WeON Hotspot

Antonio Herrán Galaviz

5 de Febrero, 2016

Documentacion del proyecto

Version 1.0 - 5 de Febrero, 2016

Contenidos

Contenidos

1 Presentación

1.1 Problemática

Se presenta la necesidad de crear un punto de acceso con un **Portal Captivo** para la presentación de publicidad para acceder al internet móvil en el transporte urbano.

En el mismo se recabará la información de los sitios visitados por los usuarios que estén utilizando el portal.

1.2 Requerimientos

- Un Sistema Linux compatible en la One Board Computer Radxa Rock Lite
- Que el Sistema sea configurable para utilizarse como punto de acceso Wireless
- Que el Sistema sea configurable para utilizarse a traves de un punto de acceso externo
- Que el Sistema registre las direcciones MAC de los usuarios a conectarse
- Que el Sistema registre las paginas solicitadas por MAC de los usuarios.
- Que el Sistema lea las direcciones GPS de un modulo conectado externamente.

1.3 Esfuerzo realizado

Se trabajo sobre la busqueda de las tecnologias optimas para el desarrollo e implementacion de un HotSpot sobre una One Board Computer capaz de permitir un **Portal Captivo** que capture el trafico por **sesion** y por **usuario**.

Investigacion

El proceso de investigación y documentación, se realizó sobre las siguientes tecnologías.

- Sistema Operativo
- Software de Portal Captivo
- Solución de monitoreo
- Lenguaje de programacion
- Base de datos.

1.4 Presentación

Se ha construido un **Portal Captivo** utilizando GRASE Hotspot. Un paquete de integración que utiliza la teconolgia Coova Chilli como portal captivo, FreeRADIUS como base de datos para los usuarios y un portal web propio para la gestión y administración.

Los servicios se levantan de manera automática a traves de servicios y demonios creados para WeON, que se aseguran que los programas se activen en el orden y configuraciones necesarias

2 Tecnologias

Se utilizaron las siguientes tecnologias en el desarrollo de la implementacion del sistema.

2.1 Debian

Descripcion

Debian o Proyecto Debian (en inglés: Debian Project) es una comunidad conformada por desarrolladores y usuarios, que mantiene un sistema operativo GNU basado en software libre. El sistema se encuentra precompilado, empaquetado y en formato deb para múltiples arquitecturas de computador y para varios núcleos.

Especificaciones

Se utilizo el sistema Debian generado para la Radxa Rock Lite conocido como RaBIAN, seleccionada su version Nightly de fecha 24-01-2016

2.2 Python

Descripción

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.
Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Implementación

Python se utilizó para escribir el demonio y los módulos de funcionalidad.
Se selecciona la version 2.7.10 por compatibilidad y estabilidad

2.3 HostAPD

Descripcion

HostAPD es un software de control de tarjetas wireless para su configuración como punto de acceso. El software toma control de la tarjeta inalambrica y la configura para la recepcion de estaciones (usuarios).

Especificaciones

Se configura el software HostAPD como punto de acceso con el nombre *WeON Hotspot*, sin contraseña

2.4 Havegd

Descripcion

Havegd provee una nueva fuente de entropia para el algoritmo de conexiones inalambricas, ahorrando capacidad de procesador y reforzando la seguridad

Especificaciones

Havegd es un servicio de arranque automático consumido por HostAPD

2.5 GRASE Hotspot

Descripcion

GRASE Hotspot es un sistema integrado de solucion de Portal Captivo para Linux Debian. Esta integracion incorpora en un gestor los programas:

- dnsmasq
- freeradius
- Coova Chilli
- Portal web propio

Especificaciones

GRASE Hotspot genera las siguientes bases de datos

2.6 FreeRADIUS

- Nombre de la base: radius
- Contraseña: radius
- Tablas
    1. mtotacct
    2. mtotaccttmp
    3. nas
    4. radacct
    5. radcheck
    6. radgroupcheck
    7. radgroupreply
    8. radpostauth
    9. radreply
    10. radusercomment
    11. radusergrousp

2.7 Portal Configuration

- Nombre de la base: radmin
- Contraseña: portal
- Tablas
    1. adminlog
    2. auth
    3. batch

