Способы использования скрытого системного API при разработке устройств

Николай Пахомов
Senior Android Developer @ SberDevices

SBER DEVICES

О чем поговорим?

— Задачи, для которых недостаточно публичного API
— 5 способов воспользоваться скрытым системным кодом
— Как Android защищает свой скрытый код
— Подведение итогов

А почему не публичное Android API?

“Обычная” разработка приложений:

— Удерживать включенным экран

— Подписаться на смену геолокации

✅ Отлично!
Все это можно
сделать при помощи
публичного API

А почему не публичное Android API?

“Обычная” разработка приложений:

— Удерживать включенным экран

— Подписаться на смену геолокации

✅ Отлично!
Все это можно сделать при помощи публичного API

Новый тип устройства:

— Наличие второго экрана

— Физическая кнопка отключения камеры

Продуктовые задачи:

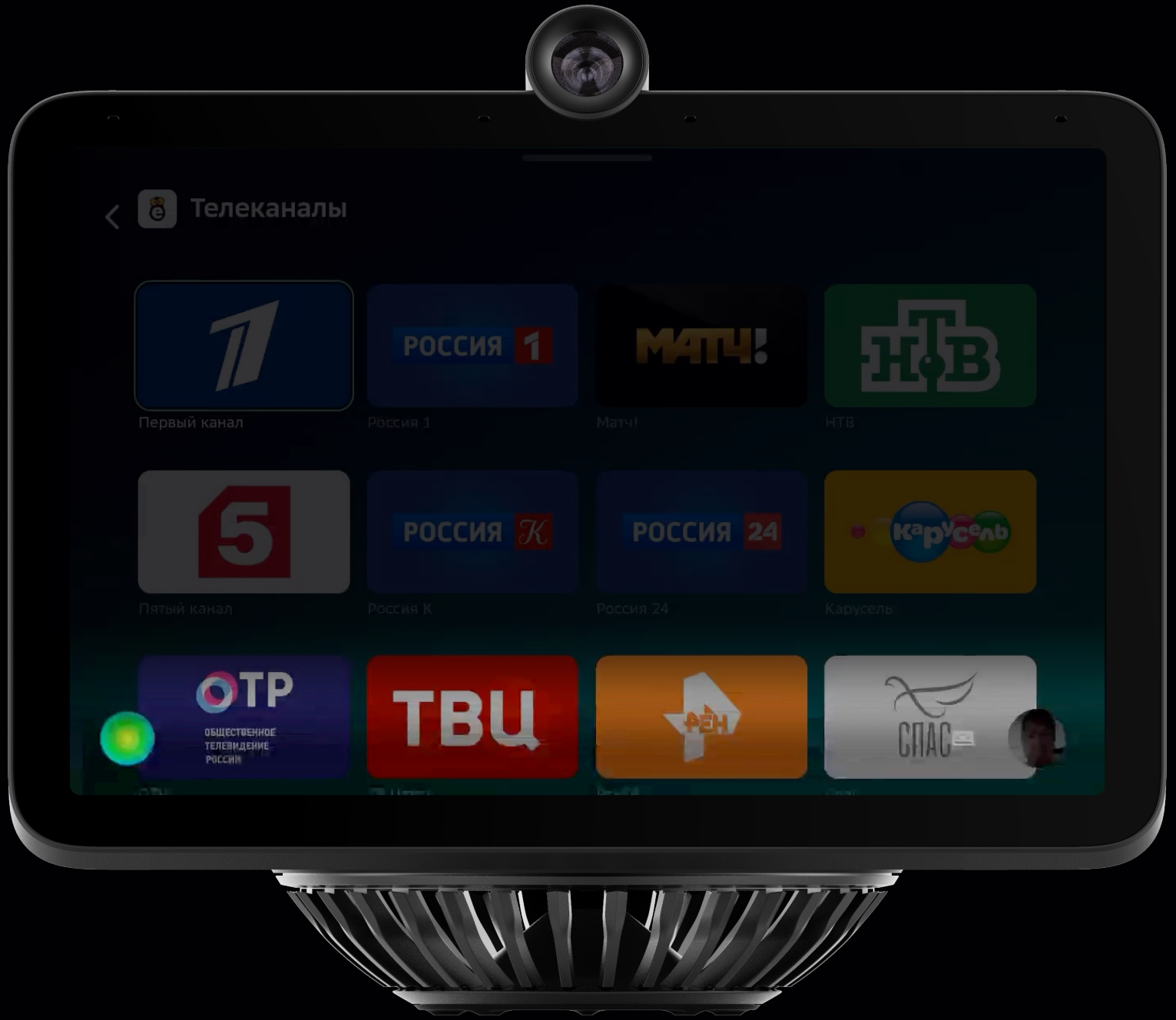
— Определять смену приложения

— Усыплять устройство при жесте рукой

— Автоматически скрывать шторку ассистента

❌ Хмм, надо бы покапаться в AOSP

Пример задачи: скрытие системного Ассистента



← Системный Ассистент

SBER DEVICES

Способы вызвать скрытый системный код

Скорость внедрения в проект

— Рефлексия
— Добавление android.jar в проект
— Добавление системных исходников в проект
— Точечное добавление системных исходников
— SDK дистрибутив от вендора

Количество подготовительной работы

Рефлексия

Используется для

— Объявления приватного поля или метода публичным

— Быстрой смены или получения значения скрытого поля

— Вызова несложного скрытого метода у имеющегося объекта

Примеры задач

— Выдать dangerous permission приложению

— Напрямую подключиться к Bluetooth устройству

Рефлексия. Пример использования

```
// Объявляем типы параметров, которые будут переданы в вызываемый метод
val classArray: Array<Class<*>> = arrayOf(String::class.java,
String::class.java, UserHandle::class.java)
```

Рефлексия. Пример использования

```
// Объявляем типы параметров, которые будут переданы в вызываемый метод
val classArray: Array<Class<*>> = arrayOf(String::class.java,
String::class.java, UserHandle::class.java)


// Заполучаем приватный метод по имени и делаем его доступным для выполнения
val method = PackageManager::class.java.getDeclaredMethod(
    "grantRuntimePermission", *classArray
).apply { isAccessible = true }
