第41卷 第6A期
2014年6月

计 算 机 科 学
Computer Science

Vol. 41 No. 6A
June 2014

基于利用方式的 Android Root 漏洞分析

杨 超 刘文庆 张 伟 陈云芳
(南京邮电大学计算机学院 南京 210023)

摘 要 Android 平台恶意软件可以使用获取系统 root 权限的方式来绕过传统的 Android 安全机制,并且由于 Android 碎片化现象的存在,出现了很多利用方式、范围不同的 Android root 漏洞,因此有必要详细了解这些漏洞的实现机制,以采取相应的安全对策。基于利用方式将 Android root 漏洞按照是否可以直接在手机端利用的角度对其分类,评估其威胁程度,并详细描述了现有漏洞提权的实现细节、利用方式以及覆盖的范围,从而为进一步制定漏洞检测方案提供帮助。

关键词 Android root,提权过程,漏洞分析
中图法分类号 TP309.1 **文献标识码** A

Utilization Pattern Based Android Root Vulnerability Analysis

YANG Chao LIU Wen-qing ZHANG Wei CHEN Yun-fang
(College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract Malware on android platform can bypass the conventional android security mechanisms through the privilege escalation process. Due to the android fragmentation phenomenon, the emergence of root vulnerabilities in utilization patterns and scopes strongly urges the researchers' exhaustive understanding of these mechanisms to implement corresponding security policy. The categorization of android root vulnerabilities was proposed based on their utilization patterns of whether they can be directly utilized on the mobile terminal. And then a detailed description of the existing vulnerabilities in the implementation particulars, exploit patterns, as well as coverage was presented to help to further develop the vulnerability detection program.

Keywords Android root, Privilege escalation process, Vulnerability analysis

1 引言

随着 Android 系统的流行,Android 系统的安全性显得越来越重要。在 Android 安全的相关研究中,Android 的自带安全机制,比如权限机制、进程隔离(沙盒)机制等是研究者关注的重点。对 Android 自身安全机制已经有了比较透彻的分析,很多研究者依托这些机制提出了若干加强或者改善其安全机制的方法[1-3]。Android 系统是建立在 Linux 内核之上的,在 Linux 中进程可以取得超级用户的权限,即 root 权限,从而获取对系统资源的全面控制。Android 系统中 root 权限不向普通用户开放,但是并不证明普通用户无法通过其他手段获得。很多 Android 用户热衷于取得他们手机的 root 权限来随心所欲地修改他们的手机,黑客也热衷于寻找 Android 系统的漏洞来取得 root 权限。目前的状况是,虽然 Google 不向用户开放 root 权限,但是已经存在很多可以获取 Android 系统 root 权限的漏洞,而恶意软件可以通过这些漏洞取得 root 权限来绕过 Android 自身保护机制或者第三方提出的高级安全机制。

针对 Android 系统面临的 root 漏洞的威胁,一些研究者已经在他们的文章中有所提及,比如 Yajin Zhou 等人的工作[4],他们在分析 Android 恶意软件的时候提到,一些恶意软件已经在利用 root 漏洞扩展自身的权限来实施恶意行为。Won-Jun Jang 等人[5]则阐述了 root 漏洞对于 Android 系统上运行的一些金融服务应用的危害,并且提出了检测方法。NSA 则提出了 SELinux 在 Android 手机上的扩展 SEAndroid[6],SEAndroid 特别定制了针对获取 root 权限漏洞的策略来应对可能的 root 漏洞的攻击,但是策略的制定依赖于对具体的漏洞进行分析。

由于存在碎片化[7]现象,导致 Android 平台上不存在一个普适性的、可以覆盖所有或者大部分手机的漏洞。现在很多的 root 漏洞,都宣称可以取得 root 权限,但是其使用的前提条件均有较大的限制。目前针对 Android root 漏洞的系统研究还处于起步阶段,文献[4,5]工作里面提到的 root 漏洞只是一小部分,而 SEAndroid 工作目前也缺乏对大部分漏洞的应对策略。

