一、摘要

智能终端的普及使得移动应用突发式的增长，各种类型应用相继出现，给人们生活带来便捷的同时，也带来了许多安全问题，比如隐私泄露问题。智能手机用户迫切需要一款手机隐私保护软件来保护个人手机隐私安全。因此我们此次程序设计的目标是设计并实现一款 Android 手机隐私保护软件。同时，由于之前从未接触过安卓开发，想借此契机对这一领域有一个简单的认识。

本说明文档首先介绍了选题的目的及意义，随后针对移动软件的特点，从功能和非功能性需求出发，结合用例图对 Android 手机隐私保护软件进行了需求分析。接下来将软件开发划分为几个模块，并对各子模块的函数进行了介绍。重点详细介绍了数据库模块和 Xposed 模块的原理、设计及开发。然后阐述了隐私保护 APP 的 logo 设计理念。最后，介绍了软件的功能测试、兼容性测试和性能测试的结果，展示了软件的运行效果，并在此基础上总结了课题的成果和可以进一步改进的地方。附录简单罗列了参考文献和博客。

Android 手机隐私保护软件的创新点也是开发的重难点在于，采用基于 Xposed 框架的 API hook 主动防御技术，通过 API 调用来进行监控。

关键字： Android 开发 隐私保护 API Hook 技术 Xposed 框架

二、选题目的及意义

智能终端的普及使得移动应用突发式的增长，各种类型应用相继出现，给人们生活带来便捷的同时，也带来了许多安全问题。由于 Android 平台的开放性以及手机用户安全意识淡薄，Android 应用成为当前攻击者的主要攻击目标。就像腾讯安全实验室第一季度的手机安全报告指出的那样，上一年度 Android 手机病毒包新增 182，9188 个，同比增长 1200%。那么如何对移动应用进行全面有效的监测，尽可能多的发现应用中存

在的隐私泄露行为并及时拦截应当成为当前咱们安全工作者重点关注的内容，所以我们组在这次网络安全大型程序设计实践课中选择隐私保护 APP 开发这一课题。同时，由于之前从未接触过安卓开发，我们也想借此契机对这一领域有一个简单的认识。

三、需求分析及设计

一、功能性需求分析及设计

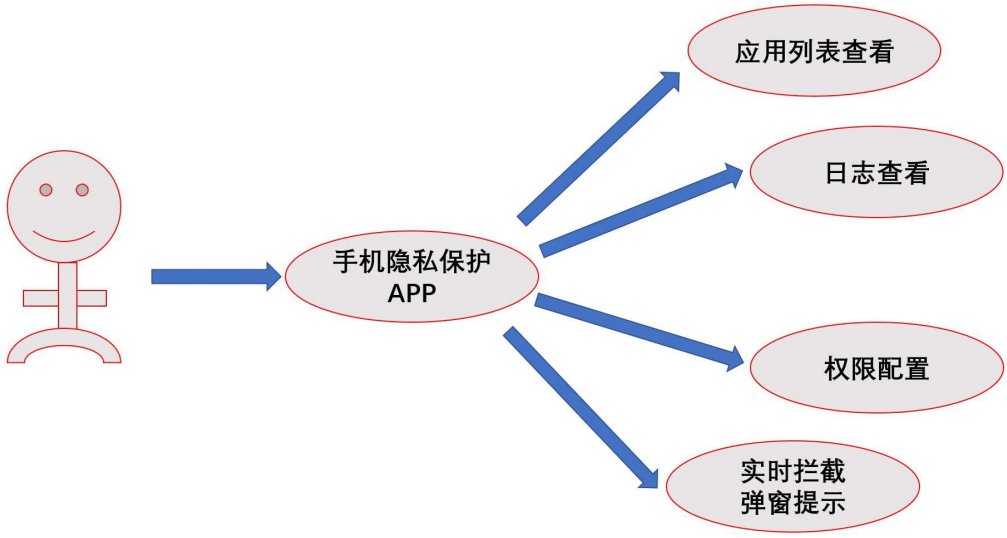
这次程设的主要任务就是对第三方软件隐私使用行为进行实时的监听、提醒和记录，全方位保护用户的私密数据。目前我们的隐私保护 APP 所能做到的有对通讯录联系人、通话记录、录音机、短信、相机、GPS 位置信息、传感器以及日历等功能的监听、提醒和记录，并对某些操作进行拦截。并且我们可以生成应用列表，用户可以查看应用列表，根据个人需求选择对某个应用是否监听、添加信任，也就是生成白名单并将其存储于数据库中。

