بنام خدا

مستند برنامه نویسی امن در محیط اندروید

مرکز ماهر

زمستان ۹۵

فهرست

۱ معماری اندروید

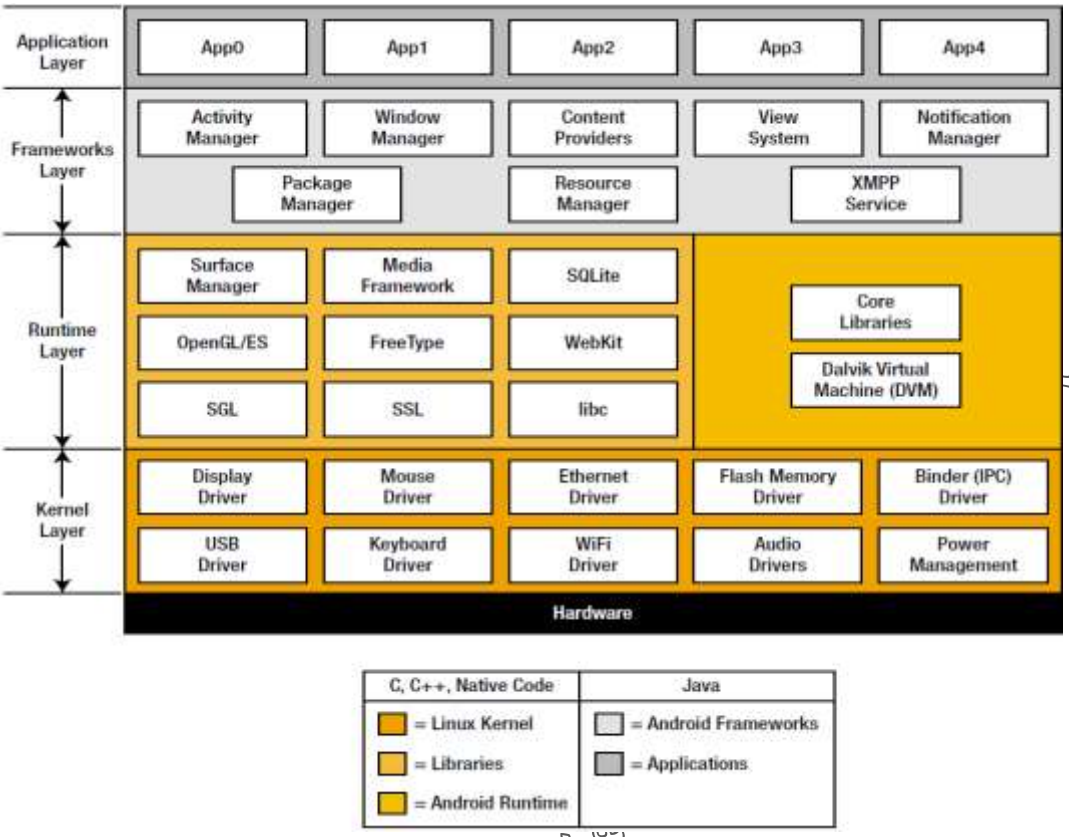
تلفن های هوشمند به یکی از ابزارهای زندگی روزمره ی مردم جهان تبدیل شده است. ارتباط با مخاطبان و آ شنایان، پرداخت های بانکی، مرور اینترنت، دریافت اخبار و اطلاعات روزانه و غیره تنها بخشی از اموری است که میتوان توسط یک تلفن هوشمند در طول روز انجام داد. رسیدن آمار ۲ میلیون تلفن هو شمند در سال ۹۲ به ۲۰ میلیون در سال ۹۳ و گذر از ٤٠ میلیون در سال ۹۵ نشان دهنده ی فراهم شدن ابزار دولت موبایل و محبوبیت فراوان این د ستگاه در بین مردم ک شور ا ست[۱]. بنابراین ب سیاری از اطلاعات حساس و مهم افراد در تلفن های هو شمند ذخیره می شود و امنیت تلفن های هو شمند حتی ممکن ا ست از امنیت حساب بانکی یک شخص بیشتر باشد. برای فراهم آوردن امنیت در تلفن های هوشمند که تحت سیستم عامل اندروید هستند ابتدا باید با معماری سیستم عامل اندروید آشنا شد. یکی از ویژگی‌های منحصربه‌فرد سیستم‌عامل اندروید که موجب رشد سریع آن شده است؛ کدهای باینری و منبع آن است که به‌صورت متن‌باز ارائه شده‌اند. هرکسی می‌تواند کد منبع آن را دانلود نماید و برای دستگاه موبایل خود سفارشی نماید.

۱-۱ مؤلفه‌های معماری اندروید

معماری اندروید به چهار مؤلفه اصلی زیر تقسیم می‌شود(تصویر۱-۱ معماری را مشاهده نمایید):

۱. هسته[۲]
۲. کتابخانه‌ها و ماشین‌مجازی Dalvik
۳. چارچوب برنامه کاربردی
٤. برنامه‌های کاربردی[۳]

[۱] http://digiato.com/
[۲] Kernel
[۳] Application



تصویر ۱-۱ معماری اندروید

۱-۱-۱ هسته

هسته اندروید براساس یکی از نسخه های هسته لینوکس است. از آوریل سال ۲۰۱٤ دستگاه های اندرویدی به طور عمده از هسته لینوکس با نسخه های ۳,٤ یا ۳,۱۰ یا ۳,۱۸ استفاده کرده اند؛ نسخه هسته لینوکسی مورد استفاده به دستگاه و بستر سخت افزاری آن وابسته است. به صورتی که در اندروید ۷ ملقب به اندروید نوقا از نسخه ی ٤,٤,۱ هسته ی لینوکس استفاده شده است. هسته، اولین لایه از نرم‌افزار است که با سخت‌افزار دستگاه در ارتباط است. هسته اندروید وظیفه مدیریت از توان، حافظه، درایورها، فرایندها، شبکه و امنیت دستگاه را بعهده دارد. هسته اندروید را می توانید از سایت https://android.googlesource.com دریافت نمایید.

اصلاح و ساخت یک هسته جدید، کاری نیست که شما بخواهید به‌عنوان یک توسعه‌دهندهٔ نرم‌افزار آن را انجام دهید. به‌طورکلی، فقط تولیدکنندگان سخت‌افزار یا دستگاه می‌خواهند هسته را به منظور اطمینان از عملکرد صحیح سیستم‌عامل بر روی سخت‌افزارِ خاص تغییر دهند.

۱-۱-۲ کتابخانه‌ها

کتابخانه‌ها فضـــایشـــان را با مؤلفه زمان اجرا[۱] به اشـــتراک می‌گذارند. کتابخانه‌ها به‌عنوان یک‌لایه ترجمه[۲]، بین هسـته و چارچوب برنامه عمل می‌کنند. کتابخانه‌ها در C/C++ نوشـته شـده‌اند اما از طریق یک API[۳] جاوا به توسعه‌دهندگان، ارائه‌شده است. API مجموعه ای از بسته ها و کدها برای تفکیک نسخه های مختلف چارچوب[۴] اندروید می باشد و در اختیار توسعه دهندگان قرار می‌گیرد تا با استفاده از آن برنامه های خود را بنویسند. توسعه‌دهندگان می‌توانند با استفاده از برنامه‌ی جاوا به هسته زیربنایی کتابخانه‌های C/C++ دسترسی داشته باشند. بعضی از کتابخانه‌های مرکزی شامل موارد زیر است:

- LibWebCore : جهت اجازهٔ دسترسی به مرورگر وب.
- کتابخانه‌های رسانه[۵]: جهت اجازهٔ دسترسی به توابع ضبط و پخش فایل صوتی تصویری.
- کتابخانه گرافیکی: جهت اجازه دسترسی به موتورهای طراحی گرافیک 2D و 3D.

مؤلفه زمان اجرا شامل ماشین‌مجازی Dalvik که با اجرای برنامه‌های کاربردی تعامل دارد. ماشین‌مجازی بخش مهمی از سیستم‌عامل اندروید است و برنامه‌های بخش ثالث[۶] و سیستم را اجرا می‌کند.

۱-۱-۳ ماشین‌مجازی Dalvik

دن‌بورن ا شتاین ما شین‌مجازی Dalvik را نو شته ا ست و در درجه اول ما شین‌مجازی Dalvik برای اجرای برنامه‌ها روی دسـتگاه‌هایی با منابع محدود نوشـته‌شـده بود که تلفن‌های همراه در این دسـته قرار می‌گیرند چراکه آن‌ها نسـبت به سـایر دسـتگاه های رایانه‌ای از قدرت پردازش، مقدار حافظه در دسـترس و عمر باطری کوتاهی برخوردارند.

ماشین‌مجازی چیست؟

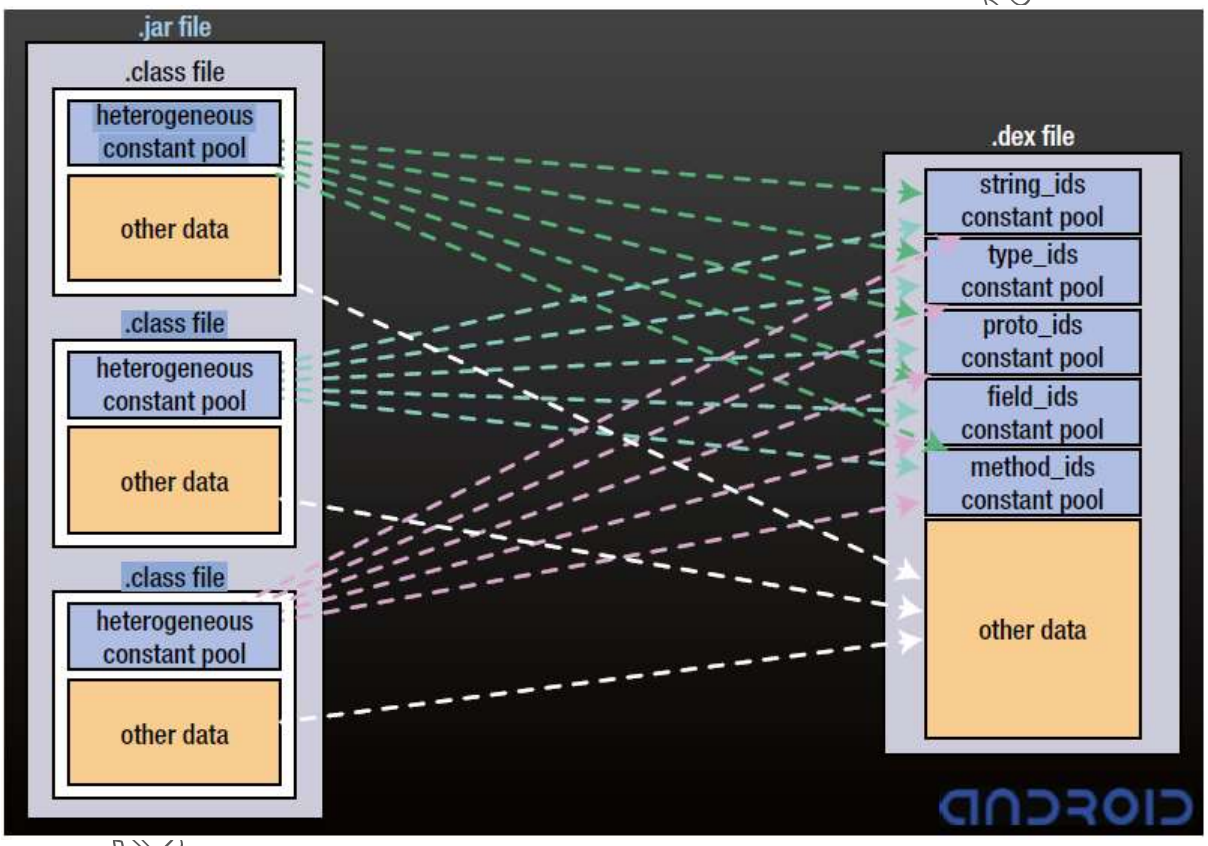
ماشـین‌مجازی یک سـیسـتم‌عامل میهمان مجزا شـده اسـت که در داخل سـیسـتم‌عامل میزبان اجرا می‌شود. یک ماشین‌مجازی برنامه‌های کاربردی را که در حال اجرا بر روی ماشین فیزیکی است اجرا خواهد

---

[۱] Runtime component
[۲] Translation Layer
[۳] Application Programming Interface
[۴] Framework
[۵] Media libraries
[۶] Third-party

کرد. یکی از مزایای اصلی ماشین‌مجازی قابل‌حمل بودن آن است. صرف‌نظر از سخت‌افزار زیربنایی، کدی که توسعه‌دهنده می‌نویسد بر روی ماشین‌مجازی اجرا خواهد شد. برای توسعه‌دهنده، بدین معنی است که فقط یک‌بار کد را می‌نویسد و می‌تواند آن را بر روی هر سخت‌افزاری که با ماشین‌مجازی سازگار است اجرا کنید.

ماشـــین‌مجازی Dalvik فایل‌های dex. را اجرا می‌کند. یک فایل dex. از فایل‌های class. یا jar. کامپایل شده ایجاد می شود و همه ثابت‌ها و داده‌های درون هر فایل با پسوند class. در یک مجموعه ثابت مشترک جمع می شوند.(تصویر۱-۲ را ببینید). ابزار dx در SDK[1] اندروید این تبدیل را انجام می‌دهد. اندازهٔ فایل‌های dex. پس از تبدیل به میزان قابل‌توجهی کوچک می‌شود، همان‌طور که در جدول ۱-۱ می‌بینید.



تصویر۱-۲. تبدیل فایل jar. به فایل dex.

