

La rétroingénierie appliquée à Android

La traque aux traqueurs

Maxime Catrice

2 février 2018



Qu'est ce que la rétroingénierie ?

Légalité et rétroingénierie

Les applications Android

L'analyse statique

Élévation de privilèges

L'analyse réseau

L'analyse dynamique

Comment s'en prémunir ?

Pourquoi ?



Qu'est ce que la rétroingénierie ?

• Qu'est ce que la rétroingénierie ?

• Légalité et rétroingénierie

• Les applications Android

• L'analyse statique

• Élévation de privilèges

• L'analyse réseau

• L'analyse dynamique

• Comment s'en prémunir ?

• Pourquoi ?

Principe :

Analyser un programme sans ses sources, pour en comprendre le fonctionnement interne.

Objectifs :

- ▶ Interopérabilité
- ▶ Documentation
- ▶ Veille compétitive
- ▶ Recherche de failles de sécurités
- ▶ Piratage



La rétro-ingénierie logicielle est le principe d'analyser un programme sans ses sources, pour en comprendre le fonctionnement interne. Deux approches complémentaires sont en général utilisées, l'analyse statique et l'analyse dynamique. Dans le premier cas il s'agit de reconstituer le code source du logiciel à partir d'un exécutable ou au moins de le traduire dans le langage assembleur. Pour le second cas, il s'agit d'étudier le programme directement pendant son exécution à l'aide d'un débogueur.

Interopérabilité d'un logiciel :

afin d'en comprendre le fonctionnement et ainsi le rendre compatible avec d'autres logiciels

documentation :

Retrouver le fonctionnement d'un logiciel avec laquelle on souhaiterait communiquer, mais dont la documentation n'est plus

disponible.

veille compétitive :

Étudier mes produits concurrents, les méthodes utilisées, estimer les coûts de développement d'une application similaire. Mais cela permet également de déceler d'éventuelles violations de brevet par un concurrent.

recherche de failles de sécurité :

C'est ainsi que certaines failles de sécurité sont trouvées dans les applications commerciales dont les sources ne sont pas disponibles. Les Virus sont eux aussi systématiquement étudiés par rétro ingénierie. D'ailleurs, ils sont très souvent très bien protégés afin de rendre leur identification plus lente.

Piratage :

Prolonger la période d'essai. 'amtlb.dll' par exemple (adobe).

Légalité et rétroingénierie

- Qu'est ce que la rétroingénierie ?
- **Légalité et rétroingénierie**
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

Logiciels et propriété intellectuelle

- ▶ Logiciel protégeable
- ▶ Fonctionnalité en tant que telle non protégeable

Article 122-6-1 du code de la propriété intellectuelle

- ▶ Acquisition légale du logiciel
- ▶ Soit :
 - ▶ La license ne l'interdit pas
 - ▶ Réalisation à des fins d'interopérabilité



« On n'a donc pas le droit en France de démontrer techniquement qu'un logiciel présente des failles de sécurité, ou que la publicité pour ces logiciels est mensongère. Dormez tranquilles, citoyens, tous vos logiciels sont parfaits. »

Guillermito



Si le logiciel peut être protégé par une licence, sa fonctionnalité en tant que telle n'est pas protégeable.

Par exemple, le programme Word inclus dans la suite bureautique :

- Office de Windows fait l'objet d'une protection juridique spécifique.
- Mais la fonctionnalité de Word, à savoir être un logiciel de traitement de texte, n'est pas protégeable par Microsoft.

Le logiciel ne doit pas avoir été piraté

Et :

- Soit la license le permet
- Soit cette rétroingénierie est effectuée dans un but d'interopérabilité,

En résumé, il est interdit de décompiler une application à des fins autres que de rétroingénierie, ce qui fait que pour analyser une application, il est illégal d'analyser le code source, et encore plus de le partager

Légalité et rétroingénierie : Les Bug BugBounty

- Qu'est ce que la rétroingénierie ?
- **Légalité et rétroingénierie**
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

Qu'est ce qu'un Bug Bounty ?

