Manuel d'utilisation du projet GSW (Géolocalisation de Smartphone par WiFi)

BOULMONT Clément - BRASSEUR Corentin
COURTIAL Azad - MENET Alan

Master 2 - Informatique
2023-2024

Projet Thématique 1

I / Introduction et contexte du projet GSW

Matériels utilisés :

- Raspberry pi en guise de capteur (éventuellement un qui sert d'unité centrale)
- Dongles WiFi
- Nos téléphones personnels
- Ordinateur (optionnel car il sert d'unité centrale)

Lieu d'expérimentation : CURI 306 (La D02 n'étant pas disponible car il y avait cours)

A savoir : pour étalonner et localiser, si nous optons pour l'utilisation de la version contenant les données (celle-ci étant stockée dans le dossier raspberry sous les noms etalonnageraspv2 et capteur1OLD sur github), il est impératif de ne pas être connecté au réseau de l'UPJV, car il préfère limiter l'accès aux données. Par conséquent, nous devrons mettre en place un serveur local créé par nous-mêmes. Dans notre situation, cela implique de partager une connexion 4G.

II / Déroulement de l'expérience

a) Procédure

Nous avons associé des dongles wifi à nos raspberry pi. En effet, ces dernières ne possèdent pas de mode monitor nécessaire pour notre projet. Sur chacune des raspberry, nous installons raspberry pi os, aircrack-ng, tcpdump et netcat. En utilisant tcpdump, nous pouvons analyser les paquets en ligne de commande. Grâce à Aircrack, nous sommes en mesure de surveiller les réseaux WiFi. Netcat est un protocole qui nous permet d'écrire et de lire des données dans le réseau. L'os de la raspberry répond parfaitement à nos besoins dans notre projet. Sur une autre machine aussi bien raspberry qu'ordinateur, nous installons les mêmes logiciels que nous installons sur nos raspberry. Une fois que nous avons fait celà nous pouvons procéder à la mise en place. Pour installer nos capteurs (raspberry) de manière optimale, nous utilisons un appareil pour l'étalonnage qui nous permet de connaître l'adresse MAC à l'aide de la commande ifconfig. Nous exécutons sur toutes les raspberry, ainsi que l'unité centrale la commande suivante :

```
$ sudo airmon-ng start wlan1
```

La commande ci-dessus est utilisée pour démarrer le mode monitor sur une interface réseau sans fil (ici wlan1) sous Linux à l'aide de l'outil Airmon-ng.

Remarque : dans le cas où l'unité centrale n'est pas une raspberry, remplacez wlan1 par la bonne interface (sudo airmon-ng affiche toutes les interfaces). Cette commande aura comme conséquence de ne plus avoir internet sur l'interface en question

b) Codes utilisés sur nos raspberry

genereCapteur.sh :

```
#!/bin/bash

cat <<EOF > "capteur$1.sh"

# contenu du fichier capteurX.sh générer par la ligne ci-dessus :
#!/bin/bash

while true; do
    data=\$(sudo tcpdump -i wlan1mon -e type data -c 1
2>/dev/null)
    mac=\$(echo "\$data" | cut -d" " -f13 | cut -d ":" -f2-)
    echo "\$data" | awk '{print "$1;" \$1 ";" \$7 ";" \$16}' | nc
-w0 192.168.43.52 12345
done
EOF
```

Ce script génère un script nommé “capteurX.sh” (X étant l’argument passé au script). Le fichier capteurX.sh permet la capture de données réseau. Cette dernière une fois capturé envoie les informations spécifiques sur un paquet capturé à une adresse IP (ici 192.168.43.52) et un port spécifié (12345 dans cet exemple).

Remarque : Les adresses IP et les macs sont à modifier en conséquence des appareils utilisés. Nous avons aussi au préalable utilisé un script étalonnage nous permettant de placer les raspberry.

Afin de rendre exécutable notre script nous utilisons la commande suivante :

```
$ chmod 777 "capteur$1.sh"
```

c) Codes utilisé sur l'unité centrale

listener.sh :

```
#!/bin/bash

while true; do
      nc -l 12345
done
```

Netcat est utilisé dans ce programme Bash pour créer un serveur de socket et le connecter au port 12345. La commande nc -l 12345 permet de surveiller le port 12345 en attendant une connexion entrante. Le script est en boucle infinie car il est basé sur une structure qui se déclenche uniquement lorsque c'est vrai. Il continuera d'écouter sur le port 12345 pour toute connexion entrante, mais ne fera rien de plus avec les données reçues jusqu'à ce que le script soit exécuté.

parser.sh :

```
#!/bin/bash

while read line; do
    mline=$(echo "$line" | sed 's/SA://')
    echo "$mline" >> "donnees.csv"
    mac=$(echo "$mline" | cut -d ";" -f4)
    if ! grep -q "$mac" "whitelist.txt"; then
        echo "$mac" >> "whitelist.txt"
        echo "$mac" >> "new.txt"
    fi
done
```

La commande read est utilisée par ce programme Bash pour lire chaque ligne d'une entrée. Il retire le préfixe 'SA' s'il est présent, puis il insère la ligne modifiée dans un fichier nommé 'donnees.csv'. La commande cut permet d'extraire l'adresse MAC de la ligne modifiée en utilisant ';' comme délimiteur de champ, et de stocker cette adresse MAC dans la variable mac. Ensuite, le script examine si l'adresse MAC est déjà présente dans le fichier 'whitelist.txt'. Il insère cette adresse MAC dans le fichier 'whitelist.txt' et dans le fichier 'new.txt'.

Remarque : Sur le moniteur central nous devons exécuter la commande :

```
$ ./listerner.sh | ./parser.sh
```

Cela nous permet de récupérer les données tout en les traitant et les interpréter sur un fichier csv. Voici le format final de notre fichier csv :

Numéro capteur	Heure	Puissance	Adresses MAC
1	00:00:00.00000	-1 dBm	00:00:00:00:00:00

Au niveau interprétation, cela signifie que plus la puissance en dBm est proche de 0, plus elle est proche d'un capteur particulier. Ce principe sera utilisé plus tard dans la partie Python du projet.