基于ubuntu的wpa\_supplicant工具的安装与使用

wpa\_supplicant是WPA Supplicant组件的一个实现，即在客户站中运行的部分。它实现了WPA密钥与WPA认证者进行协商并使用EAP进行身份验证认证服务器。另外，它控制着漫游和IEEE802.11认证/关联的WLAN驱动程序。

wpa\_supplicant被设计成一个运行在后台的守护进程，并充当控制无线连接的后端组件。wpa\_supplicant包含一个支持单独的前端程序和一个示例基于文本的前端wpa\_cli

使用WPA与AP关联时，使用以下步骤：

- wpa\_supplicant请求内核驱动程序扫描相邻的BSS

- wpa\_supplicant根据配置选择BSS

- wpa\_supplicant请求内核驱动程序关联所选内容

BSS

- 如果WPA-EAP：集成IEEE 802.1X请求者完成EAP

认证服务器认证（由代理服务器代理）

AP中的身份验证器）

- 如果从IEEE 802.1X请求者收到WPA-EAP：主密钥

- 如果WPA-PSK：wpa\_supplicant使用PSK作为主会话密钥

- wpa\_supplicant完成WPA 4次握手和组密钥握手

与认证者（AP）

- wpa\_supplicant为单播和广播配置加密密钥

- 可以发送和接收正常的数据包

# 第一章 安装

Wpa\_supplicant的安装方法有两种：

第一种是在ubuntu中使用命令：

Sudo apt-get install wpasupplicant

该命令可将wpa\_supplicant、wpa\_cli、wpa\_passphrase直接安装。十分方便

第二种是直接在官网下载源代码，编译、安装

## 使用命令安装

在终端输入sudo apt-get install wpasupplicant

## 源码编译安装

### 下载wpa\_supplicant和openssl

由于wpa\_supplicant需要使用openssl库，所以需要下载openssl库

下载源码

<http://hostap.epitest.fi/wpa_supplicant/>

        下载wpa\_supplicant-0.7.3.tar.gz （openssl用到0.7.3提供的补丁）

                #tar xvfz wpa\_supplicant-0.7.3.tar.gz

        下载[www.openssl.org/source/openssl-1.0.1c.tar.gz](http://www.openssl.org/source/openssl-0.9.8e.tar.gz)  
                #tar zxvf openssl-1.0.1c.tar.gz

### 编译openssl库

**1、首先openssl需要用到wpa\_supplicant中的补丁（可能不需要）**

把wpa\_supplicant里面的patches文件夹下的openssl-0.9.8e-tls-extensions.patch文件拷贝到openssl-0.9.8za目录下

运行：

**patch -p1 < openssl-0.9.8e-tls-extensions.patch**

**2、如果需要交叉编译，需要在makefile中修改以下三个变量。**如下以arm为例

makefile修改 (-为去掉设置， +为新加的设置)

#vi makefile

- CC= cc

+ CC= arm-none-linux-gnueabi-gcc

- AR= ar $(ARFLAGS) r

+ AR= arm-none-linux-gnueabi-ar $(ARFLAGS) r

- RANLIB= /usr/bin/ranlib  
+ RANLIB= arm-none-linux-gnueabi-ranlib  
Makefile修改完毕

**3、编译**

#./config

#make

#make install

编译需要root权限

**编译过程可能出现错误：**

**POD document had syntax errors at /usr/bin/pod2man**

　　原因分析：这是由于OpenSSL 1.0.1e 与 perl5.18 不兼容。  
　　解决方法：  
　　1、有人验证 安装perl 5.16 可以兼容，安排低版本perl 5.16可以解决问题。  
　　2、删除 pod2man文件：  
　　sudo rm /usr/bin/pod2man

**结果：**

在/usr/local/ssl目录下安装了ssl库

Openssl安装即完成。

**4、拷贝下面文件驱动到目标系统/usr/lib/**

libssl.a

libcrypto.a

openssl 移植完成.

### 编译wpa\_supplicant

**1、交叉编译**

进入wpa\_supplicant-0.7.3.tar.gz解压生成的wpa\_supplicant-0.7.3/wpa\_supplicant目录中：同样如果需要交叉编译，以arm为例，如果不需要则不用改变  
        #**cp defconfig        .config**  //首先创建配置文件

需要修改.config文件：

例如：

CONFIG\_DEBUG\_FILE //-f 路径 输出log信息

CONFIG\_IEEE80211R // 802.11r   
        #vim .config //需要交叉编译，修改配置文件  
            CC=arm-linux-gcc  -L/usr/local/ssl/lib/  
            #CC=/opt/brcm/hndtools-mipsel-uclibc/bin/mipsel-uclibc-gcc  
            CFLAGS += -I/usr/local/ssl/include  
            #CPPFLAGS += -I../src/include -I../../src/router/openssl/include  
            LIBS += -L/usr/local/ssl/lib  
**#make**

**错误1：**

../src/drivers/driver\_nl80211.c:25:31: fatal error: netlink/genl/genl.h: No such file or directory  
compilation terminated.  
make: \*\*\* [../src/drivers/driver\_nl80211.o] Error 1

安装libnl-dev 和libpopt-dev后，编译成功。

可以到主页下载：<http://www.infradead.org/~tgr/libnl/>，也可以使用git下载：git clone git://github.com/tgraf/libnl-1.1-stable.git

安装libnl

**编译并安装libnl：**

**cdlibnl-1.1-stable**

**./configure**

**make**

**sudo make install**

**如果还是不行，可能是.config文件 的CONFIG\_LIBNL32=Y被注释掉了**

**重新编译**

**错误2：**

**Has no member named ‘ssl’**

是由于编译找不到ssl模块。**安装openssl-devel**，即解决

安装方法如下：

centos：

yum install openssl-devel

ubuntu：

sudo apt-get install openssl

sudo apt-get install libssl-dev

[openssl-devel和openssl 是什么具体关系](http://www.cnblogs.com/pekkle/p/6568660.html)