本文基于漏洞是否可以被利用直接在手机端运行这种方

本文受国家自然科学基金(61272422)资助。

杨 超(1990—),男,硕士生,主要研究方向为信息安全、漏洞分析、恶意代码分析;刘文庆(1988—),女,硕士生,主要研究方向为信息安全、漏洞挖掘;张 伟(1973—),男,博士,教授,主要研究方向为网络安全、逆向分析、数据流检测;陈云芳(1976—),男,博士,副教授,主要研究方向为社会计算、人工免疫算法、信息网络安全。

式对 Android root 漏洞进行了分类分析，并且演示了这些漏洞的具体利用过程。本文第 2 节介绍 root 漏洞的传播、一般利用流程以及原始利用方式；第 3 节是对现有 root 漏洞的分类、分析及利用演示；最后是结论以及未来工作的方向。

2 root 漏洞的工作机制、传播及利用

2.1 root 漏洞的传播及利用流程

现在流行的 Android root 漏洞都是由黑客发现并且在 XDA 论坛[8]上公布的，XDA 是一个发烧友论坛，很多黑客喜欢在这里发布他们的最新成果。这些成果形式一般是一个利用 Android adb 工具的半自动化电脑端 root 工具，以及一份存放在开源代码网站 GitHub 上的源代码链接。用户可以下载 root 工具，将自己的手机通过 USB 线连上电脑，运行工具包中的 .bat 文件就可以完成自己手机的 root 工作。一个用户如果成功地获取了 root 权限，那么他会向其他发烧友推荐这一款工具，于是搭载一个漏洞的 root 工具就会传播开。在 root 工具包中，最核心的是其中的一份二进制可执行文件，此可执行文件是根据开放在 GitHub 上的源代码编译而成，编译工具是 Android NDK，ADB 工具会把它推送到手机上执行，然后获取 root 权限。

常见的 root 工具都会包含以下几个文件：漏洞可执行文件，su 文件以及 SuperUser.apk 文件。这些文件都是用户获取 root 权限不可或缺的文件，其中 su 文件是一个可执行文件，可以响应用户发出的“su”命令。在 Linux 系统中 root 权限是向普通进程开放的，普通进程要获得 root 权限只需要执行 su 命令，然后输入 root 账号密码即可获取权限。Linux 系统中，su 文件存放在/system/bin 目录下，而 Android 系统中/system/bin 目录下是没有 su 文件的。因此只要将 su 文件拷贝至这个目录下面，那么 Android 系统的普通进程也可以像 Linux 系统下的进程那样调用 su 文件来取得 root 权限。但是在 Android 系统中，/system 目录是写保护的，普通用户并没有向里面拷贝一份文件的权限，这个时候就需要执行 root 漏洞。root 漏洞在执行之后可以使进程获取临时的 root 权限（这也意味着这时重启手机就会失去这一权限），这时再通过 remount 命令重新挂载文件系统使/system 变为可写，之后就可以通过文件拷贝命令将 su 文件放入/system/bin 或者/system/xbin 目录。su 文件拷贝成功之后，即使重启手机，任何进程仍然可以通过调用“su”命令来请求 root 权限。

在 Android 系统安全管理中，虽然 Google 官方还没考虑 root 权限开放之后的管理问题，但是已经有很多黑客在考虑了，比如 Zinx 开发了 SuperUser.apk 来管理 root 权限。当系统中的一个 app 需要 root 权限时，可以使用 su 命令来请求，而该 su 文件是一个特别定制过的文件，必须通过 SuperUser.apk 来授予其他 app root 权限，目前 SuperUser.apk 由 ChainsDD 团队升级维护。

2.2 root 漏洞的利用方式

Android root 漏洞最广泛的利用方式就是搭载在电脑端的半自动化工具上供用户 root 自己的手机，这种方式需要用户将自己的手机通过 USB 线连上电脑后在电脑端进行操作，但是这种利用方式带来的对用户安全的攻击很有限。因为黑客想要进行攻击只能通过发布 PC 端的 root 助手软件，或者将嵌入式设备植入充电器或者移动电源中，当用户充电的时候也会用 USB 连接，这样也可以进行攻击，在 IOS 系统就上存在这样的攻击案例[9]。

