移动互联网安全

实验一

1、airmon-ng

airmon-ng用来实现无线接口在managed和monitor模式之间的转换。输入不带参数的airmon-ng命令，则会显示当前的接口状态。

2、iwconfig

说明：iwconfig是[LWE](https://baike.baidu.com/item/LWE)最主要的工具，可以对无线网卡的大部分参数进行配置。

参数：

essid：设置无线网卡的[ESSID](https://baike.baidu.com/item/ESSID/6482785)(Extension Service Set ID)。通过ESSID来区分不同的[无线网络](https://baike.baidu.com/item/%E6%97%A0%E7%BA%BF%E7%BD%91%E7%BB%9C)，正常情况下只有相同ESSID的无线站点才可以互相通讯，除非想监听无线网络。其后的参数为双引号括起的ESSID字符串，或者是any/on/off，如果ESSID字符串中包含

any/on/off，则需要在前面加"--"。

示例：

#iwconfig eth0 essid any 允许任何[ESSID](https://baike.baidu.com/item/ESSID/6482785)，也就是[混杂模式](https://baike.baidu.com/item/%E6%B7%B7%E6%9D%82%E6%A8%A1%E5%BC%8F/6463778)

#iwconfig eth0 essid "My Network" 设置ESSID为"My Network"

#iwconfig eth0 essid -- "ANY" 设置ESSID为"ANY"

nwid: Network ID，只用于pre-802.11的无线网卡，802.11网卡利用ESSID和AP的[MAC地址](https://baike.baidu.com/item/MAC%E5%9C%B0%E5%9D%80/1254181)来替换nwid，现在基本上不用设置。

示例：

#iwconfig eth0 nwid AB34

#iwconfig eth0 nwid off

nick: Nickname，一些网卡需要设置该参数，但是802.11[协议栈](https://baike.baidu.com/item/%E5%8D%8F%E8%AE%AE%E6%A0%88/3155224)、MAC都没有用到该参数，一般也不用设置。

示例：

#iwconfig eth0 nickname "My Linux Node"

mode：设置无线网卡的工作模式，可以是

Ad-hoc：不带AP的[点对点](https://baike.baidu.com/item/%E7%82%B9%E5%AF%B9%E7%82%B9/7452984)[无线网络](https://baike.baidu.com/item/%E6%97%A0%E7%BA%BF%E7%BD%91%E7%BB%9C)

Managed：通过多个AP组成的网络，无线设备可以在这个网络中漫游

Master：设置该无线网卡为一个AP

Repeater：设置为无线网络[中继](https://baike.baidu.com/item/%E4%B8%AD%E7%BB%A7/12002202)设备，可以转发网络包

Secondary：设置为备份的AP/Repeater

Monitor：监听模式

Auto：由无线网卡自动选择工作模式

示例：

#iwconfig eth0 mode Managed

#iwconfig eth0 mode Ad-Hoc

freq/channel：设置无线网卡的工作频率或者频道，小于1000的参数被认为是频道，大于10000的参数被认为是频率。频率单位为Hz，可以在数字后面附带k, M, G来改变数量级，比如2.4G。频道从1开始。使用lwlist工具可以查看无线网卡支持的频率和频道。参数off/auto指示[无线网络](https://baike.baidu.com/item/%E6%97%A0%E7%BA%BF%E7%BD%91%E7%BB%9C)自动挑选频率。

注意：如果是Managed模式，AP会指示无线网卡的工作频率，因此该设置的参数会被忽略。Ad-hoc模式下只使用该设定的频率初始无线网络，如果加入已经存在的Ad-hoc网络则会忽略该设置的频率参数。

示例：

#iwconfig eth0 freq 2422000000

#iwconfig eth0 freq 2.422G

#iwconfig eth0 channel 3

#iwconfig eth0 channel auto

ap：连接到指定的AP或者[无线网络](https://baike.baidu.com/item/%E6%97%A0%E7%BA%BF%E7%BD%91%E7%BB%9C)，后面的参数可以是AP的[MAC地址](https://baike.baidu.com/item/MAC%E5%9C%B0%E5%9D%80/1254181)，也可以是iwlist scan出来的标识符。如果是Ad-hoc，则连接到一个已经存在的Ad-hoc网络。使用off参数让无线网卡不改变当前已连接的AP下进入自动模式。any/auto参数，无线网卡自动选择最好的AP。

注意：如果无线信号低到一定程度，无线网络会进入自动选择AP模式。

示例：

#iwconfig eth0 ap 00:60:1D:01:23:45

#iwconfig eth0 ap any

#iwconfig eth0 ap off

rate/bit：如果无线网卡支持多速率，则可以通过该命令设置工作的速率。小于1000的参数由具体的无线网卡驱动定义，一般是传输速率的索引值，大于1000的为速率，单位bps，可以在数字后面附带k, M, G来指定数量级。auto参数让无线网卡自动选择速率 fixed参数让无线网卡不使用自动速率模式。

示例：

#iwconfig eth0 rate 11M

#iwconfig eth0 rate auto

#iwconfig eth0 rate 5.5M auto //自动选择5.5M以下的速率

[txpower](https://baike.baidu.com/item/txpower/8406698)：如果无线网卡支持多发射功率设定，则使用该参数设定发射，单位为[dBm](https://baike.baidu.com/item/dBm/7518227)，如果指定为W（毫瓦），只转换公式为：

dBm=30+log(W)。参数on/off可以打开和关闭发射单元，auto和[fixed](https://baike.baidu.com/item/fixed/4579853)指定无线是否自动选择发射功率。

示例：

#iwconfig eth0 txpower 15

#iwconfig eth0 txpower 30mW

#iwconfig eth0 txpower auto

#iwconfig eth0 txpower off

sens：设置[接收灵敏度](https://baike.baidu.com/item/%E6%8E%A5%E6%94%B6%E7%81%B5%E6%95%8F%E5%BA%A6/768866)的下限，在该下限之下，无线网卡认为该[无线网络](https://baike.baidu.com/item/%E6%97%A0%E7%BA%BF%E7%BD%91%E7%BB%9C)信号太差，不同的网卡会采取不同的措施，一些现代的无线网卡会自动选择新的AP。正的参数为raw data，直接传给无线网卡驱动处理，一般认为是百分比。负值表示[dBm](https://baike.baidu.com/item/dBm/7518227)值。

示例：

#iwconfig eth0 sens -80

#iwconfig eth0 sens 2

retry：设置无线网卡的[重传机制](https://baike.baidu.com/item/%E9%87%8D%E4%BC%A0%E6%9C%BA%E5%88%B6/4564949)。limit ‘value’ 指定最大重传次数；lifetime ‘value’指定最长重试时间，单位为秒，可以附带m和u来指定单位为毫秒和微秒。如果无线网卡支持自动模式，则在limit和lifetime之前还可以附加min和max来指定上下限值。

