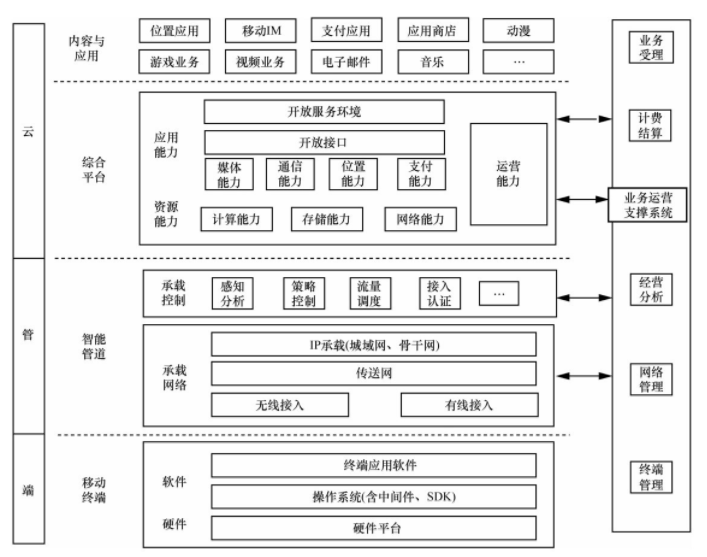
第一章 无线网络安全基础

1. 移动互联网基础

关于移动互联网具体是什么，包含哪些主要技术组件，一种普遍被认可的架构定义为：云、管、端三层架构，如下图所示：



本课程的主要内容聚焦在：管和端，更具体来说，我们将重点讲解无线接入（以下主要使用无线网络这个术语来指代这里的无线接入技术）和移动终端的相关安全问题。

2. 无线网络

提到无线网络，你的脑海里会闪现出哪些关键词？

Wi-Fi？WLAN？802.11？蓝牙？NFC？

以下是这些与无线网络技术相关的关键词简介，这些技术共同组成了无线网络构建的通信协议基础。

Wi-Fi

WiFi的名字，最早是出现在1999年，是WiFi联盟(当时不叫做WiFi联盟)雇佣当时的一个商标咨询公司Interband，替他们为“IEEE 802.11b direct sequence”起一个更简洁更具吸引力的名字，而当时WiFi联盟的创始成员(Phil Belanger)主持的“WiFi”的这个名字最后胜出，被称之为WiFi名字的发明人。Interband在起名字之后又设计了WiFi的logo，用于这个WiFi logo的产品意味着兼容WiFi技术的产品。而后来文献中误传WiFi是WirelessFidelity的缩写，这是由于WiFi联盟自己曾在广告宣传的时候使用了“The Standard Wireless Fidelity”的广告标语，但其实WiFi最开始的产生是没有任何意义的，根据Interband的说法，只是hi-fi的一种语音合成或者双关。 所以可以看出来，WiFi最开始是遵从IEEE 802.11b标准的一种通信技术，和蓝牙同级，但同时也是一个商标。

WLAN

无线局域网络(Wireless Local Area Networks, WLAN)是相当便利的数据传输系统，它利用射频(Radio Frequency, RF)的技术，取代旧式碍手碍脚的双绞铜线(Coaxial)所构成的局域网络，使得无线局域网络能利用简单的存取架构让用户透过它，达到“信息随身化、便利走天下”的理想场景。目前无线局域网络最为广泛使用的标准是IEEE 802.11协议标准。当然还有别的标准，如IEEE 802.15，蓝牙（Bluetooth）使用的就是这个标准。

802.11

802.11是IEEE 802.11 Task Group制定的一系列无线局域网标准，最初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。随着技术的发展，该系列标准演进出了许多细分标准，网络传输速率也已经伴随着802.11ac的制定和发布，正式宣告进入千兆网络时代。

蓝牙

蓝牙技术是一种开放式无线通讯标准，能够在短距离范围内无线连接桌上型电脑与笔记本电脑、便携设备、PDA、移动电话、拍照手机、打印机、数码相机、耳麦、键盘甚至是电脑鼠标。蓝牙无线技术使用了全球通用的免费频带（2.4GHz），以确保能在世界各地通行无阻。简言之，蓝牙技术让各种数码设备之间能够无线沟通，让散落各种连线的桌面成为历史。

NFC

近场通信（英语：Near Field Communication，NFC），又称近距离无线通信，是一种短距离的高频无线通信技术，允许电子设备之间进行非接触式点对点数据传输，在十厘米（3.9英吋）内交换数据。这个技术由非接触式射频识别（RFID）演变而来，由飞利浦半导体（现恩智浦半导体）、诺基亚和索尼共同研制开发，其基础是RFID及互连技术。近场通信是一种短距高频的无线电技术，在13.56MHz频率运行于20厘米距离内。目前近场通信已通过成为ISO/IEC IS 18092国际标准、EMCA-340标准与ETSI TS 102 190标准。NFC采用主动和被动两种读取模式。

2.1. 无线网络中的常见设备

在无线局域网里，常见的网络基础设施设备有无线网卡、无线网桥、AP、路由器等。

AP

无线访问接入点（Wireless Access Piont, AP）相当于一个连接有线网和无线网的桥梁，其主要作用是将各个无线网络客户端连接到一起，然后将无线网络接入以太网(这正是Access Point名称的本义)。AP又被称为无线交换机，这让它和无线路由器更容易区分角色和应用于不同的使用场景。

目前大多数的无线AP都支持多用户接入、数据加密、多速率发送等功能，一些产品更提供了完善的无线网络管理功能。对于家庭、办公室这样的小范围、小规模（20台终端设备以下）无线局域网而言，一般只需一台无线AP即可实现所有计算机的无线接入。

AP的室内覆盖范围一般是30米 ～100米，目前不少厂商的AP产品可以互联，以增加无线网络覆盖面积。也正因为每个AP的覆盖范围都有一定的限制，正如手机可以在基站之间漫游一样，无线局域网客户端也可以在AP之间漫游。

路由器

路由器（Router）是连接互联网中各局域网、广域网的设备，它会根据信道的情况自动选择和设定路由，以最佳路径、先后顺序发送信号。 路由器是互联网络的枢纽、"交通警察"。

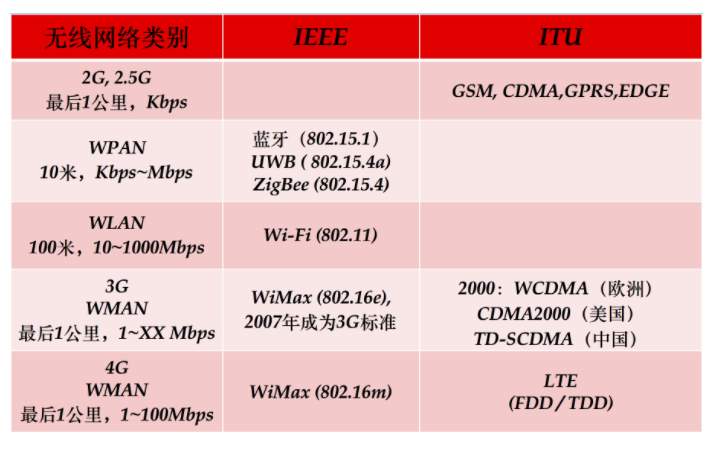
电力猫

电力线通信（Power Line Communication, PLC）技术是指利用电力线传输数据和媒体信号的一种通信方式。该技术是把载有信息的高频加载于电流然后用电线传输接受信息的适配器再把高频从电流中分离出来并传送到计算机或电话以实现信息传递。

电力线通信全称是电力线载波（Power Line Carrier, PLC）通信，是指利用高压电力线（在电力载波领域通常指35kV及以上电压等级）、中压电力线（指10kV电压等级）或低压配电线（380/220V用户线）作为信息传输媒介进行语音或数据传输的一种特殊通信方式。

电力猫即“电力线通讯调制解调器”，是通过电力 线进行宽带上网的Modem的俗称。使用家庭或办公室现有电力线和插座组建成网络，来连接PC，ADSL modem，机顶盒，音频设备， 监控设备以及其他的智能电气设备，来传输数据，语音和视频。它具有即插即用的特点，能通过普通家庭电力线传输网络IP数字信号。

除了这些构建网络的基础设施设备，和有线网络一样，无线网络里数量更为庞大的还是网络终端设备。从常见的手机、平板电脑、笔记本、台式机等“计算”设备，到物联网兴起、智能家居发展起来后开始上网的冰箱、空调、插座等等，这些终端设备通过标准的无线网络接入协议接入到无线网络。

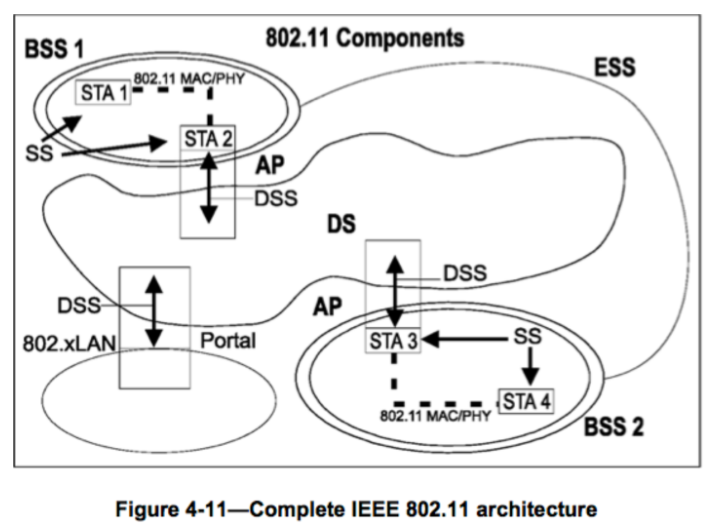


如上图所示是无线网络通信协议家谱，这里列举了常见的标准无线网络接入协议。

至此，我们初步认识到了无线网络的主要组成部分。

3. IEEE 802.11

3.1. 相关术语



上图来自《IEEE Standard for Information technology — Telecommunications and information exchange between systems Local and metropolitan area networks— Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications》，该图描述了一个典型的基于IEEE 802.11技术构建的无线局域网所包含的主要组件。下面我们将来逐个学习这些组件术语。

（1）STA，又称为Wireless Station（基站），指的是可以使用802.11协议的设备。由于802.11协议解决的是无线网络构建的物理层和数据链路层协议标准问题，因此所有的STA都是可寻址的设备，这些设备可能有固定地址、可移植地址、动态地址。无线客户端、无线路由器和无线接入点都是STA。

（2）基本服务群，(Basic Service Set, BSS)，是STA加入的集合，STA之间是否可以相互可见（通信）取决于接入点设备的配置。基本服务区域（Basic Service Area, BSA)是指BSS所覆盖的范围。

（3）IBSS这里的I指代的是Independent，所以IBSS用于指代的是独立BSS。BSS默认就是指的基础设施BSS(Infrastructure BSS)，基础设施BSS需要至少包含一个接入点，Portal（门户）是可选项（无线服务的逻辑接入点，例如学校的无线认证页面）。IBSS模式没有无线基础设施设备，但最少要包含2个STA，且不支持接入分布式系统。

（4）Station服务（Station Service, SS）在一个指定BSS内提供数据链路层通信服务。

（5）扩展服务集合（Extended Service Set, ESS）是由物理上独立的BSS组成的一个逻辑上的独立服务集合。ESS也有一个标识的名称，即ESSID。扩展服务区ESS是指由多个AP以及连接它们的分布式系统组成的结构化网络，所有AP必需共享同一个ESSID。其中包括对CMCC、ChinaNet等无线热点的扩展，同样也支持漫游。

（6）分布式系统（Distribution System，DS）可以通过AP间的无线连接实现BSS连接，从而增加BSS的覆盖范围（面积）。无线DS又被称为WDS（Wireless DS）。DSS的全称是Distribution System Service。

（7）AP

根据前文STA的定义可知，AP也属于STA。AP对已关联（associated）的STA基于⽆线介质（Wireless Medium, WM）提供接⼊分布式服务。

（8）BSSID

BSS模式下，BSSID是AP的MAC地址，对于IBSS模式来说，BSSID是随机产生的MAC地址。

（9）ESSID

ESSID⼀般在网络产品中被简称为SSID，是⽆线⽹络的接⼊唯⼀标识。需要注意的是，ESSID仅作为⼀种网络声明方式，在相关协议标准中并没有严格保证如何达到网络的唯一标识。事实上，ESSID是可以被任意声明（伪造）。

所谓无线网络的接入唯一标识指的是⼯作站与AP关联（Associate）使用的标识，按照802.11标准定义：ESSID取值区分⼤小写、2-32字节。

单个AP可以支持同时声明多个ESSID，该特性取决于设备与系统的支持情况。常见的多ESSID应用场景是创建一个与主要网络隔离的客⼈⽹络。

STA发现无线网络的方式分为主动和被动两种，其中被动发现方式依赖于AP在运行时定时不断广播ESSID，其他STA通过扫描不同频段监听对应广播数据的方式被动发现该无线网络标识。如果AP取消了ESSID广播，其他STA则无法通过前述被动监听方式“主动发现”网络。有些人认为这种“设置AP取消SSID广播”的方式可以避免攻击者发现网络“入口”，从而避免攻击者的无线入侵，增强无线网络安全性。在这里，我们先给出结论：

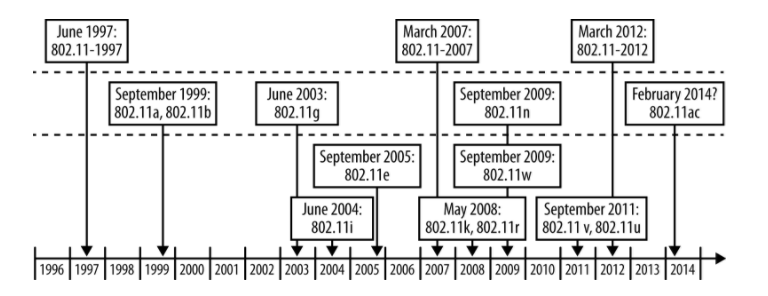
取消SSID广播，并不能增强无线网络安全性。

具体原因和原理，我们将在第三章详细讲解。

多个AP在构建ESS时，所有的BSS（AP或无线路由器）必须使用相同的ESSID。那么有了ESSID的情况下，BSSID的作用是什么呢？答案是：增强识别BSS的准确性。例如，对于同名ESSID的场景，通过BSSID可以区分不同的BSS。

下面这个例子来自真实世界的无线网络抓包结果：

3.2. 系列标准



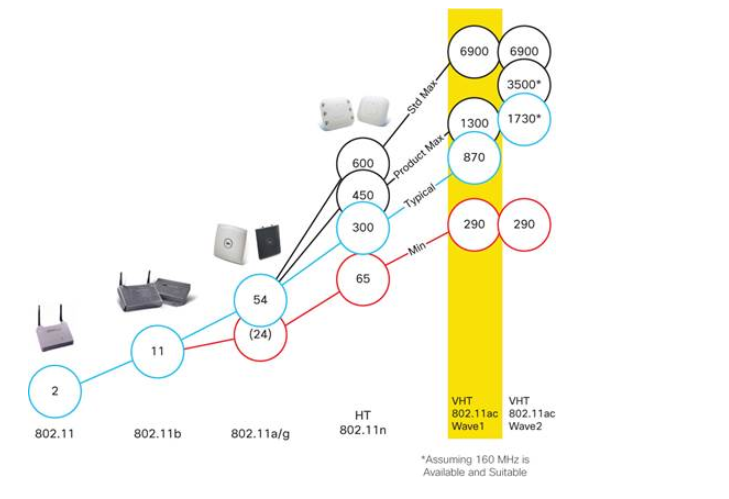
如上图所示，记录了802.11协议家族的相关协议标准的发布时间节点历史时间线。

802.11b和802.11g将2.4 GHz的频段区分为14个重复，标记的信道;每个信道的中心频率相差5兆赫兹（MHz）。一般常常被误认的是信道1，6和11（还有有些地区的信道14）是互不重叠所以利用这些不重叠的信道，多组无线网络的互相涵盖，互不影响，这种看法太过简单。802.11b和802.11g并没有规范每个信道的频宽，规范的是中心频率和频谱屏蔽（spectral mask）。802.11b的频谱屏蔽需求为：在中心频率±11 MHz处，至少衰减30 dB，±22 MHz处要衰减50 dB.

由于频谱屏蔽只规定到±22 MHz处的能量限制，所以通常认定使用频宽不会超过这个范围。实际上，当发射端距离接收端非常近时，接收端接受到的有性能量频谱，有可能会超过22 MHz的区域。所以，一般认定信道1，6和11互不重叠的说法。应该要修正为：信道1，6和11，三个频段互相之间的影响比使用其它频段小。然而，要注意的是，一个使用信道1的高功率发射端，可以轻易的干扰到一个使用信道6的，功率较低的发射站。在实验室的测试中发现，当使用信道11来传递档案时，一个使用信道1的发射台也在通讯时，会影响到信道11的档案传输，让传输速率稍稍降低。所以，即使是频段相差最远的信道1和11，也是会互相干扰的。

虽然信道1、6和11互不重叠的说法是不正确的，但是这个说法至少可以用来说明：信道距离在1，6和11之间虽然会对彼此造成干扰，而却不会强烈地影响到通讯的传输速率。下图是2.4GHz频段802.11g规范中第1-14信道的频谱屏蔽范围定义。

很多国家都有法律规管这些信道的使用，例如在一定频率范围内的最大功率电平等。网络运营商应咨询当地主管部门，因为这些规定可能会过时，世界上绝大多数国家都允许不需要申请许可证使用第1信道到第13信道



如上图所示，记录了802.11协议主要核心标准对应的最大带宽能力发展历史时间线。

3.3. 802.x LAN

局域网（Local Area Network, LAN），又称内网。指覆盖小范围、局部区域（如办公室或楼层）的计算机网络。按照网络覆盖的区域（距离）不同，其他的网络类型还包括个人网（Personal Area Network, PAN）、城域网（Metropolitan Area Network, MAN）、广域网（Wide Area Network, WAN）等。

早期的局域网网络技术都是各不同厂家所专有，互不兼容。后来，IEEE（电机电子工程师学会）推动了局域网技术的标准化，由此产生了IEEE 802系列标准。这使得在建设局域网时可以选用不同厂家的设备，并能保证其兼容性。这一系列标准覆盖了双绞线、同轴电缆、光纤和无线等多种传输媒介和组网方式，并包括网络测试和管理的内容。随着新技术的不断出现，这一系列标准仍在不断的更新变化之中。

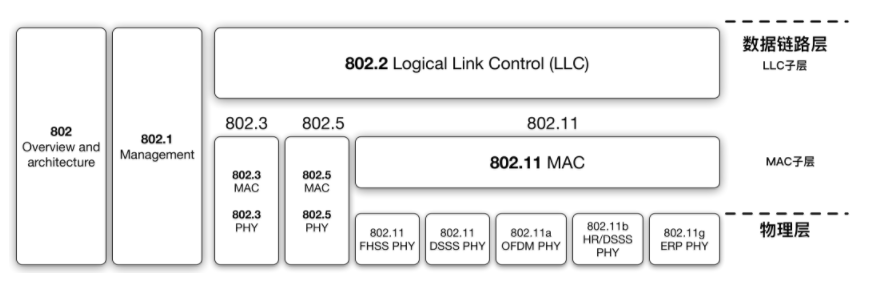
以太网（IEEE 802.3标准）是最常用的局域网组网方式。以太网使用双绞线作为传输媒介。在没有中继的情况下，最远可以覆盖200米的范围。最普及的以太网类型数据传输速率为100Mb/s，更新的标准则支持1000Mb/s和10000Mb/s的速率。

其他主要的局域网类型有令牌环（Token Ring）和光纤分布数字接口（FDDI，IEEE 802.8）。令牌环网络采用同轴电缆作为传输媒介，具有更好的抗干扰性；但是网络结构不能很容易的改变。FDDI采用光纤传输，网络带宽大，适于用作连接多个局域网的骨干网。

近两年来，随着802.11标准的制定，无线局域网的应用大为普及。这一标准采用2.4GHz 和5GHz 的频段，数据传输速度可以达到11Mbps、54Mbps和1Gbps，覆盖范围为100米。

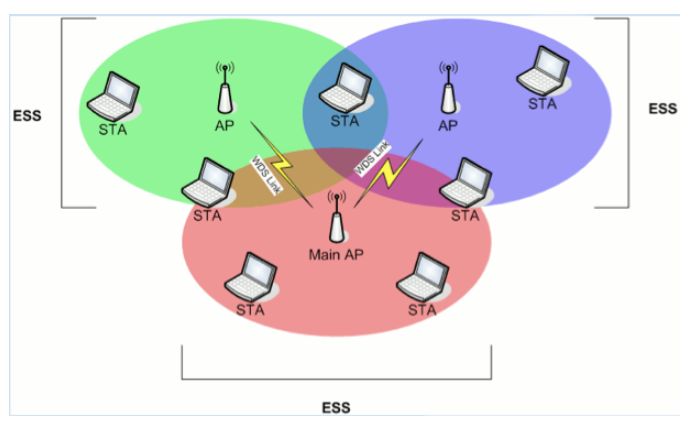
局域网标准定义了传输媒介、编码和介质访问等底层（一二层）功能。要使数据通过复杂的网络结构传输到达目的地，还需要具有寻址、路由和流量控制等功能的网络协议的支持。TCP/IP（传输控制协议/互联网络协议）是最普遍使用的局域网网络协议。它也是互联网所使用的网络协议。其他常用的局域网协议包括，IPX、AppleTalk等。

如下图所示概括了IEEE 802家族主要协议及其对应OSI模型的归属关系。

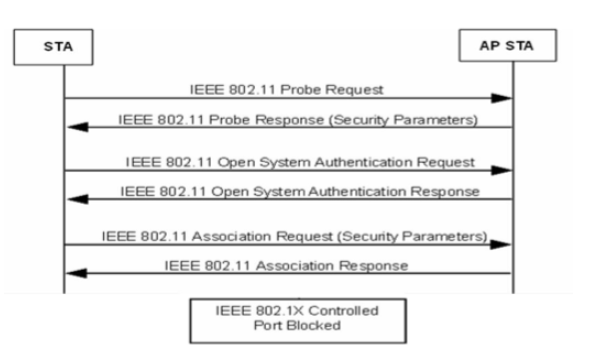


3.4. 802.11 工作模式

802.11协议标准定义了STA的三种主要工作模式：基础设施、Ad-Hoc和监听模式。其中基础设施模式下的网络类型包括BSS和ESS，Ad-Hoc模式相当于是IBSS模式的别名。监听模式下，STA不发射任何无线信号，只被动的接收无线数据帧，我们在第二章会主要讲解监听模式。如下图所示是一个典型的ESS模式网络拓扑。

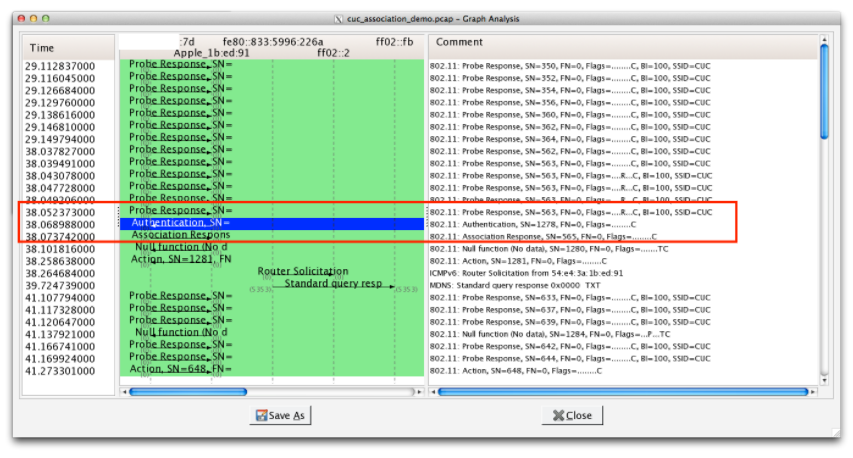


3.5. 802.11关联过程



上图是一个典型的无加密、开放认证模式下的STA关联（加入）到一个无线网络的交互式过程示意图。这张图上展示的过程跳过了STA首次加入无线网络时的获取AP的广播beacon frame过程，在AP的beacon frame中STA也能找到当前网络的认证和安全配置相关参数信息。

下图来自使用Wireless对一次真实无线网络加入过程的抓包结果的可视化展示：注意相比于我们上面的示意图，此次真实入网过程还包含一个“空”的认证请求frame。



STA除了通过“被动”方式发现正在广播的无线网络之外，不同的操作系统和无线网络管理应用程序会在802.11协议标准的建议基础之上，设计自己的STA加入（关联）AP策略。最常见的策略按照优先级从高到低，如下定义：

历史关联记录优先

先发现，先关联

信号强度⾼者优先

对于无线信号强度的定义，最常见的指标是接收信号强度指示器（Received Signal Strength Indicator, RSSI），这是接收信号的强度指示。

在IEEE 802.11系统中，RSSI表征的是一个无线网络环境的相对接收信号强度，可以使用任意量纲。RSSI代表的是接收到的信号经由天线和可能的有线损耗之后的功率等级，因此，RSSI的数值越大，说明信号强度越强。考虑到大部分情况下，RSSI取值为负数（例如：-100），因此取值越接近于0，说明接收到的信号越强。

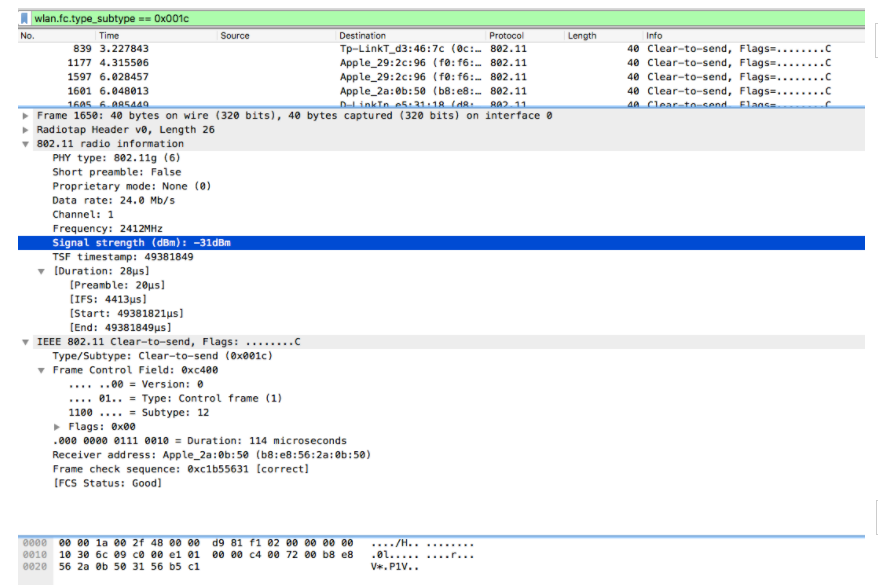
网卡内部可以设定在特定频段的RSSI低于某个特定阈值时来发送CTS数据帧。在CTS数据帧中我们可以观测到当前网络的瞬时RSSI。不同网卡内部的RSSI取值范围会有差异。例如Cisco的网卡使用的RSSI取值范围是从0到100，Atheros的网卡使用的RSSI取值范围是从0到127，其中128表示非法取值。

没有协议标准定义了任何物理参数与RSSI取值的关系。802.11协议没有定义RSSI取值使用毫瓦功率等级或与1毫瓦的比值。网卡芯片厂商制定自己的准确度、精度和取值范围来表示实际接收功率（例如常见使用毫瓦或毫瓦比值）。802.11协议中与RSSI有关的规定只有功率抽样方法：RSSI只有在接收802.11数据帧的前导（preamble）阶段进行测量计算，而不是在接收到完整数据帧的时候。

如下图所示是在Mac电脑上的RSSI实例，注意到这里RSSI的取值是负值、无量纲：



如下图所示是一个真实环境抓包结果中的RSSI数值展示：



注意到，这里的信号强度采用的计量单位是dBm：这是功率的绝对值，0表示最强，越近0接收效果越好。与之相关的一个度量单位是dB：这是功率相对值。很多软件厂商和硬件厂商为了避免RSSI数值的非标准化解读引起误解，倾向于使用dBm来表示RSSI。

如下2张图来自使用Wireshark对一次真实网络的无线抓包结果的RSSI平均值随（抓包）时间变化而变化的可视化展示：

4. 无线网络安全实践常见术语与概念

4.1. 网卡芯片

谈到无线网络安全实践，首先要了解的就是网卡芯片类型。并不是所有的无线网卡都支持“监听模式”，而支持“监听模式”的网卡是我们获取无线网络通信数据的一个必备硬件条件。WikiDevi是一个互联网众包模式的关于无线网络硬件设备参数信息的知识库，在这里我们能查阅到很多网卡所使用的芯片硬件参数信息、操作系统驱动支持情况等。

例如我们的实验器材中用到的数款网卡使用的Atheros AR9271芯片相关资料就可以在WikiDevi的Atheros\_AR9271查阅到。

4.2. 厂商

同样的，还是在WikiDevi上，我们能看到无线网卡的芯片提供商有一个汇总列表，如下：

Atheros

Broadcom

Celeno

Conexant

Cypress

Envara

Intel

Intersil

Lantiq

Marvell

MediaTek

Ralink

Realtek

Texas Instruments

Qualcomm Atheros

Quantenna

Wilocity

4.3. 驱动程序

**在Linux系统中，如果能正常加载网卡驱动，则我们可以通过以下一些常用命令来查看网卡详细信息：**

# 查看PCI接口连接的所有设备信息

lspci

# 查看USB接口连接的所有设备信息

lsusb

# 查看所有无线网卡的硬件参数信息

iw phy

# 查看所有无线网卡的接口标识信息

iw dev

# 查看最近加载的（硬件）驱动日志信息

dmesg

我们在这里有一个完整的iw phy输出信息加上详细注解的例子，供参考。

4.4. 天线

天线（Antenna）属于无源（电源）器件，通过控制信号发射的角度，来获得信号“增益”。发射信号的总能量由AP决定，天线决定信号如何传播出去。

天线是“无源器件”，所以天线本身并不能给AP的信号增加能量。然而我们一提到天线，最重要的指标就是说天线的“增益”，那么天线是如何获得信号强度的“增益”呢？答案就是，靠控制信号发射的角度。这个原理有些类似于手电筒，手电筒靠一面凹镜，让光线都集中在某一角度，来让光线照到更远的地方。手电筒及电池相当于AP设备本身，而手电筒的灯泡和凹镜就相当于我们的天线。如果摘掉手电筒的凹镜，那么就相当于使用一个增益很小的全向天线，光线照射很分散，覆盖距离很近；有了凹镜，则相当于使用了一个高增益的定向天线，光线集中，覆盖距离很远。

信号总的能量是由AP决定的，天线则决定让这些能量集中在某个角度内，这个角度越小，能量聚集度越高，获得的信号“增益”也就越大，信号覆盖的距离越远；反之，如果覆盖角度越大，能量聚集度越低，信号覆盖的距离越近。这就是天线获得增益的基本原理。

常见的天线类型有全向天线和定向天线两种：

全向（Omni）

水平各个方向增益相同，全向天线增益越大，水平方向上覆盖的范围也就越大，垂直方向上覆盖的范围越小。一般应用于室内环境。

定向（directional）

垂直方向和水平方向都不是360度覆盖，一般来说覆盖角度小，覆盖的范围也就越远。大多数情况下，使用定向天线在高处向室外热点区域覆盖，以保证可靠的信号质量。

如果需要满足多个STA，并且这些STA是分布在AP的不同方向时，需要采用全向天线；如果集中在一个方向，建议采用定向天线；另外还要考虑天线的接头形式是否和AP匹配、天线的增益大小等是否符合实际需求。

5. 知名的“路由器”操作系统

OpenWrt基于GNU/Linux的嵌入式设备（主要是无线路由器）操作系统，采用GPL授权协议。官方硬件兼容性表格列举了超过600种市面上的无线路由器。

DD-WRT也是基于GNU/Linux的用于无线路由器的的第三方固件，采用 GPL 协议，支持基于 Broadcom 或 Atheros 芯片参考设计的许多 IEEE 802.11a/b/g/n 无线路由器。固件是由 BrainSlayer 来维护的，放在dd-wrt.com。DD-WRT 的第一个版本基于Sveasoft Inc 公司的 Alchemy 开发出来的, 而 Alchemy 又是基于 GPL 的 Linksys固件及许多其它开源程序。由于后来人们需要向 Sveasoft 支付20美元才能下载Alchemy固件，直接的后果就是发展出了DD-WRT。

DD-WRT 的新版本 (v24) 是一个全新的项目。DD-WRT 提供了在 OEM 固件，甚至 Sveasoft 收费的固件中没有的高级特性。此外它去除了 Sveasoft 固件的产品激活和跟踪功能。

官方支持设备列表上列举了上百种市面上的无线路由器。

RouterOS是MikroTik公司基于Linux内核开发的路由器操作系统，可以安装在其私有硬件产品RouterBOARD系列或标准x86计算机。RouterBOARD系列设备在出厂时即预置了RouterOS产品授权秘钥，PC设备安装RouterOS需要额外获取授权秘钥，授权等级见官方Wiki。

在商业路由器领域有RouterOS很多成功案例，国内的很多酒店采用了这个品牌的路由解决方案。

6. 无线网络安全的研究范围

网络安全的本质是“持续对抗”，对于无线网络安全来说，我们主要关注10大威胁类型，即：数据拦截（Data Interception）、拒绝服务攻击（DoS）、非法接入点（Rogue APs）、无线闯入者（Wireless Intruders）、配置不当的接入点（Misconfigured APs）、自组织和软接入点（Ad Hoc and soft APs）、恶意同名接入点（Evil Twin APs）、无线钓鱼（Wireless Phishing）、针对终端的攻击（Endpoint Attacks）和异常客户端（Misbehaving Clients）。

6.1. Data Interception

Martin Beck等人在2008年11月发表的Practical attacks against WEP and WPA中首次公开了针对WPA保护的IEEE 802.11无线网络的非字典类暴力破解方式的高级实用攻击方法（命名为： Beck-Tews 攻击）。该攻击方法主要利用了TKIP加密方式的缺陷，攻击者只需要12-15分钟时间的目标网络无线数据帧嗅探就可以解密目标网络中的ARP请求或响应数据并向目标网络中注入7个任意内容数据帧。攻击者借助这种数据注入方式毒化网络通信数据，可以实现对目标网络中的主机通信数据实现中间人攻击。2009年8月，两位来自日本的研究者Toshihiro Ohigashi1和Masakatu Morii在A Practical Message Falsification Attack on WPA中公布了一种基于Beck等人的TKIP攻击方法改良：Beck-Tews攻击依赖于目标WPA实现支持IEEE 802.11e QoS机制，改进方法则对任意WPA实现均有效。攻击者通过定向天线或使用更高的发射功率来扮演一个无线中继器（wireless repeater），对AP和客户端之间的通信数据进行拦截和转发（中间人攻击），实现更高效的Beck-Tews攻击（理想情况下，该改良方式1分钟即可完成Beck-Tews攻击）。

现有的大量无线路由器、AP在出厂时的默认配置都是启用的WPA/WPA2混合模式，而对于WPA模式来说默认的数据加密方式正是TKIP，这就给攻击者以可乘之机。为了缓解上述威胁，目前的可行方式是配置AP停用WPA认证和TKIP加密方式，只允许WPA2 + AES/CCMP组合的安全认证和加密模式，Beck-Tews攻击对该模式无效。

6.2. DoS

无线信号像阳光和空气一样无处不在，我们在802.11 系列标准一节已经展示了：使用相近信道（工作频率）的AP之间会产生无意的相互干扰，这种干扰在小范围、高密度AP场景中尤其严重。拒绝服务攻击的效果是服务质量、服务可用性受到影响，非主动的干扰也能被攻击者模仿其原理实现主动的拒绝服务攻击。不仅如此，伪造数据帧强制无线客户端掉线在无线网络中是一种低成本、易实施的攻击方式，目前并没有特别完善的针对此类强制客户端掉线方式攻击的防御方案。除此之外，诸如暴力破解口令过程也会对AP的CPU资源造成性能消耗，进而影响到AP的数据传输性能，拖累网络连接和数据传输质量。

6.3. Rogue APs

攻击者可以通过把自己精心设计定制的AP悄悄接入到有线网络，实现嗅探有线网络流量、建立内网攻击的跳板和立足点、实施远程无线网络攻击等典型流氓AP行为。如下图所示，是一个知名的无线网络安全审计定制专用AP，如果这样一个AP被重新包装一下，就能成为攻击者手中的一个有效的Rogue AP。

部署无线入侵检测和防护设备来检测企业内部的非授权AP是缓解非法接入点风险的重要手段，同时，合理部署有线网络的802.1X端口认证等手段也可以阻止恶意接入点直接接入到有线网络。

6.4. Wireless Intruders

由于无线网络物理覆盖范围可以达到50米以内，导致网络边界较长，如果配置无线入侵防护设备（WIPS, Wireless Intrusion Prevention System ）需要的设备数量较多。而口令破解、嗅探、DoS等已知无线网络威胁可以随时从任意信号覆盖的位置发起和停止，这就意味着无线闯入者的行踪可以更加隐秘，攻击意图更加难以预测，WIPS需要兼顾部署覆盖率和防御能力的自动更新部署。

6.5. Misconfigured APs

相比较于个人无线网络使用的AP是独立配置和管理的，配置错误只是影响一个无线网络。而企业级无线网络则往往是采用的集中式、批量管理，由于需要兼顾无线客户端设备对新旧无线协议支持程度的差异，在配置AP时可能会启用一些不安全的旧版本协议支持。同时，网络设备的定期更新、安全补丁应急分发等能力，也是考验企业级无线网络部署和管理能力的关键指标。

6.6. Ad Hoc and soft APs

非法接入点对于企业内网安全是一个巨大的威胁，原本通过网络边界访问控制等措施保护得当的有线网络，一个简单的笔记本个人热点、手机个人热点、使用有线网络自建的个人无线网络就可能把一个隔离内网变为脆弱“公”网。

部署无线入侵检测和防护设备来检测企业内部的非授权AP是缓解自建个人无线热点风险的重要手段。

6.7. Evil Twin APs

由于AP可以随意设置ESSID，攻击者利用这个特性可以很容易宣称自己是某个合法AP，诱骗其他无线设备连入。这种仿冒AP的行为，被称为Evil Twin Attack，实施该行为的AP被称为Evil Twin AP。越来越多的便利Evil Twin攻击工具设备和软件的出现恶化了这个问题：基于软件定义驱动的软AP，能够监听无线客户端的无线网络探测请求，主动伪造信号强度更高的同名热点强制客户端连入。

基于802.1X的WPA-企业级认证是针对这个问题的有效解决方案，但实际配置部署如果不得当（具体查看本书第三章内容的详解），效果甚至会适得其反：不仅没能阻止客户端误连假冒AP，攻击者甚至可以更快的破解掉合法无线网络的认证口令。

6.8. Wireless Phishing

使用Evil Twin AP对连入的客户端投毒一旦成功，可以借助浏览器缓存和DNS解析记录缓存投毒方式实现对目标无线客户端的应用程序更长时间的持久化钓鱼攻击。这就意味着，即使无线客户端脱离了Evil Twin攻击环境，攻击效果依然会持续很长一段时间，与客户端在此之后使用何种联网方式不再有关。

除此之外，一些免费Wi-Fi、公共Wi-Fi由于配置不当，使得内网攻击十分容易得手，内网投毒攻击使得无线钓鱼的攻击成功率大大提升。

缓解此类风险的办法主要包括：外出时手机关闭Wi-Fi开关、避免使用免费和公共Wi-Fi、即使使用免费和公共Wi-Fi避免操作金融类应用和网络账户类应用，有条件的情况下在使用安全性不确定的无线网络时开启全局VPN保护数据传输的机密性和完整性。

6.9. Endpoint Attacks

针对无线终端设备的已知攻击方式包括：

终端设备的固件（操作系统）漏洞利用

终端管理软件漏洞利用

无线客户端软件漏洞利用，例如恶意SSID、格式化字符串攻击和XSS/CSRF等等

利用无线终端会自动重连已知热点的特性，离线破解该已知热点的PSK口令（详见第三章相关内容）

缓解此类风险的办法主要包括：外出时手机关闭Wi-Fi开关、保持软件和系统的自动更新。

6.10. Misbehaving Clients

例如很多路由器Web的管理系统存在大量Web应用程序漏洞可被利用，Web管理系统使用弱口令或缺省口令等，一些授权接入到无线网络的客户端会滥用自己的网络连接权限去攻击无线路由器以及内网其他系统和服务。

部署内网安全检测和防护产品，限制和阻止非可信来源的设备访问网络基础设施系统和重要信息系统的管理接口，可以有效的缓解上述内网攻击和滥用行为。

第二章 无线接入网监听

1. 实战无线网络监听

1.1. 无线网络监听的基本条件

无线网络监听的基本条件首先是能获取到原始的无线数据，具体来说又分为硬件和软件依赖条件。硬件上来说，我们需要支持可以工作在监听模式的无线网卡（通常我们会使用USB接口的无线网卡）。软件上来说，需要可以对该无线网卡识别和配置的底层驱动和用户态的配置软件或API。在满足了原始无线数据获取能力之后，我们就可以像在有线网络中一样使用抓包软件（例如，wireshark、tshark、airodump-ng等）进行无线数据的获取和转存（通常保存为pcap格式文件）了。

802.11无线网络中，由于STA的工作模式限制，无线网卡需要通过channel hopping技术在多个信道之间切换才能尽可能的去模拟对所有信道的数据获取（监听），由于channel hopping的技术限制，事实上对于任意一个指定的信道，在一个完整的抓包周期内是无法获取到所有的通信数据的（只有在网卡监听在指定信道的时隙内才能监听到该信道在该监听时隙上的通信数据，一旦切换到下一个其他信道，则上一个信道的通信数据就不再能收集到，直到重新又监听到该指定信道时）。在实战中，我们通过channel hopping来摸清当前所在位置有哪些无线网络可用、可被监听到，一旦确认了具体的监听目标网络，会放弃使用channel hopping技术，改为指定监听信道方式来进行定向抓包。除此之外还需要注意的是，不同无线网卡对802.11协议的支持也有硬件差异，对于802.11协议家族来说，支持802.11 b/g协议规范的USB无线网卡是最普遍的，其他诸如802.11 a/n/ac协议规范的支持则需要看具体网卡的硬件实现情况。本书的无线网卡（通常我们会使用USB接口的无线网卡）一节记录了本书写作过程中实际采购并测试过的十几种USB无线网卡的硬件能力，供需要动手实践无线网络监听的读者们参考。

在⽆线⽹络安全实验的操作系统选择方面本书选择的是Linux发行版Kali，主要原因是这是一个为渗透测试研究为主的安全研究人员专门量身定制的操作系统，该系统内置了很多信息安全从业人员常用的工具软件和驱动程序。特别对于我们的无线网络监听实战来说，把USB无线网卡接入到Kali之中，你不必再考虑安装驱动程序和相关工具软件的问题，基本可以做到开箱即用。

1.2. 无线网卡设置查看基本工具

本节内容涉及到的常用工具使用方法请直接查阅本章实验指导，概括来说一次无线数据嗅探实验包括以下几个主要步骤：

设置网卡进入监听模式

使用抓包器对指定的监听模式无线网卡进行抓包

channel hopping模式收集附近无线网络信息

确定抓包目标，进行定向抓包

使用报文分析软件对抓包结果文件进行离线分析

在实战无线数据监听时，如果遇到抓不到数据包的情况，建议按照以下故障排除的思路和操作顺序依次排查：

硬件相关常见故障排查

更换USB连接接口，有些电脑的不同USB口由于供电能力差异，部分耗电量较大的USB无线网卡可能只能在电脑的特定USB接口上可以正常工作；

对应上一条排查建议，还可以尝试更换USB无线网卡的USB连接线，排除连接线故障原因；

如果是使用的虚拟机环境，建议检查虚拟机的USB设备共享设置。必要时可以启用USB 3.0兼容选项或者如果USB 3.0兼容模式无法正常识别网卡或网卡无法抓包，可以降级为USB 2.0兼容模式尝试故障排查；

软件相关常见故障排查

重启大法好。无论是虚拟机还是物理主机，有时可能是由于新安装了USB驱动或其他未知不可描述原因，重启或关闭系统再启动（冷启动）就可以解决问题；

使用tail -F /var/log/messages，重新连接USB无线网卡，检查该日志中的消息是否有出现一些故障信息报错。尝试在搜索引擎中搜索相关报错信息关键词；

参考本章实验指导中的其他故障排查建议；

本书推荐的命令行方式进行无线数据监听使用到的主要工具来自于aircrack-ng工具套装，这是一个无线网络安全审计的瑞士军刀，包含一系列小工具。根据该软件的官网信息，该软件的主要特性如下：

Monitoring: Packet capture and export of data to text files for further processing by third party tools.