جدول ۱-۱: مقایسه اندازهٔ فایل (برحسب بایت) بین فایل‌های .jar و .dex

[1] Software Development Kit

Application	Uncompressed .jar	Compressed .jar	Uncompressed .dex
Common system libraries	21445320 =100%	10662048 = 50%	10311972 = 48%
Web browser app	470312 = 100%	232065 = 49%	209248 = 44%
Alarm clock app	119200 = 100%	61658 = 52%	53020 = 44%

٤-١-١ چارچوب برنامه کاربردی

چارچوب برنامه کاربردی، یکی از بخش‌های معماری اندروید به منظور ارتباط برنامه‌های کاربردی با کاربران است. این لایه یک مجموعه از خدمات یا سیستم‌های مفیدی را برای یک توسعه‌دهنده و برنامه‌نویس برنامه کاربردی فراهم می‌کند. این چارچوب معمولاً به مؤلفه‌های API (رابط برنامه‌نویسی کاربردی) اشاره دارد، که دسترسی به مؤلفه‌های رابط کاربر مثل دکمه[۱] و جعبه متن[۲]، ارائه‌دهنده محتوای مشترک[۳] (برای اشتراک داده‌ها بین برنامه‌ها)، مدیریت هشداره (برای آگاهی صاحبان دستگاه از رخدادها) و مدیر فعالیت (برای مدیریت چرخه‌ی عمر برنامه کاربردی) را برای یک توسعه‌دهنده تدارک می‌بیند.

یک توسعه‌دهنده از API ها برای دسترسی به کتابخانه ها استفاده می‌کند. کد ۱-۱، یک نمونه نمایش از یک API است که نحوه‌ی استفاده از چارچوب برنامه کاربردی برای خواندن فایل‌های تصویری را نشان می‌دهد. عبارت import، اجازه دسترسی به هسته‌ی کتابخانه‌های C/C++ از طریق API جاوا را فراهم می‌سازد.

توجه: دقت کنید تمامی نمونه کدهای به کار برده شده در کتاب در اندروید نسخه ۵ (API 21 & 22) و اندروید نسخه ٦ (API 23) تست شده اند و به دلیل اینکه در زمان ویرایش این کتاب اندروید نسخه ۷ (API 24 & 25) به صورت پایدار عرضه نشده، در ویرایش فعلی تست نشده است اما به احتمال قوی می‌توان گفت که اکثر مثال ها در نسخه ۷ اندروید نیز بدون مشکل اجرا خواهند شد.

کد ۱-۱: نسخه نمایشی اجرای فایل تصویری

```
package android.secure.programming;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.Toast;
import android.widget.VideoView;

public class MainActivity extends AppCompatActivity {
    private String mPath = "";
```

[۱] Button

[۲] Text box

[۳] content provider

```
    private VideoView mVideoView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mVideoView = (VideoView) findViewById(R.id.videoView);
        if (mPath.equals("")) {
            // Tell the user to provide a media file URL/path.
            Toast.makeText(MainActivity.this,
                "Please edit VideoViewDemo Activity, and set path" +
                " variable to your media file URL/path",
                Toast.LENGTH_LONG).show();
        } else {
             mVideoView.setVideoPath(mPath);
             mVideoView.setMediaController(new MediaController(this));
             mVideoView.requestFocus();
        }
    }
}
```

۱-۱-۵ برنامه‌های کاربردی

برنامه کاربردی اندروید نزدیک‌ترین لایه به کاربر نهایی است. در این لایه برنامه‌های تماس‌ها، تلفن، پیام، بازی‌ها و غیره قرارگرفته است. محصول نهایی شده با استفاده از کتابخانه API و ماشین‌مجازی در این فضا اجرا خواهد شد. گر چه هر بخش سیستم‌عامل اندروید می‌تواند تغییر کند ولی توسعه‌دهنده فقط روی امنیت نرم‌افزار خود کنترل مستقیم دارد.

۱-۲ درباره برنامه‌نویسی امن

حال که یک درک و دید کلی از معماری اندروید حاصل شد، اجازه دهید به چیزهایی که در اینجا نخواهید آموخت بپردازیم. اول، شما چگونگی توسعه برنامه‌های اندروید را در این کتاب یاد نخواهید گرفت. شما نمونه‌ها و لیست کد منبع‌های زیادی خواهید دید. و زمانی که هر بخشی از کد توضیح داده می شود، ممکن است شما سؤال‌هایی داشته باشید که احتمالاً جواب آن را در این نوشته پیدا نکنید. شما به یک درجه‌ی خاصی از تجربه و مهارت در نوشتن برنامه‌های کاربردی جاوا برای پلتفرم اندروید نیاز دارید. همچنین فرض می شود که شما در حال حاضر محیط توسعه اندروید خود را با استفاده از محیط توسعه Android Studio IDE راه‌اندازی کرده‌اید.[1] در اینجا، بر این تمرکز می‌شود که شما چگونه می‌توانید برنامه

[1] در زمان ویرایش این کتاب محیط توسعه Eclipse IDE به دلیل استفاده از Apache Ant منسوخ شده و محیط توسعه ی جدیدی توسط شرکت Google و با نام Android Studio IDE که بر پایه ی Gradle می باشد عرضه شده است. و این علت انتخاب این محیط توسط نویسندگان کتاب می باشد.

کاربردی امن‌تر برای سیستم‌عامل اندروید توسعه دهید.

اندروید سهم قابل‌توجهی از شکست‌های امنیتی و یک لیست روبه رشد از نرم‌افزارهای مخرب دارد که ارزش بررسی و یادگیری امن سازی آن را مشخص می‌کند. مجهز شدن به چگونگی مقابله با جنبه‌های امنیتی، شما را کد نویس بهتر نمی‌کند، اما شما را نسبت به حریم خصوصی و امنیت کاربر نهایی مسئولیت‌پذیر می‌نماید. به‌منظور درک بهتر مفاهیم امنیتی در رابطه با برنامه‌های کاربردی مثال‌های متنوعی ارائه خواهد شد.

۱-۲-۱ محافظت از کاربر

هدف از امنیت برای یک توسعه‌دهنده، حفظ حریم خصوصی کاربر نهایی است. برنامه‌ی کاربردی باید بهترین عملکرد برای محافظت از اطلاعات کاربر فراهم کند. این به معنی فکر کردن در مورد امنیت، قبل از شروع توسعه است. کاربر امکان ندارد همیشه از شیوه‌های امنیتی که در برنامه کاربردی به کار می‌رود آگاه باشد. طراحی و فکر کردن در مورد امنیت قبل از توسعه برنامه کاربردی، می‌تواند شما را از انتقادات بد و خسارت مالی به مشتریان نجات دهد. کاربر نهایی چنین سهل انگاری ها و اشتباهاتی را تقریباً هرگز زود نمی‌بخشد یا فراموش نمی‌کند؛ از این رو ممکن است در استقبال از برنامه هایتان با چالش جدی مواجه شوید. در ادامه، شما اصول و تکنیک‌های شناسایی داده‌های حساس کاربر و ایجاد طرح برای محافظت از این داده‌ها را یاد خواهید گرفت. هدف، از بین بردن هر آسیب ناخواسته برای برنامه کاربردی است.

۱-۲-۲ خطرات امنیتی

به نظر می‌رسد کاربران دستگاه تلفن همراه، خطرات منحصربه‌فردی را نسبت به زمانی که با رایانه رومیزی[1] کار می‌کنند، دارند. در کنار احتمال بالای از دست دادن یا سرقت دستگاه، کاربران دستگاه تلفن همراه درخطر از دست دادن داده‌های حساس و حریم خصوصی خود نیز قرار می‌گیرند. چرا اهمیت این خطرات برای کاربران رایانه‌های رومیزی متفاوت است؟ اول، کیفیت داده‌های ذخیره‌شده روی تلفن همراه، به شخصی بودن بیشتر آن بستگی دارد. در تلفن همراه علاوه بر e-mail و پیام‌های فوری، SMS/MMS، تماس‌ها، عکس‌ها و پست صوتی نیز وجود دارد. شاید برخی از این اطلاعات بر روی رایانه رومیزی نیز قرار داشته باشند ولی این را در نظر بگیرید: کاربر در تمام مدت تلفن را همراه خود حمل می‌کند. گوشی هوشمند یک بستر کامل از تلفن همراه و رایانه رومیزی است که مجموعه‌ای غنی‌تر از اطلاعات شخصی را شامل

[1] Desktop computer

می‌شود.

ازآنجا که کاربر با گوشی هوشمند خود تعامل مداوم و زیادی دارد؛ داده‌های روی گوشی همیشه جدیدتر از داده‌های روی رایانه است. حتی اگر داده های گوشی هوشمند با یک مکان از راه دور یا سرویس ابری به طور بلادرنگ همگام سازی شده باشد؛ این همگام سازی فقط کاربر را در برابر از دست دادن داده‌ها کمک می‌کند و در برابر نقض حریم خصوصی حفاظت نمی‌کند.

در نظر بگیرید فرمت داده‌های ذخیره شده بر روی دستگاه تلفن همراه، ثابت است. هر تلفن همراهی SMS/MMS، تماس‌ها و پست صوتی دارد. اکنون در نظر بگیرید که چگونه همه این اطلاعات برای کاربر نهایی مهم است. برای یک کاربر که پشتیبان گیری ندارد، از دست دادن این اطلاعات از این محیط غیرقابل‌تصور است. از دست دادن شماره تلفن‌های مهم، ویدیوها یا پیام‌های SMS مهم می‌تواند هرروز برای کاربر تلفن همراه مصیبت ایجاد کند.

برای کاربرانی که کار و فعالیت‌های شخصی را بر روی تلفن همراه خود انجام می‌دهند، اگر کسی تمام کلمات عبور ذخیره شده روی تلفن همراه را کپی کند چه باید کرد؟ یا اگر یک ایمیل از اسرار تجاری و قیمت‌گذاری محرمانه برای پیشنهاد، به بیرون درز کند و بر روی اینترنت قرار بگیرد چه باید کرد؟ فرض کنید یک تعقیب گر به این اطلاعات و بیشتر از آن دسترسی دارد، مثلاً، آدرس خانه و شماره تلفن.

زمانی که درباره داده‌های ذخیره شده بر روی تلفن همراه فکر شود؛ در اکثر موارد پی برده می شود که این داده‌ها از خود دستگاه باارزش‌ترند. خطرناک‌ترین نوع حمله، آن است که در سکوت و از راه دور صورت پذیرد. یک مهاجم نیاز به دسترسی فیزیکی به گوشی ندارد. این نوع حملات می‌تواند در هر زمانی اتفاق بیافتد و اغلب به علت ضعف امنیتی در جای دیگر، روی دستگاه اتفاق می‌افتد. این تهدید در امنیت، دلیل بر ناامن بودن برنامه کاربردی نیست. آن می‌تواند ناشی از اشکال در کرنل یا مرورگر وب باشد. سؤال این است: آیا نرم‌افزار شما می‌تواند حتی درزمانی که حمله‌کننده از دسترسی به دستگاه از راه‌های مختلف استفاده می‌کند، داده‌هایش را محافظت نماید.

۳-۱ معماری امنیتی اندروید

همان‌طور که قبلاً بحث کرده‌ایم، اندروید نسخه ٦ روی کرنل لینوکس نسخه ۳,۱۸,۱۰ اجرا می‌شود. ما همچنین یاد گرفتیم که کرنل لینوکسی اندروید، مدیریت امنیت را برای سیستم‌عامل کنترل می‌کند. در ادامه

معماری امنیتی اندروید به‌صورت مختصر بررسی خواهد شد.

۱-۳-۱ جداسازی امتیاز

کرنل اندروید وقتی بخواهد نرم‌افزاری را اجرا کند؛ یک مدل جدا سازی امتیاز را پیاده سازی می‌کند. این یعنی، مانند یک سیستم UNIX، سیستم‌عامل اندروید برای اجرا شدن برنامه کاربردی نیاز به شناسه کاربری و شناسه گروه دارد. بخش‌هایی از معماری سیستم، جدا از این روش عمل می‌کنند. این تضمین می‌کند که برنامه‌های کاربردی یا فرایندها هیچ مجوزی برای دسترسی به دیگر برنامه‌های کاربردی یا فرایندها را ندارند.

جداسازی امتیازی با اعطای امتیاز ویژه[۱] چیست؟

جدا سازی امتیازی یک ویژگی مهم امنیتی است، چون با یکی از معمول‌ترین انواع حملات مقابله می‌کند. در بسیاری از موارد، حمله کننده با اولین حمله‌ی خود نمی‌تواند به تمام اهداف خود بر سد بلکه اولین حمله یک قدم اساسی یا راهی برای یک حمله‌ی بزرگ‌تر است. ابتدا مهاجمان از یک بخش از سیستم، سوءاستفاده خواهند کرد و در حمله بعدی سعی می‌کنند به یک بخش مهم‌تر در سیستم نفوذ کنند. اگر این دو بخش از سیستم دارای امتیازات یکسانی باشند، انجام این حملات و رفتن از یک بخش به بخش دیگر با امتیازات یکسان کار بسیار بی ارزشی برای مهاجمان است. به‌وسیله امتیازات مجزا، وظیفه مهاجمان سخت‌تر می شود. آن‌ها باید بتوانند امتیازات خود را افزایش یا تغییر دهند تا بتوانند به آن بخش‌هایی که می‌خواهند، حمله کنند. در صورتی که نتوانند امتیازات لازم را کسب کنند؛ حمله متوقف می‌شود. برای مثال هنگامی که مهاجم یک حفره ی امنیتی در برنامه ی Gmail تلفن های همراه اندرویدی پیدا می‌کند، باید دسترسی او تنها مربوط به برنامه ی Gmail باشد. زیرا با نفوذ به یک بخش، نفوذگر نباید این اجازه را داشته باشد که به تمامی بخش‌ها نفوذ کند. در اندروید هریک از برنامه ها می‌تواند تنها به داده ها و اطلاعات برنامه ی خود دسترسی پیدا کند و دسترسی آن به اطلاعات سایر برنامه ها بسته شده است.

یکی از ویژگی‌های اصلی طراحی اندروید که در هسته پیاده سازی شده، جدا سازی امتیازی است. فلسفه‌ی پشت این طراحی این است که اطمینان حاصل شود که هیچ نرم‌افزاری نمی‌تواند کد یا داده‌های

[۱] Privilege separation

برنامه‌های کاربردی دیگر، اطلاعات کاربر یا سیستم‌عامل را بخواند و تغییر دهد. بنابراین، یک برنامه‌ی کاربردی نباید به صورت خود سرانه قادر به استفاده از شبکه دستگاه برای اتصال به سرور از راه دور باشد. یک برنامه‌ی کاربردی نباید به‌طور مستقیم از لیست تماس‌ها یا تقویم بخواند. این ویژگی به‌عنوان sand boxing نیز شناخته‌شده است. بعدازاین که دو فرایند در sand box های خود اجرا شدند، درخواست صریح اجازه دسترسی به داده‌های یکدیگر، تنها راهی است که می‌توانند با یکدیگر ارتباط برقرار کنند. مفهوم sand boxing اینگونه تعریف می شود که فرض کنید در ساحل یک دریا در حال ساخت قلعه های شنی هستید. طبیعتاً دور هر قلعه دیواری می‌کشید تا از دیگر قلعه‌ها جدا باشد. در اندروید نیز با توجه به ویژگی sand boxing هر برنامه تنها به داده ها و اطلاعات برنامه ی خود دسترسی دارد.

