Guest WiFi & GRE Tunnel Analysis on COX 4131

Created by Paul P Joby, last modified just a moment ago

Overview

Analyzing the working of GRE Tunnel and Guest Wi-Fi on 4131 and Setting up on RDK-B RPi.

Guest WiFi on COX 4131

Utopia service runs **/etc/utopia/service.d/service_multinet/handle_gre.sh** script which is responsible for bring up the GRE Tunnel. This script creates the two bridges for 2 different VLAN_ID (One for guest Wi-Fi and another one for Public/COX Wi-Fi). For example if the VLANID is 899 for the public/cox Wi-Fi then VLANID for guest Wi-Fi is obtained by prefixing a 3, i.e. 3899 will be the VLAN ID for the guest Wi-Fi interfaces. In this case following are bridges that would get created.

- hsbr_1_3899 (for Guest Wi-Fi interfaces)
- hsbr_1_899 (for Public/cox Wi-Fi interfaces)

The WLAN interfaces corresponding to the Public and Guest WIFI's are added to the **hsbr_1_899** and **hsbr_1_3899** bridges respectively.

Script also establishes a GRE Tunnel with the Remote endpoint and GRE tunnel interfaces are added to the bridge as well. Now whenever an client connects to the Guest or Public Wi-Fi interface, all the packet that reaches the Public and Guest WLAN interface get flooded to the corresponding bridges to which those interfaces were added and this is then send through the GRE tunnel to the Remote or **Wireless Access Gateway (WAG)** endpoint. Now the client makes a request for an IP using DHCP. This DHCP packets from the client are snooped and gets appended with DCHP option 82 (Relay Agent Information).

Steps involved

- DCHP Discover Packets from client are sent to the WAG via the GRE Tunnel.
- DHCP Server at the WAG endpoint responds with an DCHP Offer.
- DCHP Request Packets from client are sent to the WAG via the GRE Tunnel.
- DHCP Server at the WAG endpoint responds with an DCHP Ack to Client device via GRE Tunnel.
- At this point client obtains the IP Address
- Then clients initiates a Captive Portal Detection attempt by getting an HTTP URL
- which gets temporarily redirected to the cox Wi-Fi captive portal page.
- Incase of COX 4131 HTTP Temporary Redirection is used for prompting the captive portal page to the user/client.
- Once client performs authentication, that particular device/client is give internet access.

Captive Portal URL in case of COX 4131

- https://cwifi.dev.cox.net/?mac-address=00:24:D7:AE:B5:F8&ap-mac=6E:55:E8:9B:50:D6&ssid=SSID3-2.4&vlan=3899&nas-id=PSP6WAGB01_DEV.at.at.cox.net&block=false&unique=$HASH

Captive Portal Detection URLs used by Various Clients

- http://www.msftconnecttest.com/connecttest.txt

Public/COX Wi-Fi SSIDs

- Device.WiFi.SSID.13.
- Device.WiFi.SSID.14.

Guest Wi-Fi SSIDs

- Device.WiFi.SSID.3.
- Device.WiFi.SSID.4.
- Device.WiFi.SSID.5.
- Device.WiFi.SSID.6.
- Device.WiFi.SSID.7.
- Device.WiFi.SSID.8.

SSIDs 3, 5, 7 and 13 are 2.4GHz

SSIDs 4, 6, 8 and 14 are 5GHz

Data Models Used

- Device.X_COMCAST-COM_GRE.Tunnel.1.DSCPMarkPolicy
- Device.X_COMCAST-COM_GRE.Tunnel.1.PrimaryRemoteEndpoint
- Device.X_COMCAST-COM_GRE.Tunnel.1.SecondaryRemoteEndpoint

Setting Primary Remote Endpoint

```
dmcli eRT setv Device.X_TCH_COM_GRE.Tunnel.1.PrimaryRemoteEndpoint string 174.68.234.126
```

