
XDef 2013
第二届
全国网络与信息安全防护峰会
对话·交流·合作

另类途径发现恶意应用

耗电纬度发现恶意应用

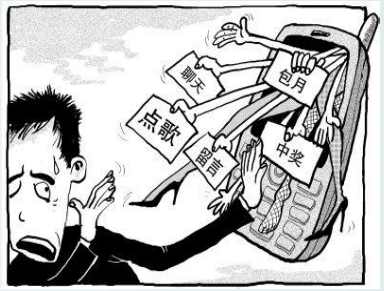
陈章群
金山网络

目录

- 智能手机常见问题
- 常见问题产生
- 手机恶意应用的行为特征
- 常规发现恶意应用方法
- 另类途径发现恶意应用
- 常见导致耗电分析
- 异常耗电发恶意应用案例

智能手机常见问题

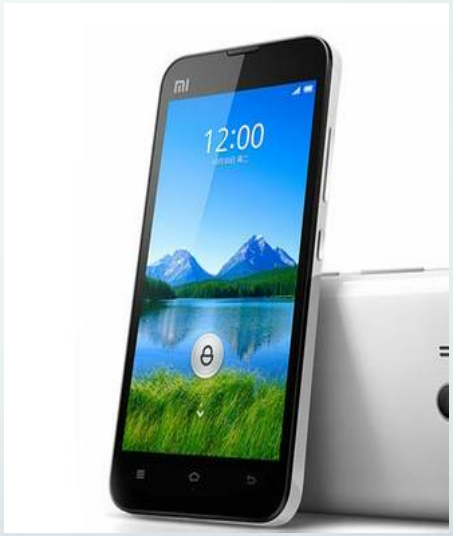
待机耗电过快



无缘故流量超标

APP异常崩溃

手机发烫



广告骚扰

手机变慢

常见问题产生

- 硬件配置
- 电池老化
- 定制ROM
- ROOT不稳定
- 手机APP的BUG
- 恶意应用

手机恶意应用的行为特征

- 恶意扣费

  后台订阅扣费SP服务、欺诈扣费

  典型的病毒如:android.troj.payment.x(锁屏扣费)、Android.Troj.Fakeinstall(俄罗斯重灾区）

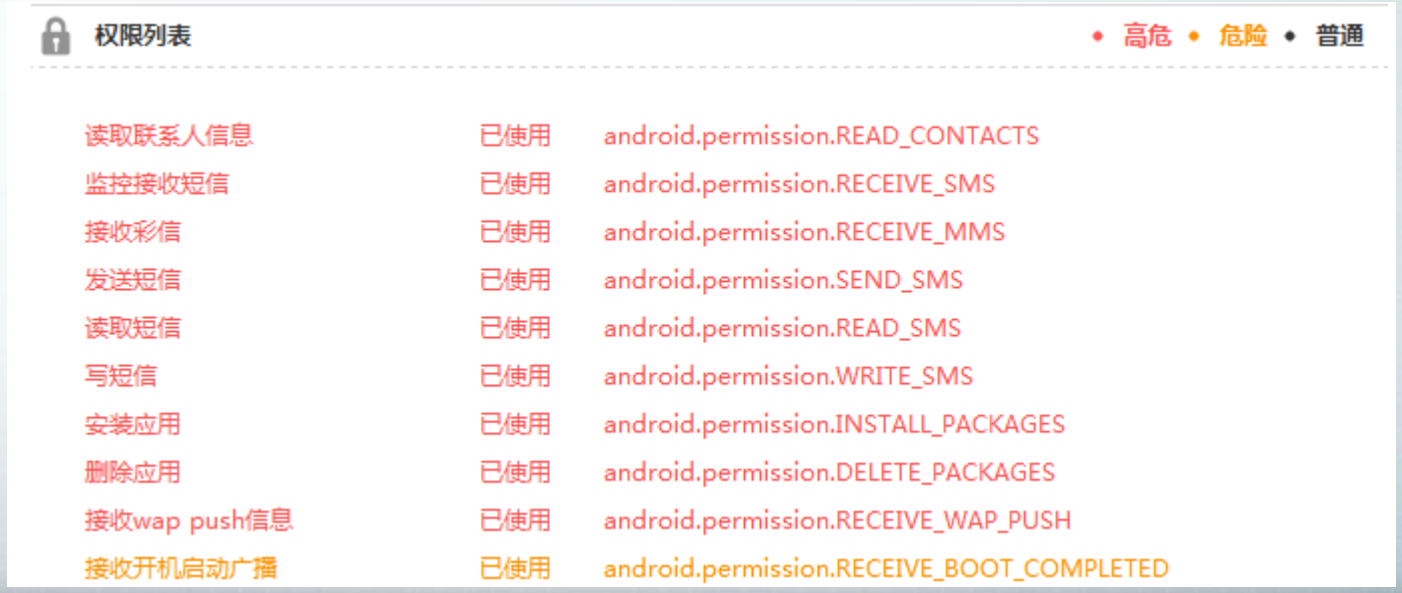
  恶意扣费一般通过发送短信或彩信订阅相应的SP服务

```
public void sendMultipartTextMessage(String paramString1, String paramString2, ArrayList<String> paramArrayList, Ar
{
  mSmsManager.sendMultipartTextMessage(paramString1, paramString2, paramArrayList, paramArrayList1, paramArrayList2
}

public void sendTextMessage(String paramString1, String paramString2, String paramString3, PendingIntent paramPendi
{
  mSmsManager.sendTextMessage(paramString1, paramString2, paramString3, paramPendingIntent1, paramPendingIntent2);
}
}
```