隐私保护 APP 通过主动防御的方式，当第三方应用程序对上述用户隐私数据进行访问或者调用一些涉及隐私泄露的 API 时,我们会对用户进行弹窗提示来让用户感知，同时出于不打扰用户的目的，隐私保护 APP 的大部分工作都是在后台，为了便于让用户查看隐私保护 APP 的工作状况，我们还提供了记录日志信息的功能，所有日志信息都存储在数据库中，用户可以通过日志来了解手机上安装的第三方软件都访问了哪些数据，进行了哪些可能造成隐私泄露的危险操作，包括具体程序的操作情况，并对预设的涉及隐私泄露的操作设置进行拦截。另外，由于日志在一段时间后会积累很多，影响查阅，所以我们提供了清空日志功能，节约用户手机空间。

我们的手机系统自带权限限制功能，比如我在安装微信 APP 的时候，他会向我索取使用摄像头的使用权限，在这时我就已经决定了是否将这个权限赋予它，那么这个隐私保护 APP 还有什么意义呢。在这里做一个区分，这正是他的意义所在。当我们把使用摄像头的权限赋予给微信 APP 之后，在这以后，无论是真正的使用还是在后台使用，系

统都不会再通知我们了。假如微信 APP 是一个恶意程序，那么对于他的所作所为在这之后我们将无从知晓。然而，当你使用了我们的隐私保护 APP 以后，即使你赋予了它使用你摄像头的权限，我们的隐私保护 APP 也会认真负责的对它的行为进行提醒和相应的日志记录，通过这些记录，你就可以知道你的微信有没有背着你随意使用你赋予给他的权限，从而达到保护你私密数据的目的。

还有一些很霸道的软件，在安装的时候强行要求使用某些权限或者使用某些功能比如摄像头，否则软件无法正常打开和运行，在这种情况下，使用我们的隐私保护 APP 就可以实现既保证软件的正常启动和运行，又没有真正赋予它实际的使用权限，即通过 hook 技术在他调用摄像头时进行拦截和拒绝。



隐私保护软件用例图

二、非功能性需求分析及设计

随着智能手机性能的不断提升，现在智能手机的 CPU 已经与个人电脑的配置相当，但是手机作为一款移动设备，在操作使用上仍然与 PC 软件有所不同。因此，我们在手机隐私保护 APP 的设计与实现中要尽可能认识到这些特点，真正符合移动软件的设计需求。所以在实际开发过程中，我们考虑了以下几个方面：

1．屏幕分辨率

首先，智能手机作为便携式设备，屏幕大小有限，因此，需要科学的界面设计，从而在有限的空间里向用户展现更加丰富且合理的信息。其次，隐私保护 APP 是基于 Android 平台，市场上的 Android 设备有数百款，并且采用不同大小的屏幕和分辨率，所以在开发过程中，我们考虑了不同分辨率和屏幕的适配。

2．电力续航

智能手机由于采用的是充电电池，其续航能力有限，尤其是在 Android 平台，如采应用程序开发不严谨，会进一步缩短续航时间。从应用程序编写的角度来看，造成应用程序耗电过高的原因无非大量数据的传输，和频繁地重复执行某项操作和代码冗余。所以在 APP 开发过程中，我们努力避免了以上几点。在后期代码优化的时候，将许多功能封装成函数，有些模块代码量从几百行精简到几十行，同时代码可读性大大提高。

3．尽量不打扰用户

由于使用智能手机往往是在用户的零碎时间，而隐私保护软件又是一款工具类软件，所以，要做到尽到不打扰用户。基于这个需求，我们所能做的就是操作简单，用户界面友好。

四、主要模块介绍及实现技术

1.基于 Xposed 的 API hook

总体工作流程

1.概述

通过调研和研究文献，我发现现在市场上的或者说传统的隐私保护 APP 基本上都是在系统内核级进行修改，需要对 Android 原生系统内核进行重新编译，定制自己的监测系统,实现对恶意行为的监控。然而这种高大上的操作,不是我们目前可以驾驭的。

