Advanced Networking Lab

Jakob Struye Thomas Hendriks

March 1, 2017

Lab 4

Wireless Security

In this lab, you will have a look at the impact of various security measures which are available for wireless networks. The main goal is to show what the effect is of a certain security method and what a possible attacker/sniffer can do on the protected network. In the first lab, you already performed some sniffing using the monitor mode. It should be clear that unless we take some security measures, it is quite easy to intercept and read any management information and data from open networks. It is even quite easy to perform **dos!** (**dos!**) attacks on open networks as you will see at the end of this lab.

In the first part of the lab, you will investigate the more basic security features like hiding a network and **mac!** (**mac!**) filtering. In the second part, you will have a look at encryption mechanisms and try to hack some of these. In the last part, we will introduce some basic attacks. These attacks are introduced to illustrate the mechanisms behind wireless networks and should only be used for this purpose.

4.1 Basic Security Measures

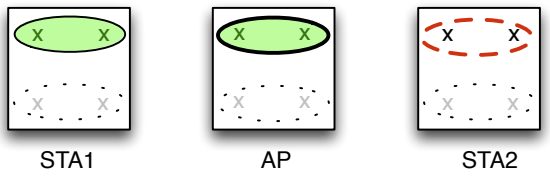

Figure 4.1: MAC filter setup

Exercise 1: Network hiding

A first step in preventing someone to use your wireless network is to hide it from being detected by a regular scan. Hiding a **ssid!** (**ssid!**) ensures it will not show up in the list of available **ap!**s (**ap!**s) when a scan is performed. You will however show that the clever sniffer is still able to detect the network.

1. Set up your network as illustrated in figure **??**, but do not set up the **wnic!** (**wnic!**) of **sta!** (**sta!**)1 yet.
2. Start a trace on the monitor interface (`/mnt/L4-1-1.sta!2.pcap` of **sta!**2. While the trace is running, bring up the **wnic!** from **sta!**1. Make sure the station is associated with the **ap!** before you stop your trace.
3. Indicate in which frames the **ssid!** is visible. Give a packet ID of an example packet in your trace file for each.
   L4-1-1

   ---

   5 broadcast beacon frame

   2528 probe 2529 probe reply 2694 association request van sta1 naar ap waar ssid in staat

   ---

4. The next step is to hide the **ssid!**. This can be done adding the following line to the hostapd.conf file: `ignore_broadcast_ssid=1`. Bring the interface down on **sta!**1, restart `hostapd` on the **ap!** and repeat steps **??** and **??**. Save your trace to `/mnt/L4-1-2.STA2.pcap`
5. Compare the two trace files and indicate how the **ssid!** is hidden. In what way can a **ssid!** still be detected?
   L4-1-2

   278: broadcasted probe request for the ssid wmn-1-A sent by STA1 279: the answer is from AP to STA1 and contains the ssid as well

   in the association request ¡pkt id¿ the ssid is still listed. further communication such as association requests also contain the ssid

   only broadcasts are now gone such as pkt 5 in the first L411

   – only authentication seen

Exercise 2: **mac!** filtering

Continuing from the setup you used in the previous exercise, a possible intruder has still complete access to our network. In the next step, you will prevent him from joining the network by **mac!** filtering.

1. The management of the **mac!** filtering is also performed by modifying the `hostapd.conf` file at the **ap!**. The entries needed for **mac!** filtering are:

   macaddr_acl= controls the behaviour of the mac filtering and expects an integer value from the following list:

   - **0** accept unless in deny list,
   - **1** deny unless in accept list,
   - **2** use external RADIUS server (accept/deny lists are searched first

   accept_mac_file= append the name of the file here that contains the list of **mac!** addresses to accept.

   deny_mac_file= append the name of the file here that contains the list of **mac!** addresses to deny.

2. Configure the **ap!** in such a way that the **mac!** address of the `wlan0` interface of **sta!**1 is blocked from the network. SSID broadcasting may be enabled again.