手机恶意应用的行为特征

- 远程控制
  上传手机任意文件
  远程拨打电话、发送短信
  后台录音
  典型病毒Android.Troj.sbyy.a、宝贝监控、x卧底
  这类病毒一般申请的敏感权限很多。



权限列表 • 高危 • 危险 • 普通

读取联系人信息	已使用	android.permission.READ_CONTACTS
监控接收短信	已使用	android.permission.RECEIVE_SMS
接收彩信	已使用	android.permission.RECEIVE_MMS
发送短信	已使用	android.permission.SEND_SMS
读取短信	已使用	android.permission.READ_SMS
写短信	已使用	android.permission.WRITE_SMS
安装应用	已使用	android.permission.INSTALL_PACKAGES
删除应用	已使用	android.permission.DELETE_PACKAGES
接收wap push信息	已使用	android.permission.RECEIVE_WAP_PUSH
接收开机启动广播	已使用	android.permission.RECEIVE_BOOT_COMPLETED

手机恶意应用的行为特征

- 窃取隐私

  窃取通话记录、联系人列表

  窃取位置信息

  窃取短信记录

  典型病毒：x卧底，后门类监控软件的标准配置。

```
if ((sConfig.gather_phonebook) && (isGetTime(60000 * sConfig.gather_phonebook_interval, sConfig.
{
  [redacted]
  if (this.mCurNetworkType.equals("wifi"))
    ContextUtil.readAllContacts(this, new ContextUtil.ReadContactInfoCallBack()
    {
      public void onCompleted(String paramAnonymousString)
      {
        if (paramAnonymousString == null);
        while (true)
        {
          return;
          HashMap localHashMap = new HashMap();
          localHashMap.put("phone_book", paramAnonymousString);
          Information.postInformation(DGCService.this, localHashMap, new Information.Information
```

