LAB 13: WiFi security

PA197

Sven Relovský, Patrik Rehuš, Michal Šnajdr
snajdr@ics.muni.cz

Warning

All of the advice/information that I'll give is **purely for educational purposes**. MU will not be responsible for any illegal use of this tutorial. Don't hack any wireless network, unless you are the owner of that network.

We will try...

Wifi security - vulnerabilities

- Hidden SSID
- MAC filtering
- Weaknesses in WEP encryption
- Security vs. Comfort (Is WPS secure?)
- WPA/WPA2 capture handshake
- Homework: WPA2 attacks

KISMET tool

- Detection of attacker's/fake access points

Lab prerequisities

1. **Hardware**
   - laptop with a WiFi module (must support monitor mode)
2. **Software**
   - specialised Linux distro Kali Linux - All-In-One solution
   - airmon-ng – a bash script designed to turn wireless cards into monitor mode
   - airodump-ng – a packet capture tool for aircrack-ng
   - aireplay-ng – inject ARP-request packets into a wireless network to generate traffic
   - aircrack-ng – a 802.11 WEP / WPA-PSK key cracker
   - wash – utility for identifying WPS enabled points
   - reaver with Pixie – modified version - exploits a security hole in wireless routers
   - kismet – network detector, packet sniffer, and intrusion detection system for 802.11 wireless LANs

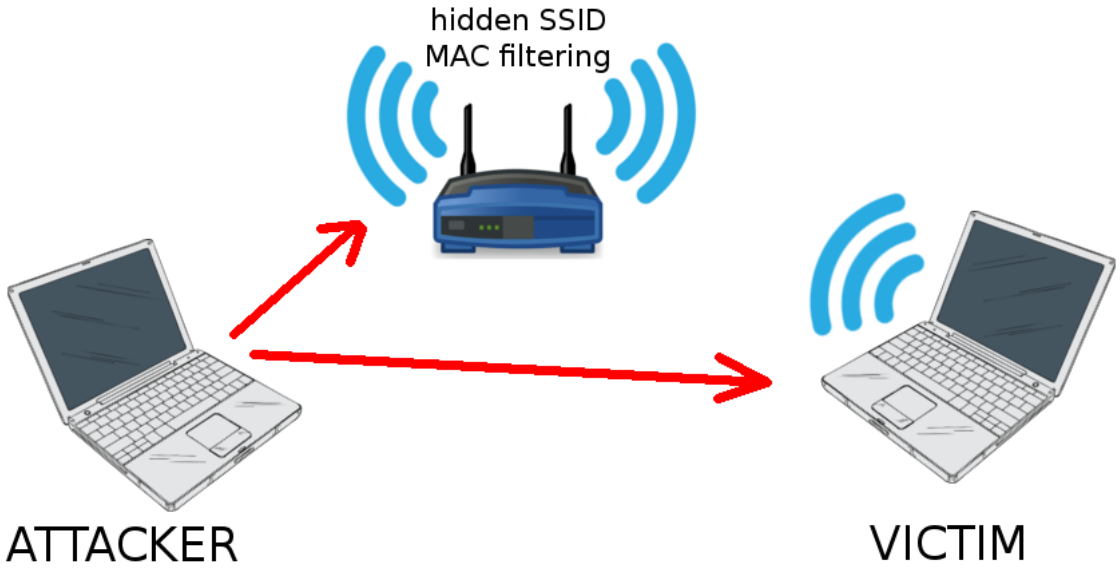
Why is WiFi security so important?

- WiFi connection is very popular (flexible, comfortable, cheap)
- number of devices is rapidly increasing due to Internet of Things (IoT)
- we transfer sensitive data

Vulnerabilities

- connect to devices in network (capture webcam, access to shared network storage, control intelligent things - heating, light . . .)
- capture and analyze sensitive data $\rightarrow$ identity theft
- set up fake AP with stronger signal $\rightarrow$ MITM attack

Initial scheme



Topology:

- SSID: (hidden) - in all labs we will only use the PA197* SSIDs!
- Encryption: **None**
- MAC filtering: **enabled**

LAB 1: Get started

1. Start wireless card in monitor mode
   - 1.1 **airmon-ng** - can see available device
   - 1.2 **airmon-ng check kill** - to kill blocking application (in some cases it is not needed)
   - 1.3 **airmon-ng start wlan0** - start monitor mode on specific device (in my case wlan0) $\rightarrow$ create a new virtual interface
   - 1.4 in next steps we will only use this one (e.g. wlan0mon, mon0...)
2. Start scanning networks **airodump-ng wlan0mon**
   - lists all of the wireless networks in your area
   - useful information about them like channel, SSID, encryption...
   - display hidden SSID of network (only if there is an active host) - SSID is included in the header of each packet