```

Рефлексия. Пример использования

```
// Объявляем типы параметров, которые будут переданы в вызываемый метод
val classArray: Array<Class<*>> = arrayOf(String::class.java,
String::class.java, UserHandle::class.java)


// Заполучаем приватный метод по имени и делаем его доступным для выпол
val method = PackageManager::class.java.getDeclaredMethod(
    "grantRuntimePermission", *classArray
).apply { isAccessible = true }


// Вызываем метод. Передаем объект класса и ранее объявленные параметры
method.invoke(packageManager, "ru.demo.test",
        Manifest.permission.READ_EXTERNAL_STORAGE,
        android.os.Process.myUserHandle()
)
```

Таблица сравнения

	Рефлексия	Android.jar	Копирование системных исходников	Точечное копирование	Vendor's SDK
Unit тесты					
Доступ к системным объектам					
Работа с приватными полями	✅				
Разные версии ОС	✅				
Доступ к системным колбэкам					
Навигация среди скрытого API					
Интеграция в SDK Manager					

Добавление android.jar в проект

— Подключение специальной зависимости в проект

Используется для

- Доступа к огромной базе системного кода
- Удобной навигации и поиска среди скрытых классов и методов в IDE

Примеры задач

- Доступ к системным properties
- Следить за списком запущенных приложений
- Подписываться на смену System UI флагов

Добавление android.jar в проект

— Объедините android.jar из SDK платформы и framework.jar, находящийся на устройстве
Полный гайд вы найдете, отсканировав QR код

— Добавьте объединённый файл к вашему проекту в качестве gradle зависимости
compileOnly files("libs/android-${versions.targetSdk}.jar")

— Для навигации среди нового кода замените студийное SDK на android.jar
@ sdk/platforms/android-XX/android.jar

Пример использования android.jar

```
private val statusBarService: IStatusBarService? =
        IStatusBarService.Stub.asInterface(ServiceManager.getService(Context.STATUS_BAR_SERVICE))
```

Пример использования android.jar

```
private val statusBarService: IStatusBarService? =
        IStatusBarService.Stub.asInterface(ServiceManager.getService(Context.STATUS_BAR_SERVICE))


private val statusBarCallbacks = object : AbstractStatusBarCallbacks() {
    override fun setSystemUiVisibility(vis: Int, _: Int, _: Int, _: Int, _: Rect?, _: Rect?) {
        onSystemUiVisibilityChanged()
    }
}
```

