

La rétroingénierie appliquée à Android

La traque aux traqueurs

Maxime Catrice

21 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 sécurité
- ▶ Piratage



Rétro-ingénierie :

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

- Deux approches complémentaires :

- Analyse statique :

Reconstituer le code source du logiciel à partir d'un exécutable ou au moins de le traduire dans le langage assembleur.

- Analyse dynamique.

Étudier le programme directement pendant son exécution à l'aide d'un débogueur.

Objectifs :

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 les produits concurrents, les méthodes utilisées, déceler d'éventuelles violations de brevet par un concurrent.

Recherche de failles de sécurité :

Failles de sécurité dans les applications commerciales dont les sources ne sont pas disponibles. Les Virus sont eux aussi systématiquement étudiés par rétro ingénierie.

Piratage :

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

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. »

Guillermi

Propriété intellectuelle :

- Le logiciel peut être protégé par une licence
- Sa fonctionnalité en tant que telle n'est pas protégeable.

Par exemple, le programme Word :

- Fait l'objet d'une protection juridique spécifique.
- Mais la fonctionnalité de Word : un logiciel de traitement de texte, n'est pas protégeable par Microsoft.

Droit de la rétroingénierie

Le logiciel ne doit pas avoir été piraté

Et :

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

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

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	Permissions, Activités...
▶ 	assets	Ressources non compilées, non standards
	classes.dex	Code binaire de l'application
▶ 	lib/	Librairies externes
▶ 	META-INF/	Informations autour de l'application
▶ 	res/	Ressources standards, non compilées
	resources.arsc	Ressources compilées

Composition :

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

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 ?

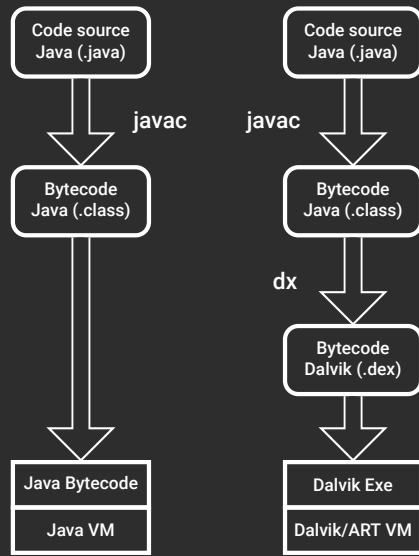


FIGURE – Compilation Java & Android

Compilation :

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

Fichier non compressé : la machine virtuelle peut mapper rapidement le code en mémoire et de le partager très facilement

JIT :

- Just In time :
L'application est compilé en langage machine uniquement au moment où le code est

nécessaire

- Occupe moins d'espace mémoire
- Plus lent
- Disponible pour les versions inférieures à Kitkat

AOT :

- Ahead of time :
L'application est compilé 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

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

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.

Objectifs :

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

Méthode d'analyse

- ▶ Analyse du code source
- ▶ Analyse par signature



L'analyse statique :

Analyse du logiciel réalisée sans exécuter le programme.

But : trouver les défauts présents dans le logiciel.

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 un peu plus 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 : décompileur dex vers Java : ne pas avoir à manipuler les fichiers smali (=assembleur pour android), qui sont peu lisibles
- Android Studio : Formattage du code, coloration syntaxique, navigation au sein du code...
- Exodus-standalone : Analyse par signature pour déterminer les permissions ainsi que les trackers utilisés par une application
- StaCoAn : Outil basé sur jadx qui analyse l'application pour retrouver des identifiants/mot de passes en dur, les url d'api, clés de chiffrements...

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.

Outils :

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



L'analyse statique :

Analyse du logiciel réalisée sans exécuter le programme.

But : trouver les défauts présents dans le logiciel.

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 un peu plus 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 : décompileur dex vers Java : ne pas avoir à manipuler les fichiers smali (=assembleur pour android), qui sont peu lisibles
- Android Studio : Formattage du code, coloration syntaxique, navigation au sein du code...
- Exodus-standalone : Analyse par signature pour déterminer les permissions ainsi que les trackers utilisés par une application
- StaCoAn : Outil basé sur jadx qui analyse l'application pour retrouver des identifiants/mot de passes en dur, les url d'api, clés de chiffrements...