Redhat在封装openssl的时候，把openssl分成了几个部分，执行码部分就是 openssl-1.0.0-27.el6.x86\_64 这种包。openssl-devel-1.0.0-27.el6.x86\_64 这个就是包含了头文件，头文件参考，某些库文件等跟开发相关的东西。mod\_ssl-2.2.15-26.el6.x86\_64 这个不是open ssl 本身的东西，是apache的模块。你在<http://www.openssl.org/source/>上下载的源码编译安装后得到的东西就是openssl-1.0.0-27.el6.x86\_64和openssl-devel-1.0.0-27.el6.x86\_64这两个包加在一起的内容。

另外，OpenSSL是分系列的，每个系列下再分版本 a b c d e…… 目前常用的是 0.9.8 1.0.0 1.0.1 三个系列。

RHEL 6.4 是openssl 1.0.0 系列的版本。RHEL 6.5 是 openssl 1.0.1 系列的版本。

Redhat 提供的openssl升级包的版本一般是 openssl-1.0.0-27.el6.X.x86\_64.rpm 这种。 Redhat 会把OpenSSL发布的补丁整合到现有版本中去，叫做backport。

例如，RHEL 6.4 目前的最新的OpenSSL就是2014-06-05发布的 openssl-1.0.0-27.el6\_4.4.x86\_64.rpm 和 openssl-devel-1.0.0-27.el6\_4.4.x86\_64.rpm   
RHEL 6.5 则是2014-08-13发布的 openssl-1.0.1e-16.el6\_5.15.x86\_64.rpm 和 openssl-devel-1.0.1e-16.el6\_5.15.x86\_64.rpm。因为不同系列的OpenSSL，存在的安全漏洞或者BUG不一定相同，所以版本要根据系列来判断。当然，如果你愿意手动编译安装openssl，那么也可以，只是注意相关软件的依赖。

经过编译后的wpa\_supplicant源程序可以看到两个主要的可执行工 具：

wpa\_supplicant和wpa\_cli。

**wpa\_supplicant**是核心程序，它和wpa\_cli的关系就是服务和客户端的关系：后台运 行wpa\_supplicant，

**wpa\_cli**用来搜索、设置、和连接网络。

**wpa\_passphrase：**网络配置可以使用wpa\_passphrase工具自动生成并添加到配置文件中。

将这三个可执行文件，copy到/usr/local/bin目录下，就可以在全局使用。

工具安装成功

## 编译文件.config

.config文件决定wpa\_supplicant编译后支持的功能。

# 第二章 使用

## 3.1 命令

<https://wiki.archlinux.org/index.php/WPA_supplicant#Troubleshooting>

该网页有整个使用过程的介绍

### 3.1.1 wpa\_supplicant

用法

wpa\_supplicant [-BddfhKLqqtuvW] [-P<pid file>] [-g<global ctrl>] \

[-G<group>] \

-i<ifname> -c<config file> [-C<ctrl>] [-D<driver>] [-p<driver\_param>] \

[-b<br\_ifname> [-MN -i<ifname> -c<conf> [-C<ctrl>] [-D<driver>] \

[-p<driver\_param>] [-b<br\_ifname>] [-m<P2P Device config file>] ...

选项

-b = 可选的桥接接口名称

-B = 后台运行

-c = 配置文件路径

-C = ctrl\_interface 参数（仅在-c不使用的时候使用）

-i = 接口名称

-d = 增加调试信息详细程度（-dd显示更多）

-D = 驱动名称

-f = 将日志输出到默认日志位置（通常为/ tmp）

-g = 全局ctrl\_interface

-G = 全局 ctrl\_interface group

-K = 包括密钥信息在调试中输出

-t = 调试信息添加时间戳

-h = 显示帮助文档

-L = 显示许可证

-p = 驱动程序参数

-P = PID 文件

-q = 减少调试信息详细程度（-qq更少）

-u = 驱动 DBus control interface

-v = 显示版本

-W = 在启动之前等待control interface monitor

-M = 开始描述匹配接口

-N = 开始描述新接口

-m = P2P Device的配置文件

驱动

nl80211 = Linux nl80211/cfg80211

wext = Linux wireless extensions (generic)

wired = wpa\_supplicant wired Ethernet driver

roboswitch = wpa\_supplicant Broadcom switch driver

bsd = BSD 802.11 support (Atheros, etc.)

