به نام خدا

موضوع پروژه: iProtect

کارکرد پروژه: مدیریت دسترسی اپلیکیشن‌ها به بلوتوث، وای‌فای و دیتای موبایل

توسعه دهنده: حسین کرمی

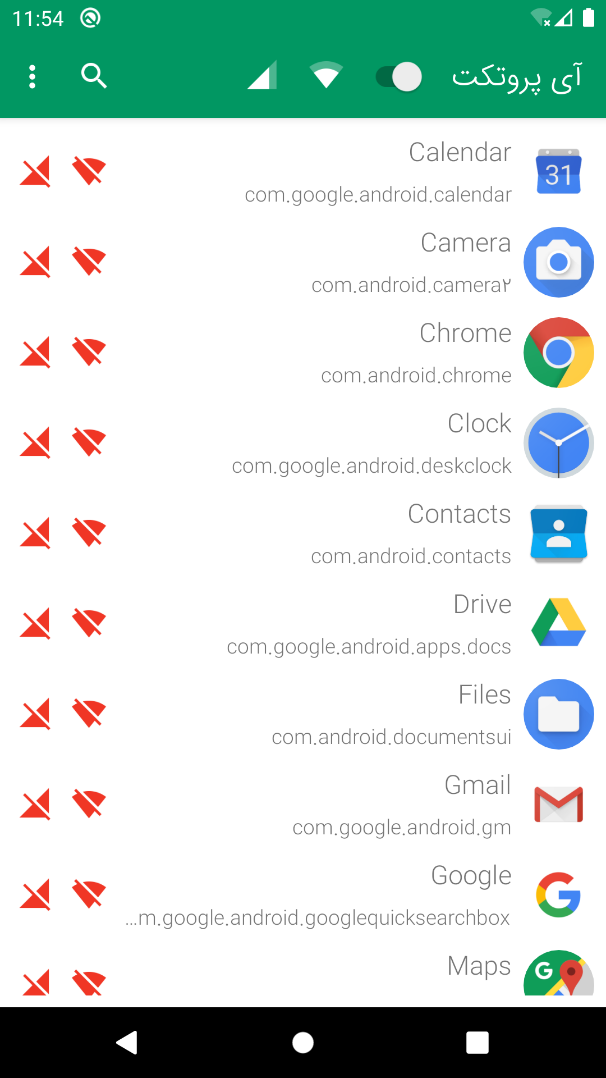
امکانات پروژه: اپلیکیشن اندرویدی iProtect

زبان‌های توسعه: Kotlin و Java

نحوه کارکرد اپلیکیشن

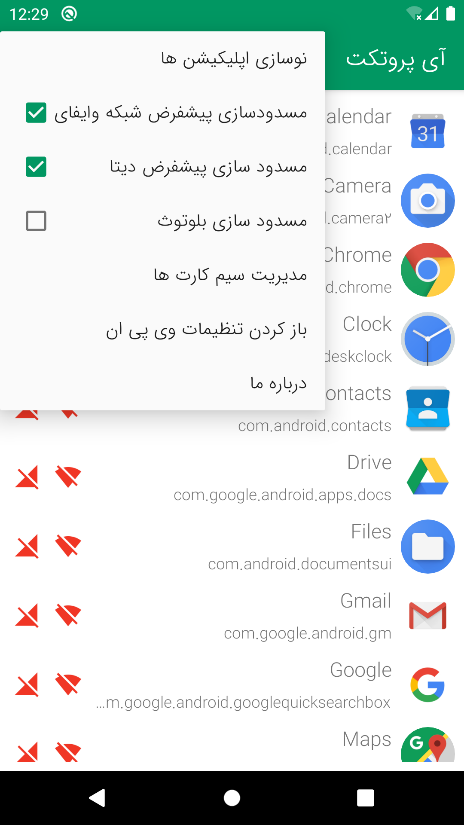
اپلیکیشن اندرویدی iProtect قابلیت مدیریت ارسال و دریافت داده از سه شبکه وای‌فای، دیتای موبایل و بلوتوث را داراست. به کمک این اپلیکیشن می‌توانید داده‌های موبایل را کنترل کنید و همچنین از ارسال و دریافت فایل توسط بلوتوث جلوگیری کنید.

برای مدیریت داده موبایل و وای‌فای، این قابلیت وجود دارد که برای تمامی اپلیکیشن‌ها به صورت دقیق مشخص کنید که کدامیک از اپلیکیشن‌ها اجازه دسترسی خواهد داشت. بدین صورت، می‌توانید وضعیت اجازه دسترسی به شبکه را به صورت مشخص برای هر اپلیکیشن تعیین کنید.