示例：

#iwconfig eth0 retry 16

#iwconfig eth0 retry lifetime 300m

#iwconfig eth0 retry min limit 8

rts：指定RTS/CTS握手方式，使用RTS/CTS握手会增加额外开销，但如果无线网络中有隐藏无线[节点](https://baike.baidu.com/item/%E8%8A%82%E7%82%B9)或者有很多无线节点时可以提高性能。

后面的参数指定一个使用该机制的最小包的大小，如果该值等于最大包大小，则相当于禁止使用该机制。可以使用auto/off/[fixed](https://baike.baidu.com/item/fixed/4579853)

参数。

示例：

#iwconfig eth0 rts 250

#iwconfig eth0 rts off

frag：设置发送数据包的分片大小。设置分片会增加额外开销，但在噪声环境下可以提高数据包的到达率。一般情况下该参数小于最大包大小，有些支持Burst模式的无线网卡可以设置大于最大包大小的值来允许Burst模式。还可以使用auto/fixed/off参数。

示例：

#iwconfig eth0 frag 512

#iwconfig eth0 frag off

key/enc[ryption]：设置无线网卡使用的加密[密钥](https://baike.baidu.com/item/%E5%AF%86%E9%92%A5)，此处为设置[WEP](https://baike.baidu.com/item/WEP/612151)模式的加密key，如果要使用WPA，需要wpa\_supplicant工具包。密钥参数可以是 XXXX-XXXX-XXXX-XXXX 或者 XXXXXXXX 格式的[十六进制](https://baike.baidu.com/item/%E5%8D%81%E5%85%AD%E8%BF%9B%E5%88%B6/4162457)数值，也可以是s:xxxxxx的[ASCII](https://baike.baidu.com/item/ASCII/309296)字符。如果在密钥参数之前加了[index]，则只是设置该索引值对应的密钥，并不改变当前的密钥。直接指定[index]值可以设置当前使用哪一个密钥。指定on/off可以控制是否使用加密模式。open/restricted指定加密模式，取决于不同的无线网卡，大多数无线网卡的open模式不使用加密且允许接收没有加密的数据包，restricted模式使用加密。可以使用多个key参数，但只有最后一个生效。

[WEP](https://baike.baidu.com/item/WEP/612151)密钥可以是40bit，用10个十六进制数字或者5个[ASCII](https://baike.baidu.com/item/ASCII/309296)字符表示，也可以是128bit，用26个十六进制数字或者13个ASCII字符表示。

示例：

#iwconfig eth0 key 0123-4567-89

#iwconfig eth0 key [3] 0123-4567-89

#iwconfig eth0 key s:password [2]

#iwconfig eth0 key [2]

#iwconfig eth0 key open

#iwconfig eth0 key off

#iwconfig eth0 key restricted [3] 0123456789

#iwconfig eth0 key 01-23 key 45-67 [4] key [4]

power：设置无线网卡的电源管理模式。period ‘value’ 指定唤醒的周期，timeout ‘value’指定进入休眠的等待时间，这两个参数之前可以加min和max修饰，这些值的单位为秒，可以附加m和u来指定毫秒和微秒。off/on参数指定是否允许电源管理，all/[unicast](https://baike.baidu.com/item/unicast/10769943)/multicast

指定允许唤醒的数据包类型。

示例：

#iwconfig eth0 power period 2

#iwconfig eth0 power 500m unicast

#iwconfig eth0 power timeout 300u all

#iwconfig eth0 power off

#iwconfig eth0 power min period 2 power max period 4

commit：提交所有的参数修改给无线网卡驱动。有些无线网卡驱动会先缓存无线网卡参数修改，使用这个命令来让无线网卡的参数修改生效。不过一般不需要使用该命令，因为无线网卡驱动最终都会使参数的修改生效，一般在debug时会用到。

为了方便配置，可以把配置写到 /etc/network/interfaces中，这样以后就不用反复配置了。

auto lo

iface lo inet loopback

auto eth1

iface eth1 inet static

address 192.168.1.3

netmask 255.255.255.0

gateway 192.168.1.1

echo nameserver 192.168.1.1>/etc/resolv.conf

pre-up /sbin/iwconfig eth1 essid "LW HOME LINK"

pre-up /sbin/iwconfig eth1 key s:liwei

auto usb0

iface usb0 inet static

address 192.168.0.200

netmask 255.255.255.0

auto dsl-provider

iface dsl-provider inet ppp

pre-up /sbin/ifconfig eth0 up # line maintained by pppoeconf

provider dsl-provider

auto eth0

iface eth0 inet manual

3、wireshark

* 无线网卡选择wifi连接，设为monitor模式抓包
* Wireless -> wlan traffic
* 规则—>右键 Apply as Filter

4、iw dev

\* 其中type为monitor时，右上角网络中不会再显示此无线网卡

5、lsusb

查看硬件接口信息

6、ifconfig -a

7、lsb\_release -a

8、iw phy

9、dmesg

列出加载到内核中的所有驱动

10、iwlist

控制无线网卡获取详细无线网络信息

11、iwlist wlan1 scanning

主动获取可用AP信息列表

12、tail -f /var/log/messages

设备运行时变化产生消息并被记录到上述日志文件

13、dpkg -L aircrack-ng

显示aircrack-ng软件包的列表

14、airmon-ng start wlan0（ 再看iw dev，Interface 变为 wlan0mon）

15、airodump-ng wlan0mon -w

ls

cd workspace/

ls

cd ..

Mkdir -p workspace/pcap\_demo

Cd workspace/pcap\_demo

Ls

airodump-ng wlan0mon -w 20171006（更换文件夹抓包）

ls

file \*（查看文件描述）

wireshark 20171006-01.cap

16、airodump-ng wlan0mon -w 20171006 -c 11

17、ifconfig：通用网卡设置工具

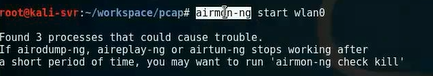
实验二：

1. 切换channel
2. 负数越接近于零，信号越好
3. 管理帧：支持认证方式（WPA），WPS，监听channel（wlan.fc.type==0）
4. Statics：编码（UTF8或GBK），同一BSSID下所有的连接设备
5. 不同类型地址命名：ra （接收地址）、wlan.addr（所有地址），多种地址可能相同（ap可以连接主动响应，有接收后会变化）
6. 分析隐藏SSID（INFO里看）
7. SSID有无广播：
8. 控制帧：wlan.fc.type==1
9. 数据帧：wlan.fc.type==2，明文和密文数据，也有可能没有有效的链路层以上数据
10. 劫持账号
11. 分析DNS
12. Teredo
13. 时间大量花费在探测信道是否可用