Attacking: Replay attacks, deauthentication, fake access points and others via packet injection.

Testing: Checking WiFi cards and driver capabilities (capture and injection).

Cracking: WEP and WPA PSK (WPA 1 and 2).

在Kali中我们可以通过dpkg -L aircrack-ng查看当前系统上安装了哪些具体的aircrack-ng小工具。

$ dpkg -L aircrack-ng | grep bin

/usr/bin/aircrack-ng

/usr/bin/airdecap-ng

/usr/bin/airdecloak-ng

/usr/bin/airolib-ng

/usr/bin/besside-ng-crawler

/usr/bin/buddy-ng

/usr/bin/ivstools

/usr/bin/kstats

/usr/bin/makeivs-ng

/usr/bin/packetforge-ng

/usr/bin/wpaclean

/usr/sbin/airbase-ng

/usr/sbin/aireplay-ng

/usr/sbin/airmon-ng

/usr/sbin/airodump-ng

/usr/sbin/airodump-ng-oui-update

/usr/sbin/airserv-ng

/usr/sbin/airtun-ng

/usr/sbin/besside-ng

/usr/sbin/easside-ng

/usr/sbin/tkiptun-ng

/usr/sbin/wesside-ng

在完成无线通信数据报文的捕获之后，我们的工作焦点将转移到⽆线通信数据报⽂分析。

2. ⽆线通信数据报⽂分析

2.1. 802.11 协议提供的⽹络服务类型

基于IEEE 802.11协议构建的无线网络提供的基本服务群（Basic Service Set, BSS）是由许多主机所构成的集合，在这个服务群中的主机之间可以相互通信且拥有相同的基本服务群标识码（BSSID）供这些用户组建一个无线局域网使用。根据第一章的介绍我们知道，可以是基础设施架构网络（BSS）或独立BSS（IBSS）。

独立BSS架构的无线局域网又称为AdHoc无线网络，可以让不限量的主机实时地架设起无线通信网络。在这种架构中，主机彼此之间可以直接通信，通常情况下都是临时通信需求驱动而搭建。而基础设施架构网络中通常会包含一个访问接入点（AP），它可以将一个或多个WLAN与已有的有线网络进行分布式连接，以提供某个WLAN中两个主机间的通信数据交换，另一方面也使得WLAN中的主机能存取有线网络中的资源。两个或多个BSS通过设置相同的ESSID可以组成一个信号覆盖范围更大的ESS。

在上述两种架构中，IEEE 802.11提供两类共9种网络服务。

2.1.1. 主机服务

主机服务让主机（STA）具有正确的收发数据能力，同时也能保证数据传送的安全性。

数据封包传送服务为最基本的网络功能，IEEE 802.11使用自身协议（链路层）将（链路层以上）数据进行封装和传送。

身份验证服务主要用来确认每个主机（STA）的身份。IEEE 802.11通常要求双向式的身份确认，它也允许同一时间一个主机和多个主机（包括AP）进行身份验证。

已完成身份认证的STA可以用解除验证服务来取消身份认证，一旦取消后连接也同时被取消。

隐私保护服务通过加密机制保护通信数据的机密性。

2.1.2. 分布式系统服务

分布式系统服务由分布在不同位置的系统（通常是AP）提供，使IEEE 802.11的封包可以在同一个ESS中的不同BSS间传送，无论主机移动到ESS中的哪个地方都能收到属于它的数据。AP是唯一同时提供主机（STA）服务和分布式系统服务的无线网络组件，也是主机与分布式系统间的桥梁。分布式系统服务提供下列5种服务：

连接服务。目的是在STA和AP之间建立一个通信链路。当分布式系统要将数据传送给主机时，必须事先知道这个主机目前是通过哪个AP接入分布式系统的，这些信息都可以由连接服务提供一个主机在被允许经由某个AP传送数据给分布式系统前，必须先和此AP进行连接。通常在一个BS内有一个AP，因此在这个区域内的任意主机若想要与外界进行通信，就必须先与此AP进行连接。这个过程类似注册，当主机完成连接后，AP就会记住这台主机目前在它的管辖范围之内。连接服务通常都由主机启动，用它来与AP进行连接。（注意，在任何时刻一台主机只会和一个AP进行连接，这样才能使分布式系统知道哪个主机是由哪个AP所管辖的，然而一个AP却可以同时与多台主机进行连接。）

重连服务。目的是将一个移动中的主机连接由一个AP转移至另一个AP。当主机从一个服务区移动到另一个服务区时，它将启动重连服务。重连服务会将主机与它所移入的服务区内的AP进行连接，使分布式系统知道此主机已经转移至另一个AP的管辖区域内。重连服务通常也是由主机启动。

取消连接服务。当一台主机数据传送结束时，可以使用取消连接服务对当前已有的连接进行取消。当主机在服务区内移动时，它除了会对新的AP启动重连服务外，还会对旧的AP启动取消连接服务。此服务可以由主机或AP任一方来启动，不论是哪一方启动的另一方都不能拒绝。（需要注意的是AP可能因网络负荷过重而是用取消连接服务对主机取消连接）。

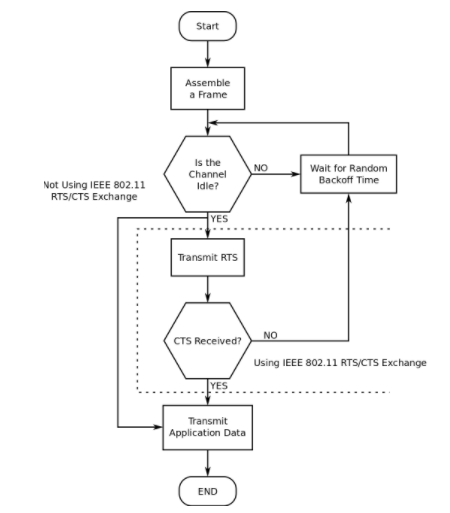
分发服务。此服务主要由BSS中的主机使用。当主机需要传送数据时，会先将数据传送至AP，再由AP通过分布式系统传送至目的地。IEEE802.11并没有规定分布式系统要如何将数据正确的送至目的地，但它说明了在连接、取消连接和重连等服务中，数据应该由哪个AP进行输出以将数据送达至正确的目标地点

整合服务。目的是让数据能够在分布式系统和现有的局域网之间进行传送。整合服务的任务就是将数据从分布式系统转送到相连的局域网络媒介，其主要工作就是将不同的地址空间做一个转换。

2.2. 802.11 协议中的重要理论基础

2.2.1. CSMA/CA

IEEE 802.11不采用有线网络中的基于碰撞检测的载波监听多路访问（Carrier Sense Multiple Access With Collision Detection, CSMA/CD）技术，而采用了基于碰撞规避的载波监听多路访问技术（Carrier Sense Multiple Access with Collision Avoidance, CSMA/CA）。这是WLAN链路层最基本的接入方法，如下图所示是一个简易流程：



STA发送信息之前，检测信道是否空闲以及空闲的时间是大于IEEE802.11规定的帧间隔时间

如果否，该STA就延迟接入，直到当前的传输结束

之后，也就是一次成功的传输刚结束，这时碰撞发生率最高，因为所有的STA都延迟等待这一时刻的到来，为进一步减少碰撞，STA选择随机退避再次延迟接入

在检测信道的同时倒数计数器，直到其值为0

这时，如果其它的Backoff计时器的数值更短，它就赢得了信道的占用权

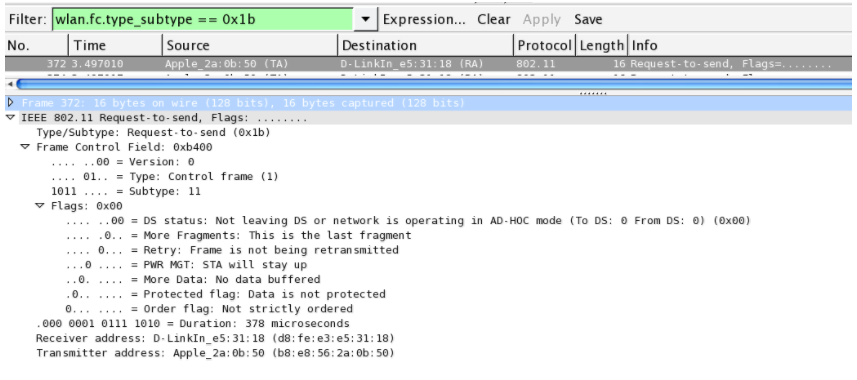
其他的STA检测到信道忙，只有再次延迟接入

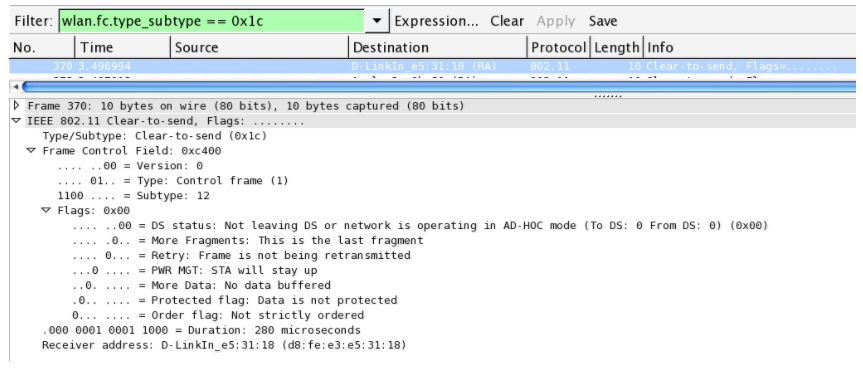
否则，只有在信道空闲时再发送信息

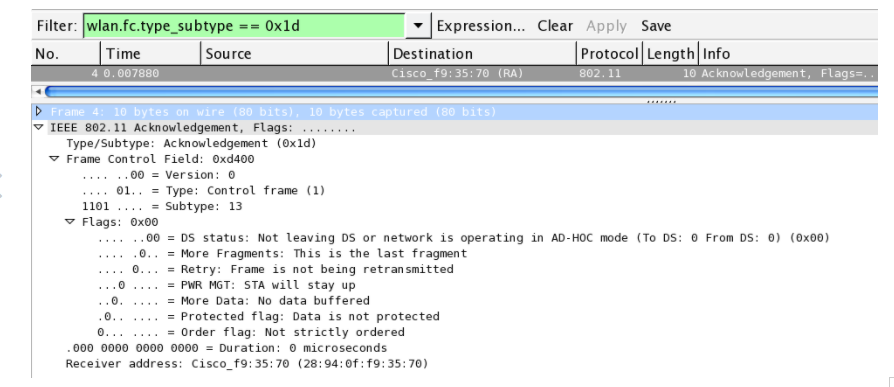
2.2.2. 四次握手协议

IEEE 802.11在CSMA/CA的基础上提供了确认帧ACK，保证在链路层对帧丢失可以检测并重新发送。为了进一步规避碰撞，又引入了RTS/CTS + ACK协议。STA在发送信息之前先发送一个发送请求控制（Request to Send, RTS）包给目的STA。如果信道空闲，则目的STA回应一个清除发送控制（Clear to Send, CTS）包。如果信道忙，不发送CTS，这样就可以避免不同STA同时向同一STA发送信息。如果源站收到CTS，证明信道空闲，它就可以继续发送有用数据。如果该数据需要，目的STA在成功接收后，经过最短帧间隔时隙后就回发确认帧ACK。如果在规定的时间间隙后，源站未收到ACK，则源站就可以判定信息发送失败，可根据需要重发，这样可避免数据丢失。

如下图所示分别展示了在Wireshark中如何通过显示过滤器（Display Filter）语法过滤出RTS、CTS和ACK数据帧，并查看数据帧详情信息。



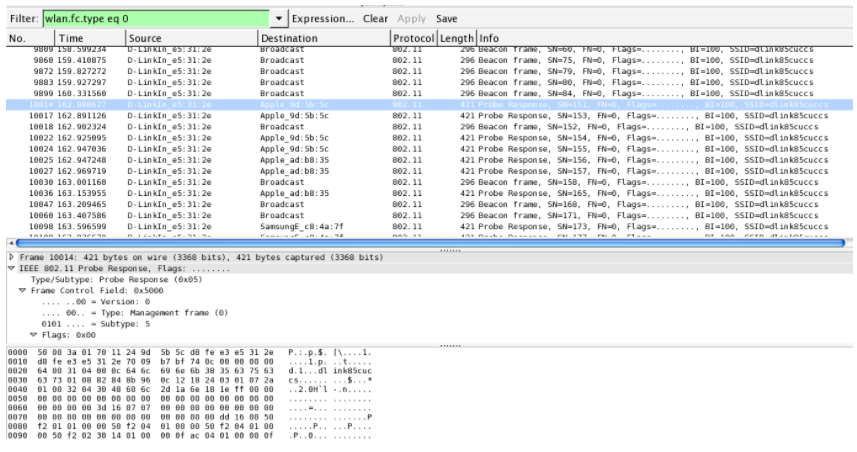




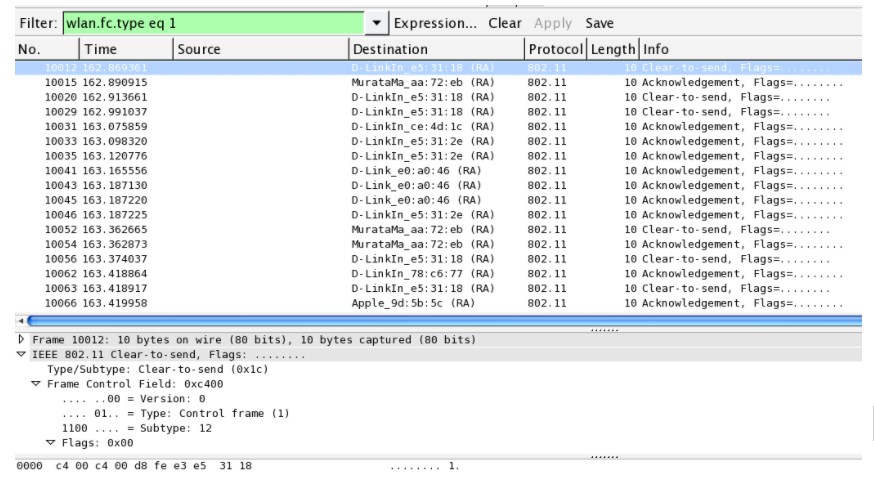
2.3. 802.11 帧结构与Wireshark过滤器语法

IEEE 802.11定义了三大类帧，分别是：管理帧、控制帧和数据帧。每个大类下，又包含了多个子类型。

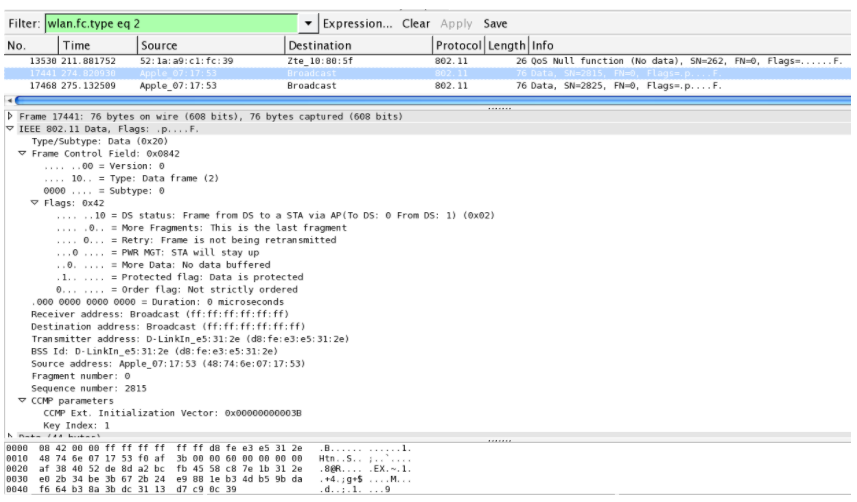
管理帧。管理帧负责监督无线网络状态，它主要用于建立第二层，即链路层，主机间的连接，管理数据包包括身份认证数据包、关联数据包和Beacon数据包等。（为了限制广播或组播管理帧所造成的副作用，收到管理帧后，必须加以查验。只有广播或者组播帧来自工作站当前所关联的BSSID时，它们才会被送至MAC管理层，唯一例外的是Beacon帧。）



控制帧。控制帧通常与数据帧搭配使用，负责清空区域、获取信道和载波监听的维护，并在收到数据时予以确认以提高工作站之间数据传送的可靠性。（因为无线收发器通常只有半双工工作模式，即无法同时收发数据，为防止冲突，802.11允许工作站使用request to send和clear to send信号来清空传送区域）



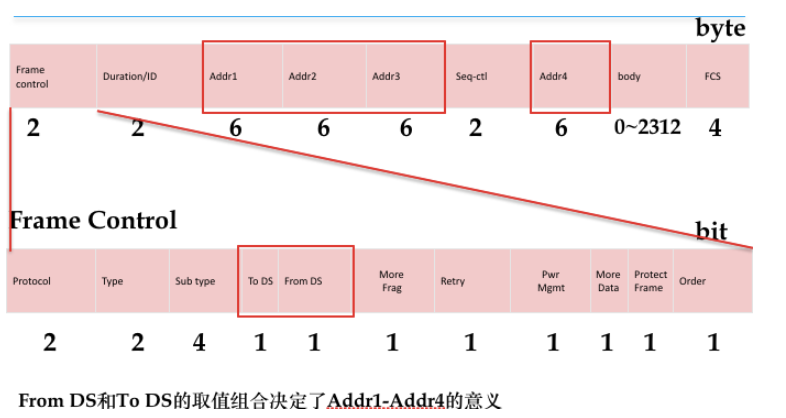
数据帧。数据帧中包含实际需要传送的数据，并且是能够从无线网络转发到有线网络的唯一帧类型。



通过Wireshark的I/O统计图功能可以直观的查看在指定时间段内满足Wireshark设置的Display Filter的所有802.11报文的三大类帧占比情况，结合上述关于802.11中的重要理论基础知识，我们可以分析出目标无线网络的繁忙程度和有效数据通信效率。



2.4. 802.11 MAC地址类型



如上图所示是IEEE 802.11帧头部字段定义示意图，上图中的Addr1 ~ Addr4的取值含义主要包括以下5种：

DA = Destination MAC Address

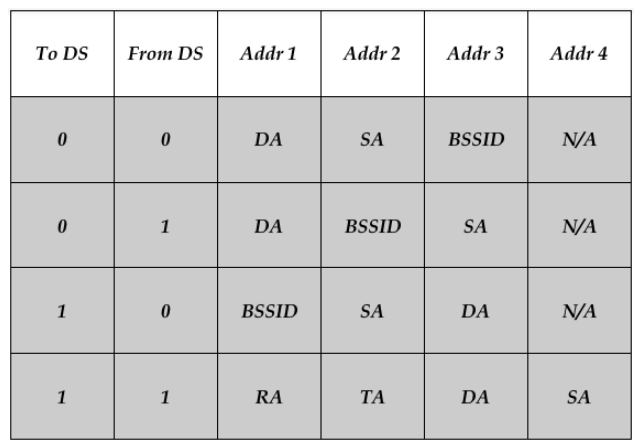
SA = Source MAC Address

RA = Receiver Address indicate MAC Address of station in WM that have to receive frame

TA = Transmitter Address indicate station which have transmitted frame in WM

BSSID

具体每个Addr取何值主要取决于To DS和From DS这2个比特位的取值组合，参见下表。



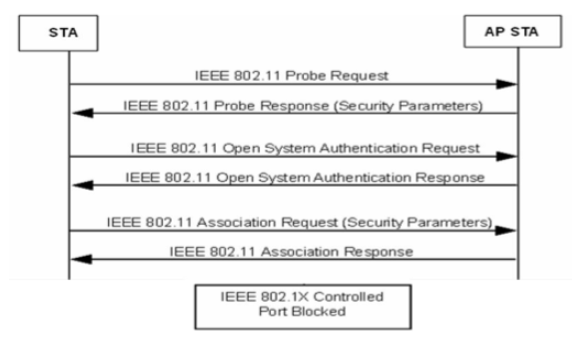
注意在wireshark的packet字段解析时，同一个字段地址可能会显示2个或多个等价地址类型名称。

更多的在分析802.11帧结构时常用的Wireshark Display Filter可以使用这2个过滤器前缀：wlan.\*和wlan\_mgt.\*。

2.5. 802.11 加密与认证机制原理

2.5.1. 开放式认证

开放式认证本身是⼀种⽆效的认证算法，如果没有配合数据加密技术AP将会准许任何来自客户端的认证要求，即任何知道基站SSID的客户端都可以连接到此⽹络。 但如果基站使用WEP加密算法对数据进⾏加密，WEP密钥就成为了另⼀种存取控制机制，即客户端即使通过了认证，但若没有WEP密钥也⽆法将资料传送到基站亦或是将基站传送出的数据进⾏解码。



如上图所示是一个STA加入一个曾经连接过的无线网络时的开放式认证和关联加入网络的步骤示意图，图中的STA发送Probe Request帧是一个可选步骤。AP如果开启了SSID⼴播，则STA可以通过beacon frame得到认证相关信息。

2.5.2. WPA/WPA2

在无线网络领域中，广播(broadcast)一词具有崭新的意义。无线网络使用开放性介质， 如果传输链路没有采取适当的加密保护，使用上的风险就会大幅增加。既然是开放性的网络介质，只要拥有适当的设备，任何人都可以偷窥未经保护的数据。本章伊始已经通过实战的方式，介绍了实施无线网络监听的软硬件条件，就目前的技术和产品发展成熟度来说，任何人都可以非常低的成本来监听无线网络通信数据。

防范数据免受中间人攻击（监听和篡改）是一个典型的信息安全问题范畴。起初，有线等级隐私(WEP)标准被视为无线安全的解决方案。不过在 802.11 问世的前四年，研究人员就已经发现 WEP 并不安全。考虑到目前市面上我们可以见到的无线网络设备已经很难再看到启用WEP安全机制的了，本书不再深究WEP，转而介绍目前最主要的无线安全机制：WPA和WPA2。

802.11i工作组采用双轨并行的方式以解决链路层加密协议的弱点，先后设计了两种新的链路层加密协议。第一种称为临时密钥完整性协议（Temporal Key Integrity Protocol, TKIP）， 被设计来尽可能强化 pre-802.11i 硬件的安全性，向后兼容（backward compatible）所有支持WEP机制的硬件设备。另外一种则是重新打造的加密协议，称为「计数器模式」搭配「区块密码锁链-信息真实性检查码」协议 （Counter Cipher Mode with block chaining message authentication code Protocol, CCMP），被设计来提供最高等级的安全性，不向后兼容老旧设备。所有新设备获得Wi-Fi认证标志必须支持CCMP协议。前者被称为WPA，后者则被称为WPA2。

2.5.3. RSN

除了 TKIP 与 CCMP，802.11i 还定义了一组程序，称为健壮安全网络（Robust SecurityNetwork, RSN）。这组程序主要定义了密钥的产生与传递方式。

802.11链路层加密协议使用了两种密钥。成对密钥（pairwise keys）用来保护工作站与 AP 之间 往来的数据。群组密钥（group keys）用来保护 AP 至所连接工作站之间的广播或组播数据。成对密钥系产生自身份认证信息，群组密钥是由基站动态产生然后传递给各工作站的。

TKIP 与 CCMP 均使用预先生成的单一主密钥（预共享密钥（Pre-Shared Key, PSK））来产生帧保护过程所需要的其他密钥。利用衍生密钥，工作站 得以更新加密密钥，无需重新执行整个认证程序。主密钥本身扮演着秘密根源的角色，必须小心保护，因为所有后续会话密钥均衍生于此。密钥分级（key hierarchy）的部分目的是为了衍生用来保护临时密钥之传送的密钥。

配钥是从主钥开始。在成对密钥体系中，主钥称为成对主钥(Pairwise Master Key, PMK)，长度为 256 个比特，如下图所示。



PMK 必然有其来源。在 WPA-个人模式 中便是使用成对主钥。在WPA-企业级模式使用认证服务器的情况下，主钥是计算产生自 RADIUS 服务器中，然后以MPPE（Microsoft Point-to-Point Encryption）这个厂商特有的 RADIUS 属性，送给基站。

为了得到前述临时密钥，必须使用预先定义好的伪随机函数来展开 PMK。为了使数据更为随机，此一展开过程是根据PMK、 申请者（supplicant ）与认证者(authenticator）的 MAC 地址以及两个作为四次密钥交换协商 (four-way key exchange handshake)的随机 nonce 值，展开结果我们称为成对临时秘钥，记为PTK（Pairwise Transient Key）。

PTK = Function(PMK, A-nonce, S-nonce, Authenticator MAC, Supplicant MAC)

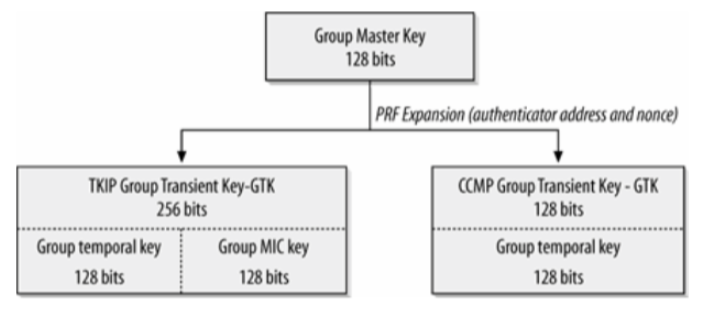
= Hash(PMK||A-nonce|| S-nonce || AP MAC || STA MAC)

TKIP 的临时密钥总长度为 512 个比特，PMK自身的 256 比特之外的 256 个比特，一半作为 TKIP 数据处理时所使用的 128 比特临时密钥，一半用于 Michael 完整性检验（MIC）。TKIP 之所以需要两把额外的密钥，是因为它使用了传统的加密与认证机制，会严格区分加密与认证。CCMP 的临时密钥只有 384 个比特，因为它只使用一把 128 比特密钥来进行认证与加密。

TKIP 与 CCMP 均会使用伪随机函数将 256 比特的 PMK 展开为成对临时密钥。在 TKIP 与 CCMP 体系中，临时密钥的两组 128 比特区块，在传递过程中被用来保护临时密钥。

这两种密钥体系均始于两把 EAPOL 密钥，通过 EAPOL-Key 信息，保护衍生密钥的传输安全。其中使用了两把 128 比特的密钥。第一把是 EAPOL 密钥确认密钥（EAPOL Key Confirmation Key, KCK），用来计算配钥信息（keying message）的信息完整性检验值。 第二把 EAPOL 密钥加密密钥（EAPOL Key Encryption Key, KEK），用来加密配钥信息。这2个密钥在接下来的四次握手认证中会详细讲解。

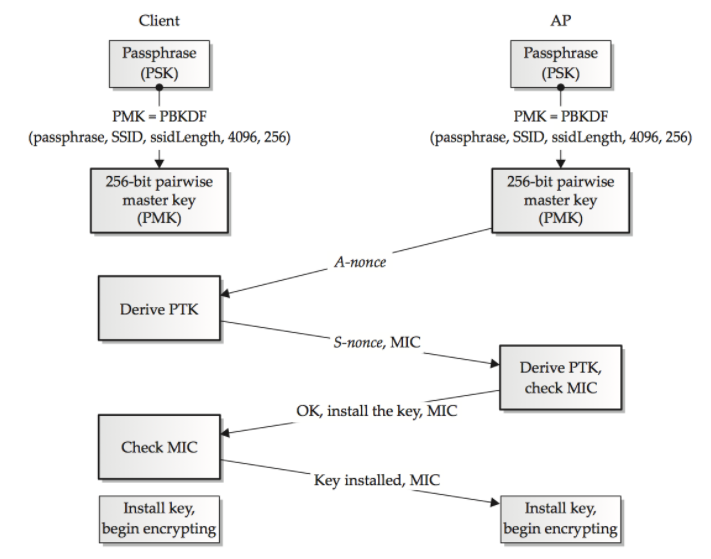
链路层安全协议为广播与组播使用了另一组不同的密钥。已连接的各工作站均拥有不同的预设主钥（pre-master key），因此无法从认证过程中推衍出组播所需要的密钥。事实上，认证者拥有群组主钥（Group Master Key, GMK)，以作为临时密钥的基础。通过伪随机函数，群组主钥会被展开成群组密钥体系，如下图所示。在此并未产生密钥加密或密钥确认密钥，因为密钥交换系以成对 EAPOL 密钥来传递密钥。



当工作站离开网络，不论是使用完毕或被踢出网络，网络系统即可更新群组密钥。在 TKIP 中，反制措施也会导致系统重新产生群组密钥。

2.5.4. 四次握手认证

802.11i 规范了一种衍生密钥的机制，而不仅是采用主钥并以之为加密协议的输入项。为了防范重放攻击，密钥的交换使用了随机乱数，并且需要经过协商。成对与群组密钥系分别通过各 自的协商加以更新，如下图所示：



成对（pairwise）或点播（unicast）密钥是通过所谓的四次握手认证（four-way handshake）加以传递，如上图所示。申请者（supplicant，在此处为Client）与认证者（authenticator，在此处为AP）均持有一把共享 的成对主钥PSK。四次握手交换用以产生临时密钥的参数，以及确认双方均已准备就绪，可以开始进行加密传输。依序传送的信息是由下一个信息来代表回应。

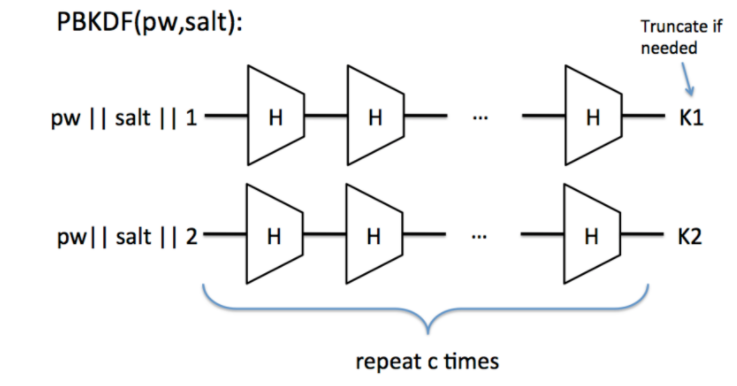
认证者将 nonce 传给申请者。nonce 是防范重放攻击的随机值，信息本身并未经过认证，但并没有被篡改的危险。如果信息遭人更改，协商就会失败并重新执行。至此，申请者就可以将成对主钥展开成完整的成对密钥阶层体系。展开的过程中，需要用到申请者与认证者的 MAC 地址、成对主钥以及两个 nonce。

申请者所送出的信息中包含申请者的 nonce 以及初次与网络连接时所取得的安全参数副本（目标网络支持的安全特性信息）。整个信息系经过以 EAPOL 密钥确认密钥计算而来的完整性检验值的验证。认证者接收到信息，取出申请者的 nonce，依此衍生出完整的密钥阶层体系。此密钥体系中，包含用来签证（sign）信息的密钥。如果认证者无法验证此信息，整个协商即告失败。

此时协商双方的密钥均已就绪，但仍需要确认。认证者会将一个信息传给申请者。此信息代表将被加入之成对密钥的序号，它同时包含了目前的群组临时密钥（Group Transient Key, GTK），以便后续能够更新群组密钥。GTK 经过EAPOL 密钥加密密钥的加密，整个信息系经过密钥确认密钥的认证。

申请者最后会送出确认信息给认证者，告诉认证者已经接收到配钥信息，可以开始使用这些密钥。此信息系经过密钥确认密钥的认证。

成对主钥计算使用的函数被称为基于密码的密钥衍生函数（Password-Based Key Derivation Function, PBKDF，如下图所示是其基本算法工作原理），PMK = PBKDF(Passphrase, ESSID, ssidLength, c)，其中c是迭代次数。对于 WPA-个人模式 来说，c = 4096。ssidLength是目标网络的ESSID长度。



PBKDF的密码学意义在于：

相同的密码输⼊，经过PBKDF运算之后每次的结果都不相同

通过增⼤迭代参数c，降低暴⼒破解的速度

salt的选择如果做到不可预测，则可以抵御预先计算PBKDF字典的加速暴⼒破解攻击⽅法

WPA/WPA2 PSK使用的salt是ESSID和ssidLength。一些老旧的无线路由器和AP，在出厂时设置的默认SSID名称是唯一的，例如dlink。而现在我们从市场上购买到的无线路由器和AP，在出厂时设置的默认SSID则通常会使用到设备本身的部分MAC地址值来确保不同设备的默认SSID名称均不相同。这就是默认安全水平在提高的一个表现，老旧设备的这种简易默认设置，很容易遭到基于预先计算好的PMK字典的认证破解攻击，而新设备则很好的规避了这个默认值风险。

3. Wi-Fi Direct

Wi-Fi CERTIFIED Wi-Fi Direct®（Wi-Fi直连）是一种认证标志，表示设备间支持Wi-Fi直接互联，帮助用户以更为简单而便捷的方式完成打印、共享、同步和显示等任务。带有Wi-Fi Direct标志的产品可以在不加入传统的家庭、办公或热点网络的情况下互相连接。支撑Wi-Fi Direct认证标志的技术规范名称为：Wi-Fi Peer-to-Peer（简写为Wi-Fi P2P）。

利用这种技术，手机、照相机、打印机、个人电脑和游戏设备将能够直接互连，以迅速而轻松的方式传输内容、共享应用。Wi-Fi Direct可以支持一对一直连，也可以实现多台设备同时连接。连接Wi-Fi Direct认证设备的操作简单方便，按下按钮、同时点击两个支持NFC的设备，或输入PIN码这三种方法（Wi-Fi Protected Setup, WPS）均可实现连接。此外，所有的Wi-Fi Direct连接都采用WPA2™加密方式。

Wi-Fi Direct发展自Ad-hoc模式，但又有所区别。在安全性方面，Ad-hoc 默认WEP，⽽Wi-Fi Direct默认WPA2。在连接方式上，Wi-Fi Direct支持连接Wi-Fi Direct同时连接已存在的⽹络，Ad-hoc只能⼯作在其中⼀种。

4. Wi-Fi Protected Setup (WPS)

Wi-Fi Protected Setup（WPS）是Wi-Fi联盟的一项可选认证项目，以专为简化家庭和小型办公环境中安全Wi-Fi网络设置的技术为基础。Wi-Fi Protected Setup支持大多数用户所熟悉的方式（按下按钮、输入PIN码，或者使用NFC）来配置网络，启用安全功能。传统方式（以WPA-个人模式为例）加入一个无线局域网需要手动输入SSID和密码，WPS能帮助用户自动获得并设置SSID、认证⽅式和密钥。无线设备制造商为了让用户能更快理解产品功能特点，对WPS这一标准功能采用了各自的别名。例如QSS - Quick Secure Setup是TP-Link设备的功能别名。Push 'N’ Connect 是Netgear设备的功能别名。WPS来自于协议规范：《Wi-Fi Simple Configuration Technical Specification》（以下简称为WSC规范），目前的最新版本是2014年8月发布的2.0.5版。

WPS改进了用户使用无线网络的体验，但目前的认证机制已经被发现存在协议漏洞，大大降低了网络接入认证的保护强度。我们将在第3章详细讲解目前WPS协议和产品的已知漏洞原理和具体利用方式。

4.1. 支持的应用场景

根据WSC规范，Wi-Fi Simple Configuration 2.0.5版主要面向家庭和小企业无线网络和P2P协议族。支持如下两大类应用模型：

主要应用模型

设置一个新的安全的 WLAN，实现开箱即用（基础设施模式）

为该 WLAN 添加无线设备

次要应用模型

从 WLAN 中移除某个无线设备

访客上网

密钥信息更换（Re-keying credentials）

通过添加新的 AP或路由器来扩充 WLAN 的覆盖范围

更改网络名称、无线频段和安全、初始化连接配置之外的其他参数

使用NFC接口或标签在“带外”交换认证凭据

4.2. 典型用户体验

场景1: 用户用一个手机去配置一个新购买的AP来提供无线网络服务。AP的通信信道只有以太网和无线网络。

配置步骤如下：

用户给AP上电；

手机上的软件自动检测到了AP并询问用户是否要配置这个AP；

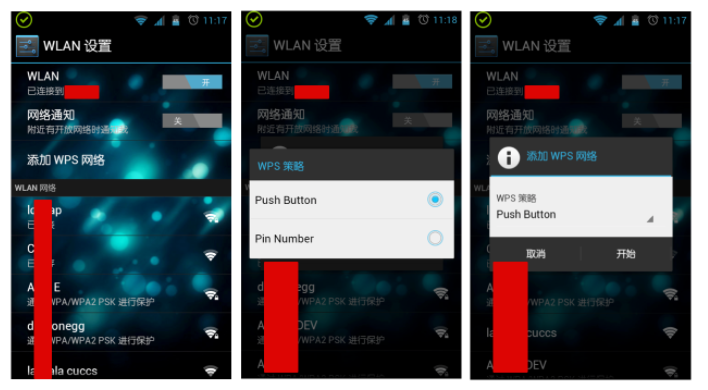
手机上提示用户输入AP上的PIN（通常是打印在AP上的一个贴纸）。用户输入PIN，接受默认配置，确认AP已成功配置完毕；

此时用户需要添加一台打印机到该无线网络

用户给打印机上电开机；

手机检测到新的无线设备并且提示用户把该设备添加到网络。用户查看打印机显示屏上的PIN码并将其输入手机；

当打印机添加到前述配置好的无线网络之后手机和打印机显示屏上同时显示确认信息；



场景2: 用户有一个移动游戏机希望添加到一个配置好的无线网络。用户认为便利性比安全性更重要，因此他决定使用PBC方式配置他的游戏机联入无线网络。

配置步骤如下：

用户按下游戏机上的PBC按钮；

用户按下AP上的PBC按钮；

游戏机和AP在其各自用户界面上显示着联网进度。当联网成功完成时，都会给出“连接成功”的提示。



4.3. 术语定义

AP: An infrastructure-mode 802.11 Access Point.

Credential: A data structure issued by a Registrar to an Enrollee, allowing the latter to gain access to the network.

Device: An independent physical or logical entity capable of communicating with other Devices across a LAN or WLAN.

Device Password: A shared secret that may be used to authenticate the in-band exchange between the Registrar and Enrollee.

Discovery Protocol: A protocol informing the Enrollee and the Registrar of each others presence and capabilities.

DMG (Directional Multi-Gigabit): A frequency band wherein the operating channel center frequency is above 45 GHz.

Domain: A set of one or more Devices governed by a common authority for the purpose of gaining access to one or more WLANs.

Enrollee: A Device seeking to join a WLAN Domain. Once an Enrollee obtains a valid credential, it becomes a Member.

External Registrar: A Registrar for an AP’s Domain that runs on a device separate from the AP.

Guest: A Member with credentials that provide only temporary or otherwise limited access to a WLAN.

In-band: Data transfer using the WLAN communication channel, including WLAN multiband devices (e.g. 2.4GHz, 5GHz, and 60GHz).

Internal Registrar: A Registrar that is embedded in an AP. All APs shall include an Internal Registrar.

Member: A WLAN Device possessing Domain credentials.

NFC Device: NFC Forum compliant contactless device that support the following Modus Operandi: Initiator, Target, and Reader/Writer. It may also support card emulator.

NFC Interface: Contactless interface of an NFC Device.

NFC LLCP: The Logical Link Control Protocol (LLCP) specification between two NFC Forum Devices.

NFC Tag: NFC Forum compliant contactless memory card that can be read or written by an NFC Device and may be powered by the RF field.

Out-of-Band: Data transfer using a communication channel other than the WLAN.

PIN (Personal Identification Number): A 4 or 8 digit device password.

Registration Protocol: A Registration Protocol is a (logically) three party in-band protocol to assign a Credential to the Enrollee. The protocol operates between the Enrollee and the Registrar and may receive support through a proxy.

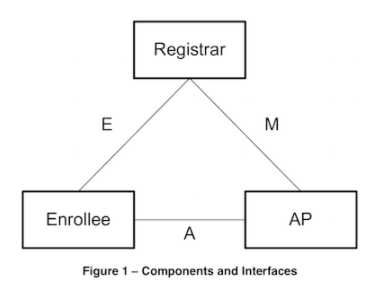
Registrar: An entity with the authority to issue and revoke Domain Credentials. A Registrar may be integrated into an AP, or it may be separate from the AP. A Registrar may not have WLAN capability. A given Domain may have multiple Registrars.

PCP: A Personal basic service set (PBSS) Control Point, peer-to-peer functionality in a 60GHz device, mandatory for all Wi-Fi CERTIFIED 60GHz Stations.

Push Button Configuration (PBC): A configuration method triggered by pressing a physical or logical button on the Enrollee and on the Registrar.

Stand-Alone External Registrar (SAER): An External Registrar that is not embedded in a wireless STA. For example, may be embedded in an Ethernet connected device, or may be software installed on any networking device.

