# Android 中 Xposed 框架篇—利用 Xposed 框架实现拦截系统方法 | 尼古拉斯. ...

> 一、前言 关于 Xposed 框架相信大家应该不陌生了，他是 Android 中 Hook 技术的一个著名的框架，还有一个框架是 CydiaSubstrate，但是这个框架是收费的，而且个人觉得不怎么好用，而 Xpos.....

关于 Xposed 框架相信大家应该不陌生了，他是 Android 中 Hook 技术的一个著名的框架，还有一个框架是 CydiaSubstrate，但是这个框架是收费的，而且个人觉得不怎么好用，而 Xposed 框架是免费的而且还是开源的，网上也有很多文章介绍了 Xposed 框架的原理实现，不了解的同学可以自行查阅即可，本文主要介绍如何通过这个框架来进行系统方法的拦截功能，比如我们开发过程中，对于一些测试环境很难模拟，特别是测试同学有时候像随机改变设备的 imei,mcc 等信息用来模拟不同测试场景，这时候如果可以去修改系统的这个值的话对于测试来说就非常方便了，其实这些在网上已经有很多类似的小工具了，下面就来详细的讲解如何使用这个框架。

在介绍如何使用这个框架之前，咋们得先解决这几个问题：

第一个问题：首先我们知道这个框架的核心点就是系统进程注入技术，那么如果要注入系统进程，就必须要 root 权限，所以你如果想用这个框架的话就必须得现有一个 root 的设备。

第二个问题：还有一个问题就是这个框架的适配问题，不是所有的设备所有的系统都支持这个框架的使用的，本人在实

验的过程中就遇到了小米 3+MIUI7 就操作失败了，结果重新刷了一个 Android 原生 4.4 系统才成功的。

第三个问题：最后一个问题就是 Xposed 框架本身的版本问题，他针对不同系统也发布了多个版本，所以你得针对于自己的设备系统安装正确的 Xposed 版本。

解决了这三个问题咋们才能成功的安装 Xposed 框架的，而在这个过程中我们会发现遇到这两个问题是最多的：

第一个问题是不兼容问题：


第二个问题是提示安装框架问题：


这两个问题都是比较普遍和蛋疼的，因为底下的安装按钮点击不了，后续没办法操作了，所以很无助的，我也是遇到了这两个问题，最后也是没有找到合适的答案，所以一激动就刷了一个原生的 Android4.4 系统，

上面就提到了现阶段这个框架使用会遇到的一些问题，下面在来看一下具体的环境搭建，如果上面的问题都解决了，咋们在打开应用点击安装框架：


这里还是提示未激活，点击进入：

这时候看到了正常了，可以点击安装了，直接点击安装即可：

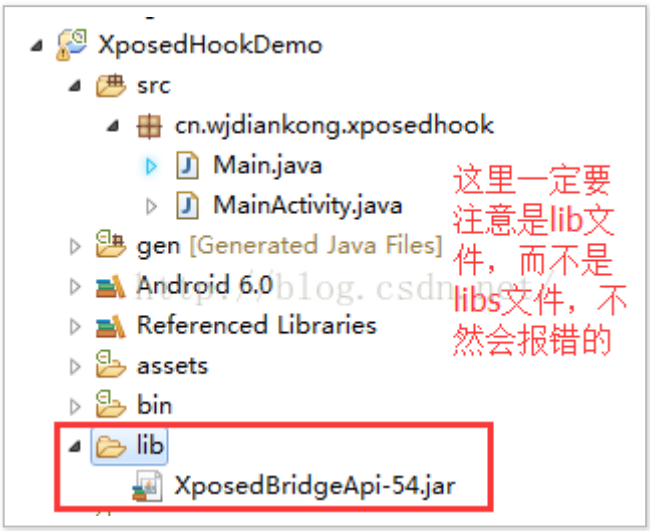

这里是需要 root 授权的，点击允许，安装成功之后也会提示你重启生效的，因为要注入系统进程，必须重启才有效果的。

到这里我们就成功的安装了 Xposed 框架了，在这个过程中肯定有同学会遇到问题，而最多的问题就是上面提到的那两个问题，关于解决办法我也没有找到。我解决的最根本办法就是刷机了，所以本文我操作的环境是：

小米 3 移动版 + Android 原生 4.4 系统 + Xposed_v33 版本

环境搭建好了，下面就开始操作了，上面安装的那个工具其实是一个模块管理器，我们如果想做一些 hook 操作还得自己编写模块也就是应用程序，然后把这个模块安装到设备中，这个工具就可以检测出来了，会提示你加载这模块然后在重启设备，模块功能就有效果了。那么下面来看一下如何编写一个 Xposed 模块呢?