\*相关实验操作：

1. 打开监听抓包（可抓特定频道）

\*ifconfig -a可查看无线网卡是否接入，抓包时的提示信息可能是抓包失败突破点



\*抓包时channel不停变更

\*抓包命令：airodump-ng wlan0mon -c 1(channel=1) -w 20171009

\*抓包文件瘦身：wireshark——>File——>export specified packets——>range（0-1000,1000条）

1. wireshark分析技巧

\*wlan.fc.type==0：管理帧

\*SSID=broadcast，广播自己，让别人访问到

\*wireless——>wlan traffic

\*SSID解析：copy——>as a Hex Stream——>使用php

Php -a

代码：$a=””; echo hex2bin($a);//UTF8编码

\*wlan.bssid==

\*四个地址as filter

\*bssid as filter

\*wlan.fc.type==1：控制帧

\*wlan.fc.type==2：数据帧

可分析HTTP、TCP数据流：File——>export objects——>HTTP

找到对应帧，Follow——>Http stream（红色字体为发出的请求，蓝色字体为响应）

\*static——>IO graphs（Display filter中可自己限定规则）

\*\*插一嘴：新版本不用再安装增强功能 apt update && dist-upgrade

\*\*四次握手认证信息截获：wireshark filter输入 eapol（可看到MIC）

实验三：

1. 视频：Megaprimer（配套电子书）
2. Securitytube
3. 手机支持wps功能
4. 实验：wps交互过程
5. 查看模拟的网络名称
6. Tshark

-r：读入

-n：禁止名称解析

-T：最后

1. airbase-ng：伪造AP

-a：指定ap当前使用mac地址

-- essid xx：开启xx无线网络，只指定1个essid

-z：选择不同加密方法

-Z：WPA2加密模式

-essids ：从文件中读取

1. 是否支持同时工作在两种模式（监听、基站）
2. 查看Probe request：filter中的MAC address
3. 查看MAC地址
4. Ieee oui 过滤
5. 无线网卡搭建共享网络
6. Wireshark --beacons：能抓到所有beacons
7. 无线网卡连接了无线网络后，使用iwconfig将看到ESSID为连入的无线网络名称

操作：

1. airmon-ng start wlan0

Airodump-ng -c channel wlan0mon --wps -w 文件名 --beacons:Display WPS information (if any)

无需记忆，在wireshak中右键copy—>filed name即得

可疑mac地址

1. ls -l:查看文件详细信息
2. Tshark：使用tshark分析抓包结果





Tshark -r -n -T field -e wlan\_mgt.ssid -Y “”| sort -u

\* sort -u：过滤重复

\* 直接解码??





\*所有不重复的mac地址

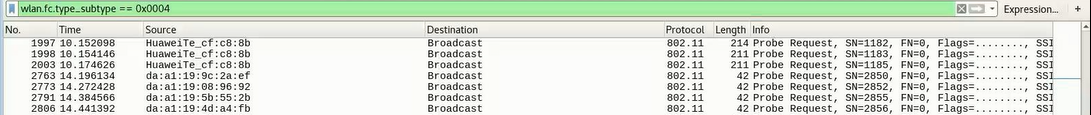
tshark -r -n -T fields -e ‘wlan.sa’ | sort -u

1. 分析伪造的bssid：假mac地址，假名称（利用php hex2bin得到了结果），在wireshark中选定为过滤规则
2. 基站模式、建立热点：airbase-ng(默认也属于监听模式iw dev查看）

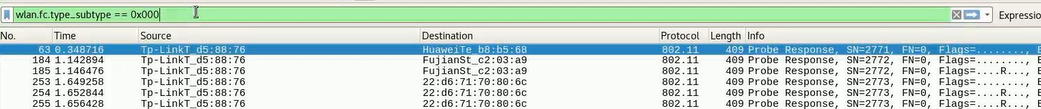
\*airbase-ng --essid HELLO23333 wlan0

\*airbase-ng --essids essid.txt wlan0

1. STA发出的probe request/response frame







实验四

1. 透明代理
2. 网卡是否支持AP模式
3. 程序后加d，代表服务器守候程序

# 创建并编辑 /etc/hostapd/hostapd.conf，内容如下实例所示

cat << EOF > /etc/hostapd/hostapd.conf

ssid=just\_a\_joke

wpa\_passphrase=

interface=wlan0

auth\_algs=3//WPA2

channel=6

driver=nl80211

hw\_mode=g

logger\_stdout=-1

logger\_stdout\_level=2

max\_num\_sta=5

rsn\_pairwise=CCMP

wpa=2

wpa\_key\_mgmt=WPA-PSK

wpa\_pairwise=TKIP CCMP

EOF

1. 用AP模式要关闭监听模式

实验五

1. WPA crack
2. 字典式破解攻击
3. 默认匹配TKIP和CCMP（包括工作模式的定义）
4. 万能wifi钥匙API接口暴露
5. 抓到握手包的情况下破解密码

Adb：

1. adb shell
2. <https://developer.android.google.cn/studio>
3. Sd卡：任意进程可读可写
4. Cat
5. Init.rc：root权限就可以修改启动加载过程
6. SEAndroid 已经过渡到 SELinux
7. 内置调试根证书
8. 工具随着版本升级
9. 数据目录不会覆盖，但也会升级（对旧的数据进行迁移，表结构重建）
10. BYOD：监控
11. TrustZone
12. 文件系统加密：文件级加密、全盘系统加密
13. Apache2、GpL授权协议
14. 内核、组件安全性问题，开源代码审计问题，私有代码安全性分析（源代码、二进制）
15. 刷机：C:\Users\47892\AppData\Local\Android\Sdk\system-images\android-19\default\x86，system.img替换（虚拟硬件要匹配）
16. 卸载应用程序：需要包名（xml里package） adb uninstall
17. 安装：adb install xx.apk
18. Adb shell
19. 考点！！：Manifest权限（链接中）

访问登记属性 android.permission.ACCESS\_CHECKIN\_PROPERTIES ，读取或写入登记check-in数据库属性表的权限

获取错略位置 android.permission.ACCESS\_COARSE\_LOCATION，通过WiFi或移动基站的方式获取用户错略的经纬度信息，定位精度大概误差在30~1500米

获取精确位置 android.permission.ACCESS\_FINE\_LOCATION，通过GPS芯片接收卫星的定位信息，定位精度达10米以内

访问定位额外命令 android.permission.ACCESS\_LOCATION\_EXTRA\_COMMANDS，允许程序访问额外的定位提供者指令

获取模拟定位信息 android.permission.ACCESS\_MOCK\_LOCATION，获取模拟定位信息，一般用于帮助开发者调试应用

获取网络状态 android.permission.ACCESS\_NETWORK\_STATE，获取网络信息状态，如当前的网络连接是否有效

访问Surface Flinger android.permission.ACCESS\_SURFACE\_FLINGER，Android平台上底层的图形显示支持，一般用于游戏或照相机预览界面和底层模式的屏幕截图