Пример использования android.jar

```
private val statusBarService: IStatusBarService? =
        IStatusBarService.Stub.asInterface(ServiceManager.getService(Context.STATUS_BAR_SERVICE))


private val statusBarCallbacks = object : AbstractStatusBarCallbacks() {
    override fun setSystemUiVisibility(vis: Int, _: Int, _: Int, _: Int, _: Rect?, _: Rect?) {
        onSystemUiVisibilityChanged()
    }
}


init {
    statusBarService?.registerStatusBar(
        statusBarCallbacks, listOf(), listOf(), IntArray(9), listOf(), Rect(), Rect()
    )
}
```

Таблица сравнения

	Рефлексия	Android.jar	Копирование системных исходников	Точечное копирование	Vendor's SDK
Unit тесты					
Доступ к системным объектам		✅			
Работа с приватными полями	✅				
Разные версии ОС	✅				
Доступ к системным колбэкам		✅			
Навигация среди скрытого API		✅			
Интеграция в SDK Manager					

Добавление системных исходников

— Подразумевает копирование файлов из исходников системы в проект

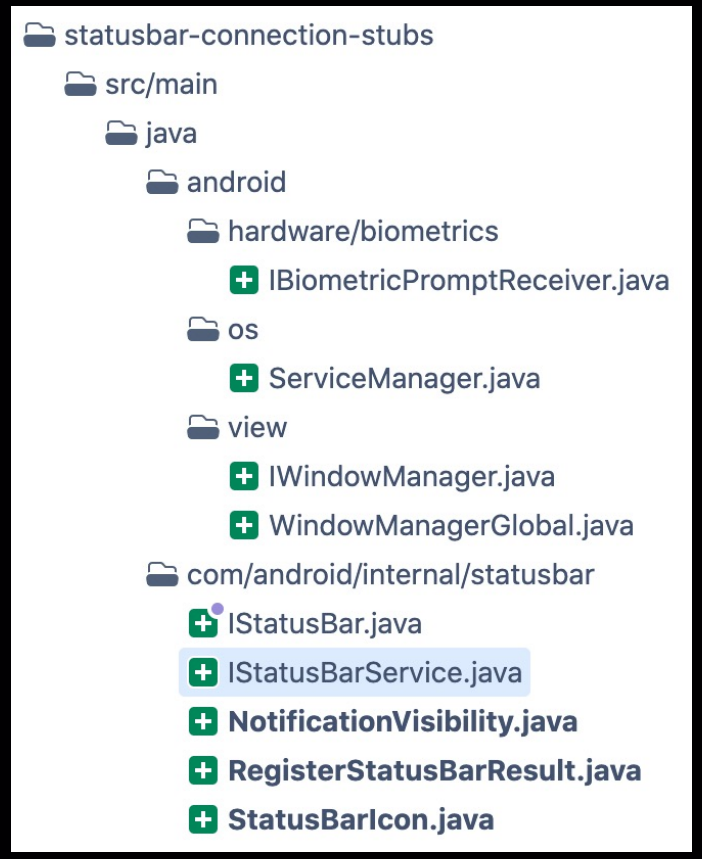
Используется для

- Того же, что и android.jar
- Разветвления бизнес логики, в зависимости от версии ОС
- Более точного контроля того, что попадет к вам в проект

Примеры задач

- Реализовать регистрацию StatusBar для нескольких версий ОС
- Переопределить механизм бэкапа приложений для нескольких версий ОС

Добавление системных исходников



1. Создайте новый compileOnly модуль
2. Найдите ресурс и исходниками системы
3. Переносите в модуль необходимые исходники AOSP. Пакеты исходников сохраните в первозданном виде
4. Проверьте наличие всех необходимых классов из импортов
5. Подключите compileOnly модуль к проекту
6. Соберите проект. В случае ошибок, проверьте пункты 3 и 4

Системные исходники. Пример

```
private val statusBarService: IStatusBarService? =
    IStatusBarService.Stub.asInterface(ServiceManager.getService("statusbar"))
```