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         IOSocket.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



Elévation de privilège :

Octroyer à un individu des autorisations supérieures à celles initialement accordées.

Par exemple, passer de droits "en lecture seule" à des droits "en lecture et en écriture" d'une façon ou d'une autre.<

Intérêts :

- Android restreint les droits utilisateurs (on ne peut pas désinstaller certaines applications)
- On va chercher à utiliser davantage de possibilités qui ne sont pas accessibles par défaut
- Un des intérêts peut par exemple être la modification en profondeur du fonctionnement des applications

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 seuls

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 ?

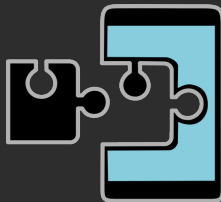
• Pourquoi ?

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/TLS
- ▶ 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, 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?

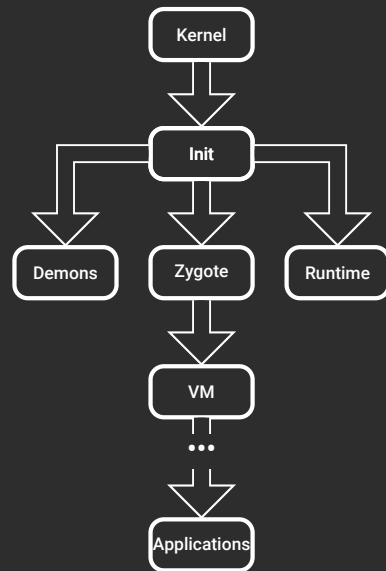


FIGURE – Initialisation d'Android

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 ?

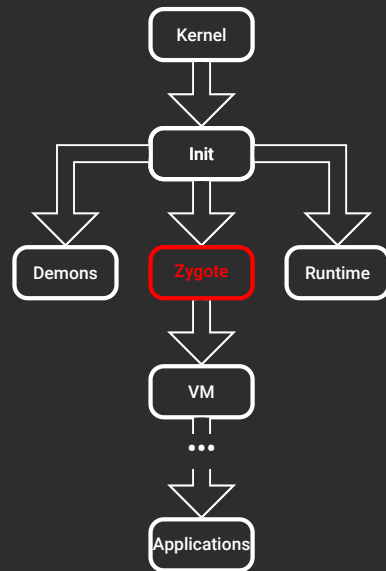


FIGURE – Initialisation d'Android

Fonctionnement

1. Modification du processus init pour ajouter des librairies au classpath
2. Ajout de librairies 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 librairies au classpath
- Ajout de librairies 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



les échanges

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 serveurs avec qui l'application échange
- Trafic http : intercepter et lire les échanges
- Trafic https : intercepter, déchiffrer et lire

Analyse réseau :

- Emulateur avec ProxyDroid : modifie les règles iptables pour s'assurer que l'application passe forcément par le proxy
- Wireshark pour analyser les paquets interceptés
- JustTrustMe pour désactiver la vérification des certificats SSL/TLS

L'analyse réseau : Principe



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

Client-serveur :

- Les échanges sont chiffrés
- Un certificat permet de s'assurer que le serveur est bien celui que l'on attend

MITM :

- Déchiffrer les échanges réalisés
- JustTrustMe pour empêcher la vérification des certificats
- Récupération des données pour être analysées

• 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 ?

