1. Android介绍

1.1 Android

Android是一种基于Linux的自由及开放源代码的操作系统，主要使用于移动设备，如智能手机和平板电脑，由Google公司和开放手机联盟领导及开发

1.2 发展历程

2003年10月，Andy Rubin等人创建Android公司，并组建Android团队。[3]

2005年8月17日，Google低调收购了成立仅22个月的高科技企业Android及其团队。安迪鲁宾成为Google公司工程部副总裁，继续负责Android项目。

2007年11月5日，谷歌公司正式向外界展示了这款名为Android的操作系统，并且在这天谷歌宣布建立一个全球性的联盟组织，该组织由34家手机制造商、软件开发商、电信运营商以及芯片制造商共同组成，并与84家硬件制造商、软件开发商及电信营运商组成开放手持设备联盟（Open Handset Alliance）来共同研发改良Android系统，这一联盟将支持谷歌发布的手机操作系统以及应用软件，Google以Apache免费开源许可证的授权方式，发布了Android的源代码。[4-5]

2008年，在GoogleI/O大会上，谷歌提出了AndroidHAL架构图，在同年8月18号，Android获得了美国联邦通信委员会（FCC）的批准，在2008年9月，谷歌正式发布了Android 1.0系统，这也是Android系统最早的版本。

2009年4月，谷歌正式推出了Android 1.5这款手机，从Android 1.5版本开始，谷歌开始将Android的版本以甜品的名字命名，Android 1.5命名为Cupcake（纸杯蛋糕）。该系统与Android 1.0相比有了很大的改进。

2009年9月份，谷歌发布了Android 1.6的正式版，并且推出了搭载Android 1.6正式版的手机HTC Hero（G3），凭借着出色的外观设计以及全新的Android 1.6操作系统，HTC Hero（G3）成为当时全球最受欢迎的手机。Android 1.6也有一个有趣的甜品名称，它被称为Donut（甜甜圈）。

2010年2月份，Linux内核开发者Greg Kroah-Hartman将Android的驱动程序从Linux内核“状态树”（“staging tree”）上除去，从此，Android与Linux开发主流将分道扬镳。在同年5月份，谷歌正式发布了Android 2.2操作系统。谷歌将Android 2.2操作系统命名为Froyo，翻译完名为冻酸奶。

2010年10月份，谷歌宣布Android系统达到了第一个里程碑，即电子市场上获得官方数字认证的Android应用数量已经达到了10万个，Android系统的应用增长非常迅速。在2010年12月，谷歌正式发布了Android 2.3操作系统Gingerbread （姜饼）。

2011年1月，谷歌称每日的Android设备新用户数量达到了30万部，到2011年7月，这个数字增长到55万部，而Android系统设备的用户总数达到了1.35亿，Android系统已经成为智能手机领域占有量最高的系统。

2011年8月2日，Android手机已占据全球智能机市场48%的份额，并在亚太地区市场占据统治地位，终结了Symbian（塞班系统）的霸主地位，跃居全球第一。

2011年9月份，Android系统的应用数目已经达到了48万，而在智能手机市场，Android系统的占有率已经达到了43%。继续在排在移动操作系统首位。谷歌将会发布全新的Android 4.0操作系统，这款系统被谷歌命名为Ice Cream Sandwich（冰激凌三明治）。

2012年1月6日，谷歌Android Market已有10万开发者推出超过40万活跃的应用，大多数的应用程序为免费。Android Market应用程序商店目录在新年首周周末突破40万基准，距离突破30万应用仅4个月。在2011年早些时候，Android Market从20万增加到30万应用也花了四个月。