Системные исходники. Пример

```
private val statusBarService: IStatusBarService? =
    IStatusBarService.Stub.asInterface(ServiceManager.getService("statusbar"))


private val statusBarCallbacks = object : AbstractStatusBarCallbacks() {
    override fun setSystemUiVisibility(vis: Int, _: Int, _: Int, _: Int, _: Rect?, _: Rect?) { … }

    @RequiresApi(Build.VERSION_CODES.Q)
    override fun setSystemUiVisibility(_: Int, vis: Int, _: Int, _: Int, mask: Int, …) { … }
}
```

Системные исходники. Пример

```
private val statusBarService: IStatusBarService? =
    IStatusBarService.Stub.asInterface(ServiceManager.getService("statusbar"))


private val statusBarCallbacks = object : AbstractStatusBarCallbacks() {
    override fun setSystemUiVisibility(vis: Int, _: Int, _: Int, _: Int, _: Rect?, _: Rect?) { … }

    @RequiresApi(Build.VERSION_CODES.Q)
    override fun setSystemUiVisibility(_: Int, vis: Int, _: Int, _: Int, mask: Int, …) { … }
}


init {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
        statusBarService?.registerStatusBar(statusBarCallbacks)
    } else {
        statusBarService?.registerStatusBar(
            statusBarCallbacks, listOf(), listOf(), IntArray(9), mutableListOf(), Rect(), Rect()
        )
    }
}
```

Таблица сравнения

	Рефлексия	Android.jar	Копирование системных исходников	Точечное копирование	Vendor's SDK
Unit тесты					
Доступ к системным объектам		✅	✅		
Работа с приватными полями	✅				
Разные версии ОС	✅		✅		
Доступ к системным колбэкам		✅	✅		
Навигация среди скрытого API		✅			
Интеграция в SDK Manager					

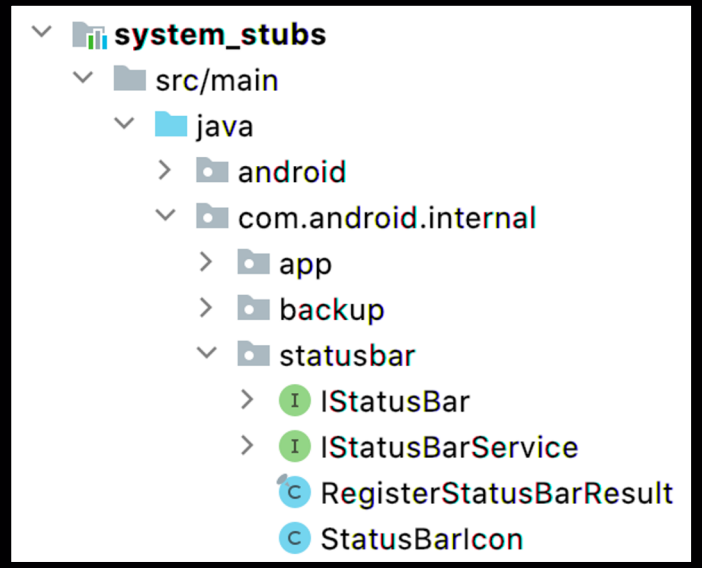
Точечное копирование исходников

— минимально необходимое количество системных файлов для требуемого функционала

Используется для

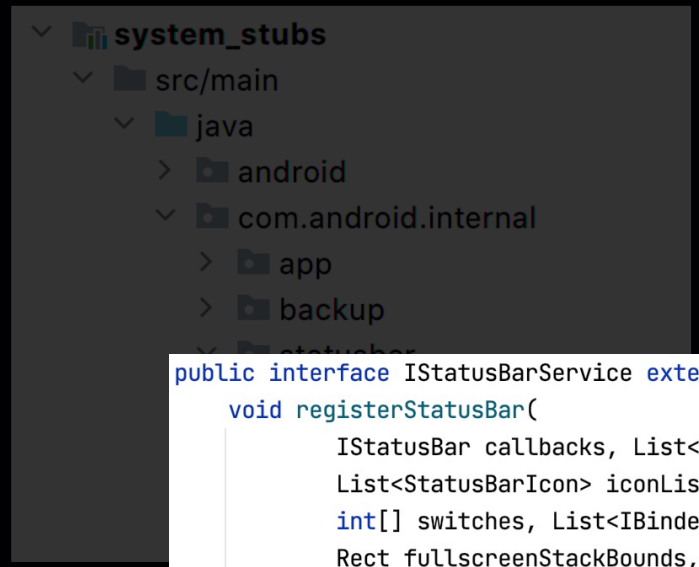
- Того же, что и обычное добавление системных исходников
- Жесткого контроля того, что попадет к вам в проект
- Уменьшения размера необходимых зависимостей
- Возможности тестирования системных вызовов