如果所有的 root 漏洞的触发一定要通过 adb 工具，由于使用场景的限制，用户遭受安全攻击的风险还不是很大。但是有些 Android root 漏洞可以直接由 app 调用执行，设想一下这样带来的安全风险：用户下载了一个 app，并没有要求太多的权限，这也意味着 Google 传统的权限机制安全评估中这个 app 的风险不是很大，但是这个 app 中搭载了可以触发的 root 漏洞可执行文件。用户在运行这台 app 的时候，触发了 root 漏洞执行文件，使这个 app 获取了 root 权限，接下来的事情一目了然，这台 app 可以获取这台手机上的一切资源。举例来说，应用权限机制是 Android 系统重要的安全机制，普通 app 要获取权限如读取通信录、短信，访问网络等必须在 app 安装的时候申请，而具有获取 root 权限能力的恶意 app 在安装的时候不需要申请这些权限，但是却可以任意得到这些权限。

基于上述论述，有必要对现存的 Android root 漏洞进行一个分类评估。上文也提到，由于 Android 碎片化的问题，产生了很多 Android root 漏洞，利用的方式和作用的范围都不尽相同。本文对这些漏洞进行了详细分类、评估、利用显示，为进一步的漏洞检测打下基础。

3 对现存 Android root 漏洞的分类

表 1 列出了 2010 年以来典型的 Android root 漏洞的名称、适用 Android 系统版本和发布时间，可以看出各类不同的漏洞的交替发展。

表 1 典型的 Android root 漏洞

Anddroid root 漏洞名称	适用 Android 系统版本	发布时间	特点
Rageagainst-thecage	2.2 及以下	2010	支持版本较低，但普适性强
Gingerbreak	2.3.3 及以下	2011	需要一定硬件支持
ZergrRush	2.3.6 及以下	2011	不能再手机端触发，易用性差
Mempodroid	4.0.4 及以下	2012	Linux 内核漏洞
ADB Restore	4.1.1 及以下	2013	无法在手机端触发
CVE-2013-2094	4.2.2 及以下	2013	Linux 内核漏洞
SAMSUNG Exynos 4 CPU 漏洞	硬件 适用具体型号手机	2012	硬件漏洞
CVE-2013-2596	硬件 适用具体型号手机	2013	硬件漏洞

现在暴露出的 Android root 漏洞已经比较多，而且支持的 Android 系统版本不一，部分漏洞针对特定的硬件，具有较强的依赖性。随着 Android 系统版本的提升，有些漏洞在系统升级的时候被 Google 修补掉了，但是这些漏洞还是具有研究意义。因为越来越多的 Android 用户会为他们的手机刷上流行的第三方 ROM，这些 ROM 版本有一个吸引用户的地方就是他们有意或无意地向用户开放了 root 权限。对于这些第三方 ROM 而言，最简单的开放 root 权限的方式就是在定制 ROM 的时候将 Google 封掉的一些 root 漏洞重新开放。对于用户而言，虽然他们使用的 Android 系统版本比较高，但是一些低版本的 root 漏洞依然存在于他们的手机上，依然会被恶意软件利用，所以研究 Android root 低版本的漏洞依然要考虑。

Android root 漏洞分类分为两类，能直接在手机上触发的漏洞和不能直接在手机上触发的漏洞。前者因能直接在手机上触发而对用户的安全造成极大的威胁，后者不能直接在手机上触发，其威胁要低一点。我们将在第一类中依据漏洞的覆盖面来进行细分，并演示漏洞的利用，在第二类中分析这些漏洞不能在手机上直接利用的原因。