1.3 Android版本



2011.10.19

2012.6.28

2011.2.2

2010.12.74.30

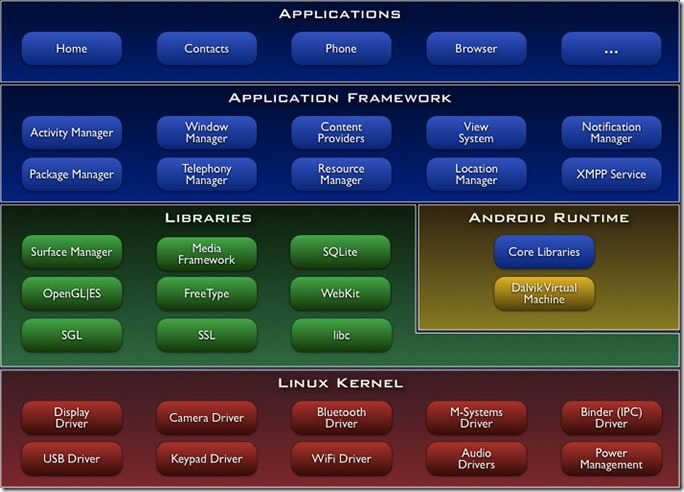
2010.5.20

2009.10.260

2009.9.15

2009.4.30

1.3 Android框架

1.3.1 Linux Kernel

Android基于Linux 2.6提供核心系统服务，例如：安全、内存管理、进程管理、网络堆栈、驱动模型。Linux Kernel也作为硬件和软件之间的抽象层，它隐藏具体硬件细节而为上层提供统一的服务。

如果你学过计算机网络知道OSI/RM，就会知道分层的好处就是使用下层提供的服务而为上层提供统一的服务，屏蔽本层及以下层的差异，当本层及以下层发生了变化不会影响到上层。也就是说**各层各司其职，各层提供固定的SAP（Service Access Point）**，专业点可以说是**高内聚、低耦合。**

如果你只是做应用开发，就不需要深入了解Linux Kernel层。

1.3.2 Android Runtime

Android 包含一个核心库的集合，提供大部分在Java编程语言核心类库中可用的功能。每一个Android应用程序是Dalvik虚拟机中的实例，运行在他们自己 的进程中。Dalvik虚拟机设计成，在一个设备可以高效地运行多个虚拟机。Dalvik虚拟机可执行文件格式是.dex，dex格式是专为Dalvik 设计的一种压缩格式，适合内存和处理器速度有限的系统。

大多数虚拟机包括JVM都是基于栈的，而Dalvik虚拟机则是基于寄存器的。 两种架构各有优劣，一般而言，基于栈的机器需要更多指令，而基于寄存器的机器指令更大。dx 是一套工具，可以將 Java .class 转换成 .dex 格式。一个dex文件通常会有多个.class。由于dex有時必须进行最佳化，会使文件大小增加1-4倍，以ODEX结尾。

Dalvik虚拟机依赖于Linux 内核提供基本功能，如线程和底层内存管理。

1.3.3 Libraries

Android包含一个C/C++库的集合，供Android系统的各个组件使用。这些功能通过Android的应用程序框架（application framework）暴露给开发者。下面列出一些核心库：

* **系统C库**——标准C系统库（libc）的BSD衍生，调整为基于嵌入式Linux设备
* **媒体库**——基于PacketVideo的OpenCORE。这些库支持播放和录制许多流行的音频和视频格式，以及静态图像文件，包括MPEG4、 H.264、 MP3、 AAC、 AMR、JPG、 PNG
* **界面管理**——管理访问显示子系统和无缝组合多个应用程序的二维和三维图形层
* **LibWebCore**——新式的Web浏览器引擎,驱动Android 浏览器和内嵌的web视图
* **SGL**——基本的2D图形引擎
* **3D库**——基于OpenGL ES 1.0 APIs的实现。库使用硬件3D加速或包含高度优化的3D软件光栅
* FreeType ——位图和矢量字体渲染
* SQLite ——所有应用程序都可以使用的强大而轻量级的关系数据库引擎

1.3.4 Application Framework

通过提供开放的开发平台，Android使开发者能够编制极其丰富和新颖的应用程序。开发者可以自由地利用设备硬件优势、访问位置信息、运行后台服务、设置闹钟、向状态栏添加通知等等，很多很多。