4.4. WPS核心架构



如上图所示是WSC规范定义的主要组件和它们的接口。WSC中一共涉及到3个逻辑组件：Registrar（注册者）、AP和Enrollee（登记者）。通常情况下这些逻辑组件有可能是合并在一起的。例如，AP通常有一个内置的Registrar用来以独立模式添加Enrollee，可以用也可以不用到Web浏览器。

一个新的无线网络建立是通过上电AP和将AP接入一个外部网络（例如将AP连入一个有线网络），这时（无线）网络中还没有其他设备。下一步就是添加一个Enrollee或Registrar设备到网络。这里会用到AP和新设备之间的Registration Protocol。如果新设备是添加为一个外部的Registrar，那么后续的Enrollee可以通过该Registrar添加到无线网络。

WSC定义了新的802.11信息元素（IE, Information Elements）包含在beacon，probe request和probe response消息中。该IE的作用是宣告设备具备Wi-Fi Simple Configuration功能，需要注意的是，该IE中的信息是无需认证即可获取到的。

4.4.1. 接口E

这是Enrollee和Registrar之间的逻辑接口（物理上，AP可以扮演一个消息传送代理的角色）。接口E的目的是让Registrar来发现和签发无线网络认证凭据给Enrollee。接口E可能只有无线通信数据或通过带外方式进行数据通信。

Enrollee通过以下方式实现接口E：

在802.11 probe request消息中包含一个Wi-Fi Simple Configuration IE；

包含一个唯一、随机产生的设备密码显示在显示屏上或打印在贴纸上。设备密码用来在Registrar和Enrollee之间的带内交换认证；

可选支持一个或多个带外信道用于更简单和更安全的配置；

实现注册协议的Enrollee部分；

Registrar通过以下方式实现接口E：

处理Enrollee（设备或AP）在Probe消息（对于无线Registrars来说）或UPnP（对于基于IP的Registrars来说）中的（能力）发现数据；

实现注册协议的Registrar部分；

可选支持一个或多个带外信道用于更简单和更安全的配置；

如果需要的话，通过接口M使用Enrollee的MAC地址和认证凭据配置AP；

如果Registrar是一个AP则通过Probe-Responses响应Enrollee的Probe-Requests；

4.4.2. 接口M

接口M是AP和Registrar之间的接口。使得外部Registrar能管理⼀个WSC AP。WSC使用相同协议配置AP管理接⼝用于向Enrollee设备发送认证凭据。

AP通过以下方式实现接口M：

Acting as the Enrollee in the Registration Protocol, sending its own Discovery message across both 802.11 and UPnP. Support for at least three external Registrars is required.

Implementing the Management Interface described in the WFADevice and WFAWLANConfiguration Service documents. The AP is required to be a UPnP device that includes support for the Wi-Fi Simple Configuration proxy service.

Monitoring 802.11 probe request and EAP messages from Enrollees and converting them to UPnP Event messages. It also accepts UPnP actions and converts them to EAP messages according to the proxy function described in the WFAWLANConfiguration Service document.

Registrar通过以下方式实现接口M：

Processing AP Discovery messages across 802.11 and/or UPnP.

Subscribing to proxy events, receiving and processing Enrollee Discovery and Registration messages from the UPnP proxy and continuing the Registration protocol message exchange via UPnP actions.

Implementing the Registrar side of the Registration Protocol to gain management rights over the AP or to issue WLAN credentials to Enrollees

Configuring the AP with the MAC address and/or the per-device Credential of the Enrollee.

Implementing the Management Interface described in the WFADevice and WFAWLANConfiguration Service documents. This implementation requires the Registrar to function as a UPnP control point.

4.4.3. 接口A

AP和Enrollee之间的接⼝。接口A的功能是支持WSC WLAN发现以及支持Enrollee和IP协议Registrar设备之间的通信。

AP通过以下方式实现接口A：

Sending out 802.11 beacons indicating support for Wi-Fi Simple Configuration and generating Probe Response messages containing a description of the AP.

Implementing an 802.1X authenticator and the Wi-Fi Simple Configuration EAP method.

Proxying 802.11 probe request and EAP messages between Enrollees and external Registrars as described in the WFADevice and WFAWLANConfiguration Service documents.

Enrollee通过以下方式实现接口A：

Discovering a Wi-Fi Simple Configuration AP and/or wireless external Registrar and sending it 802.11 probe requests including the Enrollee Discovery data.

Implementing an 802.1X supplicant and the Wi-Fi Simple Configuration EAP method.

4.4.4. 注册协议

注册协议完成以下目的：

用于故障排查⽆线传输通道的基本连接问题；

使用带外（out-of-band）信息提供Enrollee和Registrar之间双向识别，支持认证凭据配置；

建立起每个设备的角色（AP，Registrar或Enrollee）；

从Registrar向Enrollee安全传输WLAN配置和其他配置信息 ；

建立起扩展主会话密钥（EMSK，Extended Master Session Key），可以用于保护其他应用层相关配置函数的安全性；

注册协议可以完全运行在带内，也可以完全运行在带外，也可以混用两种通信信道。这种灵活性使得协议能够充分快速适配利用不同带外传输机制的优势。

注册协议分为2个阶段，第一个阶段又被称为发现阶段，用于在Registrar和Enrollee之间交换描述信息。注册协议的发现阶段是必需的。

以基站作为Enrollee为例，发现阶段达成了2个目的：

允许Enrollee发现可用于注册的Registrar；

允许Enrollee设置自己为可发现的进而Registrar可以发现预期可注册的候选；

基站Enrollee可任选以下两种方法之一来完成发现阶段：

在扫描过程中基站Enrollee可以使用主动扫描发送包含WSC IE的probe request到AP。AP通过包含WSC IE的probe response进行响应。考虑到probe response中包含的WSC IE可能包含来自一个或多个Registrar（作为一个集群），这个方法仅限于如果Enrollee设置自己为可发现但不打算同时也发现外部Registrar的详细信息时的推荐。注意：基站Enrollee可选不关联到指定WSC配置（provisioning）如果包含基站MAC地址或通配MAC地址（ff:ff:ff:ff:ff:ff）的授权MAC地址子元素没有包含在Beacon或Probe Response帧中。

基站Enrollee可以选择关联到一个支持WSC的AP然后通过发送消息M1到Registrar来初始化注册协议。假设Registrar没有准备好注册候选Enrollee则会响应消息M2D。如果Enrollee打算在设置自己为可发现的基础之上发现可用的Registrar则推荐使用本方法。

如果是AP作为一个Enrollee，则发现过程由Registrar发起：

一个无线外部Registrar发送包含WSC IE并且请求类型属性设置为Registrar或WLAN Manager Registrar的probe request。AP响应包含WSC IE并且响应类型属性设置为AP的probe response；

一个有线外部Registrar使用合适的UPnP发现机制来识别AP。

在发现阶段Enrollee可以和网络中的多个AP或Registrar交换信息。在发现阶段结束之后，如果Enrollee和Registrar打算继续完成注册过程，注册协议的第二个阶段即行开始。第二个阶段的重头戏是认证凭据签发。

注册协议以锁定步骤方式工作，以M2、M2D或M8消息终结。终结消息如下描述：

M2D：这个消息代表Registrar无法认证Enrollee，但愿意给Enrollee提供有关Registrar的描述信息；

M2：这个消息可能包含来自Registrar的配置数据，在这种情况下整个注册协议终止。物理层连接隐式认证了其承载发送的数据。在这种情况下，注册协议的第一和第二阶段就合二为一了，只需要一个回合就可以完成注册协议；

M8：这个消息是注册协议第二阶段第3次握手的结果。第二阶段3次握手用于逐步完成基于Enrollee设备密码的Enrollee和Registrar的双向认证。WLAN认证凭据包含在发送给Enrollee的消息M8之中。

注册协议的执行可能因为错误或超时而终结。详细的注册协议描述请查阅《Wi-Fi Simple Configuration Technical Specification》的第7节。

4.5. 风险分析与安全机制设计

WSC为设备制造商提供了一系列选择，每种选择对应了不同的安全等级。系统的安全性取决于其最弱组件，因此，当使用WSC来构建一个无线网络时的有效安全强度取决于该无线网络中任意设备配网方式中安全性最弱的一种方法。目前主要有两种WSC配置模式：带内配置和带外配置。

对于带内配置来说，用到了Diffie-Hellman密钥交换和使用“设备密码”作为共享密钥的认证。设备密码可以取自Enrollee输入到Registrar，或者取自Registrar输入到Enrollee。输入设备密码的方式可以是手工输入或通过一个WSC NFC密码令牌的方式NFC录入。如果使用NFC输入设备密码，Registrar会得到Enrollee的DH公钥散列值。这显著加固了从Enrollee到Registrar的认证强度同时减少了攻击者盗取NFC标签来配置无线网络的风险。

使用带外配置时，无线认证凭据通过带外信道传输到Enrollee。认证凭据和配置通过带外信道传输时可以选择加密传输。当前版本的WSC规范支持的带外传输信道是使用WSC NFC配置令牌的NFC方式。

WSC带内注册协议设计用于提供对抗被动式监听攻击并且同时可以检测和保护系统免受主动暴力破解攻击。这就意味着如果Registrar遇到一个被判定为合法Enrollee的攻击者，它首先会检测到攻击者其实并不知道密码。这个检测发生在已经提供了足够多信息给暴力破解攻击尝试破解密码之前。尽管如此，如果Registrar多次使用相同PIN来运行和一个攻击者之间的注册协议，则攻击者有可能通过离线攻击发现PIN进而通过再次运行注册协议获得网络配置信息。为了堵上这个漏洞，如果在发送消息M6之后发生PIN认证或通信错误，Registrar应警告用户，并且不得自动重复使用该PIN。此外，如果Registrar检测到前述情况并提示用户从Enrollee设备获得一个新PIN码，则不应在未警告用户面临的可能攻击情况下再次接受相同的PIN码。如果使用了一个至少32字节随机产生的强设备密码（例如带外设备密码或设备专用密码）而不是PIN码，Registrar允许在发生错误时无需警告用户的情况下重复多次使用该强密码。PIN码重用的要求不适用于PBC（按钮）方法。

PIN码认证方式是WSC规范中强制要求所有设备必须支持的一种WPS初始化工作模式，并且规范原文中特别强调了PIN码（设备密码）的产生绝不应该依赖于设备特征（例如MAC地址或序列号）。按下按钮方式的WPS网络配置方法在WSC规范中被命名为Push Button Configuration，简称为PBC，这是一种可选的WPS工作模式。除此之外，如果设备支持NFC，则可以使用带外（out-of-band）方法来在待认证设备和注册服务器之间传输强口令（例如256bit随机值）。

All devices supporting Wi-Fi Simple Configuration shall provide at least one numeric Device Password (PIN) for initial setup that is unique and randomly generated per device. Although it is possible and permitted for two devices to have the same Device Password, a group of devices should not intentionally be assigned the same Device Password, and the Device Password SHALL not be based on other characteristics of the device, such as MAC address or serial number.

WSC规范要求无头（例如那些没有显示屏的）设备必需包含一个8位数字设备密码，被称为PIN码（通常是打印在一张设备贴纸上或物理雕刻在设备上）。无头设备的PIN码同样应该被配置到设备中，一般是在设备生产过程中完成的。

基于PIN码的设备密码是WSC的基础安全等级。由于8位数字PIN码中的其中1位被用于校验和位，因此PIN码包含大约23bit的信息熵。这种机制本身不是一个大问题，但最大的限制是PIN码可能是一个固定值（如果是静态的，通常是直接显示在一个标签纸上）。对于有显示屏的设备可以使用4位数字PIN码，4位数字PIN码没有使用校验和位。因为一个固定的PIN码会被重复使用，这就意味着易遭受主动攻击。WSC协议允许用户设置一个新的设备密码来代替默认的设备密码，虽然没有从本质上提升PIN码的安全性，但可以起到“安慰”和“缓解”部分用户的“安全感”焦虑。

无线网络中最主要的一类无头设备是AP。如果可能的情况下，每次运行注册协议时，PIN码都应该是新产生、仅使用一次的临时PIN用于建立和外部Registrar注册（此时AP扮演的是Enrollee角色）。但如果使用的是静态PIN码，AP应把多次失败认证尝试记录下来并进入锁定（locked down）状态（这个状态通过设置AP配置锁定状态为真来表示）。最多10次连续失败尝试，没有时间限制，来自任意数量的外部Registrar，AP就应强制回复到锁定状态，并且AP将无限期锁定下去（例如，直到用户主动解锁AP的用于外部Registrar的PIN）。

在锁定状态下，AP应拒绝以初始化AP配置模式与任何外部Registrar运行注册协议。这种技术保护AP的PIN码免受攻击者通过暴力破解攻击强制指定一个新的外部Registrar。在AP配置锁定状态，依然可以添加新的Enrollee设备到无线网络，但无法使用AP的PIN添加新外部Registrar。

AP可能通过其他方式进入锁定状态。例如，AP可以实现一个增量式临时锁定(incremental and/or temporary lockout)过程增加失败PIN尝试时的锁定时间。然而，即使实现了这些额外方法，AP仍然应该在遇到如上描述情况时进入无限期锁定状态。

AP应提供给用户解除锁定状态的方法。例如，可以通过AP的网页管理界面来解除锁定状态或重启AP电源来恢复PIN码可用于外部Registrar注册。

除了PIN码方式之外，无头设备还可以实现PBC（具备丰富用户界面的设备也可以支持PBC方法）。PBC方法的信息熵为0比特，仅可用于对抗被动式监听攻击。PBC方法仅可在没有支持PIN码的Registrar可用时并且无线网络用户愿意接受使用PBC带来的安全风险。PBC方式的安全性弱于其他方法，但可以用于兼容没有显示屏或其他带外传输机制的设备加入无线网络。

4.5.1. 设置一个新的无线网络

设置一个新的无线网络主要有两种场景：独立AP和使用外部Registrar。独立AP意味着Registrar内置在AP之中，使用外部Registrar可以支持一个或多个Registrar。

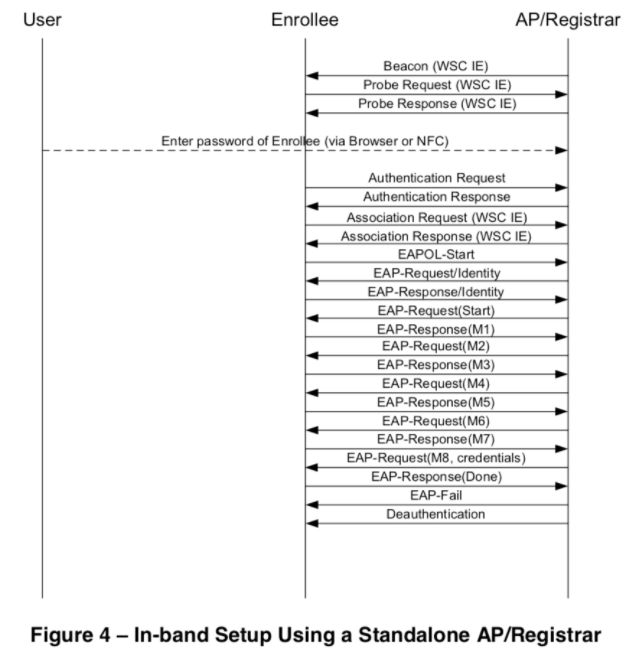
开启WSC支持的AP应开启SSID广播，如果SSID广播被禁用则对应的WSC功能会被禁用。如果开启了WSC，则AP不应阻止基于其他访问控制机制（如MAC地址过滤）的注册或阻止通信。

独立AP应自动选择一个SSID（通常是一个随机产生的SSID）和信道，并应同时默认开启WPA2-个人模式和设置一个健壮、随机产生的PSK。如果需要后向兼容不支持WPA2-个人模式的客户端则可以使用WPA混合模式。独立AP使用内置的Registrar来通过注册协议向Enrollee签发密钥。独立AP同样可以启用或关闭安全机制。AP也应提供恢复出厂设置选项来清除任何用户自定义的配置和密钥。

使用外部Registrar方式的AP设置一个新的无线网络的详细方式请查阅《Wi-Fi Simple Configuration Technical Specification》的第5节。

4.5.2. 为该无线网络添加新成员设备

理想情况下，一个支持WSC的AP应支持多密钥模式。但实际情况是，AP也允许使用一个所有设备共享的单一WPA2-个人模式密钥。和设置一个新的无线网络类似，为该无线网络添加新成员设备也会根据AP使用的是内置还是外部Registrar而分为多种不同的添加流程。除此之外，还要考虑带内和带外两种数据交换通道模式下的流程差异。以下用“标准AP内置Registrar带内设备注册”为例说明新设备如何通过WSC加入到一个无线网络。



上述流程图中我们可以看到：用户需要把Enrollee的设备密码录入到AP/Registrar中，可以是键盘录入也可以是带外信道。上述流程图没有展示probe消息交换之后因为Enrollee在等待用户把Enrollee的设备密码配置到AP/Registrar内而可能发生的M1和M2D消息交换，。

Enrollee通过probe request发送发现（Discovery）数据到WSC AP。AP或无线Registrar在probe response中响应它自己的发现数据；

用户按照提示使用键盘界面或带外信道输入Enrollee的设备密码到AP/Registrar；

Enrollee使用“WFA-SimpleConfig-Enrollee-1-0”标识连接和发起802.1x认证过程；

Enrollee和Registrar通过注册协议完成M1~M8消息交换来配置Enrollee；

Enrollee断开连接再使用上一步得到的无线网络认证凭据重新连接加入目标无线网络。

其中的M1~M8消息格式基本结构如下：

M1 = Version || N1 || Description || PK\_E

M2 = Version || N1 || N2 || Description || PK\_R [ || ConfigData ] || HMAC\_AuthKey(M1 || M2\*)

M3 = Version || N2 || E-Hash1 || E-Hash2 || HMAC\_AuthKey(M2 || M3\*)

M4 = Version || N1 || R-Hash1 || R-Hash2 || ENC\_KeyWrapKey(R-S1) || HMAC\_AuthKey (M3 || M4\*)

M5 = Version || N2 || ENC\_KeyWrapKey(E-S1) || HMAC\_AuthKey (M4 || M5\*)

M6 = Version || N1 || ENC\_KeyWrapKey(R-S2) || HMAC\_AuthKey (M5 || M6\*)

M7 = Version || N2 || ENC\_KeyWrapKey(E-S2 [||ConfigData]) || HMAC\_AuthKey (M6 || M7\*)

M8 = Version || N1 || [ ENC\_KeyWrapKey(ConfigData) ] || HMAC\_AuthKey (M7 || M8\*)

上述结构定义中的符号使用说明如下：

|| 消息中的参数（字符串）拼接操作

密码学函数名右侧\_紧跟的是该算法使用的密钥

Mn\* 不包含HMAC-SHA-256签名值的Mn消息

Version 对应注册协议消息版本号

N1 Enrollee设置的128bit随机数。每次注册协议实例都应该使用新随机值，Registrar应使用关联Enrollee的M1消息中的N1值。

N2 Registrar设置的128bit随机数。每次注册协议实例都应该使用新随机值，Enrollee应使用关联Registrar的M2D/M2消息中的N2值。

Description 包含一个人类可读的发送设备描述信息（UUID，设备商名称，型号，MAC地址等）和设备能力信息（支持算法范围，I/O信道，注册协议角色等等）。在802.11 probe request和probe response消息中也有

PK\_E和PK\_R分别是 Enrollee 和 Registrar 的 Diffie-Hellman 公钥。如果在未来使用了新的密钥交换算法（例如椭圆曲线），则会使用不同的协议版本号标识

AuthKey由DH算法中产生的会话密钥、随机值N1和N2以及Enrollee的MAC地址计算推导出。如果M1和M2通过不受中间人攻击威胁的通道传输，Enrollee的设备密码可以在密钥推导过程中被省略

E-Hash1和E-Hash2是Enrollee用于证明它掌握它自己的设备密码（分别对应前半部分和后半部分）

E-Hash1 = HMAC\_AuthKey(E-S1 || PSK1 || PK\_E || PK\_R)

E-Hash2 = HMAC\_AuthKey(E-S2 || PSK2 || PK\_E || PK\_R)

R-Hash1和R-Hash2是Registrar用于证明它掌握Enrollee的设备密码（分别对应前半部分和后半部分）

R-Hash1 = HMAC\_AuthKey(R-S1 || PSK1 || PK\_E || PK\_R)

R-Hash2 = HMAC\_AuthKey(R-S2 || PSK2 || PK\_E || PK\_R)

任何一方证明自己掌握设备密码时，设备密码首先被转换格式为2个128bit的PSK

PSK1 = first 128 bits of HMAC\_AuthKey(1st half of DevicePassword)

PSK2 = first 128 bits of HMAC\_AuthKey(2nd half of DevicePassword)

例如，如果PIN码是39358448，则转换之后的存储数据结构是8字节ASCII字符串39358448。PSK1来自于对3935计算HMAC取前128bit，PSK2来自于对8448计算HMAC取前128bit

HMAC\_AuthKey(…) 使用AuthKey作为密钥的HMAC函数。为了减小消息长度，256bit HMAC输出的前64bit被包含在认证属性字段，默认使用 HMAC-SHA-256 per FIPS 180-2 and RFC-2104.

ENC\_KeyWrapKey(…) 使用KeyWrapKey作为对称加密密钥的对称加密函数，默认使用AES-CBC per FIPS 197, with PKCS#5 v2.0 padding；

ConfigData 包含提供给 Enrollee 的 WLAN 配置和认证凭据，其他配置信息也可能包含于此。虽然 ConfigData 在这里被定义为加密存储，但只有密钥和密钥绑定信息是强制要求加密的。其他配置数据可选择加密。发送者自行决定是否加密 ConfigData 哪一部分。

关于注册协议、消息验证等更多详细描述请查阅《Wi-Fi Simple Configuration Technical Specification》的第6节。

5. WPA/WPA2 企业级认证

Wi-Fi联盟把使用“预共享秘钥”的WPA或WPA2版本叫做“WPA-个人版”或“WPA2-个人版”（WPA-Personal or WPA2-Personal），用802.1X认证的版本叫做“WPA-企业版”或“WPA2-企业版”（WPA-Enterprise or WPA2-Enterprise）。

802.1X 复杂之处，在于它本身只是一套架构。它是 IEEE 采用 IETF 的可延伸身份认证协议 (Extensible Authentication Protocol，简称 EAP)制定而成。EAP 属于一种架构协议，最初规范于RFC 2284，后来经由RFC 3748的更新。EAP本身并未规范如何辨识使用者，但允许协议设计人员打造自己的 EAP 认证方式(EAP method)，亦即用来进行交换过程的子协议。EAP 认证方式可以有不同目的，因此通常根据特殊情况的需求，采用不同的方式来辨识使用者的身份。不过在详细探讨不同认证方式之前，必须先了解 EAP 的运作方式。

注释：802.1X 里的 X 以大写表示。在 IEEE 的命名原则里，小写字母(比如 802.11a 和 802.11b)保留给修订既有标准的附加规格使用。大写字母则用加独立的规格。既然 802.1X 本身是个完整独立的协议规格，因此以大写字母来表示。

5.1. 可延伸身份认证协议（EAP）

802.1X 的基础是 EAP。由于近来无线网络的发展，原本的 EAP 标准已经不再使用，更新后的版本见于RFC 3748。当PPP在1990年代初期间世时，有两种协议可用来辨识使用者的身份，两者都会用到 PPP 协议编号。身份认证不可能“一体适用”，当时算是相当宽泛的研究领域。IETF 并未废弃可能就此无用的 PPP 协议编号，而是制定了 EAP 标准。 EAP 是一种简单的封装方式，可以执行于任何的链路层，不过它在 PPP 链路上并未广泛使用。 EAP 的基本架构如下图所示，在设计上是为了能够执行于任何的链路层，以及使用各种的身份认证方式。

除了流量控制与协商信息，EAP 也会为身份认证的方式指定类型代码。EAP 会把证明使用者身份的过程，授权给一个称为 EAP method 的附属协议，EAP method 是一组验证使用者身份的规则。

使用 EAP method 的优点是：EAP 从此可以不用去管验证使用者的细节。如果需求改变，这是无线网络普及之后常有的状况，就可以开发新的 EAP method 来满足这个需求。以下列出了 一些 EAP method 以及它们的类型代码。无线局域网络常用的 EAP method 将于本章稍后详细说明。

类型代码 身份认证协议 说明

4 MD5 challenge EAP 中类似 CHAP 的认证方式

6 GTC 原本打算搭配 RSA SecurID5 之类的标记卡(token card) 一起使用

13 EAP-TLS 以数字证书相互认证

21 TTLS 管道式 TLS; 以 TLS 加密保护较脆弱的身份认证方式

25 PEAP 防护型 EAP; 以 TLS 加密保护较脆弱的 EAP 认证方式

18 EAP-SIM 使用移动电话的SIM卡进行身份认证

29 MS-CHAP-V2 Microsoft 之经加密的密码身份认证; 兼容于 Windows 网络

可延伸性(EAP 当中的 E)既是 EAP 最大的优点也是最大的缺点。可延伸性让协议得以在新的需求出现时开发新的功能，同时正因为本身的可延伸性，EAP 已经从保留 PPP 协议编号的方式，转变为无线局域网络的安全防护基础。不过，要正确部署 EAP 可能不容易，因为要选择正确的协议选项之前必须先理清一大堆问题。EAP 之所以具有弹性，关键在于它本身只是一个架构。当新的 需求出现，就可以设计出新的认证方式，就算要用于无线局域网络也不成问题。

无线局域网络之安全防护协议这个议题十分广泛，本章试图一点一滴逐步勾勒出协议运作机制的细节。很多复杂的细节只会以文字来描述，不会用详细的封包格式图来说明。

5.1.1. 加密的方式

EAP method 的选择通常取决于后端所使用的身份认证系统。早期的 EAP method 主要将焦点摆在如何提供与认证服务器间的传输管道。针对无线网络所设计的新型认证方式，除了可以跟认证服务器进行沟通，也符合无线局域网络特有的三项要求:

为使用者的数据提供坚固的加密防护

定义上，无线局域网络应该被视为一种开放介质。若要确保安全，任何通过无线网络传送的数据均须加以保护。大多数针对无线局域网络所设计的 EAP method 均采用 TLS 来提供私密数据的加密防护。

早期无线局域网络协议的设计，将身分认证视为基站对使用者提出的要求。随着基站价格的滑落，如今攻击者已经能够部署「伪装」的基站，用来窃取用户的私密数据。依常理判断，除了使用者的身份认证，用户端设备还必须验证它们所连接的网络是否正确无误。

衍生密钥使用事先约定密钥的 WEP 机制，无法为电波链路上的帧提供多少保护。较坚固的安全防护协议必须使用由乱数集区(entropy pool)衍生而成的动态密钥。提供坚固之加密防护的副作用是，此类协议还会产生一份共享的加密串流，可用来传递密钥给链路层安全防护协议。

LEAP

Cisco 专属的 Lightweight LAP(轻量级 EAP，简称 LEAP)是最早广泛使用的无线网络身份认证方式。相对于通过手动方式设置密钥的 WEP，LEAP 可说是往前迈进了一大步，虽然它还有很多地方有待改进。基本上，LEAP 进行了两次 MS-CHAP Version 1 交换程序。第一次由使用 者对网络进行验证，第二次由网络对使用者进行身分认证。动态密钥是衍生自 MS-CHAP 交换程序。

一些最糟糕的安全问题都是由 MS-CHAP version 1 所引起的。MS-CHAP version 1 有一些安全上的问题，容易导致字典式攻击。攻击 LEAP 弱点的漏洞利用代码目前已经广泛流传。

LEAP 算是一种过渡性的解决方案，虽然相较于手动设置密钥的 WEP 在安全性上有显著的优点，不过使用过时的 MS-CHAP 作为核心反而缩短了它的使用寿命。等到其他协议陆续问世，LEAP 就没有存在的必要了。

EAP-TLS

Transport Layer Security(简称 TLS)协议原本就是设计来用在易遭窥视的链路层上。TLS 的前身是保护网际网络交易安全的协议 Secure Socket Layer(简称SSL)。从许多方面来看，无线局域网络的使用案例与网际网络类似。数据必须在完全不可信赖甚至是攻击者建立的网络环境中进行传送。TLS 的目的就是在不可信赖的网络环境中建立一条可信赖的通信信道。

TLS 通过证书交换来进行相互认证。使用者必须将数字证书送交认证服务器以进行验证，但是认证服务器也必须提供自身的证书。通过可信赖的证书发行机构验证服务器的证书真伪，用户端就可以确定所连接的网络经过证书发行机构授权无误。

EAP-TLS 是第一个符合无线网络三项要求的身份认证方式。证书提供可靠的“使用者对网络”以及“网络对使用者”双向认证。相互认证可以防范所谓的“假”（rogue）基站，让用户端得以判定基站是否由正确的部门而非为了窃取密码的攻击者所设立。TLS 还会建立一组主秘钥，用来衍生出链路层安全防护协议所需要的密钥。

EAP-TLS 虽然安全，不过并未被广泛使用。无线网络中任何潜在的用户都必须配备自身的数字证书。产生与传递证书以及遵循验证程序都是较大的挑战。已经采用公开密钥基础建设的机构如要使用 EAP-TLS 就相当容易。有些机构并不想组建 PKI，而另外选用别种方式。

EAP-TTLS与EAP-PEAP

实际上，需要使用 PKI 是无线局域网络上推行坚固的身份认证时主要的障碍。PKI 不论在技术或者程序上都是一项艰巨的任务。大多数的组织宁可利用现有的身份认证系统，例如 Windows 域控制或 活动目录，轻量级目录访问控制协议，或者 Kerberos域 。使用现成的帐号比起重新建立一套并行的身份认证系统来得简单。有两种 EAP method 能够搭配所谓的“传统（legacy）身份认证方式”一起使用，分别是管道式 TLS（Tunneled TLS，简称 TTLS）与防护型 EAP（Protected EAP， 简称 PEAP）。

TTLS 与 PEAP 的运作方式类似。首先，协议会使用类似 EAP-TLS 的方式建立起一个 TLS 管道。进行下一个步骤之前，会先使用认证服务器的数字证书来验证此网络是否可受信赖。第二个步骤是使用 TLS 管道为传统的身份认证协议加密，然后以之验证使用者身份。有时候第一个步骤也称为外层（outer）身份认证，因为它是用来保护第二个或者内层（inner）身份认证的管道。

认证还是免不了，但只有外层身份认证需要用到。TTLS 与 PEAP 将证书数目从千百张缩减为屈指可数，只有认证服务器需要使用证书。这就是为什么 TTLS 与 PEAP 远比 EAP-TLS 受到欢迎。一般机构可以自行设立小型的证书管理中心，不必依赖外部证书机构所签发的昂贵证书。

TTLS 与 PEAP 之间有些微的差异，在于内层身份认证的处理方式。TTLS 使用加密管道来交换属性-值对（Attribute Value Pair，简称 AVP）, PEAP 则是在管道内进行第二次 EAP 交换程序。采用 AVP 使得 TTLS 较具弹性，因为 AVP 可用来执行 EAP method 未提供的身份认证方式。

使用 TTLS 与 PEAP 的好处是，内层与外层身份认证可以使用不同的使用者名称。这两种协议在进行外层身份认证时可以匿名进行，只有在加密管道中才会显示使用者的真实身份，如此一来就不会在未经加密的帧中暴露使用者名称。不过并非所有用户端软件均可隐藏使用者的身份。

5.1.2. 非加密式EAP认证方式

若无坚固的加密防护，有些 EAP method 并不适合直接用于无线网络。不过，它们可以作为PEAP 或 TTLS 的内层身份认证方式。

MD-5 Challenge

MD-5 挑战相当于 RFC 1994 所规范的 CHAP 协议。身份认证要求中包含了给用户端的挑战。用户端只要能够成功回应挑战，就可以证明它的确握有共享密钥。所有的 EAP 实现必须支持 MD-5 Challenge。不过，它在无线网络领域并末得到广泛使用，因为它无法在无线网络上提供动态密钥。

Microsoft CHAP version 2

简称 MS-CHAP-V2，最早出现在 Windows 2000 操作系统，规范于 RFC 2759。它被设计来解决 MS-CHAP的缺陷，除了移除旧式用户端进行密码编码时的弱点，还提供相互认证以及改善密钥产生与更换的机制。

MS-CHAP-V2 广泛使用于微软工作站，通常作为内层身份认证方式，并搭配 PEAP 一起使用。MS-CHAP-V2 是 Windows 网络最常见的内层身份认证方式。当作 EAP 认证方式使用时，EAP-MSCHAP-V2 可以搭配 TTLS 或者 PEAP 一起使用。

5.2. 802.1X 网络端口身份认证

在链路层采用身份认证机制并不是什么新鲜事。网络连接基于端口的身份认证在拨号访问服务器上已经使用多年。大部分的机构早已采用各种机制，作为使用者身份认证之用，例如 RADIUS 服务器与 LDAP 目录服务。以 PPP over Ethernet(简称 PPPoE)的使用者认证，为以太网访问进行把关的做法并不难理解，不过如此一来会增加系统的负载与复杂度。因此，IEEE 采用 PPP 认证协议，据以开发针对局域网络的版本。最后出炉的标准称为 802.1X，即基于端口的网络访问控制（Port-Based Network Access Control）。

802.1X 为认证对话程序定义了三个组件，如图 6-6(a)所示。申请者（supplicant）是寻求访问网络资源的使用者机器。网络访问由认证者（authenticator）所控制，它扮演着传统拨号网络中访问服务器一样的角色。申请者与认证者在协议规范中称为端口认证实体（Port Authentication Entities，简称PAE）。认证者只负责链路层的身份认证交换程序，并不维护任何使用者信息。任何认证要求均会被转送至认证服务器（例如 RADIUS）进行实际的处理。

支持 802.1X 的设备上，各个端口若非处于授权状态（亦即可以使用该端口），就是处于未授权状态（亦即无法使用该端口）。不过就算处于未授权状态，协议规范中还是允许使用 DHCP 以及其他初始化信息，如果网络管理人员允许的话。

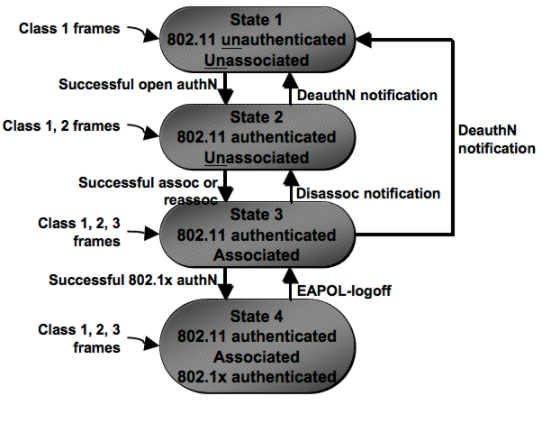
整个身份认证交换程序在逻辑上是通过申请者与认证服务器来完成的，认证者只是扮演代理服务器的角色。图 6-6(b)所示为逻辑上的协议架构。申请者与认证者之间（即前端），使用由 802.1X 所定义的 EAP over LAN（简称 EAPOL）协议。在后端，则是通过 RADIUS 封包来传递 EAP。有些文献称之为 EAP over RADIUS。就算端口尚未得到授权，也尚未取得 IP 地址，申请者还是能够持续与 RADIUS 服务器进行 EAP 交换程序。

图 6-6 可以有两种不同的解读方式。在企业场合中，申请者乃是位于企业网络边界的一部主机，而 RADIUS 服务器则位于企业核心。本图同时显示了一家以 802.1X 认证使用者身份的 ISP，图左为 ISP 的访问范围，而图右则是 ISP 的骨干网络。

RADIUS 的好处是支持多种使用者数据库。除了本地的数据库，RADIUS 服务器也可以当成通往 LDAP 目录、Unix NIS 或 PAM、Kerboros realms、Windows 使用者帐号、甚至是其他 RADIUS服务器的网关。RADIUS 相当有弹性，甚至可以用统一格式整合截然不同的使用者数据库。

802.1X 只是一个架构，并非一套完整的规范。实际的认证机制，其实是通过认证服务器来完成的。802.1X 所提供的机制，主要是用来发出挑战信息以及确认或拒绝访问，实际上并不负责判断对方是否有权访问。改变认证的方式不需要大幅更动使用者的设备，或整个网络的基础建设。认证服务器可以重新设置配置，以便外挂其他新的认证服务，不必更换使用者所使用的驱动程序或交换机的固件。

802.1X 为任何局域网络，包括无线局域网络，提供了一个使用者身份认证的架构。就本书的目的而言，无线网络上 802.1X 中的端口，相当于无线设备与基站间的连接（association）。 当链路层开始作用时，Association Request 与 Association Response 帧交换成功的信息会告知 802.1X 状态引擎（state engine，如下图所示）。一旦连接成功，工作站就可以开始进行 802.1X 帧交换程序，尝试取得授权。802.1X 身份认证交换程序与密钥传递完成后，使用者就会收到网卡已经连接的信息。



1. 无线接入网入侵与防御

1. 绕过那些似是而非的安全机制

1.1. 禁⽌SSID⼴播

AP在主动广播的beacon frame中包含SSID字段值，所有AP的默认出厂设置都是开启了SSID广播功能的。但同时也提供了关闭SSID广播功能的开关。

AP如果关闭了SSID广播功能（部分产品的管理界面使用的功能术语是：隐藏无线网络），手机等无线客户端在“刷新”无线网络列表时会忽略掉这些网络（因为收到的AP广播beacon frame中没有网络名称信息，所以无法在界面上“显示”），造成了这些无线网络似乎就“隐身”了的假象，进而给人一种假象：

攻击者发现不了我的网络，就无法发起攻击，所以网络就安全了。

事实上，攻击者可以选择“被动发现”的方式来“发现”禁止SSID广播的无线网络。这是因为：

当有STA加⼊隐藏SSID的AP时，Probe Request 中包含该无线网络的SSID；

该AP下客户端断线重连AP时发送的Association Request、Probe Request和AP发送的Probe Response中都会包含目标无线网络的SSID。

攻击者只需要“静静”的抓包，就可以“发现”目标网络的SSID。

上图中，红色方框高亮的BSSID：08:57:00:6B:11:D0对应AP在airodump-ng的控制台中ESSID字段显示为<length: 0>，这表示抓到的该AP广播的beacon frame中ESSID字段值为空，符合“禁止SSID广播”设置的AP的表现。

说明：不同AP的“禁止SSID广播”的实现方式会有差异，有些厂商的AP会在beacon frame字段中根据实际网络的SSID字段长度（假设为N字节）填充对应长度的\x00字符，这时在airodump-ng的控制台中ESSID字段就会显示为<length: N>。

下图中，通过向指定AP发送1个解除认证⼴播包，并等待客户端重连和观察airodump-ng的输出信息变化，可以看到原先BSSID 08:57:00:6B:11:D0 对应 AP 在airodump-ng的控制台中ESSID字段变化为HackMeIfYouCanHidden，这就是刚才隐藏的目标网络SSID被我们“发现”了。

aireplay-ng --deauth 1 -a 08:57:00:6B:11:D0 mon0 --ignore-negative-one

另外，我们还注意到，该目标网络当前确实有一个已经处于连接状态的无线客户端，它的MAC地址是：E0:F8:47:DC:38:C4。

下图是我们用wireshark分析我们刚才在“发现隐藏SSID”演示实验中的抓包结果截图，其中HackMeIfYouCanHidden确实出现在了无线客户端发送给AP的Association Request帧之中。

使用iwlist wlan11 scanning | grep -A 30 '08:57:00:6B:11:D0'可以再次证明，该AP的广播beacon frame中确实没有有效的ESSID值，是一个“隐藏无线网络”。

1.2. MAC地址过滤

如下图所示是一个典型的MAC地址过滤设置界面，图中的过滤规则里的“禁止”和“允许”分别对应“黑名单”和“白名单”机制。由于无线网络即使启用了数据加密功能，但MAC地址依然是明文包含在无线数据帧中的，通过嗅探模式抓包可以轻易的了解到目标网络中有哪些设备处于已连接状态。对于白名单过滤模式来说，攻击者只需要把自己设备的MAC地址修改为网络中已连接客户端的MAC地址，即可绕过白名单限制。当然，黑名单限制就更容易绕过了，攻击者如果怀疑自己的MAC地址被加入到了MAC地址过滤黑名单，只需要修改掉当前使用的MAC地址重新尝试连接即可。

在Linux系统中修改网卡MAC地址的命令行方法如下：

ifconfig wlan0 hw ether 00:11:22:33:44:55

Windows系统可以通过修改注册表项（例如：HKLM\SYSTEM\CurrentControlSet\Control\Class\{4D36E972-E325-11CE-BFC1-08002bE10318}，取决于网卡驱动和操作系统的支持情况），也可以使用一些第三方工具，例如TMAC、MAC Makeup等。

在Linux系统中可以使用命令行工具wpa\_supplicant连接WPA/WPA2认证方式的无线网络。首先把明文Passphrase通过Hash运算生成PSK，这会用到命令行工具 wpa\_passphrase 生成 wpa\_supplicant 所需的最小配置，可以快速连接到已知 SSID 的无线网络。例如：

$ wpa\_passphrase TargetSSID

# reading passphrase from stdin

hellopassword

network={

ssid="TargetSSID"

#psk="hellopassword"

psk=4cc54666ad54da9f19f3e6fde3b8521bd3e06d8be19928c3864aea13db3d5a75

}

上例表明，wpa\_supplicant 可以与 wpa\_passphrase 协同工作，只需简单地这样即可做：

# 注意: 由于存在进程替换，这个命令不能带着 sudo 执行，必须切换到 root 身份。否则会报错：

# 上一步 wpa\_passphrase 的输出结果也可以直接保存为一个独立配置文件，在这一步就可以替换 -c 参数的值为配置文件路径

# wpa\_supplicant -B -i wlan1 -c <(wpa\_passphrase TargetSSID hellopassword)

一旦完成了网络认证过程，剩下的就可以和正常自动连接无线网络一样，手工通过dhclient wlan1这样的命令来执行DHCP请求，获取网络自动配置。如下图所示展示了这个过程用到的主要命令和命令执行结果。、

1.3. 禁用DHCP，使用静态IP地址分配

无论是有线局域网还是无线局域网，监听ARP广播都是我们发现当前网络IP地址段配置的最佳方法。ARP广播的典型发生场景包括了：同⼀局域⽹下客户端相互之间首次访问；客户端要访问外⽹，寻找⽹关地址。对于一个正常提供网络服务的无线网络来说，ARP广播是非常频繁发生的。因此，只要我们完成了无线网络的身份认证和链路层连接建立，剩下的只需要耐心的监听ARP广播和分析就可以知道当前网络的IP地址配置和网关IP。

2. 已有安全机制的漏洞原理

2.1. 开放式认证

开放式认证的本质就是使用明文通信协议，而无线嗅探可以直接获得⽆线数据并分析应用层数据。如果应用层数据没有被加密传输，则“嗅探”攻击可以轻易得手。另外，一些采用HTTP协议的Portal认证方式还面临“重放攻击”和“仿冒身份”威胁。例如，可以直接提取出已通过Portal认证的客户端HTTP会话数据中的cookie信息，轻易“克隆”出另一个一模一样的合法、已认证客户端会话。如果是基于MAC地址过滤方式的AP，则已通过认证的⽆线STA的MAC地址被克隆后，克隆STA就可以连入当前无线网络。

如上动图所示，通过无线数据嗅探的方式提取到了一个已登录用户的HTTP会话使用的cookie值，只需要简单的配合使用一个浏览器的插件（例如上图中使用的是Chrome浏览器的Modify Headers插件）把该cookie值“克隆”到当前浏览器中，一刷新，就可以在无需知道目标用户登录账号信息的情况下“仿冒”身份使用网站的注册用户服务。

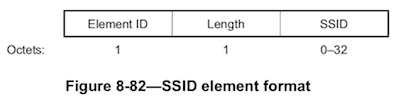
对于使用开放式认证服务的无线网络用户来说，要避免使用明文通信协议的应用，更不要在这样的网络中输入隐私信息和数据。对于网络管理员来说，使用HTTPS方式的Portal认证代替明文的Portal认证。如果有条件，可以为用户提供VPN服务，减少明文数据传输。