۲-۳-۱ مجوزها[1]

اجازه دهید یک مثال ساده بزنیم، ما یک نرم‌افزار داریم که صوت ضبط می‌کند. برای اینکه برنامه‌ی کاربردی به‌درستی کار کند، توسعه‌دهنده باید مطمئناً یک درخواست مجوز ضبط‌صوت در فایل برنامه کاربردی AndroidManifest.xml[2] اضافه کند.

این به نرم‌افزار اجازه می‌دهد تا درخواست مجوز را به بخشی از سیستم ارسال کند که میکروفون را کنترل می‌کند. اما چه کسی درخواست را رد یا قبول می‌کند؟ اندروید به کاربر نهایی امکان می‌دهد تا زمانی که کاربر، برنامه کاربردی را نصب می‌کند این تصمیم نهایی را بگیرد. همانند آنچه در تصویر ۱-۳ مشاهده می‌نمایید. پس از API 23 یا اندروید ٦، گونه ای از مجوزها تحت عنوان dangerous permission[3] تعریف شده اند که در لحظه ی اجرا، مجوزها را از کاربر نهایی دریافت می‌کنند.اگر توسعه‌دهنده صراحتاً نیاز خود را برای مجوز ضبط صوت تنظیم نکند، یا اگر صاحب دستگاه نخواهد بعد از درخواست به برنامه اجازه بدهد، بنابراین یک خطا توسط ماشین‌مجازی رخ خواهد داد که برنامه کاربردی شکست خواهد خورد. این وظیفه‌ی یک توسعه‌دهنده است که درخواست اجازه را شناسایی کند و خطای تولید شده را کنترل نماید. مثلا برای درخواست مجوز ذکر شده باید تگ زیر در فایل پروژه AndroidManifest.xml قرار بگیرد:

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

[1] Permissions

[2] فایلی که تمامی مجوزهای یک برنامه باید در آن ذکر شود.

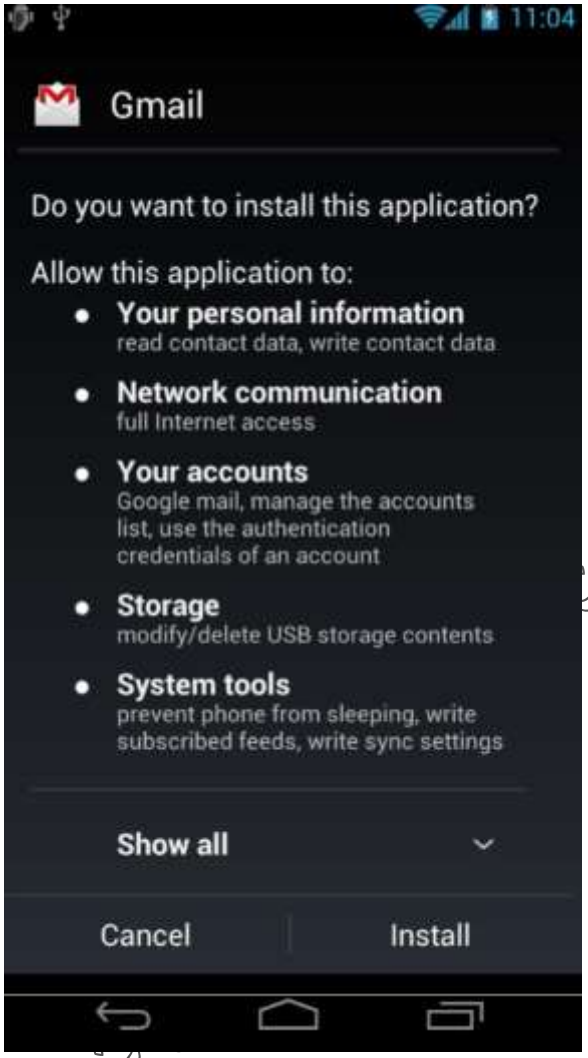
[3] تعدادی از مجوزها می باشند که باید در لحظه ی اجرای کد برنامه نویس از کاربر گرفته شود تا برنامه ها با توجه به سهل انگاری کاربر اقدام به شنود اطلاعات نکنند. مجوزهای خطرناک در آدرس developer.**android**.com/training/**permissions**/requesting.html مشخص شده‌اند.

در صورتی که برنامه ی خود را برای اندروید API 23 به بعد توسعه می‌دهید باید یک سری از مجوزها که با نام dangerous permissions شناخته می‌شوند را در کدهای جاوا معرفی کنید تا کاربر نهایی به صورت runtime اجازه ی دسترسی به مجوزها را بدهد. در ابتدا مجوزی که می‌خواهیم در لحظه ی اجرای عملیات از کاربر دریافت شود را مشخص می‌کنیم، سپس از کاربر نهایی سوال می‌نماییم که آیا اجازه ی دسترسی به بخش ضبط صدا را به برنامه می‌دهد یا خیر. در صورت موافقت کاربر نهایی، کدهای برنامه نویس اجرا می‌شود.

```
if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M) {
    int permissionCheck =
checkSelfPermission(Manifest.permission.RECORD_AUDIO);
    if (permissionCheck != PackageManager.PERMISSION_GRANTED) {
        if (shouldShowRequestPermissionRationale(
                Manifest.permission.RECORD_AUDIO)) {
        } else {
            requestPermissions(new String[]{Manifest.permission.RECORD_AUDIO},
                    CALLBACK_NUMBER);
        }
    } else {
        // got permission
    }
}
```

لیست کامل مجوزها در پایگاه وب Android Developer[1] موجود است.

[1] https://developer.android.com/reference/android/Manifest.permission.html



تصویر ۱-۳ صفحه درخواست مجوزهای اندروید

خلاصه

همان‌طور که تاکنون ذکر شد، اندروید به لطف توجه گوگل، پیشرفت فوق‌العاده‌ای داشته است. این مراقبت و توجه، کمک شایانی به رشد سریع اندروید نسبت به سایر سیستم‌عامل‌های هوشمند کرده است، همچنین دلیل دیگر این نرخ رشد، سازگاری با سخت‌افزارهای متنوع [۱] تلفن‌های همراه است. همچنین دیدیم هسته‌ی اصلی اندروید بر روی هسته‌ی لینوکس است. هسته به‌عنوان پلی بین سخت‌افزار و سیستم‌عامل خدمت می‌کند و همچنین کنترل امنیت، مدیریت حافظه، مدیریت فرایند و شبکه را انجام می‌دهد؛ این دو از وظایف اصلی هسته سیستم‌عامل است. معمولاً زمانی که تولیدکنندگان سخت‌افزار شروع به نصب سیستم‌عامل اندروید برای کار با یک سخت‌افزار متفاوت می‌کنند، هسته یکی از اجزای اصلی‌ای است که

[۱] استفاده از سخت افزارهای متفاوت باعث شده است تا هنگام طراحی یک برنامه ی اندروید، زمان بیشتری صرف واکنش طراحی در یک تلفن همراه ۳ اینچی و یک تبلت ۱۰ اینچی شود.

تغییر داده می‌شود.

لایه بعدی که در اطراف هسته اندروید حرکت می‌کند، لایه زمان اجراست که شامل کتابخانه‌های هسته و ماشین‌مجازی Dalvik است. ماشین‌مجازی Dalvik یک بخش اساسی از اجرای برنامه‌های کاربردی روی بستر اندروید است. ماشین‌مجازی Dalvik توانایی اجرای برنامه‌های کاربردی به صورت ایمن و کارآمد در یک محیط با منابع محدود را داراست. این توانایی نیازمند ویژگی‌هایی منحصربه‌فرد است.

لایه‌های بعدی که باید به معماری اضافه شوند به ترتیب لایه چارچوب[1] و لایه برنامه کاربردی هستند. شما می‌توانید از لایه چارچوب به‌عنوان پلی دیگر بین API جاوا و کد اصلی و فرایندهای سیستم که در لایه‌های زیرین در حال اجرایند، استفاده کنید. لایه چارچوب جایی است که تمام رابط‌های برنامه کاربردی اندروید در آن قرار دارند. هر کتابخانه‌ای که شما مایل به استفاده از آن در برنامه‌تان هستید، باید از قبل در چارچوب واردشده باشد. برنامه‌های کاربردی نهایت در لایه برنامه‌های کاربردی قرار می‌گیرند. شما می‌توانید این فضا را با دیگر توسعه‌دهندگان برنامه‌های کاربردی و برنامه‌های کاربردی سیستم مثل تلفن همراه، تقویم، ایمیل و پیام به اشتراک بگذارید.

پس‌ازآن به‌اختصار به خطرات امنیتی پرداخته شد و اینکه چگونه می‌توان مسئولیت امنیت کاربر نهایی را بر عهده داشت. همچنین برخی از راه‌هایی که اندروید برای امنیت ارائه داده است بررسی گردید. به طور کلی در این خصوص، جداسازی امتیازی و مجوز مورد توجه قرار گرفت. در فصل بعد درباره چگونگی استفاده از این ویژگی‌ها توضیحاتی ارایه خواهد شد.

[1] Framework

۲ اطلاعات: اساس یک برنامه

اساس تمام برنامه‌های کاربردی معنی‌دار، اطلاعات است و برنامه‌های کاربردی برای تغییر، ایجاد و ذخیره‌سازی اطلاعات طراحی و ایجاد می‌شوند. برنامه‌های کاربردی موبایل متفاوت نیستند. برای توضیح این نکته، تصور کنید یک تلفن اندروید بدون شبکه تلفن همراه یا پوشش وای فای باشد. بنابراین دسترسی به برخی از برنامه‌های کاربردی ممکن نیست؛ برای مثال، ایمیل، پیام، مرورگر وب، و هر برنامه‌ی کاربردی دیگر که نیاز به اینترنت دارد، غیرقابل استفاده می‌شود.

در فصل‌های بعد، اطلاعات مربوط به مختصات مکانی بررسی و بر روی اینکه چگونه آن‌ها را باید حفظ کرد، تمرکز می شود. این فصل، به اعمالی که در هنگام ذخیره سازی اطلاعات اتفاق می‌افتد اختصاص دارد.

۲-۱ امنیت برنامه کاربردی در برابر حملات

هنگامی‌که داده‌ها ایجاد یا دریافت می‌شوند نیاز دارند تا در جایی ذخیره شوند[۱]. در ادامه درباره چگونگیِ ذخیره اطلاعات و درنهایت اینکه چگونه می‌توان اطلاعات برنامه را امن نمود، توضیح داده می‌شود. انتشار برنامه کاربردی برای عموم مردم باید با همان احتیاط که برای راه‌اندازی یک وب‌سایت بر روی اینترنت انجام می شود، صورت پذیرد. باید فرض شود که برنامه کاربردی در برخی از زمان‌ها به‌طور مستقیم و غیرمستقیم موردحمله قرار می‌گیرد و تنها چیزی که بین حفظ حریم خصوصی کاربر نهایی و حفاظت داده‌ها قرار می‌گیرد، برنامه کاربردی است.

۲-۱-۱ حملات غیرمستقیم[۲]

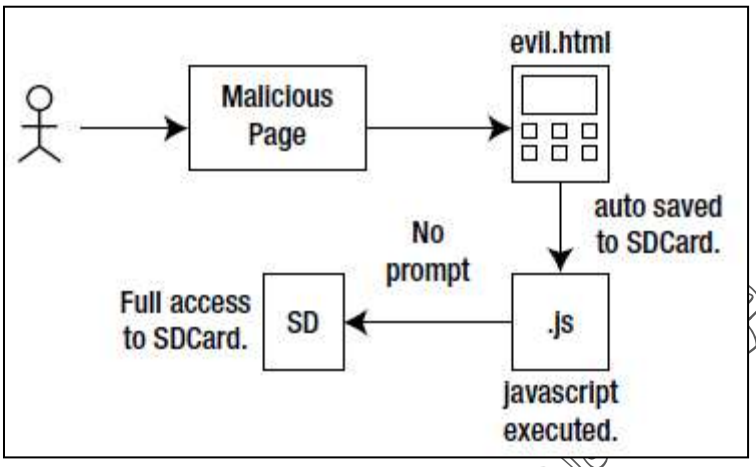
در اواخر سال ۲۰۱۰ ، دو آسیب‌پذیری[۳] به ترتیب در نسخه‌های ۲,۲ و ۲,۳ اندروید کشف شد. این

[۱] Data Storage

[۲] Indirect Attacks

[۳] Vulnerability

آسیب‌پذیری همانی است که در آن‌یک مهاجم می‌تواند هر فایلی که بر روی کارت SD [1]دستگاه شما ذخیره‌شده را بدون اجازه و یا حتی یک نشانه قابل‌مشاهده، کپی کند. این آسیب‌پذیری همان‌طور که در تصویر ۲-۱ نشان داده‌شده، کار می‌کند.



تصویر ۲-۱ حمله غیرمستقیم

موارد زیر نکات قابل‌تأمل هستند:

۱. یک کاربر از وب‌سایت مخرب که میزبان فایلی مانند ***evil.html*** **است** بازدید می‌کند.

۲. با توجه به یک بخش از آسیب‌پذیری، فایل ***evil.html*** بدون اطلاع کاربر، دانلود شده و در کارت SD دستگاه ذخیره می‌شود.

۳. با توجه به بخش دیگر آسیب‌پذیری، فایل ذخیره‌شده می‌تواند به‌محض ذخیره شدن یک کد جاوا اسکریپت اجرا کند. بنابراین امنیت کاربر نهایی دچار مخاطره می‌شود.

٤. با توجه به بخش آخر آسیب‌پذیری، کد جاوا اسکریپت اجراشده مرحله قبلی، به خاطر اینکه به صورت محلی[2] روی د ستگاه اجرا شده، د ستر سی کامل به بارگذاری فایل‌های ذخیره شده روی کارت SD را به یک وب‌سایت به انتخاب مهاجم را خواهد داشت.