Un bug bounty est un programme proposé par de nombreux sites web et développeurs de logiciel qui permet à des personnes de recevoir reconnaissance et compensation après avoir reporté des bugs, surtout ceux concernant des exploits et des vulnérabilités



FIGURE – Tweet montrant une faille dans macOS

Bug Bounty : Un bug Bounty est une récompense qu'une société offre à tous ceux qui trouvent des failles de sécurité dans un périmètre donné.

Ce périmètre peut être un site web, une application, une API...etc., etc. C'est donc à l'entreprise de déterminer les services que les gens vont pouvoir explorer à la recherche de failles de sécurité.

Il y a bien sûr des règles à respecter et chaque Bug Bounty doit énoncer clairement les limites que le hacker ou l'expert ne doit pas franchir, mais en général, comme ça se passe sur des services en production, il vaut mieux éviter de tout casser si on veut sa récompense.

D'ailleurs, concernant le montant de la récompense, c'est assez variable d'une so-








ciété à l'autre et ça dépend surtout du type de faille remontée. Plus la faille est critique, complexe, bien documentée avec si possible un PoC (Proof of concept) et pourquoi pas des recommandations, voire un patch, plus la récompense sera grande. Évidemment, si vous trouvez des failles qui ont déjà été trouvées par un autre, vous ne recevrez aucune récompense.

Cela permet également d'éviter, pour les entreprises ce genre de choses.

Étant donné qu'Apple ne fournit pas de bug bounty pour macOS, un chercheur a révélé publiquement une faille critique sur macOS. Cependant, il précise qu'il a pris cette décision car cette vulnérabilité n'est pas accessible à distance.

Les applications Android : Composition

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- **Les applications Android**
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

- ▶  `AndroidManifest.xml`
- ▶  `assets`
- ▶  `classes.dex`
- ▶  `lib/`
- ▶  `META-INF/`
- ▶  `res/`
- ▶  `resources.arsc`

Permissions, Activités...

Ressources non compilées, non standards

Code binaire de l'application

Librairies externes

Informations autour de l'application

Ressources standards, non compilées

Ressources compilées

AndroidManifest

- Fichier xml qui contient :
- Permissions
- Liste des activités
- Services

Assets

Autres ressources (polices...)

Classes.dex

Code binaire de l'application, limité à 65 000 méthodes par fichier, mais il faut éviter de le dépasser, parce que le support au-delà n'est

pas assuré

META-INF

Informations autour de l'application :

- La classe à lancer
- Version du package
- Numéro de version

res

Ressources non compilées, mais standards

ex : Layout, Drawables,

resources.arsc

Ressources compilées, en format binaires xml

Les applications Android : Compilation

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

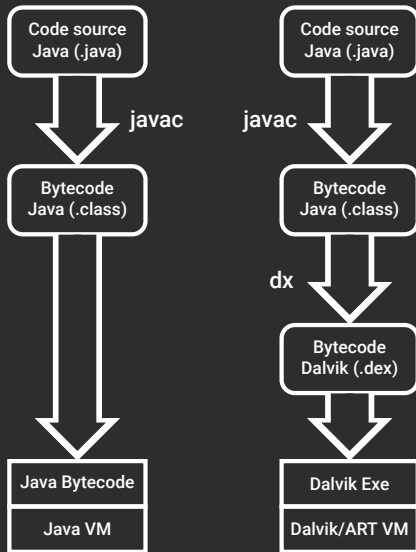


FIGURE – Compilation Java & Android

ART	vs	Dalvik
JIT		AOT
≤ 4.4		≥ 4.4
≥ 7.0 : AOT & JIT		

Le code java est compilé en byte-code Java, puis rassemblé, compacté et optimisé via le programme dx en un exécutable dalvik, byte-code spécifique, prêt à être exploité par la machine virtuelle dalvik ou ART.

Ce fichier n'est pas compressé, ce qui permet à la machine virtuelle de mapper le code en mémoire et de le partager très facilement

JIT :

- Just In time :

L'application est compilée en langage machine uniquement au moment où le code est nécessaire. Occupe moins d'espace mémoire

Plus lents

Disponible pour les versions inférieures à Kitkat

AOT :

- Ahead of time :

L'application est compilée en langage machine au moment de son installation. Bien plus rapide

Occupe plus d'espace mémoire

Disponible pour les versions supérieures à Kitkat

A partir de nougat, ART a été modifié pour utiliser à la fois un comportement JIT et AOT, permettant d'allier le meilleur des deux méthodes

L'analyse statique

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

Qu'est ce que l'analyse statique ?