第一步：新建一个 Android 项目，导入 Xposed 工具包




这里一定要注意，不能使用 libs 文件夹而是 lib 文件夹，如果这里使用了 libs 文件夹的话，在安装成功模块之后重启会发现 Hook 是失败的，通过打印 tag 为 xposed 的日志信息会发现这样的错误：

java.lang.IllegalAccessError: Class ref in pre-verified class resolved to unexpected implementation

这个错误我们在以前开发插件的时候遇到过，主要是因为把接口包含到了插件工程中了，那么这里我们可以猜想错误问题也是这个 xposed 工具导致的。那么我们只需要把 libs 文

件夹改成 lib，然后在 add buildpath 一下即可。

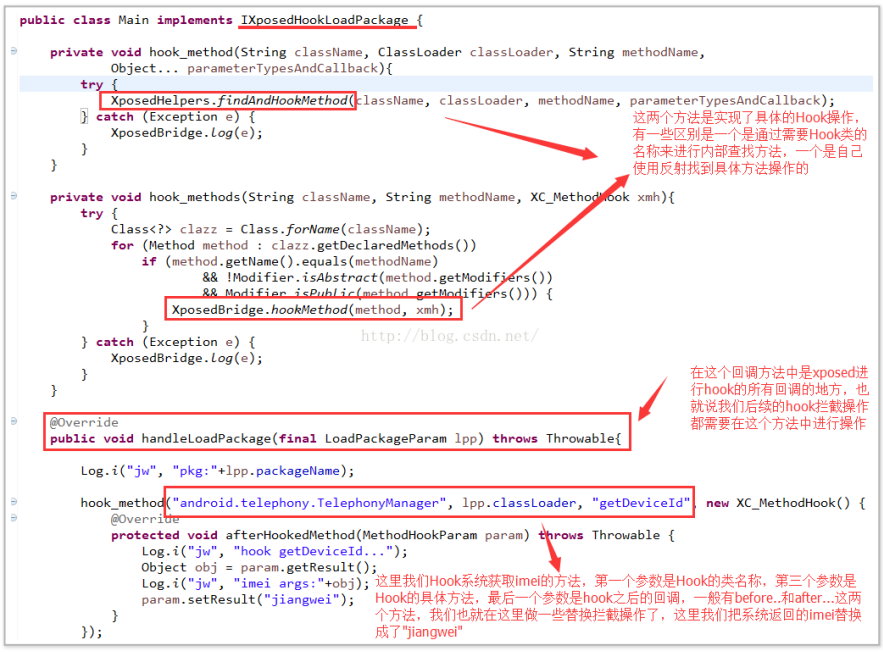
注意：

在 Eclipse 中，如果把工具包放到 libs 文件中，默认是加入到编译路径中的，同时在编译出来的程序中也是包含了这个工具包中的所有类，而对于其他非 libs 文件夹，我们添加工具包之后在 add buildpath 之后只是做到了工程引用工具包的功能，而最终并不会把这个工具包包含到程序中的。

第二步：编写模块代码

模块代码编写还是比较简单的，我们只要新建一个实现 IXposedHookLoadPackage 接口的类，然后在 handleLoadPackage 回调方法中进行拦截操作即可，而具体的拦截操作是借助 XposedHelpers.findAndHookMethod 方法和 XposedBridge.hookMethod 方法实现的，这两个方

法也是比较简单的，从参数含义可以看到，主要是 Hook 的类名和方法名，然后还有一个就是拦截的回调方法，一般是拦截之前做什么的一个 beforeHookedMethod 方法和拦截之后做什么的一个 afterHookedMethod 方法。




对于 IXposedHookLoadPackage 这个接口和回调方法，我们可以知道，应该是拦截系统中所有应用的运行信息，这里传递回来的一个 LoadPackageParam 参数类型就是包括了 Hook 应用的具体信息，我们可以打印应用的包名就可以看到

效果了。

注意：

如果你想 Hook 一个类的具体方法，那么就必须要清楚的了解到这个方法的相信信息，比如参数类型和个数，返回类型等。因为在拦截的过程中必须要对这个方法进行分析，比如得到方法参数来进行具体参数修改，返回值信息来进行返回值修改，这里看到了获取 imei 值的方法是一个无参数的返回字符串类型的方法，那么如果要拦截他的返回值，就需要修改他的返回值使用 setResult 方法即可。所以从这里可以看到不管是你 hook 系统的方法，还是日后去 hook 第三方应用的具体类方法，第一步都得了解到你 hook 对象的具体信息，关于系统方法咋们可以通过查看源码来得到信息，而对于第三方应用的话那么只能借助反编译技术了，比如修改游