2.2. Evil Twin

IEEE 802.11协议中对ESSID的使用没有任何强制认证机制，任何人都可以任意声明ESSID，STA无法区分同名ESSID谁是冒牌，BSSID也可以被任意伪造，DS机制允许单个ESSID对应关联多个BSSID。无线客户端的无线网络列表选择界面是按照独立ESSID进行显示的，如果有多个AP设置了相同的ESSID也仅仅只会显示成一个网络名称。这样，当用户手动选择网络时或根据历史连接记录自动加入网络时，客户端会大概率连入信号强度更高的无线网络，伪造同名ESSID的AP往往会将自己的信号发射功率调的较高，以诱骗更多受害者连接自己。

如果客户端成功加入到Evil Twin方式搭建的AP，则该BS的服务提供AP对当前BS内的STA的所有通信流量可见、可控，诸如DNS劫持和投毒、DHCP控制和ARP投毒等手段都将轻而易举。同时由于AP负责提供客户端的上网服务，因此所有上网流量都可以被AP引导到自己控制的服务器上并进行静默记录，实现流量监控。更进一步的，还可以提供透明代理，实现更强大的中间人攻击效果。

2.3. Evil ESSID



如上图所示是IEEE 802.11协议官方文档对SSID元素格式的定义，SSID字段长度范围限制为0~32字节，但在协议规范中并没有限制字段取值范围和使用的字符编码标准。

正是由于协议规范文档中的上述“宽泛”定义，我们在真实世界中可以发现几种常见的利用ESSID字段的攻击向量或滥用行为。

使用airbase-ng可以用一个无线网卡快速创建多个ESSID“假网络”，操作步骤如下：

# cat essids.txt

你好

听说

你在

抓包

看看

你能

抓到

我吗

# airbase-ng --essids essids.txt wlan0mon

2.4. TKIP缺陷

2002年10月31日，Wi-Fi联盟提出临时密钥完整性协议（Temporal Key Integrity Protocol, TKIP），归类于WPA标准的一部分。IEEE随后在2004年7月23日的IEEE 802.11i-2004报告中背书临时密钥完整性协议并同时提出基于计数器模式密码块链消息完整码（Counter Cipher Mode with block chaining message authentication code Protocol, CCMP）协议的802.1X和AES协议这些更坚固的安全协议。之后Wi-Fi联盟接受了IEEE的相关报告并冠以“WPA2”这个商业名称发布。

临时密钥完整性协议随后由于安全性原因于2009年1月被IEEE废弃。ZDNet于2010年7月18日的报告中表明Wi-Fi联盟应当禁止所有Wi-Fi设备使用WEP和临时密钥完整性协议。自2012年的802.11标准中，TKIP已不再视为完全，且即将废弃。

TKIP本质上是一个WEP补丁，解决了攻击者通过获得少量的路由器流量解析出路由器密钥的问题。为了解决这个问题，TKIP每隔几分钟就给出新的密钥。不给攻击者提供足够的数据来破译密钥或算法所依赖的RC4流加密。

虽然当时TKIP还提供了一个较为完善的安全升级，但是对于保护你的网络不受黑客攻击这件事上，它还是不够全面的。其中最大的漏洞被称为“chop-chop attack”，是发生在加密本身释放之前的攻击。

黑客可以利用chop-chop attack截获并分析网络中产生的数据，并最终破译出密钥、明文显示其中的数据。

我们先来回顾一下TKIP缺陷被发现和漏洞利用的历史：

2008年11月8日，两名德国人Martin Beck, Erik Tews 发表了论文：Practical attacks against WEP and WPA，在部分条件下可导致攻击者对数据包进行解密，不过这个不是密钥恢复攻击，攻击者只能恢复用于加密验证报文的密钥而不是用于加密数据的密钥。使用恢复的密钥，只有捕获的报文仅在有限的7次尝试下的窗口机会中进行伪造。攻击者每次只能解密一个报文，当前的速率是12-15分钟一个报文，另外只有报文从无线AP发送到客户端时才能被加密。只有配置了TKIP作为加密机制的设备受此漏洞攻击，使用AES-CCMP密文套件的WPA2不受此漏洞影响。同时，这个攻击方式存在限制性条件：目标WPA无线网络实现应支持IEEE 802.11e QoS特性（有时被称为Wi-Fi Multi-media, WMM）。这种攻击方式被命名为：贝克-特夫斯攻击法（Beck-Tews Attacks）。

Beck-Tews攻击可以看做是针对WEP的断续攻击（Chop-Chop Attack）的升级扩展。因为WEP所使用的校验和算法CRC32在密码学上并不安全，攻击者可以猜测到数据包中的单个字节，而后AP会对猜测正确与否作出确认或拒绝传输。如果猜测正确，攻击者可以发现猜测正确并继续猜测下一个字节。不过相较于针对WEP的断续攻击，攻击者如果猜错必须额外等待60秒才能继续猜测。这是因为尽管TKIP仍然使用CRC32校验和算法，但由于使用了序列计数器（称为Michael）而可以缓解攻击。如果一个AP在60秒内收到了2个Michael序列计数器错误，AP随后将启动反制措施，重组TKIP的会话密钥（Session Key），从而改变随后的密钥流。由此，Beck-Tews攻击则等待一个适当的时间从而避免该反制措施。由于ARP包可以非常简单通过包大小鉴别，并且包中内容通常很容易猜到（大多ARP包内容相似），从而留给攻击者需要猜测的包大小则变得非常有限（大约为14字节）。通常来说在寻常的网络配置下只需要12分钟即可破解12加密字节。

当攻击者可以接触到全部的加密包内容后，对于余下在每个包内的明码内容进行去除，攻击者即可轻易获得包内的密钥流，同时还可获得对话的MIC码。应用这些信息攻击者可以重建新的数据包并在网络上传送。Beck-Tews攻击使用QoS通道传送新组建的数据包从而绕开WPA应用的重放攻击防护。由此攻击者可以传送数据包已达到其他攻击方式，比如ARP欺骗攻击，拒绝服务式攻击或其他类似攻击。

Beck-Tews攻击的概念验证代码以Tkiptun-ng工具的形式公开，感兴趣的同学可以研究一下这个工具的源代码。

2009年10月，挪威科技大学的费恩·霍尔沃森（Finn M Helvorsen）在他的硕士论文 Cryptanalysis of IEEE 802.11i TKIP 中进一步改进了Beck-Tews攻击（并提供了tkiptun-ng代码的补丁），使得攻击者可以在18分25秒内注入一个比普通ARP包大的多的数据包（596字节）。

2009年两名日本人Toshihiro Ohigashi , Masakatu Morii以Beck-Tews攻击法为基础，进一步优化攻击。在论文 A Practical Message Falsification Attack on WPA 中发布了一种更简单且更快速的类似攻击方式——“大东-森井攻击”（Ohigashi-Morii Attack）。该种攻击在使用Beck-Tews攻击的同时使用中间人攻击，同时不需要AP必须使用QoS。在理想情况下，1分钟即可完成一次Beck-Tews攻击。

2015年7月，两名比利时研究人员Mathy Vanhoef和Frank Piessens，展示了在52小时内解密了受TLS保护且使用RC4加密的HTTPS会话中使用的Web cookie，而此前利用RC4漏洞的攻击需要花费2000小时。通过使用固定纯文本恢复技术——被称为RC4 NOMORE(Numerous Occurrence Monitoring & Recovery Exploit)，攻击者可诱骗用户访问代码来产生足够的数据，成功地解密出用户的加密cookie值。

这种攻击并不局限于解密cookie，任何被重复加密的数据或信息都可以被解密。例如，对于使用TKIP加密技术的无线网络来说，这种攻击只需要一个小时来执行。作者已经把相关工作通过论文的形式发表在USENIX 15 All Your Biases Belong To Us: Breaking RC4 in WPA-TKIP and TLS，这种攻击方法的详细原理已经被整理到一个专门的网站：RC4 NOMORE 。

2.5. WPA/WPA2 PSK破解

回顾第二章 无线接入网监听中我们对WPA/WPA2 PSK的四次握手认证原理讲解，我们了解到：

PTK = Hash(PMK||A-nonce||S-nonce|| AP Mac || STA Mac)

= Hash(PBKDF(Passphrase, SSID, ssidLength, 4096, 256)||A-nonce||S-nonce||AP Mac ||STA Mac)

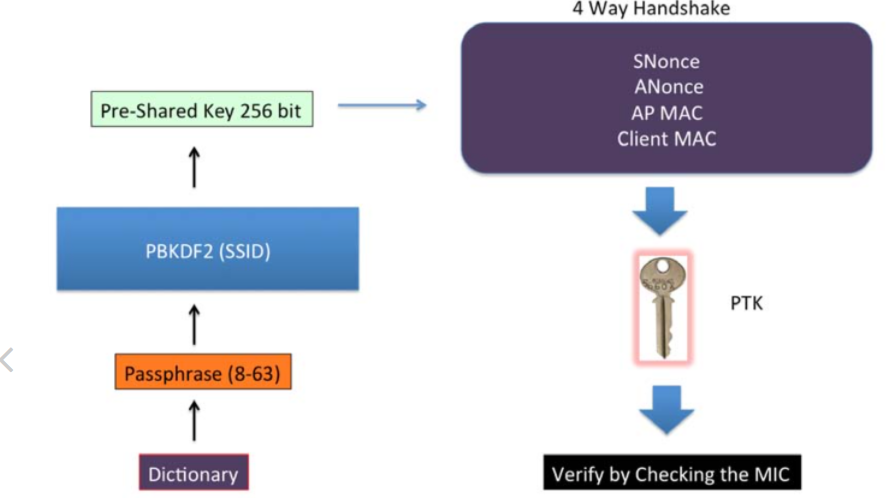
上述公式中，解密出Passphrase是我们的破解目标，剩下的变量均可以通过无线嗅探的方式获取到。其中：

A-nonce在4次握手消息中的第1个EAP包；

S-nonce在4次握手消息中的第2个EAP包；

剩下的变量在AP的beacon frame广播包、STA和AP之间的probe request、probe response、association request和association response中都可以提取到；

需要注意的是，我们在嗅探获得的数据包中并不能得到PTK，实际上我们能够提取到的是PTK的组成部分之一：校验和字段MIC值。如下图所示展示了这些变量（Passphrase、PTK和MIC）之间的“推导”关系：



对于WPA-个人模式（WPA和WPA2均是如此）来说， PSK 就是 PMK，而 PMK 和 Passphrase之间的关系就是如下公式：

PSK = PMK = PBKDF(Passphrase, SSID, ssidLength, 4096, 256)

由于网络身份认证通信过程中传输的是经由 Passphrase 通过了一系列随机值填充之后的单向散列算法生成随机的挑战-应答消息，所以首先排除了直接逆向运算解密出 Passphrase 的可能性。其次，采用穷举的方式在线破解无线网络的预共享密钥虽然存在理论上的可信性，但实践中，我们会采取一种相对更“优雅”的离线破解方案，降低被AP检测到破解攻击的风险。具体来说这个破解过程如下描述：

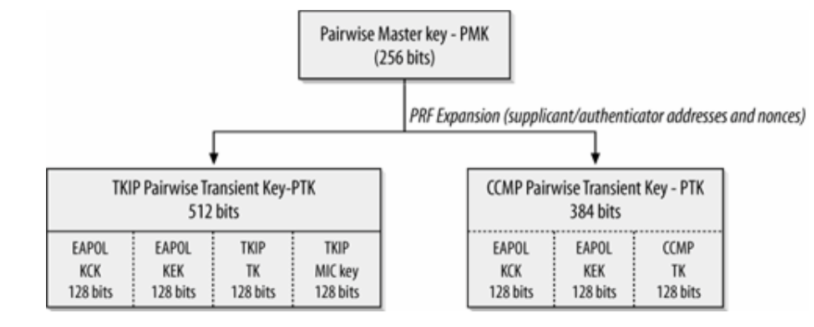
使用嗅探模式（避免channel hopping，采用固定信道方式）监听目标AP的通信数据；

如果目标AP当前没有已连接的无线客户端，则需要静候有无线客户端连接AP时发送4次握手认证包；

如果目标AP当前有已连接的无线客户端，通过伪造de-authenticate数据帧的方式强制客户端下线，静候客户端重连目标AP时产生的4次握手认证包；

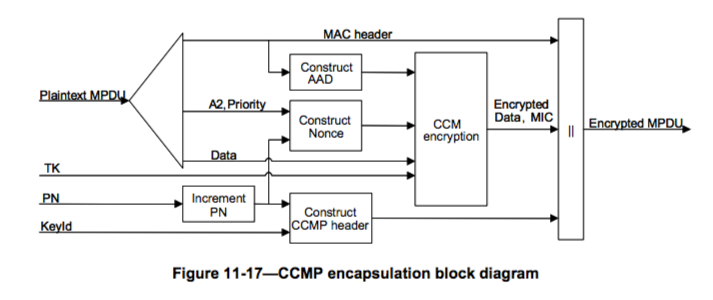
只要获取到4次握手认证包的前2个，即可开始离线破解。

如下图所示，是PMK与MIC之间的对应关系。

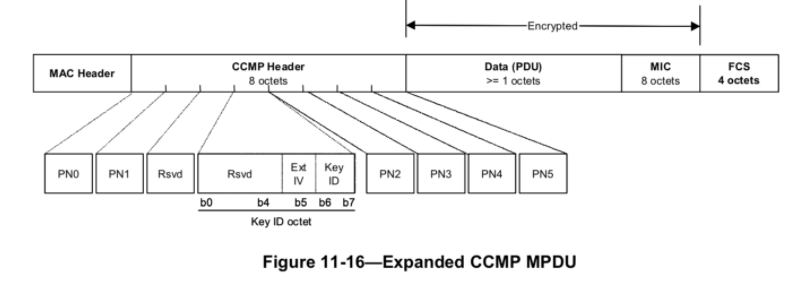


在WPA中，MIC是通过“Michael完整性校验算法”计算得到。如下图所示：

在WPA2的默认加密算法CCMP使用了更安全的 CBC-MAC 模式AES算法从明文MPDU (MAC Protocol Data Unit) 计算出MIC，CCMP加密数据封装（计算出的MIC追加在加密输出结果的尾部）过程如下图所示：



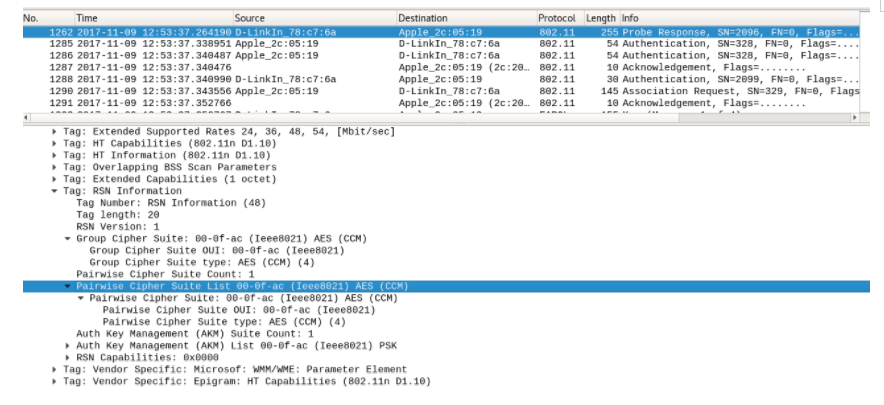
加密封装扩展后的CCMP MPDU结构如下图所示：



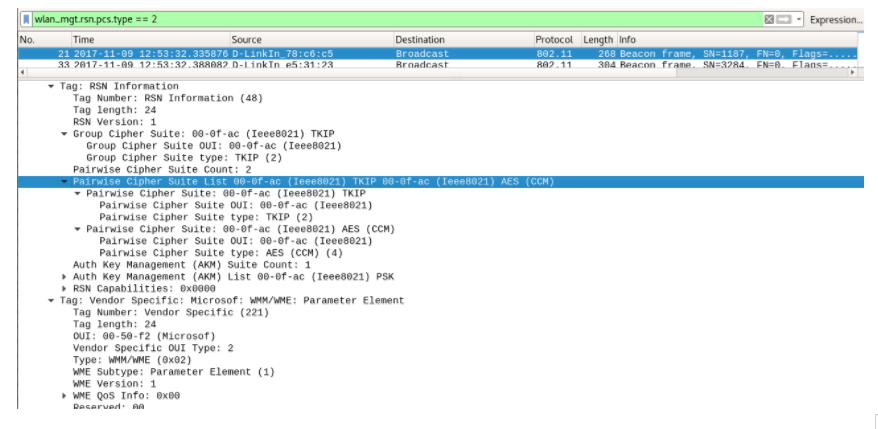
CCMP中使用的MIC计算方法改进了“Michael完整性校验算法”的2个已知缺陷：可逆恢复MIC密钥（根据任一已知消息和MIC值，能够得到MIC的密钥）和相关消息攻击导致MIC密钥在线性时间内可以恢复出来。不过，对于WPA/WPA2 PSK的离线暴力破解来说，我们只用到了针对单向散列算法的通用破解思路：散列值穷举匹配。

固定散列值算法中除“目标输入变量值”之外的所有变量值，穷举“目标输入变量值”的散列值和“已知散列值”进行匹配。如果找到匹配，则说明找到了“目标输入变量值”。

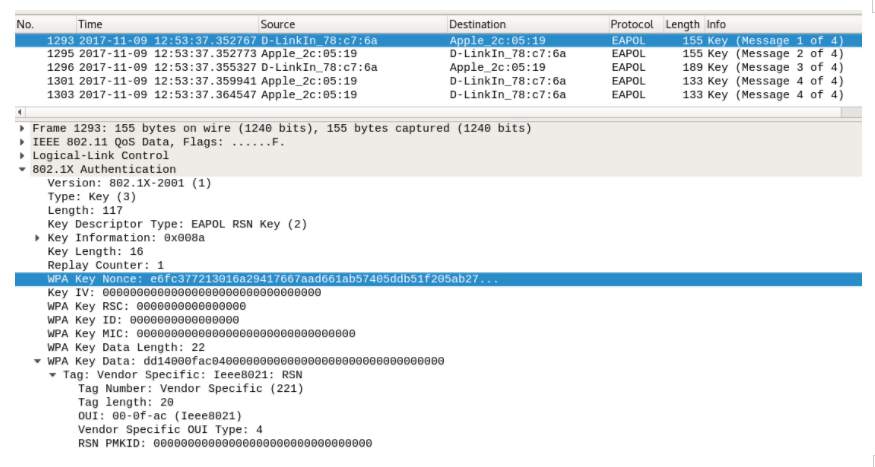
下图是一个仅支持WPA2 CCMP的probe response请求的抓包结果截图：



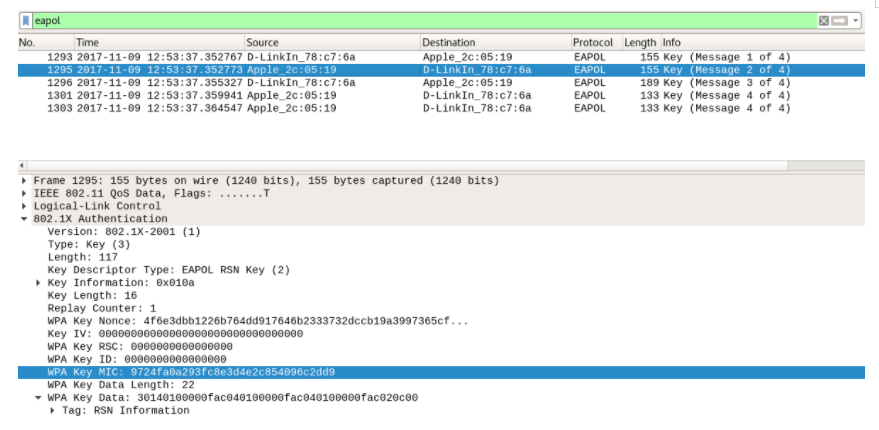
下图是一个同时支持TKIP和CCMP的beacon frame广播的抓包结果截图：



4次握手认证包的第1个包如下图所示：



4次握手认证包的第2个包如下图所示：



上述2张截图中，我们确实可以看到：A-nonce、S-nonce和MIC。

除了直接嗅探客户端和目标AP的握手认证过程来破解预共享密钥之外，由于无线客户端的“自动发现和自动重连”网络特性，我们还可以通过Evil Twin攻击方式，让一个“离线”的客户端自动连入我们伪造的AP。我们不需要知道该AP使用的预共享密钥，只需要诱骗客户端主动和我们伪造的AP完成4次握手认证中的前2个数据包交换过程即可。

如下图所示，就是一个典型的“离线”客户端（not associated）的常规网络探测行为被我们嗅探到的情况。我们通过“离线”客户端定时发送的probe request可以了解到该客户端连接过哪些网络（图中显示出该客户端一定连接过2个名称分别为a101e-lab和cuc的网络）。这样，Evil Twin攻击就可以有针对性的去设置ESSID。当Evil Twin AP运行起来之后，就会主动响应“离线”客户端的探测请求并自动继续完成接下来的网络连接和身份认证交互过程了。

一旦拿到包含4次握手认证包的前2个包的抓包文件之后，剩下的破解工作就可以交给aircrack-ng来自动化完成了。如下图所示是一次成功的口令破解结果截图：



2.5.1. 破解加速

基本思想1：空间换时间，针对常见SSID预先计算好PMK

可以用到的工具：

genpmk - WPA-PSK precomputation attack

基本思想2：使用GPU代替CPU计算

可以用到的工具：

pyrit - A GPGPU-driven WPA/WPA2-PSK key cracker

hashcat - World's fastest password cracker

基本思想3: 并行/分布式计算

基本思想4: 众包和云计算

2.6. WPS已知脆弱性和漏洞利用⽅法

2.6.1. 静态PIN码离线预测攻击

针对Headless设备静态预分配PIN码的弱随机产⽣算法或静态确定性产⽣算法，攻击者可以直接根据目标AP的MAC地址计算出可以用于加入目标无线网络的WPS PIN码。

2.6.2. 动态PIN码暴⼒枚举攻击

动态PIN码在线暴力枚举攻击的基本原理是：利用内部注册协议分段离线暴⼒枚举破解

第⼀轮M1~M4: 在线暴⼒破解枚举PSK-1

第⼆轮M5~M7: 用正确的PSK-1，在线暴⼒枚举PSK-2

得到完整PSK，进⾏⼀次完整WPS内部注册过程

无线设备在与无线路由器连接时，系统自动生成了一个随机的8位个人识别号码（PIN码），并根据这个8位PIN码进行安全的WPA链接，而绕过了WPA密码验证环节。如果黑客想通过穷举法，破解这个8位PIN码与无线路由器进行连接，理论上需要试算10^8次即1亿次，按照每秒1次的速度，需要1157天。但这个8位PIN是有规律的，实际上是一组4位PIN+另一组3位PIN+最后的1位校验位组成。校验位有固定的算法，这样只需要试算10^4+10^3总共11000次就可以了。穷举法试算11000次，几个小时就可以出来结果。

如果攻击者在发送完M4消息后接收到一个EAP-NACK消息，则说明PIN码的前半部分是错误的，继续枚举测试直到完成10000次尝试。在几分钟内尝试50次攻击时，有些路由器可能会把攻击的网卡加入黑名单。但大多数路由器都不会这样做，即使加入黑名单了还可以稍后再做攻击。攻击者也可以不断变换自己的MAC地址，对抗MAC地址黑名单机制。

如果攻击者在发送完M6消息后接收到EAP-NACK消息，就说明枚举PIN码的第二部分是错误的，继续暴力尝试下一个PIN码。使用工具 reaver 可以实现在线暴力破解动态PIN码以及恢复目标网络的WPA-PSK密钥，如下2张截图所示。

上述在线暴力枚举攻击的一种改进算法被命名为 Pixie Dust Attack) ，其算法改进的基础是：E-Hash1、E-Hash2、PKE、PKR 都是可以直接通过抓包获得的，剩下的PSK1和PSK2分别对应PIN码前后两半，可被枚举。E-S1和E-S2是整个离线破解的关键，⼀旦这2个参数被计算出来，则对照公式可以离线遍历PSK-1和PSK-2的可能性验证计算出的E-Hash1是否与抓包得到的E-Hash1相同。

E-S1和E-S2在实际设备中的实现算法使用的是 伪随机数发⽣器 ，存在以下3个主要缺陷：

嵌⼊式设备⼤多采用32位CPU，状态空间不⾜，导致产生的随机数取值空间较小

伪随机算法可能被逆向

伪随机数种⼦状态可能会被预测和恢复

除了随机数生成算法的缺陷之外，部分设备厂商在生产设备时甚至对 伪随机数发生器 进一步偷工减料，例如：

Broadcom/eCos，E-S1 + E-S2 使用与N1相同的随机数发⽣器

Realtek，E-S1 = E-S2 = N1 或 使用秒为单位的UNIX时间戳格式整数作为随机数发⽣器种⼦

Ralink / MediaTek / Celeno， E-S1 = E-S2 = 0

改进后的 Pixie Dust Attack 实现了 离线 暴力破解WPS动态PIN码，对应的攻击算法实现工具 Pixiewps 可以针对特定存在上述 伪随机数算法 缺陷的无线路由器实现杪级至分钟级别的PIN码计算。甚至对特定无线路由器实现恢复出 WPA-PSK 明文密钥。

以下2张图是使用 Pixiewps 进行离线动态PIN码破解成功的截图。

2.7. 攻击 WPA/WPA2 企业级认证⽆线⽹络

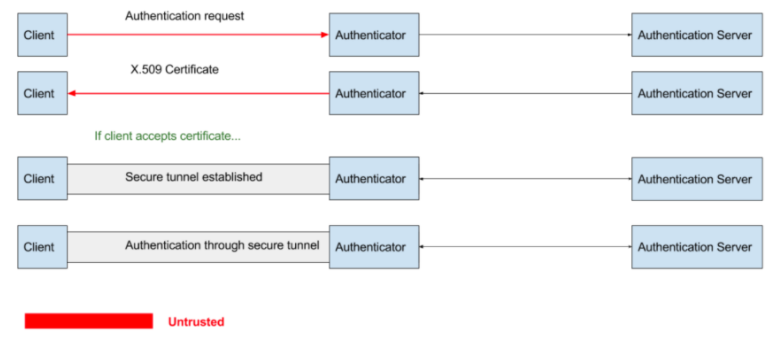
逻辑上来说，EAP认证过程发生在请求者（supplicant）和认证服务器（authentication server）之间，如下图所示：



物理上来说，AP扮演了认证过程的中间人：认证过程建立在安全隧道之上，但加密的SSL/TLS数据是在开放的无线网络上传输的。



在安全隧道建立之前，AP是一个开放访问的接入点，开放无线网络易遭受Evil Twin攻击，因为（客户端）没有办法验证AP的身份。

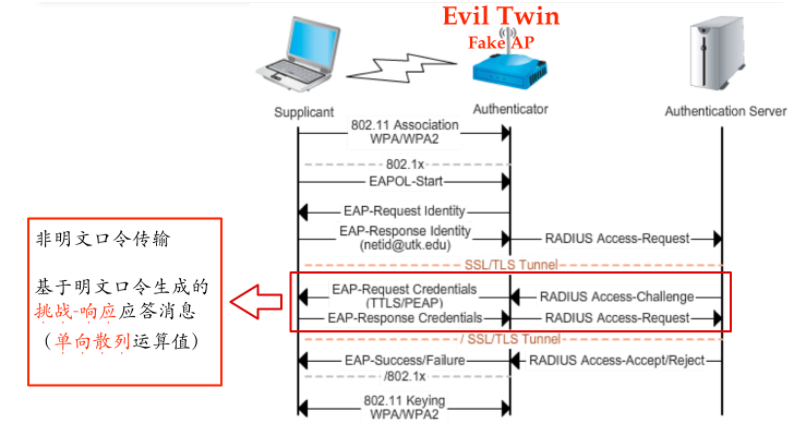


当用户连接无线网络过程中，在自己的手机上看到了如下的 信任警告 ，大多数用户应该不会查看 更多详细信息 ，即使查看了，在不借助工具的情况下，肉眼也无法鉴别该证书的合法性，所以最终的结果都是用户选择了 信任 这个 自签发 的证书，无线网络连接过程得以继续进行下去。

如下图所示，攻击者此时此刻通过 Evil Twin 构建的 假AP （包括自建的 假认证服务器），可以直接查看到无线客户端提交的认证凭据。认证凭据虽然不是明文口令传输，但对于单向散列值来说，可以使用 彩虹表 查找技术直接查出散列值对应的明文口令。

攻击者在 假AP 上看到的基于明文口令生成的挑战-响应应答消息（单向散列运算值）示例如下图所示；

把 单向散列运算值 导入到 asleep 中就可以很快恢复出明文口令。



**针对上述企业级无线网络认证方式的中间人攻击方法，企业无线网络的管理员应：**

禁用不安全的EAP实现方法，例如：EAP-MD5, EAP-OTP, EAP-GTC, LEAP

启用并 正确配置 安全的EAP实现方法，例如：PEAP, TTLS, EAP/TLS

在无线客户端上预置自签发的EAP认证证书或购买权威CA签发的用于身份认证的证书

无线客户端启用永远验证服务器证书有效性

教育无线网络用户不要信任任何被警告的证书

学会手工验证证书的CN字段值是否与公司通告一致（只是缓解风险，如果攻击者完全克隆企业的证书信息，则本方法无效）

**4. 构建安全的无线局域网**

和通用信息系统和网络的安全加固方法一样，构建安全的无线局域网也应遵守“层次化的安全加固策略”，即按照：人、应用层、网络层、链路层和物理层等这样的分层分析威胁和针对性安全加固。

4.1. 人

针对人的安全加固主要是通过安全意识教育和安全技能培训的方式来实施：

避免使用万能Wi-Fi钥匙类APP。作为一种披着“分享”经济外衣的工具类应用，信息相比较于实物，天然具有零复制成本和易于复制、快速传播、难以追溯传播源的特点。客人可以通过此类软件在未经主人知晓的情况下，将所连过的主人家中Wi-Fi的密码、SSID上传到服务器分享给其他人。其他路人后续使用这个共享密钥连入该主人的家庭网络时，主人同样面临着没有知情权和控制权的危险处境。如果说未经授权占用他人带宽事小，那么未授权的内部网络任意访问、窃取隐私和机密数据则有可能严重侵犯到网络所有者的私有数据所有权。

定期更换个人或家庭无线网络的共享密钥，一来缓解密钥被未知渠道、未知原因泄漏带来的“蹭网”风险，另一方面可以降低网络密钥被离线暴力破解成功的概率。

谨慎使用公共或陌生Wi-Fi，攻击者可以在这样的网络中使用局域网攻击技术实现数据嗅探和中间人攻击，为了避免敏感信息在传输过程中被非法嗅探，防止投毒攻击劫持正常通信连接进行钓鱼攻击和二进制文件篡改安装后门，应避免在这一类Wi-Fi中使用需要提交个人信息的应用和服务，避免使用明文方式下载和安装应用软件，避免打开在用网期间通过明文方式下载的文档和压缩包等二进制文件。

所有具备Wi-Fi功能的设备在不使用Wi-Fi功能时关闭无线开关（软开关或硬件开关），避免Evil Twin攻击套取到你连过的AP的EAPOL Packet用于离线破解WPA/WPA2 PSK密码，避免设备主动连入开放认证的恶意AP被监听和中间人劫持攻击。

4.2. 应用层

无线路由器的管理员应修改路由器默认设置，执行以下安全加固措施

修改默认的管理员用户名和密码，避免被弱口令登录；

启用登录管理界面的图形化验证码，提高身份认证暴力破解的难度；

更新到最新版固件，避免针对路由器系统的直接远程攻击。

4.3. 网络层

同样是对无线路由器的默认设置进行修改，启用客人/访客网络

仅提供互联网访问，禁止访问有线局域网；

独立网络使用独立密码；

启用AP隔离功能，可以禁止无线网络中的客户端相互直接访问，杜绝局域网内的攻击。

对于企业无线网络的管理员来说，进行合理的子网划分与隔离，按业务需求、安全等级设置无线局域网、有线局域网和互联网之间的访问控制机制。

4.4. 链路层

依然是对无线路由器的默认设置进行修改

仅使用WPA2-PSK（针对个人用户），禁用TKIP这样的缺陷加密算法；

使用强健密码，例如大小写字母、数字、特殊字符组合；

禁用WPS功能；

避免使用常见SSID名，例如：dlink、NetGear等。

对于企业无线网络的管理员来说，启用WPA-企业级身份认证

实名制、独立账号接入，在员工离职时可以及时删除账号杜绝非授权访问，对于每个员工的用网行为进行独立审计，可以精确定位是哪个用户泄漏了自己的上网账号和密码；

有IT技术能力的企业强烈建议配置EAP-TLS，实现 双向证书验证 。

4.5. 物理层

个人用户根据信号覆盖范围需求，合理设置无线路由器的信号发射功率。企业无线网络的管理员可以通过以下措施限制无线信号的覆盖范围：

缩窄发射天线覆盖范围

墙面信号反射涂料

使用定向天线

第四章 移动通信安全概述

1. 2G / 2.5G / 3G / 4G / 5G

2. 物联⽹安全

Wi-Fi 联盟制造商 的商标可做为产品的品牌认证，最基础的认证条件是符合 IEEE 802.11 标准，此外还需缴纳认证授权费用。WLAN（无线局域网）不仅可以使用Wi-Fi设备来组⽹，蓝牙、ZigBee等技术都可以用于构建⼀个⽆线局域⽹，Wi-Fi是符合IEEE 802.11标准的一种 WLAN 产品认证。如下图所示就是 Wi-Fi 认证标志：



物联⽹（Internet of Things，IoT）是互联⽹、传统电信⽹等信息承载体，让所有能⾏使独立功能的普通物体实现 互联互通 的⽹络。Machine to Machine 是狭义物联网的一大特征，核心是依赖于“通信技术”。广义的物联网中不仅是机器到机器的通信，也包括机器到人（M2P）、人到人（P2P）、人到机器（P2M）之间广泛的通信和信息交流。物联网中的机器定义为可以获取信息的各种终端。由于“万物互联”，物联网又被称为是一种 泛在（Ubiquitous）网络，即具备 4A (Anytime, Anywhere, Anyone, Anything) 级别通信能力。物联网对网络中各种物品的 全面感知 依托 4A 级别通信能力成为可能。物联网中各种不同的传感器如同人体的各种器官，对外界环境进行感知。物联网通过射频识别、传感器、二维码等感知设备对物品各种信息进行感知获取。物联网正是通过遍布在各个角落和物品上形形色色的传感器，以及由它们组成的无线传感器网络来最终感知整个物质世界。

物联网的基础是实现网络融合，现有的互联网、电信网（包括移动通信系统）、广播电视网络首先要融合成一个统一的“大网络”，即我们熟知的“三网融合”。物联网继承了IP⽹络基本组⽹和通信模型，具体表现在：通信架构方面依然遵循分层结构，通信模式方面依然支持点对点、组播、广播和任播这几种模式。支撑物联网底层联网的关键技术主要包括：RFID (物联⽹的基础设施技术之⼀，可以通过蓝牙、NFC技术来实现）、蓝牙 (IEEE 802.15.1)、NFC (ISO 13157等)和ZigBee (IEEE 802.15.4)。如下图所示，是一个典型的物联网系统 分层 架构图。



除此之外，电力行业也是物联网系统发展的先驱，智能抄表系统、智能配电系统都是物联网行业的典型应用。物联网的重点行业应用还包括城市综合管理系统、安防系统、智能物流系统、智能家居以及智能医疗等。“智能”化将物联网推向一个全新的高度，目前的2大智能化特征表现在：自动化（Autonomous）和使用了人工智能（Artificial Intelligence, AI）技术。传感器网络为“智能”处理提供了丰富的原始感知数据，对于用户来说这些原始数据只有经过转换、删除、分析处理之后才有实际价值。智能处理相当于人的大脑，能够根据神经系统传递来的各种信号做出决策，指导相应器官进行活动。物联网系统中的智能处理部分将收集来的数据进行处理运算，做出决策来指导系统进行相应的改变，是物联网应用实施的核心。

2.1. RFID

RFID的全称是Radio Frequency Identification，翻译过来是“射频标识”，属于⽆线通信技术范畴。通过⽆线电信号识别特定目标并读写相关数据，⽆需识别系统与特定目标之间建立机械或光学接触。

⽆线电的信号是通过调成⽆线电频率的电磁场，把数据从附着在物品上的标签上传送出去，以自动辨识与追踪该物品。某些标签在识别时从识别器发出的电磁场中就可以得到能量，并不需要电池，这一类标签被称为⽆源RFID。⽆源RFID (Passive RFID，被动RFID) 依靠和阅读器的电磁耦合供能，读取距离取决于：阅读器耦合线圈的尺⼨、⼯作频率和阅读器的功率（0.5W：0.7m 、 4W：2m 、30W：5.5m）。成本低,应用⼴泛。也有标签本身拥有电源，并可以主动发出⽆线电波（调成⽆线电频率的电磁场），这一类标签被称为有源RFID。标签包含了电⼦存储的信息，数米之内都可以识别。与条形码不同的是，射频标签不需要处在识别器视线之内，也可以嵌⼊被追踪物体之内。使用锂电池通常可⼯作3~10年，读取距离10米 ~ 30米，或更远。目前的应用相对⽆源ID要少，典型如：需要远距离识别的场合、⼿机钱包和⾼速公路ETC。

RFID的技术发展呈现着以下发展趋势特点：

越来越多的RFID应用，原本只是以条形码的替代者面目出现；

飞快的发展速度；

小型化，低成本化；

协议和标准泛滥，目前共有117个不同的协议，各国使用不同的标准不同的频段。

目前主要RFID使用的频段现状如下表格所示：

目前RFID的主要应用领域涵盖了：钞票及产品防伪技术、身份证、通⾏证（包括门票）、电⼦收费系统，如城市公交卡、家畜或野⽣动物识别、病⼈识别及电⼦病历、物流管理、⾏李分类和门禁系统等。

RFID 标签按照是否可写，分为：

可读写卡(RW)，Read Write，相当于CDRW。可读可写标签内部的存储器，除了ROM、缓冲存储器之外，还有非活动可编程记忆存储器。这种存储器一般是EEPROM（电可擦除可编程只读存储器），它除了存储数据功能外，还具有在适当的条件下允许多次对原有数据的擦除以及重新写入数据的功能。可读可写标签还可能有随机存取器（Random Access Memory，RAM），用于存储标签反应和数据传输过程中临时产生的数据。可读写标签一般存储的数据比较大，这种标签一般都是用户可编程的，标签中除了存储标识码外，还存储有大量的被标识项目其他的相关信息，如生产信息、防伪校验码等。在实际应用中，关于被标识项目的所有信息都是存储在标签中的，读标签就可以得到关于被标识目标的大部分信息，而不必连接到数据库进行信息读取。另外在读标签的过程中，可以根据特定的应用目的控制数据的读出，实现在不同情况下读出的数据部分不同。典型应用产品形态：第⼆代身份证；

⼀次写⼊卡(WORM)，Write Once, Read Many, 相当于CDR。如航空行李标签、特殊身份证件标签等。RW卡一般比WORM卡和RO卡价格高得多，如电话卡、信用卡等：WORM卡是用户可以一次性写入的卡，写入后数据不能改变，比RW卡要便宜。RO卡存有一个唯一的ID号码，不能修改，具有较高的安全性；

只读卡(RO)，Read Only, 相当于CD。只读标签内部只有只读存储器（Read Only Memory，ROM）。ROM中存储有标签的标识信息。这些信息可以在标签制造过程中，由制造商写入ROM中，电子标签在出厂时，即已将完整的标签信息写入标签。这种情况下，应用过程中，电子标签一般具有只读功能。也可以在标签开始使用时由使用者根据特定的应用目的写入特殊的编码信息。只读标签一般容量较小，可以用做标识标签。对于标识标签来说，一个数字或者多个数字字母字符串存储在标签中，这个储存内容是进入信息管理系统中数据库的钥匙（Key）。标识标签中存储的只是标识号码，用于对特定的标识项目，如人、物、地点进行标识，关于被标识项目的详细、特定的信息，只能在与系统相连接的数据库中进行查找。典型应用场景是：门禁。

RFID面临的主要安全风险如下：

伪造、假冒和非法篡改，详见本节接下来的具体案例说明。

泄露隐私。由于RFID标签无须直接与读卡器接触，使用者会在不知情的情况下被他人读取标签内存储的信息，面临安全风险。试想一下，如果每一件商品都内置了RFID标签，比如你的口红，通过合适的RFID阅读设备就可以读取和识别，那么通过追踪一个私人物品的位置移动就可以实现对一个人的跟踪。

植⼊⼈体？技术上已经成熟，美国的食品和药品监督管理局（Food and Drug Administration, FDA）已经于2004年通过了相关法律：CPG Sec. 400.210, Radiofrequency Identification Feasibility Studies and Pilot Programs for Drugs 允许RFID芯片植入人体。这将给那些“生物科技黑客”更多攻击人类的途径，这就不仅是泄漏隐私的问题了，对于具备双向通信和控制能力的RIFD芯片来说，这完全可以成为一颗埋在人体内的“定时炸弹”。

除了上述RFID标签之类的RFID应用之外，在我们的生活中还有一些常见的无线产品也属于RFID技术应用。例如：无线键盘和无线鼠标。最早期的无线键盘鼠标依靠红外线作为信号传输的载体，唯一的访问控制就是红外设备的发射和接收距离限制。历史上无线键鼠还使用过 27 MHz 射频技术，由于其他无线设备很少使用 27 MHz 频率，因此受干扰的可能性不大。27 MHz 射频的特点是技术成熟、成本较低，但劣势也比较明显：电力消耗较大、只支持单向传输（可能导致传出传入的频率互相干扰）、传输距离短（仅6英尺（182.88cm））、安全级别不高。目前主流的无线键盘和鼠标使用蓝牙技术传输控制信号，蓝牙的抗干扰能力、传输速率和安全性都优于以往的无线键鼠使用的无线通信技术，而且具备和其他蓝牙产品互联的能力。

在探讨IEEE 802.11构建的无线局域网的安全加固时我们可以在物理层限制无线信号的发射和覆盖范围来避免不必要的“信号泄漏”，对于物联网来说，电磁辐射泄漏无处不在，例如经典的CRT显示器行场信息还原，普通键盘和鼠标的电磁泄漏问题等。最近，国外的安全研究人员公布了一种通过软件定义无线电（SDR）的方式实现还原远程显示器上的画面的方法和工具 TempestSDR ，利用它可以捕获计算机屏幕上的射频信号，进而将屏幕显示的实时图像复原出来；如果外加一个增强信号的天线，甚至可以在几米之外成功窥探到电脑屏幕的内容。

2.2. 智能卡

智能卡的“智能”体现在：

内置“存储器”，对存储的数据可以进行访问控制，阻止未授权访问；

内置“微处理器”和RAM，支持密码学计算和可编程计算等；

支持组件式架构，例如可以按需搭载指纹识别、OTP和传感器模块。

智能卡的具体分类定义取决于： 1) 卡片数据的读取和写入方式；2) 卡片内置的芯片类型和支持的功能。

按照卡片与读卡器之间的通信方式是否需要物理接触分为：接触式卡片和非接触式卡片，大部分的RFID卡片都是非接触s式的。除此之外，还有一类多组件卡片，例如：指纹卡、OTP显示屏卡、密保卡等。

