AOSP Partie 2

Comprendre les services Android

- Les services Android jouent un rôle crucial dans le fonctionnement du système d'exploitation. Ils forment la couche de base qui permet au système et aux applications de fonctionner en parfaite harmonie, offrant ainsi une expérience utilisateur fluide.

Les différents services Android

1. **Gestionnaire de packages (PackageManagerService)**
    - Le PackageManagerService est responsable de la gestion de toutes les applications installées sur l'appareil. Cela inclut la maintenance des métadonnées des applications, des permissions, des signatures et plus encore.
    - **Démonstration:**
        - **Lister les applications installées** : Utilisez la commande `ADB adb shell pm list packages` pour afficher la liste des packages installés.
        - **Obtenir des détails sur une application** : `adb shell dumpsys package <nom du package>` pour afficher des informations détaillées sur une application particulière.

Comprendre les services Android

2. **Gestionnaire de fenêtres (WindowManagerService)** :
    - Le WindowManagerService gère tout ce qui est relatif à l'affichage des fenêtres sur l'appareil. Cela inclut les animations, les superpositions, la taille et la position des fenêtres.
    - **Démonstration :**
        - **Inspecter les propriétés des fenêtres** : Utilisez la commande ADB `adb shell dumpsys window` pour afficher les informations sur l'état actuel des fenêtres.
3. **Gestionnaire d'activités (ActivityManagerService)** :
    - Le ActivityManagerService est chargé de la gestion du cycle de vie des activités, de la gestion de la pile d'activités, et de la supervision des processus en arrière-plan.
    - **Démonstration :**
        - Lister les activités en cours d'exécution : Utilisez la commande `adb shell dumpsys activity activities` pour afficher la pile d'activités en cours.
        - Afficher la consommation mémoire des processus : `adb shell dumpsys meminfo`.

Comprendre les services Android

4. **Gestionnaire de contenu (ContentManagerService)** :
    - Ce service est responsable de l'accès aux données entre différentes applications à travers les fournisseurs de contenu.
    - **Démonstration** :
        - Accéder aux contacts : Montrez comment une application peut utiliser le fournisseur de contenu pour récupérer une liste de contacts à l'aide de l'API ContentResolver.
5. **Gestionnaire de notifications (NotificationManagerService)** :
    - Comme son nom l'indique, ce service est chargé de la gestion des notifications. Il permet aux applications d'afficher, de mettre à jour et de supprimer leurs notifications.
    - **Démonstration** :
        - Créer une simple notification : Montrez comment coder une application simple qui envoie une notification avec un texte, une icône et une action cliquable.

Comprendre les services Android

6. **Gestionnaire de puissance (PowerManagerService)** :
    - Gère la consommation d'énergie du dispositif, y compris la mise en veille, le réveil et l'acquisition de différents types de verrous de veille.
    - **Démonstration** :
        - Observer les wakelocks : Utilisez `adb shell dumpsys power` pour montrer comment certaines applications peuvent empêcher l'appareil de se mettre en veille.
7. **Gestionnaire de connectivité (ConnectivityManagerService)** :
    - Gère toutes les connexions réseau de l'appareil, y compris Wi-Fi, données mobiles, VPN, etc.
    - **Démonstration** :
        - Vérifier l'état de la connectivité : Montrez comment une application peut vérifier si elle est connectée à Internet ou non à l'aide du ConnectivityManager.

Comprendre les services Android

Interaction des services Android avec les autres éléments d'AOSP

1. **Avec le Noyau Linux** :
    - Les services Android, en particulier ceux liés au matériel comme le PowerManagerService et le ConnectivityManagerService, communiquent directement avec le noyau pour gérer les ressources matérielles.
    - Par exemple, la mise en veille et le réveil du dispositif nécessitent une communication étroite avec le noyau.
2. **Avec les applications**:
    - Les applications interagissent constamment avec les services Android pour réaliser leurs tâches. Par exemple, une application de messagerie pourrait utiliser le NotificationManagerService pour afficher une nouvelle notification.
    - Les applications peuvent également démarrer leurs propres services en arrière-plan, qui fonctionnent indépendamment des activités d'interface utilisateur.