戏金币功能，你必须先反编译游戏知道修改金币的类和具体方法才可行。

这里我不仅 Hook 了系统的 imei 信息，也简单的 Hook 了系统的地理位置信息，在 Android 中获取经纬度信息有三种方式，这里为了演示简单，用了 GPS 定位功能，一般获取经纬度信息的代码主要是两处：

一处是初始化的时候调用 getLastKnowLocation 方法获取最后一次系统中的地理位置信息

```
//获取地理位置管理器
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
//获取Location
Location location = locationManager.getLastKnownLocation(LocationManager.PASSIVE_PROVIDER);
if(location!=null){
    //不为空,显示地理位置经纬度
    showLocation(location);
}
```



还有一处就是监听地理位置变化的回调接口中的 onLocationChanged 回调方法：

```
LocationListener locationListener =  new LocationListener() {

    @Override
    public void onStatusChanged(String provider, int status, Bun

    }
```

```
    @Override
    public void onProviderEnabled(String provider) {

    }

    @Override
    public void onProviderDisabled(String provider) {

    }

    @Override
    public void onLocationChanged(Location location) {
        //如果位置发生变化,重新显示
        showLocation(location);

    }
};
```



所以如果想 Hook 系统的地理位置信息进行拦截，那么就需要操作这两处代码了，而他们有一个区别就是，第一处是通过返回值得到的，第二处是通过回调方法中的参数得到的。下面来看一下具体的 Hook 代码：

Hook 第一处代码比较简单，直接构造一个假的 Location 对象然后设置返回值即可。

```
//定位
hook_methods("android.location.LocationManager", "getLastKnownLocation", new XC_Me
    @Override
    protected void afterHookedMethod(MethodHookParam param) throws Throwable {
        Log.i("jw", "hook getLastKnownLocation...");
        Location l = new Location(LocationManager.PASSIVE_PROVIDER);
        double lo = -10000d; //经度
        double la = -10000d; //纬度
        l.setLatitude(la);
        l.setLongitude(lo);
        param.setResult(l);
    }
});
```

这里构造一个拦截之后的Location对象，然后设置了我们想要拦截的经纬度信息，最后在设置回去即可

Hook 第二处代码有点复杂，需要先找到添加位置监听的方法 requestLocationUpdates，然后通过反射得到这个回调对象，找到具体的回调方法，然后在进行操作，因为回调方法是通过参数把 Location 对象传递回来的，所以这里需要修改参数值。

```
hook_methods("android.location.LocationManager", "requestLocationUpdates", new XC_Me
    @Override
    protected void afterHookedMethod(MethodHookParam param) throws Throwable {

        Log.i("jw", "hook requestLocationUpdates...");

        if (param.args.length == 4 && (param.args[0] instanceof String)) {
            LocationListener ll = (LocationListener)param.args[3];
            Class<?> clazz = LocationListener.class;
            Method m = null;
            for (Method method : clazz.getDeclaredMethods()) {
                if (method.getName().equals("onLocationChanged")) {
                    m = method;
```

```
                    break;
                }
            }

            try {
                if (m != null) {
                    Object[] args = new Object[1];
                    Location l = new Location(LocationManager.PASSIVE_PROVIDER);
                    double lo = -10000d; //经度
                    double la = -10000d; //纬度
                    l.setLatitude(la);
                    l.setLongitude(lo);
                    args[0] = l;
                    m.invoke(ll, args);
                }
            } catch (Exception e) {
                XposedBridge.log(e);
            }
        }
    }
});
```



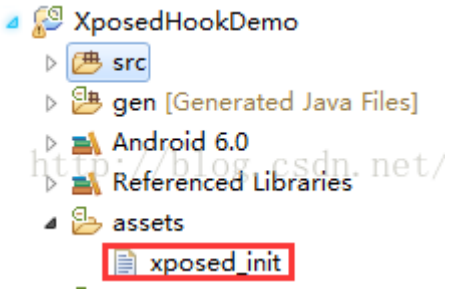
这里其实也是一样的效果，因为地理位置变化的回调监听方法是通过参数形式传递回去的，所以这里构造之后的location对象需要设置到参数中即可