4. batches
5. groups
6. settings
7. templates
8. vouchers

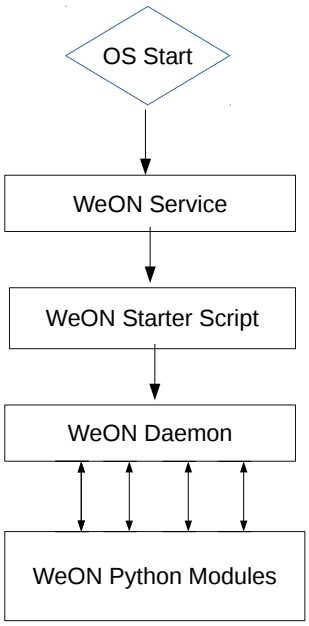
2.8 Base WeON

- Nombre de la base: weon
- Contraseña: WeonDB
- Tablas
    1. gps

3 Sistema

3.1 Estructura

El sistema esta compuesto por varios modulos, que son llamados por tres arrancadores principales. Estos arrancadores leen el archivo de configuración para algunos parametros.
El sistema esta estructurado de la siguiente manera.


OS Start
WeON Service
WeON Starter Script
WeON Daemon
WeON Python Modules

3.2 Flujo de llamadas

1. El sistema operativo llama al servicio *weon* al arranque.
2. El servicio *weon* llama al arrancador de modulos *weon-start.sh*.
3. El arrancador llama al demonio *weondaemon.py*.

3.3 Archivos del sistema

3.3.1 Archivos de registro

En la carpeta */var/log/weon/* se encuentran los archivos de registro del sistema

1. service.log En este archivo se encuentra el registro del arrancador.
2. daemon.log En este archivo se encuentra el registro del demonio.
3. hostapd.log En este archivo se encuentra el registro de arranque de hostapd.
4. records.log En este archivo se encuentra el registro de paginas visitadas por MAC.

3.3.2 Archivo de configuracion weon

Revisa si el demonio esta corriendo y si no, lo ejecuta

```
#   Nombre del archivo: weon.conf
#
#   Descripcion:        Archivo de configuraciones para el
    sistema hotspot
#

### Configuraciones para el hotspot WeON
#
#
# no_camion=X                   # Numero del camion
# essid=<nombre>                # Nombre del hotspot interno
# start_delay=int               # Tiempo de loop del daemon
```

```
# sleep_time=int                    # Tiempo de espera entre
    arranques de modulos



[Hotspot]

no_camion=1                         # Numero del camion
essid=WeON Hotspot                  # Nombre del hotspot interno
start_delay=20                      # Tiempo de loop del daemon
sleep_time=40                       # Tiempo de espera entre
    arranques de modulos


### Configuraciones de sistema para el punto de acceso.
#
# lan=[eth0 = AP externo,# wlan0 = WiFi interno]
    #Interfaz de red local
#
#

[Interfaces]

lan=wlan0
wwan=ppp0                           # La interfaz de donde tomara
    la conexion a Internet
```

Listado 1: Archivo de configuracion weon

3.3.3 Servicio weon

Servicio de arranque con el sistema

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: weon
# Required-Start: mysql
# Required-Stop:
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Start weON requirements at boot time
# Description: Enable weON required services
### END INIT INFO

. /lib/lsb/init-functions
```

```
do_start() {
            # Call weon start script.
        log_begin_msg "Starting weon upstart services"
        exec /etc/weon/weon-start.sh init-start
        log_end_msg $?
}

do_stop() {
            # Call weon stop script
        log_begin_msg "Stopping weon services"
        echo "[Stop]"
        exec /etc/weon/weon-start.sh init-stop
        log_end_msg $?
}

# Case dependant functions
case "$1" in
        start)
                do_start
                ;;
        stop)
                do_stop
                ;;
        *)
                echo "Usage: /etc/init.d/weon {start|stop}"
                exit 1
                ;;
esac

exit 0
```

Listado 2: Servicio weon

3.3.4 Arrancador weon

Script de inicio de las utilerias necesarias por el demonio