ndis = Windows NDIS driver

通常示例

wpa\_supplicant -Dnl80211,wext -c/etc/wpa\_supplicant.conf -iwlan0 -B

### 3.1.2 wpa\_cli

wpa\_cli是一个基于文本的前端程序，用于与wpa\_supplicant 进行交互。它用于查询当前状态，更改配置，触发事件并请求交互式用户输入。

wpa\_cli支持两种模式：交互式和命令行。

**wpa\_cli命令：**

status = 获取当前WPA/EAPOL/EAP 状态

mib = 获取MIB变量

help = 显示帮助文档

interface [ifname] = 连接接口

level <debug level> = 改变debug等级

license = 显示wpa\_cli证书

logoff = IEEE 802.1X EAPOL state machine logoff

logon = IEEE 802.1X EAPOL state machine logon

set = 设置变量

pmksa = 显示PMKSA缓存

reassociate = 强制重新连接

reconfigure =强制wpa\_supplicant重新读取配置文件

preauthenticate <BSSID> = 强制预先认证

identity <network id> <identity> = 为SSID配置身份

password <network id> <password> =为SSID配置密码

pin <network id> <pin> = 为SSID配置引脚

otp <network id> <password> = 为SSID配置临时密钥

passphrase <network id> <passphrase> =为SSID配置私钥

bssid <network id> <BSSID> = 为SSID设置首选BSSID

list\_networks = 列举已经配置的网络

select\_network <network id> = 选择网络（关闭其他）

enable\_network <network id> = 启用网络

disable\_network <network id> = 禁用网络

add\_network = 添加网络

remove\_network <network id> = 删除网络

set\_network <network id> <variable> <value> =设置网络变量

get\_network <network id> <variable> = 获取网络变量

save\_config = 保存当前配置

disconnect = 断开连接，等待重新连接命令

scan = 扫描

scan\_results = 扫描结果

get\_capability <eap/pairwise/group/key\_mgmt/proto/auth\_alg> = 获取能力

terminate = 终止wpa\_supplicant

quit = 退出wpa\_cli

roam bssid = 漫游到指定BSS

问题：在wpa\_supplicant的官方文档readme中，包含的wpa\_cli命令是不全的。可以在wpa\_cli交互中，help获取全部命令

**wpa\_cli命令行选项**

wpa\_cli [-p<path to ctrl sockets>] [-i<ifname>] [-hvB] [-a<action file>] \

[-P<pid file>] [-g<global ctrl>] [command..]

-h = 显示使用文档

-v =显示版本信息

-a = 以守护进程方式运行

-B = 后台运行

默认套接字路径: /var/run/wpa\_supplicant

默认接口:在套接字路径中找到的第一个接口

### 3.1.3 wpa\_passphrase

使用wpa\_passphrase快速连接到其SSID已知的网络，wpa\_passphrase是一种命令行工具，可生成wpa\_supplicant所需的最小配置。

$ wpa\_passphrase MYSSID 密码

## 3.2 运行

根据这两篇博文：

<https://jingyan.baidu.com/article/ac6a9a5e4f411a2b653eace8.html>

<https://www.linuxidc.com/Linux/2011-04/34871.htm>

注意：在删除Networkmanager之前要记住自己的接口名称，一般有三个

**enp0s25（有线）**

**wlp3s0（无线）**

**lo（本机）**

wpa\_supplicant的使用需要无线接口名称。

如果wpa\_supplicant出现问题，可以使用网线上网，上网命令即为

**dhclient enp0s25**

运行wpa\_supplicant需要注意，在ubuntu中的Networkmanager会影响wpa\_supplicant的工作。

注意：很多操作需要root权限

**第一步：**

apt-get remove network-manager

**第二步：**

创建wpa\_supplicant的配置文件。

Vi /etc/wpa\_supplicant.conf

内容如下：

ctrl\_interface=/var/run/wpa\_supplicant //此意为与wpa\_cli通信的套接字的位置

update\_config=1 //允许wpa\_cli 的save\_config命令保存配置文件

这是最简单的配置文件

**第三步：**

终端执行 /etc/init.d/networking restart

        重启网卡

注意：由于之前卸载networkmanager，所以此时的无线网卡应该是关闭状态

**第四步：**

Wpa\_supplicant -B -Dnl80211 -c/etc/wpa\_supplicant.conf -iwlan0 -d

**第五步：**

终端输入Wpa\_cli。然后使用wpa\_cli的交互模式，添加网络。

注意：wpa\_supplicant和wpa\_cli必须为同一用户。否则连接不上

**第六步：**

此时已经连接网络，但是由于dhcp未获取IP地址。所以没有IP地址。

终端dhclient命令，获取IP地址

**第七步：（可能不需要）**

可能此时没有DNS功能，无法解析域名

**解决办法**

修改 /etc/resolv.conf 添加内容 nameserver 8.8.8.8 增加 DNS。

但是，上述方法会在重启后被清除，导致再次开机时需要重新配置，经过查阅网上资料，方法很多种，比较有效的就是，直接卸载掉开机重写该文件的 resolvconf。   
执行命令：sudo apt-get autoremove resolvconf

如果仍需要使用 resolvconf ，则可在卸载后，对 /etc/resolv.conf 加锁后再重新安装该软件，这样 resolvconf 就不会在开机时重写该文件。

# 文件加锁 不可写

sudo chattr +i /etc/resolv.conf

# 文件解锁 可写

sudo chattr -i /etc/resolv.conf

## 3.3 常见问题

### 错误1  ioctl

Successfully initialized wpa\_supplicant   
ioctl[SIOCSIWENCODEEXT]: Invalid argument   
ioctl[SIOCSIWENCODEEXT]: Invalid argument

原来在命令中

sudo wpa\_supplicant -B -D n180211,wext -i wlo1 -c /home/wpa\_supplicant.cfg

错把nl80211写成n180211，注意是数字1与字母l的区别。

### 错误2  current AP

wpa supplicant: No network configuration found for current AP   
说明配置文件写的不对，尤其从网上复制过来时常有看不见的tab字符等。   
解决办法：从Archlinux文档上复制了一份格式正确的配置，再改一改就OK了   
[Archlinux文档](https://wiki.archlinux.org/index.php/WPA_supplicant)

### 错误3  delete

ctrl\_iface exists and seems to be in use - cannot override it   
Delete ‘/var/run/wpa\_supplicant/wlo1’ manually if it is not used anymore   
Failed to initialize control interface ‘/var/run/wpa\_supplicant’.   
You may have another wpa\_supplicant process already running or the file was   
left by an unclean termination of wpa\_supplicant in which case you will need   
to manually remove this file before starting wpa\_supplicant again.

