

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

L'analyse dynamique

• L'analyse dynamique

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

L'analyse dynamique

• L'analyse dynamique

Intérêt :

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

L'analyse dynamique

• L'analyse dynamique

Intérêt :

- Obtenir des informations générées dynamiquement 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 la 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

L'analyse dynamique

• L'analyse dynamique

Intérêt :

- ▶ Obtenir des informations générées dynamiquement par l'application
- ▶ Difficulté de déchiffrer des strings lourdement obfusqués

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 la 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

L'analyse dynamique

• L'analyse dynamique

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 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 la récupérer à ce moment
- Certaines requêtes ne peuvent être déchiffrée 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

L'analyse dynamique

• L'analyse
dynamique

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

Outils :



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 la récupérer à ce moment
- Certaines requêtes ne peuvent être déchiffrée 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

L'analyse dynamique

• L'analyse dynamique

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

Outils :

- ▶ Émulateur : Genymotion

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 la récupérer à ce moment
- Certaines requêtes ne peuvent être déchiffrée 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 debug et d'analyse d'application

L'analyse dynamique

• L'analyse dynamique

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

Outils :

- ▶ Émulateur : Genymotion
- ▶ Root, Xposed

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 la récupérer à ce moment
- Certaines requêtes ne peuvent être déchiffrée 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 debug et d'analyse d'application

L'analyse dynamique

• L'analyse dynamique

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

Outils :

- ▶ Émulateur : Genymotion
- ▶ Root, Xposed
- ▶ Inspeckage

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 la récupérer à ce moment
- Certaines requêtes ne peuvent être déchiffrée 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

L'analyse dynamique

• L'analyse dynamique

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

Outils :

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

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 la 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 debug et d'analyse d'application

Utilisation :

Debugger : 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, reconstituer le projet et recompiler l'application

Mémoire : Permet de récupérer certaines valeurs

L'analyse dynamique

• L'analyse dynamique

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

Outils :

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

Utilisation :

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 la 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 debug et d'analyse d'application

Utilisation :

Debugger : 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, reconstituer le projet et recompiler l'application

Mémoire : Permet de récupérer certaines valeurs

L'analyse dynamique

• L'analyse dynamique

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

Outils :

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

Utilisation :

- ▶ Utilisation d'un débogueur

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 la 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 débogage et d'analyse d'application

Utilisation :

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

Mémoire : Permet de récupérer certaines valeurs

L'analyse dynamique

• L'analyse dynamique

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

Outils :

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

Utilisation :

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

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

L'analyse dynamique : Debugger

- L'analyse dynamique



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

L'analyse dynamique : Debugger

• L'analyse dynamique

Principe



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

L'analyse dynamique : Debugger

• L'analyse dynamique

Principe

1. Décompilation de l'application



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

L'analyse dynamique : Debugger

• L'analyse dynamique

Principe

1. Décompilation de l'application
2. Import du projet dans Android Studio



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

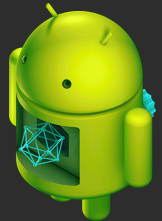
- On lance l'application
 - On analyse l'état de la mémoire de l'application aux points d'arrêts
- Enfin, si on souhaite produire une version modifiée de l'application, il est possible de recompiler le smali pour produire un nouveau APK

L'analyse dynamique : Debugger

• L'analyse dynamique

Principe

1. Décompilation de l'application
2. Import du projet dans Android Studio
3. Mise en place des points d'arrêts



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

L'analyse dynamique : Debugger

• L'analyse dynamique

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



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

L'analyse dynamique : Debugger

• L'analyse dynamique

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



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

L'analyse dynamique : Debugger

• L'analyse dynamique

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