Enabling Public WiFi on COX 4131

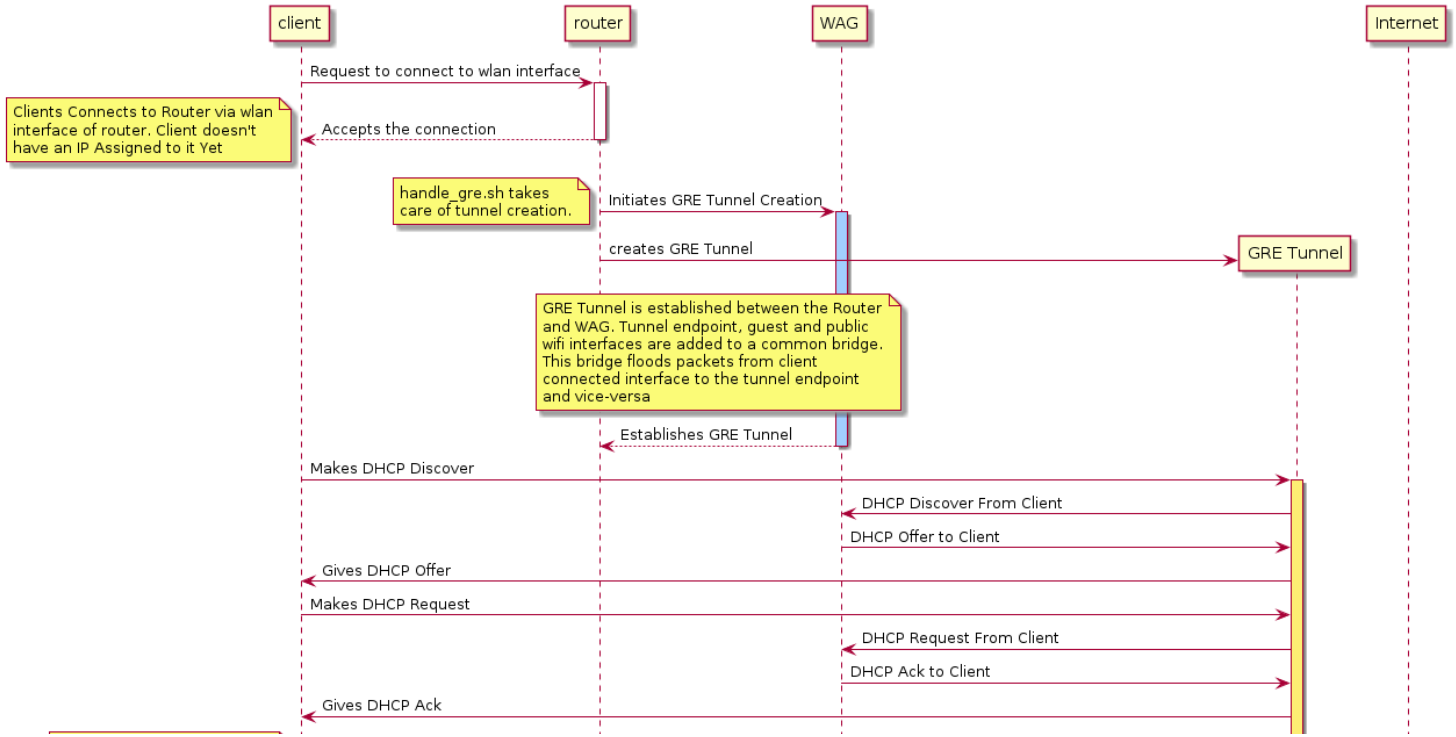
SSID 13 and 14 are used for Public/COX Wi-Fi, which could be enabled by the following commands.

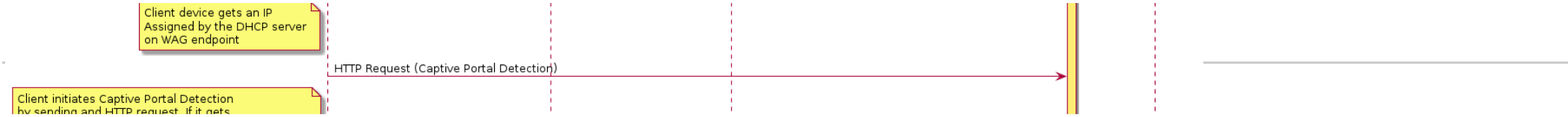
```
dmcli eRT setv Device.WiFi.SSID.13.Enable bool true
dmcli eRT setv Device.WiFi.SSID.13.X_CISCO_COM_EnableOnline bool true
dmcli eRT setv Device.WiFi.SSID.14.Enable bool true
dmcli eRT setv Device.WiFi.SSID.14.X CISCO COM EnableOnline bool true
```

```
root@Business:~# brctl show hsbr_1_899
bridge name     bridge id               STP enabled     interfaces
hsbr_1_899              8000.2278e19390bf       no              gretap1.899
                                                        wl0.6
                                                        wl1.6

root@Business:~# brctl show hsbr_1_3899
bridge name     bridge id               STP enabled     interfaces
hsbr_1_3899             8000.2278e19390bf       no              gretap1.3899
                                                        wl0.1
                                                        wl0.2
                                                        wl0.3
                                                        wl1.1
                                                        wl1.2
                                                        wl1.3
```