3.1 直接在手机端利用的漏洞

很多宣称可以一键 root 用户手机的 Android app 所调用的漏洞就是直接可以在手机端触发的漏洞，这些 app 有 KingRoot. apk[10]，SuperOneclick. apk[11]，Z4Root. apk[12]，Framaroot. apk[13]等。这些 app 都有相似的结构，都是将 3.2 节中提到的流程在一个普通 Android app 中实现出来。虽然在具体的代码实现上不尽相同，但是在思想上是比较一致的。本文以 Z4Root 为例展示在手机端利用 root 漏洞获取 root 权限的一般过程：

(a)通过 Android SDK 开发的方式开发一个 app，工程文件的结构是相似的。在 Z4Root 的工程中 root 过程用到的文件都放在 res->raw 文件夹下，这些文件包括漏洞利用二进制可执行文件、su 文件以及 superuser. apk。在需要用到这些文件时，app 通过一个 SaveIncludedFileIntoFilesFolder 方法将 raw 文件下的文件拷贝到进程数据空间中。这个方法主要使用了 JAVA 语言中的输入输出流操作。Android 平台下每个 app 存放数据的地方位于/data/data/appname 之下，此处的 appname 为该 app 的 ID，拷贝完成之后，app 就可以直接调用这些文件。

(b)在 app 中执行 root 漏洞如同在执行一个 Linux 脚本，典型的如 Z4Root 的执行过程，具体如下：

```
final FileDescriptor fd = Exec. createSubprocess ("/system/bin/sh","-",null,processId);
final FileOutputStream out=new FileOutputStream(fd);
final FileInputStream in=new FileInputStream(fd);
```

通过 JAVA OutputStream 可以向进程写入数据，即你想要执行的 sh 命令，也可以通过 InputStream 来得到进程返回的数据，这样就可以知道命令的执行结果。

(c)使用 Linux 的 chmod 更改前面拷贝进数据空间的漏洞二进制文件的执行权限(这个命令普通 sh 用户是有权限的)，然后就可以通过 Linux sh 执行这个可执行文件了。执行时需要在代码中设定条件等待这个可执行程序执行成功，取得临时 root 权限之后再通过 remount 命令重新挂载/system 目录，这样才可以继续使用 cp 命令或者 cat 命令将 su 和 superuser. apk 拷贝至/system 的子目录。如果第一步的可执行文件没有执行成功，或者没有执行完毕就执行后面的步骤，则系统会拒绝执行挂载命令或者文件拷贝命令。所以判断搭载漏洞二进制可执行文件的执行情况是一个 root app 最重要的工作，而每一个漏洞执行的情况都是不一样的，需要具体根据漏洞去判断。

3.1.1 使用限制最少，覆盖面最广的漏洞——Rageagainstthecage

Rageagainstthecage 漏洞[14]利用的是 Android 系统对权限回收的结果检查不严来获取 root 权限的。在 Android 系统启动之初，adbd 进程是具有 root 权限的，因为它要在系统启动时参与一些资源的配置，参与完成之后，系统会调用 setuid 函数将 adbd 的权限由 root 降为 shell，但是在 2.2 以及以前版本的 Android 系统中，setuid 函数没有返回值，所以也不检查这个函数执行的结果。Rageagainstthecage 漏洞利用程序的 fork 机制不停地产生僵尸进程，来挤占正常进程运行需要的进程空间，占满进程空间之后，杀死正在运行的只有 shell 权限的 adbd 进程。因为 adbd 是系统重要守护进程，所以被杀死之后系统会立刻将它重新启动，重新启动的 adbd 进程是具有 root 权限的，这时系统会调用 setuid 函数来对这个刚启动的 adbd 进程进行降权，但是这时进程执行空间已经被僵尸进程占满了，所以 setuig 函数的调用会失败，降权行为也就失败了。由于系统只是调用 setuid 函数，并不对降权的结果加以检查，于是重新启动的 adbd 进程的 root 权限就被保留下来，依托这一点来进行后续的工作，比如将 su 文件拷贝进/system 等行为。