USB令牌就是一类典型的的智能卡实现载体产品形态，除了接口和外观形状不同，其他物理和软件技术架构和智能卡无差异。接口使用USB，无需专用的“读卡设备”。

智能卡的主要应用场景如下：

信息存储卡，通常用于保存个人（隐私）信息，例如个人医疗记录卡；

储值卡，代替小额现金支付场景，无需在线联网校验，直接读写卡内余额（次数）；自动贩卖机卡、预付费一次性电话卡、公交卡；

认证令牌卡，内置加密芯片，提供散列值、数字签名和加解密能力；

多功能卡，内置操作系统。Windows for Smart Cards, MULTOS, Java Card。支持灵活丰富的应用场景。

智能卡面临的主要威胁如下：

差分功耗分析（Differential Power Analysis），基于密码学计算过程中的功耗变化统计数据来推导卡片上存储的密钥；

计时攻击（Timing Attacks），类似的攻击手段我们已在“SQL盲注攻击”中见识过；

芯片逆向（Reverse Engineering of the Chips），专家/富人/国家级研发能力和成本投入；

设计/实现缺陷，通用/缺省加密口令/密钥。

使用分层次威胁建模的方法，自下而上按照：环境、硬件、网络、操作系统、应用软件和人这几个层次来逐层分析智能卡面临的威胁：

环境：钓鱼风险；

硬件：直接读取或破坏（改变电压/温度/酸碱度/电路板重新焊接搭线等）EEPROM；

网络：明文传输风险，MITM，嗅探器调试；

操作系统：需要分 固定文件系统 和 动态文件系统 具体讨论；

应用软件：软件设计与实现漏洞；

人：遗失、外借、设置弱PIN码等。

下面重点探讨智能卡操作系统安全，智能卡操作系统主要分为 1) 固定文件系统 和 2) 动态文件系统。正如所有智能卡类型可以按照应用场景分类，操作系统选型也是取决于应用场景。除此之外，不同智能卡的差异还体现在操作系统的加密能力和选用的芯片。

1) 固定⽂件系统 通常只被用于可信安全计算环境，所有⽂件权限都是出厂时设置好⽆法修改的，例如员⼯信息卡；

2) 动态⽂件系统 常见于JavaCard®和MULTOS，操作系统和应用软件实现了解耦。例如，GSM的SIM卡，支持OTA更新，适用于频繁需要更新数据/软件的场景，例如密钥协商。

MF (Master File)

|

-------------+--------------

| | |

EF (Elementary File) EF DF (Dedicated File)

|

---------+--------

| | |

EF DF EF

|

EF

如上图所示是一个智能卡操作系统上的数据组织方式示意图，其中 MF (Master File) 和 DF (Dedicated File) 相当于“目录”，但DF依然可以独立存储数据。MF相当于根目录，包含了EF和DF的文件头信息。EF (Elementary File) 相当于“⽂件”，MF/DF/EF的⽂件头包含安全属性（相当于访问控制信息）：只要有“权限”，操作系统上的应用就可以遍历所有⽂件。智能卡的 PIN 也会被存储在 EF 文件头但访问必须获得卡片上操作系统授权。卡片上的数据访问遵循访问控制规则，以SIM卡为例，智能卡文件系统包含5类安全属性（权限）：

Always(ALW)，这一类权限意味着数据访问不受限制；

Card holder verification 1 (CHV1)，使用 PIN1 验证（用户设置的）通过或禁用了PIN验证功能；

Card holder verification 2 (CHV2)，使用 PIN2 验证，设备商设置的用于解封设置（ PIN1 和 PIN2 是相互独立的PIN）或禁用了PIN2验证功能；

Administrative (ADM)，卡片签发者在满足特定管理访问机制实现的前提下才可以访问；

Never (NEV)，通过SIM/ME接口的访问是被永久禁止的。

SIM卡中通常会存储订阅者（手机卡持有者）的通讯录和短信消息



3. “智能”硬件

智能硬件没有一个公认的统一定义，本书接下来要介绍的“智能”硬件其“智能”主要体现在⾄少具备以下特性之一：

具备通用或专用计算能⼒

具有可编程性，支持软件定义硬件

具备感知外界的能⼒

低功耗或节能型

符合上述特性的代表性产品包括：

DIY硬件

机器⼈、⽆⼈机

智能家电，例如：⾖浆机，热⽔器，空调，净⽔器，电源开关，电源插座，电视，摄像机等等

智能家居

可穿戴设备（健康医疗、VR/AR等等）

4. 蓝牙安全

蓝牙是构建无线个人网络（WPAN，Wireless Personal Area Network）的基础技术之一，支持服务能⼒描述/声明配置⽂件（声明蓝牙设备所具备的应用能⼒，例如：输⼊（键盘、鼠标）、输出（音频、⽂件传送、打印机）和设备能⼒发现。越来越多的物联⽹应用采用蓝牙（特别是蓝牙低功耗）作为底层通信协议，例如各种智能锁、智能家居产品等。

蓝牙使用的是2.4 GHz ISM (Industrial Scientific Medical Band) 频段的 2400~2483.5 MHz，根据物理层传输方式的差异分为标准蓝牙（BR/EDR和AMP）和低功耗蓝牙（BLE），2种物理层传输方式相互不兼容。以下如果没有特别说明是蓝牙低功耗或BLE，则默认均指的是使用标准蓝牙的物理层传输协议。其中，标准蓝牙的传输速率有2种，分别是721.2Kbps的基础速率（Basic Rate, BR）和可选的2.1Mbps增强数据速率（Enhanced Data Rate, EDR）。AMP (Alternate MAC and PHY layer extension) 是另一种标准蓝牙传输速率的物理层协议规范，“借用”了IEEE 802.11协议的物理层和MAC层规范，传输速率可以达到54 Mbps。

使用了AMP的标准蓝牙和IEEE 802.11协议的区别与联系可以用如下表格进行总结：



4.1. BR/EDR

标准蓝牙和低功耗蓝牙使用不同的物理层传输协议规范，具体来说，BR/EDR控制器把可用的频带分成79个频率，每个为1 MHz，使用跳频(Frequency Hopping)通信，1秒⾄多改变1600次；物理信道（Physical Channel）是由跳频图谱（Hopping Pattern）决定的，跳频图谱是⼀种规则，规定了蓝牙RF在某⼀时刻需要跳到79个频率中的哪个频率上通信。BR/EDR控制器把所有物理信道分为三类：搜索用的信道（Inquiry scan channel)、连接用的信道（Page scan channel)和连接后设备之间通信的信道。连接建立后，多个设备共享同⼀个物理信道，决定跳频图的那个设备称作Master，其他设备称作Slave。Master只有⼀个，Slave可以有多个。连接建立后，双⽅根据已建立的、基于BR/EDR控制器的L2CAP(Logical Link Control and Adaptation Protocol)，可以协商各自是否具备可以使用的AMP信道，如果具备，是否愿意将后续的数据输出转移到这些控制器上（可以提⾼蓝牙传输速率） 。

蓝牙设备之间进⾏数据通信前需要先 找到对⽅ 。具体来说，蓝牙设备在寻找对方的过程会被分别设置为搜索和可被搜索状态。处于搜索状态的⼀⽅以较快的速度跳频，被搜索设备以较慢的速度跳频并⼴播自身信息。

蓝牙设备之间进⾏数据通信前还需要 和对⽅建立连接 ，一方面蓝牙设备可以被设置为连接和可被连接状态，连接是两个蓝牙设备同步到同⼀个跳频图谱（即物理信道）的过程，发起连接的⼀⽅作为Master，接受连接的⼀⽅为Slave。

BR/EDR Controller的链路层控制体现在同⼀个物理信道上，只有Master和Slave之间可以通信，Slave和Slave之间不能通信。多个设备共享⼀个物理信道，Master和不同Slave之间可以时分复用信道，看起来就像独占了⼀个信道，称为：物理链路(Physical Link)。在物理链路之上抽象出逻辑链路，用于传输不同类型的数据，如异步数据、同步数据、单向(unidirectional)数据、⼴播数据等。逻辑链路时分复用物理链路。链路管理协议(Link Management Protocol, LMP)用于管理抽象出来的逻辑链路。LMP之上提供L2CAP层，从应用的角度复用逻辑链路。负责不同应用类型数据的封包和解包，类似TCP/IP里的端⼝。

4.2. BLE

低功耗蓝牙的别名是 Bluetooth Smart，是经典蓝牙协议的一个轻量级子集（另一个是Basic Rate），但不能与经典蓝牙协议互通。BLE是蓝牙4.0核心规范的组成部分，是面向现代移动终端平台设计，各个主流操作系统对蓝牙4.0的内置协议栈支持情况如下：

iOS5+ (iOS7+ preferred)

Android 4.3+ (numerous bug fixes in 4.4+)

Apple OS X 10.6+

Windows 8 (XP, Vista and 7 only support Bluetooth 2.1)

GNU/Linux Vanilla BlueZ 4.93+

BLE 控制器把可用的频带分成40个频率，每个为2 MHz。其中3个频率为⼴播信道（Advertising Channel），37个频率为数据信道（Data Channel）。

BLE的使用场景包括了：广播通信和连接建立。其中，广播通信是单一方向的、无连接的数据通信，数据发送者在广播信道上广播数据，数据接收者扫描、接收数据。BLE 控制器把⼴播信道为预先设定的物理信道，其个数和占用的频率，都可以设定，这些信道被划为时间单元，在这些时间单元上传输的数据包也称作事件(event)，有⼴播和连接两种事件。37个数据信道用于连接建立后的数据通信，采用跳频技术。BLE 控制器的广播通信过程中，在⼴播信道发送⼴播数据的⼀⽅称作⼴播者(Advertiser)，接收⼴播数据但不打算连接的⼀⽅称作扫描者(Scanner)。该特性主要用于那些没必要建立点对点连接的数据通信，如多播或⼴播。Advertiser发送的⼴播数据中有⼀种称为“可连接⼴播数据包”，表示它可以被其他设备连接。接收到该包的设备可以在⼴播信道上回应“连接请求”，这样可以建立起来点对点连接，同时该设备被称作发起者(Initiator)。类似BR/EDR技术，Initiator为Master，Advertiser为Slave。

GATT - Generic Attribute Profile 定义了2个BLE设备数据交换的方式，通过2种属性分类：服务（Service）和特性（Characteristic）来分类组织。Profile是由蓝牙标准委员会 或周边设备生产商预定义的一个服务集合，由蓝牙标准委员会官方采纳的服务（Service） 采用16-bit UUID进行标识，最多允许包含3个特性，私有自定义服务采用 128-bit 唯一标识。由蓝牙标准委员会官方采纳的特性 采用16-bit UUID进行标识，私有自定义特性采用128-bit唯一标识，一个特性字段存一个独立数据（可以是复合数据结构）。

Attribute Profile (ATT) 提供存储服务、特性和相关数据于一个简单的查找表，使用16-bit 唯一编号在表中查找匹配。一个BLE周边设备一次只能连接一个中心设备，连接一旦建立，广播通信立刻停止，其他设备就无法发现它所以无法再建立新连接了。

已建立BLE连接时，一个中心设备可以被多个周边设备连接，连接一旦建立就是一个双向通信的信道。

4.3. 蓝牙 4.0 安全机制

蓝牙 4.0 安全机制包括了配对(Pairing)，秘钥产⽣(Key Generation)、加密（主要是应用层（GATT）加密）和数字签名等。其中，配对是⼀次性过程，又分为三种配对方式：JustWorks(R), 如果设备没有显示屏；6位PIN，如果设备有显示屏；带外(Out of band)。

4.4. 蓝牙安全概述

蓝牙协议本身的安全问题 主要包括：劫持配对过程,窃听、伪造蓝牙通信，重放和中间人攻击，隐私问题：标识追踪。

蓝牙协议栈实现的安全问题包括：无线网络绑定的是硬件层和协议层，配对验证码PIN是默认值或弱PIN码和蓝牙直接绑定应用相对复杂（典型漏洞：针对蓝牙协议1.1版本的BlueSnarf 攻击和特定蓝牙协议栈实现的缓冲区溢出）。

2017年9月，Armis 爆出了一系列蓝牙协议栈实现的漏洞，基本上影响所有的蓝牙设备，相关受影响的系统和CVE漏洞编号情况统计如下。只要手机开启了蓝牙，就可能被远程控制。

CVE-2017-1000250 Linux bluetoothd进程信息泄露

CVE-2017-1000251 Linux 内核栈溢出

CVE-2017-8628 微软蓝牙协议栈欺骗漏洞

CVE-2017-0785 Android com.android.bluetooth进程信息泄露

CVE-2017-0781 Android com.android.bluetooth进程堆溢出

CVE-2017-0782 Android com.android.bluetooth进程堆溢出

CVE-2017-14315 LEAP (Low Energy Audio Protocol)协议堆溢出漏洞导致的远程代码执行漏洞

上述相关漏洞被发现者 Armis 命名为 BlueBorne 。BlueBorne之所以危险，是因为大多数用户都会在他们不使用蓝牙的时候将蓝牙开关打开。而攻击者根本不需要与目标设备配对(但是信号得在接收范围内)即可完全接管该设备。Armis Labs团队的负责人Ben Seri称，他们已经在实验环境下建立了一个僵尸网络，并且使用BlueBorne攻击安装了勒索软件。

与传统网络攻击不同，BlueBorne不需要用户点击URL链接，或者下载恶意文件，受害者甚至根本不需要连接到互联网上，它可以使用蓝牙协议在短距离内在空气中传播，而黑客根本就不需要与目标受害者配对，只要目标的蓝牙开关处于打开状态，黑客就可连接到这台设备，完全接管设备，还可以通过被攻陷的设备传播恶意软件，而受害者完全无法察觉。

Armis Lab在公开漏洞信息的同时也已第一时间在Google Play上发布了BlueBorne漏洞检测app ，不仅可用来检测你的设备是否受BlueBorne攻击影响，还可检测你周围的蓝牙设备是否也受影响（检测系统的补丁日期）。

上述BlueBorne漏洞相关的安全事件带给我们2点启示：

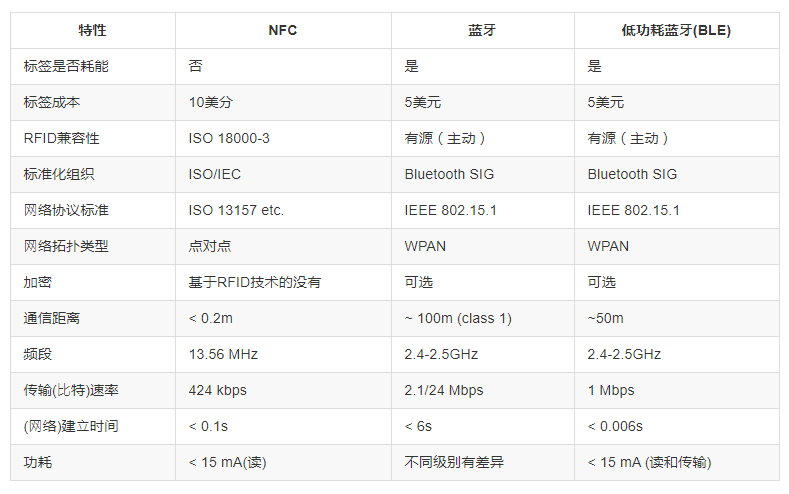
移动互联时代，官方应用商店提供的app同样可能存在仿冒和钓鱼应用；

物联网时代，物联网设备和协议漏洞将会成为安全事件的新常态。

4.5. NFC

NFC是一种短距离⾼频⽆线通信技术，由RFID演变⽽来。NFC仅限13.56MHz⾼频段，RFID有较多频段选择。NFC的有效通信距离⼤多在10厘米以内，RFID的通信距离范围从⼏厘米到⼏⼗米都有。NFC是⼀种“集成”RFID技术，单芯片内置非接触读卡器、非接触卡和点对点功能，RFID通常使用独立的阅读器和标签。RFID多用于⽣产、物流、资产管理等，NFC则更多用于公交、门禁、⼿机支付等。

以下表格总结了NFC与标准蓝牙、BLE之间的关系：



5. ⼴义⽆线⽹络安全

在研究广义无线网络安全之前，让我们先来熟悉一下《中华人民共和国无线管理条例》

第十四条 使用无线电频率应当取得许可，但下列频率除外：

（一）业余无线电台、公众对讲机、制式无线电台使用的频率；

（二）国际安全与遇险系统，用于航空、水上移动业务和无线电导航业务的国际固定频率；

（三）国家无线电管理机构规定的微功率短距离无线电发射设备使用的频率。

第十五条 取得无线电频率使用许可，应当符合下列条件：

（一）所申请的无线电频率符合无线电频率划分和使用规定，有明确具体的用途；

（二）使用无线电频率的技术方案可行；

（三）有相应的专业技术人员；

（四）对依法使用的其他无线电频率不会产生有害干扰。

知法守法，对于信息安全专业的学习者来说必须时刻谨记。攻防是信息安全的一体两面，未知攻焉知防 不等于 攻击有理 和 攻击无罪 ，网络攻击技术的学习和研究一定要确保在受控、隔离环境中进行，扩展一下上述《无线管理条例》的规定，我们应 对依法使用的其他网络与系统不会产生有害干扰 。

软件定义无线电（Software Defined Radio, SDR）是基于 通用的硬件平台 上用 软件 来实现各种通信模块，囿于本书篇幅我们不会具体介绍和讲解基于SDR的攻防技术，推荐阅读 《HackRf与GnuRadio⼊门指南》 入门基于SDR的广义无线攻防技术。

和前述无线网络的攻防实验和研究一样，基于SDR的广义无线攻防也是需要借助于专门的硬件来运行软件模拟和实现的各种通信模块。以下简单介绍几种目前常见的平价硬件解决方案：

RTL2832U 是一款廉价DVB-T调制解调电视棒常使用的芯片型号，RTL-SDR是基于该款硬件的SDR软件。

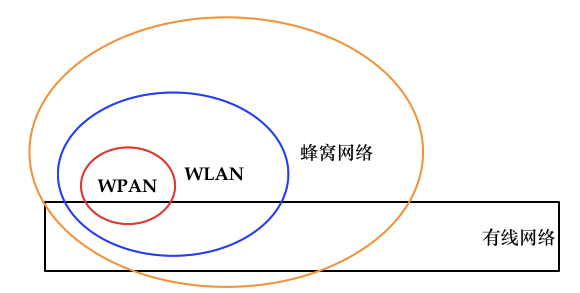
HackRF 是 greatscottgadgets.com 开发的一款低成本开源SDR平台，支持从 1 MHz 到 6 GHz 频率的无线电信号发送和接收。该团队还开发了知名的蓝牙和BLE嗅探平台Ubertooth 。

BladeRF 是一个软件无线电开放教学实验平台，支持 300 MHz ~ 3.8 GHz 频率的无线电信号发送和接收，支持USB 3.0高速模式。

USRP支持从直流信号到6 GHz频率的射频应用SDR开发平台。

LimeSDR 是一款低成本、开源和支持插件应用模式的SDR平台

6. 无线网络安全小结



如上图所示，总结了各种不同信号物理覆盖范围的无线网络构建技术和有线网络之间的关系。概括描述如下：

（1）无线网络主要解决的是终端设备接入网络的问题，有线网络主要解决的是骨干网络互联互通以及为无线网络提供广域网络传输的接入；

（2）WPAN、WLAN和蜂窝网络的物理信号覆盖范围逐渐增大。

在无线网络安全加固方面，针对蜂窝通信的建议是：

（1）尽快升级你的移动通信⽹络制式到4G；

（2）不要依赖 2G ⽹络的短信传送机密信息，避免明文嗅探风险；

（3）服务提供商要正确的实现验证码短信功能：不要在短信中同时出现完整帐号和验证码；（4）验证码有效周期尽可能短，建议重要验证码1分钟过期；

（5）遇到疑似伪造来源号码的电话和短信，回拨 可验证真伪。这里的 回拨 有两点需要注意。第一，对于陌生号码，为了避免 高额主叫话费 和落入电信诈骗的陷阱，并没有回拨的必要性。第二，需要 回拨 确认主叫号码真实性的场景主要指的是用你熟悉的电话号码（例如通讯录号码、常见短号码，如110，10086，银行客服电话等）呼入，但谈话内容疑似诈骗。这时，更换另⼀个⼿机号、固定电话，逐个号码手工输⼊⽅式进行回拨，能够规避主叫号码篡改类的电信诈骗风险。这是因为，虽然来电号码可能会通过类似VoIP等方式来伪造，但当我们回拨的时候，被叫号码却并不是诈骗犯所能控制和劫持的。类似的，通过伪基站方式发送的伪造发信人号码（例如伪造电信运营商客服号码、银行客服号码等）的短信，通过 更换另一个手机运营商号码 的手机（规避特定运营商链路被劫持和当前手机中木马的风险） 并且 离开当前位置超过2公里以上 （避开伪基站劫持） 上行 短信到该发信人号码，基本能够确保短信送达真正的号码所有者，而不是伪造号码的所有者。

针对蓝牙的安全加固建议是：

（1）默认不启用设备的蓝牙功能，除非需要用到；

（2）尽可能使用最低等级的蓝牙默认功耗，限制蓝牙传输距离，这需要蓝牙控制软件或操作系统提供这种高级配置功能；

（3）关闭蓝牙的“可被发现”能⼒；

（4）使用动态、健壮的PIN码，这主要依赖于蓝牙应用软件的开发者在开发软件时加入该特性的支持实现；

（5）尽可能使用⾼版本的蓝牙协议支持设备；

（6）关闭不需要的蓝牙功能，例如iOS系统的AirDrop功能；

（7）建议开启蓝牙配对设备的双向认证功能，这也主要依赖于蓝牙应用软件的开发者在开发软件时加入该特性的支持实现。

针对RFID的安全建议是：

（1）给你⼝袋/钱包里的RFID卡增加⼀个RFID屏蔽卡套，防⽌近距离复制；

（2）避免使用Mfiare Classic芯片卡，⽽采用更强加密算法的芯片卡，比如CPU卡；

（3）涉及⾦额等敏感数据应进⾏加密处理，禁⽌明⽂存储；

（4）读卡器与后端主机数据库实⾏线上作业，采用即时连线的⽅式进⾏系统核查；

（5）结合uid进⾏加密，并设置uid白名单，提⾼攻击者破解成本，但可能被特殊卡绕过；

（6）对全扇区采用非默认密码加密，提⾼破解成本，但可能通过DarkSide⽅式暴⼒破解。

第五章 智能终端操作系统安全概述

\*\* 终端怎么加固 网络传输怎么加密

1. 智能终端概述

1.1. Android

Linux内核和大量第三方开源组件支撑了Android底层关键技术构建。所以，我们在探讨和研究Android系统安全的时候，不能忽略这些外部项目和上游项目的代码安全性问题完全有可能影响到Android系统中的其他专属组件或模块。

Android生态圈中的厂商可以在获得Google官⽅授权允许后，制造所有类型设备。Android系统基于开源授权协议标准自⾏修改和使用。第三方和自建Android应用分发渠道Google官⽅既未授权允许，也未明令禁⽌。一些⼭寨机厂商在看到了这个高度开放和可定制化的生态圈之后，在手机出厂时、销售环节就被『刷⼊』恶意代码，这种威胁方式目前被统称为“来自供应链的威胁”。

目前Android生态圈的一些特点：

（1）Google主导的全面开放生态圈，从源代码到文档，通过开放源代码授权的方式（Android 开放源代码项目所有组成部分的首选许可是Apache 2.0许可）向全球商业化公司开放了绝大部分的源代码、文档和开发过程。但请注意，相关开源许可并未涵盖专有商标和品牌特征授权（例如这种特殊样式的字体徽标）。

（2）Apache 2.0 是适用于开放源代码的商业软件许可。大多数 Android 平台和文档都是根据 Apache 2.0 许可授权用户使用。尽管该项目会尽可能遵循此首选许可，但也可能存在例外情况，例如，如果文档（代码注释）的提取来源是根据 GPLv2 或其他许可授权的源代码模块。在这些情况下，源代码模块所遵循的许可将适用于从源代码模块中提取的文档。

（3）以Android操作系统为核心的全面多行业发展，全球范围内从硬件设备制造商、互联网软件服务提供商、电信运营商等多个行业深度参与；

（4）多样化的设备、系统与应用，小到手表、手机、平板电脑等，大到智能电视、智能汽车等，Google Play应用市场中的独立App数量从2009年时的16000个增长到2017年9月的3,300,000个，无论是绝对数量还是数量增长率都体现了Android应用生态圈的繁荣发展；

（5）开发者要面临碎片化应用运行平台，由于市场上大量设备不能及时更新到最新版本的系统，应用开发者在开发应用时就不得不做大量旧版本兼容性开发和测试工作。而对于企业级应用来说，一年一次大的Android系统版本更新，在选择使用最新特性和继续坚守已被证明的稳定可用性方面面临抉择；

（6）国内消费者目前（截止2017年12月）只能使用定制化的安卓系统，一方面无法及时升级系统和安装系统级别的安全更新。另一方面，大多数设备厂商只知道每年推出多款新设备，却普遍忽视对已有设备上系统的持续更新维护或更新维护周期较短（1年左右）。

1.2. iOS

iOS是苹果公司为移动设备所开发的专有操作系统，所支持的设备包括iPhone、iPod touch和iPad。与Android不同，iOS不支持任何非苹果的硬件设备。除了iOS，苹果公司生产的智能电视使用的是专有操作系统tvOS，智能手表使用的专用操作系统watchOS。

iOS系统的基本组成分为从底层到顶层依次为：Core OS、Core Services、Media和Cocoa Touch。其中，Core OS 是基于 FreeBSD 和 Mach 所改写的 Darwin ，是开源、符合 POSIX 标准的一个 Unix 核心。提供了整个 iOS 的一些基础功能。例如：硬件驱动，内存管理，程序管理，线程管理（POSIX），文件系统，网络（BSD Socket），以及标准输入输出等等。所有这些功能都会通过C语言的API来提供。Core OS层的驱动也提供了硬件和系统框架之间的接口。然而，由于安全的考虑，只有有限的系统框架类能访问内核和驱动。

iOS生态圈相比较于Android生态圈的一个最显著特点就是：封闭。

小结一下iOS为代表的苹果移动互联生态圈，可以概括为如下特点：

（1）苹果强势主导的授权有限开放生态圈；

（2）消费级应用商业变现能力目前强于Android生态圈；

（3）应用多样性发展逐渐落后于Android生态圈，例如在电视、盒子、投影设备、智能家居等领域的市场占有率要落后于Android生态圈。

1. iOS系统安全概述

现代操作系统的版本升级发展历程，普遍经历了一个从对安全漠视，到引起重视，到投入具体资源去进行安全设计和安全加固实现，最终将安全评估、安全检测和安全加固等内置在整个产品迭代的生命周期之中每个阶段。

以下整理了iOS系统各个版本引入的主要安全特性或新增特性带来的新型安全威胁：

iOS 4

禁止第三方程序访问所有通话历史

禁止第三方程序拦截呼入和呼出的通话

GPS定位功能需要用户明确授权

限制后台联网程序种类（只允许音乐、VoIP类应用在后台运行），iOS 7 开始系统放宽后台静默和联网程序的限制

限制后台静默运行程序（音乐、GPS 、VoIP 、消息推送守护程序以及周边配件附属的程序）

系统相机拍摄的照片默认包含GPS位置信息，如果图片未经处理上传到第三方网站或发送给他人，会暴露照片的拍摄位置信息；

iOS 5

禁止第三方程序访问短信收件箱和后台静默发送短信

推出了iMessage服务，并且很快被黑产盯上，用来群发垃圾信息（通过手机号或电子邮件地址，只要接收者启用了iMessage并绑定了手机号或电子邮件地）；

iOS 6

第三方程序访问通讯录、日历、照片需要用户明确授权

iOS 7

iphone5s引入Touch ID，支持指纹解锁

设备激活强制使用Apple ID并联网认证身份，防止设备被盗、恶意清空数据和设置等

USB首次连接时双向身份认证，只有用户处于屏幕解锁状态时，才可以点击信任当前USB连接的电脑（无需输入锁屏密码），杜绝通过USB植入恶意代码

原生通讯录支持设置电话号码黑名单

iOS 8

开放系统全局的第三方输入法，在此之前，系统没有给第三方输入法开放编程接口

允许第三方程序使用Touch ID来进行身份认证

iOS 8 设备在扫描WiFi 时，系统会使用随机的 MAC 地址来防止设备被跟踪，只有在连接成功后才会使用真正的设备物理MAC地址

系统相册在删除照片时，默认是将照片移动到“最近删除”的相册并保留30天后再自动真删除。避免用户误删除照片的同时，可能会导致一些秘密和隐私照片没有被及时删除，其他人一旦可以物理接触到手机，就可以从“最近删除”的相册中恢复出照片。这是一个典型的产品机密性和可用性矛盾调和、妥协的设计与实现；

黑产利用新增的 iOS日历邀请 功能群发垃圾信息，这种日历弹窗小广告，不显示来源，仅仅显示广告内容、发送人和一个时段。我们往往会下意识地点击「拒绝」，但没想到这个小动作就能把自己的个人名字泄露了给对方。保留iCloud日历同步功能的同时，防止垃圾邀请充斥日历的操作方法如下：

登陆iCloud

左下角齿轮-偏好设置-高级：邀请，选择发送电子邮件给XXX邮箱，保存

点击编辑，点击＋ 完成

选择想删除的事件，在弹出框标题旁的颜色点上单击，选择新建的分类，保存

点击编辑，点击新建分类旁边的－，删除分类，确定这样垃圾事件就被删除了，以后的事件邀请会发送到邮箱，然后使用邮箱的垃圾邮件屏蔽设置即可。

iOS 10

iOS 10.3推出了iCloud照片分享功能，黑产利用该功能强制推送包含广告和垃圾信息的相册，其基本原理和iMessage垃圾信息类似。除了系统内置反垃圾信息功能并持续改进之外，普通用户如果深受垃圾信息骚扰只能选择禁用“iCloud照片分享”功能；

iOS 11

位置信息访问控制更精细，iOS 10上的App开发者可以选择向用户隐藏“使用应用时访问位置信息”选项，iOS 11开始这个选项无法再被隐藏

照片库访问控制允许开发者申请只读访问授权

手机即使处于解锁状态通过数据线缆连接到一台陌生电脑时依然需要 输入设备解锁口令 才能信任这台电脑（原先只需要在解锁的手机上点击“允许”按钮）

强制已有的iCloud账户启用 双因素认证 来代替原先的“两步认证”

通过控制中心“关闭”Wi-Fi开关并不是真的关闭Wi-Fi功能，只是让当前设备从当前连接的任何网络断开

当通过控制中心停用 Wi-Fi 后，自动加入附近任何 Wi-Fi 网络的功能也会被停用，直到：您步行或驾车到新位置 和 当地时间凌晨 5 点 。这就意味着，EvilTwin 类攻击对于iOS手机用户将有更多机会得手，因为用户手机的Wi-Fi功能将会随着用户位置的变化、系统时间的变化而自动被系统强制开启。

2013.5

App Store审核标准中新增对App读取设备MAC地址的禁令

1. Android系统安全概述

由于Android底层基于Linux内核，所以Linux内核的相关漏洞在某些条件下也会影响到Android，上图中统计的CVE漏洞数据库中的Linux内核相关漏洞数据也可以用来让我们了解Android系统的安全基础现状和历史变化情况。

3.1. 系统安全评估方法

在了解了iOS和Android系统的CVE漏洞数据库中的历年漏洞数据统计信息之后，我们可以根据一个系统（历史上出现的）漏洞数量的多少来评估系统安全性吗？可以根据系统中（历史）包含漏洞的CVSS评分统计量（总和、平均值）高低来评估系统安全性吗？要想回答好这个问题，我们需要运用风险评估的基本理论和方法，从系统整体出发，从不同角度出发来评估系统安全性。

首先，信息安全的本质是持续对抗，信息资产的价值高低会影响攻击者的攻击目标选择策略。市场占有率高、用户基数大的系统和应用软件会吸引到更多的攻击技术研究者，研究和挖掘漏洞参与度的提升，一方面意味着漏洞被隐藏的可能性越低、漏洞被尽早曝光的可能性越高，另一方面系统和应用软件开发团队需要时间去深入研究和落地应用安全加固策略和措施，所以反映在公开漏洞数量和严重性程度（例如CVSS平均分）随时间变化曲线方面会经历一个从上升到下降到相对平稳运行的阶段。

其次，漏洞数量和软件代码行数成非线性正相关性比例，虽然不同安全开发水平的团队开发出来的软件的“漏洞密度”（每千行代码包含漏洞的数量）有差异 ，但总体上随着代码行数的增加，软件包含的漏洞数量也会随之增长。从软件规模的角度来看，大型软件、操作系统中包含漏洞的数量是要大于小型软件和简单功能软件的，只是是否被曝光和曝光时间的早晚问题。

再者，单一的漏洞数量指标无法说明系统面临的威胁多少和风险高低。首先，漏洞的生命周期管理方法会影响到实际漏洞的发现和利用难度。例如对于开源和闭源软件来说，同样的漏洞保密和曝光周期条件下，开源软件由于开发过程的完全透明化，更容易通过源代码补丁比对的方式发现0-day漏洞（厂商还未推送补丁）并以相对最低的时间和人力成本开发出漏洞利用程序。而闭源软件就无法使用源代码补丁比对技术发现0-day漏洞，需要通过分析成本更高的二进制补丁比对技术在厂商推送补丁程序之后（N-day）才能发现漏洞。其次，单一漏洞对系统安全性的威胁等级是有差别的，例如，远程任意代码执行类漏洞对系统完整性和机密性的破坏性程度会受到实际攻击过程是否需要通过身份认证、是否需要高权限账户等制约。最后，系统的安全性评估是要结合系统的具体安全需求来进行评估的。如果一个漏洞被利用并不会影响到系统的具体安全需求，那么，这样的单个漏洞是无法影响系统整体安全性的。

不仅如此，系统的安全性评估是一个系统工程，不能简单的将系统中各个组件的漏洞CVSS评分进行统计（累加或求均值或求最大最小值等）。CVSS是针对单一漏洞危害的评价指标，而实际的网络与系统攻击过程往往是多个漏洞的组合利用过程，多个低危漏洞的优化组合同样可能会对企业网络和系统带来致命一击。根据信息系统安全评估的“木桶原理”，系统整体安全性高低取决于系统中最短板的安全性水平，换句话来说高危漏洞的存在性会显著降低系统整体安全性。但高危漏洞的存在性如果不能转化为一次成功的漏洞利用过程（由于各种安全机制的保护），则对于系统整体安全性目标保护而言，高危漏洞并没有降低系统的安全性。除此之外，漏洞的生命周期管理方法、开源或闭源软件开发模式、系统具体如何使用和部署缺陷组件的方法等，都会影响到系统安全性的评估结果。

最后，已修复的漏洞即使被公开，只要被评估系统已经安装了对应的安全补丁，那么历史上出现的漏洞数量再多也不会影响到当前系统的安全性。

3.2. Android VS. iOS 面临的威胁



认为iOS开发平台比Android安全的几个看似合理的理由：

1、iOS对开发者的行为有更加严格的限制，并且有更加严谨的沙箱保护机制；

2、iOS应用程序在发布到应用商店前会经过严格审查，以防止有恶意软件流入。

3.2.1. iOS 上的企业级应用分发方式风险

苹果的应用分发授权模式主要有2种，分别是个人应用和企业级应用。个人应用，必须且仅允许通过App Store审核通过后允许消费者下载安装使用。企业级应用，无需App Store审核，消费者可以直接下载安装使用。如果企业作恶或者企业的分发证书被盗取滥用呢？假面攻击 最早是在 2014 年 7 月由FireEye移动安全人员提交，通过使用企业账户安装App，替代通过苹果官方App Store安装的App应用，该漏洞只要替换的应用程序使用了和App Store安装的应用程序有相同的 bundle identifier，即可完成IPA安装程序的替换。假面攻击的危害主要包括：

通过诱导用户替换App Store 中下载的App，因为做了二次封装和改动，可以获取用户登录App的账户及密码，比如邮箱账户密码或者银行账户密码；

该APP应用下的隐私数据，比如配置信息，缓存文件，本地sqlite数据库可以被假冒程序盗取访问；

虽然苹果对应用有沙盒保护，只能获取该应用下的敏感数据，但是可以通过已有的漏洞绕过沙盒保护机制对系统层进行攻击。

假面攻击可以通过USB连接安装或者无线网络安装App：

USB连接方式使用PC与手机端底层驱动libimobiledevice通讯，在非越狱的情况下将替换的App安装到手机中。知名的iOS平台木马 WireLurker 就是通过这种方式实现的；

无线网络环境下可通过短信、iMessage、邮件中包含一个链接的方式诱导用户去安装，导致原有通过App Store下载的APP被替换，这种方式受众面更广。

假面攻击利用的漏洞对越狱和非越狱的 iOS 7.1.1, 7.1.2, 8.0, 8.1 及其 8.1.1 beta版本以下均受影响。

我们在此奉劝iOS用户：谨慎越狱。一方面，越狱程序本身有可能已经被恶意捆绑后门和木马程序，安装这样的修改版越狱程序之后你的手机就成为黑客可以远程控制的对象了。另一方面，越狱应用本就可以无视 iOS 的沙盒机制、无限制访问原本受保护的资源和调用私有API，当系统面临类似假面攻击威胁时，越狱会增加系统被完全攻破的风险。不仅如此，一旦系统有更新升级的时候，越狱版用户一旦选择同步更新，则越狱程序利用的旧版系统提权漏洞往往就会立刻失效。所以，大多数情况下，越狱系统的用户会选择等待下一个升级版越狱程序推出之后，在更新到对应的可越狱版本。这就意味着用户主动放弃了安装最新的安全补丁的机会，系统会较长时间处在 N-day漏洞 （官方已有安全补丁的漏洞利用程序目标漏洞，但由于用户选择不安装补丁升级导致非 0-day 漏洞利用依然可以成功利用）攻击时间窗口。

3.2.2. iOS 上的恶意配置文件风险

攻击者利用配置文件可以绕过iOS的安全机制，配置文件可以重新定义各种系统功能参数，如运营商、MDM（移动设备管理）和网络设置。用户可能被诱骗下载这样的恶意配置文件，从而在不知情的情况下被导向攻击者控制的服务器，进一步被安装恶意软件，甚至被解密通信。

查看手机上是否安装了配置文件的操作方法是：设置 - 通用 - 描述文件与设备管理，如果发现有未知的配置文件安装，请及时删除。在该实例中我们可以看到，一共安装了3个配置文件。其中，有1个Wi-Fi网络配置文件，另外2个是企业自签发的根CA证书。试想，如果上述配置文件和证书是攻击者添加的，那么攻击者通过添加的CA证书可以模拟银行等安全加密站点（利用预置的根证书任意签发仿冒站点的证书）实现完美钓鱼攻击。恶意Wi-Fi网络配置文件可以用来实现对用户无线网络连接的完美劫持，配合恶意AP可以完全监听和控制用户的网络流量。

攻击者可以通过发送钓鱼邮件和设置网页链接文件的方式，诱骗用户下载安装恶意配置文件。如下图所示是一次鱼叉式钓鱼邮件附件方式诱骗用户安装恶意iOS配置文件的示例：

虽然iOS对于没有使用可信CA证书签发的配置文件会向用户展示 红色 的 未签名 警告，但由于大多数企业使用的配置文件也是未签名的，大多数用户会选择忽略这些警告信息，直接安装。

3.3. 系统安全与网络安全是密不可分的

系统与系统之间的通信过程时刻存在着监听、劫持等中间人攻击风险：Wi-Fi钓鱼、Wi-Fi口令破解、Wi-Fi中间人攻击、GSM监听、GSM伪基站等等。这些风险的来源是网络攻击，但是攻击的目标往往是为了获得系统控制权。从防御者的角度来看，在一个不可信、不安全的网络中进行安全的数据传输与通信控制，可以使用端到端加密技术，但这种加密传输能够达到机密性和完整性保护的前提是通信双方的系统安全是有保障的。一旦系统被植入木马、安装了键盘记录器、屏幕监控等，网络安全机制就会失效。网络加密传输的可靠实现依赖于密钥管理的安全性，而系统安全又是密钥管理安全性的基础和保障。越来越多的系统采用专用硬件加密芯片来管理重要密钥的生命周期（存储、使用、更新、销毁等）。

第六章 安卓系统访问控制策略与机制

1. Android安全机制概览

Android 平台的主要组件包括：设备硬件、Android 操作系统、Android 应用运行时、预先安装的应用、用户安装的应用

1.1. 硬件相关安全机制

1.1.1. 密钥存储与密码学算法

借助系统芯片 (SoC) 中提供的可信执行环境，Android 设备可以为 Android 操作系统、平台服务甚至是第三方应用提供由硬件支持的强大安全服务。寻求 Android 专用扩展程序的开发者应访问 android.security.keystore。

Keystore 在 Android 6.0 中得到了显著增强，不仅增加了对称加密模块（AES 和 HMAC），还增加了针对由硬件支持的密钥的访问控制系统。访问控制在密钥生成期间指定，并会在密钥的整个生命周期内被强制执行。可以将密钥限定为仅在用户通过身份验证后才可使用，并且只能用于指定的目的或只有在具有指定的加密参数时才可使用。对于工程实现方面的细节可以查阅Android官方 面向实现人员的参考资料

在 Android 6.0 之前的版本中，Android 已有一个非常简单的由硬件支持的加密服务 API（由 0.2 和 0.3 版的 Keymaster 硬件抽象层 (HAL) 提供）。该 Keystore 能够提供数字签名和验证操作，以及不对称签名密钥对的生成和导入操作。该 API 在许多设备上都已实现，但有许多安全目标无法只通过一个签名 API 来轻松达成。Android 6.0 中的 Keystore 在该 Keystore API 的基础上进行了扩展，能够提供更广泛的功能。

1.1.2. Trusty TEE

Trusty 是一组在移动设备上支持可信执行环境 (Trusted Execution Environment, TEE) 的软件组件。

Trusty 包含以下组件：

一个在用于提供 TEE 的处理器上运行的操作系统（Trusty 操作系统）

适用于 Android 内核 (Linux) 的驱动程序，旨在促进与在 Trusty 操作系统下运行的应用之间的通信

一组适用于 Android 系统软件的库，旨在使用内核驱动程序促进与在 Trusty 操作系统中执行的可信应用之间的通信

重要提示：Trusty 和 Trusty API 随时可能发生变化。

所有 TEE 操作系统（不仅仅是 Trusty）均可用于 TEE 实现。

TEE 处理器通常是系统内的单独微处理器，或是主处理器的虚拟化实例。TEE 处理器与系统的其余部分之间使用由硬件支持的内存和 I/O 保护机制隔离开来。

TEE 处理器如今已成为移动设备的基本组成部分。这些设备上的主处理器会被视为“不可信”，它们无法访问制造商用于存储机密数据（例如，设备专用加密密钥）的特定 RAM、硬件寄存器和 用户空间文件系统（Filesystem in Userspace，Fuse） 区域。在主处理器上运行的软件会将所有需要使用机密数据的操作委派给 TEE 处理器。