```
#!/bin/bash

# Script Name: weon-start.sh
# Author: Antonio Herran
```

```
#
# Description: WeON services caller script.
#               Calls WeON daemon,
#               hostapd. Logs calls
#


LOG_DIR=/var/log/weon                                   #
    Logging files location
SYSTEM_LOG_FILE=system.log                              # Log
    for systems started
HOSTAPD_LOG_FILE=hostapd.log                            # Log
    for HostAPD output
TCPDUMP_LOG_FILE=tcpdump.log                            # Log
    for tcpdump output
SYSTEM_LOG_PATH=$LOG_DIR/$SYSTEM_LOG_FILE               # Full
    filename for system
HOSTAPD_LOG_PATH=$LOG_DIR/$HOSTAPD_LOG_FILE             # Full
    filename for hostapd
TCPDUMP_LOG_PATH=$LOG_DIR/$TCPDUMP_LOG_FILE             # Full
    filename for tcpdump
HOTSPOT_NAME=`grep essid /etc/weon/weon.conf | sed -r
    's/essid=//'`

start_from_init(){                                  # Create
    directories and save old logs

if [ ! -d /var/log/weon ]; then
        mkdir /var/log/weon
fi

if [ ! -d /var/run/weondaemon ]; then
        mkdir /var/run/weondaemon
fi

for i in $(ls $LOG_DIR/*.log); do
        if [ -f $i ]; then
                mv $i $i.old
        fi
done

sed -i "s/^essid=.*$/essid=$HOTSPOT_NAME/"
    /etc/hotspot/hotspot.conf

echo "[WeON system log] Start of file" > $SYSTEM_LOG_PATH
```

```
echo "[Hostapd log] Start of file" > $HOSTAPD_LOG_PATH
echo "[tcpdump log] Start of file" > $TCPDUMP_LOG_PATH
}

stop_from_init() {                                          #Stop
    function -placeholder-
echo "[End of session]" >> $SYSTEM_LOG_PATH
}

log_sys(){                                                  #
    Routes console output to systemlog
exec >> $SYSTEM_LOG_PATH 2>&1
set -x
}

log_host(){                                                 #
    Routes console output to hostapd.log
exec >> $HOSTAPD_LOG_PATH 2>&1
set -x
}


log_tcpdump(){                                              #
    Routes console output to tcpdump.log
exec >> $TCPDUMP_LOG_PATH 2>&1
set -x
}


start_wlan() {                                              #
    Ensure interfaces files are in place
        log_sys
        rm /etc/network/interfaces
        cp /etc/weon/backupfiles/wlan0 /etc/network/interfaces
        ifdown wlan0
        ifup wlan0
}

start_eth() {                                               #
    Ensure interfaces files are in place
        log_sys
        rm /etc/network/interfaces
        cp /etc/weon/backupfiles/eth0 /etc/network/interfaces
        ifdown eth0
        ifup eth0
```

```
}

start_hostapd() {                                       #Start
    HostAPD in the background
        log_host
        pidof hostapd
        if [ $? -ne 0 ]
        then
                exec nohup hostapd -B -dd
                    /etc/hostapd/hostapd.conf &
        fi
}

stop_hostapd() {                                        #Stop
    HostAPD
        log_host
        HAPD = pidof hostapd
        kill `pidof hostapd`
        echo "[HostAPD Killed PID $HAPD]"
}

start_tcpdump() {                                       # Start
    tcpdump
        log_tcpdump
        echo "[tcpdump Start]"
        exec nohup tcpdump &
}

start_pppd() {
        log_sys
        exec nohup /usr/sbin/pppd nodetach lock nodefaultroute
            ipv6 , user webgprs ttyUSB1 noipdefault noauth
            usepeerdns lcp-echo-failure 5 lcp-echo-interval 30
            idle 0 ipparam /org/freedesktop/NetworkManager/PPP/0
            plugin /usr/lib/pppd/2.4.6/nm-pppd-plugin.so &
}


start_weondaemon() {                                    #Start
    daemon
        log_sys
        echo "[Starting WeON Python Daemon]"
        exec python /etc/weon/weondaemon.py start
}
```