这个漏洞的二进制可执行文件是可以直接在手机上运行的，运行完成之后就可以取得临时 root 权限。这样的漏洞不受手机品牌型号的限制，所以对于版本比较低的 Android 系统具有较大的普适性，恶意程序只需要调用漏洞可执行文件就可以获取 root 权限，易用性较强；这个漏洞虽然在 2.2 以上版本的官方系统中已经被封闭，但是如果有类似的漏洞出现，危害性还是很大的。

3.1.2 需要一定硬件支持的漏洞——GingerBreak

有一类需要硬件支持的漏洞是 Android 系统自身的漏洞，Android 系统在对硬件进行操作，或者响应一些硬件行为(比如热插拔)的时候需要 root 权限的参与。此类型的漏洞就是让一些硬件参与进来，使系统 root 权限与用户行为产生联系，进而获取。目前能在手机上直接开发的这样类型的漏洞是 GingerBreak[15]，这个漏洞也是由“The Android Exploid Crew”发布的。

GingerBreak 采用 HOOK 机制，通过代码修改/system/bin/vold 程序的 GOT 表项(全局偏移表)，将 strcmp()、atoi()等函数的地址更改为 system()函数的地址，然后触发调用 strcmp()或 atoi()来达到执行 system()的目的，而后者真正被执行后可以让用户执行 root shell，然后就可以执行从临时 root 向永久 root 转变的那些步骤。

这个漏洞在手机端直接利用的方式也是通过 sh 来执行 gingerbreak 二进制文件，然后这个可执行文件就会去执行上述几个步骤。但是这个可执行文件的执行不像上一小节提到的 Rageagainstthecage 那么简单。首先 gingerbreak 的执行需要手机有 SD 卡，因为这个程序在执行的时候需要把 SD 卡卸载掉。现在很多手机已经不支持外接 SD 卡了，如果没有外接 SD 卡这样的硬件支持，这个漏洞无法被成功执行。gingerbreak 漏洞在执行的时候会有复杂的文件复制检查，以及针对 Android 系统进程 vold 的系统消息的发送，这一整套的复杂机制不是每次都可以成功执行。我们在做关于这个漏洞的利用实验的时候，因为利用代码中的文件拷贝以及计算内存地址问题，就遇到过数次不成功的情况。

这个漏洞虽然没有特定的手机品牌机型的限制，但是因为需要硬件支持，而且触发的过程复杂，不能保证每次都成功，所以普适性并没有 Rageagainstthecage 这样的漏洞大；而且在利用的过程中不能保证成功率，所以易用性也不如 Rageagainstthecage；但是它可以支持版本为 2.3.3 以前所有

的手机,所以危害性比 Rageagainstthecage 漏洞要大。

3.1.3 Linux 本地提权漏洞在 Android 手机上的移植

Android 系统使用的是经过裁剪的 Linux 内核,所以一些 Linux 内核提权漏洞也可以移植到 Android 平台上,下面展示两个此类型的漏洞:

(1)CVE-2012-0056

CVE-2012-0056[16] mempodroid exploid 利用了 Linux 内核 2.6.39 和其他一些版本中的 mem_write 函数,在 ASLR 关闭时,其不能正确检查/proc/〈pid〉/mem 的写入权限,导致本地用户通过修改进程内存提升权限。在移植到 Android 系统平台之后有了一个新的名字叫 mempodroid,对应的二进制可执行文件也叫 mempodroid。漏洞利用程序在运行时需要提供 3 个参数:exit()函数地址、setresuid()函数地址以及一个命令,如 Root Galaxy Nexus 手机中该命令为:./data/local/tmp/mempodroid 0xd7f4 0xad4b sh。这个漏洞可以 root 4.0.4及以下的手机,所以普适性较强;在利用的过程中需要根据不同手机的特点提供不同的参数,所以易用性较弱;覆盖的手机比较多,危害性较大。