L'analyse réseau : Principe

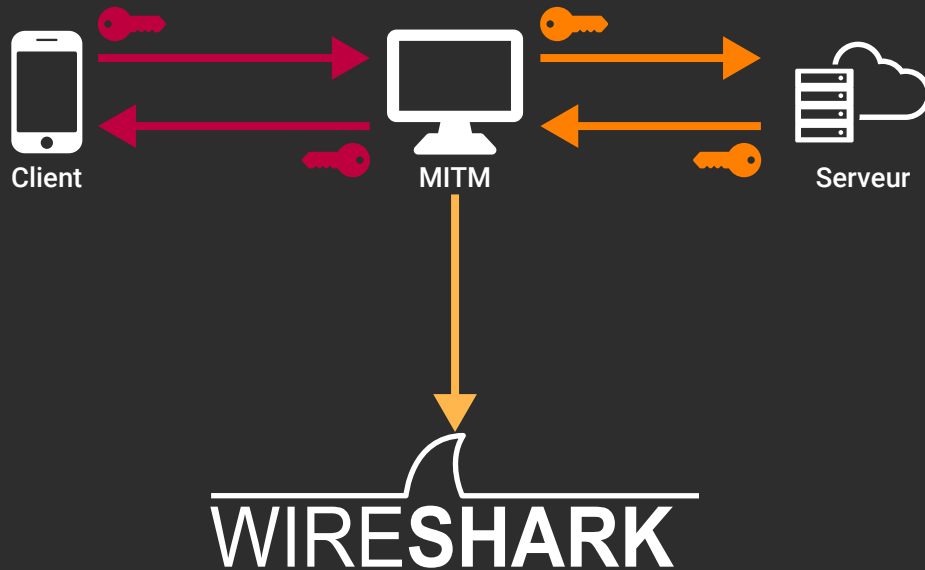


FIGURE – Principe d'une attaque man-in-the-middle

Client-serveur :

- Les échanges sont chiffrés
- Un certificat permet de s'assurer que le serveur est bien celui que l'on attend

MITM :

- Déchiffrer les échanges réalisés
- JustTrustMe pour empêcher la vérification des certificats
- Récupération des données pour être analysées

• 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 ?

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 ?

Qu'est ce que l'analyse dynamique ?

Analyse d'un programme en l'exécutant, dans un environnement dédié permettant d'observer son comportement et son fonctionnement en situation réelle

Intérêt :

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



Intérêt :

- Informations générées dynamiquement ne peuvent être récupérées par une analyse statique
- Certaines méthodes d'obfuscation sont difficiles à déchiffrer. Cependant, étant donné que ces valeurs vont être déchiffrées au cours de l'exécution, on peut essayer de les récupérer à ce moment
- Certaines requêtes ne peuvent être déchiffrées 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

Environnement :

Emulateur : plus de facilité pour le router

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 débog et d'analyse d'application

Utilisation :

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

• 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 dynamique ?

Analyse d'un programme en l'exécutant, dans un environnement dédié permettant d'observer son comportement et son fonctionnement en situation réelle

Outils :

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



Exemples d'utilisation :

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

Intérêt :

- Informations générées dynamiquement ne peuvent être récupérée par une analyse statique
- Certaines méthodes d'obfuscation sont difficiles à déchiffrer. Cependant, étant donné que ces valeurs vont être déchiffrées au cours de l'exécution, on peut essayer de les récupérer à ce moment
- Certaines requêtes ne peuvent être déchiffrées 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

Environnement :

Emulateur : 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 débog et d'analyse d'application

Utilisation :

Débogueur : 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

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, mais nécessite une connexion internet

Cacher de code :_Obfuscation de code :

- Instructions inutiles/Arguments inutiles :
ALourdi le programme
- Minification : pratique répandue, permettant de réduire la taille du code en renommant les variables et classes par une lettre, ralentie la lecture du code à la décompilation
- Ralenti l'application, car ces opérations

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 bibliothèques externes non obfusquables



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

Limites :

- Débogage et récupération de log plus difficile. Privant alors le support d'outils importants pour l'aide aux utilisateurs.
- L'obfuscation protège le code source contre la piraterie intellectuelle durant un temps as-

sez 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.
- 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 fait par le nom ne peut pas être obfusqué.

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

Environnement ouvert où retailers, marques et éditeurs collaborent en vue d’un même objectif : générer des ventes et des profits.

Fidzup :

Ils connectent l’analyse des comportements des consommateurs en magasin avec la publicité digitale sur mobile, afin de faire ve-

nir 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.

Génère du trafic dans les magasins des 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 ?
- Légacité et rétroingénierie
- Les aplications 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 anonymat

Kimetrak Extension Chrome/Firefox pour détecter les traqueurs



Exodus Privacy :

Études eXpérimentales d'Ordiphones Dé-biles 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!

<https://hazegard.github.io/CLOCK/>