获取WiFi状态 android.permission.ACCESS\_WIFI\_STATE，获取当前WiFi接入的状态以及WLAN热点的信息

账户管理 android.permission.ACCOUNT\_MANAGER，获取账户验证信息，主要为GMail账户信息，只有系统级进程才能访问的权限

验证账户 android.permission.AUTHENTICATE\_ACCOUNTS，允许一个程序通过账户验证方式访问账户管理ACCOUNT\_MANAGER相关信息

电量统计 android.permission.BATTERY\_STATS，获取电池电量统计信息

绑定小插件 android.permission.BIND\_APPWIDGET，允许一个程序告诉appWidget服务需要访问小插件的数据库，只有非常少的应用才用到此权限

绑定设备管理 android.permission.BIND\_DEVICE\_ADMIN，请求系统管理员接收者receiver，只有系统才能使用

绑定输入法 android.permission.BIND\_INPUT\_METHOD ，请求InputMethodService服务，只有系统才能使用

绑定RemoteView android.permission.BIND\_REMOTEVIEWS，必须通过RemoteViewsService服务来请求，只有系统才能用

绑定壁纸 android.permission.BIND\_WALLPAPER，必须通过WallpaperService服务来请求，只有系统才能用

使用蓝牙 android.permission.BLUETOOTH，允许程序连接配对过的蓝牙设备

蓝牙管理 android.permission.BLUETOOTH\_ADMIN，允许程序进行发现和配对新的蓝牙设备

变成砖头 android.permission.BRICK，能够禁用手机，非常危险，顾名思义就是让手机变成砖头

应用删除时广播 android.permission.BROADCAST\_PACKAGE\_REMOVED，当一个应用在删除时触发一个广播

收到短信时广播 android.permission.BROADCAST\_SMS，当收到短信时触发一个广播

连续广播 android.permission.BROADCAST\_STICKY，允许一个程序收到广播后快速收到下一个广播

WAP PUSH广播 android.permission.BROADCAST\_WAP\_PUSH，WAP PUSH服务收到后触发一个广播

拨打电话 android.permission.CALL\_PHONE，允许程序从非系统拨号器里输入电话号码

通话权限 android.permission.CALL\_PRIVILEGED，允许程序拨打电话，替换系统的拨号器界面

拍照权限 android.permission.CAMERA，允许访问摄像头进行拍照

改变组件状态 android.permission.CHANGE\_COMPONENT\_ENABLED\_STATE，改变组件是否启用状态

改变配置 android.permission.CHANGE\_CONFIGURATION，允许当前应用改变配置，如定位

改变网络状态 android.permission.CHANGE\_NETWORK\_STATE

，改变网络状态如是否能联网

改变WiFi多播状态 android.permission.CHANGE\_WIFI\_MULTICAST\_STATE，改变WiFi多播状态

改变WiFi状态 android.permission.CHANGE\_WIFI\_STATE，改变WiFi状态

清除应用缓存 android.permission.CLEAR\_APP\_CACHE，清除应用缓存

清除用户数据 android.permission.CLEAR\_APP\_USER\_DATA，清除应用的用户数据

底层访问权限 android.permission.CWJ\_GROUP，允许CWJ账户组访问底层信息

手机优化大师扩展权限 android.permission.CELL\_PHONE\_MASTER\_EX，手机优化大师扩展权限

控制定位更新 android.permission.CONTROL\_LOCATION\_UPDATES，允许获得移动网络定位信息改变

删除缓存文件 android.permission.DELETE\_CACHE\_FILES，允许应用删除缓存文件

删除应用 android.permission.DELETE\_PACKAGES，允许程序删除应用

电源管理 android.permission.DEVICE\_POWER，允许访问底层电源管理

应用诊断 android.permission.DIAGNOSTIC，允许程序到RW到诊断资源

禁用键盘锁 android.permission.DISABLE\_KEYGUARD，允许程序禁用键盘锁

转存系统信息 android.permission.DUMP，允许程序获取系统dump信息从系统服务

状态栏控制 android.permission.EXPAND\_STATUS\_BAR，允许程序扩展或收缩状态栏

工厂测试模式 android.permission.FACTORY\_TEST，允许程序运行工厂测试模式

使用闪光灯 android.permission.FLASHLIGHT，允许访问闪光灯

强制后退 android.permission.FORCE\_BACK，允许程序强制使用back后退按键，无论Activity是否在顶层

访问账户Gmail列表 android.permission.GET\_ACCOUNTS，访问GMail账户列表

获取应用大小 android.permission.GET\_PACKAGE\_SIZE，获取应用的文件大小

获取任务信息 android.permission.GET\_TASKS，允许程序获取当前或最近运行的应用

允许全局搜索 android.permission.GLOBAL\_SEARCH，允许程序使用全局搜索功能

硬件测试 android.permission.HARDWARE\_TEST，访问硬件辅助设备，用于硬件测试

注射事件 android.permission.INJECT\_EVENTS，允许访问本程序的底层事件，获取按键、轨迹球的事件流

安装定位提供 android.permission.INSTALL\_LOCATION\_PROVIDER，安装定位提供

安装应用程序 android.permission.INSTALL\_PACKAGES，允许程序安装应用

内部系统窗口 android.permission.INTERNAL\_SYSTEM\_WINDOW，允许程序打开内部窗口，不对第三方应用程序开放此权限

访问网络 android.permission.INTERNET，访问网络连接，可能产生GPRS流量

结束后台进程 android.permission.KILL\_BACKGROUND\_PROCESSES，允许程序调用killBackgroundProcesses(String).方法结束后台进程

管理账户 android.permission.MANAGE\_ACCOUNTS，允许程序管理AccountManager中的账户列表

管理程序引用 android.permission.MANAGE\_APP\_TOKENS，管理创建、摧毁、Z轴顺序，仅用于系统

高级权限 android.permission.MTWEAK\_USER，允许mTweak用户访问高级系统权限

社区权限 android.permission.MTWEAK\_FORUM，允许使用mTweak社区权限

软格式化 android.permission.MASTER\_CLEAR，允许程序执行软格式化，删除系统配置信息

修改声音设置 android.permission.MODIFY\_AUDIO\_SETTINGS，修改声音设置信息

修改电话状态 android.permission.MODIFY\_PHONE\_STATE，修改电话状态，如飞行模式，但不包含替换系统拨号器界面

格式化文件系统 android.permission.MOUNT\_FORMAT\_FILESYSTEMS，格式化可移动文件系统，比如格式化清空SD卡

挂载文件系统 android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS，挂载、反挂载外部文件系统