3. Now bring down the **wnic!** at **sta!**1 again, restart a capture session on **sta!**2 and save it to `/mnt/L4-2-1.sta!2.pcap` and bring the **wnic!** back up at **sta!**1 when this capture session is active. Compare the results from this capture file with the first one you made and discuss the differences between both. Illustrate with packet IDs.
   L4-2-1

   ---

   in de eerdere kreeg authentication als reply succesful, nu is het unspecified failure ¡pkt ids¿

   ---

4. Of course, in a real world setup, blocking unwanted MAC addresses is not feasible. One would use the option with the `accept_mac_file` parameter, allowing only known **mac!** addresses to connect. How would you circumvent this security measure? If an attacker has joined the network, will this have an effect on legitimate stations on that network?
   L4-2-2

Mac address spoof. either two mac addresses will be on the network causing problems and crashing the network or the spoofed mac address will not be present at the time granting the attacker access to the network.

4.2 Encryption

4.2.1 wep!

The security techniques from the previous section will "prevent" a possible attacker from joining the network by hiding the network or preventing him access to the network. However, it is still possible to read any communication over the network by using a **wnic!** in monitor mode. Encryption mechanisms will counter this and obfuscate the network traffic for an eavesdropper. You will take a look at **wep!** (**wep!**) here as this method is still sometimes being used although it is very vulnerable to attacks. **wpa!** (**wpa!**)2 is far more resilient.

Exercise 3: **wep!** encryption

wep! was the first available encryption method, standardised together with the IEEE 802.11 standard. It was however quickly been proven insecure. In fact, as of 2004, **wep!** was declared deprecated by IEEE because of its flaws. In this exercise, you will first set up a **wep!** encrypted network to check what is exactly encrypted. In the next exercise you will crack the encryption key.

Because **wep!** is quite old, the tools you will use assume that IPv4 is used, as they will use **arp!** (**arp!**) messages. Therefore, for this exercise, you will configure IPv4 addresses instead of IPv6 addresses.

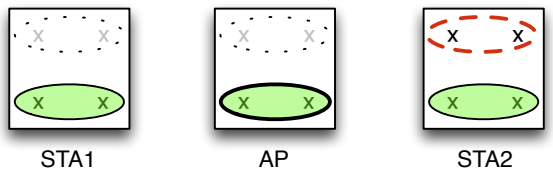

Figure 4.2: WEP setup

1. Start from the network setup as shown in figure **??**. Disable any **mac!** filter that is still in place.

2. In order to avoid capturing needless IPv4 traffic, disable the `avahi-daemon` on all three nodes:
   `wmn!:~# killall avahi-daemon`

3. Create a tracefile on the monitor interface of **sta!**2 (`/mnt/L4-3-1.sta!2.cap`), containing the scan, authentication and association phase of **sta!**1 and a 5 packet ping session between **sta!**1 and **sta!**2. Use `10.0.gid!.1/24` for **sta!**1 and `10.0.gid!.2/24` for **sta!**2.
   `sta!:~# ip addr add 10.0.gid!.X/24 broadcast 10.0.gid!.255 dev wlan1`

4. Make sure that **sta!**2 is connected to the **ap!** and that the **wnic!** of **sta!**1 is down before the start of the capture session. To ping, use `ping` instead of `ping6`. This trace will be our reference trace.

5. Activate the **wep!** encryption on the **ap!**. You can choose a random key as long as it is either 5 or 13 characters long ("arandomwepkey" in this example):
   Change the `hostapd.conf` file, so that it includes the following lines:
   `wep_key0="arandomwepkey"`
   `wep_default_key=0`

   The WEP key must be ***exactly 5 or 13*** characters long!

6. Make sure the **ap!** is running. Connect **sta!**2 to the **ap!**:
   `sta!2:~# iw dev wlan1 connect wmn-0-A key 0:arandomwepkey`

7. Make sure the interface of **sta!**1 is down and then start a new trace on the monitor interface of **sta!**2. Save it to `/mnt/L4-3-1.sta!2.wep.pcap`.

8. Perform the same test as in step **??**, but be sure to add `key 0:arandomwepkey` when you connect **sta!**1 to the network.

9. Compare both trace files. In the second trace file, you should no longer be able to see the `ping` session in the clear.

   - Which frame types are encrypted?
     L4-3-1

     ---

     icmp packets

   - Which part of a frame is encrypted and which part is still readable?
     L4-3-2

the payload is encrypted, only data shows up in wireshark ip addresses enc as well b/c part of payload = ip header

Refer to specific frames from the second trace file to illustrate your answers.