برای اثبات، فرض کنید که برنامه کاربردی، تمام اطلاعات ذخیره‌شده را برای ذخیره‌سازی بحث دایرکتوری خود را بر روی کارت SD بنویسد. چون آ سیب‌پذیری مورد بحث ا ست، اطلاعات ا ستفاده شده توسط برنامه کاربردی درخطر دزدیده شدن است. هر دستگاه اندروید که برنامه و نسخه نرم‌افزار آسیب‌پذیر

[1] SD Card

[2] Local

را اجرا کند، یک نوع خطر سرقت اطلاعات برای کاربر نهاییِ آن به شمار می‌آید. این‌یک نمونه‌ی حمله‌ی غیرمستقیم بر روی برنامه کاربردی است.

آسیب‌پذیری برنامه کاربردی به یک حملهٔ غیرمستقیم تا حد زیادی بستگی به این دارد که تا چه اندازه به معماری و جنبه‌های امنیتی برنامه قبل از شروع به نوشتن کد توجه شده باشد.

۲-۱-۲ حمله‌های مستقیم[۱]

حملات مستقیم به‌طور قابل‌توجهی متفاوت هستند و می‌توانند اشکال مختلفی داشته باشند. در یک حمله مستقیم، یک برنامه کاربردی می‌تواند هدف قرار داده شود. بنابراین، مهاجم در جستجوی نقاط ضعف در نرم‌افزار است تا بتواند اطلاعات حساس کاربران برنامه کاربردی را جمع‌آوری کند و یا به سرور مرتبط با برنامه کاربردی حمله کند. برای مثال، در رابطه با برنامه کاربردی تلفن‌بانک؛ یک مهاجم ممکن است به برنامه کاربردی تلفن همراهی که متعلق به یک بانک خاص است، وارد شود. اگر طراحی برنامه کاربردی ضعیف باشد – برای مثال، اگر اطلاعات حساس کاربر به‌صورت آشکار[۲] ذخیره‌شده باشد، یا ارتباط بین برنامه کاربردی و سرور توسط SSL[۳] امن نشده باشد، مهاجم می‌تواند حملات خاصی را دنبال کند که هدفش سوءاستفاده از این ضعف‌ها است. این‌یک حمله مستقیم بر روی یک برنامه خاص است. اطلاعات بیشتر در مورد حملات مستقیم در فصل‌های بعد بحث خواهد شد.

۲-۲ پروژه ۱: ذخیره‌سازی داده

پروژه Proxim یک برنامه کاربردی است که می‌تواند SMS را به‌صورت خاص ارسال کند، اطلاعات تماس در زمانی که یک کاربر در مجاورت خاص با مجموعه‌ای از مختصات GPS است تعریف می‌شود. برای مثال، با این برنامه کاربردی، یک کاربر می‌تواند همسر خود را به‌عنوان یک مخاطب اضافه کند. بنابراین، هرگاه همسرش در سه کیلومتری از محل کار ویا خانه باشد برنامه sms برای کاربر فعال می‌شود. به‌این‌ترتیب، همسرش می‌داند که او نزدیک به خانه و یا محل کارش است. روال ذخیره داده در کد۲-۱ نشان داده‌شده است.

---

[۱] Direct Attacks

[۲] Clear text

[۳] Secure Socket Layer

SSL یک استاندارد امنیتی برای ارتباط امن بین دو طرف ارتباط است. در این ارتباط امن تمامی پیام ها با استفاده از روش های امنیتی به صورت رمزنگاری شده منتقل می‌شوند.

کد ۲-۱ روال ذخیره، SaveController.java

```
package android.secure.programming;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import android.secure.programming.Contact;
import android.secure.programming.Location;
import android.content.Context;
import android.os.Environment;
import android.util.Log;
public class SaveController {
    private static final String TAG = "SaveController";
    public static void saveContact(Context context, Contact contact) {
        if (isReadWrite()) {
            try {
                File outputFile = new
File(context.getExternalFilesDir(null),contact.getFirstName());
                FileOutputStream outputStream = new
FileOutputStream(outputFile);
                outputStream.write(contact.getBytes());
                outputStream.close();
            } catch (FileNotFoundException e) {
                Log.e(TAG,"File not found");
            } catch (IOException e) {
                Log.e(TAG,"IO Exception");
            }
        } else {
            Log.e(TAG,"Error opening media card in read/write mode!");
        }
    }
    public static void saveLocation(Context context, Location location) {
        if (isReadWrite()) {
            try {
                File outputFile = new
File(context.getExternalFilesDir(null),location.getIdentifier());
                FileOutputStream outputStream = new
FileOutputStream(outputFile);
                outputStream.write(location.getBytes());
                outputStream.close();
            } catch (FileNotFoundException e) {
                Log.e(TAG,"File not found");
            } catch (IOException e) {
                Log.e(TAG,"IO Exception");
            }
        } else {
            Log.e(TAG,"Error opening media card in read/write mode!");
        }
    }
    private static boolean isReadOnly() {
        Log.e(TAG,Environment.getExternalStorageState());
        return Environment.MEDIA_MOUNTED_READ_ONLY.equals(Environment
        .getExternalStorageState());
    }
    private static boolean isReadWrite() {
        Log.e(TAG,Environment.getExternalStorageState());
        return Environment.MEDIA_MOUNTED.equals(Environment
        .getExternalStorageState());
    }
```

```
}
```

هرزمانی که یک کاربر کلید ذخیره موقعیت یا کلید ذخیره تماس را انتخاب کند، کد بالا راه‌اندازی می شود. کلاس‌های موقعیت در کد ۲-۲ و تماس در کد ۲-۳ با جزئیات بیشتر ارائه شده است. البته می‌توان یک‌روال ذخیره‌سازی اصلی پیاده‌سازی کرد ولی به دلیل توضیح بهتر جنبه‌های امنیتی به این شیوه پیاده‌سازی شده است.

کد ۲-۲ کلاس موقعیت، Location.java

```
package android.secure.programming;

public class Location {
    private String identifier;
    private double latitude;
    private double longitude;

    public double getLatitude() {
        return latitude;
    }
    public void setLatitude(double latitude) {
        this.latitude = latitude;
    }
    public double getLongitude() {
        return longitude;
    }
    public void setLongitude(double longitude) {
        this.longitude = longitude;
    }
    public void setIdentifier(String identifier) {
        this.identifier = identifier;
    }
    public String getIdentifier() {
        return identifier;
    }
    public String toString() {
        StringBuilder ret = new StringBuilder();
        ret.append(getIdentifier());
        ret.append(String.valueOf(getLatitude()));
        ret.append(String.valueOf(getLongitude()));
        return ret.toString();
    }
    Public byte[] getBytes() {
        return toString().getBytes();
    }
}
```

کد ۲-۳ کلاس تماس، Contact.java

```
package android.secure.programming;
public class Contact {
```

```
    private String firstName;
    private String lastName;
    private String address1;
    private String address2;
    private String email;
    private String phone;

    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getAddress1() {
        return address1;
    }
    public void setAddress1(String address1) {
        this.address1 = address1;
    }
    public String getAddress2() {
        return address2;
    }
    public void setAddress2(String address2) {
        this.address2 = address2;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getPhone() {
        return phone;
    }
    public void setPhone(String phone) {
        this.phone = phone;
    }
    public String toString() {
        StringBuilder ret = new StringBuilder();
        ret.append(getFirstName() + "|");
        ret.append(getLastName() + "|");
        ret.append(getAddress1() + "|");
        ret.append(getAddress2() + "|");
        ret.append(getEmail() + "|");
        ret.append(getPhone() + "|");
        return ret.toString();
    }
    public byte[] getBytes() {
```

```
        return toString().getBytes();
    }
}
```

کلاس‌های موقعیت و تماس، کلاس‌های استانداردی هستند که برای نگه‌داری داده‌های خاص به هر نوع طراحی‌شده‌اند. هرکدام از آن‌ها شامل متدهای ()toString و ()getBytes می باشند که تمام محتویات کلاس را به‌صورت یک رشته یا آرایه‌ای از بایت‌ها برمی‌گرداند.

به‌منظور اضافه کردن یک شیء Contact به‌صورت دستی باید از کدی شبیه به آنچه در کد ۲-٤ آمده است، بهره برد.

کد ۲-٤ کد اضافه نمودن شیء Contact جدید

```
final Contact contact = new Contact();
contact.setFirstName("Android");
contact.setLastName("Programming");
contact.setAddress1("");
contact.setAddress2("");
contact.setEmail("android@android.com");
contact.setPhone("12120031337");
```

لحظه‌ای را فرض کنید، که یک کاربر اطلاعات یک مخاطب جدید را به برنامه کاربردی اضافه می‌کند، کد ۲-٤ فراخوانی می شود. البته می توان به جای استفاده از مقادیر تعریف شده، از متد ()getText برای خواندن مقادیر از اشیای EditText که درview اصلی تعریف شده‌اند، استفاده کرد. اگر شما کد SaveController.saveContact(getApplicationContext(),contact)) را در شبیه‌ساز اندروید اجرا کنید، SaveController مخاطب جدید را می‌گیرد و در منبع رسانه‌های خارجی ذخیره می‌کند. (به کد ۲-۱ مراجعه کنید).

نکته: استفاده از متد () getExternalStorageDirectory برای پیدا کردن موقعیت کارت SD بر روی دستگاه اندروید پیشنهاد می‌شود. چرا که اندروید می‌تواند بر روی تعداد زیادی از دستگاه‌ها با مشخصات متفاوت اجرا شود. موقعیت دایرکتوری کارت SD ممکن است همیشه در sdcard/ نباشد. متد ()getExternalStorageDirectory از سیستم‌عامل، موقعیت کارت SD را درخواست می‌کند و آن موقعیت را به عنوان خروجی برمی‌گرداند.

برای بسط بهتر مطالب اجازه دهید با متد سازنده SaveContact شروع کنیم:

```
public static void saveContact(Context context, Contact contact) {
```

```
if (isReadWrite()) {
    try {
```

قطعه کد ذکر شده در بالا منتظر یک شیء Context و یک شیء contact است. هر برنامه کاربردی روی اندروید Context خودش را دارد. یک شیء Context، کلاس‌های خاص، متدها و منابع را نگه می‌دارد که می‌تواند در میان کلاس‌ها در یک برنامه کاربردی به اشتراک گذاشته شود. برای مثال، یک شیء Context شامل اطلاعات موقعیت مسیر کارت SD است. برای دسترسی به آن، باید متد Context.getExternalFilesDir() فراخوانی شود. بعدازآن که متد مقادیر موردنظر را دریافت کرد؛ با متد isReadWrite() بررسی می‌کند که کارت SD روی دستگاه نصب و قابل‌نوشتن هست یا نه؟

```
File outputFile = new
File(context.getExternalFilesDir(null),contact.getFirstName());
```

این کد یک شیء فایل ایجاد می‌کند که به موقعیت دایرکتوری کارت SD اشاره می‌کند وازنام شیء Context به‌عنوان اسم فایل استفاده می‌شود.

```
FileOutputStream outputStream = new FileOutputStream(outputFile);
outputStream.write(contact.getBytes());
outputStream.close();
```

با استفاده از این کد، یک FileOutputStream ایجاد می‌شود که به محل شیء فایل اشاره می‌کند. سپس، محتویات شیء Contact با استفاده از متد getByte() به جریان خروجی نوشته می‌شود تا آرایه‌ای از بایت‌ها را برگرداند.

هنگامی‌که اجرا کامل شد، باید یک فایل با نام "android" در مسیر Android/data/android.secure.programming/files[1] بر کارت SD وجود داشته باشد.

اگر فایل با یک ویرایشگر متن باز شود، می‌توان داده‌های متنی ساده را که توسط برنامه کاربردی در آن نوشته‌شده است، مشاهده کرد.

```
Android|Programming|||android@android.com|12120031337
```

تصویر ۲-۲ محتویات شیء Contact

۲-۳ طبقه‌بندی اطلاعات

برخی اوقات برنامه‌نویسان برای ایجاد و توسعه یک برنامه کاربردی از کدهای آماده استفاده می‌کنند و

[1] باتوجه به نحوه ی پکیج بندی ممکن است این فایل در مسیر دیگری در تلفن همراه ذخیره شود اما مسیر کلی آن به صورتی که اشاره شد، است.

سعی می‌کنند تا این کدها را متناسب با آنچه می‌خواهند تغییر دهند که بر زمان پروژه و خروجی کار اثرات مخربی دارد و همچنین تأثیر منفی بر امنیت برنامه کاربردی خواهد دا شت. اما زمانی که برنامه‌نویسان سعی می‌کنند یک خلا صه مختصر از پروژه را بنویسند موجب می شود تا جلوتر از زمان فکر کنند. هرچند که به نظر می‌رسد این نکته واضحی است؛ ولی تو سعه‌دهندگان زیادی وجود دارند که موفق به انجام این مرحله ساده نشده‌اند. یکی دیگر از مواردی که باید دقت شود، توجه به اطلاعات یا داده‌هایی است که توسط برنامه کنترل خواهد شد. برای مثال، جدول ۲-۱ برخی از انواع داده‌هایی که توسط یک برنامه کنترل خواهد شد، را شامل می‌شود. این جدول بسیار ساده است؛ بااین‌حال، می‌توان یک نمای کلی از انواع داده‌هایی داشت که توسط یک برنامه کنترل می‌شود و همچنین می‌توان یک طرح برای امنیت این اطلاعات تنظیم نمود.