Guest WiFi Setup with RDKB RPi





Setting up Guest WiFi & GRE Tunnel on RDK-B RPi

In RDKB RPi the Guest WiFi and Public WiFi interfaces are added to brlan2 and brlan3 bridges respectively. Public WiFi interfaces are given a VLANID of 103 and Guest WiFi interfaces are given a VLANID of 102. Currently RPI only has one WiFi interface i.e. wlan0, which is acting as the Private WiFi 2.4GHz (SSID 1). So we will be moving this wlan0 interface to brlan2 bridge by setting the bridge value in **/nvram/hosptapd0.conf.** After which hostapd service is restarted.

```
systemctl restart hostapd
```

Now any packets received from devices connecting to wlan0 will be flooded to **brlan2** from which it will send to the remote endpoint via **gretap0.102** interface.

```
root@RaspberryPi-Gateway:~# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.000000000000       no
br106           8000.000000000000       no
br403           8000.000000000000       no
brebhaul                8000.000000000000       no
brlan0          8000.000000000000       no
brlan2          8000.2a15f4753e1d       no              gretap0.102
                                                        wlan0
brlan3          8000.2a15f4753e1d       no              gretap0.103
root@RaspberryPi-Gateway:~#
```

Setting Up GRE Tunnel on RDKB RPi

```
# Adding SSID 1 to Tunnel's Local Interface instead SSID 5
psmcli set dmsb.hotspot.tunnel.1.interface.1.LocalInterfaces Device.WiFi.SSID.1.
psmcli get dmsb.hotspot.tunnel.1.interface.1.LocalInterfaces
psmcli set dmsb.hotspot.gre.1.LocalInterfaces Device.WiFi.SSID.1.,Device.WiFi.SSID.6.,Device.WiFi.SSID.9.,Device.WiFi.SSID.10.
psmcli get dmsb.hotspot.gre.1.LocalInterfaces
  Device.WiFi.SSID.1.,Device.WiFi.SSID.6.,Device.WiFi.SSID.9.,Device.WiFi.SSID.10.

# Restart CcspPandMSsp
dmcli eRT getv Device.X_COMCAST-COM_GRE.Tunnel.1.
systemctl restart CcspPandMSsp
```

```
dmcli eRT setv Device.DeviceInfo.X_COMCAST_COM_xfinitywifiEnable bool true
```

> ⚠ **Note : Prevent Addition of DHCP Option 82**
>
> *In RDKB **CcspHotspot** component snoops on DHCP packets and appends the DHCP Option 82, i.e. Relay Agent Information. This was causing packet corruption by adding an incorrect length in the IP Header and hence packet was not accepted by the DHCP server and preventing the DHCP from working. So as a temporary fix DHCP packet snooping and modification could be disabled the following two commands.*
>
> *sysevent set snooper-remote-enable 0*
> *sysevent set snooper-circuit-enable 0*

DHCP relay agent information option (option 82)

The DHCP relay agent information option (option 82) enables you to include additional useful information in the client-originated DHCP packets that the DHCP relay forwards to a DHCP server. DHCP Option 82 contains Circuit Id (Routers MAC address) and Remote ID (MAC address of the client device). which can be disabled by using the below commands.

Sample:

Agent-Information Option 82, length 54:
Circuit-ID SubOption 1, length 33: b8:27:eb:ac:29:0e;RPi3_RDKB-AP0;s
Remote-ID SubOption 2, length 17: c8:3d:de:ce:aa:4f

2. Creating WAG Endpoint

2.1 Setting up SoftGREd

SoftGREd is a Linux open source solution for L2oGRE, used to Auto Provisioning for Tunnel GRE (Generic Routing Encapsulation). Kindly use SoftGREd Setup Guide to setup and install and run SoftGREd on the WAG server that we created.

Running SoftGREd on WAG Ubuntu Server

```
sudo brctl addbr brtun
sudo brctl addif brtun enp0s8
```

```
sudo ip link set enp0s8 up
sudo ip link set gre0 up
sudo brctl show
# Start the SoftGREd
sudo docker start softgre_container
```

2.2 Setting up DHCP Server

For DCHP server we will be using isc-dhcp-server. It can be installed on the ubuntu machine (that will be acting as our WAG) by the following command.

```
sudo apt install isc-dhcp-server
```

Once the isc-dhcp-server is install we will modify or create the configurations for DHCP server, update **/etc/dhcp/dhcpd.conf** and **/etc/default/isc-dhcp-server** as shown below.

/etc/dhcp/dhcpd.conf

```
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 10.0.2.255;
option captive-portal-rfc8910 code 114 = text;
log-facility local0;
#allow unknown-clients;
authoritative;

subnet 10.0.2.0 netmask 255.255.255.0 {
      #For bootp support
      #range dynamic-bootp 10.0.2.10 10.0.2.100;
      range 10.0.2.15 10.0.2.100;
      option routers 10.0.2.1;
      option domain-name-servers 8.8.8.8;
      # option domain-name-servers 10.0.2.1; #if using this need to run a dns server in using dnsmasq
      option captive-portal-rfc8910 "http://10.0.2.1:8000/";
}
```

```
...
INTERFACES="brtun"
...
```

2.3 General Settings to be done on Ubuntu WAG Server

Run the following commands on the ubuntu machine for avoiding iptables restrictions and for enabling ipv4 forwarding.