Exercise 4: Cracking the **wep!** keys

Without going into details about the flaws in **wep!**, the main problem arises from the fact that traffic keys are easily repeated on a busy network. A traffic key is the key actually used by the encryption algorithm RC4. As RC4 is a stream cipher, the same key should not be used twice. Therefore, the shared **wep!** key, which is configured in each **ap!** and **sta!**, is combined with a random **iv!** (**iv!**). This **iv!** is transmitted in plaintext between two wireless clients so the receiver can recreate the encryption key used for a specific packet, by combining the chosen (shared) network key with the received **iv!**. As the **iv!** has a length of only 3 bytes, the number of possibilities is relatively low and in busy networks, the same **iv!**s are frequently reused, weakening the cryptographic strength of **wep!**. Furthermore, statistical relations between used keys and encrypted text can be exploited to find the used key.

Various methods have been developed to crack a **wep!** key, but you will just illustrate the mere ease with which a **wep!** encrypted network can be hacked using the aircrack tool [**?**]. This tool implements various approaches, but you will limit ourselves to **ptw!** (**ptw!**) method. The attack is based on the fact that the first 16 bytes of an **arp!** packet are fixed. This is also the reason we switched to IPv4 for this exercise. . . For more details read [**?**], specifically sections 1 and 5 for a general overview. Exploiting the relation between cleartext and the captured **iv!**s makes it possible to quite easily determine the network key.

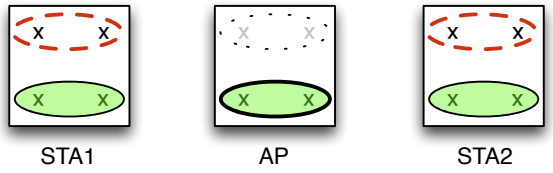

Figure 4.3: WEP cracking setup

1. Before you start to crack the configured **wep!** key, first have a second look at the traces from the previous exercise. Using Wireshark [**?**], you can decrypt an encrypted trace. Using the menu Edit→ Preferences → Protocols → IEEE 802.11

you can specify the key in HEX. To convert your key to HEX, use e.g. the following website: http://www.dirtymonday.net/key_convert.html For our example it will be 6172616E646F6D7765706B6579.
Make sure you also check the 'decrypt packets' box. Select an ICMP request in both trace files (unencrypted and encrypted, but decrypted by Wireshark) and identify the extra information added to a frame in order to encrypt it. Also indicate where in the frame this information is added.
L4-4-1

Difference : WEP parameters 4 bytes + 4 bytes that dont show up when you look at wep params in the pkt IV en key index voor data, WEP ICV na data maar voor fcs

---

2. Start from the **wep!** encrypted network you created in the previous exercise.

3. Add an additional monitor interface on **sta!**1, resulting in the network scheme shown in figure **??**. The monitor interface on **sta!**2 will be used to perform the actual attack. **arp!** packets will be injected into the network in order to trigger **arp!** responses which will contain new **iv!**s with every response . These packets will be captured and stored in files. In parallel with this capturing of new **arp!** responses and the associated **iv!**s, we can start the process of analyzing these packages and searching for the key. The monitor interface on **sta!**1 will be used to monitor the channel so we can afterwards take a look at what happened.

4. Start a packet capture to `/mnt/L4-4-2.sta!1.pcap` on the monitor interface of **sta!**1.

5. Start the **iv!** capture process at **sta!**2. This command will store its captured frames in files `output-xy.cap` and `output-xy.txt`. Make sure to run this command with your active directory set to the `/mnt` folder, in which a remote directory is mounted.
`STA2:~# airodump-ng --band a --channel x --bssid <mac! addr ap!> -w output wlan0`
This should give you an overview screen where at least the incoming beacon count is rising. If no activity is shown, bring the interface down and then bring it up again and retry. This command will store its captured frames in files `output-xy.cap` and `output-xy.txt`. Make sure to run this command in `/mnt`, where a remote location is mounted.