جدول ۲-۱ جدول طبقه‌بندی داده

Data Type	Personal?	Sensitive?	Create	Store	Send	Receive
Name	Yes	No	X	X	x	
E-mail Address	Yes	Yes	X	X	x	
Phone No.	Yes	Yes	X	X		
Address	Yes	Yes	X	X		

اگر با دقت بیشتر به جدول طبقه‌بندی داده در جدول ۲-۱ توجه شود، انتزاعی بودن بعضی از عنوان‌ها نمایان می‌شود. افراد مختلف، نظرات متفاوتی روی داده حساس و شخصی دارند. بااین‌حال، بهتر است که از یک چارچوب مرجع معمولی شروع کرد تا بتوان نوع داده حساس یا شخصی را تعیین نمود. این جدول می‌تواند شامل این اطلاعات باشد:

- انواع داده[1] : این داده در برنامه استفاده و تحلیل می‌شود.
- شخصی: این ستون نشان می‌دهد که آیا این نوع داده، یک داده شخصی است؟
- حساس: این ستون نشان می‌دهد که آیا این نوع داده، یک داده حساس است؟
- ایجاد: آیا برنامه کاربردی به کاربر اجازه می‌دهد که این نوع داده را ایجاد کند؟
- ذخیره: آیا برنامه کاربردی این نوع داده را روی دستگاه یا یک سرور راه دور ذخیره می‌کند یا خیر؟
- ارسال: آیا این نوع داده از طریق شبکه به سروری دیگر یا برنامه دیگر ارسال می‌شود؟
- دریافت: آیا این نوع داده از طریق شبکه از دیگران دریافت می‌شود؟

[1] Data Type

۲-۳-۱ اطلاعات شخصی چیست؟

اطلاعات شخصی می‌توانند به‌عنوان داده‌هایی دسته‌بندی شوند که شما و تعداد محدودی از افراد مرتبط با شما، آن را می‌دانند. اطلاعات شخصی معمولاً چیزهای خصوصی شما هست، اما شما مایل به اشتراک آن‌ها تنها با دوستان نزدیک و اعضای خانواده هستید. برای مثال اطلاعات شخصی شما می‌تواند شامل شماره تلفن، آدرس منزل و آدرس ایمیل باشد. ذخیره این اطلاعات ممکن است موجب فاش شدن آن‌ها شود ولی معمولاً موجب آسیب مادی و معنوی نمی‌شود.

۲-۳-۲ اطلاعات حساس چیست؟

اطلاعات حساس ارزشِ بیشتری نسبت به اطلاعات شخصی دارند. اطلاعات حساس معمولاً اطلاعاتی هستند که شما تحت هیچ شرایطی آن‌ها را با کسی به اشتراک نمی‌گذارید. داده‌های از این نوع شامل رمز عبور، گواهی بانکداری اینترنتی(مثل pin code)، شماره موبایل، شماره امنیت اجتماعی، یا آدرس است. اگر اطلاعات حساس در معرض خطر قرار داده شوند، اثرات آن ممکن است موجب آسیب‌های جسمی و عاطفی شود. این اطلاعات باید همیشه محافظت شوند، صرف‌نظر از اینکه آیا ذخیره‌شده‌اند یا خیر.

۲-٤ تجزیه‌وتحلیل کد

اگر به حمله‌ی غیرمستقیم که در فصل قبل درباره آن صحبت شد توجه شود، واضح است که داده‌ای که به‌صورت آشکار بر روی کارت SD قرارگرفته، یک خطر مهم است و باید به هر قیمتی از خطر دوری کرد. سرقت یا در معرض خطر قرار دادن داده یکی از دلایل عمده ضرر مالی و اعتباری شرکت‌ها بوده است. در پروژه Proxim ، ضعفِ ذخیره داده آشکار وجود دارد. هرکس که به کارت SD دستگاه دسترسی داشته باشد قادر به کپی کردن اطلاعات شخصی خواهد بود؛ مثل نام‌ها، آدرس‌ها، شماره تلفن‌ها و آدرس‌های ایمیل. برای ذخیره این اطلاعات می‌توان از رمزنگاری استفاده نمود. در ادامه نحوه استفاده از رمزنگاری در پروژه proxim توضیح داده می‌شود. در فصل‌های آینده در رابطه با زیرساخت رمزنگاری بیشتر بحث خواهد شد. بنابراین، هدف از این تمرین آن است که با مثال‌های بسیار ساده از رمزنگاری متقارن[۱]، رمزنگاری توسط روش AES[۲] توضیح داده شود. در رمزنگاری متقارن تنها یک کلید به عنوان کلید مشترک[۳] بین دو طرف

[۱] Symmetric Encryption

[۲] Advanced Encryption Standard

[۳] Shared Key

ارتباط وجود دارد. رمزنگاری با کلید عمومی یا رمزنگاری نامتقارن[۱] یکی دیگر از روش‌های رمزنگاری یا مبهم کردن داده‌ها با ا ستفاده از دو نوع متفاوت از کلیدها ا ست. هر کاربر دو کلید دارد، یک کلید عمومی[۲] و یک کلید خصوصی[۳]. کلید خصوصی، تنها داده‌هایی را می‌تواند رمزگشایی کند که توسط کلید عمومی رمزگذاری شده با شد و کلید عمومی، تنها داده هایی را می تواند رمزگشایی کند که تو سط کلید خصو صی رمزگذاری شده با شد[٤]. کلید عمومی، کلیدی ا ست که آزادانه در اختیار دیگران قرارگرفته می شود و کاربران دیگر از آن برای رمزنگاری داده‌ها استفاده خواهند کرد اما کلید خصوصی نزد هر کاربر مخفی می باشد.

۲-٤-۱ رمزگذاری

با توجه به مباحث مطرح شده باید اطلاعات را قبل از ذخیره شدن در کارت SD رمزنگاری نمود. یک مهاجم که داده‌های رمزنگاری‌شده را جمع‌آوری کرده است؛ برای دسترسی به داده‌ها مجبور است از یک رمز عبور برای رمزگشایی داده‌ها استفاده کند. برای رمزنگاری داده‌ها می‌توان از روش AES استفاده نمود؛ این الگوریتم متن را با یک کلید مشترک که نزد گیرنده و فرستنده مخفی است، رمز می‌کند. یک کلید برای رمزنگاری و رمزگشایی موردنیاز است. این روش به‌عنوان رمزنگاری کلید متقارن شناخته می‌شود. این کلید باید به‌صورت ایمن ذخیره شود و در صورت فاش شدن کلید تمامی اطلاعات برای افراد غیرمجازی که دارای کلید باشند قابل شنود[۵] است. کد ۲-۵ روال عادی رمزنگاری را نشان می‌دهد.

کد ۲-۵ روال عادی رمزنگاری

```
private static byte[] encrypt(byte[] key, byte[] data){
    SecretKeySpec sKeySpec = new SecretKeySpec(key,"AES");
    Cipher cipher;
    byte[] ciphertext = null;
    try {
        cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, sKeySpec);
        ciphertext = cipher.doFinal(data);
    } catch (NoSuchAlgorithmException e) {
        Log.e(TAG,"NoSuchAlgorithmException");
    } catch (NoSuchPaddingException e) {
        Log.e(TAG,"NoSuchPaddingException");
    } catch (IllegalBlockSizeException e) {
        Log.e(TAG,"IllegalBlockSizeException");
```

[۱] Asymmetric Encryption

[۲] Public key

[۳] Private key

[٤] عمدتا از این روش به منظور احراز هویت استفاده میشود.

[۵] Sniff

```
    } catch (BadPaddingException e) {
        Log.e(TAG,"BadPaddingException");
    } catch (InvalidKeyException e) {
        Log.e(TAG,"InvalidKeyException");
    }
    return ciphertext;
}
```

در قسمت اول کد، برای آماده‌سازی تولید کلید امنیتی AES، کلاس SecretKeySpec مقداردهی شده است و یک متغیر جدید از کلاس Cipher تعریف‌شده است.

```
SecretKeySpec sKeySpec = new SecretKeySpec(key,"AES");
Cipher cipher;
byte[] ciphertext = null;
```

در این کد همچنین یک آرایه از بیت‌ها برای ذخیرهٔ Cipher مقداردهی اولیه شده است. قسمت بعدی کد، کلاس Cipher را برای استفاده از الگوریتم AES آماده می‌کند:

```
cipher = Cipher.getInstance("AES");
cipher.init(Cipher.ENCRYPT_MODE, sKeySpec);
```

متد cipher.init() شیء cipher را مقداردهی می‌کند، بنابراین می‌تواند با استفاده از کلید امنیتی ایجادشده، رمزنگاری را انجام دهد. در خط بعدی کد، داده‌ی متن آشکار رمزنگاری می‌شود و محتوای رمزنگاری‌شده را در آرایه بیتی ciphertext ذخیره می‌کند.

```
ciphertext = cipher.doFinal(data);
```

مهم است که از کلید یکسان برای روال رمزنگاری استفاده شود زیرا در حال استفاده از روش رمزنگاری متقارن می‌باشیم. و بهتر است که مولد کلید با تابعی نوشته شود که کلید را بر اساس عدد تصادفی تولید کند؛ زیرا حدس زدن یک رمز غیرمعمولی برای یک مهاجم سخت‌تر است. برای نمونه، از الگوریتم تولید کلید نشان داده‌شده در کد ۲-۶ استفاده شده است.

کد ۲-۶ الگوریتم تولید کلید

```
public static byte[] generateKey(byte[] randomNumberSeed) {
    SecretKey sKey = null;
    try {
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
        random.setSeed(randomNumberSeed);
        keyGen.init(256,random);
        sKey = keyGen.generateKey();
    } catch (NoSuchAlgorithmException e) {
        Log.e(TAG,"No such algorithm exception");
    }
    return sKey.getEncoded();
}
```

در کد بالا، دو خط کد زیر، کلاس KeyGenerator را مقداردهی اولیه و کلید خاص AES را تولید کرده است. سپس دستگاه مولد اعداد تصادفی، اعداد تصادفی را تولید خواهد نمود:

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES");
SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
```

این اعداد تصادفی با استفاده از روش SHA1 کدگذاری شده‌اند. SHA1[1] یک تابع درهم‌سازی رمزنگاری است. این الگوریتم روی قطعه‌ای از اطلاعات عمل می‌کند که دارای یک طول دلخواه است و یک‌رشته‌ی کوتاه با طول ثابت را تولید خواهد کرد. با استفاده از مقایسه‌ی مقادیر درهم سازی شده می‌توان دریافت که آیا اطلاعات دستکاری شده است یا خیر و به این منظور از صحت متن[2] رمز شده اطمینان حاصل نمود. قطعه کد زیر از عدد تصادفی استفاده می‌کند که یک کلید ۲۵۶ بیتی را با استفاده از عدد تصادفی تولید می‌کند:

```
random.setSeed(randomNumberSeed);
keyGen.init(256,random);
sKey = keyGen.generateKey();
```

الگوریتم تولید کلید را اجرا نمایید و کلیدهای تولیدشده را برای استفاده در روال عادی رمزگشایی ذخیره کنید.

۲-٤-۲ نتایج رمزنگاری

زمانی که یک شیء Contact در کارت SD بررسی می‌شود، به نظر می‌رسد که محتویات آن به‌هم‌ریخته‌اند (تصویر ۲-۳) و برای هر حمله‌کننده‌ای **غیرقابل**خواندن است.

1java2s.c$jHYfTgR90gb9RKb2GewSr0

تصویر ۲-۳ تماس‌های رمزنگاری‌شده‌ی شیء Contact

۲-۵ پروژه اصلاح‌شده

تغییرات روی پروژه‌ی Proxim عمدتاً بر روی متد saveController() است (کد ۲-۷ را ببینید).

[1] Secure Hash Algorithm 1

[2] Message Integrity

کد ۲-۷ متد SaveController.java

```
package android.secure.programming;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;
import net.zenconsult.android.crypto.Crypto;
import net.zenconsult.android.model.Contact;
import net.zenconsult.android.model.Location;
import android.content.Context;
import android.os.Environment;
import android.util.Log;
public class SaveController {
    private static final String TAG = "SaveController";
    public static void saveContact(Context context, Contact contact) {
        if (isReadWrite()) {
            try {
                File outputFile = new File(context.getExternalFilesDir⊃
                (null),contact.getFirstName());
                FileOutputStream outputStream = new FileOutputStream⊃
                (outputFile);
                byte[] key =
Crypto.generateKey("randomtext".getBytes());
                outputStream.write(encrypt(key,contact.getBytes()));
                outputStream.close();
            } catch (FileNotFoundException e) {
                Log.e(TAG,"File not found");
            } catch (IOException e) {
                Log.e(TAG,"IO Exception");
            }
        } else {
            Log.e(TAG,"Error opening media card in read/write mode!");
        }
    }
    public static void saveLocation(Context context, Location location)
{
        if (isReadWrite()) {
            try {
                File outputFile = new File(context.getExternalFilesDir⊃
                (null),location.getIdentifier());
                FileOutputStream outputStream = new FileOutputStream⊃
                (outputFile);
                byte[] key =
Crypto.generateKey("randomtext".getBytes());
```