LAB 2: MAC filtering

MAC addresses are broadcast unencrypted in header of frame, so it can be captured with a tool such as Wireshark. For this purpose we will use **airodump-ng** with additional options.

airodump-ng --bssid AA:AA:AA:AA:AA:AA **-c** B **wlan0mon**

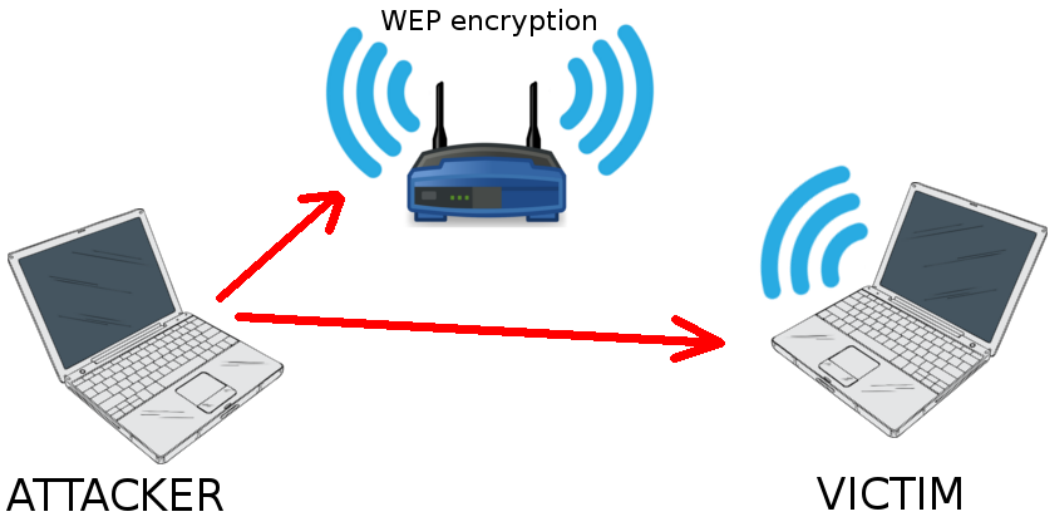
- **--bssid AA:AA:AA:AA:AA:AA** (equivalent for **-d**) is the BSSID of the AP
- **-c B** (equivalent for **--channel**) is the channel the AP is operating on
- **wlan0mon** is the monitoring wireless adapter

Using this method we can capture (or use a script to save to file) allowed MAC addresses. If we want to connect to a 'secured' AP we can change the MAC address in OS (Linux tool is **macchanger**; in Windows it is in the settings of the adapter)

WEP - Wired Equivalent Privacy

- the oldest encryption of WiFi (1997)
- at the time was suppose to provide confidentiality comparable to a traditional wired network.
- uses a shared key for encrypting packets
- most common is 64-bit or 128-bit WEP
  - 64-bit = 40-bit key + 24-bit Initialization Vector(**IV**) $\rightarrow$ **RC4 keystream**
  - 128-bit = 104-bit key + 24-bit Initialization Vector(**IV**) $\rightarrow$ **RC4 keystream**
- WEP encryption is based on RC4 keystream XORed with plain-text
- exist (very rarely) 152-bit and a 256-bit WEP encryption (vendor specific implementations)

Topology



- administrator realized that MAC filtering is very bad protection and he needs some encryption → enable WEP
- SSID: **PA197-WEP[2]**
- MAC filtering: **none**
- encryption: **WEP**

WEP attack

- known as PTW attack
  - created by Andrei **P**ychkine, Erik **T**ews, and Ralf-Philipp **W**einmann (in 2007)
- cryptanalytic method to discover a 104-bit WEP key
- needs only 40 000 captured packets to reach 50% probability of success (60 000 is 80% chance and 85 000 packet is about 95% chance to discover the WEP key)
- use deauth and ARP re-injection $\rightarrow$ 40 000 packets can be captured in less than minute under ideal conditions
- Let's it try ☺