همچنین اگر مایل به قطع کامل دسترسی دستگاه اندرویدی به شبکه هستید، کافیست از همه اپلیکیشن‌ها اجازه دسترسی را بگیرید تا داده شبکه به کلی قطع شود.

یکی از قابلیت‌های دیگر اپلیکیشن، امکان صدور مجوز برای دسترسی به داده شبکه برای سیم‌کارت‌های خاص است. برای اینکه بتوانید برای سیم‌کارت خاصی مجوز صادر کنید، به مراحل آن در زیر نگاهی بیندازید:

برنامه‌نویسی اپلیکیشن

برای برنامه‌نویسی این اپلیکیشن از دو زبان کاتلین و جاوا استفاده شده است. کاتلین زبان رسمی توسعه اپلیکیشن‌های اندروید است و سازگاری بسیار زیادی با زبان جاوا و کلاس‌ها و وابستگی‌های نوشته شده به زبان جاوا دارد.

همچنین زبان کاتلین برای مدیریت وابستگی‌ها از Gradle استفاده می‌کند.

توضیحات کد پروژه

در این بخش به بررسی برنامه‌نویسی پروژه می‌پردازیم. در اینجا ما به بررسی مهمترین مسائل و کارکرد کدی اپلیکیشن خواهیم پرداخت.

کلاس BlackHoleService

در مسیر /app/ir/stackcode/iprotect یک کلاس تحت عنوان BlackHoleService است. وظیفه این سرویس این است که در حالت اجرا، یک کانکشن Tunnel کند. در حین ایجاد کانکشن ما اپلیکیشن‌هایی که مجاز به دسترسی هستند را مشخص خواهیم کرد. این اطلاعات از داده‌های ذخیره شده در SharedPreferences استفاده می‌کند. این اطلاعات ذخیره شده، همان انتخاب‌های کاربر برای دسترسی به اپلیکیشن‌های خود است. در واقع یک SharedPreference به نام همان پکیج نزم‌افزار ذخیره می‌کنیم و سپس به کمک آن متوجه خواهیم شد که تنظیمات پیشفرض کاربر چیست.

*// Check if Wi-Fi***boolean** wifi = Util.*isWifiActive*(**this**);  
Log.*i*(***TAG***, **"wifi="** + wifi);  
  
*// Build VPN service***final** Builder builder = **new** Builder();  
builder.setSession(getString(R.string.***app\_name***));  
builder.addAddress(**"10.1.10.1"**, 32);  
builder.addAddress(**"fd00:1:fd00:1:fd00:1:fd00:1"**, 128);  
builder.addRoute(**"0.0.0.0"**, 0);  
builder.addRoute(**"0:0:0:0:0:0:0:0"**, 0);  
  
*// Add list of allowed applications***for** (Rule rule : Rule.*getRules*(**this**))  
 **if** (!(wifi ? rule.**wifi\_blocked** : rule.**other\_blocked**) && HelperSimCard.***INSTANCE***.isAllowed(**this**)) {  
 Log.*i*(***TAG***, **"Allowing "** + rule.**info**.**activityInfo**.**packageName**);  
 **try** {  
 builder.addDisallowedApplication(rule.**info**.**activityInfo**.**packageName**);  
 } **catch** (PackageManager.NameNotFoundException ex) {  
 Log.*e*(***TAG***, ex.toString() + **"\n"** + Log.*getStackTraceString*(ex));  
 }  
 }

در قطعه کد بالا، تنظیمات اولیه builder کانکشن و نشست را تنظیم می‌کنیم. در حلقه for تمامی Rule‌ها را پابش می‌کنیم و در صورتی که نیاز به بلاک کردن وای‌فای و یا دیتای موبایل بود را بررسی می‌کنیم. همچنین بر اساس ICCID سیم کارت‌های مجاز را هم بررسی کرده و در صورتی که مشکلی نبود و از همه نظر دسترسی مجاز بود، به پکیج مورد نظر که همان اپلیکیشن است، دسترسی داده می‌شود.

در غیر اینصورت دسترسی از آن پکیج گرفته می‌شود.

پس در حالت کلی، این سرویس کانکشن را با بررسی پارامتر‌های مختلف مانند ICCID‌های مجاز، وضعیت دسترسی به وای‌فای و دیتای موبایل، امکان دسترسی را به اپلیکیشن‌های مورد نظر فراهم می‌کند و در صورتی هم که نیاز به بلاک شدن اپلیکیشن بود، اینکار انجام خواهد گرفت.

