Wpa

WPAS 所有工作都围绕事件展开。即，他是基于事件驱动的。事件驱动和消息驱动类似，主线程的等待事件的发生并处理他们。WPAS 没有使用多线程编程，所有事件处理都在主线程中完成。从这一点看，WPAS的运行机制到时很简单。

位于eventloop模块下放的driver i/f （i/f代表inerface）接口模块用于隔离和底层驱动直接交互的那些driver控制模块（如wext ndiswrapper等，WPAS中称之为driver wrapper）.这些driver wrapper 和平台以及芯片所使用的驱动相关。不过，由于driver if的隔离作用，WPAS 中其他模块将最大程度的保持平台以及驱动无关性。

Driver wrapper 经常要返回一些信息给上层。WPAS中，这些信息通过driver events的方式反馈给WPAS其他模块进行处理。

EAP以及EAPOL协议。除了定义消息格式外，RFC4127文档定义了EAP状态机，而802.1x文档中还定义了EAPOL状态机。WPAS根据这两个协议分别实现了EAP和EAPOL状态机。除此之外，WPAS还定义了自己的状态机（statemachine）.

WPAS实现了多种EAP 方法。另外还包含了TLS模块和CRYPTO模块用于支持对应的EAP 方法。

EAPOL以及EAP消息都属于LLC层数据，所以WPAS的I2\_packet模块用于收发EAPOL 和EAP消息。

WPAS支持较多的配置参数，这些参数的处理有configuration模块完成。

WPAS 是C/S结构中的server端，。它通过ctrl i/f 模块向客户端提供通信接口。Linux/unix平台中，client端利用unix域cocket与其通信。目前常用的client端wpa\_cli和 wpa\_gui实现。

WPAS定义了许多命令，常见：

。ping 心跳测试命令。客户端利用它判断WPAS是否工作正常。WPAS收到ping命令后需要回复PONG。

。MIB：客户端利用该命令来获取设备的MIB信息。

。STATUS： 客户端用该命令获取WPAS的工作状态。

。ADD\_NETWORK ： 为WPAS添加一个新的无线网络。它将返回此新无线网络的id（注意从0开始）。注意：此network\_id非常重要，客户端后续将通过它来致命自己想操作的无线网络。

。SET\_NETWORK <NETWORK\_ID> <VARIBLE> <VALUE> : network id是无线网络的id.此命令用于设置指定无线网络的信息。其中的varible为参数名，value为参数的值。

。ENABLE\_NETWORK <NETWORK\_ID> ：是能某个无线网络。此命令最终将促使WPAS发起一系列操作以加入该无线网络。

除了接收来自client的命令外，WPAS 也会主动给Client发送命令。例如，WPAS 需用户为木有无线网络输入密码，这类命令称之为interactive Request,其格式如下。

WPAS 向客户端发送的命令遵循 CTRL-REQ <FIELE NAME> <NETWORK\_ID> <HUMAN\_READABLE TEXT> 格式。如CTRL-REQ-PASWORD-0-PASSWORK NEEDED FOR SSID TEST-NETWORK.这条指令表示用户为0号网络输入密码。

客户端处理完后，需回复 CTRL-RSP<FIELD\_NAME> <NETWORK\_ID><VALUE>。 目前支持的field包括password,Identity(EAP中的identity 或者用户名)、PIN等。

最后，WPAS还可通过形如“CTRL-EVENT <filed ><text>”的命令向客户端通知一些事情。

提示：除了”CTRL-EVENT-XXX”之外，WPAS还支持形如WPA:XXX和WPS-XXX的通知事件。这些事件和WPA和WPS有关。

控制 介绍

客户端 使用wpa\_ctrl时 首先要分配控制对象。下面的两个api用于创建和销毁控制对象wpa\_ctrl:

Struct wpa\_ctrl \* wpa\_ctrl\_open()

Void wpa\_ctrl\_close()

下面这个函数用于发送指令给WPAS。

客户端发送命令给wpa\_supplicant,护肤的消息保存在reply中。

Int wpa\_ctrl request()

Msg\_cb是一个回调函数，该参数的设置和WPAS中C/S通信进制的设计有关。

从Client角度来看，它发给WPAS的命令所对应的回复属于solicited event（意为有请求的事件），而前面所提到额CTRL\_EVENT时间（用于通知事件），对应为unsolicited event（意为 未请求的事件）。当Client 在等待某个命令的回复时，WPAS同事可能有些通知事件要发送给客户端，这些通知事件不是该命令的回复，所以不能通过wpa\_ctrl\_request的reply参数返回。为了防止丢失这些通知事件，wpa\_cli设计了一个msg\_cb回调用于客户端在等待命令恢复的时候处理那些unsolicited event.这种一个函数完成两样完全不同的功能设计实在有些特别，所以wpa\_supplicant规定只有打开通知事件监听功能的wpa\_ctrl对象才能在wpa\_ctrl\_request中通过msg\_cb 获取通知事件。而打开通知事件监听功能相关的api如下所示：

// 打开通知事件监听功能。

Int wpa\_ctrl\_attach()

Int wpa\_ctrl\_recv()

如果客户端并不发送命令，而只是想接受unsolicite event的话，可通过wpa\_ctrl\_recv() 函数来达到此目的。

Wpa\_ctrl\_recv 和wpa\_ctrl\_request 不方便，所以通用的方法：

客户端创建两个wpa\_ctrl对象来简化自己跌逻辑处理。

一个打开了通知事件监听功能的wpa\_ctrl对象将只通过wap\_ctrl\_recv()来接受通知事件。

另一个wpa\_ctrl专职用于发送命令和接收回复。由于没有调用wpa\_ctrl\_attach，故它 不会受到通知事件。

**4.5  wpa\_supplicant连接无线网络分析**

本节将介绍本章第二条分析路线，即通过命令行发送命令的方式触发wpa\_supplicant进行相关工作，使手机加入一个利用WPA-PSK进行认证的无线网络。

以笔者的Note 2为例，整个过程用到的命令如下所示。

[命令示例]

adb root #获取手机root用户权限。只有root被破解的手机才能成功

adb shell #登录手机shell

#笔者事先已编译wpa\_cli并将其放到/system/bin目录中。这个命令用于启动wpa\_cli，-i参数指明unix域控制

#socket文件名，它应该和wpa\_supplicant启动时设置的控制接口文件名一致

wpa\_cli -iwlan0 #该命令执行后，将进入wpa\_cli进程，后续操作都在此进程中开展

#发送ADD\_NETWORK命令给wpa\_supplicant，它将返回一个新网络配置项的编号。请参考4.3.3.1"wpas\_ssid结构

#体介绍"一节

ADD\_NETWORK #假设wpa\_supplicant返回的新网络配置项编号为0

SET\_NETWORK 0 ssid "Test" #设置0号网络的ssid为“Test”

SET\_NETWORK 0 key\_mgmt WPA-PSK #设置0号网络的key\_mgmt为“WPA-PSK”

SET\_NETWORK 0 psk "12345Test" #设置0号网络的psk为“12345Test”

ENABLE\_NETWORK 0 #使能0号网络，它将触发wpa\_supplicant扫描、关联等一系列操作直到加入无线网络“Test”

CTRL+C #退出wpa\_cli

dhcpcd wlan0 #启动dhcpd，wlan0为无线接口设备名。dhcpcd可为手机从AP那获取一个IP地址

dhcpcd成功执行后，手机将从AP那分配到一个IP地址。至此，手机就可以使用“Test”无线网络了。

注意：上述命令执行前有几个注意事项：