6. On a new terminal, perform a fake authentication to the **ap!** from the attacking machine (**sta!**2). This ensures the frames we will be injecting for `wlan1` will be accepted by the **ap!**:

```
STA2:~# aireplay-ng --fakeauth 0 -e wmn-gid!-A -a <mac! addr ap!> -h <mac!
addr wlan0> wlan0.
```

7. It is now time to actively inject packets into the wireless network. This is done by the following command: `STA2:~# aireplay-ng --arpreplay -b <MAC addr AP> -h <MAC addr wlan0> -x 1024 -o 512 wlan0`

   This command will monitor incoming packets and when an **arp!** request passes, it will re-inject it at a rate of several 100 frames/s. You should be able to see this in the output from `airodump-ng`.

   
   Due to a bug in the driver, the packet injection rate will not be high enough. You should run 10 of the above `aireplay-ng` commands in parallel to generate enough packets. You can do so easily by editing the `wepcrack.sh` script found on the nodes. Insert the correct **mac!** addresses in that script and run it. To stop the replay attack, simply type `killall aireplay-ng`.

8. In order to get an **arp!** request, just start a ping from **sta!**1 to **ap!**. This should trigger an **arp!** request. If the **arp!** tables already contained the IP addresses from both hosts (check it with `STA1:~# ip neigh`), remove the entry using `ip neigh flush dev wlan1`.

9. To crack the key, the tool `aircrack-ng` can be used. This tool processes the files generated by `airodump-ng`. The needed number of packets to crack the key depends on the bit length of the key. In general, you need at least 20,000 packets for a 64-bit key and between 40,000 and 85,000 for a 128-bit key. To start the analysis, run the following command in the folder containing the output from `airodump-ng`:
   `STA2:~# aircrack-ng -b <mac! addr ap!> output*.cap -q`

10. If you captured enough packets, after a while, the `aircrack-ng` tool should provide you with the **wep!** key used in your network. Copy its output here:
    L4-4-2

    STA2:/mnt aircrack-ng -b 00:0e:8e:20:83:df output-02.cap -q KEY FOUND! [61:72:61:6E:64:6F] (ASCII: arandomwepkey) Decrypted correctly: 100

    ---

11. Using the following `tshark` command on a lab PC, you can get a list of **iv!**s used during your session.
    `tshark -r <trace file> -T fields -e wlan.wep.iv`

12. How many **iv!**s did you capture? And how many of them occured at least twice?
    L4-4-3

    We captured 1100000 +- iv's with 735095 unique ivs see tshark.out

    ---

13. When handing in the report, you must include the `output-*.csv` and `*.netxml` files. `output-*.cap` is not required, as it will be quite large. Limit the trace file of **sta!**1 to only a few (e.g. a hundred) of packets that show the **arp!** injection in progress.

4.2.2 wpa!2

As you have demonstrated in the previous exercise, **wep!** encryption is easily circumvented. You will now take a look at **wpa!**2 encryption. You will no try to crack it, as **wpa!**2 uses the **ccmp!** (**ccmp!**) algorithm with **aes!** (**aes!**) encryption, which is extremely resilient to attacks. To give you an idea, the best known method until now to crack a 128 bit **aes!** key, which is used by **wpa!**2, still takes at least $2^{126.1}$ computations [**?**].

A security flaw concerning **wps!** (**wps!**) has been discovered that makes **ap!**s running **wps!** vulnerable to a relatively easy attack (which would still take far too much time for this lab exercise) [**?**]. This, of course, does not hold true for **ap!**s that do not have **wps!** enabled. If you want to read up on how **wpa!**2 works and on the evolution in security from **wep!** over **wpa!** to **wpa!**2, you should read [**?**].

Exercise 5: **wpa!**2

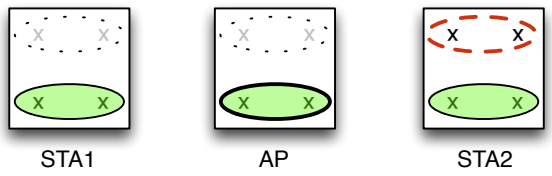

Figure 4.4: **wpa!** setup

1. Start from the **wpa!**2 enabled network setup as seen in figure **??**.
2. On the **ap!**, to enable **wpa!**2, make sure that the line `wpa=0` in the `hostapd.conf` file is changed to `wpa=2`, and add the following lines to `hostapd.conf`:

```
wpa_passphrase=aRandomPassphrase
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
```

Use your own passphrase if you like. Unlike **wep!**, it is not restricted to a certain amount of characters.