Examen d'un programme permettant d'obtenir des informations par rapport à son comportement sans l'exécuter.

Méthode d'analyse

- ▶ Analyse du code source
- ▶ Analyse par signature

Outils :

- ▶ jadx
- ▶ Android Studio
- ▶ exodus-standalone

Objectifs :

- ▶ Permissions de l'application
- ▶ Trackers inclus
- ▶ Portions de codes utilisables pour l'analyse dynamique

L'analyse statique :

L'analyse statique de code correspond à une analyse du logiciel réalisée sans exécuter le programme. Le but de l'analyse statique est de trouver les défauts présents dans le logiciel. Elle permet également de s'assurer que le code est écrit selon des règles de programmations définies,

Méthodes :

- Analyse du code source : peut être illégale
- Analyse par signature : pas de décompilation, ce qui rend l'opération légale (mais pas forcément approuvée par les créateurs des applications que l'on analyse)

Objectifs :

- Permissions : déterminer si l'application nécessite des permissions qui semblent incohérentes
- Lister les trackers qui y sont inclus
- Déterminer si des portions de code peuvent être intéressantes pour l'analyse dynamique

Outils :

- Jadx : décompilateur dex vers Java. Il permet de ne pas avoir à manipuler les fichiers smali (qui correspondent à peu près à de l'assembleur pour android), qui sont peu lisibles
- Exodus-standalone : Analyse par signature pour déterminer les permissions ainsi que les trackers utilisés par une application

L'analyse statique : Exemple

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

```
1 private void sendPhoto(byte[] data) {
2     try {
3         Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0, data.length);
4         ByteArrayOutputStream bos = new ByteArrayOutputStream();
5         bitmap.compress(CompressFormat.JPEG, 20, bos);
6         JSONObject object = new JSONObject();
7         object.put("image", true);
8         object.put("buffer", bos.toByteArray());
9         IO Socket.getInstance().getIoSocket().emit("x0000ca", object);
10    } catch (JSONException e) {
11        e.printStackTrace();
12    }
13 }
```

FIGURE – Méthode permettant la prise et l'envoi d'une photo

```
1 public static boolean sendSMS(String phoneNo, String msg) {
2     try {
3         SmsManager.getDefault().sendTextMessage(phoneNo, null, msg, null, null);
4         return true;
5     } catch (Exception ex) {
6         ex.printStackTrace();
7         return false;
8     }
9 }
```

FIGURE – Méthode permettant l'envoi d'un SMS

à un serveur distant

sendPhoto : Méthode qui envoie des photos

sendSMS : Méthode qui envoie un SMS

Élévation de privilèges

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- **Élévation de privilèges**
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

Qu'est ce qu'une élévation de privilège ?

Obtention de permissions accordées à un utilisateur supérieures aux permissions qu'il possède

Intérêt :

- ▶ Android est un système qui restreint l'utilisateur
- ▶ Accéder aux fonctionnalités bloquées
- ▶ Modifier en profondeur le fonctionnement des applications



L'élévation de privilège résulte de l'octroi à un intrus d'autorisations supérieures à celles initialement accordées. Par exemple, un intrus

avec un jeu de privilèges contenant des autorisations « en lecture seule » élève d'une façon ou d'une autre le jeu pour inclure des autorisations « en lecture et en écriture ».

Élévation de privilèges : Root

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- **Élévation de privilèges**
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

Qu'est ce que le root ?

Obtention de permissions avancées pour l'utilisateur ("droits super-utilisateurs"), permettant de contourner les limitations constructeurs

Principe du root : /system

1. Utilisation d'une vulnérabilité par un processus pour changer son uid à 0
2. Remontage de la partition /system en écriture
3. Copie des binaires su, busybox
4. Remontage de /system en lecture seule



Root :

Utilisation de privilèges avancés, permettant de limiter des limitations imposées par le système

Par exemple, cela permet de supprimer les applications systèmes, qui ne sont pas désinstallables en tant que simple utilisateur.