```
                outputStream.write(encrypt(key,location.getBytes()));
                outputStream.close();
            } catch (FileNotFoundException e) {
                Log.e(TAG,"File not found");
            } catch (IOException e) {
                Log.e(TAG,"IO Exception");
            }
        } else {
            Log.e(TAG,"Error opening media card in read/write mode!");
        }
    }
    private static boolean isReadOnly() {
        Log.e(TAG,Environment
        .getExternalStorageState());
        return Environment.MEDIA_MOUNTED_READ_ONLY.equals(Environment
        .getExternalStorageState());
    }
    private static boolean isReadWrite() {
        Log.e(TAG,Environment
        .getExternalStorageState());
        return Environment.MEDIA_MOUNTED.equals(Environment
        .getExternalStorageState());
    }
    private static byte[] encrypt(byte[] key, byte[] data){
        SecretKeySpec sKeySpec = new SecretKeySpec(key,"AES");
        Cipher cipher;
        byte[] ciphertext = null;
        try {
            cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.ENCRYPT_MODE, sKeySpec);
            ciphertext = cipher.doFinal(data);
        } catch (NoSuchAlgorithmException e) {
            Log.e(TAG,"NoSuchAlgorithmException");
        } catch (NoSuchPaddingException e) {
            Log.e(TAG,"NoSuchPaddingException");
        } catch (IllegalBlockSizeException e) {
            Log.e(TAG,"IllegalBlockSizeException");
        } catch (BadPaddingException e) {
            Log.e(TAG,"BadPaddingException");
        } catch (InvalidKeyException e) {
            Log.e(TAG,"InvalidKeyException");
        }
        return ciphertext;
    }
}
```

خلاصه

برنامه‌نویس باید از ذخیره‌سازی متن ساده[1] یا داده‌های خواندنی روی دستگاه تلفن همراه اجتناب کند.

[1] Plain Text

اگرچه برنامه‌ی کاربردی می‌تواند خودش امن نوشته شود؛ بااین‌حال یک حمله‌ی غیرمستقیم که از یک جای کاملاً متفاوت آغاز شده، می‌تواند اطلاعات حساس یا شخصی نوشته شده به‌وسیله آن برنامه کاربردی را جمع‌آوری کند و بخواند. یک برنامه‌نویس باید موارد زیر را در گام‌های اساسی در طی طراحی برنامه کاربردی دنبال کند:

۱. اول، تعیین کند چه نوع داده‌هایی به‌وسیله برنامه کاربردی ذخیره، ایجاد، یا تغییر داده می‌شود. بعد آن‌ها را به داده‌های شخصی یا حساس طبقه‌بندی کند. بنابراین فهمیده می‌شود که چگونه باید با داده‌هایی که در برنامه‌ی کاربردی هستند، رفتار کرد. در صورت استفاده ی غیرضروری از روش‌های رمزنگاری سرعت عملکرد برنامه کاربردی به دلیل محاسبات کاهش می‌یابد.

۲. مجموعه روال‌هایی برای رمزنگاری داشته باشد تا بتواند در برنامه‌های کاربردی استفاده مجدد کند. بهتر است این مجموعه به‌عنوان کتابخانه‌ای جدا در نظر گرفته شود که می‌تواند در پروژه قرار گیرد.

۳. برای هر برنامه‌ی کاربردی یک کلید متفاوت تولید شود. یک الگوریتم تولید کلید خوب نوشته شود که کلیدهای رمزی طولانی و غیرقابل‌پیش‌بینی را تولید کند. قدرتمندتر بودن کلید منجر به طولانی شدن فرآیند رمزگشایی و شکستن رمز برای مهاجمین می‌شود.

٤. داده‌های حساس باید در هنگام ایجاد یا ذخیره‌سازی رمزنگاری شود.

۳ معماری امن اندروید[۱]

در فصل دوم مثال‌هایی از چگونگی استفاده از رمزنگاری در حفاظت از اطلاعات مطرح شد. اما این مثال‌ها در مورد معماری امن اندروید و امنیت داخلی اندروید نبود. در این بخش به بررسی امکاناتی که اندروید برای حفاظت از اطلاعات در اختیار برنامه‌نویس‌ها قرار می‌دهد، می‌پردازیم. همچنین حملات مستقیمی که بر روی برنامه‌ها رخ می‌دهد و چگونگی جلوگیری از آن‌ها برای مراقبت بیشتر از اطلاعات شخصی افراد، مورد توجه قرار خواهد گرفت.

محیط اندروید مکانیسم‌هایی کارا برای کنترل امنیت سیستم و برنامه‌ها داراست. هر فرایند در اندروید با مجموعه‌ای از امتیازات و مجوزهای[۲] مخصوص به خود اجرا می‌شود. بنابراین سایر برنامه‌ها نمی‌توانند بدون مجوز از طرف کاربر به این برنامه یا اطلاعات آن دسترسی داشته باشند.

اندروید APIهای زیادی را در اختیار برنامه‌نویس قرار می‌دهد اما برای استفاده برنامه از همه آن‌ها نیاز است تا کاربر نهایی[۳] این دسترسی‌ها را تائید نماید.

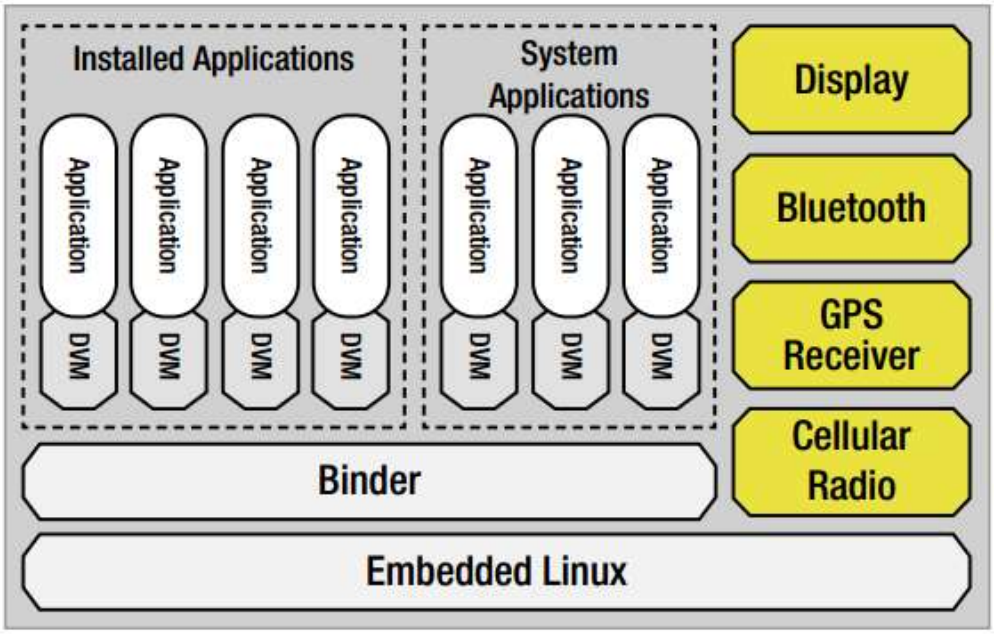
۳-۱ بازنگری معماری سیستم

در اینجا دوباره به معماری اندروید نگاه می‌شود. معماری سیستم اندروید به‌طور کامل در بخش ۱ پوشش داده شد. در حالت عادی هیچ تعاملی بین برنامه‌های مختلف روی یک دستگاه ممکن نیست مگر اینکه کاربر مجوز آن را داده باشد.

[۱] Android Secure Architecture

[۲] Permission

[۳] End User



تصویر ۳-۱ معماری اندروید با تمرکز بر برنامه‌ها

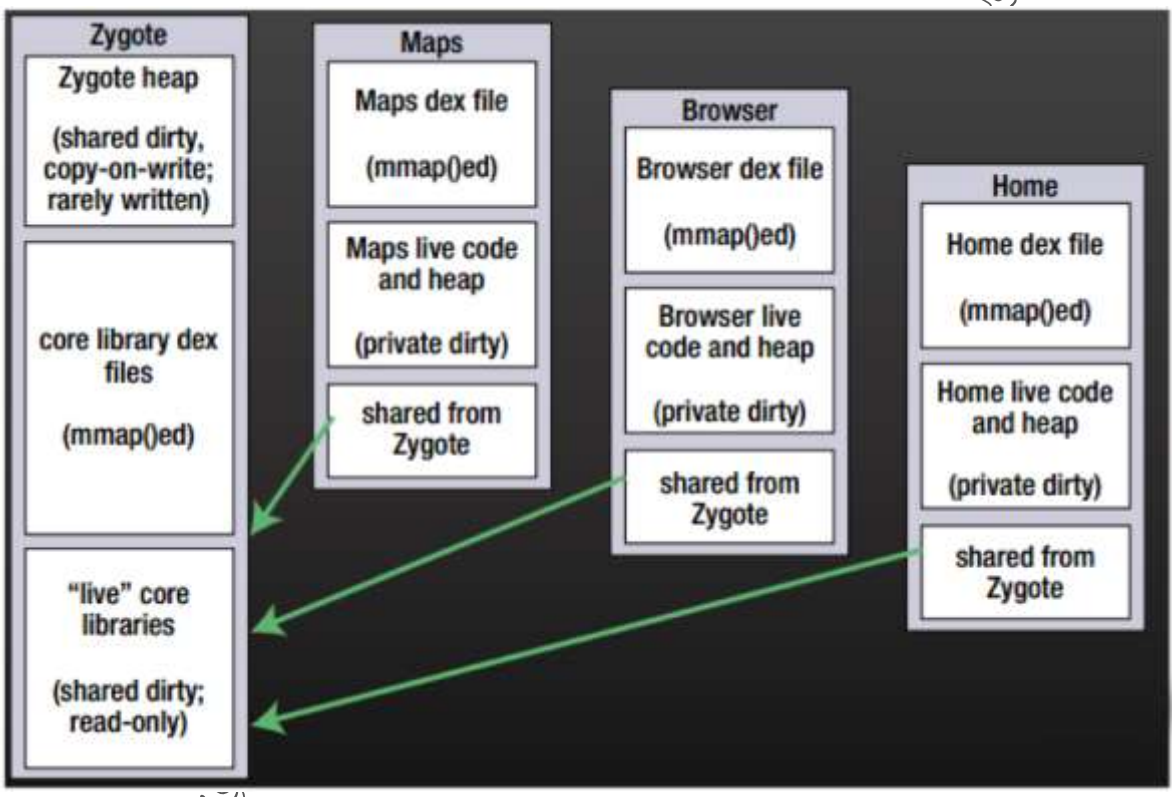
تصویر ۳-۱ نمونه ساده‌تری از معماری اندروید را به نمایش می‌گذارد که از نمونه ارائه شده در فصل ۲ ساده‌تر است. این تصویر بیشتر بر روی خود برنامه تمرکز دارد. همان‌طور که مشاهده می شود برنامه‌های اندروید بر روی ماشین‌مجازی [۱] Dalvik اجرا می‌شوند. جایی که اساسی‌ترین بلوک‌های کد بر روی آن اجرا می‌شوند، مشابه ماشین‌مجازی جاوا هست که امروزه بر روی رایانه‌های شخصی و سرورها موجود است. هر برنامه به‌تنهایی در ماشین‌مجازی Dalvik اجرا می‌شود، به‌بیان‌دیگر هر برنامه‌ای بدون تعامل خارجی در طول برنامه دیگر اجرا می‌شود[۲] مگر مواردی که مجوز داشته باشند. زمانی که ماشین‌های مجازی مستقل شروع به کار کنند، موجب می شوند زمان بیشتری مصرف شود و در راه‌اندازی برنامه‌ها از زمان شروع، تأخیر ایجاد کنند؛ اندروید به کمک یک مکانیسم قبل از بارگذاری برنامه، تصمیم به افزایش سرعت می‌گیرد. این مکانیسم Zygote نام دارد که دو عمل اساسی انجام می‌دهد: اول، در مرحله اول به‌عنوان یک lunch pad برای برنامه‌های جدید کار می‌کند و دوم، در ادامه به‌عنوان یک کتابخانه مرجع که همه برنامه‌ها در طول چرخه فعالیتشان می‌توانند به آن مراجعه کنند کار می‌کند.

فرایند zygote تلاش می‌کند تا نمونه‌ای از یک ماشین‌مجازی را راه‌اندازی کند و تمام کلاس‌های کتابخانه‌ای که ماشین‌مجازی به آن‌ها احتیاج دارد را بارگذاری و مقداردهی اولیه می‌نماید و سپس منتظر می‌ماند تا سیگنال راه‌اندازی یک برنامه را دریافت کند. zygote در زمان بوت، راه‌اندازی می‌شود و مشابه یک

[۱] Dalvik virtual machine

[۲] ویژگی sand boxing که در فصل اول بررسی شد.

صف کار می‌کند. هر دستگاه اندروید یک پروسه اصلی zygote در حال اجرا دارد. زمانی که مدیر فعالیت‌های اندروید یک فرمان تحت عنوان شروع یک برنامه را آغاز می‌کند، نمونه‌ای از یک ما شین‌مجازی را که جزئی از zygote اســـت را صـــدا می‌زند. زمانی که نمونه ای از zygote به یک برنامه اختصـــاص داده می‌شـــود، بلافاصله نمونه[1] ی دیگری از آن ساخته می شود تا جای خالی نمونه ی قبلی را پر کند. برنامه های جدید از نمونه ی جدیدی که تولید شده است استفاده می‌کنند و این روند ادامه می‌یابد. چرا که مخزن zygote همیشه مجموعه ای از کتابخانه‌های هسته را برای برنامه ها، در طول چرخه زندگی[2] شان آماده می‌کند. تصویر ۳-۲ نمایانگر این مطلب است.



تصویر ۳-۲ نحوه استفاده برنامه‌ها از مخزن کتابخانه‌های هسته zygote

۳-۲ درک معماری مجوزها

همان‌طور که در فصل اول اشاره شد، برنامه‌ها با مجموعه کاربران و گروه‌های هویتی خودشان بر روی سیستم‌عامل اندروید اجرا می‌شوند و باتوجه به ویژگی sand boxing هر برنامه تنها به داده ها و اطلاعات برنامه ی خود دسترسی دارد. در راستای تسهیل اشتراک اطلاعات و انجام ارتباطات در طول برنامه‌ها،

[1] Instance

[2] Life Cycle

اندروید از سیستم مجوزها استفاده می‌کند.

در حالت پیش‌فرض هیچ برنامه‌ای مجوزی برای انجام فعالیت‌هایی که موجب آسیب به برنامه‌های دیگر می‌شود را ندارد. همین‌طور هیچ قابلیتی برای تعامل با سیستم‌عامل یا استفاده از APIهای حفاظت‌شده برای استفاده از دوربین، GPS یا پشته‌های شبکه را دارا نیست. نهایتاً یک برنامه هرگز اجازه ندارد تا اطلاعات یک کاربر را بخواند یا در آن بنویسد. این وظیفه را هسته لینوکس انجام می‌دهد.

یک برنامه در صورت نیاز به دسترسی به APIهای ویژه یا داده‌های کاربر لازم است تا از کاربر نهایی اجازه این اعمال را بگیرد/ به‌عنوان یک برنامه‌نویس، لازم است تا قبل از انتشار برنامه خود برای عموم، بدانید که برنامه شما به چه مجوزهایی احتیاج دارد. باید همه آن‌ها را لیست کنید و به فایل AndroidManifest.xml اضافه نمایید و همچنین در گدهای جاوا درخواست مجوز را به کاربر بدهید. پس زمانی که برای اولین بار، برنامه توسط کاربر نصب می‌شود از او در مورد پذیرفتن یا رد کردن مجوزها سؤال می‌شود. بهتر است برنامه به گونه‌ای تنظیم شود که اگر کاربر یکی از مجوزهای اساسی را رد کرد برنامه بازهم نصب شود. فرض کنید شما یک برنامه نوشته‌اید که به استفاده از GPS و دسترسی به داده‌های کاربر و ارسال پیام کوتاه نیاز دارد. کاربر به دو مورد از موارد گفته‌شده مجوز می‌دهد و فرستادن پیامک را رد می‌کند. باید برنامه طوری باشد که بتواند بدون ارسال پیامک کار کند. بدین شکل برنامه‌های مختلف می‌توانند با کارایی کمتر نصب شوند.

قبل از کاوش بیشتر در مورد مجوزها، باید با دو موضوع آشنایی پیدا کنید، که در مفهوم برنامه‌نویسی اندروید و امنیت به کار می‌آیند. ارائه‌دهندگان محتوا و intent ها، این عبارات را باید قبلاً شنیده باشید اما بگذارید تا مطمئن شویم که درک درستی از آن‌ها دارید.

۱-۲-۳ ارائه‌دهندگان محتوا[۱]

ارائه‌دهندگان محتوا تقریباً همان ذخیره‌کنندگان داده‌ها هستند. آن‌ها به‌عنوان مخازن اطلاعات برنامه‌هایی که می‌توانند بنویسند یا بخوانند عمل می‌کنند. از زمانی که معماری اندروید، ذخیره‌سازی مشترک در یک مکان را منع کرد، ارائه‌دهندگان محتوا تنها راهی هستند که برنامه‌ها می‌توانند از طریق آن‌ها به تبادل داده بپردازند. شاید برای شما جذاب باشد که می‌توانید ارائه‌دهندگان محتوای خودتان را داشته باشید. بدین ترتیب دیگر برنامه‌ها خواهند توانست که به اطلاعات شما دسترسی داشته باشند.علاوه بر امکان ایجاد ارائه‌دهندگان محتوا برای خودتان، اندروید این امکان را فراهم کرده تا به ارائه‌دهندگان دیگر نیز دسترسی داشته باشید که انواع

[۱] Content Provider

رایج داده‌ها را در دسترس قرار می‌دهند. مانند عکس‌ها، فیلم‌ها و فایل‌های صوتی و اطلاعات مخاطبین.

تأمین‌کننده بسته‌های اندروید android.provider شامل کلاس‌های زیادی است که این امکان را به شما می‌دهد تا به ارائه‌دهندگان محتوای زیادی دسترسی داشته باشید.

برای استفاده از ارائه‌دهندگان محتوا به دانش قبلی درباره موارد زیر نیاز است:

- اشیاء ارائه‌دهنده محتوا (فیلم، تصویر، مخاطبین و غیره).
- ستون‌های موردنیاز این ارائه‌دهنده محتوا.
- درخواست برای گرفتن اطلاعات.

همان‌طور که گفته شد، یک ارائه‌دهنده محتوا همچون یک پایگاه داده رابطه‌ای مانند Oracle، Microsoft SQL Server و MySQL رفتار می‌کند.برای مثال شما به ارائه‌دهنده محتوای MediaStore.Images.Media دسترسی دارید تا تصاویری را درخواست دهید. تصور کنید می‌خواهید به اسم هرکدام از عکس‌های ذخیره‌شده بر روی دستگاه کاربر دسترسی داشته باشید. در ابتدا لازم است تا یک ارائه‌دهنده محتوای URI ایجاد کنید.