(2)CVE-2013-2094

CVE-2013-2094[17] 在 2013 年 5 月 14 日发布,所以一些搭载比较新的 4.2.2Android 系统的手机也可以被 root。这个漏洞的主要机制是 Linux kernel 3.8.9 之前版本在 kernel/events/core.c 的函数 perf_swevent_init 中使用了错误的整数数据类型,本地用户利用此漏洞,通过特制的 perf_event_open 系统调用,来提升自己的权限。但是在用户特制 perf_event_open 系统调用过程中需要用到几个内存地址,这在不同的手机上是不一样的。目前黑客们还在针对这个漏洞进行研究,计算更多型号的手机在该 root 过程中需要用到的参数。这个漏洞理论上可以适用于任何没有打过补丁的 Linux 内核,所以普适性很强,在利用的过程中同样需要不同参数,所以易用性较弱;覆盖手机面广,一旦发现增强普适性的办法,危害性将极大。

3.1.4 基于硬件的漏洞

对用户安全产生威胁的除了 Android 系统漏洞和 Linux 系统移植漏洞,还有各大 OEM 厂商对自己的 Android 手机定制的过程中产生的漏洞,其中会产生威胁较大的 root 漏洞的地方就是各大 OEM 厂商为自己的手机编写内核驱动代码的时候产生的漏洞。目前此类漏洞有:

(1)SAMSUNG Exynos 4 CPU 漏洞

SAMSUNG Exynos 4 CPU 漏洞[18]产生的原因是三星内核源代码中存在 graphics 用户组拥有读写/dev/exynos-mem(三星读写内存物理设备)的权限,导致任意程序在非 root 的条件下可以通过该设备接口直接修改内核 root 权限管理代码。这个漏洞的二进制可执行文件 exynos 运行之后,打开/dev/exynos-mem 内存映射,在内存中查找"%pK%c%s"并替换"%p"(用于强制显示内核描述符),然后打开"/proc/kallsyms"文件,找到 sys_setresuid(该函数为 root 权限管理函数)函数地址进行修改。

这是三星设备上的一个重大内核漏洞,此漏洞出现在包括 Galaxy S2、Galaxy S3、Galaxy Note、Galaxy Note2、Galaxy Note10.1、MEIZU MX 等使用三星 exynos 4210/4412 cpu 的设备上,无论设备的 Android 版本为多少,都可以被成功利用。这个漏洞可以直接在手机端触发,非常轻量级,用户不会有任何察觉。依据以上描述这个漏洞普适性较弱;易用性很强,只需调用漏洞执行文件即可;因为受普适性影响,危害性适中。

(2)CVE-2013-2596 高通平台漏洞

CVE-2013-2596 漏洞[19]和三星类似,有一款名为 Framaroot 的手机端一键 root app 就可以 root 多平台下的各类手机,其中有一大类全部都是高通平台的手机,就是利用的类似漏洞。这个 app 也支持上文提到的三星漏洞,当然还有其他硬件厂商的漏洞,读者可以进一步研究。

上面提到的两个漏洞都是基于硬件的漏洞,和具体 Android 系统无关。这样的漏洞对用户而言威胁很大,因为基于硬件的漏洞非常轻便,成功率也非常高。如果一个恶意软件搭载了几个不同平台的硬件漏洞,那么就足以威胁到很大一部分用户的安全了。

3.2 无法在手机端利用的漏洞

前面提及的漏洞都可以直接在手机端被直接利用,但是正如上文所分析,这些漏洞利用场景会被各种因素限制,这些因素包括 Android 系统版本、手机品牌和机型等。但是还存在一些漏洞,它们的利用方式更加苛刻,无法在手机上被触发,必须在电脑端通过 adb(Android Debug Bridege)工具来使用。因为 adb 工具可以使用 adb shell 在电脑端发出一些命令来让手机执行,而且 adb shell 比在手机上获取的普通用户 shell 权限更高。这也就是为什么一些漏洞需要使用 adb shell 来获取 root 权限的原因。此类漏洞有 ZergRush[20] 以及 ADB Restore[21]。这类漏洞的普适性较强,但是易用性很弱,所以危害性较弱。