Principe du root :

Utilisation d'une faille d'android, ou alors du mode récupération d'android pour obtenir temporairement un uid à 0, c'est à dire root

Remontage de la partition système en écriture, afin de pouvoir la modifier

Copie de nouveaux binaires, tels que su, busybox

Remontage de la partition système en lecture seule

Exemples d'utilisation :

- Accéder aux partitions systèmes
- Installation de busybox
- Sauvegarder une application en conservant l'état de l'application au moment de la sauvegarde
- Modifier des propriétés systèmes (densité d'écran, adresse mac...)

Élévation de privilèges : Root

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- **Élévation de privilèges**
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

Qu'est ce que le root ?

Obtention de permissions avancées pour l'utilisateur ("droits super-utilisateurs"), permettant de contourner les limitations constructeurs

Exemples d'utilisation

- ▶ Accéder aux partitions systèmes
- ▶ Ajouter un binaire BusyBox
- ▶ Sauvegarder l'état actuel d'une application
- ▶ Modifier les propriétés systèmes



Root :

Utilisation de privilèges avancés, permettant de limiter des limitations imposées par le système

Par exemple, cela permet de supprimer les applications systèmes, qui ne sont pas désinstallables en tant que simple utilisateur.

Principe du root :

Utilisation d'une faille d'Android, ou alors du mode récupération d'Android pour obtenir temporairement un uid à 0, c'est à dire root

Remontage de la partition système en écriture, afin de pouvoir la modifier

Copie de nouveaux binaires, tels que su, busybox

Remontage de la partition système en lecture seule

Exemples d'utilisation :

- Accéder aux partitions systèmes
- Installation de busybox
- Sauvegarder une application en conservant l'état de l'application au moment de la sauvegarde
- Modifier des propriétés systèmes (densité d'écran, adresse mac...)

Élévation de privilèges : Xposed

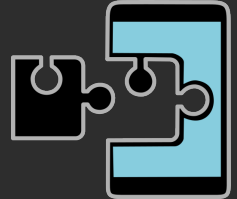
- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- **Élévation de privilèges**
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

Qu'est ce que le module Xposed ?

Framework permettant d'intercepter toutes méthodes d'une application, pour injecter du code supplémentaire

Exemple d'utilisation

- ▶ Lire les preferences
- ▶ Désactiver la vérification des certificats SSL
- ▶ Modifier son IMEI
- ▶ Modifier sa position GPS



Xposed :

Framework permettant d'intercepter toutes méthodes d'une application, pour injecter du code supplémentaire

Exemple d'utilisation

Ne fonctionne qu'avec les applications java, mais pas avec les bibliothèques natives, par exemple

Exemples d'utilisation de Xposed :

- Lire les paramètres des applications
- Désactiver la vérification SSL, pour par exemple, pouvoir déchiffrer le trafic
- Modifier son IMEI
- Simuler sa position GPS

Élévation de privilèges : Xposed

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- **Élévation de privilèges**
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

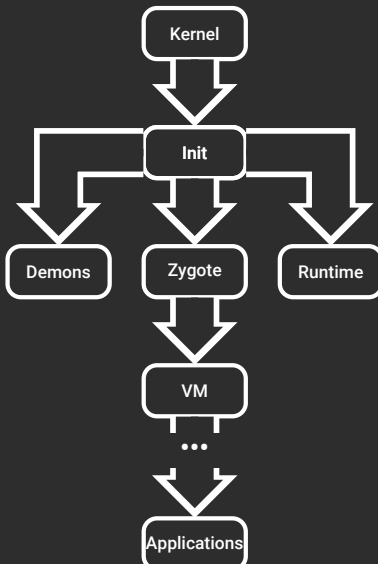


FIGURE – Initialisation d'Android

Démarrage d'Android

1. Le kernel lance le processus init
2. Init lance des demons, runtime
3. Init lance Zygote

Le processus Zygote :

1. Initialise une instance de la VM
2. Pré-charge des classes
3. Fork pour chaque application
4. Partage une partie de sa mémoire avec ses fils

Le démarrage d'Android :