开发者可以完全使用核心应用程序所使用的框架APIs。应用程序的体系结构旨在简化组件的重用，任何应用程序都能发布他的功能且任何其他应用程序可以使用这些功能（需要服从框架执行的安全限制）。这一机制允许用户替换组件。

所有的应用程序其实是一组服务和系统，包括：

* **视图（View）**——丰富的、可扩展的视图集合，可用于构建一个应用程序。包括包括列表、网格、文本框、按钮，甚至是内嵌的网页浏览器
* **内容提供者（Content Providers）**——使应用程序能访问其他应用程序（如通讯录）的数据，或共享自己的数据
* **资源管理器（Resource Manager）**——提供访问非代码资源，如本地化字符串、图形和布局文件
* **通知管理器（**Notification Manager**）**——使所有的应用程序能够在状态栏显示自定义警告
* **活动管理器（**Activity Manager**）**——管理应用程序生命周期,提供通用的导航回退功能

1.3.5 Applications

Android装配一个核心应用程序集合，包括电子邮件客户端、SMS程序、日历、地图、浏览器、联系人和其他设置。所有应用程序都是用Java编程语言写的。更加丰富的应用程序有待我们去开发！

第二章 Android Root分析

* 1. Root定义

root存在于UNIX系统（如AIX、BSD等）和类UNIX系统（如Debian、Redhat、Ubuntu以及Android系统等各个发行版的Linux）中，超级用户一般命名为root，相当于Windows系统中的Administrator。root是系统中唯一的超级用户，具有系统中所有的权限，如启动或停止一个进程，删除或增加用户，增加或者禁用硬件等等

* 1. Android Root原理

Andoid的内核是linux,在linux下获取root权限的时候是执行sudo或者su,接下来系统会提示你输入root用户的密码，密码正确就获取了root权限。Android本身就不想让你获取root权限，大部分手机出厂的时候根本就没有su这个程序。所以你想获得android的root权限，第一步就是要把编译好的su文件拷贝到android手机的/system/bin或者/system/xbin目录下，接下来你就可以在android手机的adb shell或串口下输入su,但android里的 su和linux里su是不一样的,android里的su不是靠验证密码，而是看你原来的权限是什么。意思是如果你是root,那你可以通过su切换到别的用户，但是如果你是root以外的用户，就不能切换回root,会提示你permission denied.linux下的文件一般有四种权限，r代表可读，w代表可写，x代表可执行，-代表没有该权限，但是有的文件比较特殊，它的执行权限标志位是一个s,代表当任何一个用户执行该文件的时候,都拥有该文件所有者的权限，如果这个文件的所有者是root,那么，不管谁执行它，都是以root身份执行的

所以，要root手机，需要把一个所有者是root的su拷贝到android手机上，并把su的权限标志位置成-rwsr-xr-x。它需要这两行代码

cp /data/tmp/su /system/bin/ #copy su 到/system/分区  
chown root:root su #su的所有者置成root  
chmod 4775 /system/bin/su #把su置成-rwsr-xr-x

但是执行上面的每一行代码都需要root权限。只有有root权限的情况下才能执行上面两行代码，而这两行代码就是为了让你获得root权限的，这是一个逻辑闭环，那么如何打破这个逻辑闭环呢？一个办法就是找一个本身已经有root权限的进程来启动我上面的两行代码，那我这两行代码一启动就是root权限，就可以顺利执行了。

1. Root漏洞
   1. sendpage漏洞分析

1.1.1 sendpage漏洞

发现时间：2009年

发现者：zinx

编号：CVE-2009-2692

漏洞影响范围：android2.1及android2.1以下

漏洞简介：sock\_sendpage()的空指针解引用。因为sock\_sendpage没有对socket\_file\_ops结构的sendpage字段做指针检查，有些模块不具备sendpage功能，初始时赋为NULL，这样，没有做检查的sock\_sendpage有可能直接调用空指针而导致出错并提升权限！