实际上，Android 应用的功能都是通过系统提供的 SDK（软件开发工具包）来实现的,按照 SDK 接口来执行规范调用。由于应用运行期间的行为都是通过调用系统 API 来实现，那么我们就可以通过监控应用的系统 API 调用行为来实现对恶意行为的监控。基于此原理和我们之前所学过的 hook 技术，本次 Android 开发我们采用一种基于 Xposed 框架的监控方案，将 Android 系统中敏感 API 调用劫持，通过直接监控系统敏感 API 调用来实现对恶意行为的监控。

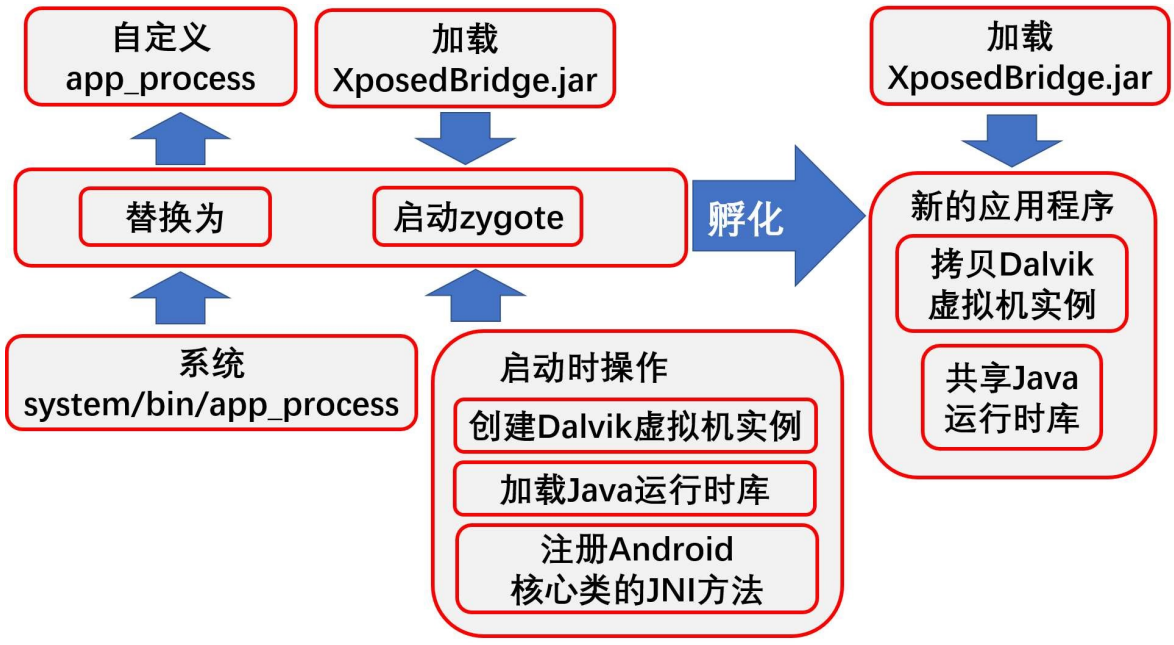
2.关键技术分析

在对 java 方法进行 hook 时，先将虚拟机里面这个方法的 Method 改为 native Method(其实就是更改一个标识字段)，然后将该方法的 nativeFunc 指向自己实现的一个 native 方法，这样在调用方法时，就会调用到这个 native 方法，从而接管了控制权。在这个 native 方法中，直接调用了一个 java 方法，这个 java 方法里面对原方法进行了调用，并在调用前后插入了钩子，于是就 hook 住了这个方法。

Java 的 hook 原理就是这么简单，但它有其他的问题要解决：如何将 hook 的代码注入到目标 app 的进程中？

Xposed 框架是一款可以在不修改 APK 的情况下影响程序运行(修改系统)的框架服务，基于它可以制作出许多功能强大的模块，且在功能不冲突的情况下同时运作。

基于 Xposed 框架可以在不修改 APK(安卓安装包)的情况下影响程序运行或者修改系统的框架服务，通过替换/system/bin/app_process 程序控制 Zygote 进程，使得 app_process 在启动过程中加载 XposedBridge.jar 包，从而实现对 Zygote 进程及其创建的 Dalvik 虚拟机的劫持。