系统已经存在打开的多个wpa\_supplicant实例，执行一下sudo killall wpa\_supplicant杀死所有wpa\_supplicant即可。

### 错误4 PF\_UNIX failed

ctrl\_iface bind(PF\_UNIX) failed

不知道原因，但是将wpa\_supplicant的配置文件中wpa\_supplicant.conf中的：  
ctrl\_interface=/var/run/wpa\_supplicant  
注释掉，此时无法与wpa\_cli连接，但是wpa\_supplicant可以工作。

所以肯定是在于wpa\_cli通信的套接字处失败。但是具体原因不清楚

原博客地址：

<http://bbs.chinaunix.net/thread-3588469-1-1.html>

### 错误5： wpa\_cli：re-trying

wpa\_cli的用法：

运行时可能会出现如下的错误：

Could not connect to wpa\_supplicant - re-trying

这个错误可能是因为你的wpa\_supplicant进程没有启动起来造成的。

请参见如下的命令：

1、修改wpa\_supplicant.conf文件

将所有的内容都注释掉，只留下这一行语句：

　　# /etc/wpa\_supplicant.conf

　　ctrl\_interface=/var/run/wpa\_supplicant

　　#

2、启动wpa\_supplicant进程：

wpa\_supplicant -Dwext -iwlan0 -c/etc/wpa\_supplicant.conf &

或

wpa\_supplicant -B -Dwext -iwlan0 -c/etc/wpa\_supplicant.conf

3、然后ps -aux查看一下，是不是已经启动完成。

4、如果存在此进程，则你可以用wpa\_cli命令进行进一步的配置。

wpa\_cli -iwlan0

此时就不会再出现“Could not connect to wpa\_supplicant - re-trying" 的错误信息了。

而且会得到一个新的提示符" > "

### 错误5： wpa\_cli：re-trying

如果wpa\_supplicant是使用root用户启动，那么wpa\_cli也要使用root用户。

即wpa\_cli和wpa\_supplicant需要同一用户下启动

### 错误6：libssl.so.1

./wpa\_supplicant: error while loading shared libraries: libssl.so.1: cannot open shared object file: No such file or directory

找不到之前编译的libnl获得的库。

只要将wpa\_supplicant找到咱们的库就ok了：

方法1： 将/usr/local/ssl/lib下的libssl.so.x.x.x 和 libcrypto.so.xxx 拷贝到/lib目录下即可。

方法2： export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/usr/local/ssl/lib

方法3： 在 /etc/ld.so.conf 文件中添加库的搜索路径。(或者在/etc/ld.so.conf.d 下新建一个.conf文件，将搜索路径一行一个加入)