LAB 3: How simple is it to hack WEP

Start capturing data

airmon-ng start wlan0 (only if monitor mode is not enabled)

airodump-ng wlan0mon

- start scanning WiFi networks
- needs to find target of attack (BSSID, channel)

airodump-ng --bssid {*BSSID*} **-c** B **--write** {*PATH*} **wlan0mon**

- **--bssid** the BSSID of the AP
- **-c** the channel of AP
- **--write** (equivalent for **-w**) name of the file you want to save the captured data to (e.x. /root/Desktop/WEPattack)

LAB 3: How simple is it to hack WEP

Fake Authentication

- we need to capture a lot of IV's to recover the password
  - at least 20 000, ideally 100 000 IV's
- AP ignores the packet from not associated MAC (send out "DeAuthentication" packet)
- open new terminal (first one captures data)

```
aireplay-ng -1 0 -e HackMe -a 08:86:30:74:22:76
-h BB:BB:BB:BB:BB:BB wlan0mon
```

- attack mode: **-1** 0, (**--fakeauth=**{**delay**} can be used too) Fake authentication with AP. (0 is delay)
- **-e** HackMe Set target SSID for Fake Authentication attack
- **-a** 08:86:30:74:22:76 Set Access Point MAC address
- **-h** BB:BB:BB:BB:BB:BB MAC address client we are forging
- Next options
  - **-o** {n-packets} Set the number (in pour case only 1) of packets for every authentication and association attempt. Default is multiple and this confuses some APs
  - **-q** 10 - Send keep alive packets every 10 seconds

LAB 3: How simple is it to hack WEP

Generate traffic to get new IVs

- now as we are authenticated so AP will accept ARP request packets
- each response include unique IV
- Aireplay-ng listen to the AP for ARP request, and inject them as soon as they find one
- it causes that data and the beacons should began to grow quickly

aireplay-ng -3 -b AA:AA:AA:AA:AA:AA
-h BB:BB:BB:BB:BB:BB
-**e** HackMe **wlan0mon**

- same tool but different attack mode (In previos command we use -1 fake authentication; now we use -3 arpreplay mode)
- **-3** (or –**arpreplay**) Standard ARP-request replay
- **-b** AA:AA:AA:AA:AA:AA MAC address of access point
- **-h** BB:BB:BB:BB:BB:BB MAC address client we are forging
- **-e** HackMe Set target SSID
- **wlan0mon** is the monitoring wireless adapter
- we can specify source MAC adress (option **-h** {smac}), but it is not necessary. Implicitly is chosen MAC address of wifi adapter in

LAB 3: How simply is to hack WEP

Cracking the password

- in the first terminal we can see a lot of sent packets
- we can try to crack the password from captured data
- open new (third) terminal

```
aircrack-ng -z /root/Desktop/WEPattack-01.cap
```

- **-z** Uses PTW (Andrei Pyshkin, Erik Tews and Ralf-Philipp Weinmann) attack. (In case we don't specify type, airodump-ng analyze data and choose type based on captured data)
- /root/Desktop/WEPattack-01.cap path to file with captured IV's (*.cap/*.ivs file)
- if there is not enough captured IVs to crack key, aircrack-ng will wait for next capturing and try it crack again
- we must wait about 1-5 minutes to successful recovering WEP password (based on captured data)

WPS - Is it secure?