```
stop_weondaemon() {                                    #Stop
    daemon
        log_sys
        echo "[Stopping WeON Python Daemon]"
        exec python /etc/weon/weondaemon.py stop
}

### Cases of script

case "$1" in
        init-start)
                start_from_init
                start_weondaemon
                ;;
        start-daemon)
                start_weondaemon
                ;;
        stop-daemon)
                stop_weondaemon
                ;;
        start-wlan)
                start_wlan
                start_hostapd
                ;;
        start-eth)
                start_eth
                ;;
        start-pppd)
                start_pppd
                ;;
        start-hostapd)
                start_hostapd
                ;;
        stop-hostapd)
                stop_hostapd
                ;;
        start-tcpdump)
                start_tcpdump
                ;;
        *)
                echo "Usage: $0
                    {init-start|start-daemon|stop-daemon|start-hostapd|stop-hostapd}"
                exit 1
                ;;
```

```
esac
```

Listado 3: Arrancador weon

3.3.5 weon Daemon Check

Revisa si el demonio esta corriendo y si no, lo ejecuta

```
#!/bin/bash
#
# Script Name: weon-daemoncheck.sh
# Author: Antonio Herran
#
# Description: Called from cronjob. Checks if WeON daemon is
   up. Starts it if not
#

if [ ! `pidof python` ]; then
        python /etc/weon/weondaemon.py start
fi
```

Listado 4: weon Daemon Check

3.3.6 weon nm-applet

Script que arranca el applet para NetworkManager

```
#!/bin/bash
while [ ! `pidof nm-applet` ]; do
#               systemctl restart NetworkManager
#       sleep 5
        sudo -u rock DISPLAY=:0 nm-applet &
        sleep 15
        sudo ntpdate -b -u pool.ntp.org
done
```

Listado 5: weon nm-applet

3.3.7 Demonio weon

Corre en segundo plano asegurando la ejecucion de lo necesario para el sistema

```
# Script Name: weondaemon.py
# Author: Antonio Herran
#
# Description: WeON system daemon. Checks for sanity of the
    system and keeps system up
#
#

import logging                          #Standard Python lib
import time
import datetime
import ConfigParser
import subprocess
import mysql.connector
from daemon import runner

import weon_sanity                      #WeON libs
import weon_databaser
import weon_conf
import weon_gps
import weon_connections

class App():

    def __init__(self):
        self.stdin_path = '/dev/null'
                          # Redirect console output to log
            files
        self.stdout_path = '/var/log/weon/system.log'
                          #
        self.stderr_path = '/var/log/weon/system.log'
                          #
        self.pidfile_path =
            '/var/run/weondaemon/weondaemon.pid'
        self.pidfile_timeout = 5
        self.config = ConfigParser.ConfigParser()
        self.config.read('/etc/weon/weon.conf')
        self.checker = {}

                  # WeON Variables - List for sanity checks
        self.connection = 0

                  # WeON Variable -
        self.sleep_time =
            int(weon_conf.parse_config('/etc/weon/weon.conf')['sleep_time'])
```

```
                                        # WeON Variable -
        self.start_delay =
            int(weon_conf.parse_config('/etc/weon/weon.conf')['start_delay'])
                                        #

    def run(self):
                            # Main module
        logger.debug(self.config.sections())
        logger.debug('[WeON Daemon start]')
        self.startup()
        time.sleep(20)
        while True:
                                # Main code loop
            self.sanity_check()
                                    # Sanity Checker
#           weon_sanity.do_sanity(self.checker)
            self.insert_db()
            time.sleep(self.sleep_time)

    def startup(self):
        """Do at load functions """
        self.nmapplet()
        time.sleep(10)
        logger.debug('[nm-applet started]')
        if self.config.get('Interfaces', 'lan') == 'eth0':
                self.start_eth()
                logger.debug('[Interface up eth0]')
        if self.config.get('Interfaces', 'lan') == 'wlan0':
                self.start_wlan()
                logger.debug('[Interface up wlan0]')
        time.sleep(self.start_delay)
        self.tcpdump()
                                # Workaround for iptables (without
            this, Grase doesn't open ports)
        logger.debug('[tcpdump started]')

    def tcpdump(self):
        subprocess.Popen('/etc/weon/weon-start.sh
            start-tcpdump', shell=True)


    def start_wlan(self):
        subprocess.Popen('/etc/weon/weon-start.sh start-wlan',
            shell=True)
```