Comprendre les services Android

3. **Avec le HAL (Hardware Abstraction Layer)** :
    - Certains services Android, en particulier ceux qui nécessitent un accès direct au matériel, communiquent avec le matériel via le HAL.
      Le HAL fournit des interfaces standardisées pour le matériel, garantissant que les services Android peuvent fonctionner sur une variété d'appareils avec des composants matériels différents.
4. **Avec le Framework Android** :
    - Les services Android sont étroitement intégrés au framework Android. En fait, ils peuvent être considérés comme une partie intégrante du framework.
      Ils fournissent des APIs backend pour de nombreuses fonctions du framework. Par exemple, le framework peut offrir une API pour envoyer une notification, mais c'est le NotificationManagerService qui exécute réellement cette tâche.

Comprendre les applications système

- Une application système, parfois appelée "app système" ou "app intégrée", est une application qui est pré-installée sur le système d'exploitation Android et qui ne peut généralement pas être désinstallée par l'utilisateur final sans privilèges root. Ces applications sont intégrées dans la ROM du système et sont généralement stockées dans le répertoire /system/app ou /system/priv-app.

Comprendre les applications système

- **Fonctionnalités essentielles** : Les applications système fournissent des fonctionnalités essentielles qui permettent au téléphone de fonctionner de manière optimale dès la première utilisation. Par exemple, le dialer (téléphonie), les messages (SMS) et les paramètres.
- **Intégration étroite avec le système**: En raison de leur nature intégrée, les applications système peuvent avoir des autorisations et des privilèges supplémentaires qui ne sont pas disponibles pour les applications tierces. Cela leur permet d'offrir des fonctionnalités qui ne pourraient pas être fournies par des applications téléchargées depuis le Play Store.
- **Sécurité**: Les applications système sont installées dans une partition en lecture seule, ce qui les rend plus résistantes à la modification malveillante. De plus, certaines applications dans /system/priv-app ont des autorisations spéciales qui ne sont pas accessibles aux applications utilisateur ordinaires.

Comprendre les applications système

- **Performance**: Étant donné que ces applications sont pré-installées, elles peuvent être optimisées spécifiquement pour le hardware sur lequel elles tournent, garantissant ainsi une meilleure performance.
- **Uniformité et Cohérence**: Les applications système garantissent une expérience utilisateur uniforme et cohérente sur tous les appareils qui exécutent le système d'exploitation Android, indépendamment de la marque ou du modèle.
- **Interaction avec les couches inférieures de l'AOSP**: Les applications système, telles que NetworkManagement et ConnectivityService, interagissent directement avec les couches HAL (Hardware Abstraction Layer) et Kernel pour gérer et contrôler le matériel du dispositif.

Comprendre les applications système

- Chaque application système d'AOSP a son propre répertoire de code source, situé dans le dépôt AOSP
- **SystemUI**: platform/frameworks/base/packages/SystemUI
- **Paramètres**: platform/packages/apps/Settings
  Téléphonie (Dialer): platform/packages/apps/Dialer
- **Messagerie**: platform/packages/apps/Messaging
  Launcher (par défaut, Launcher3): platform/packages/apps/Launcher3
- **Package Manager**: Le code principal de cette fonctionnalité se trouve dans platform/frameworks/base/services/core/java/com/android/server/pm
- ***Input Methods (LatinIME)***: platform/packages/inputmethods/LatinIME
- **ConnectivityService**: Ce n'est pas une application en soi, mais un service. Son code est dans platform/frameworks/base/services/core/java/com/android/server

Comprendre les applications système

Comment interagir avec ces applications

1. **Intent**: Les Intents sont le principal mécanisme de communication entre les applications et les composants d'applications dans Android. Vous pouvez démarrer une application système ou un service spécifique en envoyant un Intent approprié.
2. **Binder**: Android utilise Binder pour la communication interprocessus (IPC). De nombreuses applications système exposent des services qui peuvent être appelés par d'autres applications via IPC.
3. **Content Providers**: Certaines applications système, comme Contacts, exposent des données via des Content Providers qui peuvent être consultées à l'aide d'URI spécifiques.
4. **APIs publiques**: Android SDK fournit des API pour interagir avec presque toutes les fonctionnalités du système. Par exemple, pour envoyer un SMS, vous pouvez utiliser la classe SmsManager.
5. **Modification du code source**: Comme AOSP est open-source, vous pouvez directement modifier le code source de n'importe quelle application système pour ajuster son comportement.