- On s'intéresse au processus une fois le lancement du kernel
- Le kernel initialise le processus Init
- Init lance à son tour des démons (usb, adb, ril), et le runtime
- Init lance aussi Zygote

Zygote :

Zygote est un processus primordial pour Android :

- Il initialise la machine virtuelle
- pré-charge des classes communes aux applications

- Se fork pour chaque nouvelle application lancée
- Partage une partie de sa mémoire avec les applications forkées

Fonctionnement de Xposed :

- Le processus init est modifié pour changer le comportement de Zygote en ajoutant des bibliothèques au classpath
- Ajout de bibliothèques à Zygote permettant de détecter le lancement d'application
- A chaque lancement d'une application, Zygote va remplacer le code de l'application pour injecter du code externe

Élévation de privilèges : Xposed

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- **Élévation de privilèges**
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

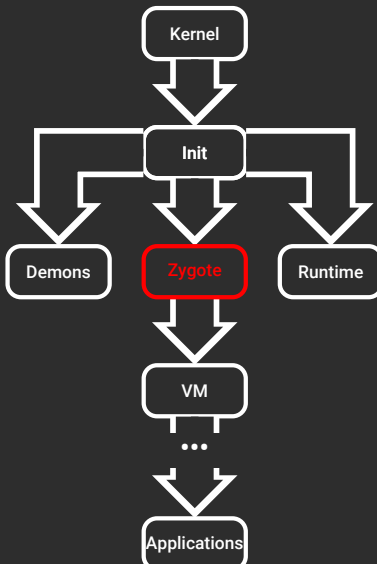


FIGURE – Initialisation d'Android

Fonctionnement

1. Modification du processus init pour ajouter des bibliothèques au classpath
2. Ajout de bibliothèques à Zygote pour détecter le lancement d'applications
3. A chaque nouvelle application forké de Zygote, il est possible de modifier le code exécuté par la VM

Le démarrage d'Android :

- On s'intéresse au processus une fois le lancement du kernel
- Le kernel initialise le processus Init
- Init lance à son tour des démons (usb, adb, ril), et le runtime
- Init lance aussi Zygote

Zygote :

Zygote est un processus primordial pour Android :

- Il initialise la machine virtuelle
- pré-charge des classes communes aux applications

- Se fork pour chaque nouvelle application lancée
- Partage une partie de sa mémoire avec les applications forkées

Fonctionnement de Xposed :

- Le processus init est modifié pour changer le comportement de Zygote en ajoutant des bibliothèques au classpath
- Ajout de bibliothèques à Zygote permettant de détecter le lancement d'application
- A chaque lancement d'une application, Zygote va remplacer le code de l'application pour injecter du code externe

L'analyse réseau

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

Qu'est ce que l'analyse réseau ?

Intercepter le trafic entrant et sortant de l'application, pour déterminer qui sont les destinataires et comment sont échangés les messages

Objectifs :

- ▶ Déterminer les échanges effectués par l'application
- ▶ Lire le trafic http
- ▶ Déchiffrer le trafic https

Environnement utilisé

- ▶ Emulateur genymotion avec ProxyDroid
- ▶ WireShark (Analyseur de paquet)
- ▶ Xposed : JustTrustMe



Analyse réseau :

Intercepter le trafic entrant et sortant de l'ap-

plication, pour déterminer qui sont les destinataires et comment sont échangés les messages

L'analyse réseau : Principe

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- **L'analyse réseau**
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?



FIGURE – Principe d'une attaque man-in-the-middle

L'analyse réseau : Principe

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- **L'analyse réseau**
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