کلاس BluetoothReceiver

اگر به Android Manifest نگاهی بیندازید، می‌توانید کد‌های زیر را در آن مشاهده کنید:

<**receiver  
 android:name=".receivers.BluetoothReceiver"  
 android:exported="true"**>  
 <**intent-filter**>  
 <**action android:name="android.bluetooth.device.action.ACL\_CONNECTED"** />  
 <**action android:name="android.bluetooth.device.action.ACL\_DISCONNECTED"** />  
 <**action android:name="android.bluetooth.device.action.ACL\_DISCONNECT\_REQUESTED"** />  
 </**intent-filter**>  
</**receiver**>

در اینجا به یک Receiver اشاره شده که در صورتی که وضعیت بلوتوث تغییر کند، Bluetooth Receiver و متد onReceiver صدا زده خواهد شد.

وظیفه این Recevier این است که اگر بلوتوث متصل شد ولی بلوتوث اجازه ارسال و دریافت داده را نداشت، آن را غیر فعال کند.

اگر به کلاس Bluetooth Receiver نگاهی بیندازید، این کد‌ها را مشاهده خواهید کرد:

**if** (intent.*action* == BluetoothDevice.*ACTION\_ACL\_CONNECTED* && !HelperPreferences.isBluetoothAllowed(  
 context  
 )  
)  
 BluetoothAdapter.getDefaultAdapter().disable()

همانگونه که می‌بینید، اگر بلوتوث متصل باشد ولی بلوتوث مجوز دسترسی نداشته باشد، به کمک کلاس BluetoothAdapter آن را غیر فعال می‌کنیم و عملا ارسال و دریافت داده متوقف می‌شود.

کلاس HelperSimCard

این کلاس در واقع یک Helper برای کار با سیم کارت است. در اپلیکیشن iProtect این قابلیت وجود دارد که برای سیم‌کارت‌های مختلف مجوز صادر کنید.

به این معنا که اگر قصد دسترسی دادن به سیم‌کارت‌ها خاص برای کار با داده هستید، می‌توانید از شناسه منحصربفرد ICCID هر سیم کارت استفاده کنید و آن را در بخش سیم‌کارت‌های مجاز قرار دهید.

حال این کلاس یک متد تحت عنوان isAllowed دارد که وظیفه آن بررسی امکان دسترسی کاربر به داده است.

به کد زیر توجه کنید:

**fun** isAllowed(context: Context): Boolean {  
 **val** database = SimCardDatabaseHelper(context)  
  
 **val** wifi = Util.isWifiActive(context)  
  
 **val** simCards = database.all()  
 **if** (simCards.**size** == 0)  
 **return true  
 if** (ActivityCompat.checkSelfPermission(  
 context,  
 Manifest.permission.*READ\_PHONE\_STATE* ) != PackageManager.*PERMISSION\_GRANTED* ) {  
 Toast.makeText(  
 context,  
 context.getString(R.string.*phone\_state\_permission\_not\_allowed*),  
 Toast.*LENGTH\_LONG* ).show()  
  
 **return true** }  
  
 **val** sis = SimCardManager.getICCIDList(context)  
  
 sis.*forEach* **{** si **->** simCards.*forEach* **{** si2 **->  
 if** (  
 *// si.subscriptionId == getDefaultDataSubscriptionId(sm) &&* si == si2.**ICCID** && !(**if** (wifi) si2.**wifi else** si2.**other**))  
 **return true  
 }  
 }  
  
 return false**}

در این متد ما سیم‌کارت‌های فعال را از دیتابیس بدست می‌آوریم. اگر تعداد سیمکارت‌های ذخیره شده صفر باشد، به این معنیست که هیچ محدودیتی برای هیچ سیم‌کارتی وجود ندارد و لذا می‌توانید بدون مشکل به داده دسترسی داشته باشید.

اگر سیم‌کارتی ذخیره شده باشد، قبل از اینکه ادامه کار را انجام دهیم، باید بررسی کنیم که آیا به وضعیت دستگاه دسترسی داریم یا خیر. دسترسی به این مجوز (PHONE\_STATE) برای ادامه مهم است.

اگر دسترسی وجود داشت، ICCID را بررسی می‌کنیم و اگر دسترسی برای آن سیم‌کارت وجود داشت، مجوز صادر شده و مقدار true برمیگردد.