在 Android 生态系统中，最广为人知的不可信主处理器示例是用于受保护内容的 DRM 框架。在 TEE 处理器上运行的软件可以访问解密受保护内容所需的设备专用密钥。主处理器只能看到已加密的内容，这样一来，就可以针对基于软件的攻击提供高级别的安全保障和保护。

TEE 还有许多其他用法，例如移动支付、安全银行、全盘加密、多因素身份验证、设备重置保护、抗重放攻击的持久存储、无线显示（“投射”）受保护的内容、安全的 PIN 码和指纹处理，甚至还有恶意软件检测。

Trusty 提供用于开发以下两类应用的 API：

在 TEE 处理器上运行的可信应用或服务

在主处理器上运行并使用可信应用提供的服务的普通/不可信应用

在主处理器上运行的软件可以使用 Trusty API 连接到可信应用并与它们交换任意消息，就像通过 IP 提供的网络服务一样。应用负责使用应用级协议确定这些消息的数据格式和语义。消息传递的可靠性由底层 Trusty 基础架构（采用在主处理器上运行的驱动程序的形式）来保证，并且通信完全是异步进行的。

可信应用会以单独进程的形式在 Trusty 操作系统内核下运行。每个进程都会利用 TEE 处理器的 MMU 功能在各自的虚拟内存沙盒中运行。内核会使用由安全计时器驱动且按优先级进行调度的轮询调度程序安排这些进程。在最新版本的 Trusty 中，所有 Trusty 应用均具有相同的优先级。

可以使用 C/C++（对 C++ 的支持有限）编写适用于 Trusty 操作系统的应用，此类应用可以访问小型的 C 库。main() 函数目前不接受任何参数。系统调用存根是作为该库的一部分在本机汇编代码中提供的，因此可按名称访问系统调用。

所有 Trusty 应用均为单线程应用；目前不支持在 Trusty 用户空间中使用多线程。

Trusty 应用会在加载期间初始化一次，并且在 TEE 处理器重置之前，会一直保留在内存中。Trusty 目前不支持动态加载和取消加载应用。

可信应用是作为 事件驱动型服务器 编写的，会等待其他应用或主处理器上运行的应用发出的命令。此外，可信应用也可以作为其他可信服务器应用的客户端。以下 API 部分中介绍的事件将由 Trusty 内核传送到可信应用。

1.2. 软件相关安全机制

1.2.1. 系统和内核安全

在操作系统级别，Android 平台不仅提供 Linux 内核的安全功能，而且还提供安全的进程间通信 (IPC) 机制，以便在不同进程中运行的应用之间安全通信。操作系统级别的这些安全功能旨在确保即使是本机代码也要受应用沙盒的限制。无论相应代码是自带应用行为导致的结果，还是利用应用漏洞导致的结果，系统都能防止恶意应用危害其他应用、Android 系统或设备本身。要了解可以采取哪些措施来增强设备上的内核，请参阅Android官网的 内核配置。要了解必需的设置，请参阅 Android 兼容性定义文档 (CDD)。

Linux安全

Android 平台的基础是 Linux 内核。Linux 内核多年来一直应用得非常广泛，并且用在了数百万种安全敏感型环境中。在历经数以千计的开发者不断研究、攻击和修复之后，Linux 已成为许多公司和安全专业人士信任的一款既稳定又安全的内核。

作为移动计算环境的基础，Linux 内核为 Android 提供了一些关键的安全功能，其中包括：

基于用户的权限模式

进程隔离

用于实现安全 IPC 的可扩展机制

能够移除内核中不必要的和可能不安全的部分

作为多用户操作系统，Linux 内核的一个基本安全目标是将用户资源彼此隔离开来。Linux 的安全理念是防范用户资源之间相互侵扰。因此，Linux 可以：

防止用户 A 读取用户 B 的文件

确保用户 A 不会占用用户 B 的内存

确保用户 A 不会占用用户 B 的 CPU 资源

确保用户 A 不会占用用户 B 的设备（例如，电话、GPS、蓝牙）

应用沙盒

Android 平台利用基于用户的 Linux 保护机制来识别和隔离应用资源。Android 系统会为每个 Android 应用分配一个独一无二的用户 ID (UID)，并使它们以这个用户身份在单独的进程中运行。这种方法与其他操作系统（包括传统的 Linux 配置）采用的方法不同。在其他操作系统中，多个应用会以相同的用户权限运行。

这样就设置了一个内核级应用沙盒。内核会在进程级别通过标准的 Linux 内容（例如，分配给应用的用户 ID 和组 ID）强制执行应用和系统之间的安全功能。默认情况下，应用不能彼此交互，而且应用对操作系统的访问权限会受到限制。如果应用 A（一个单独的应用）尝试执行恶意操作，例如在没有权限的情况下读取应用 B 的数据或拨打电话，操作系统会阻止此类操作，因为应用 A 没有适当的用户权限。沙盒非常简单，可审核，并且基于已有数十年历史的 UNIX 风格的进程用户隔离和文件权限机制。

由于应用沙盒位于内核中，因此该安全模型的保护范围扩展到了本机代码和操作系统应用。位于内核上方的所有软件（例如，操作系统库、应用框架、应用运行时和所有应用）都会在应用沙盒中运行。在某些平台上，为了强制执行安全功能，会限制开发者只能使用特定的开发框架、API 组或语言。在 Android 上，并没有限制必须如何编写应用才能强制执行安全功能；在这一方面，本机代码与直译码一样安全。

在某些操作系统中，一个应用中的内存损坏错误可能会导致位于同一内存空间中的其他应用出现损坏，进而导致设备的安全性完全遭到破坏。由于所有应用及其资源都在操作系统级别的沙盒内，因此，如果出现内存损坏错误，将只有在相应应用的环境中才能发生任意执行代码的行为，而且只能是以操作系统确立的权限执行代码。

与所有安全功能一样，应用沙盒并不是坚不可摧的。不过，要在经过适当配置的设备上攻破应用沙盒这道防线，必须要先攻破 Linux 内核的安全功能。

系统分区和安全模式

系统分区包含 Android 的内核，以及操作系统库、应用运行时、应用框架和应用。该分区设为了只读分区。当用户将设备启动到安全模式时，第三方应用可由设备所有者手动启动，但不会默认启动。

文件系统权限

在 UNIX 风格的环境中，文件系统权限可确保一个用户不能更改或读取另一个用户的文件。在 Android 中，每个应用都以自己的用户身份运行。除非开发者明确地与其他应用共享文件，否则一个应用不能读取或更改另一个应用创建的文件。

安全增强型 Linux

Android 使用安全增强型 Linux (SELinux) 来应用访问控制策略并对进程建立强制访问控制 (mac)。如需详细信息，请参阅 Android 中的安全增强型 Linux。

验证启动

Android 6.0 及更高版本支持验证启动功能和 device-mapper-verity。验证启动功能旨在保证设备软件（从硬件信任根直到系统分区）的完整性。在启动过程中，无论是在每个阶段，都会在进入下一个阶段之前先验证下一个阶段的完整性和真实性。

Android 7.0 及更高版本支持严格强制执行的验证启动，这意味着遭到入侵的设备将无法启动。

如需更多详细信息，请参阅验证启动。

加密

Android 提供了一套加密 API 供应用使用，其中包括标准和常用加密算法（例如，AES、RSA、DSA 和 SHA）的实现 API。此外，Android 还提供了适用于更高级别协议（例如，SSL 和 HTTPS）的 API。

Android 4.0 中引入了 KeyChain 类，以便应用使用系统凭据存储空间来存储私钥和证书链。

加密是使用对称加密密钥对 Android 设备上的所有用户数据进行编码的过程。设备经过加密后，所有由用户创建的数据在存入磁盘之前都会自动加密，并且所有读取操作都会在将数据返回给调用进程之前自动解密数据。加密可确保未经授权方在尝试访问相应数据时无法读取它们。

Android 有两种设备加密方法：全盘加密和文件级加密。

全盘加密

全盘加密是使用已加密的密钥对 Android 设备上的所有用户数据进行编码的过程。设备经过加密后，所有由用户创建的数据在写入磁盘之前都会自动加密，并且所有读取操作都会在将数据返回给调用进程之前自动解密数据。

全盘加密是在 Android 4.4 版中引入的，不过 Android 5.0 中又引入了以下新功能：

创建了快速加密方式，这种加密方式只会对数据分区中已使用的分块进行加密，以免首次启动用时过长。目前只有 EXT4 和 F2FS 文件系统支持快速加密。

添加了 forceencrypt fstab 标记，以便在首次启动时进行加密。

添加了对解锁图案和无密码加密的支持。

添加了由硬件支持的加密密钥存储空间，该空间使用可信执行环境（TEE，例如 TrustZone）的签名功能。如需更多详细信息，请参阅存储已加密的密钥。

注意：对于升级到 Android 5.0 的设备，如果升级之后进行了加密，则可以通过恢复出厂设置还原到未加密状态。在首次启动时加密的新 Android 5.0 设备无法还原到未加密状态。

Android 5.0 及更高版本支持全盘加密。全盘加密是使用单个密钥（由用户的设备密码加以保护）来保护设备的整个用户数据分区。设备启动后，用户必须提供其凭据才能访问磁盘的任何部分。

虽然这非常有利于确保安全性，但如果采用这种加密方式，当用户重新启动设备后，手机的大多数核心功能都将无法立即可用。由于对数据的访问受单个用户凭据的保护，因此闹钟等功能将无法运行，无障碍服务将无法使用，并且手机将无法接听电话。

文件级加密

Android 7.0 及更高版本支持文件级加密。采用文件级加密时，可以使用不同的密钥对不同的文件进行加密，并且可以对这些文件进行单独解密。支持文件级加密的设备还支持一种称为直接启动的新功能。该功能处于启用状态时，已加密设备在启动后将直接进入锁定屏幕，从而可让用户快速访问重要的设备功能，例如无障碍服务和闹钟。

引入文件级加密和新 API 后，便可以将应用设为加密感知型应用，这样一来，它们将能够在受限环境中运行。这些应用将可以在用户提供凭据之前运行，同时系统仍能保护私密用户信息。

获取设备的 Root 权限

默认情况下，在 Android 上，只有内核和一小部分核心应用能够以 Root 权限运行。Android 不会阻止具有 Root 权限的用户或应用修改操作系统、内核或任何其他应用。一般来说，Root 对所有应用和所有应用数据拥有完整访问权限。如果用户在 Android 设备上更改权限来向应用授予 Root 访问权限，则会使遭受恶意应用攻击以及遭受潜在应用缺陷侵扰的安全风险增加。

能够修改自己的 Android 设备对于使用 Android 平台的开发者来说非常重要。在许多 Android 设备上，用户都可以解锁引导加载程序(bootloader)，以便安装替代操作系统。这些替代操作系统可能会允许所有者获得 Root 访问权限，以便他们调试应用和系统组件，或者访问 Android API 未提供给应用的功能。

在有些设备上，能够亲手控制设备并拥有 USB 数据线的用户可以安装能够向其提供 Root 权限的新操作系统。为了保护所有现有用户数据免遭入侵，引导加载程序解锁机制要求引导加载程序在解锁期间清空所有现有用户数据。利用内核错误或安全漏洞获得 Root 访问权限后，可以绕过这种保护机制。

使用存储在设备上的密钥对数据进行加密的做法并不能防止 Root 用户访问应用数据。应用可以使用存储在设备之外的密钥（例如，存储在服务器上的密钥，或用户密码）进行加密，从而添加一道数据保护屏障。如果不提供密钥的话，这种方法可以提供临时保护，但在某些时候，必须要先将密钥提供给应用，然后 Root 用户才能访问相应应用。

一种更强大的防止 Root 用户获取数据的方式是使用硬件解决方案。原始设备制造商 (OEM) 可以选择实现仅允许访问特定类型的内容的硬件解决方案，例如，适用于视频播放的 DRM 或适用于 Google 电子钱包的 NFC 相关可信存储空间。

如果设备丢失或被盗，Android 设备上的全文件系统加密功能可以使用设备密码来保护加密密钥，这样一来，修改启动加载程序或操作系统的做法将不足以在没有用户设备密码的情况下访问用户数据。

用户安全功能

文件系统加密

Android 3.0 及更高版本提供全文件系统加密功能，因此所有用户数据都可以在内核中进行加密。

Android 5.0 及更高版本支持全盘加密。全盘加密功能旨在使用单个密钥（由用户的设备密码加以保护）来保护设备的整个用户数据分区。在启动时，用户必须先提供其凭据，然后才能访问磁盘的任何部分。

Android 7.0 及更高版本支持文件级加密。采用文件级加密时，可以使用不同的密钥对不同的文件进行加密，并且可以对这些文件进行单独解密。

如需关于实现文件系统加密的更多详细信息，请参阅加密部分。

密码保护

Android 可以配置为在提供对设备的访问权限之前先验证用户提供的密码。除了防止未经授权使用设备外，该密码还可以保护用于进行全文件系统加密的加密密钥。

设备管理员可以要求使用密码和/或密码复杂度规则。

设备管理

Android 2.2 及更高版本提供 Android Device Administration API，该 API 在系统级别提供设备管理功能。例如，内置的 Android 电子邮件应用可以使用该 API 来改善 Exchange 支持。通过电子邮件应用，Exchange 管理员可以跨设备强制执行密码政策（其中密码包括字母数字密码或数字 PIN 码）。管理员还可以远程清除（即恢复出厂默认设置）丢失或被盗手机上的数据。

除了在 Android 系统自带的应用中使用外，该 API 还可供提供设备管理解决方案的第三方使用。如需关于该 API 的详细信息，请参阅设备管理。

1.2.2. 应用安全性

Android 提供了一个适用于移动设备的开放源代码平台和应用环境。核心操作系统基于 Linux 内核。Android 应用通常都是使用 Java 编程语言编写的，并在 Dalvik 虚拟机中运行。不过，也可以使用本机代码编写应用。应用是通过文件扩展名为 .apk 的单个文件安装的。

Android 应用的主要构造组件包括：

AndroidManifest.xml：AndroidManifest.xml 是控制文件，用于告诉系统如何处理应用中的所有顶级组件（具体来说就是下面介绍的活动、服务、广播接收器和内容提供程序）。该文件还用于指定需要哪些权限。

Activity：一般情况下，活动（Activity） 是指聚焦于用户的单个任务的代码。活动通常包括向用户显示界面，但并不一定会这样，有些活动就从不显示界面。通常情况下，应用的入口点是应用的其中一项活动。

Service：服务 是指在后台运行的代码的主体。服务可以在自己的进程中运行，也可以在其他应用的进程中运行。其他组件会“绑定”到某项服务，并通过远程过程调用来调用该服务的方法。比如媒体播放器就是一项服务：即使用户退出媒体选择界面，也可能仍然希望音乐继续播放。即使界面已关闭，服务也可使音乐继续播放。

BroadcastReceiver：广播接收器 是在操作系统或其他应用发出称为 Intent 的 IPC 机制时实例化的对象。例如，应用可以注册一个接收器来接收电量不足消息，并可以根据该信息改变自己的行为。

Android 权限模式：访问受保护的 API

Android 上的所有应用均在应用沙盒（本文档的前面对其进行了介绍）内运行。默认情况下，Android 应用只能访问有限的系统资源。系统负责管理 Android 应用对资源的访问权限。如果资源使用不当或被恶意使用，可能会给用户体验、网络或设备上的数据带来不利影响。

这些限制是通过多种不同的形式实现的。有些功能会因 Android 有意未提供针对敏感功能的 API（例如，Android 中没有用于直接操控 SIM 卡的 Android API）而受到限制。在某些情况下，角色分离能够提供一种安全措施，就像按应用隔离存储空间一样。在其他情况下，敏感 API 旨在供可信应用使用，并由一种称为“权限”的安全机制进行保护。

这些受保护的 API 包括：

摄像头功能

位置数据 (GPS)

蓝牙功能

电话功能

短信/彩信功能

网络/数据连接

这些资源只能通过操作系统进行访问。要使用设备上受保护的 API，应用必须在其清单中定义所需的功能。在准备安装应用时，系统会向用户显示一个对话框，其中会注明应用请求的权限并询问是否继续安装。如果用户选择继续安装，系统会认定用户已授予应用请求的所有权限。用户不能单独授予或拒绝个别权限，而是必须要一起授予或拒绝应用请求的所有权限。

获得授权后，应用只要安装在设备上，便会一直拥有这些权限。为了避免用户混淆，系统不会再次通知用户向应用授予的权限，而核心操作系统中包含的应用或由原始设备制造商 (OEM) 绑定的应用不会向用户请求权限。应用卸载后，权限也会被移除，因此如果用户之后重新安装卸载的应用，系统会再次显示应用请求的权限。

在设备设置中，用户可以查看之前安装的应用的权限。此外，用户还可以根据需要在全局范围内停用某些功能，例如停用 GPS、无线功能或 WLAN。

如果应用尝试使用未在其清单中声明的受保护功能，权限失败通常会导致系统向应用抛回一个安全异常。受保护 API 权限检查会在最底层被强制执行，以防止出现规避行为。图 2 中显示了如果应用在安装时请求获得受保护 API 的访问权限，会导致系统向用户显示的消息示例。

如需关于系统默认权限的说明，请访问：https://developer.android.com/reference/android/Manifest.permission.html。 应用可以声明自己的权限以供其他应用使用。上述位置中未列出此类权限。

在定义权限时，protectionLevel 属性用于告诉系统如何让用户知道哪些应用需要或可以获得相应权限。如需关于如何创建和使用应用特有权限的详细信息，请访问：https://developer.android.com/guide/topics/security/security.html。

有些设备功能（例如，发送短信广播 Intent 的功能）不会供第三方应用使用，但可供原始设备制造商 (OEM) 预先安装的应用使用。这些应用使用 signatureOrSystem 权限。

用户如何了解第三方应用

当用户与第三方应用互动时，Android 会尽力让用户清楚这一情况，并让用户知道这些应用具备的功能。在安装任何应用之前，系统都会向用户显示一条明晰的消息，让用户知道要安装的应用请求获得的各项权限。安装完毕后，系统不会再次提示用户确认任何权限。

在安装前一刻显示权限的原因有很多。这时用户正在主动查看应用、开发者和功能方面的信息，以确定其是否符合自己的需求和期望。同样非常重要的一点是，他们尚未对要安装的应用做出心理或财务方面的承诺，并且可以轻松地将要安装的应用与其他替代应用进行比较。

有些其他平台会使用不同的方式通知用户，即在每个会话开始时或用户正在使用应用时请求权限。Android 的愿景是让用户能够随意在应用之间无缝切换。每次都让用户确认会拖慢用户的操作速度，而且会导致 Android 无法提供良好的用户体验。如果让用户在安装应用时查看权限，用户便可以在不愿意授予相应权限时选择不进行安装。

此外，许多界面研究表明，过度提示用户会导致用户开始在看到任何对话框时都选择“确定”。Android 的安全目标之一是向用户有效地传达重要的安全信息，而使用让用户习惯性忽略的对话框则无法做到这一点。如果只向用户提供一次重要信息并且仅在重要时刻提供，用户更有可能慎重思考他们要同意的是什么。

有些平台会选择完全不显示与应用功能有关的任何信息。这种方式会导致用户无法轻松了解和讨论应用功能。尽管无法使所有用户都是在充分了解相关信息的情况下做出决定，但 Android 权限模式可让众多用户轻松获取与应用相关的信息。例如，如果遇到意外的权限请求，经验更丰富的用户可能会询问有关应用功能的关键问题，并在 Google Play 等所有用户都可以看到的位置分享他们的疑问。

进程间通信

进程可以使用 UNIX 类型的任何传统机制进行通信。例如，文件系统、本地套接字或信号。不过，Linux 权限仍然适用。

Android 还提供了一些新的 IPC 机制：

Binder：一种基于功能的轻量型远程过程调用机制，在执行进程内调用和跨进程调用时能够实现出色的性能。Binder 是使用自定义 Linux 驱动程序实现的。请访问 https://developer.android.com/reference/android/os/Binder.html。

服务：服务（如上文所述）可提供能够使用 Binder 直接访问的接口。

Intent：Intent 是简单的消息对象，表示想要执行某项操作的“意图”。例如，如果当前应用想要显示某个网页，则会创建一个 Intent 实例并将其传递给系统，以此来表示想要访问相应网址的“意图”。然后，系统会找到一些知道如何处理该 Intent 的其他代码（在本例中为浏览器），然后运行该代码。Intent 也可用于在系统范围内广播有趣的事件（例如通知）。请访问 https://developer.android.com/reference/android/content/Intent.html。

ContentProvider：ContentProvider 是一个数据存储库，用于访问设备上的数据；典型的示例就是用于访问用户通讯录的 ContentProvider。应用可以访问其他应用通过 ContentProvider 提供的数据，还可以定义自己的 ContentProviders 来提供自己的数据。请访问 https://developer.android.com/reference/android/content/ContentProvider.html。 虽然可以使用其他机制（例如，网络套接字或全局可写文件）来实现 IPC，但上面这些都是建议使用的 Android IPC 框架。建议 Android 开发者遵循保护用户数据及避免引入安全漏洞方面的最佳做法。

虽然可以使用其他机制（例如，网络套接字或全局可写文件）来实现 IPC，但上面这些都是建议使用的 Android IPC 框架。建议 Android 开发者遵循保护用户数据及避免引入安全漏洞方面的最佳做法。

费用敏感 API

费用敏感 API 指可能会给用户或网络带来费用的任何功能。Android 平台已将费用敏感 API 放入由操作系统控制的受保护 API 列表中。如果有第三方应用请求使用费用敏感 API，必须要由用户授予明确的权限，它们才能使用这些 API。这些 API 包括：

电话

短信/彩信

网络/数据

应用内结算

NFC 访问

Android 4.2 进一步增强了对使用短信的控制。如果有应用尝试向使用付费服务的短代码发送短信（可能会产生额外的费用），Android 将会通知用户。用户可以选择是允许还是阻止该应用发送短信。

SIM 卡访问

第三方应用无法对 SIM 卡进行底层访问。操作系统负责处理与 SIM 卡之间的所有通信，包括访问 SIM 卡内存中的个人信息（通讯录）。应用也无法访问 AT 命令，因为这些命令完全由无线接口层 (RIL) 进行管理。RIL 不会为这些命令提供任何高层 API。

个人信息

Android 已将能够用于访问用户数据的 API 放入受保护 API 组中。在正常使用期间，Android 设备还会收集用户安装的第三方应用内的用户数据。选择分享这些信息的应用可以使用 Android 操作系统权限检查功能保护来自第三方应用的数据。

可能包含个人信息或个人身份信息（例如，通讯录和日历）的系统内容提供程序在创建时便已拥有明确确定的权限。这种精细的设计可让用户清楚地知道哪些类型的信息可能会提供给相应应用。在安装过程中，第三方应用可能会请求获得访问这些资源的权限。获得授权后，应用便可以进行安装，并且只要安装在设备上，便会一直有权访问请求的数据。

默认情况下，收集个人信息的所有应用都会仅限特定应用访问这些数据。如果某个应用选择通过 IPC 将数据提供给其他应用，那么这个授予访问权限的应用便可以限制由操作系统强制执行的 IPC 机制的权限。

敏感数据输入设备

Android 设备经常会提供可让应用与周围环境进行互动的敏感数据输入设备（例如，摄像头、麦克风或 GPS）。对于要使用这些设备的第三方应用，必须先由用户通过使用 Android 操作系统权限向其明确提供使用权限。安装应用时，安装程序会以提供名称的方式请求用户授予使用相应传感器的权限。

如果某个应用想要知道用户所在的位置，则需要获得获取用户位置信息的权限。安装应用时，安装程序会询问用户是否允许相应应用获取用户的位置信息。如果用户不希望任何应用获取其位置信息，可以随时运行“设置”应用，转到“位置和安全”，然后取消选中“使用无线网络”和“启用 GPS 卫星”。这将针对用户设备上的所有应用停用需要使用位置信息的服务。

设备元数据

Android 还会尽力限制访问本身并不属于敏感数据，但可能会间接透露用户特征、用户偏好以及用户使用设备的方式的数据。

默认情况下，应用无权访问操作系统日志、浏览器历史记录、电话号码以及硬件/网络标识信息。如果应用在安装时请求获得访问此类信息的权限，安装程序会询问用户是否允许相应应用访问此类信息。如果用户没有授予该权限，系统将不会安装相应应用。

证书授权中心

Android 中收录了一组已安装的系统证书授权中心，这些授权中心在整个系统范围内均可信。在 Android 7.0 之前的版本中，设备制造商可以修改其设备上搭载的 CA 组。不过，运行 7.0 及更高版本的设备将具有一组统一的系统 CA，并且不再允许设备制造商对其进行修改。

要作为新的公共 CA 添加到 Android 收录的 CA 组中，相应 CA 必须要完成 Mozilla CA 收录流程，然后提交一项针对 Android 的功能请求 (https://code.google.com/p/android/issues/entry)，以便请求添加到 Android 开放源代码项目 (AOSP) 收录的 Android CA 组中。

此外还有一些设备专用 CA，这些 CA 不应被收录到 AOSP CA 核心组中，例如，安全访问运营商基础架构组件（例如，短信/彩信网关）时可能需要的运营商私有 CA。建议设备制造商将私有 CA 仅收录在需要信任这些 CA 的组件/应用中。如需更多详细信息，请参阅网络安全配置。

应用签名

通过代码签名，开发者可以标识应用创作者并更新其应用，而无需创建复杂的接口和权限。在 Android 平台上运行的每个应用都必须要有开发者的签名。Google Play 或 Android 设备上的软件包安装程序会拒绝没有获得签名就尝试安装的应用。

在 Google Play 上，应用签名可以将 Google 对开发者的信任和开发者对自己的应用的信任联系在一起。开发者知道自己的应用是以未经修改的形式提供给 Android 设备的，并且开发者可以对自己的应用的行为负责。

在 Android 上，应用签名是将应用放入其应用沙盒的第一步。已签名的应用证书定义了哪个用户 ID 与哪个应用相关联；不同的应用要以不同的用户 ID 运行。应用签名可确保一个应用无法访问任何其他应用，通过明确定义的 IPC 进行访问时除外。

当应用（APK 文件）安装到 Android 设备上时，软件包管理器会验证 APK 是否已经过适当签名（已使用 APK 中包含的证书签名）。如果该证书（或更准确地说，证书中的公钥）与为设备上的任何其他 APK 签名时使用的密钥一致，那么这个新 APK 可以选择在清单中指定它将与其他以类似方式签名的 APK 共用一个 UID。

应用可以由第三方（原始设备制造商(OEM)、运营商、其他相关方）签名，也可以自行签名。Android 提供了使用自签名证书进行代码签名的功能，而开发者无需外部协助或许可即可生成自签名证书。应用并非必须由核心机构签名。Android 目前不对应用证书进行 CA 认证。

应用还可以在“签名”保护级别声明安全权限，以便仅限使用同一个密钥签名的应用访问它们，同时维持单独的 UID 和应用沙盒。通过共用 UID 功能，可以与共用的应用沙盒建立更紧密的联系，这是因为借助该功能，使用同一个开发者密钥签名的两个或更多应用可以在其清单中声明共用的 UID。

应用验证

Android 4.2 及更高版本均支持应用验证。用户可以选择启用“验证应用”，并在安装应用之前由应用验证程序对其进行评估。如果用户尝试安装的应用可能有害，应用验证功能可以提醒用户；如果应用的危害性非常大，应用验证功能可以阻止安装。

数字版权管理

Android 平台提供了一个可扩展的 DRM 框架，以便应用根据与受版权保护的内容相关的许可限制条件来管理这些内容。DRM 框架支持多种 DRM 方案；设备具体支持哪些 DRM 方案由设备制造商决定。

Android DRM 框架是在以下两个架构层中实现的（请参见下图）：

DRM 框架 API：通过 Android 应用框架提供给应用，并通过适用于标准应用的 Dalvik VM 运行。

本机代码 DRM 管理器：用于实现 DRM 框架，并为 DRM 插件（代理）提供接口，以便处理各种 DRM 方案的版权管理和解密操作。

2. Android应用软件基础

2.1. Android开发基本流程

\*一个Android应用软件开发的5个主要阶段，分别是：开发环境配置、编写代码、构建和运行、迭代（调试、测试和优化）和发布应用。

Android官方推荐的编写代码环境是Android Studio。Android Studio 是基于 IntelliJ IDEA 的官方 Android 应用开发集成开发环境 (IDE)。 除了 IntelliJ 强大的代码编辑器和开发者工具，Android Studio 提供了更多可提高 Android 应用构建效率的功能，例如：

基于 Gradle 的灵活构建系统

快速且功能丰富的模拟器

可针对所有 Android 设备进行开发的统一环境

Instant Run，可将变更推送到正在运行的应用，无需构建新的 APK

可帮助开发者构建常用应用功能和导入示例代码的代码模板和 GitHub 集成

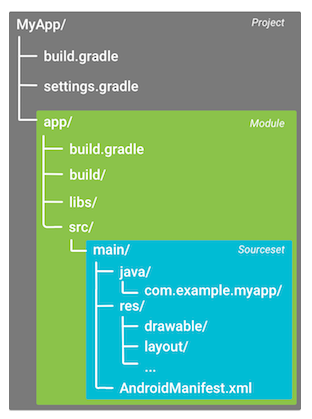
丰富的测试工具和框架

可捕捉性能、易用性、版本兼容性以及其他问题的 Lint 工具

C++ 和 NDK 支持

内置对 Google 云端平台的支持，可轻松集成 Google Cloud Messaging 和 App 引擎

使用Android Studio创建的一个新Android项目时的典型目录结构如下图所示：



settings.gradle 文件位于项目根目录，用于指示 Gradle 在构建应用时应将哪些模块包括在内。对大多数项目而言，该文件很简单，只包括以下内容：

include ‘:app’

不过，多模块项目需要指定应包括在最终构建之中的每个模块。

模块级 build.gradle 文件位于每个 // 目录，用于配置适用于其所在模块的构建设置

Gradle 还包括两个属性文件，位于项目根目录，可用于指定适用于 Gradle 构建工具包本身的设置：

gradle.properties，可以在其中配置项目范围 Gradle 设置，例如 Gradle 后台进程的最大堆大小。如需了解详细信息，请参阅构建环境；

local.properties，为构建系统配置本地环境属性，例如 SDK 安装路径。由于该文件的内容由 Android Studio 自动生成并且专用于本地开发者环境，因此开发者不应手动修改该文件，或将其纳入版本控制系统。

Android Studio 按逻辑关系将每个模块的源代码和资源分组为源集（source sets）。模块的 main/ 源集包括其所有构建变体共用的代码和资源，其他源集目录为可选项（配置新的构建变体时，Android Studio 不会自动创建这些目录）。

构建和运行、发布应用

在完成了代码的编写之后，Android 构建系统编译应用资源和源代码，然后将它们打包成可供测试、部署、签署和分发的 APK。Android Studio 使用 Gradle 这一高级构建工具包来自动化执行和管理构建流程，同时也允许开发者定义灵活的自定义构建配置。每个构建配置均可自行定义一组代码和资源，同时对所有应用版本共有的部分加以重复利用。Android Plugin for Gradle 与这个构建工具包协作，共同提供专用于构建和测试 Android 应用的流程和可配置设置。

Gradle 和 Android 插件独立于 Android Studio 运行。这意味着，开发者可以在 Android Studio 内、使用计算机上的命令行工具或在未安装 Android Studio 的计算机（例如持续性集成服务器）上构建 Android 应用。如果不使用 Android Studio，可以从命令行构建和运行应用。无论是从命令行、在远程计算机上还是使用 Android Studio 构建项目，构建的输出都相同。

典型 Android 应用模块的构建流程通常依循下列步骤：

1、编译器将源代码转换成 DEX（Dalvik Executable) 文件（其中包括运行在 Android 设备上的字节码），将所有其他内容转换成已编译资源。

2、APK 打包器将 DEX 文件和已编译资源合并成单个 APK。不过，必须先签署 APK，才能将应用安装并部署到 Android 设备上。

3、APK 打包器使用调试或发布密钥库签署 APK：

如果构建的是调试版本的应用（即专用于测试和分析的应用），打包器会使用调试密钥库签署应用。Android Studio 自动使用调试密钥库配置新项目。

如果构建的是打算向外发布的发布版本应用，打包器会使用发布密钥库签署应用。要创建发布密钥库，请阅读在 Android Studio 中签署您的应用。

4、在生成最终 APK 之前，打包器会使用 zipalign 工具对应用进行优化，减少其在设备上运行时的内存占用。

构建流程结束时，我们将获得可用来进行部署、测试的调试 APK，或者可用来发布给外部用户的发布 APK。

### **2.2. Android应用基础**

[Android 应用采用 Java 编程语言编写](https://developer.android.com/guide/components/fundamentals.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。Android SDK 工具将源代码连同所有数据和资源文件编译到一个 APK：***Android 软件包***，即带有 .apk 后缀的存档文件中。一个 APK 文件包含 Android 应用的所有内容，它是基于 Android 系统的设备用来安装应用的文件。

安装到设备后，每个 Android 应用都运行在自己的安全沙箱内：

* Android 操作系统是一种多用户 Linux 系统，其中的每个应用都是一个不同的用户；
* 默认情况下，系统会为每个应用分配一个唯一的 Linux 用户 ID（该 ID 仅由系统使用，应用并不知晓）。系统为应用中的所有文件设置权限，使得只有分配给该应用的用户 ID 才能访问这些文件；
* 每个进程都具有自己的虚拟机 (VM)，因此应用代码是在与其他应用隔离的环境中运行；
* 默认情况下，每个应用都在其自己的 Linux 进程内运行。Android 会在需要执行任何应用组件时启动该进程，然后在不再需要该进程或系统必须为其他应用恢复内存时关闭该进程。

Android 系统可以通过这种方式实现最小权限原则。也就是说，默认情况下，每个应用都只能访问执行其工作所需的组件，而不能访问其他组件。 这样便营造出一个非常安全的环境，在这个环境中，应用无法访问系统中其未获得权限的部分。

不过，应用仍然可以通过一些途径与其他应用共享数据以及访问系统服务：

* 可以安排两个应用共享同一 Linux 用户 ID，在这种情况下，它们能够相互访问彼此的文件。 为了节省系统资源，可以安排具有相同用户 ID 的应用在同一 Linux 进程中运行，并共享同一 VM（应用还必须使用相同的证书签署）。
* 应用可以请求访问设备数据（如用户的联系人、短信、可装载存储装置 [SD 卡]、相机、蓝牙等）的权限。 用户必须明确授予这些权限。 如需了解详细信息，请参阅 [使用系统权限](https://developer.android.com/training/permissions/index.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。

#### **2.2.1. 应用组件**

应用组件是 Android 应用的基本构建基块。每个组件都是一个不同的点，系统可以通过它进入具体应用。 并非所有组件都是用户的实际入口点，有些组件相互依赖，但每个组件都以独立实体形式存在，并发挥特定作用。每个组件都是唯一的构建基块，有助于定义应用的总体行为。

共有四种不同的应用组件类型。每种类型都服务于不同的目的，并且具有定义组件的创建和销毁方式的不同生命周期。

以下便是这四种应用组件类型：

Activity（活动）

Activity 表示具有用户界面的单一屏幕。例如，电子邮件应用可能具有一个显示新电子邮件列表的 Activity、一个用于撰写电子邮件的 Activity 以及一个用于阅读电子邮件的 Activity。 尽管这些 Activity 通过协作在电子邮件应用中形成了一种紧密结合的用户体验，但每一个 Activity 都独立于其他 Activity 而存在。 因此，其他应用可以启动其中任何一个 Activity（如果电子邮件应用允许）。 例如，相机应用可以启动电子邮件应用内用于撰写新电子邮件的 Activity，以便用户共享图片。

Activity 作为 [Activity](https://developer.android.com/reference/android/app/Activity.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 的子类实现，开发者可以在 [Activity 开发者指南](https://developer.android.com/guide/components/activities.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 中了解有关它的更多详情。

Service（服务）

服务是一种在后台运行的组件，用于执行长时间运行的操作或为远程进程执行作业。服务不提供用户界面。 例如，当用户位于其他应用中时，服务可能在后台播放音乐或者通过网络获取数据，但不会阻断用户与 Activity 的交互。 诸如 Activity 等其他组件可以启动服务，让其运行或与其绑定以便与其进行交互。

服务作为 [Service](https://developer.android.com/reference/android/app/Service.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 的子类实现，开发者可以在 [服务开发者指南](https://developer.android.com/guide/components/services.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 中了解有关它的更多详情。

Content Provider（内容提供者）

内容提供者管理一组共享的应用数据。开发者可以将数据存储在文件系统、SQLite 数据库、网络上或当前应用可以访问的任何其他持久化存储位置。 其他应用可以通过内容提供者查询数据，甚至修改数据（如果内容提供者允许）。 例如，Android 系统可提供管理用户联系人信息的内容提供者。 因此，任何具有适当权限的应用都可以查询内容提供者的某一部分（如 [ContactsContract.Data](https://developer.android.com/reference/android/provider/ContactsContract.Data.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) ），以读取和写入有关特定人员的信息。

内容提供者也适用于读取和写入当前应用不共享的私有数据。 例如，记事本示例应用使用内容提供者来保存笔记。

内容提供者作为 [ContentProvider](https://developer.android.com/reference/android/content/ContentProvider.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 的子类实现，并且必须实现让其他应用能够执行事务的一组标准 API。 如需了解详细信息，请参阅[内容提供者开发者指南](https://developer.android.com/guide/topics/providers/content-providers.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。

Broadcast Receiver（广播接收器）

广播接收器是一种用于响应系统范围广播通知的组件。 许多广播都是由系统发起的。例如，通知屏幕已关闭、电池电量不足或已拍摄照片的广播。应用也可以发起广播。例如，通知其他应用某些数据已下载至设备，并且可供其使用。 尽管广播接收器不会显示用户界面，但它们可以[创建状态栏通知](https://developer.android.com/guide/topics/ui/notifiers/notifications.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)，在发生广播事件时提醒用户。 但广播接收器更常见的用途只是作为通向其他组件的“通道”，设计用于执行极少量的工作。 例如，它可能会基于事件发起一项服务来执行某项工作。

广播接收器作为 [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 的子类实现，并且每条广播都作为 [Intent](https://developer.android.com/reference/android/content/Intent.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 对象进行传递。 如需了解详细信息，请参阅 [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 类。

Android 系统设计的独特之处在于，****任何应用都可以启动其他应用的组件****。 例如，如果开发者想让用户使用设备的相机拍摄照片，很可能有另一个应用可以执行该操作，那么当前应用就可以利用该应用，而不是开发一个 Activity 来自行拍摄照片。 开发者不需要集成甚至链接到该相机应用的代码，而是只需启动拍摄照片的相机应用中的 Activity。 完成拍摄时，系统甚至会将照片返回当前（调用者）应用，以便（调用者）使用。对用户而言，就好像相机真正是当前应用的组成部分。

当系统启动某个组件时，会启动该应用的进程（如果尚未运行），并实例化该组件所需的类。 例如，如果当前应用启动相机应用中拍摄照片的 Activity，则该 Activity 会在属于相机应用的进程，而不是当前应用的进程中运行。因此，与大多数其他系统上的应用不同，****Android 应用并没有单一入口点****（例如，没有 main() 函数）。

由于系统在单独的进程中运行每个应用，且其文件权限会限制对其他应用的访问，因此当前应用无法直接启动其他应用中的组件，但 Android 系统却可以。因此，要想启动其他应用中的组件，当前应用必须向系统传递一则消息，说明当前应用想启动特定组件的 Intent。 系统随后便会为当前应用启动该组件。

四种组件类型中的三种：Activity、服务和广播接收器通过名为 Intent 的异步消息进行启动。Intent 会在运行时将各个组件相互绑定（可以类比为在不同组件间传递消息的信使），无论组件属于当前应用还是其他应用。

Intent 使用 [Intent](https://developer.android.com/reference/android/content/Intent.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 对象创建，它定义的消息用于启动特定组件或特定类型的组件。Intent 对象携带了 Android 系统用来确定要启动哪个组件的信息（例如，准确的组件名称或应当接收该 Intent 的组件类别），以及收件人组件为了正确执行操作而使用的信息（例如，要采取的操作以及要处理的数据）。Intent 可以是显式的，也可以是隐式的。

* 显式（Explicit）Intent：按名称（完全限定类名）指定要启动的组件。 通常，开发者会在自己的应用中使用显式 Intent 来启动组件，这是因为开发者知道要启动的 Activity 或服务的类名。例如，启动新 Activity 以响应用户操作，或者启动服务以在后台下载文件。
* 隐式（Implicit）Intent：不会指定特定的组件，而是声明要执行的常规操作，从而允许其他应用中的组件来处理它。创建隐式 Intent 时，Android 系统通过将 Intent 的内容与在设备上其他应用的清单文件中声明的 [Intent 过滤器](https://developer.android.com/reference/android/content/IntentFilter.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)进行比较，从而找到要启动的相应组件。 如果 Intent 与 Intent 过滤器匹配，则系统将启动该组件，并向其传递 Intent 对象。 如果多个 Intent 过滤器兼容，则系统会显示一个****对话框****，****支持用户选取要使用的应用****。Intent 过滤器是应用清单文件中的一个表达式，它指定该组件要接收的 Intent 类型。 例如，****通过为 Activity 声明 Intent 过滤器，开发者可以使其他应用能够直接使用某一特定类型的 Intent 启动 Activity****。同样，如果开发者没有为 Activity 声明任何 Intent 过滤器，则 Activity 只能通过显式 Intent 启动。隐式 Intent 指定能够在可以执行相应操作的设备上调用任何应用的操作。 如果当前应用无法执行该操作而其他应用可以，且开发者希望用户选取要使用的应用，则使用隐式 Intent 非常有用。

// 显式 Intent 实例// Executed in an Activity, so 'this' is the Context// The fileUrl is a string URL, such as "http://www.example.com/image.png"

Intent downloadIntent = new Intent(this, DownloadService.class);

downloadIntent.setData(Uri.parse(fileUrl));

startService(downloadIntent);

// 隐式 Intent 实例// Create the text message with a string

Intent sendIntent = new Intent();

sendIntent.setAction(Intent.ACTION\_SEND);

sendIntent.putExtra(Intent.EXTRA\_TEXT, textMessage);

sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activityif (sendIntent.resolveActivity(getPackageManager()) != null) {

startActivity(sendIntent);

}

对于 Activity 和服务， Intent 定义要执行的操作（例如，“查看”或“发送”某个内容），并且可以指定要执行操作的数据的 URI（以及正在启动的组件可能需要了解的信息）。 例如， Intent 传达的请求可以是启动一个显示图像或打开网页的 Activity。 在某些情况下，应用可以启动 Activity 来接收结果，在这种情况下，Activity 也会在 [Intent](https://developer.android.com/reference/android/content/Intent.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 中返回结果（例如，可以发出一个 Intent，让用户选取某位联系人并将其返回给包括指向所选联系人的 URI）。

为了确保应用的安全性，[Android开发者官网上的Intent 和 Intent 过滤器使用指南](https://developer.android.com/guide/components/intents-filters.html?hl=zh-cn" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)建议：启动 Service 时，应始终使用显式 Intent，且不要为服务声明 Intent 过滤器。使用隐式 Intent 启动服务存在安全隐患，因为开发者无法确定哪些服务将响应 Intent，且用户无法看到哪些服务已启动。从 Android 5.0（API 级别 21）开始，如果使用隐式 Intent 调用 bindService()，系统会引发异常。