```
            subprocess.Popen('/usr/share/grase/scripts/update_grase_networksettings.sh',
                shell=True)

        def start_eth(self):
            subprocess.Popen('/etc/weon/weon-start␣start-eth',
                shell=True)
            subprocess.Popen('/usr/share/grase/scripts/update_grase_networksettings.sh',
                shell=True)

        def nmapplet(self):
            subprocess.Popen('/etc/weon/weon-nmapplet.sh',
                shell=True)

        def sanity_check(self):
            logger.info("Sanity␣check")
            weon_sanity.do_all_checks(self.checker)
            logger.info(self.checker)

        def insert_db(self):
            self.weondb =  mysql.connector.connect( user='weon',
                password='WeonDB', host='127.0.0.1',
                database='weon')       # Open Connections
            self.cursor = self.weondb.cursor()
            camion =
                int(weon_conf.parse_config('/etc/weon/weon.conf')['no_camion'])
            self.latitude, self.longitude = weon_gps.readgps()
            connections = weon_connections.conns()
            self.cursor.execute("INSERT␣INTO␣
                gps(camion,latitud,longitud,conectados)␣VALUES(%s,␣
                %s,␣%s,␣%s)", (camion, self.latitude,
                self.longitude, connections, ))
            self.weondb.commit()



app = App()
logger = logging.getLogger("WeON␣Daemon")
logger.setLevel(logging.DEBUG)
formatter = logging.Formatter("%(asctime)s␣-␣%(name)s␣-␣
    %(levelname)s␣-␣%(message)s")
handler = logging.FileHandler("/var/log/weon/daemon.log")
handler.setFormatter(formatter)
logger.addHandler(handler)

daemon_runner = runner.DaemonRunner(app)
```

```
#This ensures that the logger file handle does not get closed
    during daemonization
daemon_runner.daemon_context.files_preserve=[handler.stream]
daemon_runner.do_action()
```

Listado 6: Demonio weon

3.3.8 Módulo de lectura de configuración

Lee el archivo de configuración

```
import sys

def parse_config(filename):
        COMMENT_CHAR = '#'
        OPTION_CHAR = '='
        options = {}
        f = open(filename)
        # Remove comments
        for line in f:
                if COMMENT_CHAR in line:
                        line, comment =
                            line.split(COMMENT_CHAR, 1)
        # Find values
                if OPTION_CHAR in line:
                        option, value = line.split(OPTION_CHAR,
                            1)
                        # strip spaces:
                        option = option.strip()
                        value = value.strip()
                        # store in dictionary:
                        options[option] = value

        f.close()
        return options

if __name__ == "__main__":
        filename = sys.argv[1]
        options = parse_config(filename)
        print options
```

Listado 7: Lector de configuración

3.3.9 Módulo de lectura de gps

Lee el gps del tty serial 0. Pines 13 -UART0_RX- y 14 -UART0_TX-. Voltaje en Pines 1 -GND- y 2 -VCC 5V-

```
# Script Name: weon_gps.py
# Author: Antonio Herran
#
# Description: Reads gps and returns latitude and longitude
    using NMEA standard
#               Sanitizes to zero on garbage.
#

import serial
import socket


def readgps():
        ser = serial.Serial('/dev/ttyS0', 9600, timeout=1)
        """Read the GPG LINE using the NMEA standard"""
        latitude=''
        longitude=''
        while True:
                line = ser.readline()
                if "GPGGA" in line:
                        latitude = line[18:26]
                        longitude = line[31:39]
                        if longitude[4] == 'M':
                           # Sanitize garbage
                                latitude = '0000.0000'
                                longitude = '0000.0000'
                        return(latitude, longitude)

if __name__ == "__main__":
        output = readgps()
        print output
```

Listado 8: Lector de gps

3.3.10 Módulo de sanidad

Revisa sanidad del sistema