允许NFC通讯 android.permission.NFC，允许程序执行NFC近距离通讯操作，用于移动支持

永久Activity android.permission.PERSISTENT\_ACTIVITY，创建一个永久的Activity，该功能标记为将来将被移除

处理拨出电话 android.permission.PROCESS\_OUTGOING\_CALLS，允许程序监视，修改或放弃播出电话

读取日程提醒 android.permission.READ\_CALENDAR，允许程序读取用户的日程信息

读取联系人 android.permission.READ\_CONTACTS，允许应用访问联系人通讯录信息

屏幕截图 android.permission.READ\_FRAME\_BUFFER，读取帧缓存用于屏幕截图

读取收藏夹和历史记录 com.android.browser.permission.READ\_HISTORY\_BOOKMARKS，读取浏览器收藏夹和历史记录

读取输入状态 android.permission.READ\_INPUT\_STATE，读取当前键的输入状态，仅用于系统

读取系统日志 android.permission.READ\_LOGS，读取系统底层日志

读取电话状态 android.permission.READ\_PHONE\_STATE，访问电话状态

读取短信内容 android.permission.READ\_SMS，读取短信内容

读取同步设置 android.permission.READ\_SYNC\_SETTINGS，读取同步设置，读取Google在线同步设置

读取同步状态 android.permission.READ\_SYNC\_STATS，读取同步状态，获得Google在线同步状态

重启设备 android.permission.REBOOT，允许程序重新启动设备

开机自动允许 android.permission.RECEIVE\_BOOT\_COMPLETED，允许程序开机自动运行

接收彩信 android.permission.RECEIVE\_MMS，接收彩信

接收短信 android.permission.RECEIVE\_SMS，接收短信

接收Wap Push android.permission.RECEIVE\_WAP\_PUSH，接收WAP PUSH信息

录音 android.permission.RECORD\_AUDIO，录制声音通过手机或耳机的麦克

排序系统任务 android.permission.REORDER\_TASKS，重新排序系统Z轴运行中的任务

结束系统任务 android.permission.RESTART\_PACKAGES，结束任务通过restartPackage(String)方法，该方式将在外来放弃

发送短信 android.permission.SEND\_SMS，发送短信

设置Activity观察其 android.permission.SET\_ACTIVITY\_WATCHER，设置Activity观察器一般用于monkey测试

设置闹铃提醒 com.android.alarm.permission.SET\_ALARM，设置闹铃提醒

设置总是退出 android.permission.SET\_ALWAYS\_FINISH，设置程序在后台是否总是退出

设置动画缩放 android.permission.SET\_ANIMATION\_SCALE，设置全局动画缩放

设置调试程序 android.permission.SET\_DEBUG\_APP，设置调试程序，一般用于开发

设置屏幕方向 android.permission.SET\_ORIENTATION，设置屏幕方向为横屏或标准方式显示，不用于普通应用

设置应用参数 android.permission.SET\_PREFERRED\_APPLICATIONS，设置应用的参数，已不再工作具体查看addPackageToPreferred(String) 介绍

设置进程限制 android.permission.SET\_PROCESS\_LIMIT，允许程序设置最大的进程数量的限制

设置系统时间 android.permission.SET\_TIME，设置系统时间

设置系统时区 android.permission.SET\_TIME\_ZONE，设置系统时区

设置桌面壁纸 android.permission.SET\_WALLPAPER，设置桌面壁纸

设置壁纸建议 android.permission.SET\_WALLPAPER\_HINTS，设置壁纸建议

发送永久进程信号 android.permission.SIGNAL\_PERSISTENT\_PROCESSES，发送一个永久的进程信号

状态栏控制 android.permission.STATUS\_BAR，允许程序打开、关闭、禁用状态栏

访问订阅内容 android.permission.SUBSCRIBED\_FEEDS\_READ，访问订阅信息的数据库

写入订阅内容 android.permission.SUBSCRIBED\_FEEDS\_WRITE，写入或修改订阅内容的数据库

显示系统窗口 android.permission.SYSTEM\_ALERT\_WINDOW，显示系统窗口

更新设备状态 android.permission.UPDATE\_DEVICE\_STATS，更新设备状态

使用证书 android.permission.USE\_CREDENTIALS，允许程序请求验证从AccountManager

使用SIP视频 android.permission.USE\_SIP，允许程序使用SIP视频服务

使用振动 android.permission.VIBRATE，允许振动

唤醒锁定 android.permission.WAKE\_LOCK，允许程序在手机屏幕关闭后后台进程仍然运行

写入GPRS接入点设置 android.permission.WRITE\_APN\_SETTINGS，写入网络GPRS接入点设置

写入日程提醒 android.permission.WRITE\_CALENDAR，写入日程，但不可读取

写入联系人 android.permission.WRITE\_CONTACTS，写入联系人，但不可读取

写入外部存储 android.permission.WRITE\_EXTERNAL\_STORAGE，允许程序写入外部存储，如SD卡上写文件

写入Google地图数据 android.permission.WRITE\_GSERVICES，允许程序写入Google Map服务数据

写入收藏夹和历史记录 com.android.browser.permission.WRITE\_HISTORY\_BOOKMARKS，写入浏览器历史记录或收藏夹，但不可读取

读写系统敏感设置 android.permission.WRITE\_SECURE\_SETTINGS，允许程序读写系统安全敏感的设置项

读写系统设置 android.permission.WRITE\_SETTINGS，允许读写系统设置项

编写短信 android.permission.WRITE\_SMS，允许编写短信

写入在线同步设置 android.permission.WRITE\_SYNC\_SETTINGS，写入Google在线同步设置

1. R.java：自动生成
2. 测试和发布版本区别：使用的签名文件不一样
3. Android App基本组件使用

Apk重命名——>解压缩——>

1. adb shell pm list package：已安装的包名

持久化存储：MainActivity

UserConfig.xml

内部存储 外部存储

差别：文件系统类型差别

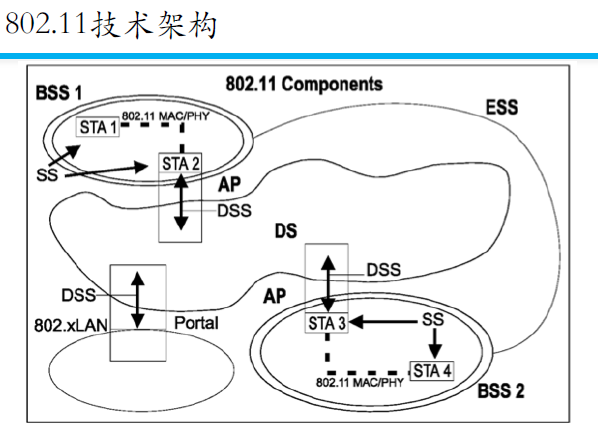
升级安装 复制安装(version:1,2)