注意：如果在/usr/local/ssl/lib目录下没有该文件，查看一下libnl是否install。

使用上面任何一种方法就可以成功运行wpa\_supplicant了。

## 3.4 脚本的使用

wpa.sh脚本四种功能

start：执行wpa\_supplicant

stop：终止wpa\_supplicant

dhcp：获取IP

run：查看wpa\_supplicant是否正在运行

# 第三章 配置文件

Wpa\_supplicant.conf配置文件的简单介绍。详细请看官网文档： 

## 3.1 全局参数

1、ctrl\_interface=/var/run/wpa\_supplicant

//与wpa\_cli通信的接口套接字所在路径

2、eapol\_version=1

//IEEE802.1X/EAPOL版本，默认版本为1

3、ap\_scan=1

//wpa\_Supplicant请求驱动扫描AP，然后选择一个合适的AP

1：wpa\_supplicant初始化扫描，选择合适的已经启用的AP。如果没有符合的AP，则会根据配置文件创建一个连接

0：不会试图连接AP。如果使用有线以太网驱动时，必须为0

2：与0类似，但是会使用安全的连接方法和SSID连接AP

4、fast\_reauth=1

//EAP快速重新认证

5、load\_dynamic\_eap=/usr/lib/wpa\_supplicant/eap\_md5.so

//动态EAP方法，默认为静态，所以不是必须的

6、scan\_cur\_freq

//扫描当前频率

0：扫描当前有效的工作频率（默认）

1：

7、preassoc\_mac\_addr=0

//预先连接时的MAC地址政策

0：使用永久MAC地址

1：使用随机MAC地址

2：随机MAC地址，但是OUI相同

## 3.2 credential block

Temporary：证书是临时的，不保存

Priority：优先级，默认为0

Pcsc：使用PC/SC 和 SIM/USIM card

Username：选择互联网络的用户名

Password：选择互联网络的密码

……

## 3.3 network block

用于连接AP

disable：

0：默认。该网络可以被使用

1：该网络被禁用。可以通过wpa\_cli启用

id\_str：

网络标识字符串

ssid：网络名

scan\_ssid：

0：默认。不发送探测帧进行ssid扫描

1：发送探测帧扫描

bssid：可选。如果配置，则只能连接配置的BSSID

priority：优先级，默认为0

mode：802.11工作模式

0：基础模式 默认

1：IBSS

2：AP

’注意：IBSS只能在key\_mgmt NONE和WPA-PSK情况下使用

frequency：IBSS的通道频率

pbss：是否是个人基础服务集（802.11ad独有）

0：不是

1：是

2：不关心

scan\_freq：频率列表用于扫描

freq\_list：允许的频率列表

bgscan：后台扫描

proto：能够接受的协议列表（默认是WPA RSN）

WPA = WPA/IEEE 802.11i/D3.0

RSN = WPA2/IEEE 802.11i

key\_mgmt：可接受的认证密钥管理协议列表

# WPA-PSK = WPA pre-shared key (this requires 'psk' field)

# WPA-EAP = WPA using EAP authentication

# IEEE8021X = IEEE 802.1X using EAP authentication and (optionally) dynamically

# generated WEP keys

# NONE = WPA is not used; plaintext or static WEP could be used

# WPA-NONE = WPA-None for IBSS (deprecated; use proto=RSN key\_mgmt=WPA-PSK

# instead)

# FT-PSK = Fast BSS Transition (IEEE 802.11r) with pre-shared key

# FT-EAP = Fast BSS Transition (IEEE 802.11r) with EAP authentication

# WPA-PSK-SHA256 = Like WPA-PSK but using stronger SHA256-based algorithms

# WPA-EAP-SHA256 = Like WPA-EAP but using stronger SHA256-based algorithms

# SAE = Simultaneous authentication of equals; pre-shared key/password -based

# authentication with stronger security than WPA-PSK especially when using

# not that strong password

# FT-SAE = SAE with FT

# WPA-EAP-SUITE-B = Suite B 128-bit level

# WPA-EAP-SUITE-B-192 = Suite B 192-bit level

# OSEN = Hotspot 2.0 Rel 2 online signup connection

默认为WPA-PSK WPA-EAP

ieee80211w：管理帧保护是否开启

0：关闭（默认）

1：可选

2：必需的

auth\_alg：802.11认证方式

OPEN：open认证

SHARED：共享密钥

LEAP

pairwise：可接受的WPA的成对算法列表。（默认CCMP和TKIP）

CCMP

TKIP

NONE

group: 可接受的WPA的组成对算法列表。（默认CCMP、TKIP、WEP104、WEP40）

CCMP

TKIP

WEP104

WEP40

psk：WPA预共用密钥 256bit

mem\_only\_psk：是否保存PSK到内存

0：允许保存到配置文件

1：不允许

eapol\_flags：EEE 802.1X/EAPOL选项

认证成功之后必须设置为0

macsec\_policy: IEEE 802.1X/MACsec选项

mixed\_cell：

proactive\_key\_caching

peerkey

wpa\_ptk\_rekey

group\_rekey

proactive\_key\_caching：（okc）使能

eap：eap方法列表

identity：eap身份

anonymous\_identity：匿名eap身份

password：eap密码

ca\_cert：ca证书路径（网络获取）

ca\_path：ca证书目录路径

client\_cert：客户证书路径

private\_key：私有密钥路径

private\_key\_passwd：

……

## 3.4 Fast Session Transfer

fst\_group\_id

fst\_priority

fst\_llt

## 3.5 例子

### 3.5.1 PSK认证

network={

ssid="example"