(1)ZergRush

ZergRush 漏洞产生的原因是:具有 root 权限的 vold 进程使用了 libsysutils.so 库,该库的一个函数存在栈溢出,因此可以在 root 权限执行输入的 shellcode。存在漏洞的函数为 FrameworkListener::dispatchCommand,位于源码的\system\core\libsysutils\src\FrameworkListener.cpp 中,其中的局部变量 argv 为固定大小的指针数组,当输入参数的数量超过其大小时,会越界写入栈中。

ZergRush 漏洞的利用代码和 GingerBreak 差不多,但是使 ZergRush 漏洞无法在手机端直接利用的原因是:ZergRush 程序要伪造一条系统通知让 vold 进程去调用 libsysutils.so 库。在普通 app 中执行 ZergRush 漏洞执行文件,返回的结果是无法发送这样的消息给 vold 进程。而通过 adb 工具将 ZergRush push 到手机上再执行,就可以成功使 vold 进程调用 libsysutils.so 库进而完成接下来的工作。

(2)ADB Restore

ADB Restore 利用的是 adb shell 相比于普通 shell 具有更高的权限。在 Linux 系统中/tmp 目录是比较特殊的,任何用户对于这样的目录都有读写权限。而/data 目录则是 Android 系统存放重要数据的地方,普通用户对其连读权限都不具备。但是在/data 目录下存在一个/data/local/tmp 的子目录。ADB Restore 漏洞通过 Linux 的 ln 命令将/data 重新挂载到了/data/local/tm 目录下。这条 adb shell 命令为:adb shell ln-s/data/data/local/tmp。这样就可以对/data 目录下

(下转第 356 页)

[4] 王薇. 分组密码 CLEFIA 与基于四圈 AES 的消息认证码的安全性分析[D]. 济南:山东大学,2009

[5] Tsunoo Y, Tsujihara E, Shigeri M, et al. Impossible differential Cryptanalysis of CLEFA[C]// Kaisa Nyberg. FSE 2008, LNCS 5086. Lausanne, Switzerland: Springer Berlin Heidelberg, 2008: 398-411

[6] 孙兵. 分组密码的分析方法及应用研究[D]. 长沙:国防科学技术大学,2009

[7] Tang X, Sun B, Li R, et al. Impossible differential cryptanalysis of 13-round CLEFIA-128[J]. Journal of Systems and Software, 2011, 84(7): 1191-1196

[8] Mala H, Dakhilalian M, Shakiba M. Impossible differential attacks on 13-round CLEFIA-128[J]. Journal of Computer Science and Technology, 2011, 26(4): 744-750

[9] 吴文玲, 张文涛. 分组密码的设计与分析[M]. 北京:清华大学出版社, 2009: 68-72

[10] 刘青, 卫宏儒. 对完整轮数 ARIRANG 加密模式的新的相关密钥矩形攻击[J]. 计算机科学, 2013, 40(8): 109-114

(上接第 346 页)

的文件进行操作了,之后再编辑 /data/local.prop 文件,将里面的参数 ro.kernel.qemu 的值由 0 改为 1。ro.kernel.qemu 值代表 Android 厂商调试模式,在 Android 设备出厂前厂商需要对设备的各项功能进行调试,因而需要开放 root 权限,将这个值设为 1,出厂后这个值设为 0。ADB Restore 漏洞将重新把这个值设为 1,使手机就会处于调试模式,这样 adb shell 就取得了 root 权限。

在手机 app 中执行了相关的 shell 命令,比如 ln -s /data /data/local/tmp,得到的结果是:Permission denied。证明普通的用户 shell 无法完成这样的操作,所以这个漏洞也只有在 adb 条件下才能利用。

必须通过 adb 工具来利用的漏洞,黑客也可以利用其来进行攻击,利用的关键就是手机 app 必须取得 adb shell 的权限。这样的思路是具有可行性的,现在有很多的无线 adb 工具可以使手机不必通过 USB 线连接至电脑来接受电脑端程序的 adb 调试命令。黑客可以通过无线 adb 的场景,对手机进行调试,触发 root 漏洞,获取最高权限。接下来的工作中,我们会针对这样的攻击场景进行模拟实现,评估由此带来的危害,提出预防这样的攻击的办法。