- Android 系统是基于 Linux 内核的，在 Android 系统中，zygote 是 android 系统最最初运行的程序，所有的应用程序进程以及系统服务进程都是由 Zygote 进程孕育

孵化（fork）出来的，于是 zygote 中加载的代码，在所有 fork 出来的子进程都含有（app 进程也是 fork 出来的）。所以 Xposed 是一个可以 hook android 系统中任意一个 java 方法的 hook 框架。

Android 应用自动联网、收发短信、通讯录访问等应用行为都是通过系统提供的 API 实现的，因此应用运行中的行为必然都体现在应用程序对接口的调用上，通过监控系统接口调用即可实现对应用行为的监控。所以，我们实现监听的方法就是，先通过调研，研究得出常见的敏感 API，分析其所属包以及参数和返回值等信息，利用 Xposed 提供的 XposedBridge 编写定制的 modules，并在 XposdInstaller 中注册，开启对敏感 API 的监听，当有应用命中所监听的 API 时，进行 hook 劫持。

3.系统敏感 API 分析

行为	所属类	敏感 API
联网	android.net.wifi.IWifiManager	setWifiEnabled
短信	android.telephony.SmsManager	sendTextMessage sendDataMessage, sendMultipartTextMessage getAllMessagesFromIcc
电话	android.telephony.TelephonyManager	getLine1Number,listen
通讯录	android.provider.ContactsContract	PhoneLookup
录音	android.media.MediaRecorder	start, stop
摄像头	android.hardware.Camera	takePicture setPreviewCallback setPreviewCallbackWithBuffer setOneShotPreviewCallback
蓝牙	android.bluetooth.IBluetoothManager	enable

4.Xposedinit 模块函数说明

XposedInit		
接口	fun myLog(context: Context)	
说明	日志：Xposed+Toast+数据库	
参数	类型	含义
context	Context	系统当前应用的上下文
返回值	void	

接口	fun getStateAndName()	
说明	获取当前应用及模块监测开关状态并返回，应用及模块名称赋给类中的私有变量	
参数	类型	含义
context	Context	系统当前应用的上下文
methodName	String	被调用的类名+方法名
packageName	String	调用者应用的包名
返回值	Boolean	开关状态

接口	fun addHook()	
说明	统一的hook操作	
参数	类型	含义
lpparam	LoadPackageParam	包含调用者应用的信息，如包名
className	String	hook的类名
methodName	String	hook的方法名
返回值	void	

接口	fun getModuleName()	
说明	获取ContentProvider访问的模块	
参数	类型	含义
uri	String	ContentProvider访问的uri
返回值	String	uri对应的模块名moduleName

