HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

SOS

Sécurité des systèmes

LAB03

Analyse d'application mobile

Fabrice Caralinda

Jeudi 6 juin 2019

Noms : Garanis, Mizutani

Prénoms : Nikolaos, Nathanaël

Introduction :

L'objection de ce laboratoire est de mettre en pratique la théorie vue durant le cours de sécurité mobile.

Rendu Attendu :

- Ce laboratoire doit être réalisé **par groupe de deux au maximum**
- Un rapport répondant de manières détaillées aux questions posées dans ce document doit être remis à la fin du travail
- Le rapport au format **PDF** doit être transmis par mail au plus tard le **13.06.2019** à **16h30** à :
  - fabrice@scrt.ch
  - lucie.steiner@heig-vd.ch
- Chaque jour de retard réduira la note d'un point. Le nom du document doit respecter le format suivant : **sos-lab03_nom1_nom2.pdf**

Liste des questions :

Ce laboratoire comporte un total de **48 points.**

Question 1. Analyse statique (13 pts)

[a] Quels outils avez-vous utilisé pour « depacker » votre APK (détails de la manipulations) **(2 pts)**

apktool d babyrev.apk -o output

unzip babyrev.apk -d output (ne désassemble pas les .dex)

[b] Lister les fichiers contenus dans l'APK. **(1 pt)**

AndroidManifest.xml, apktool.yml, original/, res/, smali/, smali, classe2/

[c] Énumérer le point d'entrée (entrypoints) de l'application (détails de la manipulations). **(1 pt)**

On trouve le point d'entrée *com.mobisec.babyrev.MainActivity* de l'application dans l'élément *activity* de l'élément *application* dans AndroidManifest.xml.

On peut aussi utilizer la commande *aapt dump badging babyrev.apk | grep activity*

[d] Quel nom de « package » utilise l'application « babyrev ». Lister deux manières de récupérer cette information. **(2 pt)**

L'application « babyrev » utilise le nom de « package » *com.mobisec.babyrev*.

On peut le récupérer en lisant l'attribut package de l'élément manifest dans AndroidManifest.xml ou avec la commande *aapt dump badging babyrev.apk | grep activity*.

[e] Lister les permissions requises par l'application pour se lancer sur un appareil. **(1 pt)**

La commande *aapt dump permissions babyrev.apk* ne nous renvoie que le nom du package utilisé par l'application. On en déduit donc que celle-ci ne demande aucune permission pour fonctionner.

[f] Qu'utilise-t-on pour désassembler le fichier « `classes.dex` ». Qu'obtient-on après cette manipulation (détails de la manipulation) **(3 pt)**

Si on utilise *apktool* les fichiers ".dex" sont déjà désassemblés (car *apktool* utilise *baksmali*). Pour désassembler le fichier "classes.dex" obtenu avec un unzip, on utilise la commande "*baksmali classes.dex -o output*" et on obtient des fichiers ".smali".

[g] Qu'utilise-t-on pour décompiler le fichier « `classes.dex` ». Qu'obtient-on après cette manipulation (détails de la manipulation) **(3 pt)**

Pour décompiler le "classes.dex" on utilise la commande "*jadx -d out classes.dex*". On obtient les sources Java correspondant au code décompilé.

Question 2. Reverse Engineering (22 pts)

[a] D'après vous pourquoi la méthode « checkFlag » de la classe « FlagChecker » n'a pas pu être décompilée. (utiliser l'outil « jadx-gui » pour récupérer le code source de l'application) **(2 pts)**

La méthode *checkFlag* n'a pas pu être récupérée car la classe *FlagChecker* est obfusquée.

[b] Décrire la fonctionnalité de cet extrait de code. **(1 pt)**

```
        r0 = "HEIG-VD{";
        r0 = r11.startsWith(r0);
        r1 = 0;
        if (r0 != 0) goto L_0x000a;
    L_0x0009:
        return r1;
```

Cet extrait de code vérifie que **r11** commence par « *HEIG-VD{* ».