魔术变量

移动互联网安全

1. 无线网络安全基础
2. 无线网络通信协议中UWB和WiMax协议已经过时
3. ZigBee：个人短距离。ZigBee是基于IEEE802.15.4标准的低功耗局域网协议。根据国际标准规定，ZigBee技术是一种短距离、低功耗的无线通信技术。其特点是近距离、低复杂度、自组织、低功耗、低数据速率。主要适合用于自动控制和远程控制领域，可以嵌入各种设备。简而言之，ZigBee就是一种便宜的，低功耗的近距离无线组网通讯技术。ZigBee是一种低速短距离传输的无线网络协议。ZigBee协议从下到上分别为物理层(PHY)、媒体访问控制层(MAC)、传输层(TL)、网络层(NWK)、应用层(APL)等。其中物理层和媒体访问控制层遵循IEEE 802.15.4标准的规定。

3、



\*\*802.11体系结构的组成包括：无线站点STA（station），无线接入点AP（access point），独立基本服务组IBSS（independent basic service set），基本服务组BSS（basic service set），分布式系统DS（distribution system）和扩展服务组ESS（extended service set）。

STA：具备联网功能的设备。一个无线站点STA通常由一台PC机或笔记本计算机加上一块无线网卡构成

BSS：相对独立的无线网络，基本服务集（BSS）。基本服务集是802.11 LAN的基本组成模块。能互相进行无线通信的STA可以组成一个BSS（Basic Service Set） 。如果一个站移出BSS的覆盖范围，它将不能再与BSS的其它成员通信。对于个人PC来说，使用最多的所谓"无线Wi-Fi"指的就是BSS网络模式，我们通过AP(Access Point)接入点来接入网络

DS：链路层，相当于switch。连接BSS的组件称为分布式系统（Distribution System，DS）。ESS中的DS(分布式系统)是一个抽象系统，用来连接不同BSS的通信信道(通过路由服务)，这样就可以消除BSS中STA与STA之间直接传输距离受到物理设备的限制。

Portal：门户认证

DSS：连接地面网络。发布服务系统。提供的服务是一端连接到BSS，另一端延伸到网络中任意可能的端点中去，来实现逻辑和物理区域的网间互联。

802.11 MAC/PHY：多台ap级联，跨局域网通信

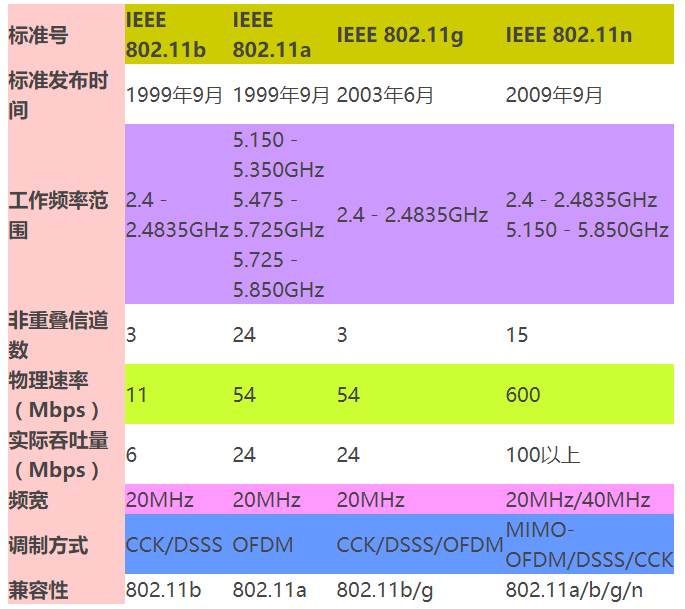
ESS：扩展服务集（ESS），多个BSS可以构成一个扩展网络，称为扩展服务集（ESS）网络，一个ESS网络内部的STA可以互相通信，是采用相同的SSID的多个BSS形成的更大规模的虚拟BSS。

SSID/BSSID：SSID，服务集的标识，在同一SS内的所有STA和AP必须具有相同的SSID，否则无法进行通信。SSID是一个ESS的网络标识(如:TP\_Link\_1201)，BSSID是一个BSS的标识，BSSID实际上就是AP的MAC地址，用来标识AP管理的BSS，在同一个AP内BSSID和SSID一一映射。在一个ESS内SSID是相同的，但对于ESS内的每个AP与之对应的BSSID是不相同的。如果一个AP可以同时支持多个SSID的话，则AP会分配不同的BSSID来对应这些SSID。

IBSS：BSS的一种特殊结构，这种结构内没有提供DSS功能的STA，所有的STA都自组实现对等通信。

AD Hoc模式：可以在无AP的环境下实现两台或多台STA的对等通讯功能

4、802.11简介



5、频谱划分

1）IEEE 802.11b/g标准工作在2.4G频段，频率范围为2.400—2.4835GHz，共83.5M带宽

2）划分为14个子信道

3）每个子信道宽度为22MHz

4）相邻信道的中心频点间隔5MHz

5）相邻的多个信道存在频率重叠(如1信道与2、3、4、5信道有频率重叠)

6）整个频段内只有3个（1、6、11）互不干扰信道

6、802.11协议

802.11

IEEE最初制定的一个无线局域网标准，主要用于解决办公室局域网和校园网中用户与用户终端的无线接入，业务主要限于数据存取，速率最高只能达到2Mbps。由于它在速率和传输距离上都不能满足人们的需要，因此，IEEE小组又相继推出了802.11b和802.11a两个新标准。

802.11a

802.11a标准工作在5GHzU-NII频带，物理层速率最高可达54Mbps，传输层速率最高可达25Mbps。可提供25Mbps的无线ATM接口和10Mbps的以太网无线帧结构接口，以及TDD/TDMA的空中接口；支持语音、数据、图像业务；一个扇区可接入多个用户，每个用户可带多个用户终端。

根据需要，数据率还可降为48，36，24，18，12，9或者6Mb/s。802.11a拥有12条不相互重叠的频道，8条用于室内，4条用于点对点传输。它不能与802.11b进行互操作，除非使用了对两种标准都采用的设备。

802.11b

IEEE802.11b是无线局域网的一个标准。其载波的频率为2.4GHz，传送速度为11Mbit/s。IEEE802.11b是所有无线局域网标准中最著名，也是普及最广的标准。它有时也被错误地标为Wi-Fi。实际上Wi-Fi是无线局域网联盟（WLANA）的一个商标，该商标仅保障使用该商标的商品互相之间可以合作，与标准本身实际上没有关系。在2.4-GHz-ISM频段共有14个频宽为22MHz的频道可供使用。IEEE802.11b的后继标准是IEEE802.11g，其传送速度为54Mbit/s。

