落落无尘

闲看庭前花开花落

支付宝后台不死的黑科技

By Jezhee

发表于 2015-07-28

近期支付宝升级到了9.0，除了加入微信功能外，还新增了金钟罩功能，不要说普通的各种加速球，火箭神马的不能杀死他外，连格外牛逼的绿色守护也奈何不了他。在绿色守护里面，支付宝始终显示为“服务正在被 支付宝 使用”，不会自动休眠，恼火死了。那么问题来了，他是怎么做到的呢？

文章目录

支付宝干了啥？

这就是支付宝在绿色守护里的状态，它一直处在不会被自动休眠的这个category里面。不要问我第一个是啥，后面会有。

sneaky_alipay

我们知道，Android有一个oom的机制，系统会根据进程的优先级，给每个进程一个oom权重值，当系统内存紧张时，系统会根据这个优先级去选择将哪些进程杀掉，以腾出空间保证更高优先级的进程能正常运行。要想让进程长期存活，提高优先级是个不二之选。这个可以在adb中，通过以下命令查看：

```
su cat /proc/pid/oom_adj
```

这个值越小，说明进程的优先级越高，越不容易被进程kill掉。通常如果是负数，表示该进程为系统进程，肯定不会被杀掉，如果是0，表示是前台进程，即当前用户正在操作的进程，除非万不得已，也不会被杀掉，1则表示是可见进程，通常表示有一个前台服务，会再通知栏有一个划不掉的通知，比如放歌，下载文件什么的。再增大，则优先级逐渐降低，顺序为服务进程，缓存进程，空进程等等。

因此首先，我们来查看支付宝进程的oom_adj。不出意料，其值很高：1。但这又在意料之外，因为支付宝既没有添加系统服务，也没有常驻通知栏，也没有显示弹窗啊。不过既然情况已经这样了，那我们还是看一下他的服务状态吧：

```
shell@NX505J:/ $ dumpsys activity services com.eg.android.AlipayGphone

* ServiceRecord{17d6e8a5 u0 com.eg.android.AlipayGphone/com.alipay.android.launcher.service.LauncherService$InnerService}
    intent={cmp=com.eg.android.AlipayGphone/com.alipay.android.launcher.service.LauncherService$InnerService}
    packageName=com.eg.android.AlipayGphone
    processName=com.eg.android.AlipayGphone
    baseDir=/data/app/com.eg.android.AlipayGphone-2/base.apk
    dataDir=/data/data/com.eg.android.AlipayGphone
    app=ProcessRecord{38385c98 14176:com.eg.android.AlipayGphone/u0a66}
    isForeground=true foregroundId=168816881 foregroundNoti=Notification(pri=0 contentView=com.eg.android.AlipayGphone/0x1090077 vibrate=null sound=null defaults=0x0 flags=0x40 color=0x1
    createTime=-10m10s188ms startingBgTimeout=-18m55s186ms
    lastActivity=-10m10s188ms restartTime=-10m10s188ms createdFromFg=false
    startRequested=true delayedStop=false stopIfKilled=true callStart=true lastStartId=1
```

原来真有一个前台服务！

这不科学！Service的startForeground函数是要提供一个Notification参数的啊。查资料麻烦，还不如直接反编译支付宝的代码来得快。知道了这个service的名字，我们就可以直捣黄龙了。在LauncherService里面，我们找到了如下代码：

```
void startForegroundCompat()
{
  if (this.mIsForeground)
    return;
  try
  {
    LoggerFactory.getTraceLogger().debug(TAG, "LauncherService.startForegroundCompat: " + Build.VERSION.SDK_INT);
    if (Build.VERSION.SDK_INT < 18)
    {
      startForeground(168816881, new Notification());
      this.mIsForeground = true;
      return;
    }
    Intent localIntent = new Intent(AlipayApplication.getInstance().getApplicationContext(), LauncherService.InnerService.class);
    localIntent.putExtra("NotificationID", 168816881);
    AlipayApplication.getInstance().getApplicationContext().startService(localIntent);
    return;
  }
  catch (Exception localException)
  {
  }
}
```

✎ sneaky_foreground_service

这是一段开启前台服务的代码，针对不同api，18一下直接简单粗暴的给了一个空的

notification，并没有用到上面抓到的InnerService。当然，打开adb看一看，18一下确实就LauncherService自己在负责前台。18以上，则另行处理。试了试，18一下这么干，还真的可以。在用这个老版本手机的时候，一并发现原来好多的应用都是这么干的，什么豌豆荚啦，369手机助手神马的，好吧，算我孤陋寡闻了。但18以上的这段代码没有看太明白，于是，想打开他的InnerService看看到底有什么鬼。然而让我失望的是，InnerService里面并没有开启前台服务的代码，而是把主流程给隐藏在了onStartCommand的一个包裹函数中。这里他用到了aspectj框架，在onStartCommand上做了一个切点，那么具体开启前台服务的流程肯定就在这个切点里面了。但是，但是，尼玛的，我看不懂aspectj啊，跟也跟不过去。没办法，只好绕回来，去google了。幸好幸好，让我找到了方法，看上去，也是和支付宝的思路是一样，

android的漏洞在哪？

原来，google在4.3的时候终于发现有太多流氓应用，为了提高优先级，开启了前台服务，但又不想让用户知道，因此需要不显示通知栏。于是找到了android在开启前台服务时候的一个漏洞，给一个非法的Notification参数，让系统先把ServiceRecord里面的isForeground标记给打上，然后等到显示通知栏时，发现参数并不合法，android于是什么也不干，就不显示通知栏提醒了。