[c] Décrire la fonctionnalité de cet extrait de code. **(1 pt)**

```
    L_0x000a:
        r0 = new java.lang.StringBuilder;
        r0.<init>(r11);
        r0 = r0.reverse();
        r0 = r0.toString();
        r0 = r0.charAt(r1);
        r2 = 125; // 0x7d float:1.75E-43 double:6.2E-322;
        if (r0 == r2) goto L_0x0020;
    L_0x001f:
        return r1;
```

Cet extrait de code vérifie que **r11** termine par « *}* ».

[d] Décrire la fonctionnalité de cet extrait de code. **(1 pt)**

```
    L_0x0020:
        r0 = r11.length();
        r2 = 35;
        if (r0 == r2) goto L_0x0029;
    L_0x0028:
        return r1;
```

Cet extrait de code vérifie que la longueur de **r11** vaut 35.

[e] Décrire la fonctionnalité de cet extrait de code. **(1 pt)**

```
    L_0x0029:
        r0 = r11.toLowerCase();
        r2 = 8;
        r0 = r0.substring(r2);
        r3 = "this_is_";
        r0 = r0.startsWith(r3);
        if (r0 != 0) goto L_0x003c;
    L_0x003b:
        return r1;
```

Cet extrait de code vérifie que le flag à l'intérieur de « *HEIG-VD{...}* » commence par "*this_is_* " en ignorant la casse.

[f] Décrire la fonctionnalité de cet extrait de code. **(3 pt)**

```
    L_0x003c:
        r0 = new java.lang.StringBuilder;
        r0.<init>(r11);
        r0 = r0.reverse();
        r0 = r0.toString();
        r0 = r0.toLowerCase();
        r3 = 1;
        r0 = r0.substring(r3);
        r4 = 2131427368; // 0x7f0b0028 float:1.847635E38 double:1.053065039E-314;
        r4 = r10.getString(r4);
        r0 = r0.startsWith(r4);
        if (r0 != 0) goto L_0x0060;
    L_0x005f:
        return r1;
```

Cet extrait de code vérifie que la fin du flag corresponde à une certaine chaîne de caractère localisée.

[g] Décrire la fonctionnalité de cet extrait de code. **(1 pt)**

```
L_0x0060:
    r0 = 17;
    r0 = r11.charAt(r0);
    r4 = 95;
    if (r0 != r4) goto L_0x0116;
```

Cet extrait de code vérifie que le 18e caractère soit un underscore.

[h] Décrire la fonctionnalité de cet extrait de code. **(3 pt)**

```
L_0x006a:
    r0 = getY();
    r4 = (double) r0;
    r0 = getX();
    r6 = (double) r0;
    r0 = getY();
    r8 = (double) r0;
    r6 = java.lang.Math.pow(r6, r8);
    r4 = r4 * r6;
    r0 = (int) r4;
    r0 = r11.charAt(r0);
    r4 = 4611686018427387904; // 0x4000000000000000 float:0.0 double:2.0;
    r6 = java.lang.Math.pow(r4, r4);
    r4 = java.lang.Math.pow(r6, r4);
    r4 = (int) r4;
    r4 = r4 + r3;
    r4 = r11.charAt(r4);
    if (r0 == r4) goto L_0x0098;
L_0x0096:
    goto L_0x0116;
```

Cet extrait de code vérifie que le 15e caractère soit un underscore. Avec X = 2, Y = 3, Z = 5.

[i] Décrire la fonctionnalité de cet extrait de code. **(3 pt)**

```
L_0x0098:
    r0 = r11.toUpperCase();
    r4 = getY();
    r5 = getX();
    r4 = r4 * r5;
    r5 = getY();
    r4 = r4 * r5;
    r5 = getZ();
    r5 = (double) r5;
    r7 = getX();
    r7 = (double) r7;
    r5 = java.lang.Math.pow(r5, r7);
    r7 = 4607182418800017408; // 0x3ff0000000000000 float:0.0 double:1.0;
    r5 = r5 - r7;
    r5 = (int) r5;
    r0 = r0.substring(r4, r5);
    r0 = bam(r0);
    r4 = "ERNYYL";
    r0 = r0.equals(r4);
    if (r0 != 0) goto L_0x00cf;
L_0x00ce:
    return r1;
```

Cet extrait de code vérifie que la partie du flag entre le 19e et 25e caractère inclus vaut « *really* » en ignorant la casse.