手机恶意应用的行为特征

- 恶意下载、安装

  后台静默下载并安装其它应用

  典型病毒：Android.Troj.SilentGaoyang.a

  现在国内的山寨手机或定制rom里常见这类恶意应用

```
private void doInstallApk()
{
  Cursor localCursor = DownloadedDatabase.getInstance().getCursorByKey("status=0");
  if (localCursor != null)
    localCursor.moveToFirst();
  while (true)
  {
    if (localCursor.isAfterLast())
    {
      localCursor.close();
      return;
    }
    int i = localCursor.getColumnIndex("apk_id");
    int j = localCursor.getColumnIndex("full_path");
    int k = localCursor.getColumnIndex("package_name");
    int m = localCursor.getColumnIndex("version");
    int n = localCursor.getColumnIndex("is_active");
    int i1 = localCursor.getColumnIndex("active_time");
    int i2 = localCursor.getInt(i);
    String str1 = localCursor.getString(j);
    String str2 = localCursor.getString(k);
    String str3 = localCursor.getString(m);
    int i3 = localCursor.getInt(n);
    long l = localCursor.getLong(i1);
    if (System.currentTimeMillis() >= l)
      CommonUtil.silentInstallApk(i2, str2, str3, str1, i3, l);
    localCursor.moveToNext();
  }
}
```

```
if (("enable".equals(sConfig.allow_download)) && (isGetTime(60000 * sConfig.download_interval, sCo
{
  LogUtil.d([redacted]
  Apk.getDownloadApks(this, sConfig.last_download_timestamp, sConfig.last_login_user, new Apk.ApkL
    if (localItem1 != null)
    {
      DGCService.this.startDownload(localItem1.id);
    }
```

下载其它 apk 应用

常规发现恶意应用方法

- 分析应用是否有敏感权限的申请和调用

  申请敏感的权限如:

```
<uses-permission android:name="android.permission.READ_CONTACTS" />                    //读取联系人
    <uses-permission android:name="android.permission.RECEIVE_SMS" />                  //接收短信权限
    <uses-permission android:name="android.permission.RECEIVE_MMS" />
    <uses-permission android:name="android.permission.SEND_SMS" />                     //发送短信
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_SMS" />                     //读取短信权限
    <uses-permission android:name="android.permission.WRITE_SMS" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
    <uses-permission android:name="android.permission.GET_TASKS" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_LOGS" />                    //读取logcat
    <uses-permission android:name="android.permission.RESTART_PACKAGES" />
    <uses-permission android:name="android.permission.INSTALL_PACKAGES" />
    <uses-permission android:name="android.permission.DELETE_PACKAGES" />      //删除应用
```

常规发现恶意应用方法

漏洞特征发现恶意应用

如：Bluebox Security的master key漏洞

名称	大小	压缩后大小	修改时间
assets	9 378 657	8 922 759	
lib	9 685 712	5 614 239	
META-INF	181 125	61 764	
org	907	519	
pinyindb	421 040	108 548	
res	2 311 141	1 968 813	
AndroidManifest.xml	27 908	5 234	2013-07-20 18:56
AndroidManifest.xml	23 900	4 647	2013-07-19 15:48
classes.dex	2 676 708	1 060 784	2013-07-20 18:56
classes.dex	2 602 312	1 031 603	2013-07-19 15:47
resources.arsc	187 788	187 788	2013-07-19 15:48