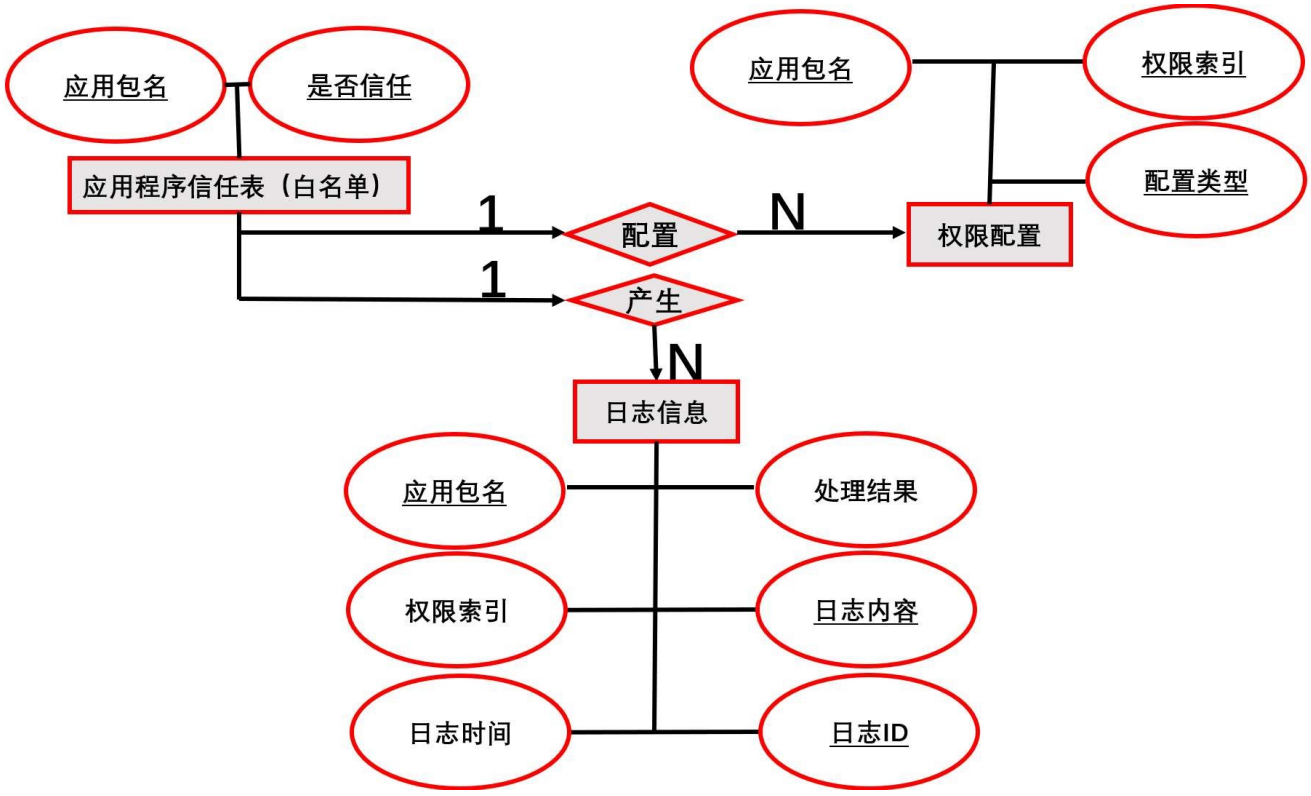
2.数据库模块

Android 系统的数据库采用的是 SQLite，这是一种轻量级的数据库。隐私保护 APP 的数据库模块提供的是隐私保护策略存储（白名单和监听功能等）和日志存储。

隐私保护 APP 的数据库的表结构设计如下：

AppState	该表用于存储监听的应用
packageName	text
appName	text
state	integer
ModuleState	该表用于记录监听的功能
methodName	text
moduleName	text
state	integer
Log	该表用于存储日志记录
id	integer
time	text
appName	text
moduleName	text

数据库 E-R 实体关系图：



3.其他模块

应用列表模块函数说明

AppListActivity		
接口	protected void onCreate(Bundle savedInstanceState)	
说明	用于启动AppListActivity	
参数	类型	含义
savedInstanceState	Bundle	之前活动结束时保存的状态
返回值	void	无返回值

接口	public void onClick(View v)（Adapter中定义）	
说明	在点击每个应用的监听开关的时候触发，修改数据库的内容	
参数	类型	含义
v	View	点击的开关对应的View
返回值	void	无返回值

接口	public void onClick(View v)（onCreate中定义）	
说明	点击重置按钮清库，然后重新加载显示	
参数	类型	含义
v	View	点击的按钮对应的View
返回值	void	无返回值

接口	public View getView(int position, View convertView, ViewGroup parent)	
说明	从数据库中读取信息（如果没有就用默认值），设置要显示出来的View	
参数	类型	含义
position	int	当前视图在数据集中的位置
convertView	View	可能会被重用的旧的视图
parent	ViewGroup	当前视图最终会被附加到的父级
返回值	View	指定位置对应的视图

ModuleListActivity
LogActivity
和AppListActivity中的函数功能大体一致，不赘述

MyDBOpenHelper		
接口	public void onCreate(SQLiteDatabase db)	
说明	第一次启动应用时初始化数据库	
参数	类型	含义
db	SQLiteDatebase	数据库对象
返回值	void	无返回值