Comprendre les applications système

Exercice

Ajoutez une nouvelle préférence dans l'application Paramètres pour activer ou désactiver une fonctionnalité fictive appelée "Mode SuperPower".

Comprendre la gestion de la mémoire

- La gestion de la mémoire est un aspect essentiel d'Android, le système d'exploitation open source sur lequel AOSP (Android Open Source Project) est basé. Comprendre cette gestion est crucial pour optimiser les performances et la fluidité des applications et du système lui-même

1. **Mémoire physique**: Il s'agit de la mémoire RAM (Random Access Memory) qui est disponible sur le dispositif. Android gère cette mémoire de manière dynamique pour s'assurer que les ressources sont utilisées efficacement.
2. **Mémoire virtuelle**: Comme la plupart des systèmes d'exploitation modernes, Android utilise un système de mémoire virtuelle pour séparer la mémoire physique (RAM) de la mémoire que les applications et le système d'exploitation pensent utiliser. La mémoire virtuelle permet à chaque application de fonctionner comme si elle disposait de sa propre mémoire privée.
3. **Gestion de la mémoire pour les applications**: Dalvik/ART: Dans les premières versions d'Android, la machine virtuelle Dalvik était responsable de l'exécution du bytecode des applications Android. Depuis Android 5.0 (Lollipop), Dalvik a été remplacé par ART (Android Runtime), qui précompile le bytecode en code machine natif lors de l'installation de l'application. ART est conçu pour mieux gérer la mémoire et offrir de meilleures performances.

Comprendre la gestion de la mémoire

4. **Garbage Collector**: Android utilise le ramasse-miettes (Garbage Collector) pour libérer la mémoire qui n'est plus utilisée par les applications. Ceci est essentiel pour prévenir les fuites de mémoire et garantir que les ressources sont disponibles pour les nouvelles applications.
5. **Low Memory Killer**: Lorsque le système est à court de mémoire, Android utilise le "Low Memory Killer" pour fermer les processus qui ne sont pas essentiels. Cela permet de libérer de la mémoire pour les applications actuellement en cours d'exécution.
6. **Priorités des processus**: Android attribue des priorités à chaque processus en fonction de son état actuel (en avant-plan, en arrière-plan, service, etc.). Cela permet au système de décider quels processus tuer en premier lorsqu'il est à court de mémoire.

Comprendre la gestion de la mémoire

7. **ZRAM (ou swap compressé)**: Android utilise ZRAM, un espace de swap compressé dans la RAM, pour augmenter virtuellement la quantité de mémoire disponible. Cela aide à améliorer les performances sur les appareils avec une mémoire limitée.
8. **Cache**: Android utilise un système de mise en cache pour stocker des données fréquemment utilisées. Cela accélère l'accès aux données et améliore les performances générales.
9. **Mémoire partagée (Ashmem)**: Android utilise Ashmem (Android Shared Memory) pour permettre à différentes applications de partager des données sans avoir à les copier plusieurs fois en mémoire.

Comprendre la gestion de la mémoire

Exercice

Objectif: Analyser l'utilisation de la mémoire d'un appareil Android.

1. Connectez un appareil Android à votre ordinateur avec le débogage USB activé.
2. À l'aide d'adb, récupérez les informations de `/proc/meminfo`. Quelle est la quantité totale de RAM sur l'appareil?
3. Utilisez encore adb pour obtenir les informations de `/proc/vmallocinfo`. Pouvez-vous identifier une entrée liée à `ioremap` ?
4. En utilisant Android Studio, créez une application simple qui crée de nombreux objets en appuyant sur un bouton. Utilisez le Memory Profiler pour observer les effets sur la mémoire.
5. Expliquez la différence entre la mémoire physique et la mémoire virtuelle en vous basant sur vos observations.