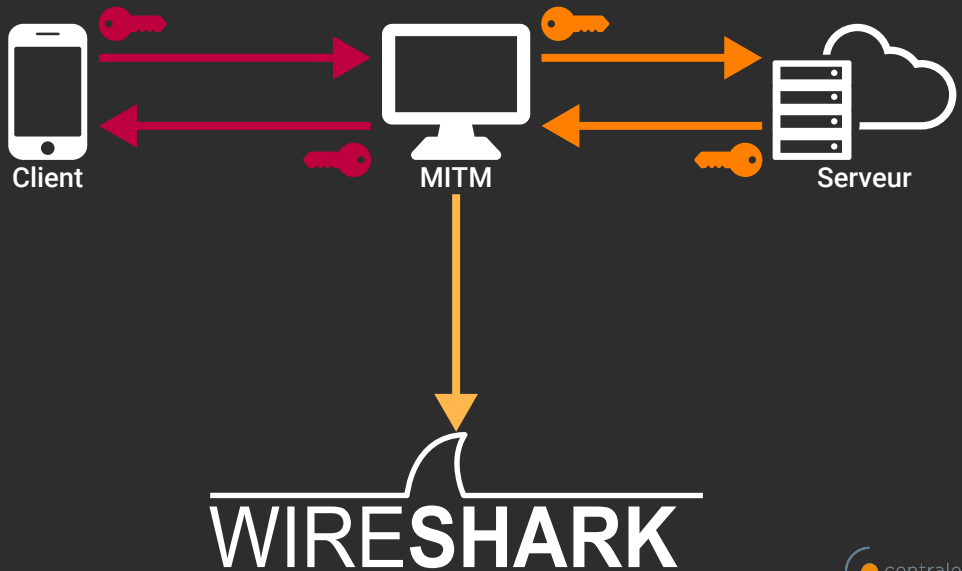


FIGURE – Principe d'une attaque man-in-the-middle

L'analyse dynamique

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

Intérêt :

- ▶ Obtenir des informations générées dynamiquement par l'application
- ▶ Difficulté de déchiffre des strings lourdement obfusqués
- ▶ Requêtes qui ne peuvent pas être interprétées par un MITM

Environnement utilisé :

- ▶ Émulateur : Genymotion
- ▶ Root, Xposed
- ▶ Inspeckage
- ▶ Android Device Monitor

Utilisation :

- ▶ Utilisation d'un débbugger
- ▶ Analyse de la mémoire utilisée par l'application

- Les informations générées dynamiquement ne peuvent être récupérée par une analyse statique
- Certaines méthodes d'obfuscation sont difficiles à déchiffrer. Cependant, étant donné qu'au sein de l'application, ces valeurs vont être déchiffrées, on peut essayer de la récupérer à ce moment
- Certaines requêtes ne peuvent être déciffrée par un MITM, on peut alors essayer de lire les données de la requête au moment de l'envoi ou de la réception de la requête

Émulateur : plus de facilité pour le rooter ainsi qu'installer Xposed

Root, Xposed :

Inspeckage : Démarrer des activités non déclarées, Désactiver le SSL, remplacer des paramètres d'application... Android Device Monitor : Outil intégré à Android Studio offrant des fonctions de debug et d'analyse d'application

Débugger : Permet de faire mettre des breakpoints. Cependant, étant donné qu'on a pas accès au code source, il est nécessaire de décompiler l'application en smali, reconstruire le projet et recompiler l'application

Mémoire : Permet de récupérer certaines valeurs

L'analyse dynamique : Debugger

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- **L'analyse dynamique**
- Comment s'en prémunir ?
- Pourquoi ?

Principe

1. Décompilation de l'application
 2. Import du projet dans Android Studio
 3. Mise en place des points d'arrêts
 4. Lancement du mode debug
 5. Analyse de l'état de l'application aux points d'arrêts
- Il est par la suite possible de recompiler l'application avec les modifications apportés au smali



Principe : On décompile l'APK, mais on reste au stade du smali

- On importe les fichiers dans Android Studio pour y générer un nouveau projet
- On

place les points d'arrêts

- On lance l'application
- On analyse l'état de la mémoire de l'application aux points d'arrêts
- Enfin, si on souhaite produire une version modifiée de l'application, il est possible de recompiler le smali pour produire un nouveau APK

Comment s'en prémunir ? : La sécurité par l'obscurité

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

Obscurcir son code

- Obfuscation de code :

- ▶ Ajout d'instructions inutiles
- ▶ Ajout d'arguments inutiles sur les méthodes
- ▶ Minimisation du code
- ▶ Génération dynamique de string

► Chiffrement du programme

- ▶ **Exécution de code distant**



FIGURE – XKCD 257

Obfuscation de code :

- **Instructions inutiles/Arguments inutiles :** Allourdi le programme
- **Minification :** pratique répandue, permettant de réduire la taille du code en renommant les variables et classes par les lettres, ralentit la lecture du code à la décompilation
- **Ralentit l'application, car ces opérations**

sont lourdes

Chiffrement du programme :

- Ralentir le lancement du programme, le programme doit forcément être déchiffré pour être exécuté, on peut alors essayer de l'analyser

Execution du code distant :

- Les applications web, nécessite une connexion internet

Comment s'en prémunir ? : L'absurdité de l'obscurité

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?