1.1.2 漏洞原理

一个本地文件通过socket发送出去，过程：打开文件fd和一个socket,然后循环的从文件中read数据，并将读取的数据发送到socket中，这样，每次读写我们都需要两次系统调用，并且数据被从内核拷贝到用户空间(read),再从用户空间拷贝到内核(send),而sendfile就将这个过程封装在一个系统调用中，避免了多次系统调用，和数据在内核空间和用户空间之间大量拷贝。

/\*in 只能是普通文件， out 是 socket\*/

ssize\_t sendfile(int out\_fd, int in\_fd, off\_t \*offset, size\_t count);

写数据到out\_fd过程会调用sock\_sendpage()函数

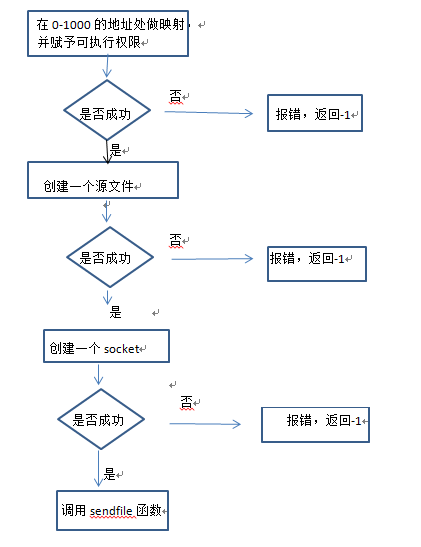
sock\_sendpage的代码如下:

struct socket \*sock;  
int flags;  
sock = file->private\_data;  
flags = !(file->f\_flags & O\_NONBLOCK) ? 0 : MSG\_DONTWAIT;  
if (more)  
     flags |= MSG\_MORE;  
return sock->ops->sendpage(sock, page, offset, size, flags);

调用sock->ops->sendpage 之前没有判断这个函数指针是否为NULL而导致错误并提升权限

1.2.3．漏洞的利用

攻击流程图:



sendfile 函数的调用

unlink(template);   
ftruncate(fdin, PAGE\_SIZE);   
sendfile(fdout, fdin, NULL, PAGE\_SIZE); //调用一个值为NULL的函数指针

sendfile将在系统调用中，cpu进入内核态，触发对0地址的调用，所以，只需要把恶意代码映射到0地址处

1.2.4 漏洞的修补

在调用之前，先判断函数指针是否为空

2.1 udev漏洞分析

2.1.1 udev漏洞

发现时间：2010年

发现者:” The Android Exploid Crew”小组

编号：CVE-2010-EASY

漏洞影响范围：android2.3及以下

2.1.2 udev漏洞原理

由于udev对热插拔消息检测不严，恶意程序可以发送伪热插拔消息，让内核执行自身恶意代码，而内核由于没有检查校区发送是内核还是用户，就执行了恶意程序，从而取得root权限

注：热拔插即带电插拔，热插拔功能就是允许用户在不关闭系统，不切断电源的情况下取出和更换损坏的硬盘、电源或[板卡](http://baike.baidu.com/view/559655.htm)等部件，从而提高了系统对灾难的及时恢复能力、扩展性和灵活性等

vold工作机制：

       kernel检测到硬件事件，然后广播出去，Native层的init进程通知vold(root权限)去处理该广播信息，vold再通知JNI层的MountService,其与Java应用层交互

       vold进程可以将用户态数据copy到内核，由内核写入设备文件（恶意程序的依赖）

       vold进程应该只处理kernel发送的device的NETLINK的socket消息，但实际上并未检测NETLINK的socket消息的来源，

       这样可以广播add device的socket信息，触发硬件处理事件，将恶意代码传入kernel，由其写入设备文件

2.1.3 udev漏洞的利用

构建NETLINK套接字

创建要热拔插文件件

构建热拔插内容，发送给内核

触发漏洞

创建一个NETLINK的套接字，并构建热拔插的信息，里面还含有恶意代码发送给内核，由于内核没有检查消息的发送者是内核还是用户，就以root权限执行了代码，这样就获得了权限提升