```
Uri images = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
```

در گام دوم باید یک گیرنده برای داده‌ها ایجاد کنید. آرایه‌ها بصورت داده، این کار را انجام می‌دهند.

```
String[] details = new String[] {MediaStore.MediaColumns.DISPLAY_NAME};
```

در ادامه به یک اشاره‌گر نیاز است:

```
Cursor cur = managedQuery(details,details, null, null null);
```

از متد cur.moveToFirst() بدین شکل برای حرکت در طول ردیف اول و خواندن اسم‌ها استفاده می‌شود:

```
String name =
cur.getString(cur.getColumnIndex(MediaStore.MediaColumns.DISPLAY_NAME));
```

بعدازاین اشاره‌گر را با فراخوانی cur.moveToNext() به رکورد بعدی می‌برید.

به این موضوع توجه کنید که بعضی ارائه‌دهندگان محتوا تحت کنترل هستند و برنامه شما قبل از دسترسی به آن‌ها به مجوزهای خاصی نیاز دارد.

۳-۲-۲ Intent

intent ها نوعی پیام هستندکه یک برنامه به برنامه‌ای دیگر به منظور کنترل وظیفه‌ها یا انتقال داده‌ها

می‌فرستد. intent ها با سه مؤلفه از برنامه‌ها کار می‌کنند: فعالیت[1] ، سرویس[2] و گیرنده پخشی [3]. intent action و intent data از اجزای اصلی یک شیء intent بشمار می‌روند.

در ادامه یک مثال ساده بررسی خواهد شد، در این مثال از کاربر خواسته می‌شود که یک سایت را در مرورگر مشاهده نماید بنابراین از ثابت Intent.ACTION_VIEW به عنوان آرگومان اول همراه با مقدار داده‌ای یک URL (مانند: http://www.google.com.) به عنوان آرگومان دوم استفاده می‌شود.

```
Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse(http://www.google.com));
```

فراخوانی این intent به صورت زیر انجام می‌پذیرد:

```
startActivity(intent);
```

برای کنترل این‌که کدام برنامه intentها را دریافت می‌کنند باید یک مجوز به intent قبلی اضافه شود تا آن را هدایت کند.

۳-۳ بررسی مجوزها

ارائه‌دهندگان محتوا و intent ها در قسمت قبل توضیح داده شد. بدین‌صورت که متوجه شدیم چگونه سیستم‌عامل اندروید با استفاده از مجوزهای دسترسی این موارد را کنترل می‌کند. در قسمت اول دیدیم که چگونه یک برنامه می‌تواند از کاربر مجوزهای خاصی را درخواست کند تا با سیستم تعامل داشته باشد. حال توجه کنید که کنترل مجوزها چگونه و کجا رخ می‌دهد.

زمانی که یک برنامه API خاصی را درخواست می‌کند، یک مکانیسم معتبر کنترل می‌کند که آیا برنامه مجوزهای موردنیاز را برای ادامه کار دارد یا خیر. اگر کاربر مجوز را بدهد این درخواست کامل می‌شود در غیر این صورت یک استثنا امنیتی [4]رخ می‌دهد.

فراخوانی APIها در سه مرحله انجام می‌پذیرد: در ابتدا کتابخانه مربوطه فراخوانی می‌شود سپس proxy interface کتابخانه را فراخوانی می‌کند که جزئی از خود کتابخانه API است و درنهایت proxy interface از یک ارتباط بین پردازشی برای تکمیل فراخوانی استفاده می‌کند.

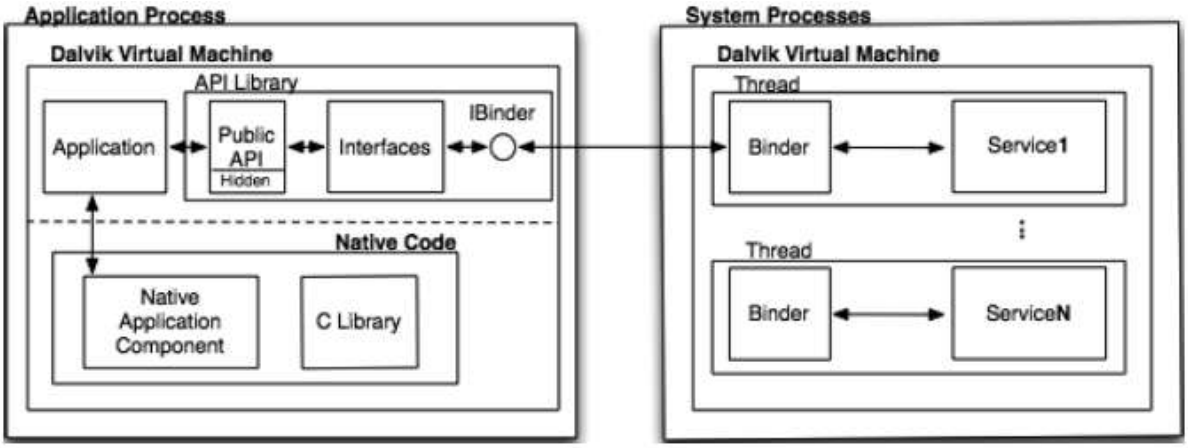
---

[1] Activity

[2] Service

[3] Broadcast Receiver

[4] Security Exception



تصویر ۳-۳ فرایند فراخوانی API

۳-۳-۱ استفاده از مجوزهای سفارشی

اگر یک برنامه نوشته باشید که دسترسی به قابلیت‌های زیادی را برای دیگر برنامه‌نویس‌ها ایجاد کند، شما می‌توانید از این قابلیت‌ها با تعریف مجوزهای سفارشی حفاظت کنید. اندروید به برنامه‌نویس این امکان را می‌دهد تا مجوزهای خودش را بسازد. بدین منظور باید تگ‌ها و مشخصه‌های خاصی در androidmanifest.xml تعریف شود:

کد ۳-۱: androidmanifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package=" android.secure.programming" >
    <permission
android:name="android.secure.mobile.testapp.permission.PURGE_DATABASE"
        android:label="@string/label_purgeDatabase"
        android:description="@string/description_purgeDatabase"
        android:protectionLevel="dangerous" />
        ...
 </manifest>
```

ابتدا باید نام مجوز در android:name تعریف شود. android:label و android:description که اشاره‌گر رشته‌ای هستند نیز در فایل AndroidManifest.xml تعریف شده‌اند. این رشته‌ها مشخص می‌کنند که این مجوز برای چیست و درنهایت چه‌کاری انجام می‌دهد. این رشته‌ها به‌صورت توصیفی همانند زیر تعیین می‌شوند.

```
<string name=" label_purgeDatabase ">purge the application database
</string>
```

```
<string name="permdesc_callPhone">Allows the application to purge the
core database of the information store. Malicious applications may be
able to wipe your entire application
information store.</string>
```

همچنین مشخصه android:protectionLevel برای تعیین سطح حفاظتی مجوز، مورد نیاز است.

اگر تمایل دارید مجوز شما در گروه‌های د ستگاهی یا گروهی که خودتان تعریف کرده‌اید، قرار گیرد با استفاده از android:permissionGroup این عمل میسر خواهد بود. گروه کردن یک مجوز سفارشی با یک گروه مجوزهای موجود بهترین راه است تا مجوزها به‌صورت گروهی به کاربر نشان داده شود تا بتواند راحت‌تر مجوزها را مرور نماید. برای مثال برای اضافه کردن مجوز purgeDatabase به گروهی که به SD card دسترسی دارد باید مشخصه مزبور به فایل AndroidManifest.xml در بخش تعریف مجوز اضافه گردد.

```
android:permissionGroup="android.permission-group.STORAGE"
```

نکته قابل‌توجه این ا ست که باید قبل از هرگونه برنامه وابسته‌ای، برنامه شما بر روی د ستگاه نصب شود. این مورد معمولی ابتدا ممکن است قابل‌تحمل باشد اما بعد از مدتی در حین پیاده‌سازی حتماً با مشکل مواجه خواهید شد.

در اندروید نسخه ٦ مفهوم جدیدی به نام مجوزهای لحظه اجرا[1] فراهم شده ا ست. مجوزهای لحظه اجرا تعدادی از مجوزهای خطرناک هستند که در صورت عدم آگاهی کاربر می‌تواند منجر به شنود اطلاعات شود. به همین منظور اینگونه از مجوزها علاوه بر لحظه‌ی نصب برنامه کاربردی، لحظه اجرای عملیات نیز نیاز به تایید کاربر نهایی دارند. برای درک بهتر مجوزهای لحظه اجرا فرض کنید یک برنامه ی کاربردی نیاز دسترسی به کارت SD کاربر را دارد. قبل از اینکه کدهای مربوط به دسترسی کارت SD کاربر اجرا شود باید مجوز د ستر سی از کاربر نهایی گرفته شود. در صورتی که قصد اجرای برنامه مزبور در اندروید نسخه ٦ به بالا را دارید همانند آنچه در فصل های قبل بیان شد باید مجوزهای خود را به صورت کد ۳-۲ بنویسید.

کد ۳-۲: MainActivity.java

```
int permissionCheck = checkSelfPermission(this,  android.permission-
group.STORAGE);
if (!permissionCheck == PackageManager.PERMISSION_GRANTED) {

    if (shouldShowRequestPermissionRationale(thisActivity,
android.permission-group.STORAGE)) {

    } else {
```

[1] Runtime Permission

```
        requestPermissions(thisActivity,  new String[]{
android.permission-group.STORAGE }, CALLBACK_NUMBER);

    }
} else {

}
```

۳-۳-۲ سطوح محافظت

یک مجوز با مشخصه سطح حفاظت خطرناک یا بالاتر به‌طور خودکار برنامه را وادار می‌کند تا برای کاربر پیام هشدار نمایش دهد. در واقع به کاربر درباره یک خطر احتمالی آگاهی می‌دهد. زمانی که مجوز خاص خود را می‌سازید، می‌توانید آن را وابسته به میزان امنیتی که برای سیستم‌عامل به وجود می‌آورد، دسته‌بندی کنید. فهرست سطوح مختلف حفاظت را در جدول ۳-۱ خواهید دید:

جدول ۳-۱ سطوح حفاظت مجوزها

Constant	Value	Description
normal	0	A somewhat low-risk permission that gives an application access to isolated application-level features, with minimal risk to other applications, the system, or the user. The system automatically grants this type of permission to a requesting application at installation, without asking for the user's explicit approval (though the user always has the option to review these permissions before installing).
dangerous	1	A higher risk permission that gives a requesting application access to private user data or control over the device in a way that can negatively impact the user. Because this type of permission introduces potential risk, the system may not automatically grant it to the requesting application. Any dangerous permissions requested by an application may be displayed to the user and require confirmation before proceeding, or some other approach may be taken so the user can avoid automatically allowing the use of such facilities.
signature	2	The system will grant this permission only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval.
signatureOrSystem	3	The system grants this permission only to packages in the Android system image or that are signed with the same certificates. Please avoid using this option because the signature protection level should be sufficient for most needs, and it works regardless of exactly where applications are installed. This permission is used for certain special situations where multiple vendors have applications built into a system image, and these applications need to share specific features explicitly because they are being built together.