对于广播接收器， Intent 只会定义要广播的通知（例如，指示设备电池电量不足的广播只包括指示“电池电量不足”的已知操作字符串）。

Intent 不会启动另一个组件类型：内容提供者，后者会在成为 ContentResolver 的请求目标时启动。 内容解析程序通过内容提供者处理所有直接事务，使得通过提供程序执行事务的组件可以无需执行事务，而是改为在 ContentResolver 对象上调用方法。 这会在内容提供者与请求信息的组件之间留出一个抽象层（以确保安全）。

每种类型的组件有不同的启动方法：

* 开发者可以通过将 Intent 传递到 startActivity() 或 startActivityForResult()（当前应用想让 Activity 返回结果时）来启动 Activity（或为其安排新任务）。
* 开发者可以通过将　Intent 传递到 startService() 来启动服务（或对执行中的服务下达新指令）。 或者，也可以通过将 Intent 传递到 bindService() 来绑定到该服务。
* 开发者可以通过将 Intent 传递到 sendBroadcast()、sendOrderedBroadcast() 或 sendStickyBroadcast() 等方法来发起广播；
* 开发者可以通过在 ContentResolver 上调用 query() 来对内容提供者执行查询。

如需了解有关 Intent 用法的详细信息，请参阅 [Intent 和 Intent 过滤器](https://developer.android.com/guide/components/intents-filters.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)文档。 以下文档中还提供了有关启动特定组件的详细信息： [Activity](https://developer.android.com/guide/components/activities.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)、[服务](https://developer.android.com/guide/components/services.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)、[BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 和[内容提供者](https://developer.android.com/guide/topics/providers/content-providers.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。

#### **2.2.2. AndroidManifest.xml**

在 Android 系统启动应用组件之前，系统必须通过读取应用的 AndroidManifest.xml 文件（“清单”文件）确认组件存在。 当前应用必须在此文件中声明其所有组件，每个应用的根目录中都必须包含一个 [AndroidManifest.xml](https://developer.android.com/guide/topics/manifest/manifest-intro.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 文件（且文件名精确无误）。 清单文件向 Android 系统提供应用的必要信息，系统必须具有这些信息方可运行应用的任何代码。

此外，清单文件还可执行以下操作：

* 为应用的 Java 软件包命名。软件包名称充当应用的****唯一标识****符。
* 描述应用的各个组件，包括构成应用的 Activity、服务、广播接收器和内容提供程序。它还为实现每个组件的类命名并发布其功能，例如它们可以处理的 [Intent](https://developer.android.com/reference/android/content/Intent.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 消息。这些声明向 Android 系统告知有关组件以及可以启动这些组件的条件的信息。
* 确定托管应用组件的进程。
* 声明应用必须具备哪些权限才能访问 API 中受保护的部分并与其他应用交互。还声明其他应用与该应用组件交互所需具备的权限
* 列出 [Instrumentation](https://developer.android.com/reference/android/app/Instrumentation.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)（插桩） 类，这些类可在应用运行时提供分析和其他信息。这些声明只会在应用处于开发阶段时出现在清单中，在应用发布之前将移除。
* 声明应用所需的最低 Android API 级别
* 列出应用必须链接到的库

#### **2.2.3. 应用资源**

Android 应用并非只包含代码，它还需要与源代码分离的资源，如图像、音频文件以及任何与应用的视觉呈现有关的内容。 例如，开发者应该通过 XML 文件定义 Activity 用户界面的动画、菜单、样式、颜色和布局。使用应用资源能够在不修改代码的情况下轻松地更新应用的各种特性，并可通过提供备用资源集让开发者能够针对各种设备配置（如不同的语言和屏幕尺寸）优化应用。

对于 Android 项目中包括的每一项资源，SDK 构建工具都会定义一个唯一的整型 ID，开发者可以利用它来引用应用代码或 XML 中定义的其他资源中的资源。 例如，如果当前应用包含一个名为 logo.png 的图像文件（保存在 res/drawable/ 目录中），则 SDK 工具会生成一个名为 R.drawable.logo 的资源 ID，可以利用它来引用该图像并将其插入到当前用户界面。

提供与源代码分离的资源的其中一个最重要优点在于，开发者可以提供针对不同设备配置的备用资源。例如，通过在 XML 中定义 UI 字符串，开发者可以将字符串翻译为其他语言，并将这些字符串保存在单独的文件中。 然后，Android 系统会根据向资源目录名称追加的语言限定符（如为法语字符串值追加 res/values-fr/）和用户的语言设置，对当前 UI 应用相应的语言字符串。

Android 支持许多不同的备用资源限定符。限定符是一种加入到资源目录名称中，用来定义这些资源适用的设备配置的简短字符串。 再举一例，开发者应该经常会根据设备的屏幕方向和尺寸为 Activity 创建不同的布局。例如，当设备屏幕为纵向（长型）时，可能想要一种垂直排列按钮的布局；但当屏幕为横向（宽型）时，应按水平方向排列按钮。 要想根据方向更改布局，开发者可以定义两种不同的布局，然后对每个布局的目录名称应用相应的限定符。然后，系统会根据当前设备方向自动应用相应的布局。

如需了解有关可以在应用中包括的不同资源类型以及如何针对不同设备配置创建备用资源的详细信息，请阅读[提供资源](https://developer.android.com/guide/topics/resources/providing-resources.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。

### **2.3. APK签名与分发（安装）**

Android 要求所有 APK 必须先使用证书进行数字签署，然后才能安装。公钥证书（也称为数字证书或身份证书）包含公钥/私钥对的公钥，以及可以标识密钥所有者的一些其他元数据（例如名称和位置）。证书的所有者持有对应的私钥。

在开发者签署 APK 时，签署工具会将公钥证书附加到 APK。公钥证书充当“指纹”，用于将 APK 唯一关联到开发者及开发者的对应私钥。这有助于 Android 确保 APK 开发者的任何将来更新都是原版更新并来自原始作者。用于创建此证书的密钥称为 ****应用签名密钥****。

密钥库是一种包含一个或多个私钥的二进制文件。每个应用在其整个生命周期内必须使用相同证书，以便用户能够以应用更新的形式安装新版本。

从 IDE 中运行或调试项目时，Android Studio 将自动使用通过 Android SDK 工具生成的调试证书签署当前项目对应的 APK。当在 Android Studio 中首次运行或调试项目时，IDE 将自动在 $HOME/.android/debug.keystore 中创建调试密钥库和证书，并设置密钥库和密钥密码。

由于调试证书通过构建工具创建并且在设计上不安全，大多数应用商店（包括 Google Play 商店）都不接受使用调试证书签署要发布的 APK。

Android Studio 会自动将开发者的调试签署信息存储在签署配置中，因此开发者不必在每次调试时都输入此信息。签署配置是一种包含签署 APK 所需全部必要信息的对象，这些信息包括密钥库位置、密钥库密码、密钥名称和密钥密码。

由于应用签名密钥用于验证应用开发者的应用所有权身份，并确保为用户进行无缝而安全的更新，因此，管理和保护开发者的发布密钥对于开发者和他的用户而言都非常重要。开发者可以选择使用 Google Play 应用签名以利用 Google 的基础架构安全地管理和存储开发者的应用签名密钥，也可以选择自行管理和保护自己的密钥库和应用签名密钥。

关于具体的APK文件签名操作方法，请参阅[Android开发者官网上的应用签署指南](https://developer.android.com/studio/publish/app-signing.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。

#### **2.3.1. APK 签名方案v2**

[APK 签名方案 v2](https://source.android.com/security/apksigning/v2" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 是一种****全文件签名方案****，该方案能够发现对 APK 的受保护部分进行的所有更改，从而有助于加快验证速度并增强完整性保证。

使用 APK 签名方案 v2 进行签名时，会在 APK 文件中插入一个 APK 签名分块，该分块位于“ZIP 中央目录”部分之前并紧邻该部分。在“APK 签名分块”内，v2 签名和签名者身份信息会存储在 APK 签名方案 v2 分块中。



APK 签名方案 v2 是在 Android 7.0 (Nougat, API Level 24.0.3) 中引入的。为了使 APK 可在 Android 6.0 (Marshmallow) 及更低版本的设备上安装，应先使用 [JAR 签名](https://source.android.com/security/apksigning/" \l "v1" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)功能对 APK 进行签名，然后再使用 v2 方案对其进行签名。为了最大限度地提高兼容性，应同时采用 v1 和 v2 这两种方案对应用进行签名。与只通过 v1 方案签名的应用相比，通过 v2 方案签名的应用能够更快速地安装到 Android Nougat 及更高版本的设备上。更低版本的 Android 平台会忽略 v2 签名，这就需要应用包含 v1 签名。

v1 签名不保护 APK 的某些部分，例如 ZIP 元数据。APK 验证程序需要处理大量不可信（尚未经过验证）的数据结构，然后会舍弃不受签名保护的数据。这会导致相当大的受攻击面。此外，APK 验证程序必须解压所有已压缩的条目，而这需要花费更多时间和内存。为了解决这些问题，Android 7.0 中引入了 APK 签名方案 v2。

为了保持与当前 APK 格式向后兼容，v2 及更高版本的 APK 签名会存储在“APK 签名分块”（APK Signing Block）内，该分块是为了支持 APK 签名方案 v2 而引入的一个新容器。在 APK 文件中，“APK 签名分块”位于“ZIP 中央目录”（位于文件末尾）之前并紧邻该部分。

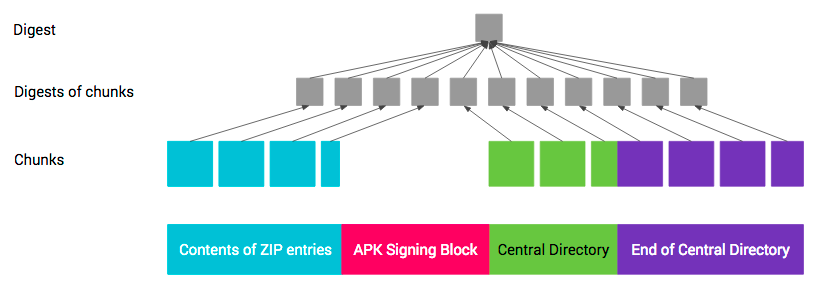
为了保护 APK 内容，APK 包含以下 4 个部分：

* ZIP 条目的内容（从偏移量 0 处开始一直到“APK 签名分块”的起始位置）
* APK 签名分块
* ZIP 中央目录
* ZIP 中央目录结尾

APK 签名方案 v2 负责保护第 1、3、4 部分的完整性，以及第 2 部分包含的“APK 签名方案 v2 分块”中的 signed data 分块的完整性。

第 1、3 和 4 部分的完整性通过其内容的一个或多个摘要来保护，这些摘要存储在 signed data 分块中，而这些分块则通过一个或多个签名来保护。

第 1、3 和 4 部分的摘要采用以下计算方式，类似于两级 [Merkle 树](https://en.wikipedia.org/wiki/Merkle_tree" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。每个部分都会被拆分成多个大小为 1 MB（220 个字节）的连续块。每个部分的最后一个块可能会短一些。每个块的摘要均通过字节 0xa5 的连接、块的长度（采用小端字节序的 uint32 值，以字节数计）和块的内容进行计算。顶级摘要通过字节 0x5a 的连接、块数（采用小端字节序的 uint32 值）以及块的摘要的连接（按照块在 APK 中显示的顺序）进行计算。摘要以分块方式计算，以便通过并行处理来加快计算速度。



由于第 4 部分（ZIP 中央目录结尾）包含“ZIP 中央目录”的偏移量，因此该部分的保护比较复杂。当“APK 签名分块”的大小发生变化（例如，添加了新签名）时，偏移量也会随之改变。因此，在通过“ZIP 中央目录结尾”计算摘要时，必须将包含“ZIP 中央目录”偏移量的字段视为包含“APK 签名分块”的偏移量。

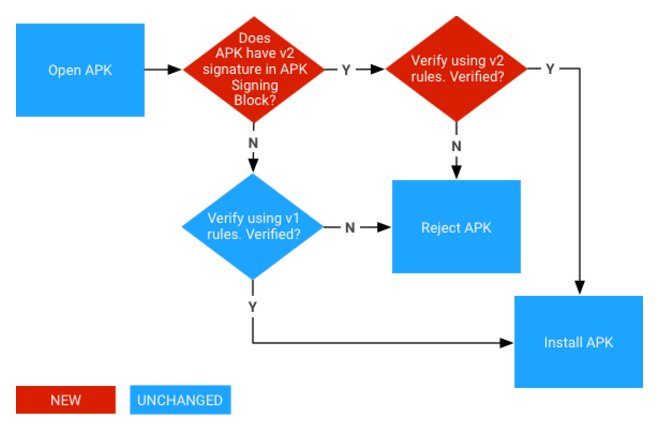
#### **2.3.2. APK 签名方案的防回滚保护**

攻击者可能会试图在支持对带 v2 签名的 APK 进行验证的 Android 平台上将带 v2 签名的 APK 作为带 v1 签名的 APK 进行验证。为了防范此类攻击，带 v2 签名的 APK 如果还带 v1 签名，其 META-INF/*.SF 文件的主要部分中必须包含 X-Android-APK-Signed 属性。该属性的值是一组以英文逗号分隔的 APK 签名方案 ID（v2 方案的 ID 为 2）。在验证 v1 签名时，对于此组中验证程序首选的 APK 签名方案（例如，v2 方案），如果 APK 没有相应的签名，APK 验证程序必须要拒绝这些 APK。此项保护依赖于内容 META-INF/*.SF 文件受 v1 签名保护这一事实。请参阅 [JAR 已签名的 APK 的验证部分](https://source.android.com/security/apksigning/v2" \l "v1-verification" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。

攻击者可能会试图从“APK 签名方案 v2 分块”中删除安全系数较高的签名。为了防范此类攻击，对 APK 进行签名时使用的签名算法 ID 的列表会存储在通过各个签名保护的 signed data 分块中。

#### **2.3.3. APK 签名的验证**

在 Android 7.0 中，可以根据 APK 签名方案 v2（v2 方案）或 JAR 签名（v1 方案）验证 APK。更低版本的平台会忽略 v2 签名，仅验证 v1 签名。



如上图所示就是一个APK 签名验证过程，新步骤以红色显示。

#### **2.3.4. APK 安装**

Android应用安装有如下四种方式：

* ****系统应用安装****：开机时加载系统的APK和应用，没有安装界面；
* ****网络下载应用安装****：通过各种market应用完成，没有安装界面；
* ****ADB工具安装****：即通过Android的SDK开发tools里面的adb.exe程序安装，没有安装界面；
* ****第三方应用安装****：通过SD卡里的APK文件安装(比如双击APK文件触发)，有安装界面，系统默认已经安装了一个安装卸载应用的程序，即由[packageinstaller.apk](https://developer.android.com/reference/android/content/pm/PackageInstaller.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)应用处理安装及卸载过程的界面，分别是：PackageInstallerActivity 和 UninstallerActivity 。这两个Activity可以通过发送Intent的方式启动。

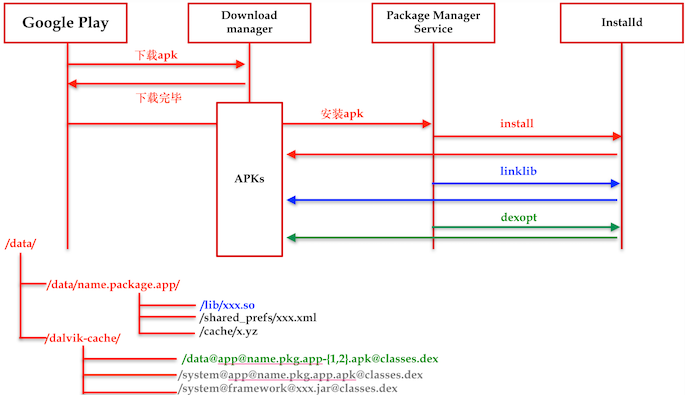
APK安装过程可能会涉及到以下目录：

* /system/app ：系统自带的应用程序，获得adb root权限才能删除
* /data/app ：用户程序安装的目录。安装时把apk文件复制到此目录
* /data/data ：存放应用程序的数据
* /data/dalvik-cache ：将apk中的dex文件安装到dalvik-cache目录下(dex文件是dalvik虚拟机的可执行文件,当然，ART–Android Runtime的可执行文件格式为oat，启用ART时，系统会执行dex文件转换至oat文件)
* /data/system ：该目录下的packages.xml文件，类似于Windows的注册表，这个文件是在解析apk时由writeLP()创建的，里面记录了系统的权限列表，以及每个APK的name， codePath， flags， ts， version， uesrid等信息，这些信息主要通APK的 AndroidManifest.xml 解析获取，解析完APK后将更新信息写入这个文件并保存到内部存储，下次开机直接从里面读取相关信息添加到内存相关列表中。当有APK升级，安装或删除时会更新这个文件。packages.list文件中保存着应用的包名，uid，所属的用户，以及 home 目录.因为Android中每一个应用都类似Linux系统中用户的概念，它们都有自己的 home 目录，用来保存应用特定的数据。

安卓系统 ****安装**** 一个应用时，系统大致会进行如下操作：

* 复制APK安装包到 /data/app 目录下，文件名会以应用的package命名；
* 解压并扫描安装包，把dex文件(Dalvik字节码)保存到dalvik-cache目录(一般情况下，会先执行dexopt即dex文件的优化，将优化后的dex文件保存至该目录下)，并在 /data/data 目录下创建对应的应用数据目录( /data/data 目录可读可写)；
* 更新 /data/system/packages.xml 中的内容，将APK的信息加入进去。

如下图所示，总结了一个典型Android应用安装流程和应用安装涉及到的主要目录。



上图中，PackageManagerService（简记为PMS）由system\_server启动，它全面负责应用包的安装，卸载，权限检查等工作。在每次开机的时候，PMS 都会在其构造函数中，对指定的目录的APK进行扫描。对于没有安装的APK文件会触发安装过程。PMS 在扫描过程中会给 installd 发送消息，请求创建应用程序的数据目录和执行代码优化。在应用程序的管理工作中，有时候需要对存储设备做一些操作，比如创建目录修改目录权限等，这些操作有的是需要特权级的权限。PMS 存活在system\_server进程中，这个进程的用户为 system，它是没有特权级的权限的，出于最小化授权和进程隔离的角度考虑，Android单独将一部分需要特权的工作，转交给 installd 进程去完成。installd 也算是个服务端了，它是通过socket来接收客户端的消息，然后进行处理。

对照安装过程，****卸载**** 过程与之相逆：删除安装过程中在上述目录下创建的文件及目录。

#### **2.3.5. 免安装应用（Instant Apps）**

Android 免安装应用使原生 Android 应用能够在启动网址时运行，无需安装应用。 当 ****Google Play**** 商店收到与 免安装应用匹配的网址请求时，它将必需的代码文件发送到发送该请求的 Android 设备。 然后，该设备将运行此应用。

注：Android 免安装应用仅在运行 Android 5.0（API level 21）或更高版本的 Android 设备上，且系统必须内置Google Play服务才可以正常工作。

应用即功能

Android 免安装应用为开发者构建应用和用户使用应用提供了一种新颖而独特的方式。 在介绍 Android 免安装应用的核心概念之前，了解一些基本术语很有用处。

基本而言，应用至少有一项功能（即应用执行的操作）：例如，查找地图上的位置、发送电子邮件或读取每日新闻。 许多应用都提供多种功能。 例如，除查找地图上的位置外，地图应用可能还允许用户查找附近的餐馆或通过电子邮件发送位置链接。

每一个此类操作（查找位置、查找附近的餐馆、分享位置链接）都是地图应用中的一项功能。

借助 Android 免安装应用，用户可以从 Google Play 商店下载某个应用的某项功能，而无需下载和安装整个应用。 当用户从免安装应用请求一项功能时，他们仅收到运行该特定功能所必需的代码，不多也不少。 在用户使用完此功能后，系统可以丢弃该功能的代码。

回到前面的地图应用示例，该地图免安装应用可将其每项功能对外显示为该应用中的某个分散实体。

用户可以只下载和使用位置查找器功能，也可以只下载和使用餐馆指南功能或分享功能。 在用户切换到另一个应用后，系统可以安全地移除此功能的代码。

免安装应用中的每项功能应至少有一个可充当该功能 ***入口点*** 的 Activity 类。 入口点 Activity 可托管此功能的 UI，并定义整个用户流。 当用户在其设备上启动此功能时，他们首先看到的是入口点 Activity。 一项功能可以有多个入口点 Activity，但它只需要一个。

功能模块和功能 APK

要提供按需功能下载，开发者需要将应用分解成较小的模块，并将它们打包成功能模块。

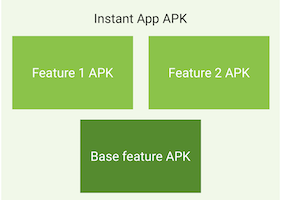
在构建免安装应用项目时，构建输出是一个包含一个或多个功能 APK 的免安装应用 APK。 每项功能 APK 都是从项目中的功能模块构建的，可以由用户按需下载，并且可以作为免安装应用进行启动。

每个免安装应用都必须有且仅有一个基础功能 APK。 如果免安装应用只有一项功能，那么，用户只需要基础功能 APK；其他功能 APK 是可选的。如果免安装应用具有多项功能，则基础功能 APK 通常包含其他功能所依赖的共享资源和代码文件。 像地图免安装应用一样，基础 APK 可能包含此地图应用的基础 styles.xml 文件或数据结构类，以用于位置和景点建模。 无论用户请求哪项功能，均会下载基础功能 APK。

除基础功能 APK 外，开发者还可以有其他功能 APK。 其他功能 APK 可以包含与某项功能对应的应用片段。 功能 APK 包含针对该功能的入口点 Activity，以及该功能所需的任何独特资源。

当用户从免安装应用请求一项功能时，他们将获得两个功能 APK：所请求的功能 APK 和基础功能 APK。 如果同一个用户从该免安装应用请求另一项功能，他们可能只会收到该功能 APK，因为他们已下载基础功能 APK。 当然，如果免安装应用只有一项功能，因而只有一个基础功能 APK，则用户只会收到基础功能 APK。

下图描绘了免安装应用 APK 和功能 APK 之间的关系。



从 Google Play 商店请求功能

要从 Google Play 商店下载免安装应用的一项功能，用户只需点击一个 ****链接****。 当 Google Play 商店收到针对某个免安装应用的请求时，它将解析网址以确定用户请求哪项功能。 在找到匹配的功能后，Google Play 商店将与该功能对应的功能 APK 发送到用户设备，然后，免安装应用运行时启动该功能。 如果 Google Play 商店无法找到匹配的功能，它将在设备上提醒免安装应用运行时。 然后，免安装应用运行时将向系统发送一个广播 Intent 以便处理该网址。

因此，免安装应用中的每个入口点 Activity 都必须 ***可寻址***：它需要与一个唯一的网址对应。 如果免安装应用中各个功能的网址共享一个网域，则每项功能都需要与该网域中的一个不同路径对应。

关于免安装应用的开发指南，请参阅[Android开发者官网的Instant Apps入门指南](https://developer.android.com/topic/instant-apps/getting-started/index.html?hl=zh-cn" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。

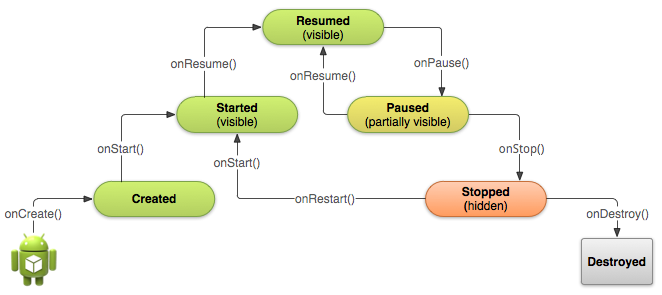
### **2.4. 应用⽣命周期管理**

#### **2.4.1. Activity**

不同于使用 main() 方法启动应用的其他编程范例，Android 系统会通过调用对应于其生命周期中特定阶段的特定****回调****方法在 [Activity](https://developer.android.com/reference/android/app/Activity.html?hl=zh-cn" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 实例中启动代码。****有一系列可启动 Activity 的回调方法，以及一系列可分解 Activity 的回调方法****。

在 Activity 的生命周期中，系统会按类似于阶梯金字塔的顺序调用一组核心的生命周期方法。 也就是说，Activity 生命周期的每个阶段就是金字塔上的一阶。 当系统创建新 Activity 实例时，每个回调方法会将 Activity 状态向顶端移动一阶。 金字塔的顶端是 Activity 在前台运行并且用户可以与其交互的时间点。

当用户开始离开 Activity 时，系统会调用其他方法在金字塔中将 Activity 状态下移，从而销毁 Activity。 在有些情况下，Activity 将只在金字塔中部分下移并等待（比如，当用户切换到其他应用时），Activity 可从该点开始移回顶端（如果用户返回到该 Activity），并在用户停止的位置继续。



如上图所示是简化的 ****Activity 生命周期****图示，以阶梯金字塔表示。此图示显示，对于用于将 Activity 朝顶端的“继续”状态移动一阶的每个回调，有一种将 Activity 下移一阶的回调方法。 Activity 还可以从“暂停”和“停止”状态回到继续状态。

根据 Activity 的复杂程度，开发者可能不需要实现所有生命周期方法（大多数情况下如此）。 但是，了解每个方法并实现确保应用按照用户期望的方式运行的方法非常重要。 正确实现 Activity 生命周期方法可确保应用按照以下几种方式良好运行，包括：

* 如果用户在使用应用时接听来电或切换到另一个应用，它不会崩溃。
* 在用户未主动使用它时不会消耗宝贵的系统资源。
* 如果用户离开应用并稍后返回，不会丢失用户的进度。
* 当屏幕在横向和纵向之间旋转时，不会崩溃或丢失用户的进度。

Activity 会在如上简化的 ****Activity 生命周期**** 所示不同状态之间过渡的几种情况。但是，这些状态中只有三种可以是静态。也就是说，Activity 只能在三种状态之一下存在很长时间。

* ****继续****。在这种状态下，Activity 处于前台，且用户可以与其交互。（有时也称为“运行”状态。）
* ****暂停****。在这种状态下，Activity 被在前台中处于半透明状态或者未覆盖整个屏幕的另一个 Activity 部分阻挡。 暂停的 Activity 不会接收用户输入并且无法执行任何代码。
* ****停止****。在这种状态下，Activity 被完全隐藏并且对用户不可见；它被视为处于后台。 停止时，Activity 实例及其诸如成员变量等所有状态信息将保留，但它无法执行任何代码。

其他状态（“创建”和“开始”）是瞬态，系统会通过调用下一个生命周期回调方法从这些状态快速移到下一个状态。 也就是说，在系统调用 onCreate() 之后，它会快速调用 onStart()，紧接着快速调用 onResume()。

以上就是基本 Activity 生命周期的管理方式和运行机理，对于用户****启动应用****这个日常操作（从主屏幕点击应用图标）来说，系统会为应用中已声明为“启动器”（或“主要”）Activity 的应用中的 Activity 调用 [onCreate()](https://developer.android.com/reference/android/app/Activity.html?hl=zh-cn" \l "onCreate(android.os.Bundle" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)) 方法。这是作为 应用的****用户界面主入口****的 Activity。

开发者可以在 Android 清单文件 AndroidManifest.xml 中定义哪个 Activity 用作主 Activity，该文件位于Android应用开发项目目录的根目录中。

应用的主 Activity 必须使用 （包括 MAIN 操作和 LAUNCHER 类别）在清单文件中声明。例如：

<activity android:name=".MainActivity" android:label="@string/app\_name">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter></activity>

注：使用 Android SDK工具创建新 Android 项目时，默认的项目文件包括使用过滤器在宣示说明中声明的 Activity 类。

如果未对应用的 Activity 之一声明 [MAIN](https://developer.android.com/reference/android/content/Intent.html?hl=zh-cn" \l "ACTION_MAIN" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 操作或 [LAUNCHER](https://developer.android.com/reference/android/content/Intent.html?hl=zh-cn" \l "CATEGORY_LAUNCHER" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 类别，那么应用图标将不会出现在应用的主屏幕列表中。

注：上述 主Activity 的实现规范是我们在进行 ****逆向分析**** 时常用的查找应用 ****主入口**** 的一个方法。

大多数应用包含若干个不同的 Activity，用户可通过这些 Activity 执行不同的操作。无论 Activity 是用户单击应用图标时创建的主 Activity 还是应用在响应用户操作时开始的其他 Activity，系统都会通过调用其 onCreate() 方法创建 Activity 的每个新实例。

开发者****必须实现**** onCreate() 方法执行只应在 Activity 整个生命周期出现一次的基本应用启动逻辑。 例如， onCreate() 的实现应定义用户界面并且可能实例化某些类范围变量。例如，onCreate() 方法中可以定义显示执行 Activity 某些基本设置的一些代码，比如声明用户界面（在 XML 布局文件中定义）、定义成员变量，以及配置某些 UI。

一旦 onCreate() 完成执行操作，系统会相继调用 onStart() 和 onResume() 方法。Activity 从不会驻留在“已创建”或“已开始”状态。在技术上，Activity 会在 onStart() 被调用时变得可见，但紧接着是 onResume()，且 Activity 保持“继续”状态，直到有事情发生使其发生变化，比如当接听来电时，用户导航至另一个 Activity，或设备屏幕关闭。

#### **2.4.2. Service**

Service 是一个可以在后台执行长时间运行操作而不提供用户界面的应用组件。服务可由其他应用组件启动，而且即使用户切换到其他应用，服务仍将在后台继续运行。 此外，组件可以绑定到服务，以与之进行交互，甚至是执行进程间通信 (IPC)。 例如，服务可以处理网络事务、播放音乐，执行文件 I/O 或与内容提供程序交互，而所有这一切均可在后台进行。

服务基本上分为两种形式：

* ****启动****。当应用组件（如 Activity）通过调用 startService() 启动服务时，服务即处于“启动”状态。一旦启动，服务即可****在后台无限期运行****，****即使启动服务的组件已被销毁也不受影响****。 已启动的服务通常是执行单一操作，而且不会将结果返回给调用方。例如，它可能通过网络下载或上传文件。 操作完成后，服务会自行停止运行。处于“启动”状态的服务根据是否会通知用户，又可以细分为“前台”服务和“后台”服务。
  + ****前台****。前台服务执行的操作会通知用户执行状态。例如，音频播放应用会使用前台服务播放音频。前台服务必须显示一个[状态栏图标](https://developer.android.com/guide/topics/ui/notifiers/notifications.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。即使用户不操作应用，前台服务也能一直运行。
  + ****后台****。后台服务不会直接通知用户执行状态。例如，应用使用后台服务执行存储压缩过程。如果应用运行环境是API level 26（Android 8.0）或以上版本，系统会对[应用未在前台时执行后台服务进行限制](https://developer.android.com/about/versions/oreo/background.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。默认情况下，这些限制仅适用于针对 Android 8.0 的应用。 不过，用户可以从 ****设置**** 屏幕为任意应用启用这些限制，即使应用并不是以 Android 8.0 为目标平台。在大多数情况下，应用都可以使用 [JobScheduler](https://developer.android.com/reference/android/app/job/JobScheduler.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 作业克服这些限制。这种方式让应用安排为在未活跃运行时执行工作，不过仍能够使系统可以在不影响用户体验的情况下安排这些作业。
* ****绑定****（Bound）。当应用组件通过调用 bindService() 绑定到服务时，服务即处于“绑定”状态。绑定服务提供了一个客户端-服务器接口，允许组件与服务进行交互、发送请求、获取结果，甚至是利用进程间通信 (IPC) 跨进程执行这些操作。 仅当与另一个应用组件绑定时，绑定服务才会运行。多个组件可以同时绑定到该服务，但全部取消绑定后，该服务即会被销毁。

服务可以同时以这两种方式运行，也就是说，它既可以是启动服务（以无限期运行），也允许绑定。问题只是在于开发者是否实现了一组回调方法：onStartCommand()（允许组件启动服务）和 onBind()（允许绑定服务）。

无论应用是处于启动状态还是绑定状态，抑或处于启动并且绑定状态，任何应用组件均可像使用 Activity 那样通过调用 Intent 来使用服务（即使此服务来自另一应用）。 不过，开发者可以通过[清单文件将服务声明为私有服务](https://developer.android.com/guide/components/services.html?hl=zh-cn" \l "Declaring" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)，并阻止其他应用访问。

注意：服务在其托管进程的主线程中运行，它既不创建自己的线程，也不在单独的进程中运行（除非另行指定）。 这意味着，如果服务将执行任何 CPU 密集型工作或阻止性操作（例如 MP3 播放或联网），则应在服务内创建新线程来完成这项工作。通过使用单独的线程，可以降低发生“应用无响应”(ANR) 错误的风险，而应用的主线程仍可继续专注于运行用户与 Activity 之间的交互。如需在主线程外部执行工作，不过只是在用户正在与应用交互时才有此需要，则应创建新线程而非服务。

如同 Activity（以及其他组件）一样，开发者必须在应用的清单文件中声明所有服务。

要声明服务，请添加 <service> 元素作为 <application> 元素的子元素。例如：

<manifest ... >

...

<application ... >

<service android:name=".ExampleService" />

...

</application></manifest>

如需了解有关使用清单文件声明服务的详细信息，请参阅 元素参考文档。

开发者还可将其他属性包括在 <service> 元素中，以定义一些特性，如启动服务及其运行所在进程所需的权限。android:name 属性是唯一必需的属性，用于指定服务的类名。应用一旦发布，即不应更改此类名，如若不然，可能会存在因依赖****显式 Intent**** 启动或绑定服务而破坏代码的风险。

为了确保应用的安全性，请****始终使用显式 Intent 启动****或****绑定 Service****，且不要为服务声明 Intent 过滤器。启动哪个服务存在一定的不确定性，而如果对这种不确定性的考量非常有必要，则可为服务提供 Intent 过滤器并从 Intent 中排除相应的组件名称，但随后必须使用 setPackage() 方法设置 Intent 的软件包，这样可以充分消除目标服务的不确定性。

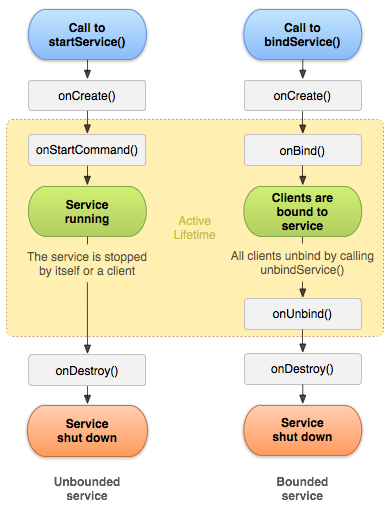
此外，还可以通过添加 android:exported 属性并将其设置为 "false"，确保服务仅适用于当前应用。这可以有效阻止其他应用启动当前应用中的****私有****服务，即便在使用显式 Intent 时也如此。

服务的生命周期比 Activity 的生命周期要简单得多。但是，密切关注如何创建和销毁服务反而更加重要，因为服务可以在用户没有意识到的情况下运行于后台。

服务生命周期（从创建到销毁）可以遵循两条不同的路径：

* ****启动服务****。该服务在其他组件调用 startService() 时创建，然后无限期运行，且必须通过调用 stopSelf() 来自行停止运行。此外，其他组件也可以通过调用 stopService() 来停止服务。服务停止后，系统会将其销毁。
* ****绑定服务****。该服务在另一个组件（客户端）调用 bindService() 时创建。然后，客户端通过 IBinder 接口与服务进行通信。客户端可以通过调用 unbindService() 关闭连接。多个客户端可以绑定到相同服务，而且当所有绑定全部取消后，系统即会销毁该服务。 （服务不必自行停止运行。）

这两条路径并非完全独立。也就是说，开发者可以绑定到已经使用 startService() 启动的服务。例如，可以通过使用 Intent（标识要播放的音乐）调用 startService() 来启动后台音乐服务。随后，可能在用户需要稍加控制播放器或获取有关当前播放歌曲的信息时，Activity 可以通过调用 bindService() 绑定到服务。在这种情况下，除非所有客户端均取消绑定，否则 stopService() 或 stopSelf() 不会实际停止服务。



上左图显示了使用 startService() 所创建的服务的生命周期，上右图显示了使用 bindService() 所创建的服务的生命周期。

通过实现这些方法，开发者可以监控服务生命周期的两个嵌套循环：

* 服务的整个生命周期从调用 onCreate() 开始起，到 onDestroy() 返回时结束。与 Activity 类似，服务也在 onCreate() 中完成初始设置，并在 onDestroy() 中释放所有剩余资源。例如，音乐播放服务可以在 onCreate() 中创建用于播放音乐的线程，然后在 onDestroy() 中停止该线程。 无论服务是通过 startService() 还是 bindService() 创建，都会为所有服务调用 onCreate() 和 onDestroy() 方法。
* 服务的有效生命周期从调用 onStartCommand() 或 onBind() 方法开始。每种方法均有 {Intent 对象，该对象分别传递到 startService() 或 bindService()。 对于启动服务，有效生命周期与整个生命周期同时结束（即便是在 onStartCommand() 返回之后，服务仍然处于活动状态）。对于绑定服务，有效生命周期在 onUnbind() 返回时结束。

注：尽管启动服务是通过调用 stopSelf() 或 stopService() 来停止，但是该服务并无相应的回调（没有 onStop() 回调）。因此，除非服务绑定到客户端，否则在服务停止时，系统会将其销毁 — onDestroy() 是接收到的唯一回调。

上图说明了服务的典型回调方法。尽管该图分开介绍通过 startService() 创建的服务和通过 bindService() 创建的服务，但是请记住，不管启动方式如何，任何服务均有可能允许客户端与其绑定。因此，最初使用 onStartCommand()（通过客户端调用 startService()）启动的服务仍可接收对 onBind() 的调用（当客户端调用 bindService() 时）。

## **3. Android应用软件的安全策略与机制**

### **3.1. APK制作过程面临的威胁与挑战**



如上图所示是经典的Android APK详细制作过程示意图，图中出现的[ProGuard](https://www.guardsquare.com/en/proguard" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)和[DexGuard](https://www.guardsquare.com/en/dexguard" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)分别是APK制作过程中常用的2款代码压缩和混淆工具。其中ProGuard是开源、免费Java class字节码压缩，优化, 混淆, 静态验证⼯具，能检测并删除⽆用类、成员、⽅法和属性;优化Java字节码并删除⽆用指令；使用简短⽆意义名称重命名类名、成员名和⽅法名（代码混淆）；压缩⽣成的⼆进制程序体积；增加⼆进制Java程序被逆向⼯程得到源代码和解读理解的难度。DexGuard是商业Android平台Dalvik⼆进制字节码压缩，优化, 混淆, 静态验证⼯具，可以加密字符串常量、类、原⽣库、资源⽂件等；隐藏敏感API调用⾏为，改用运⾏时反射⽅法等价调用；程序完整性检测与保护机制；发布版代码自动清理所有日志⾏为代码；进⼀步增加⼆进制Android平台Java程序被逆向⼯程得到源代码和解读理解的难度。

使用ProGuard和DexGuard类解决⽅案防⽌已发布的⼆进制程序被逆向⼯程恢复源代码，增加攻击者通过源代码审计，发现漏洞的难度和破解⼯作量。开发者编写的某些类⽅法如果使用ProGuard混淆会导致程序运⾏出错，所以会配置禁⽌ProGuard对这些代码的混淆，这给了攻击者还原出这部分代码的便利。

从Android Studio 3.0开始，Google默认开启了aapt2作为资源编译的编译器，aapt2的出现，为资源的增量编译提供了支持。

[AndResGuard](https://github.com/shwenzhang/AndResGuard" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)是微信Android开发团队开源出来的Android资源文件混淆打包工具，是一个帮助你缩小APK大小的工具，它的原理类似Java Proguard，但是只针对资源。它可以实现混淆资源ID长度(例如将res/drawable/welcome.png混淆为r/s/a.png)，同时利用7z深度压缩，大大减少了安装包体积，提升了破解难度。

AndResGuard不涉及编译过程，只需输入一个apk（无论签名与否，debug版，release版均可，在处理过程中会直接将原签名删除），可得到一个实现资源混淆后的apk（若在配置文件中输入签名信息，可自动重签名并对齐，得到可直接发布的apk）以及对应资源ID的mapping文件。

### **3.2. 权限模型**

从 Android 6.0（API 级别 23）开始，用户开始在应用运行时向其授予权限，而不是在应用安装时授予。此方法可以简化应用安装过程，因为用户在安装或更新应用时不需要授予权限。它还让用户可以对应用的功能进行更多控制；例如，用户可以选择为相机应用提供相机访问权限，而不提供设备位置的访问权限。用户可以随时进入应用的“Settings”屏幕调用权限。

系统权限分为两类：正常权限和危险权限：

* 正常权限不会直接给用户隐私权带来风险。如果应用在其清单（AndroidManifest.xml）中列出了正常权限，系统将自动授予该权限。
* 危险权限会授予应用访问用户机密数据的权限。如果您的应用在其清单中列出了正常权限，系统将自动授予该权限。如果您列出了危险权限，则用户必须明确批准您的应用使用这些权限。

需要注意的是，Android的权限模型并不是对抗攻击和威胁的万能手段，零『敏感』（危险）权限声明的App也可以实现⾼危操作，例如：香港中文大学的研究人员在2014年发表的论文：[Your Voice Assistant is Mine: How to Abuse Speakers to Steal Information and Control Your Phone](https://arxiv.org/abs/1407.4923" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)，介绍了一种新颖的权限绕过攻击方法：****Google语音搜索攻击****。攻击者可利用一个****零权限****的Android应用VoicEmployer，前台激活操作系统内置的语音助手模块Google Voice Search，后台播放预先准备好的音频文件，语音搜索能识别语音命令执行相应操作。借助这一机制，攻击者不需要任何权限就可以拨打任意电话号码，伪造短信/电子邮件，访问私人信息，传输敏感数据，实现远程控制。

类似上述 ****Google语音搜索攻击**** 的思路，当某apk权限请求与权限声明不对等的时候，那么恶意应用在没有对应 <uses-permission> 请求下，可以通过构造 Intent 访问用户安装其他合法应用的 Activity、Broadcast 、Service 的高权限接口来达到“暗度陈仓”和“借刀杀人”的效果。除此之外，如果拥有敏感数据的APK通过 ContentProvider 暴露敏感数据访问接口给任意应用调用也会导致Android系统的权限保护和沙盒隔离机制失效，恶意应用可以借此访问应用原本的私有数据空间。如果Android应用使用SQLite来管理持久化存储的数据，那么SQL注入威胁也会在系统权限模型保护能力范围之外危害到用户的隐私数据机密性和完整性。

在所有版本的 Android 中，应用都需要在其应用清单中同时声明它需要的正常权限和危险权限，如[声明权限](https://developer.android.com/training/permissions/declaring.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)中所述。不过，该声明的影响因系统版本和应用的目标 SDK 级别的不同而有所差异：