结束语　在恶意软件利用 root 漏洞进行攻击时,因为 app 是最容易安装进手机的,一般的攻击手段为将 root 漏洞通过重打包的方式装进一个看似无害的 app 中,然后在用户安装后触发实施攻击,所以其在手机上直接触发的漏洞危害性较大,这一点在文献[4]的工作中得到了体现。

越来越多的研究者意识到了 root 漏洞对 Android 系统的安全产生的威胁,但是目前对 Android 系统的漏洞的研究还处于开始阶段。本文详细梳理了现有的 Android root 漏洞,对每个典型漏洞的特点进行了详细的描述,并且依据漏洞的利用方式进行了分类,为进一步开发这些漏洞的检查机制打下了基础。

在制定检测策略的时候,首先要注意的就是那些可以直接在手机上触发、适用的 Android 版本比较新的漏洞,比如 Linux 内核提权漏洞的 Android 平台移植版,以及一些基于 OEM 厂商硬件的漏洞。其次是那些适用 Android 版本比较低,但是利用起来比较方便的漏洞,如 Rageagainstthecage 和 GingerBreak 也要制定针对它们的检测机制,一方面是兼顾低版本 Android 用户,另一方面是防止一些第三方 ROM 在新版本 Android 系统上重新开放这些漏洞。

参 考 文 献

[1] Zhang Q, Li X, Yu X, et al. ASF: Improving Android Security with Layered Structure Instrumentation[M]// Contemporary Research on E-business Technology and Strategy. Springer Berlin Heidelberg, 2012: 147-157

[2] Ongtang M, McLaughlin S, Enck W, et al. Semantically rich application-centric security in Android[J]. Security and Communication Networks, 2012, 5(6): 658-673

[3] Bartel A, Klein J, Le Traon Y, et al. Automatically securing permission-based software by reducing the attack surface: An application to Android[C]// 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2012: 274-277

[4] Zhou Y, Jiang X. Dissecting Android malware: Characterization and evolution[C]// 2012 IEEE Symposium on Security and Privacy (SP). IEEE, 2012: 95-109

[5] Jang W J, Cho S W, Lee H W, et al. Rooting attack detection method on the Android-based smart phone[C]// 2011 International Conference on Computer Science and Network Technology (ICCSNT). IEEE, 2011, 1: 477-481

[6] SEAndroid[OL]. http://selinuxproject.org/page/SEAndroid

[7] Android fragmentation[OL]. http://www.webopedia.com/TERM/F/fragmentation.html

[8] XDA[OL]. http://forum.xda-developers.com

[9] 福布斯中文网[EB/OL]. 201206 恶意软件可借由充电器入侵 iPhone 手机, http://www.forbeschina.com/review/201306/0026176.shtml

[10] KingRoot[OL]. http://www.pc6.com/az/75398.html

[11] SuperOneclick[OL]. http://luozhihao.wodemo.com/file/92858

[12] Z4Root http://forum.xda-developers.com/showthread.php?t=833953

[13] Framaroot[OL]. http://forum.xda-developers.com/showthread.php?t=2130276

[14] Rageagainstthecage [OL]. https://github.com/bibanon/Android-development-codex/wiki/rageagainstthecage

[15] GingerBreak [OL]. http://c-skills.blogspot.com/2011/04/yummy-ymuuy-gingerbreak.html

[16] CVE-2012-0056[OL]. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-0056

[17] CVE-2013-2094[OL]. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-2094

[18] Root exploit on Exynos [OL]. http://forum.xda-developers.com/showthread.php?t=2048511

[19] CVE-2013-2596[OL]. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-2596

[20] Revolutionary-zergRush local root 2.2/2.3. http://forum.xda-developers.com/showthread.php?t=1296916

[21] ADB Restore[OL]. http://forum.xda-developers.com/showthread.php?t=1439429