2.1.4 udev漏洞的修复

只需要给system/core/init/devices.c文件打个补丁就可以了，具体内容如下

   static int open\_uevent\_socket(void)

 {

   setsockopt(s, SOL\_SOCKET, SO\_PASSCRED, &on, sizeof(on));

//在open\_uevent\_socket对套接字增加一个选项 SO\_PASSCRED，这样可以让套接字增加一个认证，让接收者可以知道发送者的uid和gid

}

void handle\_device\_fd(int fd)

{

   for(;;) {

       char msg[UEVENT\_MSG\_LEN+2];

       char cred\_msg[CMSG\_SPACE(sizeof(struct ucred))];

       struct iovec iov = {msg, sizeof(msg)};

       struct sockaddr\_nl snl;

       struct msghdr hdr = {&snl, sizeof(snl), &iov, 1, cred\_msg, sizeof(cred\_msg), 0};

       ssize\_t n = recvmsg(fd, &hdr, 0);

       if (n <= 0) {

           break;

       }

   while((n = recv(fd, msg, UEVENT\_MSG\_LEN, 0)) > 0) {

       struct uevent uevent;

       if ((snl.nl\_groups != 1) || (snl.nl\_pid != 0)) {

           /\* 如果不是内核的多播信息则抛弃 \*/

           continue;

       }

       struct cmsghdr \* cmsg = CMSG\_FIRSTHDR(&hdr);

       if (cmsg == NULL || cmsg->cmsg\_type != SCM\_CREDENTIALS) {

           /\* 如果发送者的认证没有则抛弃 \*/

           continue;

       }

       struct ucred \* cred = (struct ucred \*)CMSG\_DATA(cmsg);

       if (cred->uid != 0) {

           /\* 消息不是来自于root用户则抛弃 \*/

           continue;

       }

}

}

3.1 adb setuid漏洞

3.1.1 adb setuid漏洞信息

发现时间：2010年

发现者:” The Android Exploid Crew”小组

漏洞编号：无

影响范围：android4.0以下

3.1.2 adb setuid漏洞原理

adb.c中的代码是以root权限运行，以完成部分初始化工作，初始化完成之后，会调用下面代码及setuid()将用户从root切换回shell,但是，setuid()在shell用户进程数达到上限时，会失败，而系统没有检查setuid函数的返回值，因此adb.c继续以root身份运行，没有报错

|  |
| --- |
| android\_src/system/core/adb/adb.c |
| /\* then switch user and group to "shell" \*/  setgid(AID\_SHELL);  setuid(AID\_SHELL);//没有检查返回值  3.1.3 adb setuid漏洞的利用  步骤：  1、在Android的shell用户下，制造大量的僵尸进程，直至达到shell用户的进程数上限RLIMIT\_NPROC；2、kill当前系统中的adb进程，并再次占据其进程位置以保持达到上限；  3、系统会在一段时间后重启一个adb进程，该进程最初是root用户，在完成少许初始化工作后，调用setuid()切换至shell用户；  4、此时shell用户的进程数已经达到上限，所以setuid()失败，返回-1，并且用户更换没有完成，adb还是root权限；  5、adb没有检查setuid()的返回值，继续后续的工作，因此产生了一个具有root权限的adb进程，可被用于与用户的下一步交互。  搜索adb进程pid  fork一个新进程  在子进程中继续不断fork  杀掉adb进程，并占据留下的进程空位  漏洞触发  不断fork子进程(其实是孙进程)，然后退出，产生僵尸进程，使得当前shell用户的pid数达到上限，然后kill掉adbd进程，然后再迅速fork一个，使之仍然达到上限，这样将无法再为当前用户创建进程过段时间，init进程会检测到当前没有adbd进程，会运行adb.c，利用漏洞得到root权限的adbd进程  3.1.4 adb setuid漏洞的修复 |