```
# Script Name: weon_sanity.py
# Author: Antonio Herran
```

```
#
# Description: WeON system sanity module. Checks for sanity of
    the components and takes action if needed
#
#

import os
import time
import subprocess
import syslog


#def check_connection():
                                                    # Check
    status of connection
#       cmd = subprocess.Popen('nmcli -t -f TYPE,STATE dev',
    shell=True, stdout=subprocess.PIPE)                 # Not
    enabled in first version
#       output, err = cmd.communicate()
#       for line in output.splitlines():
#               if "gsm" in output:
#                       if "disconnected" in line:
#                               return False
#       else:
#               return True


def check_interface():
        cmd = subprocess.Popen('ifconfig -a', shell=True,
            stdout=subprocess.PIPE)
        output, err = cmd.communicate()
        for line in output.splitlines():
                if "ppp0" in line:
                        return True
        else:
                return False

def check_nmapplet():
        pid = subprocess.Popen('pidof nm-applet',
            stdout=subprocess.PIPE, stderr=subprocess.PIPE,
            shell=True)
        output, errors = pid.communicate()
        if not output:
                return False
        else:
```

```
                return True

def enable_connection():
        print "bogus"
        #subprocess.Popen('/etc/weon/weon-connection.sh',
            shell=True, stdout=subprocess.PIPE)


def do_all_checks(checker):
        #checker['connection']=check_connection()
                                                        #
            Not implemented in first version
        checker['nmapplet']=check_nmapplet()
        checker['interface']=check_interface()
        return checker

def do_sanity(checker):
#       if checker['bam'] == False:
#               subprocess.Popen('reboot', shell=True)
        if checker['nmapplet'] == False:
                subprocess.Popen('/etc/weon/weon-nmapplet.sh',
                    shell=True)
        time.sleep(5)
        checker['interface'] = check_interface()
        if checker['interface'] == False:
                subprocess.Popen('reboot', shell=True)


if __name__ == "__main__":
        checker = {}
        do_all_checks(checker)
        print checker
```

Listado 9: Modulo de sanidad

3.3.11 Módulo de bases de datos

Lee y escribe en las bases de datos

```
# Script Name: weon_databaser.py
# Author: Antonio Herran
#
# Description: WeON module for collecting data and connecting
    to databases.
#               Opens weon and radmin databases
```

```
#
#

import weon_conf
import weon_gps
import weon_connections
import mysql.connector
import os
import sys
import datetime


class Weondb:                                   # Class for
    connecting to weonDB

        def __init__(self):
                self.camion =
                    int(weon_conf.parse_config('/etc/weon/weon.conf')['no_camion'])
                self.latitude, self.longitude =
                    weon_gps.readgps()
                self.connections = weon_connections.conns()
                self.dbconn = self.connect_db()


        def close(self):                        # Close
            connection
                self.dbconn.close()

        def connect_db(self):
                db = mysql.connector.connect( user='weon',
                    password='WeonDB', host='127.0.0.1',
                    database='weon')        # Open Connections
                return db

        def return_data(self):
                self.latitude, self.longitude =
                    weon_gps.readgps()
                self.connections = weon_connections.conns()
                list = [self.camion, self.latitude,
                    self.longitude, self.connections]
                return list

        def insert_record(self):
                self.close()
                self.dbconn = self.connect_db()
```

```
                cur = self.dbconn.cursor()
                cur.execute("INSERT INTO
                    gps(camion,latitud,longitud,conectados)
                    VALUES(%s, %s, %s, %s)", (self.camion,
                    self.latitude, self.longitude,
                    self.connections, ))
                self.dbconn.commit()



class Radmindb:                         # Class for connecting
    to radminDB

        def __init__(self):
                self.eth =
                    'a:7:{s:12:"lanipaddress";s:11:"192.168.2.1";s:11:"networkmask";s

                self.wlan =
                    'a:7:{s:12:"lanipaddress";s:11:"192.168.2.1";s:11:"networkmask";s


                self.dbconn = self.connect_db()

        def close(self):
                self.dbconn.close()

        def connect_db(self):
                db = mysql.connector.connect( user='radmin',
                    password='portal', host='127.0.0.1',
                    database='radmin')
                return db


if __name__ == "__main__":
        wdb = weondb()
        print wdb.latitude, wdb.longitude
        print wdb.camion
        print wdb.connections
        wdb.close()
```

Listado 10: Manejo de bases de datos

3.3.12 Módulo de conteo de conexiones

Lee de la base de datos las conexiones