proto=WPA

key\_mgmt=WPA-PSK FT-PSK（802.11r）

pairwise=CCMP TKIP

group=CCMP TKIP WEP104 WEP40

psk=06b4be19da289f475aa46a33cb793029d4ab3db7a23ee92382eb0106c72ac7bb

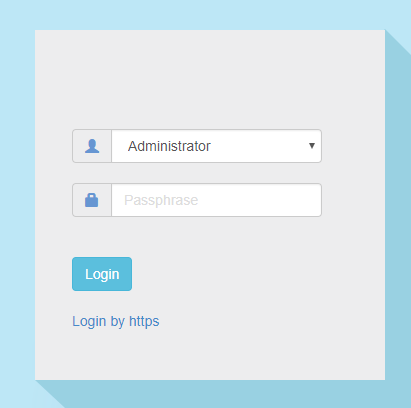
priority=2

}

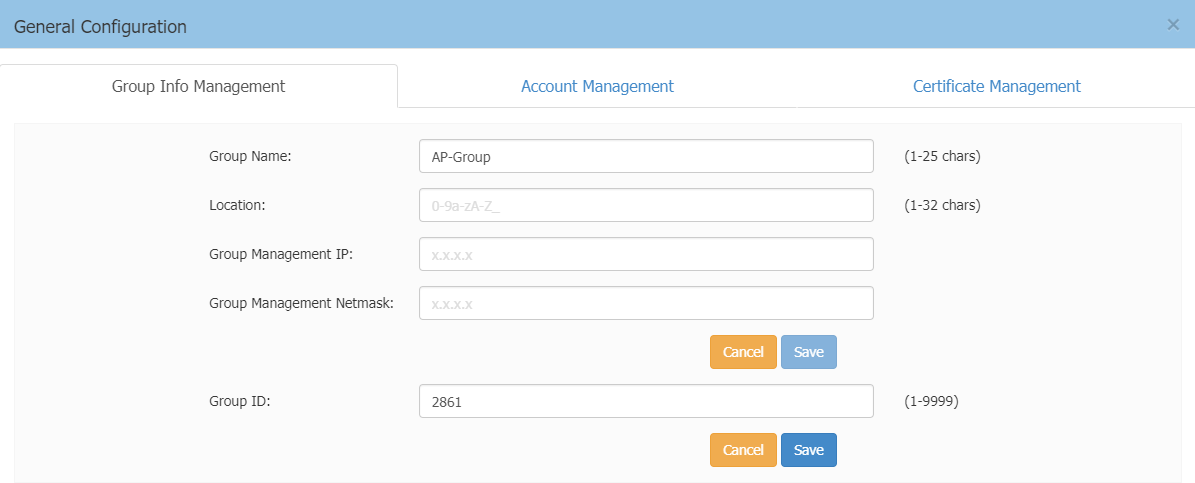
# 第四章 测试

## 4.1、环境搭建

1. 两台使用R3.0版本的AP，连接到相同网段
2. 登录页面

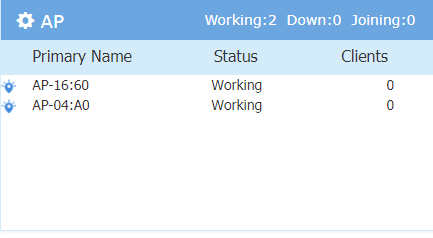


1. 设置相同的group ID

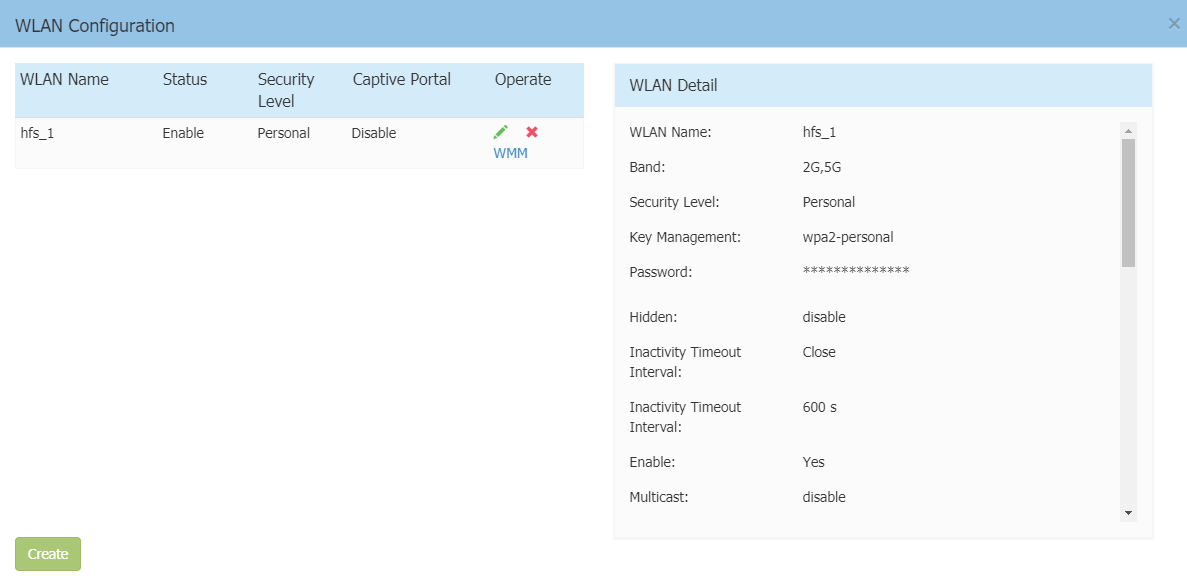


其他不用在意，两台AP的group ID 必须相同。

这样，两台设备就组成了AP组，登录任一AP的设置页面，都会跳转到相同的设置页面。



1. 设置SSID，password等参数



5、环境搭建完成

## 4.2、主动漫游

思路：通过使用wpa\_cli的命令bssid控制wpa\_supplicant选择AP进行连接，实现主动切换AP，完成漫游

过程：

1. 编写配置文件wpa\_supplicant.conf

ctrl\_interface=/var/run/wpa\_supplicant

update\_config=1

network={

ssid=”hfs\_1”

psk=”12345678”

}

出现的问题：

CTRL-EVENT-SSID-TEMP-DISABLED：由于认证没有通过，网络暂时禁用

修改bssid的方式进行连接，总是出现该问题

而采用disable enable的方式就能连接，无问题。

难道是上下线请求对其有影响，但是使用bssid也会出现disconnect。应该都是相同的啊

经过试验：由于在启动wpa\_supplicant之前，执行了/etc/init.d/networking restart命令，去除后即可连接成功。

肯定是在认证过程中出现的问题：可能出现的原因

1. AP收到认证报文之后，未发送认证回应（不太可能）

不成立：

a:hostapd中的认证回应中并没有超时跳出等问题，只要接收到正确请求，就会回应

b:手机能连接AP，无任何问题，证明AP功能是正常的

c:在使用抓包工具，抓包中能抓到认证回应报文，只不过出现的次数比较少

1. AP发送认证回应，而wpa\_supplicant为收到认证回应报文或者正确解析报文。所以应该是sta收到报文后，因为AP被删除，所以断开

很有可能：

a：正常wpa\_supplicant发送认证请求之后会，建立AP工作站，然后处理认证回应报文。在认证失败的情况下，

1. wpa\_supplicant未发送连接请求报文（不成立）

wpa\_supplicant成功和失败打印log对比

失败：

nl80211: Authentication request send successfully

nl80211: Event message available

nl80211: Drv Event 19 (NL80211\_CMD\_NEW\_STATION) received for wlp3s0

nl80211: New station 34:e7:0b:00:04:a1

nl80211: Event message available

nl80211: Drv Event 20 (NL80211\_CMD\_DEL\_STATION) received for wlp3s0//为什么要多一个删除工作站？？？？

nl80211: Delete station 34:e7:0b:00:04:a1

nl80211: Event message available（接收到认证回应报文后才会执行）

nl80211: Drv Event 37 (NL80211\_CMD\_AUTHENTICATE) received for wlp3s0

nl80211: MLME event 37; timeout with 34:e7:0b:00:04:a1 //此处不同，timeout

wlp3s0: Event AUTH\_TIMED\_OUT (14) received /\*超时可能是早期连接的，所以会出现已经连接成功，后断开。发送 CONN\_FAILED\*/

wlp3s0: SME: Authentication timed out//在认证报文发送之后开启计时

wlp3s0: Radio work 'sme-connect'@0x18c6580 done in 0.352394 seconds

Added BSSID 34:e7:0b:00:04:a1 into blacklist

成功：

nl80211: Authentication request send successfully //发送认证请求报文（由wpa\_cli命令发送的；wpa\_priv\_receive函数）;后面log都是回调

nl80211: Event message available

nl80211: Drv Event 19 (NL80211\_CMD\_NEW\_STATION) received for wlp3s0

nl80211: New station 94:65:2d:33:da:ea

nl80211: Event message available（在这里就接收到报文了）

nl80211: Drv Event 37 (NL80211\_CMD\_AUTHENTICATE) received for wlp3s0

nl80211: MLME event 37 (NL80211\_CMD\_AUTHENTICATE) on wlp3s0(e0:06:e6:c5:55:48) A1=e0:06:e6:c5:55:48 A2=94:65:2d:33:da:ea //此处认证不同；此处是上一个函数的调用

nl80211: MLME event frame - hexdump(len=30): b0 00 3a 01 e0 06 e6 c5 55 48 94 65 2d 33 da ea 94 65 2d 33 da ea 80 83 00 00 02 00 00 00

nl80211: Authenticate event（驱动调用上层函数wpa\_supplicant\_event进行处理）

wlp3s0: Event AUTH (11) received //wpa\_supplicant\_event(event.c);调用sme\_event\_auth发送连接请求。此过程绝对是接收到认证回应才会执行

wlp3s0: SME: Authentication response: peer=94:65:2d:33:da:ea auth\_type=0

auth\_transaction=2 status\_code=0

SME: Authentication response IEs - hexdump(len=0): [NULL]

wlp3s0: set\_disable\_max\_amsdu: -1

报文：b0 00 3a 01 \*e0 06 e6 c5 55 48 \* 94 65 2d 33 da ea \* 94 65 2d 33 da ea \*80 83 \*