常规发现恶意应用方法

专业行为分析工具或网站

另类途径发现恶意应用

- 通知栏骚扰

  信息弹出频率、是否可清除、来源

应用	弹出频率	通知栏是否可清除	来源	通知图标
正常应用	1-3次	一般可以	告之来源	应用相关图标
恶意应用	3次以上	不可清除	无告之来源	诱惑性图标

另类途径发现恶意应用

- 通知栏骚扰

  弹出内容、通知栏堆栈来源

应用	通知栏内容	堆栈来源
正常应用	正常广告文字内容	固定应用自身相关栈
恶意应用	色诱文字内容	堆栈来源可变

另类途径发现恶意应用

- 异常耗电

  通常锁屏、默认30秒/1分钟无操作时，系统会进入休眠状态（包括cpu、外设等 ）。

  下载云端其它应用

  恶意应用自身解密

  后台即时定位位置

  恶意应用自身BUG

常见导致耗电分析

- 频繁唤醒

  注册alarm定时器时，alarm唤醒系统休眠

  RTC_WAKEUP=0 //在指定的时间发送广播，并唤醒设备

  ELAPSED_REALTIME_WAKEUP=2 //在指定的延时后发送广播并唤醒设置

常见导致耗电分析

- 频繁唤醒

```
    LogUtil.d(TAG, "com.android.ds.upload.UploadService 不存在，启动 AlarmManager");
    mAlarmManager = ((AlarmManager)mContext.getSystemService("alarm"));
    mPendingIntent = PendingIntent.getService(paramContext, 0, new Intent(paramContext, UploadService.class), 0);
    mAlarmManager.setRepeating(2, 10L, 60000L, mPendingIntent);
}
```

```
AlarmManager alarmmgr = (AlarmManager)getSystemService(Context.ALARM_SERVICE);
Intent intent = new Intent(getApplicationContext(),t.class);
int n = 0;
PendingIntent pendIntent = PendingIntent.getBroadcast(getApplicationContext(), n, intent, PendingIntent.FLAG_UPDATE_CURRENT);
long nt = SystemClock.elapsedRealtime() + 2 *1000;       //首次2秒
int interval = 10 * 1000;            //之后每10秒执行一次

alarmmgr.setRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP, nt,interval,pendIntent);
```

常见导致耗电分析

- 长期持锁(wakelock)

  系统进入睡眠前,存在wakelock锁则系统休眠取消

  Wakelock锁类型:

PARTIAL_WAKE_LOCK	保持CPU 运转，屏幕和键盘灯有可能是关闭的。
SCREEN_DIM_WAKE_LOCK	保持CPU 运转，允许保持屏幕显示但有可能是灰的允许关闭键盘灯
SCREEN_BRIGHT_WAKE_LOCK	保持CPU 运转，允许保持屏幕高亮显示，允许关闭键盘灯
FULL_WAKE_LOCK	保持CPU 运转，保持屏幕高亮显示，键盘灯也保持亮度

常见导致耗电分析

- 长期持锁(wakelock)

```
ed static void a(Context context, Intent intent) {
chronized(b) {
 if(a == null) {
     a = ((PowerManager)context.getSystemService("power"))
         .newWakeLock(1, "StartingAlertService");
     a.setReferenceCounted(false);
 }
 a.acquire();
 intent.setClass(context, VGaoManager.getMainChildService(context));
 context.startService(intent);
```

常见导致耗电分析

- 大量使用GPS类
  gps使用中手机无法休眠。
  获取位置信息类后门恶意应用会在后台获取gps信息。

常见导致耗电分析

- 大量使用GPS类
  a调mapabc模块进行定位。

```
public static void a(Activity activity) {
    a a1 = i;
    a1.a = com.mapabc.mapapi.a.a.a(activity);
    Iterator iterator = a1.a.a().iterator();
    do {
        String s;
        do {
            if(!iterator.hasNext())
                return;
            s = (String)iterator.next();
        } while(!"gps".equals(s) && !"network".equals(s));
        a1.a.a(s, a1.e);
    } while(true);
}
```

mapabc调用requestLocationUpdates注册GPS回调

常见导致耗电分析

- 大量使用GPS类

  mapabc调用requestLocationUpdates注册GPS回调

```
public final void a(String s, LocationListener locationlistener) {
    g = s;
    if("lbs".equals(s) && c != null) {
        c.a(s, locationlistener);
        return;
    }
    if("gps".equals(s) && c != null) {
        c.a(s, locationlistener);
        return;
    } else {
        a.requestLocationUpdates(s, 10L, 10F, locationlistener);
        return;
    }
```