[j] Décrire la fonctionnalité de cet extrait de code. **(1 pt)**

```
L_0x00cf:
    r0 = r11.toLowerCase();
    r4 = 16;
    r0 = r0.charAt(r4);
    r5 = 97;
    if (r0 == r5) goto L_0x00de;
L_0x00dd:
    return r1;
```

Cet extrait de code vérifie que le 17e caractère vaut « *a* ».

[k] Décrire la fonctionnalité de cet extrait de code. **(2 pt)**

```
L_0x00eb:
    r0 = r11.toUpperCase();
    r5 = 25;
    r0 = r0.charAt(r5);
    r5 = r11.toUpperCase();
    r4 = r5.charAt(r4);
    r4 = r4 + r3;
    if (r0 == r4) goto L_0x0101;
L_0x0100:
    return r1;
```

Cet extrait de code vérifie que le 26e char est égal au 27e + 1.

[l] Décrire la fonctionnalité de cet extrait de code. **(1 pt)**

```
L_0x0101:
    r0 = getR();
    r4 = r11.length();
    r4 = r4 - r3;
    r2 = r11.substring(r2, r4);
    r2 = r2.matches(r0);
    if (r2 != 0) goto L_0x0115;
L_0x0114:
    return r1;
L_0x0115:
    return r3;
L_0x0116:
    return r1;
```

Cet extrait de code vérifie que le flag alterne majuscules et minuscules.

[m] Télécharger l'outil « dex2jar » disponible au lien suivant :

- https://sourceforge.net/projects/dex2jar/files/latest/download

Visualiser la class « FlagChecker.class » en utilisant « jadx-gui », après avoir récupéré les « .class » avec « dex2jar ». Qu'observez-vous ? (détails de la manipulation) **(2 pts)**

On utilise les commandes suivantes :

cd dex2jar-2.0

*chmod +x *.sh*

cp ../../classes.dex

./d2j-dex2jar.sh classes.dex

jadx-gui classes-dex2jar.jar

On remarque que la méthode *checkFlag()* a été décompilée correctement.

Question 3. Analyse Dynamique (13 pts)

[a] Installer l'application sur l'émulateur (détails de la manipulation) **(2 pt)**

[b] Décrire une méthode d'extraction d'un APK provenant du PlayStore Android (détailler la manipulation avec l'APK de babyrev) **(2 pt)**

[c] En se basant sur les techniques d'injection de code vu en classe, compléter le fichier « hook.js » suivant (détailler la manipulation pour obtenir le résultat final) **(4 pt)**

```
    L_0x003c:
        r0 = new java.lang.StringBuilder;
        r0.<init>(r11);
        r0 = r0.reverse();
        r0 = r0.toString();
        r0 = r0.toLowerCase();
        r3 = 1;
        r0 = r0.substring(r3);
        r4 = 2131427368; // 0x7f0b0028 float:1.847635E38 double:1.053065039E-314;
        r4 = r10.getString(r4);
        r0 = r0.startsWith(r4);
        if (r0 != 0) goto L_0x0060;
    L_0x005f:
        return r1;
```

Le fichier « hook.js » suivant est disponible dans le répertoire « lab » de votre « home ». Pour réaliser cette manipulation nous utiliserons « Frida », l'outil d'instrumentation binaire dynamique, à lancer en tâche de fond sur l'émulateur. Le binaire « **`frida-server`** » est disponible au répertoire « **`/data/local/tmp`** » du file system virtuel de votre émulateur.

```
Java.perform(function () {
    console.log("[!] Hooking Babyrev chall");

  var Activity = Java.use("com.mobisec.babyrev.FlagChecker");
    Activity.checkFlag.overload('android.content.Context', 'java.lang.String').implementation = function (context, str) {

    console.log("[!] Hook checkFlag!");
    console.log("android.content.context: " + context.toString());


    ...

    };
});
```

[d] Illustrer la vérification du flag avec une capture d'écran de l'émulateur. **(5 pts)**

[e] **Bonus** Qu'auriez-vous fait pour contourner la méthode de vérification du « Flag » afin qu'un utilisateur puisse utiliser n'importe quelle valeur (détails de la manipulation à réaliser). **(4 pts)**