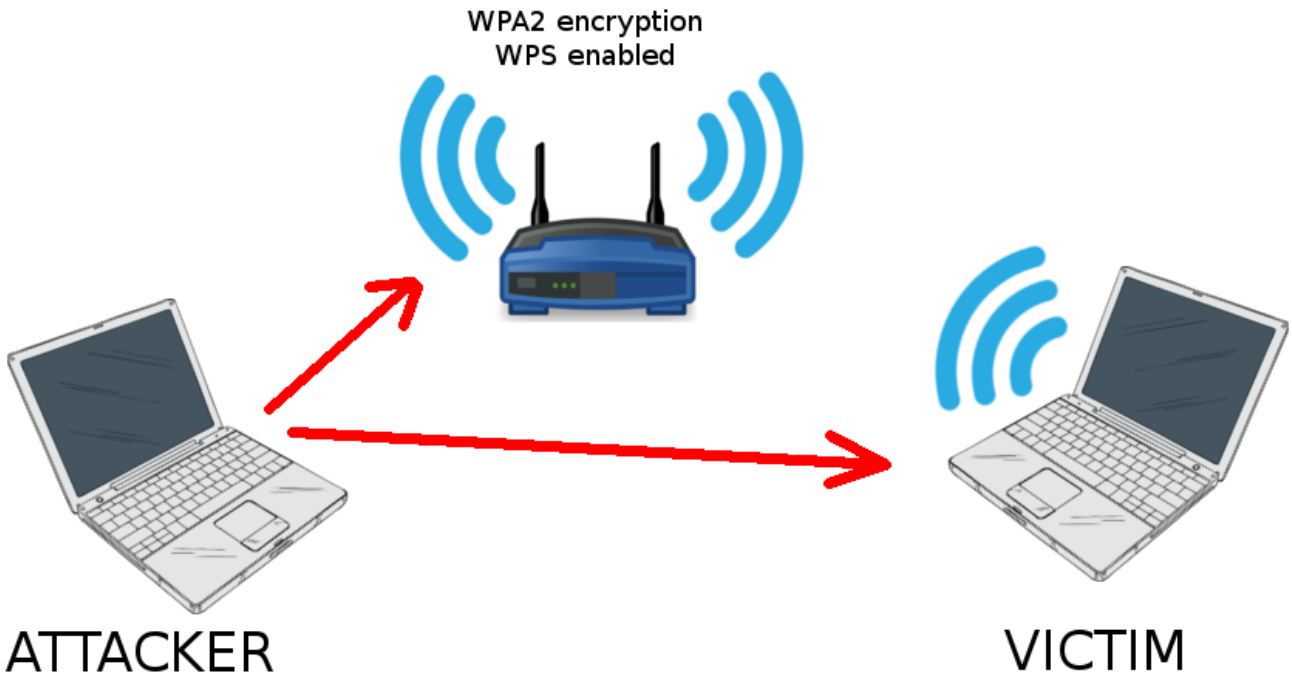
Wifi Protected Setup designed to quickly & easily authenticate a client to an AP mainly aimed for home users

1. AP & the Client exchange a series of EAP messages
2. At the end of this transaction, the Client will have the encryption key & the AP's signature and is ready to connect
3. Client re-associates with the new credentials & signatures
4. The actual passphrase is not exchanged during WPS initiation (an eight digit **PIN** is used for authentication)
5. The client is first authenticated using the PIN and then the actual passphrase is exchanged

WPS PIN

- WPS PIN consists of 8 digits
- last digit is a checksum $\rightarrow$ 7 digits to guess
- the PIN is validated by dividing it into 2 halves
- first half leaves $10^4 = 10\ 000$ guesses
- second half leaves $10^3 = 1000$ guesses
- all together we need only 11 000 guesses instead of 10^8=100 000 000
- can use **brute-force**
  - *on-line attack* **reaver tool**
  - *off-line attack* - PixieDust attack **reaver with PixieWPS** (This attack is only applicable to vulnerable devices)