Principe de Kerckhoffs

"Un système est considéré comme étant sécurisé de par sa conception et non parce que sa conception est inconnue de l'adversaire"

Les limites de l'obfuscation

- ▶ Débogage difficile
- ▶ Protection temporaire
- ▶ Potentiel perte de performances
- ▶ Qualité du code en baisse
- ▶ Appel à des librairies externes non obfusables



Limites :

- Les opérations de débogage et la réalisation de traces ne pourront plus être proposées sur la version commerciale de l'application. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
- La sécurité par l'opacité est un mythe ! L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps assez court. Mais elle ne protège pas des pirates voulant exploiter les failles de sécurité de l'application.

- L'augmentation de la complexité algorithmique et la modification des structures de données augmentent le temps d'exécution. Les patrons de conceptions choisis par le développeur disparaissent et peuvent être remplacés par d'autres moins efficaces.
- La qualité (algorithmique) du code source baisse considérablement et peut être un frein à la certification par des organismes tiers.
- L'appel à des API externes (notamment en Java) fait par le nom ne peut PAS être obfusqué, et donnent alors des indices aux pirates.

Pourquoi ? : La traque aux utilisateurs

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?



“Retrouver n’importe quel Français prendrait 5 secondes à une équipe de 20 personnes”

“Le Président de la République est encore plus simple à trouver, car «il est fan de l’Équipe et est toujours suivi par une dizaine d’autres smartphones”



Criteo :

Reciblage publicitaire personnalisé sur internet

Criteo Commerce Marketing Ecosystem, c’est un environnement ouvert où retailers, marques et éditeurs collaborent en toute confiance en vue d’un même objectif : générer des ventes et des profits. Fort d’une technologie de machine learning optimisée pour le commerce et de quantités massives de données consommateur, notre écosystème offre des performances hors normes, à très grande échelle.

Fidzup :

Ils connectent l’analyse des comportements des consommateurs en magasin avec la publicité digitale sur mobile, afin de faire venir ou revenir des visiteurs qualifiés dans les points de vente de leurs clients.

Teemo :

Teemo est la plateforme Drive-to-Store qui révolutionne le marketing des magasins physiques.

Notre technologie unique génère du trafic dans les magasins de nos clients en combinant données de géolocalisation et algorithmes. Nous mesurons avec précision l’impact de notre solution pour optimiser la performance.

Pourquoi ? : La traque aux traqueurs

- Qu'est ce que la rétroingénierie ?
- Légalité et rétroingénierie
- Les applications Android
- L'analyse statique
- Élévation de privilèges
- L'analyse réseau
- L'analyse dynamique
- Comment s'en prémunir ?
- Pourquoi ?



Comment savoir qui nous traque ?

Exodus Pricacy Association Française

Yale Privacy Lab Laboratoire de recherches mêlant vie privée, sécurité et anonimat

Kimetrak Extension Chrome/Firefox pour détecter les traqueurs



Exodus Privacy :

Études eXpérimentales d'Ordiphones Débiles et Universellement Sales

Suite à la publication en août 2017 d'articles concernant les pratiques de Teemo, un groupe d'Hacktiviste s'est formé pour monter une plateforme qui recense les traqueurs utilisés par les applications les plus populaires du play Store

Yale Privacy Lab :

Laboratoire de recherche à propos de la sécurité, la vie privée et l'anonymat en ligne

Kimetrak :

L'objectif est de vous permettre de détecter simplement les services qui vous pistent en ligne à travers les sites que vous visitez, et de distinguer les bons et les mauvais élèves en la matière.

Merci!

Souriez, vous êtes tracés!