of the systemservices. Based on a new understanding of the security risksintroduced by the callback mechanism in system services,we have discovered a general type of design flaw. A vulner-ability detection tool has been designed and implementedbased on static taint analysis.We applied the tool on all the80 system services in the SS of Android 5.1.0. With its help,we have discovered six previously unknown vulnerabilities,which are further confirmed on Android 2.3.7-6.0.1. Accord-ing to our analysis, about 97.3% of the entire 1.4 billion real-world Android devices are vulnerable. Our proof-of-conceptattack proves that the vulnerabilities can enable a maliciousapp to freeze critical system functionalities or soft-reboot thesystem immediately. It is a neat type of denial-of-service at-tack. We also proved that the attacks can be conducted atmission critical moments to achieve meaningful goals, suchas anti anti-virus, anti process-killer, hindering app updatesor system patching. After being informed, Google confirmedour findings promptly. Several suggestions on how to usecallbacks safely are also proposed to Google.

Android is an operating system for mobile devices, whichis based on the Linux kernel. It occupies a large marketshare [7, 24] and is used in various mission critical tasks, suchas vehicle-mounted systems [3], POS devices [5, 6], medicaldevices [1, 2, 4] and aircraft navigation [22, 28]. In orderto make systems more powerful and secure, new versions ofAndroid are released at a fast pace. One important but often∗ Corresponding author.Permission 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/10.1145/2976749.2978342unnoticed result of the system updates is that the numberof system services has increased in every new version fromabout 50 in v2.3.7 to more than 100 in v6.0.0 1 .The number of system services is continually increasedbecause Android needs to: 1) support emerging hardware,such as Near Field Communication (NFC) and fingerprintscanning; and 2) support new functions, such as dynamicpermission authorization. It is clear that system servicesare critical function components in Android. They packagethe low level functionalities and provide essential higher lev-el functions to apps through the Inter-Process Communica-tion (IPC) mechanism in Android, named Binder. However,system services are very fragile since they provide easily ac-cessible interfaces to third-party apps, including maliciousapps. On Nexus 6 with Android 5.1.0, the System Server(SS) provides 80 Java-based services and exposes as manyas 1572 interfaces. In this sense, if one failure situation oc-curs during the handling of one service request, the wholeprocess may be affected. Since the SS is in fact the AndroidApplication Framework, such failure situations can disablesome core functionalities or even crash the entire system,which is clearly a single point of failure for Android system.This paper uncovers a general type of design flaw in theSS which is caused by improper use of synchronous callback.The callback handle is received from a client process (i.e.,an app). It is used to flexibly inform the client app aboutthe handling result of a service request. A malicious ap-p can forge a callback handle and inject it to the SS. Wefound that, if a synchronous callback is invoked under spe-cific conditions inside the SS or inside a cooperator systemapp, vulnerability would occur. This new family of vulner-ability is named as the “call me back” vulnerabilities.Using a synchronous callback to “communicate” with un-trusted apps without anticipating the worst-case situationsis indeed a design flaw from the security viewpoint. In thiswork, we have uncovered most if not all of these worst-casesituations. According to our study, in order to exploit a“call me back” vulnerability, a malicious app only needs toissue a single IPC call to the SS. The IPC sends a set ofparameters to a particular service interface in the SS. For avulnerable service interface, one of the parameters is a syn-chronous callback method handle. The hazard situations ofthe vulnerabilities are varied because the callbacks could beinvoked in different contexts of the SS, or could alternative-ly be invoked in the context of system apps, which are thecooperators of the SS. When invoked, the malicious callback1 Summarized based on the Genymotion emulator, whosesource code is identical to Android Open Source Project.92method can leverage two measures to conduct an attack: 1)prevent the callback method from returning, or 2) throw anexception. The attacks will result in the “freeze” of systemfunctionalities or even the soft-reboot of the system.According to our analysis, the attacks on the “call meback” vulnerabilities are difficult to detect and prevent. Webelieve the best defense method is to identify and patch thevulnerabilities as quickly as possible. However, there areseveral unique challenges: 1) callback handles can be inject-ed not only as an IPC call parameter, but also as an innerfield of a parameter object; 2) a callback handle can staydormant inside the SS context for a long period of time be-fore its invocation is triggered by some “not suspicious atall” SS operations; 3) a malicious app could try any partic-ular combination of the IPC call parameter values; 4) anysystem service and any system app could be vulnerable.We have designed and implemented a vulnerability de-tection tool which is based on static taint analysis. Ourtool can successfully address these challenges. We appliedit on all the 80 system service in the SS of Android 5.1.0and successfully identified 6 vulnerabilities. The vulnerabil-ities are further confirmed on Android 2.3.7-6.0.1. It meansthat about 97.3% [8] of the entire 1.4 billion real-world An-droid devices [9] are vulnerable. The attacks prove that thevulnerabilities can enable a malicious app to freeze criticalsystem functionalities or soft-reboot the system immediate-ly. We also proved that our attacks can be conducted atmission critical moments to achieve very meaningful goals.Our contributions are summarized as follows:• New Understanding and Discovery. Based on new un-derstanding of the security risks introduced by the callbackmechanism in system services, we have discovered a generaltype of design flaw which makes the Android system vulner-able to denial-of-service attacks.• Designing a New Vulnerability Detection Tool. We havedesigned and implemented a vulnerability detection tool basedon static taint analysis, which is the first work on detectingthe “call me back” vulnerabilities in the SS.• Identifying New Vulnerabilities. Our tool successfully an-alyzed 1,591 service interfaces of all the 80 system services inAndroid 5.1.0. We have discovered six previously unknownvulnerabilities which can affect about 97.3% of the entire 1.4billion real-world Android devices.• Attack. We have implemented several attack scenariosto show that attacks can be conducted at mission criticalmoments to achieve meaningful goals, such as anti anti-virus,anti process-killer, hinder app updates or system patching.• Defenses. We proved it is hard to distinguish the attackfrom benign service requests. The best way is to detect andpatch the vulnerabilities promptly. We also proposed severalsuggestions about how to use callbacks more safely.