00 00\* 02 00 \*00 00

这是AP94 65 2d 33 da ea发送给工作站e0 06 e6 c5 55 48的open认证的回应报文。；

综上所述：

1. 由于wpa\_supplicant的删除AP的行为，导致未接收到报文？
2. 由于AP未发送认证回应报文？

通过抓包情况来看，两种情况好像都有

首先，如果wpa\_supplicnat走到发送连接请求，可以抓包到认证回应，反之，抓不到。（大概率这样，还是能抓到的）

还有就是，如果wpa\_supplicnat走到发送连接请求，之前肯定没有删除AP情况出现。

难道是因为认证请求次数过多引起的？

手机连接AP抓包测试，看一下认证请求次数是否也是同样多。

实验：

确定：认证请求只发送一次

问题：删除AP，为什么会发生？谁下的命令？

假设：wpa\_supplicant发送认证请求时，发送多个认证请求报文，并创建AP

然后AP发送认证回应报文，wpa\_supplicant接收后发送连接请求报文。

AP发送连接回应报文。

而此时出现的问题，在于，删除AP的行为经常出现在关联认证过程中。

所以说，wpa\_Supplicant只有两个异常行为，

1. 发送多次认证请求，连接请求
2. 删除AP（只要执行该操作，一定timeout）

AP对于多次接收到相同的报文如何处理（怀疑是只回应一份）

由上述过程可以看出，失败主要因为NL80211\_CMD\_DEL\_STATION将工作站删除。

定位：

什么原因导致调用NL80211\_CMD\_DEL\_STATION命令？

所有的NL80211\_CMD命令都是有process\_global\_event函数进行处理。该函数在初始化过程中与global->nl\_cb进行了绑定

在哪个报文过程中出现的？

在wpa\_supplicant.c wpa.c wpa\_ft.c event.c 中添加打印信息，分析log

问题：802.11r到底如何工作的？

怎样实现wpa\_Supplicant的主动漫游？

通过log分析。802.11r可以支持，但是修改bssid scan的方法不是漫游

wpa\_cli使用roam addr 命令可以实现漫游

## 4.3、OKC

pmk caching：在单个AP下能，进行快速认证

okc：可以在同一网络下的多个AP间，进行快速认证

okc会在association 或者 reassociation request报文中加入PMKID，以便AP在okc中查找对应的PMKID。

所以省掉了1x认证获取PMK的过程。

经过测试，wpa\_supplicant只要在network中加入proactive\_key\_caching=1.即支持OKC功能。

但是出现过一次问题，即发送四次握手报文1时，出现问题无法获得PMK。

重启电脑后，恢复正常。

## 4.4、802.11R

分为两种认证方式：

over-the-air：将密钥握手过程放在了认证过程

over-the-ds：发送remote帧给AP，AP去找之前认证的AP。

FT-PSK

在抓包中，beacon帧中没有FT capabilities字段。所以实验无法进行

原因：

1. 在OV中修改wlan时，要重新apply to device
2. 在OV中修改配置后，需要重启wam

结果：支持802.11r功能

FT-EAP

结果：支持802.11r

# 第五章 wpa\_supplicant代码

## 5.1 发送认证请求

wpas\_select\_network\_from\_last\_scan

wpa\_supplicant\_pick\_network（选择一个合适的ssid）

wpa\_supplicant\_select\_bss

wpa\_scan\_res\_match

wpa\_supplicant\_ssid\_bss\_match

wpa\_supplicant\_connect（连接）

wpa\_supplicant\_associate发送连接

sme\_authenticate

radio\_add\_work(设置回调sme\_auth\_start\_cb)

sme\_auth\_start\_cb

wpas\_valid\_bss\_ssid

wpas\_valid\_bss

wpas\_valid\_ssid

sme\_send\_authentication

pmksa\_cache\_set\_current

wpa\_supplicant\_set\_suites（设置参数）

wpa\_sm\_set\_param （设置参数）

wpa\_sm\_set\_assoc\_wpa\_ie\_default

wpa\_key\_mgmt\_wpa\_psk

wpa\_sm\_set\_pmk（设置PMK）

wpa\_sm\_set\_ft\_params

sme\_auth\_handle\_rrm

wpas\_build\_ext\_capab

wpa\_supplicant\_cancel\_sched\_scan（停止正在进行的扫描）

wpa\_supplicant\_cancel\_scan（关闭扫描请求）

wpa\_clear\_keys（清除为驱动配置的密钥）

wpa\_supplicant\_set\_state设置状态WPA\_AUTHENTICATING

wpa\_supplicant\_stop\_autoscan关闭自动扫描

wpa\_supplicant\_rsn\_supp\_set\_config

wpa\_sm\_set\_config

wpa\_supplicant\_initiate\_eapol

wpa\_drv\_authenticate（调用底层驱动）

**driver\_nl80211\_authenticate**

**wpa\_driver\_nl80211\_authenticate**

//发送认证请求报文

## 5.2 漫游

wpa\_cli发送roam命令。

wpa\_supplicant程序中的wpa\_supplicant\_ctrl\_iface\_process函数进行处理，

然后调用wpa\_supplicant\_ctrl\_iface\_roam函数。

wpa\_supplicant\_ctrl\_iface\_roam

hwaddr\_aton//检查MAC地址格式是否正确

wpa\_bss\_get//根据MAC地址获取bss

wpa\_supplicant\_connect//根据当前的ssid 和指定的bss进行正常的连接

wpa\_supplicant\_associate发送连接

sme\_authenticate

radio\_add\_work(设置回调sme\_auth\_start\_cb)