Comprendre la sécurité

- La sécurité est l'un des piliers du système d'exploitation Android, et AOSP (Android Open Source Project) met en œuvre de nombreux mécanismes pour protéger les utilisateurs et les données

SELinux (Security Enhanced Linux)

- SELinux est un module de sécurité pour le noyau Linux qui offre un contrôle d'accès obligatoire. Android utilise SELinux pour renforcer le modèle de sécurité Android en confinant les processus à un domaine et en limitant les capacités à l'intérieur de ce domaine

Les principales entités de SELinux sont:

1. **Types** : chaque fichier, processus ou autre objet dans le système a un type SELinux associé.
2. **Domaines** : ce sont les types associés aux processus.
3. **Politiques** : ce sont des règles définissant les interactions entre différents types.

Modes SELinux:

1. **Enforcing** : Toutes les politiques sont appliquées et les violations sont bloquées et enregistrées.
2. **Permissive** : Les politiques ne sont pas appliquées, mais les violations sont enregistrées. C'est utile pour le débogage.

Comprendre la sécurité

Configurer une politique SELinux:

- Les politiques SELinux sont écrites dans un langage spécifique et compilées en un format binaire que le système utilise.
    - **Politiques sources** : Ces fichiers sont écrits dans le langage de politique SELinux et sont situés dans /system/sepolicy.
    - **Compiler la politique** : Android fournit un outil appelé checkpolicy pour compiler les politiques.
    - **Chargement d'une nouvelle politique** : Après avoir modifié et compilé une politique, elle doit être chargée. C'est généralement fait pendant le démarrage.

Comprendre le langage des politiques SELinux

1. **Types** : Au cœur de SELinux se trouvent les "types". Tout, des fichiers aux processus, a un type associé dans SELinux. Par exemple, le type httpd_t pourrait être associé à un serveur web.
2. **Domaines** : Les domaines sont des types associés aux processus. Lorsqu'un processus est en cours d'exécution dans un domaine particulier, cela détermine les ressources auxquelles il peut accéder.
3. **Étiquettes** : Il s'agit d'attributions de type (et d'autres attributs SELinux) à des objets, comme des fichiers ou des processus.

Comprendre la sécurité

Compilation et chargement de la politique

- Pour associer un processus à un domaine, nous utilisons ce qu'on appelle des "transitions de type". C'est essentiellement un mécanisme qui décrit comment un processus (lancé depuis un certain type/domaine) peut transiter vers un autre type/domaine lorsqu'il exécute un certain fichier.
- Une fois la politique rédigée, elle doit être compilée en un format binaire que SELinux peut utiliser. Vous utiliserez généralement l'outil checkpolicy pour cela. Ensuite, la politique compilée peut être chargée dans le système.

Conseils

- **Utilisez les outils d'audit** : Les outils d'audit SELinux peuvent vous aider à détecter les violations et à comprendre quelles permissions sont nécessaires.
- **Commencez petit** : Lorsque vous débutez, écrivez des politiques simples et étendez-les progressivement.
- **Testez en mode "permissive"** : Avant de mettre en place une nouvelle politique, testez-la en mode permissif pour voir quelles violations se produisent sans bloquer les opérations.

Comprendre la sécurité

Autorisations d'application

- Les autorisations Android déterminent quelles ressources système spécifiques une application peut accéder. Les autorisations sont déclarées par l'application et octroyées par l'utilisateur lors de l'installation ou lors de l'exécution de l'application.

Signature d'application

- La signature d'application garantit que chaque mise à jour d'application est issue de la même source et ne peut être modifiée par des tiers. Les applications doivent être signées pour être installées sur un appareil

Comprendre la sécurité

Exercice : Création d'une politique SELinux pour une nouvelle application

Contexte

Vous développez une nouvelle application pour Android nommée `CustomApp`. Cette application stocke ses fichiers de configuration dans `/data/customapp/config`. Vous devez écrire une politique SELinux pour garantir que seul `CustomApp` peut accéder à ce répertoire.

Instructions

1. Définissez un nouveau type pour l'exécutable `CustomApp`.
2. Définissez un nouveau domaine pour le processus `CustomApp` lorsqu'il est en cours d'exécution.
3. Écrivez une règle de transition pour assurer que lorsque `CustomApp` est lancé, il s'exécute dans le domaine que vous avez défini.
4. Écrivez les règles d'accès pour permettre à `CustomApp` de lire et d'écrire dans `/data/customapp/config`.
5. Testez votre politique en mode permissif.