Setting on Ubuntu WAG Server

```
# This  is to prevent iptables from interfering with bridge
echo 0 > /proc/sys/net/bridge/bridge-nf-call-iptables

# iptables rules to allow internet access.
sudo iptables -I INPUT -j ACCEPT #for packets to work
sudo iptables -P FORWARD ACCEPT #for forwarding to work
sudo iptables --table nat -A POSTROUTING -o brtun -j MASQUERADE
sudo iptables --table nat -A POSTROUTING -o enp0s3 -j MASQUERADE

# You have to enable forwarding:
sysctl -w net.ipv4.ip_forward=1

# The above line will work immediately. but will be gone the next time you reboot your system.
# for a persisten setting, put the following into /etc/sysctl.conf (but it will only take effect after rebooting):
# net.ipv4.ip_forward=1
```

2.4 Introduction to Nodogsplash(NDS)

Nodogsplash is a Captive Portal that offers a simple way to provide restricted access to the Internet by showing a splash page to the user before Internet access is granted. It was derived originally from the codebase of the Wifi Guard Dog project. Nodogsplash is released under the GNU General Public License. Once the client is connect and gets an IP Address, we shall prompt the client device with a captive portal page before allowing access to internet for which we make use of

By default, NDS blocks everything, but intercepts port 80 requests. An initial port 80 request will be generated on a client device, either by the user manually browsing to an http web page, or automatically by the client device's built in **Captive Portal Detection** (CPD). As soon as this initial port 80 request is received, NDS will redirect the client to either its own splash page, or a splash page on a configured Forwarding Authentication Service (FAS). The user of the client device will then be expected to complete some actions on the splash page, such as accepting terms of service, entering a username and password etc. (this will of course be on either the basic NDS splash.html or the page presented by the FAS, depending on the NDS configuration).

2.4.2 Installing Nodogsplash

Now the on the ubuntu machine do the following steps to install NDS.

Setting up Nodogsplash on WAG

```
mkdir nodogsplash
cd nodogsplash/
git clone https://github.com/nodogsplash/nodogsplash.git

apt install build-essential debhelper devscripts
sudo apt install build-essential debhelper devscripts
apt install libmicrohttpd-dev dh-systemd
sudo apt install libmicrohttpd-dev dh-systemd
git checkout stable
git branch
sudo dpkg-buildpackage
sudo dpkg-buildpackage -b -rfakeroot -us -uc
cd ../../
sudo dpkg -i nodogsplash_3.3.2-1_amd64.deb
```

2.4.3 Configuring Nodogsplash

You can find the Nodogsplash config file in **/etc/nodogsplash/nodogsplash.conf**

```
GatewayInterface brtun

FirewallRuleSet authenticated-users {
  FirewallRule allow all
```

```
FirewallRuleSet preauthenticated-users {
   FirewallRule allow tcp port 53
   FirewallRule allow udp port 53
}

FirewallRuleSet users-to-router {
   FirewallRule allow udp port 53
   FirewallRule allow tcp port 53
   FirewallRule allow udp port 67
   FirewallRule allow tcp port 22
   FirewallRule allow tcp port 80
   FirewallRule allow tcp port 443
}

GatewayName Hotspot
GatewayAddress 10.0.2.1
GatewayPort 2050

MaxClients 250
SessionTimeout 0
PreAuthIdleTimeout 10
AuthIdleTimeout 120
CheckInterval 30
MACMechanism block
#BlockedMACList none
#AllowedMACList none
TrustedMACList none
TrafficControl no

DownloadLimit 0
UploadLimit 0
GatewayIPRange 0.0.0.0/0
DebugLevel 5
```

Now start the NDS by using following commands:

```
sudo systemctl enable nodogsplash
sudo systemctl start nodogsplash
```

Conclusion

So here we have analyzed the working of Guest Wi-Fi and Public Wi-Fi on COX 4131 and how client devices connecting to Public/Guest Wi-Fi get the IP assigned from Wireless Access Gateway (WAG) server. Also discussed how to setup Guest Wi-Fi on RDK-B RPi with SoftGREd running on an ubuntu machine acting as WAG endpoint.

References

Understanding SoftGre Setup With RPi

Captive Portal Using Nodogsplash Trevor Phillips

SoftGREd Wiki · GitHub

Nodogsplash (NDS) Docs

No labels