当一票app都这么干的时候，google觉得不能忍了，于是在`ServiceRecord`的`postNotification`函数里面里面，我们看到了如下代码：

```
try {
    if (localForegroundNoti.icon == 0) {
        // It is not correct for the caller to supply a notification
                icon, but this used to be able to slip through, so for
                those dirty apps give it the app's icon.
        localForegroundNoti.icon = appInfo.icon;

        // Do not allow apps to present a sneaky invisible content view either.
        localForegroundNoti.contentView = null;
        localForegroundNoti.bigContentView = null;
        CharSequence appName = appInfo.loadLabel(
                        ams.mContext.getPackageManager());
        if (appName == null) {
            appName = appInfo.packageName;
        }
        ctx = null;
        try {
            ctx = ams.mContext.createPackageContext(
                appInfo.packageName, 0);
        Intent runningIntent = new Intent(
            Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
        runningIntent.setData(Uri.fromParts("package",
            appInfo.packageName, null));
        PendingIntent pi = PendingIntent.getActivity(ams.mContext, 0,
            runningIntent, PendingIntent.FLAG_UPDATE_CURRENT);
        localForegroundNoti.setLatestEventInfo(ctx,
            ams.mContext.getString(
                com.android.internal.R.string
                                        .app_running_notification_title,
                        appName),
        ams.mContext.getString(
             com.android.internal.R.string
                                .app_running_notification_text,
             appName),
            pi);
    } catch (PackageManager.NameNotFoundException e) {
        localForegroundNoti.icon = 0;
    }
}
```

里面可以看到，android再也不让你欺骗系统了，是前台服务就必须要显示一个通知栏提醒。于是，像豌豆荚，360这些，在高版本系统上也没有前台服务了，默默的再后台，然后等着被干掉。

既然如此，那支付宝是怎么做到的呢？真实应了那句话，流氓不可怕，就怕流氓有文化。大门给关上了，那我就翻窗户吧。

在启动前台服务的时候一定要给一个Notification，那我可不可以在启动前台服务后，调用NotificationManager的cancelNotification把这个Notification给取消掉呢？来看看源码：

```
// 在NotificationManagerService.java中
public void cancelNotificationWithTag(String pkg, String tag, int id, int userId) {
        checkCallerIsSystemOrSameApp(pkg);
        userId = ActivityManager.handleIncomingUser(Binder.getCallingPid(),
                Binder.getCallingUid(), userId, true, false, "cancelNotificationWithTag", pkg);
        // Don't allow client applications to cancel foreground service notis.
        cancelNotification(pkg, tag, id, 0,
                Binder.getCallingUid() == Process.SYSTEM_UID
                ? 0 : Notification.FLAG_FOREGROUND_SERVICE, false, userId);
}

// 实现函数的声明，如果在mustNotHaveFlags里面，请求会被直接弹掉，通知不会被取消。
boolean cancelAllNotificationsInt(String pkg, int mustHaveFlags,
            int mustNotHaveFlags, boolean doit, int userId) {
            ...
        }
```

从中间可以看到，对于不是系统UID的用户来说，如果Notification带有FLAG_FOREGROUND_SERVICE标记，你是取消不掉的，这类通知只有系统能取消，而给用户的唯一入口就是`Service#stopForegroundService`，在调用这个接口的时候，会将前台状态置为false，如果传入参数为true，再同时把Notification给cancel掉。看似没有漏洞，然而聪(jian)明(zha)的人还是很多很多的。stop是单向的先取消前台状态，然后cancel notification。也就是说，cancel的时候就不会再去修改service的状态了。所以，假如我同时启动*两个*service，并且将他们同时置为前台状态，然后*共享同一个NotificationID*，也是就说，此时会出现两个前台服务，但通知管理器里只有一个关联的通知。 这时我们在其中一个服务中调用`stopForeground(true)`，这个服务前台状态会被取消，同时状态栏通知也被移除。 `*BUTTTTT！` * 另外一个服务呢？他并没有受到影响，还是前台服务状态，但是此时，状态栏通知已经没了！ 这就是支付宝的黑科技。

下面是一个示例程序，Service名字就叫AlipayService啦: sneaky_foreground_service. 第一个图里面那个和支付宝一起不会被干掉的就是它了。支付宝的service因为还被其他进程绑定，所以显示 服务正在被 支付宝 使用，而这个Circles纯粹是演示用，就懒得整了。

Android 黑科技

上一篇：
ScrollView嵌套GridView的处理
下一篇：
基于Github pages使用hexo建立本博客的步骤记录

标签

Android 4 UI 1 黑科技 1 github 1 hexo 1 陷阱 1 IM 1

友情链接

RSS 订阅