给我打电话！ 通过同步回叫攻击Android系统服务器和系统应用程序

Android是最常用的移动设备操作系统。 Android的核心是系统服务器（SS），它是一个多线程进程，提供大部分系统服务。基于对系统服务回调机制引入的安全风险的新认识，我们发现了一种通用的设计缺陷。基于静态污点分析设计并实现了一种易感性检测工具。我们将该工具应用于Android 5.1.0 SS系统的所有80个系统服务中。在它的帮助下，我们发现了六个先前未知的漏洞，这些漏洞在Android 2.3.7-6.0.1中进一步确认。根据我们的分析，全部14亿个真实Android设备中约有97.3％是易受攻击的。我们的概念验证证明，这些漏洞可以使恶意应用程序冻结关键系统功能或立即软启动系统。这是一种干净的拒绝服务攻击。我们还证明，攻击可以在关键时刻进行，以实现有意义的目标，例如反病毒，反过程杀手，阻止应用程序更新或系统修补。得到通知后，Google立即证实了我们的发现。 Google也提出了几条安全使用回调的建议。

Android是基于Linux内核的移动设备操作系统。它占据了很大的市场份额[7,24]，并用于各种任务关键任务，例如车载系统[3]，POS设备[5,6]，医疗设备[1,2,4]和飞机导航[22,24] 28。为了使系统更加强大和安全，新版本的Android版本将以更快的速度发布。注意到系统更新的结果是系统服务的数量在每个新的版本从v2.3.7中的约50个增加到v6.0.0中的100个以上1。由于Android需要：1）支持新兴硬件，如近场通信（NFC）和指纹扫描等，系统服务数量不断增加; 2）支持动态许可授权等新功能。很显然，系统服务是Android中的关键功能组件。它们通过Android中的进程间通信（IPC）机制打包低级功能并为应用程序提供基本的高级功能，名为Binder。但是，系统服务非常脆弱，因为它们提供了对第三方应用程序（包括恶意应用程序）的轻松访问接口。在采用Android 5.1.0的Nexus 6上，系统服务器（SS）提供80个基于Java的服务，并提供1572个接口。从这个意义上说，如果在处理一个服务请求期间出现一个失败情况，整个过程可能会受到影响。由于SS实际上是Android应用程序框架，所以这些故障情况可能会禁用某些核心功能甚至使整个系统崩溃，这显然是Android系统的单点故障。本文揭示了SS引起的一般设计缺陷类型通过不正确使用同步回调。回调句柄是从客户端进程（即应用​​程序）接收的。它用于灵活地通知客户端应用程序关于服务请求的处理结果。恶意ap-p可以伪造回调句柄并将其注入到SS中。我们发现，如果在SS内或合作者systemapp内的特定条件下调用同步回调，就会发生漏洞。这个新的弱点系列被称为“回电”漏洞。使用同步回调与不受信任的应用程序进行“沟通”，而无需预见最糟糕的情况，这确实是安全角度的设计缺陷。在这项工作中，我们发现了大部分（如果不是全部）这些最糟糕的情况。根据我们的研究，为了利用“回电”漏洞，恶意应用程序只需要向SS发出单个IPC调用。 IPC将一组参数发送到SS中的特定服务接口。对于不易受损的服务接口，其中一个参数是同步回调方法句柄。漏洞的危险情况是多种多样的，因为回调可以在SS的不同上下文中被调用，或者可以替代地在系统应用（SS的合作者）的上下文中调用。调用时，恶意回调1基于Genymotion模拟器汇总，源码与Android开源项目相同.92方法可以利用两种方法进行攻击：1）防止回调方法返回或2）抛出异常。这些攻击会导致系统功能“冻结”，甚至导致系统软重启。根据我们的分析，对“call meback”漏洞的攻击难以检测和防范。 Webelieve最好的防御方法是尽快识别并修补漏洞。但是，有一些独特的挑战：1）回调句柄不仅可以作为IPC调用参数注入，而且可以作为参数对象的内部字段注入; 2）回调句柄可以在SS上下文中保持很长一段时间，直到其调用被某些“不可疑的所有”SS操作触发; 3）恶意应用程序可以尝试IPC调用参数值的任何特定组合; 4）系统服务和任何系统应用程序可能是脆弱的。我们已经设计并实施了基于静态污点分析的漏洞检测工具。 Ourtool可以成功解决这些问题