802.11g

IEEE 802.11g2003年7月，通过了第三种调变标准。其载波的频率为2.4GHz（跟802.11b相同），原始传送速度为54Mbit/s，净传输速度约为24.7Mbit/s（跟802.11a相同）。802.11g的设备与802.11b兼容。802.11g是为了提高更高的传输速率而制定的标准，它采用2.4GHz频段，使用CCK技术与802.11b后向兼容，同时它又通过采用OFDM技术支持高达54Mbit/s的数据流，所提供的带宽是802.11a的1.5倍。从802.11b到802.11g，可发现WLAN标准不断发展的轨迹：802.11b是所有WLAN标准演进的基石，未来许多的系统大都需要与802.11b向后向兼容，802.11a是一个非全球性的标准，与802.11b后向不兼容，但采用OFDM技术，支持的数据流高达54Mbit/s，提供几倍于802.11b/g的高速信道，如802.11b/g提供3个非重叠信道可达8-12个；可以看出，在802.11g和802.11a之间存在与Wi-Fi兼容性上的差距，为此出现了一种桥接此差距的双频技术——双模（dual band)802.11a+g(=b），它较好地融合了802.11a/g技术，工作在2.4GHz和5GHz两个频段，服从802.11b/g/a等标准，与802.11b后向兼容，使用户简单连接到现有或未来

802.11n

IEEE802.11n，2004年1月IEEE宣布组成一个新的单位来发展新的802.11标准。资料传输速度估计将达475Mbps（需要在物理层产生更高速度的传输率），此项新标准应该要比802.11b快45倍，而比802.11g快8倍左右。802.11n也将会比之前的无线网络传送到更远的距离。的802.11网络成为可能。

802.11ac

主流厂商（Qualcomm，Broadcom，Intel等）正在开发的协议版本，它使用5GHz频段（也可以说是6GHz频段），采用：更宽的基带（最高扩展到160Mhz）、更多的MIMO、高密度的调制解调（256 QAM）。理论上，11ac可以为多个站点服务提供1Gbit的带宽，或是为单一连接提供500Mbit的传输带宽。支持千兆网络

世界上第一只采用802.11ac无线技术的路由器,于2011年11月15日, 由美国初创公司Quantenna推出了。2012年1月5日，业界巨头Broadcom发布了它的第一款支持802.11ac的芯片。

802.11i

IEEE802.11i是IEEE为了弥补802.11脆弱的安全加密功能（WEP,Wired Equivalent Privacy）而制定的修正案，于2004年7月完成。其中定义了基于AES的全新加密协议CCMP（CTR with CBC-MAC Protocol），以及向前兼容RC4的加密协议TKIP（Temporal Key Integrity Protocol）。

无线网络中的安全问题从暴露到最终解决经历了相当的时间，而各大厂通信芯片商显然无法接受在这期间什么都不出售，所以迫不及待的Wi-Fi厂商采用802.11i的草案3为蓝图设计了一系列通信设备，随后称之为支持WPA（Wi-Fi Protected Access）的；之后称将支持802.11i最终版协议的通信设备称为支持WPA2（Wi-Fi Protected Access 2）的。

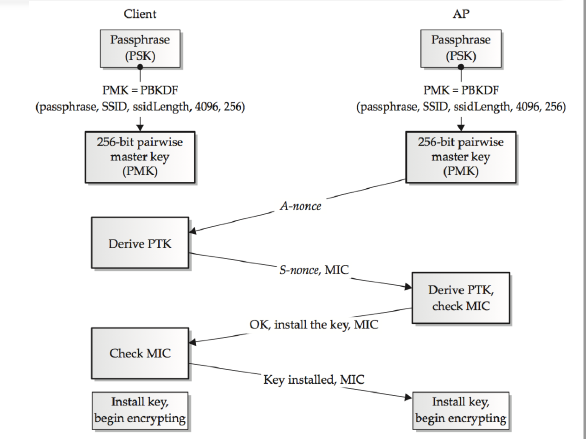
1. 传输时协议的属性包含在beacon frame里
2. 802.11的三种工作模式具有排他性，混杂模式下也为监听模式，不需要建立连接
3. 路由器：交换机具有网络功能
4. man + 命令名称：查阅命令帮助信息
5. 无线接入网监听
6. dhcp无法获取ip地址，就无法建立网络层连接
7. 2.4G：1~13 channel
8. 无线网卡可以工作在多种模式之下。常见的有Master，Managed，ad-hoc，monitor等模式。
9. 对于Master模式，它主要使用于无线接入点AP提供无线接入服务以及路由功能。可以想象我们使用的无线路由器就是工作在Master模式下了，不过对于普通的pc机来说，如果有合适的硬件它也可以变成一台无线AP。在LINUX下使用软件HOSTAP可以使一台linux pc具有ap功能，但目前HOSTAP的所支持的无线网卡有限，我的INTEL Centrino PRO 4965 wireless card就不被hostap所支持。对于一般的无线网卡来说，它们最常见的模式还是managed，ad-hoc和monitor。Managed模式用于和无线AP进行接入连接，在这个模式下我们才可以进行无线接入internet上网。对于需要两台主机进行直连的情况下可以使用ad-hoc模式，这样主机之间是采用对等网络的方式进行连接。Monitor模式主要用于监控无线网络内部的流量，用于检查网络和排错。
10. 连接服务：存储转发的过程
11. 当ap过多时网速时快时慢，这与冲突检测有关
12. 隐藏节点和暴露节点问题

隐藏节点：比如说网络中有三个节点A,B,C。A,C都想传数据给B，但A,C都不在彼此的传输范围内。因而也就不会感知到对方，但都能向B传输数据，在这种情况下，来自A & C的数据会在B处碰撞，造成数据丢失，网络性能下降。这样因为传送距离而发生误判的问题称为隐藏节点问题。

暴露节点：当有一个节点要发送数据给另一个节点，但因为邻居节点也正发送数据时，因此影响了原本节点的数据传送。如有四个节点R1， S1，S2，R2,但R1，R2不在彼此的传送范围内，而s1和s2，r1和s1，s2和r2都在彼此的传输范围内。当s1传数据给r1时，s2却不能传数据给r2,此时s2检测到s1正在传送数据，就会影响s1传送，事实上s2可以将数据传送到r2，因为r2不在s1的传输范围内。

可以用RTS（request to send）和CTS(clear to send)的控制封包来避免碰撞。在传输之前，传送端先传送一个RTS封包，告知在传送端传送范围内所有节点不要有任何传送操作。如果接收端目前是空闲的，则响应CTS封包，进而传输数据。从而可有效解决隐藏节点的问题。