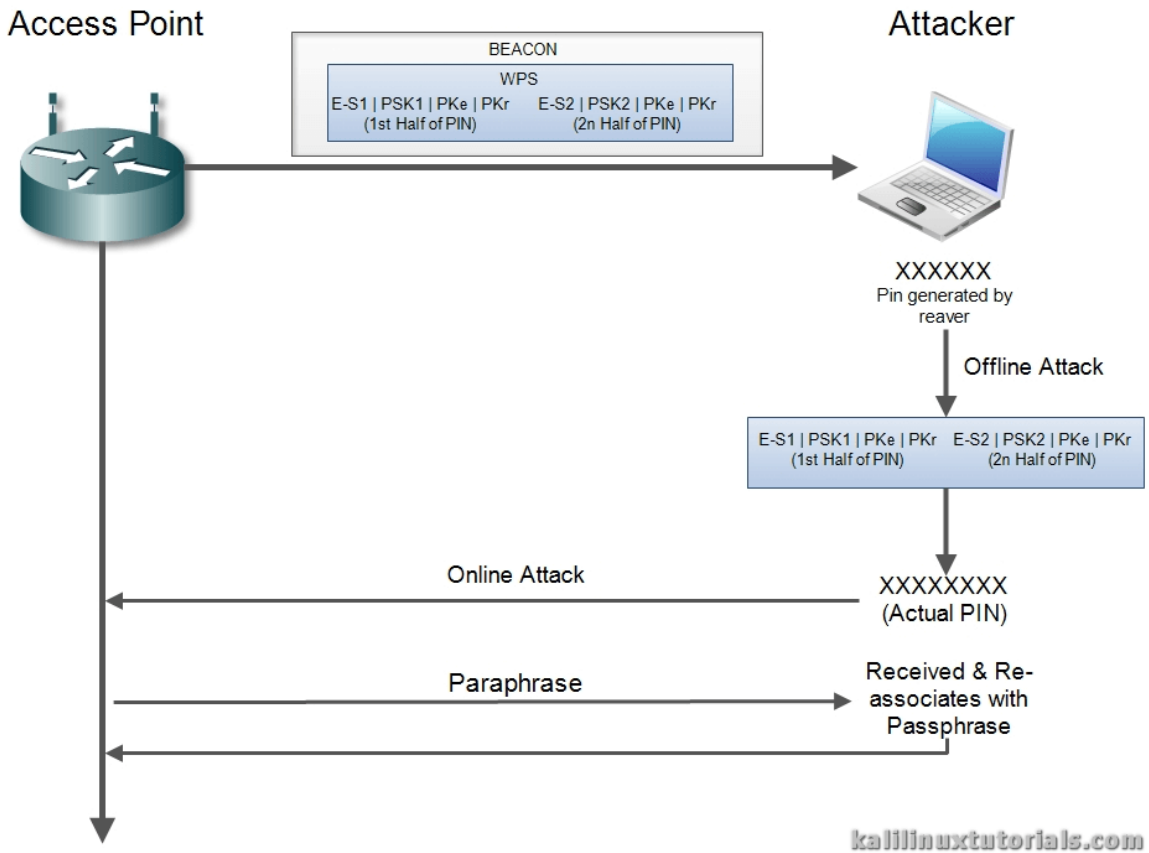
Topology



- administrator improve WiFi security → change WEP to WPA2
- SSID: **PA197** (visible)
- encryption: **WPA2**
- WPS: **enabled**

WPS offline attack

- lack of randomization in the components of the 2 halves of the PIN
- capture hashes and can start offline attack



LAB 4: WPS online attack

- Simply try PIN and wait for response
- Speed of founding passphrase is individual
- Some firmware may be protected against brute-force by **Lock down** time
- This protection radically decrease chance of successful attack

reaver -i mon0 -b AA:AA:AA:AA:AA:AA -vv

Attempts before lock	Lock down time	Attempts per minute	Maximum attack time	Maximum attack time	Comment
11000	0 minutes	46.15	3.97 hours	0.17 days	no lock down
?[7]		4.20	43,65 hours	1,82 days	Netgear WGR614v10
3	1 minutes	2.82	65.08 hours	2.71 days	Requirement for WSC 2.0
15	60 minutes	0.25	737.31 hours	30.72 days	Lock down configurations making brute force less practical
10	60 minutes	0.17	1103.97	46.00 days	
5	60 minutes	0.08	2203.97	91.83 days	

Assumed time per attempt: 1.3 seconds

WPA/WPA2

- currently the safest way to secure a WiFi connection
- all modern devices support WPA2 with AES (Advanced Encryption Standard)
- TKIP (Temporal Key Integrity Protocol) vs. AES (in mixed mode TKIP is there as a fallback method)
  - TKIP is no longer considered secure
- hackable with dictionary attack or brute-force (expect short password)
- WPA/WPA2 with strong pass-phrase is not possible to hack in short time (with current technology) -
  - brute-force lower+upper case characters, password length 8, AMD hd7970 + oclHashcat $\approx$ 12 years 60 days
    `http://calc.opensecurityresearch.com/`
- in next lab we will try to capture data (handshake) required for an offline dictionary and brute-force attack

LAB 5: WPA/WPA2 attack

airodump-ng --bssid {*BSSID*} **-c** B **--write** {*PATH*} **wlan0mon**

- **--bssid** (equivalent for **-d**)is the BSSID of the AP
- **-c** 6 (equivalent for **--channel**) is the channel the AP is operating on
- **--write** /root/Desktop/WPAcrack (equivalent for **-w**)name of file you want to save captured data

We need to capture 4-way handshake from the connected client (in new terminal:)

aireplay-ng --deauth 2 **-a** {AP bssid} **-c** {client bssid} **wlan0mon**

- **--deauth** (equivalent for **--0**) is a short cut for the deauth mode and the *2* is the number of deauth packets to send.
- **-a** indicates the access point/router's BSSID, replace {AP bssid} with the BSSID of the target network
- **-c** indicates the client's BSSID, the device we're trying to deauth, noted in the previous picture. Replace the {client bssid} with the BSSID of the connected client, this will be listed under "STATION."