Точечное копирование исходников



1. Создайте новый compileOnly модуль системных стабов
2. Скопируйте только необходимые классы и методы для нужного функционала. Не изменяйте пакеты и имена классов

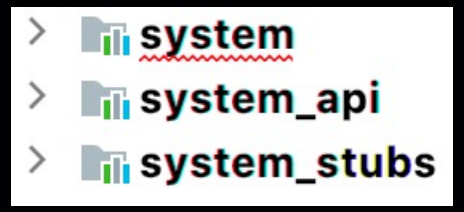
Точечное копирование исходников



1. Создайте новый compileOnly модуль системных стабов
2. Скопируйте только необходимые классы и методы для нужного функционала. Не изменяйте пакеты и имена классов

```
public interface IStatusBarService extends IInterface {
    void registerStatusBar(
            IStatusBar callbacks, List<String> iconSlots,
            List<StatusBarIcon> iconList,
            int[] switches, List<IBinder> binders,
            Rect fullscreenStackBounds, Rect dockedStackBounds);

    // Since API 29
    RegisterStatusBarResult registerStatusBar(IStatusBar callbacks);

    abstract class Stub extends Binder implements IStatusBarService {

        public Stub() { throw new RuntimeException("Stub!"); }

        public static IStatusBarService asInterface(IBinder obj) {
            throw new RuntimeException("Stub!");
        }
    }
}
```

Точечное копирование исходников




```
dependencies {
    api project(':common:system_api')
    compileOnly project(':common:system_stubs')
}
```

Для поддержки тестирования:

1. Создайте новый system модуль, который будет оберткой над модулем system_stubs
2. Добавьте system_stubs модуль к system в качестве compileOnly зависимости
3. Теперь оберните системные вызовы своими методами-обертками
4. Теперь вы можете добавить system модуль к другим модулям
5. Вместо тестирования вызовов системного кода, тестируйте вызовы методов-оберток

Таблица сравнения

	Рефлексия	Android.jar	Копирование системных исходников	Точечное копирование	Vendor's SDK
Unit тесты				✅	
Доступ к системным объектам		✅	✅	✅	
Работа с приватными полями	✅				
Разные версии ОС	✅		✅	✅	
Доступ к системным колбэкам		✅	✅	✅	
Навигация среди скрытого API		✅			
Интеграция в SDK Manager					

Vendor’s SDK

— специальный дистрибутив СДК, который совмещает в себе код вендора и(ли) код AOSP

Используется для

— Взаимодействия с новыми HAL’ами (системными драйверами)

— Все, что было описано в предыдущих способах 🙂

Примеры задач

— Взаимодействие с кастомными командами HDMI-CEC

— Выключение и включение микрофона при нажатии на кнопку на устройстве

Vendor’s SDK

Что нужно сделать вендору

1. Написать и собрать SDK
2. Положить полученный .jar в /vendor/framework в качестве артефакта прошивки
3. Опубликовать .jar в репозиторий, который будет добавлен через SDK Manager

Что нужно сделать разработчику

1. Добавить ссылку на репозиторий вендора в SDK Manager
2. Включить SDK для нужных версий платформы
3. Если SDK вендора переопределяет код AOSP, то отключить штатное SDK платформы
4. Выставить compileSDK версию на ту, что определил вендор

Таблица сравнения

	Рефлексия	Android.jar	Копирование системных исходников	Точечное копирование	Vendor's SDK
Unit тесты				✅	✅
Доступ к системным объектам		✅	✅	✅	✅
Работа с приватными полями	✅				✅
Разные версии ОС	✅		✅	✅	✅
Доступ к системным колбэкам		✅	✅	✅	✅
Навигация среди скрытого API		✅			✅
Интеграция в SDK Manager					✅

Как compileOnly исходники оказываются в рантайме?