MyAppInfo		
接口	public static List<MyAppInfo> scanLocalInstallAppList (PackageManager packageManager)	
说明	获取当前android系统中所有的应用包信息	
参数	类型	含义
packageManager	PackageManager	包管理器对象
返回值	List<MyAppInfo>	应用信息列表

4.图标设计

盾牌，象征着抵御，防护，最本质，隐私保护 APP，安全 APP 开发约定俗成的，腾讯电脑管家，金山卫士，360 安全卫士。

中间的小人，其实是戴了眼镜的安卓图标，表示对 Android 平台上 APP 的行为进行监控

外面的圆，将整个 Android 小人圈起来，像一个手铐，意思是对所有 Android APP 的权限进行限制,行为进行控制,同时又像放大镜,表示全方位监控(通讯录,相机……)

中间很抽象的分为上下两部分(左右两部分),成为两个 P,意思是 privacy protect,点明 APP 的作用，保护用户隐私。

整体颜色是黑白，黑色和白色永远是设计中的最佳拍档。黑色象征与众不同，保持距离，划清界限。白色飘溢着不容妥协，难以侵犯的气韵。

五、整体展示

见附件：基于 Xposed 框架的隐私保护 APP 效果演示

六、总结

这次程设的主要任务就是对第三方软件隐私使用行为进行实时的监听、提醒和记

录，全方位保护用户的私密数据。目前我们的隐私保护 APP 所能做到的有对通讯录联系人、通话记录、录音机、短信、相机、GPS 位置信息、传感器以及日历等功能的监听、提醒和记录，并对某些操作进行拦截。并且我们可以生成应用列表，用户可以查看应用列表，根据个人需求选择对某个应用是否监听、添加信任，也就是生成白名单并将其存储于数据库中。通过效果展示，可以看出任务全部完成并进行了拓展和创新。Android 手机隐私保护一方面需要依赖于传统的恶意软件病毒库以及人工智能的算法，另一方面，更需要本次开发实现的实施拦截并记录的方法。本次课题采用的基于 Xposed 框架的隐私保护解决方案具有更好的实时性、稳定性、兼容性和可拓展性，这也是与其他等价方案或同类软件相比更有优势和领先的地方。

七、不足和进一步改进的方向

1. 我们的软件只关注了比较重要的几个分支，例如通讯录联系人、通话记录、录音机、短信、相机、GPS 位置信息、传感器以及日历，后续还可以在功能上继续扩充。
2. 本软件只对权限的控制进行了允许、禁止和提示三种操作，对于一些比较特殊的分支，可以构造一些假的数据或者空数据进行返回，这样可以开发出一些关联的隐私保护应用。

附录：参考文献和博客

https://blog.csdn.net/liu149339750/article/details/8111462
https://blog.csdn.net/xxxzhi/article/details/52167056
https://blog.csdn.net/xinxinsky/article/details/51443255
https://blog.csdn.net/thl789/article/details/7880650
https://www.jianshu.com/p/08b9b69f1050
https://www.cnblogs.com/renqingping/archive/2012/10/25/Parcelable.html
http://www.it1352.com/115015.html
https://blog.csdn.net/luojiusan520/article/details/47696891
https://www.jianshu.com/p/bf70742cf8d8
https://blog.csdn.net/serapme/article/details/45559239
https://blog.csdn.net/love_xsq/article/details/50501874
https://blog.csdn.net/qinjuning/article/details/6867806

