# Android Security Lab 1

**Groupe :** Quentin Dubois, Vinh Faucher, Florian Latapie

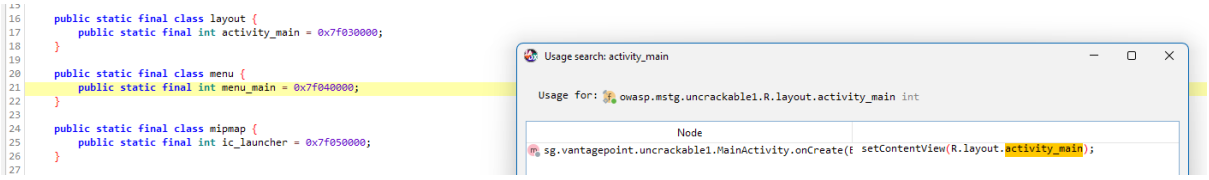
GitHub : [github.com/FlorianLatapie/PNS-SI5-S9_ProtApps_Lab1](github.com/FlorianLatapie/PNS-SI5-S9_ProtApps_Lab1)

Objectif : Essayez de trouver un secret caché dans une application Android

## Étape 1 : Analyse statique

### Question : Trouvez « MainActivity » dans section « Resources »

Dans la classe "R", il suffit de faire un clic droit sur "activity_main" et de sélectionner "find usage", afin de voir où l'occurrence "activity_main" est utilisée dans le code. Nous trouvons que cette variable est utilisée un seul endroit, dans la MainActivity dans le package sg.vantagepoint.uncrackable1.




### Question : Trouvez le nom du package grâce à la classe R

Pour trouver le nom du package, il nous suffit de regarder la première ligne de la classe R. Cette ligne nous indique que le nom du paquet est owasp.mstg.uncrackable1. Ce paquet correspond au paquet de l'application Android. Nous pouvons également confirmer cette information en consultant le fichier AndroidManifest.xml dans le dossier Ressources. Dans ce fichier, le nom de package de l'application est stocké dans l'attribut "package" dans la balise "manifest" (balise racine).

### Question : Parcourir les fichiers sources et essayez de comprendre:

#### La détection root

On s'intéresse aux 3 méthodes de la classe : sg.vantagepoint.a.c

- Méthode a : Vérifie la présence du fichier "su" dans le PATH de l'utilisateur.
- Méthode b : Vérifie les Build.Tags existent et contiennent la valeur "test-keys". Si les Build.TAGS contiennent la valeur "test-keys", cela veut dire que le kernel a été signé par une clé générée par une tierce partie.
- Méthode c : Vérifie dans le système de fichier les applications connues qui permettent de devenir "root" sur un appareil Android.

#### La logique de chiffrement

**Classe :** sg.vantagepoint.a.a

**Méthode a :**

Cette méthode déchiffre une donnée avec pour entrées, la clé secrète (bArr) et la donnée chiffrée (bArr2), et renvoie la donnée déchiffrée.

**Classe :** `sg.vantagepoint.uncrackable1.a`

**Méthode a :**
Cette méthode permet de vérifier le mot de passe entré par l'utilisateur dans l'application Android avec le secret hardcodé dans l'application. Cette méthode utilise une fonction interne : `sg.vantagepoint.a.a.a`. La clé secrète est `stringIntoByteArray("8d127684cbc37c17616d806cf50473cc")` et le mot de passe chiffré est `Base64.decode("5UJiFctbmgbDoLXmpL12mkno8HT4Lv8dlat8FxR2GOc=", 0)`.

### La logique de principale de l'application

- Si l'appareil est détecté comme rooté, alors
    - L'application affiche un pop-up d'erreur, puis se ferme
- Sinon, l'application crée la `MainActivity` à l'aide de son layout.
- L'application propose ensuite à l'utilisateur d'entrer le mot de passe.
    - L'application vérifie l'entrée utilisateur à chaque essai.
    - En cas de réussite,
        - L'application affiche un pop-up de succès,
    - Sinon on obtient un pop-up d'échec.

### Question : Qu'est-ce qui serait nécessaire pour récupérer le secret ?

Nous avons besoin de recréer le comportement effectué par la méthode « a » de la classe `sg.vantagepoint.a.a`. Pour réaliser cette tâche, nous avons utilisé du code Java natif, pour des raisons de simplicité.

Une fois le code exécuté, nous obtenons le secret : "**`I want to believe`**".

Le fichier de code est trouvable dans le projet GitHub, sous le nom `Main.java`.

## Étape 2 : Analyse Dynamique

Notre première solution (`hook_m1.js`) a été de hooker la méthode `sg.vantagepoint.a.a.a`, pour afficher le résultat renvoyé par la méthode. Pour cela, nous avons dû hooker chacune des méthodes s'occupant de la vérification de root, afin de pouvoir accéder à l'application. Une fois l'accès, nous avons besoin de rentrer un mot de passe quelconque et bingo ! Le secret s'affiche dans la console. Nous pouvons ensuite tester le mot de passe obtenu, et nous obtenons un message de succès.

La deuxième solution (`hook_m2.js`) qui a été présentée par le professeur et que nous avons ensuite essayé de reproduire, consiste à faire exécuter la fonction `sg.vantagepoint.a.a.a`, avec les bons paramètres. Chaque étape de transformations est réalisée directement par l'application en appelant la fonction correspondante. Cela permet de ne pas altérer les fonctionnalités de l'application, tout en utilisant ses fonctions internes !