* 如果设备运行的是 Android 5.1 或更低版本，或者应用的****目标 SDK**** 为 22 或更低：如果开发者在清单中列出了危险权限，则用户必须在安装应用时授予此权限；如果用户不授予此权限，系统根本不会安装应用。
* 如果设备运行的是 Android 6.0 或更高版本，或者应用的****目标 SDK**** 为 23 或更高：应用必须在清单中列出权限，并且它必须在运行时请求其需要的每项危险权限。用户可以授予或拒绝每项权限，且即使用户拒绝权限请求，应用仍可以继续运行有限的功能。

通过扫描清单文件，我们可以在不运行、甚至不安装应用的情况下，了解应用可能（应用可以只声明需要用到哪些权限，但没有实现代码。但实践中，几乎不会有应用或恶意软件采用这种“狼来了”的行为）会请求访问哪些硬件设备（例如：GPS、相机、录音设备等），使用了哪些组件（例如，已知恶意软件组件名称），应用可以处理哪些 Intent （例如：恶意软件通过注册 BOOT\_COMPLETED 消息实现开机自启动）。

通过反编译APK和DEX文件，我们可以发现危险API调用和和敏感权限相关API调用，硬编码的关键字特征，例如：域名、IP等。

注：从 Android 6.0（API 级别 23）开始，用户可以随时从任意应用调用权限，即使应用面向较低的 API 级别也可以调用（通过使用***Android 支持库***）。无论开发的应用面向哪个 API 级别，开发者都应对应用进行测试，以验证它在缺少需要的权限时行为是否正常。

***[Android 支持库](https://developer.android.com/topic/libraries/support-library/index.html" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)*** 提供了诸多未内置于框架的功能。这些库提供向后兼容版本的新功能、框架中未包含的实用 UI 元素，以及应用可以利用的一系列实用程序。许多情况下，某项功能可能对应用开发者很有用，但是添加到 Android 框架却并不合适。例如，某个应用可能仅需要用于特定用例的某项功能，如在不同版本的 Android 系统之间顺畅切换。为了解决这一问题，Android SDK 添加了多个库，这些库统称为 ***Android 支持库***。如果应用开发者想要在应用中集成库功能，他们可以添加其中任意一个库。

支持库提供一系列不同的功能：

* 向后兼容版本的框架组件。
* 用于实现建议的 Android 布局模式的 UI 元素。
* 支持不同的设备类型。
* 其他实用程序功能。

### **3.3. 检查权限**

如果应用需要危险权限，则每次执行需要这一权限的操作时您都必须检查自己是否具有该权限。用户始终可以自由调用此权限，因此，即使应用昨天使用了相机，它不能假设自己今天仍具有该权限。

要检查当前应用是否具有某项权限，可以调用 ContextCompat.checkSelfPermission() 方法。例如，以下代码段显示了如何检查 Activity 是否具有在日历中进行写入的权限：

// Assume thisActivity is the current activityint permissionCheck = ContextCompat.checkSelfPermission(thisActivity,

Manifest.permission.WRITE\_CALENDAR);

如果应用具有此权限，方法将返回 PackageManager.PERMISSION\_GRANTED，并且应用可以继续操作。如果应用不具有此权限，方法将返回 PERMISSION\_DENIED，且应用必须明确向用户要求权限。

### **3.4. 处理权限请求响应**

当应用请求权限时，系统将向用户显示一个对话框。当用户响应时，系统将调用应用的 onRequestPermissionsResult() 方法，向其传递用户响应。应用开发者必须重写（override）该方法，以了解是否已获得相应权限。回调会将开发者传递的相同请求代码传递给 requestPermissions()。

系统显示的对话框说明了当前应用需要访问的权限组；它不会列出具体权限。例如，如果开发者请求 READ\_CONTACTS 权限，系统对话框只显示当前应用需要访问设备的联系人。用户只需要为每个权限组授予一次权限。如果当前应用请求该组中的任何其他权限（已在当前应用清单中列出），系统将自动授予应用这些权限。当应用请求此权限时，系统会调用应用内的 onRequestPermissionsResult() 回调方法，并传递 PERMISSION\_GRANTED，如果用户已通过系统对话框明确同意当前应用的权限请求，系统将采用相同方式操作。

注：当前应用仍需要明确请求其需要的每项权限，即使用户已向应用授予该权限组中的其他权限。此外，权限分组在将来的 Android 版本中可能会发生变化。开发者在开发代码时不应依赖特定权限属于或不属于相同组这种假设。

例如，假设在当前应用清单中列出了 READ\_CONTACTS 和 WRITE\_CONTACTS。如果应用请求 READ\_CONTACTS 且用户授予了此权限，那么，当应用请求 WRITE\_CONTACTS 时，系统将立即授予当前应用该权限，不会与用户交互。

### **3.5. 免安装应用的安全策略与机制**

免安装应用使用Android 6.0（API Level 23）引入的[运行时权限模型](https://developer.android.com/training/permissions/requesting.html?hl=zh-cn" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)。目前（Android 8.1）免安装应用可以使用以下Android权限：

* BILLING
* ACCESS\_COARSE\_LOCATION
* ACCESS\_FINE\_LOCATION
* ACCESS\_NETWORK\_STATE
* CAMERA
* INSTANT\_APP\_FOREGROUND\_SERVICE only in Android 8.0.
* INTERNET
* READ\_PHONE\_NUMBERS. This permission is available only in Android 8.0 (API level 26).
* RECORD\_AUDIO
* VIBRATE

所有免安装应用的对外通信流量强制使用****HTTPS****，禁止使用****HTTP****。目前免安装应用只能通过Google Play Console发布，用户也只能通过Google Play使用免安装应用。

免安装应用只能通过 ****隐式 Intent**** 调用（唤起）已安装应用，绝大多数已安装应用无法通过 显式 Intent 被调用，除非已安装应用通过显式 Intent声明过滤器来允许免安装应用与其交互。即使当前系统中已安装应用的版本低于免安装应用的版本，当用户点击免安装应用URL时也是优先启动本地已安装应用。免安装应用只要能被访问到，总是能保证使用到的是最新发布版本。

### **3.6. 证书与PKI**

通信过程的机密性、完整性和身份认证保护场景使用到的 SSL/TLS 会使用一个包含公钥及与其匹配的私钥的证书配置服务器。作为 SSL/TLS 客户端与服务器握手的一部分，服务器将通过使用公钥加密签署其证书来证明自己具有私钥。

不过，任何人都可以生成他们自己的证书和私钥，因此，一个简单的握手只能说明服务器知道与证书公钥匹配的私钥，除此之外什么都证明不了。解决此问题的一个方法是让客户端拥有其信任的一个或多个证书集。如果证书不在此集合中，则不会信任服务器。

但这个简单的方法有几个缺点。服务器应能够随时间的推移升级到更强的密钥（“密钥旋转”），使用新的公钥替换证书中的公钥。遗憾的是，客户端应用现在必须根据服务器配置发生的变化进行更新。如果服务器不在应用开发者的控制下（例如，如果服务器是一个第三方网络服务），则很容易出现问题。如果应用必须与网络浏览器或电子邮件应用等任意服务器通信，那么，此方法也会带来问题。

为弥补这些缺点，通常使用来自知名颁发者（称为 ****证书颁发机构**** (Certificate Authority,CA)）发放的证书配置服务器。主机平台一般包含其信任的知名 CA 的列表。从 Android 4.2 (Jelly Bean) 开始，Android 目前包含在每个版本中更新的 100 多个 CA。CA 具有一个证书和一个私钥，这点与服务器相似。为服务器发放证书时，CA 使用其私钥签署服务器证书。然后，客户端可以验证该服务器是否具有平台已知的 CA 发放的证书。

不过，在解决一些问题的同时，使用 CA 也会引发其他问题。因为 CA 为许多服务器发放证书，因此，开发者仍需要某种方式来确保客户端应用与需要的服务器通信。为解决这个问题，CA 发放的证书通过 ***gmail.com*** 等具体名称或 ***\*.google.com***等通配型主机集识别服务器。

一般情况下，应用信任所有预装 CA。如果有预装 CA 签发欺诈性证书，则应用将面临被中间人攻击的风险。例如历史上曾经出现过的2次CA公司被入侵的安全事件： [Comodo](http://en.wikipedia.org/wiki/Comodo_Group" \l "Breach_of_security" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 和 [DigiNotar](http://en.wikipedia.org/wiki/DigiNotar" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 出现了信息泄露，从而导致某个主机名的证书被发放给服务器或域的所有者以外的其他人。

为了降低此风险，Android 提供了将某些证书甚至整个 CA 列入黑名单的功能。尽管此名单过去已内置到操作系统中，但从 Android 4.2 开始，可以远程更新此名单，便于处理将来的泄露问题。

有些应用通过限制信任的 CA 集或通过 证书固定（Pinning Certificates）来选择限制其接受的证书集。

通过按公钥的哈希值（X.509 证书的 SubjectPublicKeyInfo）提供证书集完成证书固定。然后，只有至少包含一个已固定的公钥时，证书链才有效。

请注意，使用证书固定时，开发者应始终包含一个备份密钥，这样，当被强制切换到新密钥或更改 CA 时（固定到某个 CA 证书或该 CA 的中间证书时），当前应用的连接性不会受到影响。否则，开发者必须推送应用的更新以恢复连接性。

此外，可以设置固定的到期时间，在该时间之后不执行证书固定。这有助于防止尚未更新的应用出现连接性问题。不过，设置固定的到期时间可能会绕过证书固定。以下是一个配置固定证书的实例：

res/xml/network\_security\_config.xml：

<?xml version="1.0" encoding="utf-8"?><network-security-config>

<domain-config>

<domain includeSubdomains="true">example.com</domain>

<pin-set expiration="2018-01-01">

<pin digest="SHA-256">7HIpactkIAq2Y49orFOOQKurWxmmSFZhBCoQYcRhJ3Y=</pin>

<!-- backup pin -->

<pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3UKgO/04cDM1oE=</pin>

</pin-set>

</domain-config></network-security-config>

### **3.7. 『root』**

Android系统的root是 ****使得用户获取系统超级用户权限****的一种简称。root的基本原理利用Android系统的****本地提权漏洞****或****官方后门****获得root权限，不同手机厂商、系统和版本可能存在的漏洞不同、厂商对待用户获得root权限的态度不同，因此不同手机的root原理、方法、难度都可能不同。在获得root权限后用户和应用可以重新以可写模式挂载 /system ，将root权限管理应用[Superuser.apk](https://play.google.com/store/apps/details?id=eu.chainfire.supersu" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank) 写⼊/system/ app，将[su](https://github.com/ChainsDD/su-binary" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/_blank)写⼊ /system/xbin/ 和 /system/bin/，并用 chmod 命令为其设置可执行权限和 setuid 权限。其中Superuser.apk（提供用户操作权限管理的图形界面）负责对普通用户身份运行进程的root权限请求进⾏管理。

root之后的系统，一方面，用户对自⼰的⼿机拥有完全控制权，可以实现更加丰富的个性化定制。另一方面，提权漏洞利用效果的平台差异性通常较⼤，可能会引起系统不稳定，甚至部分手机厂商会对root手机拒绝提供硬件保修服务。恶意代码在一台root手机上可以实现的攻击⼿段会更丰富，例如：静默安装、静默卸载、静默访问任意数据等，而达成这些高级攻击效果的方式可能是一次社会工程学攻击、UI欺骗攻击等。另外，如果安装的Superuser.apk本身已经被恶意替换，相当于把系统的最高权限管理权交到了恶意代码和攻击者手里，后果不堪设想。

第七章 移动终端应用通信协议逆向分析

1. 软件逆向分析基础之静态分析

1.1. 静态分析基础

静态分析是指在不运⾏代码的情况下，采用词法分析、语法分析等各种技术⼿段对程序⽂件进⾏扫描从⽽⽣成程序的反汇编代码，然后阅读反汇编代码来掌握程序功能和理解程序原理的⼀种技术。在整个分析过程中，阅读理解反汇编代码是主要分析工作。反汇编（或反编译）代码⽣成⼯具简称反汇编（反编译）⼯具，选择一个好的反汇编工具可以帮助逆向工程师更有效率的去分析目标程序的内部实现原理。

静态分析Android应用程序有两种常用方法：第一种方法是阅读反汇编生成的Dalvik 字节码。常见的Dalvik字节码反汇编工具有IDA Pro、Apktool（内置了baksmali用于反编译生成smali文件，smali之于Dalvik字节码的关系就像汇编语言之于CPU机器指令的关系）。另一种方法是阅读反汇编生成的Java源代码，可以使用dex2jar从dex文件生成jar文件，然后再使用JD-GUI阅读jar文件的代码。本书重点介绍第一种方法，即反编译并阅读理解Dalvik 字节码。

1.2. APK反编译⼯具集

1.2.1. Apktool

1.2.2. Android Studio的APK Analyzer

在Android Studio 3.0中通过菜单 Build -> Analyze APK... 功能打开 Deliberately Vulnerable Android Hello World 编译出来的release版apk文件的截图。通过对 DisplayMessageActivity 类右键单击弹出菜单里选择 Show Bytecode 可以直接查看该类所在源代码文件的smali伪机器指令代码，如上图二所示。

使用 APK Analyzer 可以为应用“瘦身”，如上图一所示，文件和文件夹根据文件大小降序排列。这让我们很容易看出对 APK 大小优化最容易从哪里入手。每当你深入到某个文件夹的时候，你能看到占用了 APK 大部分空间的资源和其他实体。

通过APK Analyzer可以快速查看APK中文件的最终版本（比如AndroidManifest.xml、resources.arsc在被打包到APK文件中后，本来是二进制形式存储的，APK Analyzer能够解析并显示这些文件的内容）。

当你点击任何 DEX 文件，你可以看到一个含有多少类和方法被定义，以及它包含的总引用数（引用数不包括在单个 DEX 文件的 64K 限制里）的总结。你甚至可以深入到包里查看到底是谁使用了这些引用。

APK Analyzer 可以帮助你验证你的 Proguard 的配置是否正确，比如查看你需要的方法或类是否在 APK 中存在或者它们是否被重命名（混淆）。你可以确定你不需要的类是不是被移除了，且没有占用之前说的引用方法数。

APK Analyzer能比较两个不同APK中各个实体的大小。这对于我们了解为何你的APP相比上个版本变大了是很有用的。在发布一个新版APK之前，在APK Analyzer中导入你即将发布的APK。在右上角点击Compare With，选择上一个版本的APK，点击OK。然后就会出现类似图6的对话框，允许你比较之间的差别。

APK Analyzer还有命令行版apkanalyzer，该工具被包含在Android SDK Tools中，位于<android\_sdk>/tools/bin/apkanalyzer。更多关于APK Analyzer的使用请参与android开发者官网的Analyze Your Build with APK Analyzer。

1.3. DVM与JVM

1.4. ART

1.5. Dalvik 字节码

1.6. 定位关键代码的方法（静态）

在逆向一个 Android 应用程序时，无论是有无源代码，我们都不可能随机选择一个源文件开始分析，亦或完整阅读所有源代码或反汇编代码后再开始进行分析。本小节将介绍几种快速定位程序关键代码的静态分析方法。

每个 apk 文件中都包含的 清单文件（AndroidManifest.xml） 是最常见的定位关键代码的第一个突破口，因为在这个文件中记录了软件的包名、运行的目标系统版本、使用到的组件等。除此之外，在第六章 安卓系统访问控制策略与机制 的 应用生命周期管理 之 Activity 一节中，我们已经介绍过了定位“用户界面主入口的 Activity”的方法，该方法也是需要在 清单文件 进行关键字查找即可找到用户界面主入口的 Activity。

在上述以 清单文件 分析为基础的快速定位关键代码方法之外，在实践中，常见的静态定位关键代码的方法主要分为：信息反馈法、特征函数法、和顺序查看法三大类。

1.6.1. 信息反馈法

先运行目标程序，然后根据程序运行时给出的反馈信息作为突破口寻找关键代码。例如，Deliberately Vulnerable Android Hello World在模拟器中运行，输入注册码错误时的提示信息页面。注意到其中的提示消息内容为：注册失败。这就是程序反馈给我们的信息。通常情况下，程序中用到的字符串会存储在 strings.xml 文件或者硬编码到程序代码中，如果是前者的话，字符串在程序中会以 id 的形式访问，只需要在反汇编代码中搜索字符串 id 值即可找到调用代码处；如果是后者的话，在反汇编代码中直接搜索字符串即可定位到关键代码附近。

1.6.2. 特征函数法

本方法和信息反馈法的相似之处在于都是从程序运行时给出的反馈信息作为突破口寻找关键代码，区别在于 特征函数法 侧重于从消息的 UI（用户界面） 和 UE（用户交互）实现技术原理推测用到了Android SDK中提供的哪些相关API函数，而 信息反馈法 则是从消息的文本内容特征出发。例如，弹出注册失败错误提示信息可能是调用的 Toast.MakeText().Show() 方法，在反汇编代码中直接搜索 Toast 应该很快能定位相关调用代码，如果 Toast 在程序中被多次调用的话，可能需要分析人员逐个阅读核对。

1.6.3. 顺序查看法

顺序查看法通常是从 用户界面主入口 Activity 或其他启动代码入手分析，逐行向下分析，掌握软件的执行流程，这种分析方法在病毒分析时经常用到：大多数病毒都会实现开机自启动或随常用应用软件的启动而自动加载执行。

2. 软件逆向分析基础之动态分析

软件逆向分析中的动态分析指的是在没有软件源代码的情况下，调试程序时只能跟踪与分析汇编代码，查看寄存器的值，这些数据远远没有源码级调试展示的信息那么直观，但动态调试程序同样能跟踪软件的执行流程，反馈程序执行时的中间结果，在静态分析程序遇到瓶颈时，动态调试往往可以另辟蹊径，找到程序关键算法、关键代码、关键变量的突破口。

2.1. 定位关键代码的方法（动态）

2.1.1. 代码注⼊法（动态调试和分析）

代码注入法属于动态调试方法，基本原理是修改 apk 文件的smali反汇编代码，加入打印语句输出，配合 LogCat 查看程序执行到特定点时的（变量和关键数据结构）状态数据。这种方法在解密程序数据时经常使用，具体的例子可以查阅本章实验。

2.1.2. 栈跟踪法（动态调试和分析）

栈跟踪法的基本原理是输出运行时的栈跟踪信息，然后查看栈上的函数调用序列来理解方法的执行流程。和代码注入法类似，栈跟踪法主要是手动向反汇编后的smali文件中加入栈跟踪信息输出的代码。与注入 Log 输出的代码不同的是，栈跟踪法只需要知道大概的代码注入点。而且注入代码后的反馈信息比 Log 注入要详细的多。

例如一个典型的栈跟踪Java源代码如下：

new Exception("print trace").printStackTrace();

将其转换成smali语法的反汇编代码如下：

new-instance v0, Ljava/lang/Exception;

const-string v1, "print trace";

invoke-direct {v0, v1}, Ljava/lang/Exception;-><init>(Ljava/lang/String;)V

invoke-virtual {v0}, Ljava/lang/Exception;->printStackTrace()V

栈跟踪信息记录了程序从启动到 printStackTrace() 被执行期间所有被调用过的方法，从下往上查看栈跟踪信息，很容易就可以找到在打印栈跟踪信息之前的完整函数调用链，这样函数的执行流程就一清二楚了。

2.1.3. Method Profiling（动态调试和分析）

Method Profiling（方法剖析）主要用于代码覆盖热点分析和性能优化。该功能除了可以记录每个函数占用的CPU时间之外，还能够追踪所有的函数调用关系，并提供比栈跟踪法更详细的函数调用序列报告，这种方法在实践中可以帮助分析人员节省很多时间，也被广泛使用。

从Android Studio 3.0开始，Android Profile 被设计用来代替原先的Dalvik Debug Monitor Server (DDMS)来实现对程序执行过程额CPU、内存和网络信息等调试方法。

如果手动注入 Method Tracing代码的起始点与结束点不好确定，我们可以将它的范围设置的大一些，如在Activity的onCreate()方法中注入startMethodTracing()) 的代码反汇编代码如下：

const-string v0, "cuc"

invoke-static {v0}, Landroid/os/Debug;->startMethodTracing(Ljava/lang/String;)V

然后在 Activity 的 onStop() 方法中注入 stopMethodTracing() 的代码，反汇编代码如下：

invoke-static {}, Landroid/os/Debug;->stopMethodTracing()V

这样当程序打开关闭后就会生成 `cuc.trace 文件，接下来使用 traceview 工具来手动分析它即可。

3. 通信协议逆向分析基础

3.1. 模拟器

Android 模拟器可以模拟设备并将其显示在当前计算机上。利用该模拟器，开发者可以对 Android 应用进行原型设计、开发和测试，无需使用硬件设备。模拟器支持 Android 电话、平板电脑、Android Wear 和 Android TV 设备，并随附一些预定义的设备类型，便于开发者快速上手。开发者可以创建自己的设备定义和模拟器皮肤。需要注意的是，目前（Android SDK API Level 27）Android模拟器支持模拟真实设备的多数功能，不过虚拟硬件不包含以下功能：

WLAN、蓝牙、NFC（NFC近场通信技术是由非接触式射频识别（RFID）及互联互通技术整合演变而来，在单一芯片上结合感应式读卡器、感应式卡片和点对点的功能，能在短距离内与兼容设备进行识别和数据交换。）、SD 卡插入/弹出、连接到设备的耳机、USB

以命令行模式的Android模拟器控制软件为例，该程序位于 Android SDK的 tools 目录下。

# 确保 emulator 在系统环境变量 PATH 中已设置

# Linux & Mac 上的默认路径: $ANDROID\_SDK\_PATH/tools/emulator

# 查看系统中目前所有可用的Android模拟器

emulator -list-avds

# 启动上一条命令输出结果中的一个模拟器（使用模拟器代号）

# 该模拟器的所有通信数据会被写入当前目录下的 test.pcap 文件

emulator -tcpdump test.pcap -avd "Nexus\_5\_API\_19"

3.2. 真机

从 www.tcpdump.org 下载libpcap和tcpdump源代码，交叉编译出tcpdump的目标CPU架构可执⾏程序(静态链接，真机普遍使用ARM，模拟器可以使用x86或ARM)，将该 tcpdump 上传到真机的/data/local目录（先上传到SD卡（例如：/mnt/sdcard ），再su（所以真机需要先root）后移动到 /data/local）。在真机上通过 adb shell 命令行方式直接执行 tcpdump 二进制程序的效果如下图所示：

如果不想自己从源代码交叉编译Android平台上可运行的tcpdump，可以使用这里预先编译好的各个版本的tcpdump。

另外，如果希望直接在手机上运行tcpdump，而不使用adb shell方式，可以在手机上安装一个终端管理软件。例如，Terminal Emulator for Android、Termux或Termius等。

3.3. 针对HTTP协议通信监听与逆向分析

可以在PC/Mac上开启HTTP代理软件，Android手机或模拟器能够访问到该HTTP代理服务，设置并使用该代理。例如，可以参考burpsuite官网的Android手机HTP代理开启和配置方法。

和用tcpdump抓包方式相比，本方法无需手机root。

3.3.1. 在电脑上自建无线热点

具体可以参考第三章 Evil Twin基础实验，一旦手机连入到自建无线热点，那么手机所有的通信数据都可以在电脑上通过对指定的无线接入点网卡抓包实现完全监听和控制。本方法同样不需要手机root。

3.3.2. 通信数据监听的旁门左道——『投毒』

利用DHCP服务器，可以指定内⽹任意可抓包主机作为⽆线⽹络中无线设备的默认⽹关。除此之外，DHCP响应数据里可以指定客户端使用我们自建的DNS服务器，在配置DNS解析记录时，我们可以配置所有解析记录指向开启了透明正向HTTP代理的可抓包主机，实现手机的特定HTTP/HTTPS请求都被我们的透明正向HTTP代理服务器捕获。

除此之外，利用局域网的ARP欺骗配合dnsspoof，我们可以实现对所有基于域名解析请求的相关流量被发送到我们指定的IP地址上，我们只需要在该IP地址上完成对相应协议通信数据的代理，即可实现透明的流量监听。

除此之外，对于已root的Android手机，我们还可以直接修改Android系统里的/etc/hosts⽂件，实现主动劫持已知域名相关的通信流量到我们指定的IP地址上。

3.4. 监听蓝牙协议通信数据

Android 4.4+可以通过开发者选项中的“启用蓝牙HCI信息收集日志”功能捕获蓝牙设备与⼿机之间的点对点通信数据和⼿机的蓝牙⼴播数据。默认抓包结果写⼊到⼿机上的 /sdcard/btsnoop\_hci.log，该文件可以用wireshark读取并解析。⽆线监听其他非本⼿机蓝牙点对点通信数据需要专门的硬件设备，具体查阅本书第四章 移动通信安全概述的蓝牙通信数据嗅探基础一节。

如下图所示是在手机上监听手机和蓝牙键盘之间通信数据的一个抓包实例。

3.5. 通信协议逆向的核心

在获取到原始的网络通信数据抓包转储文件之后，需要分析工程师熟练使用⽹络协议分析⼯具对获取到的数据进行统计和深度分析。例如，使用wireshark的显示过滤规则和辅助⼯具。另外，网络协议的逆向往往需要结合软件逆向分析结果，特别是对于加密和私有协议的逆向分析更是如此。本章前文所述的静态和动态逆向分析技术，例如源代码级别、反汇编代码级别、注⼊代码并输出关键变量运⾏时状态值等方法，都常常被应用在网络协议逆向分析的辅助工作之中。

第八章 移动终端应用攻防与代码最佳安全实践

（简述，攻击手段有哪些）

1. 应用组件安全

1.1. 概述

Android应用组件包括 Activity、Service、Broadcast Receiver、和Content Provider，它们是 Android 应用软件开发人员每天都会接触到的东西。这些如果在 AndroidManifest.xml 配置不当，会被其他应用调用,引起风险。Android应用内部的Activity、Service、Broadcast Receiver等，他们通过 Intent 通信，组件间需要通信就需要在 AndroidManifest.xml 文件中暴露组件，前面提到的风险就有可能是不恰当的组件暴露引起的。

\*\* 应用组件风险总结概括如下：



1.2. 恶意调用 Activity

以恶意调用 Activity 为例，我们在[第六章 的Activity Manager (am) 实验](https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/exp.html)中已经通过 am start 命令展示了只要知道了 Activity 组件名称，就可以直接调用唤起。不仅如此，Service 也可以通过 am startservice 启动。类似于Web安全里的任意URL访问和请求参数枚举类攻击，一旦Android攻击者掌握了目标应用的敏感Activity组件名称，就有可能绕过脆弱的访问控制，直接操作原本受访问控制保护的用户交互界面。

在AndroidManifest.xml中的<activity>元素下有一个属性字段android:exported=["true" | "false"]用于决定当前Activity 是否可由其他应用的组件启动。 “true”表示可以，“false”表示不可以。若为“false”，则 Activity 只能由同一应用的组件或使用同一用户 ID 的不同应用启动。默认值取决于 Activity 是否包含 Intent 过滤器。没有任何过滤器意味着 Activity 只能通过指定其确切的类名称进行调用。 这意味着 Activity 专供应用内部使用（因为其他应用不知晓其类名称）。因此，在这种情况下，默认值为“false”。另一方面，至少存在一个过滤器意味着 Activity 专供外部使用，因此默认值为“true”。例如以下示例定义：

<activity

android:name=".LoginActivity"

android:label="@string/app\_name"

android:exported="false">

该属性并非限制 Activity 对其他应用开放度的唯一手段。开发者还可以利用权限来限制哪些外部实体可以调用 Activity（请参阅 [permission](https://developer.android.com/guide/topics/manifest/activity-element.html?hl=zh-cn" \l "prmsn" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x08/_blank) 属性）。如下是一个标准的权限设置范例：

参与跨应用调用的组件或者公开的广播、服务设置权限。设置权限如下：

（1）组件添加android：permission属性。

<activity android:name=".Another" android:label="@string/app\_name"

android:permission="com.test.custompermission"></activity>

（2）声明属性

<permission android:description="test"

android:label="test"

android:name="com.test.custompermission"

android:protectionLevel="normal"></permission>

protectionLevel 有四种级别 normal、dangerous、signature、signatureOrSystem。signature、signatureOrSystem时，只有相同签名时才能调用。

（3）调用组件者声明

<uses-permission android:name="com.test.custompermission" />

Android 提供各种 API 来在运行时检查、执行、授予和撤销权限。这些 API

是 android.content.Context 类的一部分，这个类提供有关应用程序环境的全局信息。

if (context.checkCallingOrSelfPermission("com.test.custompermission")

!= PackageManager.PERMISSION\_GRANTED) {

// The Application requires permission to access the

// Internet");

} else {

// OK to access the Internet

}

### **1.3. Activity （钓鱼）劫持**

当用户安装了带有 Activity 劫持功能的恶意程序后，恶意程序会遍历系统中运行的程序。当检测到需要劫持的 Activity（通常是网银或者其他网络程序的登录界面）在前台运行时，恶意程序会启动一个带有 [FLAG\_ACTIVITY\_NEW\_TASK](https://developer.android.com/reference/android/content/Intent.html" \l "FLAG_ACTIVITY_NEW_TASK" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x08/_blank) 标志的钓鱼式 Activity 覆盖正常的 Activity 从而欺骗用户输入用户名或者密码信息。当用户输入完信息后，恶意程序会将信息发送到指定的网址或者邮箱，然后切换到正常的 Activity中去。

这种劫持手法透明，一般防病毒软件无法检测，手机用户更是防不胜防。有个比较简单的检测方式是长按Home键查看最后运行的程序来判断 Activity 是否被劫持过。然而这种方法并不是时常有效，攻击者可以设置属性 android:exludeFromRecents 的值为 true，即使运行过的进程也不会显示最近运行过的列表中，这样前述简单检测方法就失效了。

### **1.4. Service 安全**

service一般用作后台任务，由于它没有界面，执行起来更加隐蔽，同样可以使用 android:exported 属性为 false 来阻止其他恶意软件调用。 同一作者如果想多个程序共享使用该服务可以定义权限，没有声明该权限下调用该服务会报 SecurityException 异常。

### **1.5. Broadcast Receiver 安全**

凡是通过intent-filter来接收的 Activity 都可以通过外部调用来访问，之前讲过解决方案是组件属性android:exported属性设置为false，让组件只能接收到同用户ID和签名的进程发送的广播。要想指定哪类进程可以发送，也可以配置permission。

广播分为有序广播和无序广播两种，分别是 sendBroadcast() 和 sendOrderedBroadcast()方法。无序广播可以被所有广播接收者接收并且不能通过 abortBroadcast() 方法中止广播。有序广播按照优先级度顺序接受，优先度高的广播可以拦截广播调用 abortBroadcast() 中止广播。优先级度通过属性 android:priority 属性的数值来决定：数值越大，优先度越高。取值范围-1000到1000。我们可以设置接收短信的 receiver 的值为最大值 就可以拦截短信，短信不会存到收件箱。

针对伪造广播消息绕过访问控制和广播劫持类威胁，目前主要有五种安全解决方案：

1. 静态注册广播时添加 android:exported="false"属性，禁止本应用的广播接收器响应其他应用的广播。

<receiver android:name="com.test.MyBroadCastReceiver" android:exported="false">

<intent-filter >

...

</intent-filter></receiver>

1. 动态注册广播时，通过报名设置intent.setPackage("com.test.demo")，来决定你的广播对那个应用有效。

Intent intent=new Intent("com.test.action");

intent.setPackage("com.test.demo");this.sendBroadcast(intent);

1. 使用 LocalBroadcastManager 来实现 进程内广播，其他进程无法接收广播信息。

注意：同一应用的其他进程也接收不到广播信息。

// 发送广播

Intent intent = new Intent("com.test.action");

LocalBroadcastManager.getInstance(this).sendBroadcast(intent);

// 接收广播

IntentFilter filter = new IntentFilter();

filter.addAction("com.test.action");

mReceiver = new MyBroadCastReceiver();

mLocalBroadcastManager = LocalBroadcastManager.getInstance(this);

mLocalBroadcastManager.registerReceiver(mReceiver, filter);

1. 使用 sendBroadcast(Intent intent, receiverPermission) 发送权限广播，只有具有这个权限的应用才能处理这个广播。

// 广播发送

Intent intent = new Intent("com.test.action");

sendBroadcast(intent,"com.test.custom.permission");

<!-- 广播接收 --><!-- 在接收广播的APP 的 Manifest.xml文件中添加自定义权限 -->

<uses-permission android:name="com.test.custom.permission" />

<permission

android:name="com.test.custom.permission"

android:protectionLevel="normal"></permission>

// 注册广播接收器

IntentFilter filter = new IntentFilter();

filter.addAction("com.test.action");

mReceiver = new MyBroadCastReceiver();

registerReceiver(mReceiver, filter);

1. 使用 registerReceiver(BroadcastReceiver receiver, IntentFilter filter, String broadcastPermission, Handler scheduler) 注册广播，只处理具有这个权限的APP发送的广播。

<!-- 广播发送 --><!-- 在发送广播的APP 的 Manifest.xml文件中添加自定义权限 --><uses-permission android:name="com.test.custom.permission" /><permission

android:name="com.test.custom.permission"

android:protectionLevel="normal"></permission>

// 发送广播

Intent intent = new Intent("com.test.action");

sendBroadcast(intent);

// 广播接收

IntentFilter filter = new IntentFilter();

filter.addAction("com.test.action");

mReceiver = new MyBroadCastReceiver();

registerReceiver(mReceiver, filter, "com.test.custom.permission", null);

### **1.6. Context Provider 安全**

Content Provider 组件是Android应用的重要组件之一，管理对数据的访问，主要用于不同的应用程序之间实现数据共享的功能。Content Provider 的数据源不止包括SQLite数据库，还可以是文件数据。通过将数据持久化存储层和应用层分离，Content Provider 为各种数据源提供了一个通用的接口。

如果在 AndroidManifest.xml 文件中将某个 Content Provider 的 exported 属性设置为true，则当前应用就多了一个威胁来源点。如果此 Content Provider的实现有问题，则可能产生任意数据访问、SQL注入、目录遍历等风险。更多关于Content Provider组件安全可以查阅阿里聚安全的这篇文章：[Android安全开发之Provider组件安全](https://jaq.alibaba.com/community/art/show?articleid=352" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x08/_blank)。

## **2. 数据安全**

Android 手机中存放着许多用户个人相关数据，例如：手机号码、通讯录、短信、聊天记录、电子邮件等，这些数据涉及用户个人隐私和商业秘密，但实际应用程序在存储和网络传输时往往采取了一些错误的编码实现方案，导致了包括数据被窃、数据被篡改等安全事件的发生。以下，我们从应用程序开发的角度来看一看这些问题是如何产生的。

### **2.1. 存储安全**

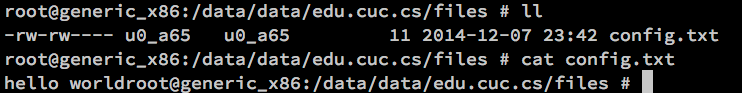
数据分外部存储和内部存储两种，使用 new File("/sdcard/config.txt"); 就可以在外部存储设备上读写文件。AndroidManifest.xml中声明以下权限就可以读写外部存储设备上的任意数据：

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE">

乌云网上相关漏洞实例如下：

* ⼿机QQ 2012（Android）3.0可导致用户聊天信息泄露；
* 印象笔记客户端设计缺陷可导致用户信息泄露；
* ⼈⼈⽹Android客户端可能导致劫持或恶意软件安装，从不可信位置安装软件未做⽂件真实性和完整性检查。

使用 openFileOutput("/sdcard/config.txt", MODE\_PRIVATE) 这类为内部存储文件读写。注意第二个参数 MODE\_PRIVATE 使其他程序无法访问这个文件。如果改为 MODE\_WORLD\_READABLE 就会被其他程序访问到。应用使用内部存储时，数据默认是只能被当前应 用访问（基于⽂件访问权限设置）。⼀旦系统被root或代码编写错误，受保护数据将被其他程序访问到。



### **2.2. 密钥认证**

详细内容请参阅[第六章 Android安全机制概览中的密钥存储与密码学算法](https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/main.html" \l "%E5%AF%86%E9%92%A5%E5%AD%98%E5%82%A8%E4%B8%8E%E5%AF%86%E7%A0%81%E5%AD%A6%E7%AE%97%E6%B3%95)一节内容。

### **2.3. 传输（通信）**

详细内容请参阅[第六章 Android安全机制概览中的证书与PKI](https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x06/main.html" \l "%E8%AF%81%E4%B9%A6%E4%B8%8EPKI)一节内容。

### **2.4. ROM 安全**

根据 ROM 制作者不同， Android 系统的 ROM 分为如下三类：原生ROM、设备厂商定制ROM和第三方ROM。其中原生ROM指的是直接使用 Android Open Source Project 源代码编译出来的系统镜像，消费者实际能够购买到的设备基本都是来自设备厂商预装了厂商基于上游 AOSP 源代码修改定制之后的系统。第三方ROM则指的是独立于设备厂商的个人或社区团队组织基于 AOSP 代码修改编译出来的系统镜像。

安天实验室在2011年披露的《对CarrierIQ木马的综合分析报告》中报道了世界首例手机ROM中被植入隐私窃取软件的安全事件：

**根据该公司公开的产品专利和产品培训资料，该产品搜集与网络相关的信息，包括语音和数据服务；还搜集与网络无关的信息，包括设备类型、可用内存和电池电量、设备中软件的类型、设备的地理位置信息、设备用户的按键信息、设备的使用历史等。这些信息被传回至 Carrier IQ 公司的服务器进行统计分析。其提供的后台产品可以根据 IMEI 或 IMSI对任何一个设备进行详细的历史记录查询，即用户的隐私被完全暴露给该公司及其产品用户。**

**运营商 Verizon 和 Sprint 在其多款手机中预装了这一软件，涉及 Android、Symbian 和 BlackBerry 三个平台。相关报道称受影响设备数量达 1.41 亿。多个著名的第三方定制 ROM 提供商，例如 CyanogenMod，也曾采用这一软件。**

类似Windows盗版系统横行时代，盗版软件作者在修改版软件中加入后门代码、恶意代码等，针对移动终端系统的恶意ROM还经常会被加入恶意扣费程序、间谍程序和窃密程序等。

### **2.5. 边信道信息泄漏**

当应用处理用户或其它数据源输入的数据时，可能会把数据放在不安全的位置，而这些数据能够被同一设备上其它的恶意应用读取，这就造成了风险。由于利用这样的边信道信息泄露非常容易，导致应用容易受到很严重的攻击，攻击者通过一小段代码就能够读取不安全的敏感信息，我们也能通过adb之类的工具来读取。以下是以下边信道信息泄露的常见场景：

* 日志。Android提供的日志功能也是一个会造成信息泄露的地方，日志一般是开发者在开发期间调试使用的，我们在第七章实验中提到的adb logcat就可以读取到应用运行时产生的所有日志。
* 系统剪贴板信息。如果用户把信用卡号这样的敏感信息复制到了剪贴板，攻击者通过一小段代码就能轻轻松松读取到数据。
* URL缓存和浏览器Cookie对象。已经有大量基于 webview 的应用程序造成URL，cookie和缓存等泄露的问题，这允许攻击者劫持用户的会话。这样的缓存可能存在日志，流量历史，浏览缓存等多种形式。
* 第三方统计数据。某些情况下，有的应用会使用第三方API。在使用这类应用时，第三方的API可能会读取诸如设备ID及位置信息等敏感信息。

## **3. WebView安全**

WebView 可在应用中嵌入一个内置的 Web 浏览器，是 Android 应用开发常用的组件之一。通过 WebView 对 Android 应用的攻击案例屡见不鲜，例如2013年时爆发的 addJavascriptInterface 远程代码执行。但修复了 addJavascriptInterface并不表示就能高枕无忧。应用在 WebView 上为 Javascript 提供的扩展接口，可能因为接口本身的问题而变成安全漏洞。

除此之外，在没有启用进程隔离的 WebView 与 App 具有相同权限，获得任意代码执行后可以访问应用私有数据或其他系统接口，可以将浏览器漏洞移植到手机平台上对应用进行针对性攻击。部分厂商使用自行基于开源浏览器引擎 fork 而来的内核，也可能因为同步上游补丁代码不及时而出现可利用的漏洞。

已知的 WebView 任意代码执行漏洞有 4 个，较早被公布是 [CVE-2012-6636](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-6636" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x08/_blank)，揭露了 WebView 中 addJavascriptInterface 接口会引起远程代码执行漏洞。接着是 [CVE-2013-4710](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-4710" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x08/_blank)，针对某些特定机型会存在 addJavascriptInterface API 引起的远程代码执行漏洞。之后是 [CVE-2014-1939](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-1939" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x08/_blank) 爆出 WebView 中内置导出的 “searchBoxJavaBridge\_” Java Object 可能被利用，实现远程任意代码。再后来是 [CVE-2014-7224](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-7224" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x08/_blank)，类似于 [CVE-2014-1939](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-1939" \t "https://sec.cuc.edu.cn/huangwei/textbook/mis/chap0x08/_blank) ，WebView 内置导出 “accessibility” 和 “accessibilityTraversal” 两个 Java Object 接口，可被利用实现远程任意代码执行。

出于安全考虑，Google 在 API 17 版本中就规定能够被调用的函数必须以 @JavascriptInterface 进行注解，理论上如果 APP 依赖的 API 为 17（Android 4.2）或者以上，就不会受该问题的影响，但在部分低版本的机型上依然受影响。

在 Android N 中增加了一个开发者选项，就是在所有的应用中将 WebView 的渲染进程运行在独立的沙箱中。即使恶意网页通过漏洞在渲染进程中执行了代码，还需要更多的漏洞绕过沙箱的限制。这一特性在 Android O 中默认启用。但在这一缓解措施正式部署到大部分设备之前，通过攻击 WebView 获得远程代码执行进而直接攻击应用仍然是可行的。

常见的XSS攻击的是因为客户端或服务端的代码开发不严谨等问题而存在漏洞的目标网站或者应用程序。这些攻击的先决条件是页面存在漏洞，而它们的影响往往也围绕着漏洞页面本身的用户会话。换句话说，因为浏览器的安全功能的影响，XSS攻击只能读取受感染的会话，而无法读取其他的会话信息，也就是同源策略的影响。UXSS（Universal XSS）保留了基本XSS的特点，利用漏洞，执行恶意代码，但是有一个重要的区别：

不同于常见的XSS，UXSS是一种利用浏览器或者浏览器扩展漏洞来制造产生XSS的条件并执行代码的一种攻击类型。

从2011年开始，Google的chromium项目的Bug列表就陆续被报告webkit存在一些UXSS漏洞。比如CVE-2011-3881 WebKitHTMLObjectElement UXSS漏洞，其它很多UXSS漏洞也是对源检测不全导致的跨域问题，虽然在Chrome浏览器上修复了，但由于Android系统引用了同一套WebKit，导致曾被修复的漏洞再次重现。2013年底，国内安全人员就开始陆续将这类Android UXSS爆光在网上。例如，腾讯SRC的安全研究人员做了个演示demo，通过提供一个二维码或者URL发送给他人，当使用聊天工具或者购物应用扫描二维码或者打开链接后，那么就可以窃取到他人的个人资料。

前面提到的这些UXSS漏洞，目前已经在Android 4.4中修复，同时它也提供了自动升级WebKit的功能，以便及时修复漏洞，因此历史经验总是在告诉我们：建议广大用户尽量采用最新版的软件和系统。