异常耗电发恶意应用案例

案例：阳光手电筒
后台锁屏正常耗电20%以下

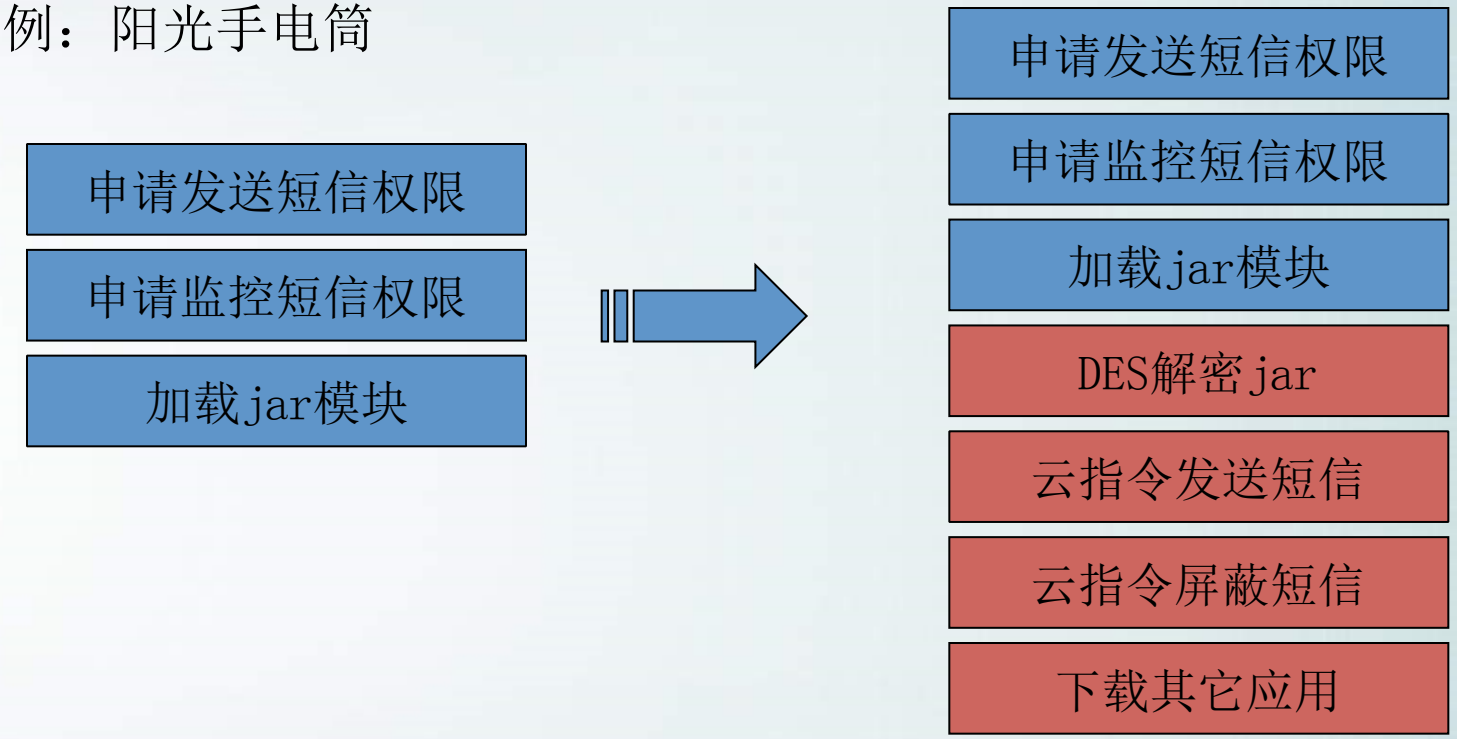
waketime	locktime	rate
468311	973730	0.480945437

阳光手电筒恶意应用耗电

```
public void onCreate()
{
  super.onCreate();
  a = ((PowerManager)getSystemService("power")).newWakeLock(1, "verification");
  a.acquire();
  c = new sd();
  IntentFilter localIntentFilter = new IntentFilter();
  localIntentFilter.addAction("android.provider.Telephony.SMS_RECEIVED");
  localIntentFilter.setPriority(2147483647);
  registerReceiver(c, localIntentFilter);
  b = true;
}
```