- У каждого приложения есть свой ClassLoader
  ClassLoader – инструмент Java, ответственный за загрузку классов в JVM
- У каждого класса есть свой идентификатор, состоящий из:
  - Имени класса
  - Пакета, в котором находится класс
  - ClassLoader, который загружает этот класс
- При совпадении идентификатора с классами из /system/framework/*.jar, compileOnly класс подтягивается в рантайме

Как Android защищает себя от использования скрытого API?

@Hide аннотации ← Говорят компилятору SDK, что метод, поле или класс нужно удалить

Non-SDK API lists

- SDK: PowerManager.WakeLock#acquire
- Unsupported: PowerManager#goToSleep
- Conditionally blocked: PowerManager#shutdown
- Blacklist: PowerManager#getLastSleepReason

Пермишены

Signature

— Выдаются приложениям с одинаковой подписью (и даже, если совпадает с системной!)

— Расположение приложения не имеет значения

Privileged

— Не требуют системной подписи у приложения

— Требуют наличия приложения в priv-app директории:
/ [**system**/**product**/**vendor**] /priv-app

— Начиная с Android 8, вендор должен сам указывать приложения, которым выдается этот пермишен через специальный xml

Пример использования пермишенов

```
public RegisterStatusBarResult registerStatusBar(IStatusBar bar) {
    enforceStatusBarService(); // Здесь будет проверка пермишена
    ...
}
```

Пример использования пермишенов

```
public RegisterStatusBarResult registerStatusBar(IStatusBar bar) {
    enforceStatusBarService(); // Здесь будет проверка пермишена
    ...
}


private void enforceStatusBarService() {
    mContext.enforceCallingOrSelfPermission(
            Manifest.permission.STATUS_BAR_SERVICE, "StatusBarManagerService");
}
```

Пример использования пермишенов

```
public RegisterStatusBarResult registerStatusBar(IStatusBar bar) {
    enforceStatusBarService(); // Здесь будет проверка пермишена
    ...
}


private void enforceStatusBarService() {
    mContext.enforceCallingOrSelfPermission(
            Manifest.permission.STATUS_BAR_SERVICE, "StatusBarManagerService");
}


<permission android:name="android.permission.STATUS_BAR_SERVICE"
    android:protectionLevel="signature"/>
```

Linux PID / UID

- PID (ProcessID) – уникальный ID процесса, под которым запущено приложение или его компонента
- UID (UserID) – ID пользователя, под которым запущено приложение. Не уникально!
- Binder.getCallingPid – возвращается PID приложения или компоненты, которое вызвало текущий метод. Если IPC не было, вернет текущий PID
- Binder.getCallingUid – возвращается UID приложения, которое дернуло текущий метод

  android:sharedUserId="android.uid.system”
  android:sharedUserId="android.uid.systemui"

Пример проверки UID

```
private void enforceStatusBarOrShell() {
    if (Binder.getCallingUid() == Process.SHELL_UID) {
        // Метод вызвали из ADB shell
    ...
    }

    if (Binder.getCallingUid() == 1000 /* Системный UID */) {
        // Метод был вызван системой или системных приложением
    ...
    }

    throw SecurityException()
}
```

Как защитить свои скрытые APIs?

От использования рефлексии

— Использовать приватные конструкторы

— Кидать exception внутри конструктора

— Использовать NDK

От копирования файлов / android.jar / Vendor SDK

— Проверять UID и(ли) PID

— Использовать собственный ClassLoader

— Использовать NDK

Итоги

	Рефлексия	Android.jar	Копирование системных исходников	Точечное копирование	Vendor's SDK
Unit тесты				✅	✅
Доступ к системным объектам		✅	✅	✅	✅
Работа с приватными полями	✅				✅
Разные версии ОС	✅		✅	✅	✅
Доступ к системным колбэкам		✅	✅	✅	✅
Навигация среди скрытого API		✅			✅
Интеграция в SDK Manager					✅

И еще кое-что!

Николай Пахомов, SberDevices
pakhomov.nick@gmail.com
Telegram: @FlyRanger

SBER DEVICES


SBER DEVICES