https://blog.csdn.net/lusing/article/details/52229795
https://www.jianshu.com/p/d088a042bf6d
https://www.jianshu.com/p/fa472095ad58
https://blog.csdn.net/flowingflying/article/details/12095933
https://blog.csdn.net/wtoria/article/details/52038518
https://blog.csdn.net/pugongying1988/article/details/7910873
https://www.2cto.com/kf/201408/328297.html
https://blog.csdn.net/qinjuning/article/details/6867806
https://www.cnblogs.com/android-for-dh/p/4449524.html
https://www.cnblogs.com/renqingping/archive/2012/10/25/Parcelable.html
https://blog.csdn.net/qq_16064871/article/details/79038083
https://blog.csdn.net/flowingflying/article/details/38871219
https://blog.csdn.net/baidu_26994091/article/details/51686218
https://blog.csdn.net/u013430507/article/details/44064761
https://blog.csdn.net/baidu_26994091/article/details/51686218
https://blog.csdn.net/baimy1985/article/details/7671059
https://www.2cto.com/kf/201302/188055.html
https://blog.csdn.net/u010784534/article/details/47836165
http://www.360doc.com/content/11/0511/17/5962017_115996330.shtml
http://www.360doc.com/content/11/0511/17/5962017_115996330.shtml
https://blog.csdn.net/u013045971/article/details/42396539
https://www.jianshu.com/p/9b7abdff9e13
https://blog.csdn.net/zhangcanyan/article/details/52817866
https://zhidao.baidu.com/question/1864607157611716507.html
https://www.jb51.net/article/73252.htm
https://blog.csdn.net/zqiang_55/article/details/8115007
http://1028826685.iteye.com/blog/1666619
https://www.itstrike.cn/Question/4c9b1327-a84e-4c3d-9fbd-832bf23307f8.html
https://www.jianshu.com/p/acefb23a7da5
https://www.jianshu.com/p/02c7508b2d1f
https://blog.csdn.net/wzsong123/article/details/6741973
https://stackoverflow.com/questions/28522594/why-is-the-ssid-from-getconfigurednetworks-added-quotation-mark
https://bbs.csdn.net/topics/390444517
https://blog.csdn.net/feiyue0823/article/details/9763135
https://www.cnblogs.com/kafeibuku/p/5091333.html
http://blog.163.com/liu_jun_y/blog/static/188086312201171572833546/
http://blog.sina.com.cn/s/blog_560e7f660101gzw1.html
https://blog.csdn.net/aqi00/article/details/50511517
https://blog.csdn.net/itfootball/article/details/25421015
https://blog.csdn.net/mockingbirds/article/details/53453326
https://blog.csdn.net/lazyer_dog/article/details/52230712
https://blog.csdn.net/guotong1988/article/details/7690481
https://blog.csdn.net/wds1181977/article/details/50594977
https://blog.csdn.net/yangbin0513/article/details/68490291
https://www.2cto.com/kf/201609/545101.html
https://blog.csdn.net/qq_34452829/article/details/54017555
https://blog.csdn.net/dong5488/article/details/53022696
http://blog.163.com/liu_jun_y/blog/static/188086312201171572833546/

https://blog.csdn.net/zx249388847/article/details/80077894

https://www.jianshu.com/p/9e89a382927d

https://www.cnblogs.com/zhongxia/p/5554294.html

http://www.uml.org.cn/j2ee/201409023.asp

https://msdn.microsoft.com/zh-cn/library/vs/alm/2ey6a271(d=printer,v=vs.85).aspx?cs-save-lang=1&cs-lang=vb

https://blog.csdn.net/mbh12333/article/details/72541277?locationNum=3&fps=1

https://blog.csdn.net/lissdy/article/details/7039332

https://www.cnblogs.com/whoislcj/p/5583833.html

https://blog.csdn.net/gaojinshan/article/details/44856935

https://blog.csdn.net/weixin_37730482/article/details/77847183

https://www.cnblogs.com/android100/p/android-send-msg.html

https://www.cnblogs.com/roadone/p/7977544.html

https://firebase.google.cn/support/guides/disable-analytics?hl=zh-cn

https://baike.baidu.com/item/AppActivate/8875980

https://www.jianshu.com/p/ae57de07448c

https://msdn.microsoft.com/zh-cn/library/microsoft.teamfoundation.client.ilogger.logevent.aspx

http://gundumw100.iteye.com/blog/1559632

https://baike.baidu.com/item/getInstance/10954093?fr=aladdin

https://www.jianshu.com/p/8858a92374f5

https://blog.csdn.net/u011228356/article/details/45026441

https://www.jianshu.com/p/7f766eb2f4e7

https://www.cnblogs.com/cjcblogs/articles/5647766.html

https://blog.csdn.net/dzkdxyx/article/details/78879521

https://blog.csdn.net/gjy211/article/details/52015198

https://www.jianshu.com/p/b6f4b0aca6b0

https://blog.csdn.net/word_code/article/details/78116286