3. Set the IP addresses of `wlan1` on **sta!**1 and **sta!**2 to `fc00:gid!::1/64` and `fc00:gid!::2/64`, respectively.
4. Connect **sta!**2 to the network. In order to do so, modify the `wpa_supplicant.conf` file so that it includes the correct **essid!** and passhprase. Afterwards, run:
   `sta!2:~# wpa_supplicant -B -c/root/wpa_supplicant.conf -iwlan1`

   You might get an error like `ioctl[SIOCSIWENCODEEXT]: Invalid argument`. You can safely ignore this error.
5. Make sure the monitor interface on **sta!**2 is configured and start a trace. Save it in `/mnt/L4-5-1.sta!2.pcap`.
6. Now bring the interface on **sta!**1 up an do `ping6 -c 5` to **sta!**2.
7. Which frame types are encrypted? Give examples.
   L4-5-1

8. Which part of a frame is encrypted and which part is still readable?
   L4-5-2

9. Compare this trace file with the one made in exercise **??**. Compare the authentication/association phase of **sta!**1. What are the similarities and what are the differences? Refer to packet IDs!
   L4-5-3

10. Now, compare an encrypted frame in both traces. Indicate where they differ.
    L4-5-4

4.3 Denial of Service Attacks

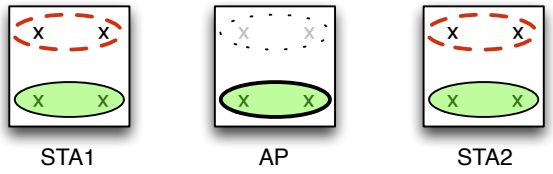

Figure 4.5: DoS setup

In the previous section, you have gained access to a **wep!** encrypted network by actively injecting packets and cracking the key. This kind of attack makes it possible that you, the attacker, can afterwards associate with the network and sniff all traffic on that network. Alternatively, you can also use the network yourself as if you were a normal user.

Within this section, you will illustrate a **dos!** attack that shows the vulnerability of a wireless network. You will not gain access to the network as such, but you will disturb normal network operation. Tools to do so are freely available and current encryption techniques cannot prevent these attacks as they make use of management frame injection. As you have seen, management frames are not encrypted and thus can be easily injected into a wireless network.

Exercise 6: Disassociation Attack

The principle behind a disassociation attack is quite easy. A disassociation frame is sent by either a **sta!** or **ap!** to signal the disconnection from the wireless network. For instance, it is sent by a **sta!** just before you disable an associated **wnic!**. Although the sending of a disassociation message is not mandatory by the standard, at reception of such frame, a client should consider its connection lost and an **ap!** will remove a **sta!** from the list of connected stations. Injecting a malicious disassociation frame for a wireless node will thus force this node to scan for and associate again to its **ap!**, resulting in a connectionless period. If the disassociation frame is sent periodically, this can render the wireless network useless to the attacked **sta!**.

1. Start from the network scheme as illustrated in figure **??**.
2. Be sure to have configured your network with **wpa!**2 encryption.
3. Configure the IP addresses of **sta!**1 and **sta!**2 to `fc00:gid!::1/64` and `fc00:gid!::2/64` respectively.

4. On the monitor interface of **sta!**1, start a capture session and save it to `/mnt/L4-6-1.sta!1.pcap`.
5. Start a `ping6` from **sta!**1 to **sta!**2.
6. On **sta!**2, use the following command to send a disassociation to **sta!**1:
   `STA2:~# aireplay-ng --deauth 1 wlan0 -a <MAC addr AP> -c <MAC addr STA1>`
7. What is the result for the ping?
   L4-6-1

   _____

   _____
8. Open the trace file and explain using this trace what exactly happened.
   L4-6-2

   _____

   _____

Acronyms

AES Advanced Encryption Standard

AP access point

ARP Address Resolution Protocol

CCMP Counter Cipher Mode with Block Chaining Message Authentication Code Protocol

DoS denial of service

ESSID extended **ssid!**

grID group ID

IV initialization vector

MAC Media Access Control

PTW Pychkine, Tews, Weinmann

SSID Service Set ID

STA station

WEP Wired Equivalent Privacy

wmn wireless mesh node

wnic wireless **nic!**

WPA Wi-Fi Protected Access

WPS Wi-Fi Protected Setup