好了，到这里我们就编写好了 Hook 系统的 imei 值和地理位置信息的模块了。

第三步：添加模块入口

这一步是非常重要的，也是最容易忘记的，就是要告诉 Xposed 框架一个模块中 Hook 的入口，这里可以看到模块

的入口是 Main 类，所以需要在模块的 assets 中添加一个 xposed_init 文件：






这里的内容很简单，就是模块入口类的全称名称即可：




第四步：添加模块的额外信息

最后一步就是需要在模块的 AndroidManifest.xml 文件添加额外信息，具体包括模块的描述信息，版本号等：

```
<meta-data
    android:name="xposedmodule"
    android:value="true" />
```

```
<meta-data
    android:name="xposeddescription"
    android:value="Hook系统信息" />
<meta-data
    android:name="xposedminversion"
    android:value="30" />
```

xposedmodule：代表的是 Android 程序作为 Xposed 中的一个模块，所以值为 true；
xposeddescription：代表的是对本模块的功能的描述，可以自己简单叙述下就可以了；
xposedminversion：代表的是本模块开发时用到的 xposed 的 jar 包的最低版本号，这里是 30，而我所用的 xposed 的 jar 包版本是 54；

经过上面四步之后咋们就完成了模块的定义了，最后咋们为了验证我们 Hook 的结果，在新建一个 Activity 类，在内部

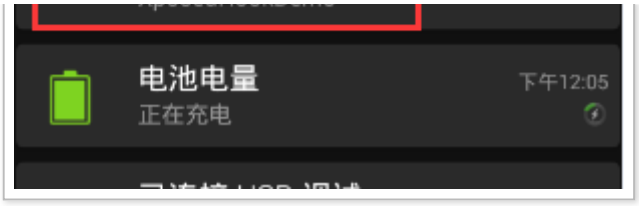
调用一下系统的获取 imei 方法以及位置信息方法，并且显示在屏幕中：

```
locationTxt = (TextView)findViewById(R.id.location);
imeiTxt = (TextView)findViewById(R.id.imei);

//获取地理位置管理器
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
//获取Location
Location location = locationManager.getLastKnownLocation(LocationManager.PASSIV
if(location!=null){
    //不为空,显示地理位置经纬度
    showLocation(location);
}
//监视地理位置变化
locationManager.requestLocationUpdates(LocationManager.PASSIVE_PROVIDER, 3000,

TelephonyManager telephonyManager = (TelephonyManager) this.getSystemService(Co
String imei = telephonyManager.getDeviceId();
imeiTxt.setText("imei:"+imei);
```

下面咋们就来运行一下模块程序，安装到设备之后，Xposed 会提示模块未激活：



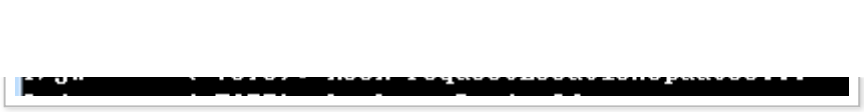

这个 XposedInstaller 程序应该是通过安装广播，然后得到这个应用信息分析他是否包含了 Xposed 模块的特殊属性来判断的。我们点击进行激活：


这时候看到，激活成功之后，会提示你再次重启设备才能生效，所以这里可以看到每次如果有新的模块或者是模块代码有更新了，比如这样：


都是需要重启设备，模块才能生效的，这一点还是有点蛋疼的和麻烦的。然后咋们重启设备之后，在运行我们的模块代码看看效果：


从这显示结果看到了，Hook 成功了，在没有 Hook 之前的效果是：


这时候咋们在来看一下打印的日志信息：



看到了，百度地图在获取我们设备的 imei 和位置信息，当然这是符合正常情况的，从这里可以看到，我们还可以利用这个技术来观察设备中有哪些应用在获取设备的一些隐私数据。

项目下载地址：
https://github.com/fourbrother/xposedhookdemo

本文主要是介绍了 Xposed 框架的基本使用以及一个简单作用，但是在实际过程中，这个框架是非常有用的，比如在文章开头就说到了，我们可以通过修改系统的一些信息来帮助测试模拟复杂的测试环境，但是这个框架现在用的最广泛的当属破解了，这个也是我们后续讲解的重点，用这个框架咋们可以进行应用的脱壳，游戏的外挂等。

本文是介绍 Xposed 的基础篇，主要介绍了 Xposed 的具体使用，XposedInstaller.apk 其实是一个模块载体和管理器，如果想实现具体的 Hook 操作，就必须自己在编写模块程序，然后在激活加载方可生效。后续会继续介绍用这个框架咋们来进行其他一些操作，比如应用的脱壳，游戏外挂编写，系统信息篡改等知识，期待大家多多期盼和点赞啦啦！！


更多内容： 点击这里

关注 微信 公众号，最新技术干货实时推送


觉得文章对您有帮助，就打个赏呗！

转载请注明： 尼古拉斯. 赵四 » Android 中 Xposed 框架篇—利用 Xposed 框架实现拦截系统方法

全文完