```
# Script file: weon_connections.py
# Author: Antonio Herran
#
# Description: WeON module for counting connections open.
#
#
#

import weon_databaser
import mysql.connector
import os
import sys

def connect_db():
        db = mysql.connector.connect( user='radius',
            password='radius', host='127.0.0.1',
            database='radius')        # Open Connections
        return db

def conns():
        w = connect_db()
        cur = w.cursor()
        cur.execute("SELECT CallingStationId FROM radacct WHERE
            AcctStopTime IS NULL")
        data = cur.fetchall()
        conns = len(data)
        if data[0] == (u'00-00-00-00-00-00',):
                conns = 0
        w.close()
        return conns
```

Listado 11: Conteo de conexiones label

3.4 Configuración

Para configurar el sistema es necesario utilizar dos archivos que se encuentran en acceso rapido sobre el escritorio del sistema

En weon.conf, se puede configurar el nombre del ESSID asi como el numero del camión.

Ethernet

Para utilizarlo en modo ethernet (access point externo) es necesario configurar en weon.conf *lan=eth0* y en el portal de configuración GRASE *lan=eth0*

Wireless

Para utilizarlo en modo wireless(access point interno) es necesario configurar en weon.conf *lan=wlan0* y en el portal de configuración GRASE *lan=wlan0*

*IMPORTANTE*En ambos casos, es necesario asegurarse que en el portal GRASE, la configuracion este seleccionada como *wan=ppp0*

4 Configuraciones clave

Si el sistema quisiera reimplementarse sobre otro OS Debian, estas son las configuraciones clave.

4.1 Paquetes a instalar

```
mate-desktop-environment-extra
gir1.2-notify-0.7
inetutils-ping
grase-conf-openvpn
python3-dbus
wireless-tools
nload
```

```
vibrancy-colors
fonts-wqy-microhei
linux-image-3.0.36-rock-lite
libpepflashplayer
usbutils
tcpdump
fonts-wqy-zenhei
gir1.2-gconf-2.0
ssh
haveged
grase-repo
live-config
radiance-flat-colors
seahorse
lightdm
python-serial
libwiringx
mc
mate-core
python-mysql.connector
grase-www-portal
gir1.2-appindicator3-0.1
gnome-mplayer
ambiance-flat-colors
python-daemon
live-boot
rock-lite-overlay
vim
```

Listado 12: Paquetes de dependencias

4.2 Orden importante

Si se genera de nuevo un paquete para la Radxa Rock Lite, es necesario utilizar el instalador de HostAPD encontrado en la carpeta backupfiles. De lo contrario sera necesario obtener la información de la pagina de soporte de la Radxa Rock

IMPORTANTE Es necesario instalar hostapd si se fuera a utilizar, *antes* que GRASE Hotspot, y es necesario sobreescribir el archivo /etc/dnsmasq.conf con el encontrado en la carpeta weon/backupfiles

4.3 GRASE Hotspot

4.3.1 BD radmin

Esta base de datos contiene las configuraciones del portal. La principal es la tabla *settings*. La entrada *networksettings* contiene los registros de IP de la LAN y configuracion de interfaces de la WAN y LAN. Esta entrada esta serializada a través de la pagina de administración de GRASE

4.3.2 BD radius

En esta base de datos se encuentran los registros de usuarios, de aqui se puede extraer la información de todas las conexiones.
Las tablas importantes son *radacct* que contiene las entradas de los usuarios con su MAC address, tiempo de entrada, tiempo de salida y razon de salida y *radreply* que contiene las configuraciones de CoovaChilli.

Referencias

[1] Radxa Homepage, *Pagina Oficial de la Radxa Rock*, http://radxa.com/

[2] GRASE Homepage, *Pagina Oficial de GRASE Hotspot*, https://grasehotspot.org/