LAB 5: WPA/WPA2 attack

- after succesful deauth we can see in top left corner "WPA handshake: [BSSID]". That is what we want. The 4-way handshake is captured so we can try to crack it now.
- the handshake is saved in *.cap file from airodump-ng
- based on this handshake we can try to discover the pass-phrase either by brute-force or dictionary attacks (Your Homework)
  - brute-force = systematically checking **all** possible keys or passwords until the correct one is found (very fast for short passwords)
  - dictionary attack = based on trying the strings **only** in a pre-arranged list of possible pass-phrases (tries only those possibilities which are deemed most likely to succeed)
- if we try brute-force we need a lot of luck and time

WPA2 Brute-force attack

- need *.cap file with handshake (using airodump-ng, wifite . . .)
- **wpaclean** cleans captured files to get only the 4-way handshake and a beacon
- *take care of the right order of output and input file option!*

```
wpaclean <out.cap> <in.cap>
```

- converting the .cap file to a format cudaHashcat, oclHashcat or Hashcat

```
/usr/share/hashcat-utils/cap2hccapx.bin input.pcap output.hccapx
```

Brute-Force Attack with Hashcat

- cudaHashcat or oclHashcat or Hashcat

```
hashcat -m 2500 -a 3 capture.hccap <MASK>
```

- **-m 2500** means we are attacking a WPA/WPA2 handshake file
- **-a 3** Brute Force Attack mode
- **capture.hccap** source file (generated using wpaclean and cap2hccapx)
- <**MASK**> type of characters to try (all combinations)

MASK

- Built-in charsets
  - ?l = abcdefghijklmnopqrstuvwxyz
  - ?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - ?d = 0123456789
  - ?s = special characters
  - ?a = ?l?u?d?s
- own charset using .hcmask file (for specific charset, variable length...)
- ex. ?u?u?u?u?u?u?u?u try all 8 Letter passwords in CAPS (exactly 8-character passwords)

Brute-Force Attack

- How long it will take?
- Example:
  - minimum password length (8 characters)
  - only lowercase letters
  - 26^8 combinations $= 208\,827\,064\,576$
  - Thinkpad T400 with Intel P8600 @ 2.40 Ghz 1300 hashes/s => 5 years

KISMET

- detector, sniffer, and intrusion detection system
- WiFi **passive** sniffer, always operates in monitor mode
- Linux, OSX, Windows, *BSD, ...
  - can run on extremely lightweight devices (APs, handhelds, etc)
- log GPS coordinates for mapping of network positions
- capture all management frames and also raw data frames
- can also detect active sniffers like Netstumbler
- able to decloak hidden SSIDs by watching client connections (like airodump-ng)
- can operate distributed
  - Kismet drones are very light version of the Kismet core (more lightweight than the server)
  - an entire building can report to one Kismet engine for logging and IDS (Intrusion detection system)

AP Spoofing

Situation:

- you are network administrator in a big corporation
- you want to protect employees in the company against connecting to a fake AP
- decided to use KISMET tool

Configure KISMET:

- configuration file **/etc/kismet/kismet.conf**
- section **APSPOOF alert**
- add line with SSID of AP and allowed MAC addresses

```
apspoof=PA197:ssid="PA197",validmacs="02:15:6D:85:83:C5,aa:bb:cc:dd:ee:ff"
```

- notice there is no space!
- if there are multiple MACs, list of MAC addresses must be comma-separated and enclosed in quotes
- in case there will be other MAC adress broadcasting same SSID, KISMET generate **alert**

How to protect your Wifi?

1. Never use WEP encryption!
2. Use WPA2 whenever it is possible
3. Disable WPS: Analyze whether your WiFi AP needs to be using WPS at all.
4. Increase timeout period: For advanced routers, we can increase the Receive timeout to slow down the attack
5. WPS Lock: Set the WPS Lock time to a large value
6. MAC Filtering: Of course really old-school, but sometimes can protect against script kiddies.
7. Physically Secure the Router

References

- Reaver + PixieWPS, http://kalilinuxtutorials.com/reaver-pixewps/
- Brute forcing Wi-Fi Protected Setup, https://sviehb.files.wordpress.com/2011/12/viehboeck_wps.pdf
- Linux man page, http://linux.die.net/man/
- List of WPS Pixie Dust vulnerable device, https://docs.google.com/spreadsheets/d/1tSlbqVQ59kGn8hgmwcPTHUECQ3o9YhXR91A_p7Nnj5Y
- Kismet tool, https://www.kismetwireless.net/presentations/5hope-kismet.pdf
- Generic brute-force time calculation, http://calc.opensecurityresearch.com/