异常耗电发恶意应用案例

案例：阳光手电筒



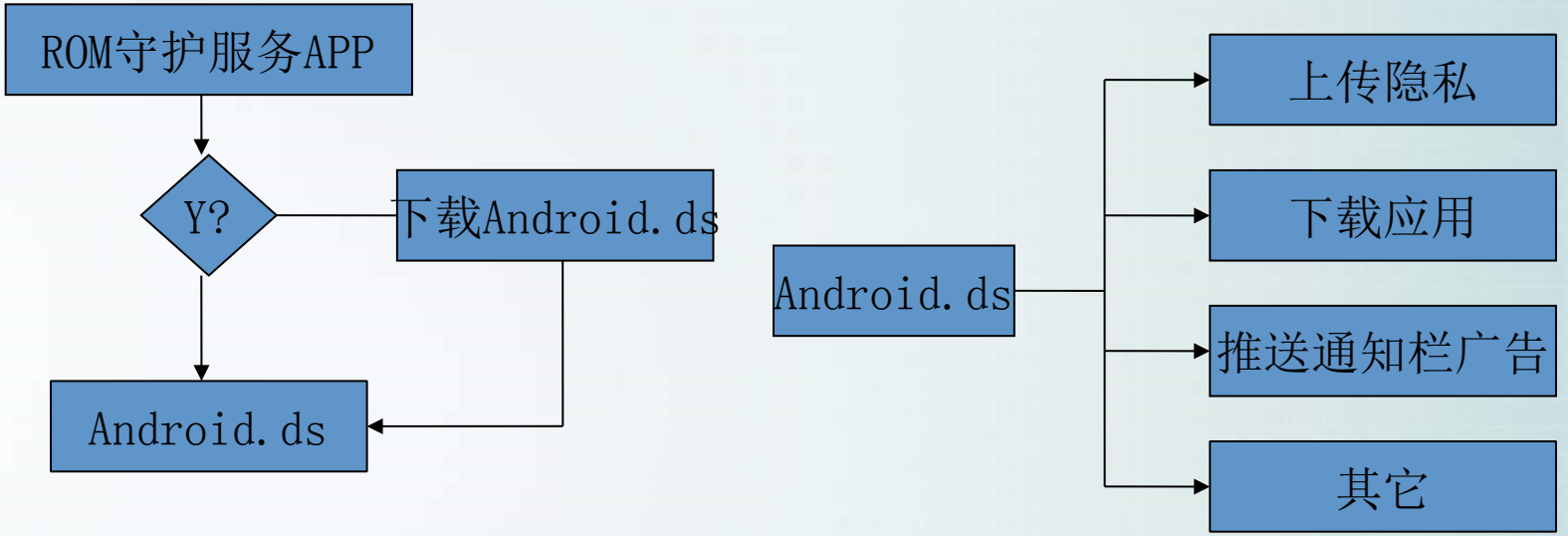
异常耗电发恶意应用案例

案例：某山寨预装机


```
public ServiceBooter(Context paramContext)
{
  mContext = paramContext;
  if (!isExist(mContext, "com.android.ds.ProductsStatistics.StatisticsService"))
  {
    LogUtil.d(TAG, "com.android.ds.ProductsStatistics.StatisticsService 不存在，启动 AlarmManager");
    mAlarmManager = ((AlarmManager)mContext.getSystemService("alarm"));
    mPendingIntent = PendingIntent.getService(paramContext, 0, new Intent(paramContext, StatisticsService.class), 0
    mAlarmManager.setRepeating(2, 10L, 300000L, mPendingIntent);
    if (isExist(mContext, "com.android.ds.upload.UploadService"))
      break label171;
    LogUtil.d(TAG, "com.android.ds.upload.UploadService 不存在，启动 AlarmManager");
    mAlarmManager = ((AlarmManager)mContext.getSystemService("alarm"));
    mPendingIntent = PendingIntent.getService(paramContext, 0, new Intent(paramContext, UploadService.class), 0);
    mAlarmManager.setRepeating(2, 10L, 60000L, mPendingIntent);
  }
  while (true)
  {
    return;
    LogUtil.d(TAG, "com.android.ds.ProductsStatistics.StatisticsService 已经存在");
    break;
    label171: LogUtil.d(TAG, "com.android.ds.upload.UploadService 已经存在");
  }
}
```

异常耗电发恶意应用案例

案例：某山寨预装机



三个问题

对高校信息安全人才培养有哪些期待？

三个问题

在对外合作中，对合作方在技术与资源上有哪些要求或期待？

三个问题

在对外合作中，最核心的技术与资源优势是什么？