/\* then switch user and group to "shell"

if (setgid(AID\_SHELL) != 0) {

exit(1);

}

if (setuid(AID\_SHELL) != 0) {

exit(1);}

增加一个判断，如果返回值不为0，就退出

4.1 zergRush漏洞分析

4.1.1 zergRush漏洞

发现时间:2011年

发现者：Revolutionary工具开发小组

编号：CVE-2011-3874

漏洞影响范围：android3.0及以下

4.1.2 漏洞原理

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，栈变量argv[FrameworkListener::CMD\_ARGS\_MAX]由于允许的最大下标为16，如果我们特意传送超过 16 个空格分割的字符串，函数就会溢出，因此可以在root权限执行输入的shellcode

注：Shellcode实际是一段代码（也可以是填充数据），是用来发送到服务器利用特定漏洞的代码，一般可以获取权限

4.1.3 漏洞利用

步骤：

1. 计算出vold的堆地址

2. 查到system调用的地址

3. 尝试出栈缓冲区大小

4. 通过崩溃产生的调试信息，取得栈地址和栈结构信息

5. 在libc.so中找寻跳板指令

6. 根据缓冲区大小、栈结构和上述各种地址，构造出有效的shellcode来，发送到vold

7. shellcode在vold中以root权限运行，它通过system调用运行该利用程序的一个副本boomsh

8. 程序副本boomsh以root权限运行时，会置上shell程序的S位，并设置系统属性ro.kernel.qemu

9. 结束掉adb，后续开启的adb进程将具有root权限

利用了android中几个特殊之处：

1. vold的溢出会在adb logcat中输出调试信息，这些信息说明了其内存结构，而其他程序可以读取到这些信息；

2. 在ARM架构下，跳板指令有了更多的选择，ret2libc的攻击也可能更容易实现

3. adb的降低权限过程又一次被利用。

5.1 psneuter漏洞分析

5.1.1 psneuter漏洞

发现时间：2011年

发现者：scotty2

漏洞编号：[CVE-2011-1149](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-1149)

漏洞影响范围：android3.0及以下

5.2.2 psneuter漏洞原理

ADB 是 Android 手机中的一个调试服务 是由 ROOT [程序](http://www.xuebuyuan.com/)派生的，

看 adb.c 中有一样一段[代码](http://www.xuebuyuan.com/" \t "_blank" \o "代码)

    property\_get("ro.kernel.qemu", value, "");

    if (strcmp(value, "1") != 0) {

        property\_get("ro.secure", value, "");

        if (strcmp(value, "1") == 0) {

            // don't run as root if ro.secure is set...

            secure = 1;

            // ... except we allow running as root in userdebug builds if the

            // service.adb.root property has been set by the "adb root" command

            property\_get("ro.debuggable", value, "");

            if (strcmp(value, "1") == 0) {

                property\_get("service.adb.root", value, "");

                if (strcmp(value, "1") == 0) {

                    secure = 0;

                }

            }

        }

    }

只要让 secure 不等于1，就是以 root 运行，否则就是以 shell 运行,secure 默认值是为 0 的，看上面代码，property\_get 是 Android 中属性服务，用的是共享内存，因为在这里没有加上合适的判断，可以让这片内存设定不可读，按上面的代码逻辑，ADB 也就成 ROOT 用户了。所以只需要找到这片内存，并将secure的值设为0

### 6.1 GingerBreak漏洞分析

### 6.1.1GingerBreak漏洞

### 发现时间：2011年

### 发现者: “The Android Exploid Crew”小组

### 编号: [CVE-2011-1823](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-1823)

### 7.1 [mempodroid漏洞分析](http://blog.csdn.net/mldxs/article/details/14497719)

### 7.1.1 mempodroid漏洞

### 发现时间：2012年

### 发现者：zx2c4

### 编号：CVE-2012-0056

### 