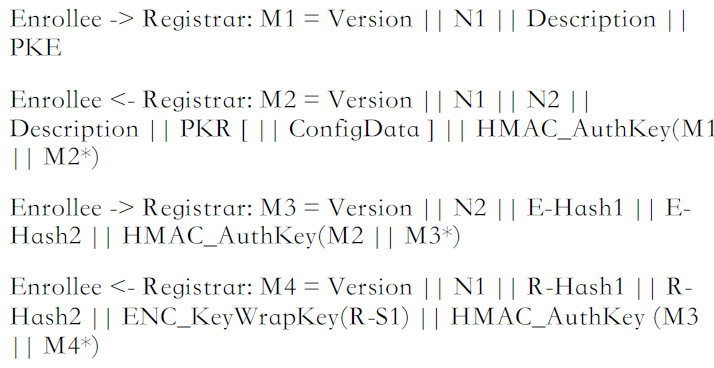
1. wep非常不安全
2. 向前兼容：backward
3. 四次握手认证



1. PBKDF：密码算法
2. Passphrase：明文口令
3. 4096，256：输出长度和迭代次数
4. PMK：随机数
5. Install the key：密钥每隔时间更换会话加密密钥，给攻击留出可能
6. Client前两步在本地完成
7. wps工作模式中

带内传输：带内传输（intraband transmission）是指一种利用话路频带内的一段窄频带，使电话与时间离散信号得以同时传输的传输方式。主数据协议以外的传输，和主数据协议位于同样的传输介质之上。管理协议是一种带内传输的例子。

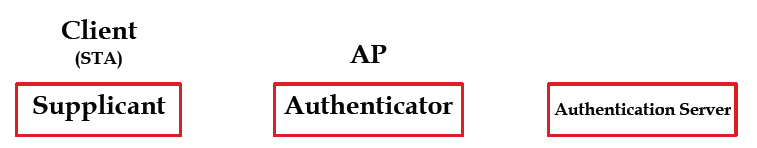
带外传输：使用一个分离的控制链接，控制信息和数据传输由不同的进程控制。数据传输机制和密钥协商不在一个信道内



抓包可以获得前四次数据包

HMAC——AuthKey：校验消息完整性

1. 注册消息协议版本号会决定如何发送和解析数据
2. 随机数防止重放攻击
3. 802.11状态机中第一步需查看是否经过认证和是否进入关联状态
4. 802.1x技术架构



Supplicant：提供认证凭据（STA包含路由器）

Authentication：完成验证逻辑（在数据库中查询计算）

Authentication Server：比如数据库，在物理层

17、802.11中提供STA和认证服务器之间的端到端认证

功能：收和转发，不会修改数据包，保证端到端通信

1. 服务端证书保证服务器认证客户端真实身份，有可能遭受降级攻击
2. 根ca预制在操作系统或软件内部，是离线过程
3. U盾存储客户端安全证书
4. 自签发证书问题：无法区分是攻击者构造的还是自己构造的证书
5. TLS对称加密主密钥不在AP上存储，但要在传输过程中协商

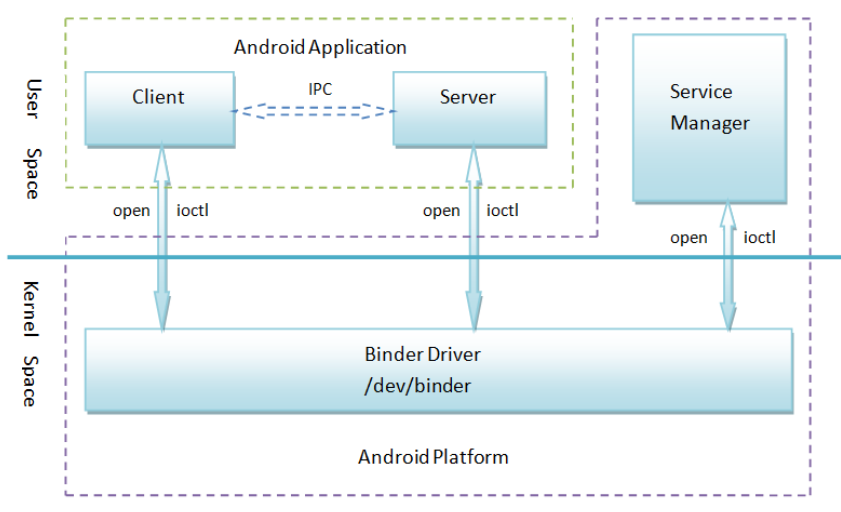
第六章

Android进程间通信（IPC）机制Binder简要介绍

Android系统没有采用上述提到的各种进程间通信机制，而是采用Binder机制，难道是因为考虑到了移动设备硬件性能较差、内存较低的特点？不得而知。Binder其实也不是Android提出来的一套新的进程间通信机制，它是基于OpenBinder来实现的。OpenBinder最先是由Be Inc.开发的，接着Palm Inc.也跟着使用。现在OpenBinder的作者Dianne Hackborn就是在Google工作，负责Android平台的开发工作。

前面一再提到，Binder是一种进程间通信机制，它是一种类似于COM和CORBA分布式组件架构，通俗一点，其实是提供远程过程调用（RPC）功能。从英文字面上意思看，Binder具有粘结剂的意思，那么它把什么东西粘结在一起呢？在Android系统的Binder机制中，由一系统组件组成，分别是Client、Server、Service Manager和Binder驱动程序，其中Client、Server和Service Manager运行在用户空间，Binder驱动程序运行内核空间。Binder就是一种把这四个组件粘合在一起的粘结剂了，其中，核心组件便是Binder驱动程序了，Service Manager提供了辅助管理的功能，Client和Server正是在Binder驱动和Service Manager提供的基础设施上，进行Client-Server之间的通信。Service Manager和Binder驱动已经在Android平台中实现好，开发者只要按照规范实现自己的Client和Server组件就可以了。说起来简单，做起难，对初学者来说，Android系统的Binder机制是最难理解的了，而Binder机制无论从系统开发还是应用开发的角度来看，都是Android系统中最重要的组成，因此，很有必要深入了解Binder的工作方式。要深入了解Binder的工作方式，最好的方式莫过于是阅读Binder相关的源代码了，Linux的鼻祖Linus Torvalds曾经曰过一句名言RTFSC：Read The Fucking Source Code。

总结一下，Android系统Binder机制中的四个组件Client、Server、Service Manager和Binder驱动程序的关系如下图所示：



1. Client、Server和Service Manager实现在用户空间中，Binder驱动程序实现在内核空间中

2. Binder驱动程序和Service Manager在Android平台中已经实现，开发者只需要在用户空间实现自己的Client和Server

3. Binder驱动程序提供设备文件/dev/binder与用户空间交互，Client、Server和Service Manager通过open和ioctl文件操作函数与Binder驱动程序进行通信

4. Client和Server之间的进程间通信通过Binder驱动程序间接实现

5. Service Manager是一个守护进程，用来管理Server，并向Client提供查询Server接口的能力