sme\_auth\_start\_cb

wpas\_valid\_bss\_ssid

wpas\_valid\_bss

wpas\_valid\_ssid

sme\_send\_authentication

pmksa\_cache\_set\_current//根据pmksa或者bssid，或者okc标志获取pmksa caching’

pmksa\_cache\_get

pmksa\_cache\_get\_opportunistic

获取pmksa（包含pmkid）

eapol\_sm\_notify\_pmkid\_attempt（使用pmksa）

wpa\_supplicant\_set\_suites（设置参数）

wpa\_sm\_set\_param （设置参数）

wpa\_sm\_set\_assoc\_wpa\_ie\_default //生成rsn ie

wpa\_gen\_wpa\_ie

wpa\_gen\_wpa\_ie\_rsn

wpa\_key\_mgmt\_wpa\_psk

wpa\_sm\_set\_pmk（设置PMK）

wpa\_sm\_set\_ft\_params

sme\_auth\_handle\_rrm

wpas\_build\_ext\_capab

wpa\_supplicant\_cancel\_sched\_scan（停止正在进行的扫描）

wpa\_supplicant\_cancel\_scan（关闭扫描请求）

wpa\_clear\_keys（清除为驱动配置的密钥）

wpa\_supplicant\_set\_state设置状态WPA\_AUTHENTICATING

wpa\_supplicant\_stop\_autoscan关闭自动扫描

wpa\_supplicant\_rsn\_supp\_set\_config

wpa\_sm\_set\_config

wpa\_supplicant\_initiate\_eapol

wpa\_drv\_authenticate（调用底层驱动）

**driver\_nl80211\_authenticate**

**wpa\_driver\_nl80211\_authenticate**

//发送认证请求报文

问题：

wlp3s0: nl80211: MLME command failed (auth): ret=-2 (No such file or directory)

CTRL-EVENT-AUTH-REJECT：拒绝试图认证的连接。auth\_type=2 auth\_transaction=2 status\_code=1

代表收到认证回应报文，然后出现了认证拒绝。（？？？？？？）

记录时间：11:22

漫游a1成功：但是出现ip地址没过去。可是使用sta\_list：有IP地址啊。为什么不能传输？

漫游61成功：未重新获取IP，维持原有IP。即可发送数据包

问题原因：两个AP的网关不同。请将AP放到同一交换机下

**driver\_nl80211\_authenticate**

wpa\_driver\_nl80211\_authenticate 通知驱动进行认证，如果失败，不会发送认证请求

send\_and\_recv\_msgs

send\_and\_recv(与内核通信)（返回的错误时No such file or directory）

也就是说在发送认证之前就失败了。

send\_and\_recv函数：

nl\_cb\_clone：克隆一个回调函数

nl\_send\_auto\_complete：最后确定并发送netlink信息

// netlink支持用户进程和内核相互交互（两边都可以主动发起），同时还支持用户进程之间相互交互（虽然这种应用通常都采用unix-sock）

ret=-2：系统回调被中断

A signal occurred before any data was transmitted;

**一个正常的漫游过程：**

Considering connect request: reassociate

wpas\_valid\_bss\_ssid

PMKSA cache search

set AP RSN IE

Stored MDIE and FTIE from (Re)Association Response

SME: Trying to authenticate with

State: COMPLETED -> AUTHENTICATING

Authentication request send successfully // 25:159547usec

Drv Event 20 (NL80211\_CMD\_DEL\_STATION)

Drv Event 39 (NL80211\_CMD\_DEAUTHENTICATE)

Drv Event 48 (NL80211\_CMD\_DISCONNECT)

Drv Event 19 (NL80211\_CMD\_NEW\_STATION)

Drv Event 36 (NL80211\_CMD\_REG\_CHANGE)

Drv Event 37 (NL80211\_CMD\_AUTHENTICATE)

Event AUTH (11) received

Authentication request send successfully

FT: Installing PTK to the driver

nl80211: Association request send successfully

Drv Event 38 (NL80211\_CMD\_ASSOCIATE)

FT: Completed successfully

State: ASSOCIATING -> ASSOCIATED

State: ASSOCIATED -> COMPLETED // 25:292054

## 5.3、四次握手（1x）

**根wpa\_sm\_rx\_eapol**

wpa\_supplicant\_process\_1\_of\_4

wpa\_sm\_set\_state(sm, WPA\_4WAY\_HANDSHAKE);

wpa\_supplicant\_parse\_ies（如果有RSN信息元素，则获取）

wpa\_supplicant\_get\_pmk //获取pmk

首先利用RSN中的PMK，在pmk cache中寻找。

eapol\_sm\_get\_key //从EAP中获取key（msk）sm->eapKeyData

wpa\_supplicant\_key\_mgmt\_set\_pmk

pmksa\_cache\_add //将pmk添加到cache。返回entry

rsn\_pmkid(pmk, pmk\_len, aa, spa, entry->pmkid, //计算PMKID，计算结果要和wpa\_supplicant\_parse\_ies获取的pmkid相同

wpa\_key\_mgmt\_sha256(akmp)) （计算的PMKID需要和AP计算的进行对比）

random\_get\_bytes //生成SNonce

wpa\_derive\_ptk //生成ptk

wpa\_supplicant\_send\_2\_of\_4 //发送EAPOL-KEY（2/4）

wpa\_sm\_alloc\_eapol //生成回应报文

wpa\_eapol\_key\_send //发送报文

**根wpa\_sm\_rx\_eapol //接收出来EAPOL-KEY(3/4)**

wpa\_supplicant\_verify\_eapol\_key\_mic //检验报文的MIC

wpa\_supplicant\_process\_3\_of\_4 //处理报文3

wpa\_supplicant\_send\_4\_of\_4 //发送报文4