خلاصه:

در این فصل نگاهی بر مجوزهای استاندارد و تعریف شده توسط کاربر داشتیم. همین‌طور intentها و ارائه‌دهندگان محتوا موردبررسی قرار گرفت، نکات مهمی که به دست آمد شامل موارد زیر است:

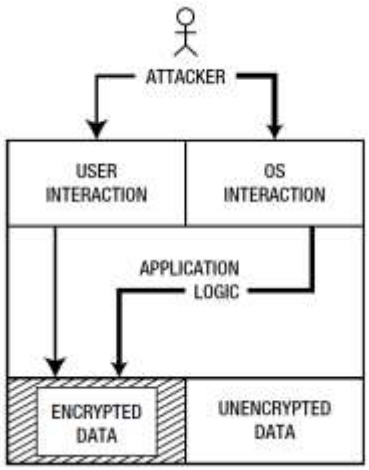
- اندروید شامل مجموعه‌ای از مکانیسم‌هایی است که جداسازی برنامه‌ها از یکدیگر و حفاظت از آن‌ها را بر عهده دارد.
- هر برنامه در فضای جداگانه خود با کاربر یکتا و مجموعه‌ای از شناسه‌ها اجرا می‌شود.
- برنامه‌ها به‌هیچ‌عنوان بدون اجازه کاربر، مجوز تبادل داده‌ها را باهم ندارند.
- ارائه‌دهندگان محتوا، ذخیره و دسترسی به داده‌ها را امکانپذیر می‌نمایند، آن‌ها تقریباً شبیه به پایگاه داده‌ها هستند.
- Intent ها پیام‌هایی هستند که بین برنامه‌ها برای فراخوانی یا قطع یک سرویس یا برنامه دیگر ردوبدل می‌شوند.
- دسترسی به APIهای خاص با استفاده از مجوزها کنترل می‌شود. مجوزها به چهار گروه تقسیم می‌شوند که مجوزهای سطح یک، دو و سه همیشه به کاربر هشدار می‌دهند تا زمانی که این مجوزها احتمال تأثیر مخرب بر اطلاعات کاربر را دارند برای تائید نهایی به کاربر اعلان می‌شوند.

مجوزهای سفارشی برای محافظت از منابع برنامه‌های مستقل ایجاد می‌شود. برنامه شما برای استفاده از منابع آن برنامه مستقل باید به‌طور صریح مجوز موردنظر را با استفاده از تگ <uses-permission> در manifest درخواست دهد.

٤ ذخیره‌سازی داده‌ها و رمزنگاری

در فصل ۳ رمزنگاری بازگشتی متقارن و نامتقارن به‌طور مختصر بررسی شد. فصل ٤ تمرکز، بیشتر بر روی اهمیت اســـتفاده از رمزنگاری به‌منظور مبهم و ایمن کردن داده‌هایی که کاربر ذخیره خواهد کرد یا انتقال خواهد داد، است. بدین منظور، نخست اساس رمزنگاری و چگونگی اعمال آن بررسی می‌شود. سپس، روش‌های مختلف ذخیره‌ســازی داده روی پلتفرم اندروید ذکر می‌گردد، همچنین در رابطه با چگونگی ذخیره و بازیابی داده‌ها نمونه‌هایی بیان می شود.

رمزنگاری مو ضوع بسیاری و سیعی هـست. لذا یک تو سعه‌دهنده برنامه کاربردی شاید علاقه‌مندی و فعالیت در یکی از زمینه‌های خاص رمزنگاری برایش مقدور با شد. نکته مهمی که باید همواره به خاطر سپرد آن ا ست که باید همواره داده‌های کاربری حسا سرا برای کاربران غیرمجازناخوانا نمود. اگر یک مهاجم با یک حمله م ستقیم یا غیرم ستقیم به برنامه کاربردی حمله کند، آنگاه لایه ا ضافی رمزنگاری (تصویر ٤-۱) اجازه دسترسی به داده‌های کاربری حساس را به مهاجم نمی‌دهد. مهاجم باید ابتدا از لایه اضافی عبور کند تا حمله موفقی دا شته با شد. این قوانین مشابه ا صول تضمین اطلاعات دفاع در عمق[1] ا ست که آژانس امینت ملی در ایالات‌متحده توسعه داده است.



تصویر ٤-۱: مثالی از دفاع در عمق

[1] Defense in Depth

٤-١ پیدایش کلید عمومی

ازآنجایی‌که بحث رمزنگاری مطرح اســـت؛ کمی یادگیری در رابطه با پیدایش کلید عمومی (PKI[1]) ارز شمند ا ست. PKI بر ا ساس ا صول شنا سایی و تائید اطمینان مبتنی بر شخص ثالث مورد اعتماد هـ ست. اجازه دهید یک سناریو که اصول را شرح می‌دهد را بیازماییم. در نظر داشته باشید که این مثال برای مدتی با توسعه برنامه کاربردی کاری ندارد. به‌زودی موضوع عمیقاً بررسی خواهد شد.

آقای اکبر، صاحب رستوارن اکبر جوجه، یکی از مشهورترین رستوران‌های غذا است. او مشهورترین اکبر جوجه (یکی از خوشمزه‌ترین غذاها) را طبخ می‌نماید. هیچ‌کس به‌جز آقای کامران دســتورالعمل فوق سری این غذای خوشمزه را نمی‌داند. با توجه به شهرتش، او اخیراً امتیاز رستورانش را می‌فروشد. همان‌طور که بیشــتر زیرشــاخه‌های تحت امتیاز او ازنظر جغرافیایی دور می‌باشــند، آقای کامران تصــمیم دارد که راز دســتورالعمل خود را با پیک به آن‌ها انتقال دهد. تنها مشــکل این روش این اســت که رقیب آقای اکبر، آقای اصغر، اخیراً سعی داشت که راز این دستورالعمل را بیابد و احتمالاً دوباره تلاش خواهد کرد.

ما تصــمیم به بازگشــایی یک رســتوران اکبرجوجه در شــهر خود را گرفته‌ایم. با آقای اکبر تماس می‌گیریم و همراه با برگه مربوطه، او سندی را انتقال می‌دهد که شامل چگونگی دریافت و حفظ کردن راز د ستورالعمل اکبر جوجه است. در اینجا دستورالعمل‌هایی که انجام می‌دهیم به‌این‌صورت خواهد بود:

- در نزدیک‌ترین اداره پلیس که از برنامه اکبر جوجه پیشتیبانی می‌کند در بخش ثبت ر ستوران، ثبت نام کنید.
- یک قفل با یک کلید دریافت کنید که قفل دریافت شده از بخش ثبت رستوراناجازه ی ورود به اداره پلیس را می‌دهد.
- قفل را به اداره پلیس تحویل بدهید.
- از کلید محافظت کنید.
- جعبه فلزی که با پیک فرستاده می‌شود را دریافت و با کلیدی که در دست دارید بازکنید.

مطمئناً زمانی که این مراحل تکمیل شد، یک بسته با پست می‌ر سد. به طرز عجیبی، قسمت خارجی مقوای

[1] Public Key Infrastructure

بسته دستکاری‌شده است، اما قفل یا جعبه فلزی درون آن دست‌کاری نشده است. کلید، قفل را به‌راحتی باز می‌کند. راز دستورالعمل اکبر جوجه به دست می‌آید. پس از مدتی، از آقای اکبر شنیده میشود که آقای اصغر سعی بر دزدی ماشین پست و باز کردن جعبه فلزی را داشته اما موفق نشده است و این نشان می‌دهد که مقوای بیرونی دستکاری‌شده است.

آقای اکبر و ما: به ترتیب فرستنده و گیرنده هستند، ما نیازمند مبادله داده‌های حساس (دستورالعمل سری) هستیم و سیاست‌های کلید عمومیو روش‌هایی برای انجام آن را دنبال می‌کنیم.

آقای اصغر: او مهاجم است. او می‌خواهد به داده‌های حساس دست یابد و تصمیم می‌گیرد هنگام انتقال به آن حمله کند.

جعبه فلزی: این پیام رمزنگاری‌شده است. فرستنده آن را رمزنگاری می‌کند یا آن را قفل می‌کند به‌گونه‌ای که تنها دارنده کلید می‌تواند آن را باز کند. نگه‌دارنده کلید دریافت‌کننده است.

قفل: این کلید عمومی است. زمانی که داستان را بررسی می‌کنید، ممکن است تعجب کنید که چگونه یک قفل می‌تواند کلید هم باشد، اما به‌صورت مجازی به آن نگاه کنید. قفل چیزی است که هرکس می‌تواند از آن استفاده کند تا یک پیام را رمزنگاری کند یا بگشاید.

از اینکه قفل یا کلید عمومی را به هرکسی داده شود واهمه‌ای ندارد زیرا تنها کسی که دارنده ی کلید خصوصی باشد می‌تواند پیام را بازکند. هر شخص، کلید عمومی و کلید خصوصی منحصر به فرد خود را دارد.

کلید قفل: این کلید شخصی است، زیرا هیچ‌کس مثل آن را ندارد. تنها ما با این کلید قادر به باز کردن قفل خود هستیم. مجبوریم که همواره از این کلید محافظت کنیم زیرا اگر یک مهاجم به این کلید دست یابد، آنگاه او می‌تواند همه جعبه‌های فلزی قفل‌شده با قفل را باز کند درنتیجه به اطلاعات حساس دست یابد.

اداره پلیس: یک مرکز صدور گواهی[1] است. یکی از عناصر اساسی یک PKI، مرکز صدور گواهی مورد اعتماد است. آقای اکبر و ما به اداره پلیس محلی اعتماد داریم و درنتیجه آن‌ها نامزدهای خوبی برای CA به وجود می‌آورند. برای حمایت از قانون و صداقت عمل به آن‌ها تکیه می‌کنیم. بنابراین، حتی اگر کسی که ما نمی شناسیم یا تاکنون ندیده‌ایم بخواهد به ما یک پیام امن ارسال کند، لازم نیست که در مورد اطمینان

[1] Certificate Authority

به فرد نگران باشیم. تنها باید به قانون و مجوزی که می‌گوید او فردی که خود می‌گوید[1] است، اعتماد کنیم.

برنامه اکبر جوجه: برای داستان این محدوده CA است. برای مثال اداره پلیس یا CA ممکن است قادر با شد که برای بسیاری از سناریوهای مختلف به‌عنوان شخص ثالث عمل کند. دامنه CA اطمینان خواهد داد که تمام تراکنش‌ها درزمینه یکسانی اتفاق خواهند افتاد. بنابراین برنامه اکبرجوجه تنها برای پرداختن به امتیاز آقای اکبر است.

اداره ثبت رستوران: RA[2] اعتبار ثبت‌نام است. اگر فردی بخواهد پیام‌های امنی را دریافت یا ارسال کند، ابتدا باید خود را با RA ثبت‌نام کند. RA نیاز خواهد داشت که هویت یک فرد با یک سند رسمی صادرشده مثل کارت شناسایی ملی یا گذرنامه ثبت شود. RA صحت این سند را مشخص می‌کند و ممکن است از روش‌های دیگر برای شناسایی فرد استفاده کند. در جلسه‌ای با ارائه نیازمندی‌های ثبت‌نام RA، فرد ثبت‌نام می‌شود و به او یک کلید عمومی و خصوصی تعلق می‌گیرد.

سؤالی که ممکن است پیش آید آن است که: دو اداره پلیس چگونه در دو شهرستان جداگانه یا حتی کشور متفاوت به یکدیگر اعتماد می‌کنند؟ فرض می‌کنیم همه اداره‌های پلیس از طریق یک روش داخلی، یک‌درجه‌ای از اعتماد را به وجود می‌آورند که بسیاری از بخش‌ها می‌توانند به‌عنوان یک‌نهاد عمل کنند. بنابراین به‌طور خلاصه، آقای کامران و ما برای اطمینان از جلوگیری از سال یا دریافت پیام اشتباه، هر دو از یک شخص ثالث مورداطمینان استفاده می‌کنیم. دو روش عملی برای حمله به این سیستم وجود دارد: ۱- مهاجم می‌تواند درروند ثبت‌نام با استفاده از جعل هویت، هویت یک کاربر مشروع را برباید. ۲- مهاجم می‌تواند یک حمله فیزیکی به پیام رمز شده در حال انتقال انجام دهد.

فایده این زیرساخت آن است که اگر آقای اصغر تلاش کند که خود را به‌جای آقای اکبر یا ما جا بزند او باید این کار را با حقه به فرآیند ثبت‌نام CA انجام دهد. در بسیاری از موارد، انجام این کار به دلیل اثبات مرحله هویت بسیار دشوار هست. برای کاهش حملات فیزیکی پیام‌ها در هنگام انتقال، سیستم قفل‌های قدرتمند و غیرقابل شکستن به کار گرفته می‌شود. این قفل‌ها همان الگوریتم‌های رمزنگاری می‌باشد.

۲-٤ اصطلاحات مورداستفاده در رمزنگاری

یادگیری اصطلاحات به‌صورت درست بسیار ضروری است. جدول ٤-۱ اصطلاحات مورداستفاده در

[1] احراز هویت

[2] Registration Authority

رمزنگاری را در زمینه‌ی نوشتن و ایمن‌سازی برنامه های کاربردی فهرست می‌کند.

جدول ٤-١ اصطلاحاتی که در رمزنگاری استفاده می‌شود

اصطلاح	توضیحات
Plaintext/cleartext	پیامی که هنوز رمز نشده است و به صورت آشکار قابل خواندن و درک است.
Encryption	فرآیند تبدیل متن ساده به یک متن رمز شده.
Ciphertext	متنی که در فرآیند رمزنگاری، رمز شده است.
Cryptographic algorithm/ algorithm/cipher	یک نوع تابع خاص ریا ضی ا ست که برای رمزنگاری و رمزگ شایی متن وا ضح استفاده می‌شود.
Decryption	معکوس رمزنگاری است. فرایندی است که توسط آن متن رمز شده به متن واضح و قابل خواندن تبدیل می‌شود.
Key	این مقدار به‌طور واحد روی الگوریتم رمزنگاری و رمزگشایی تأثیر می‌گذارد. می‌توان کلیدها را برای رمزنگاری یا رمزگشایی جدا کرد. بیشتر الگوریتم‌های مشترک برای کار وابسته به کلید می‌باشند.
Shared key/Symmetric key	یک کلید است که هم اطلاعات را رمزنگاری و هم رمزگشایی می‌کند. فرستنده و گیرنده هر دو این کلید رادارند، بنابراین، به‌عنوان یک کلید مشترک تعریف می‌شود.
Asymmetric key	این برای زمانی است که یک کلید برای رمزنگاری و یک کلید برای رمزگشایی با شد. می‌توانید این نوع کلید را برای رمزنگاری داده‌ها برای فرد خاصی ا ستفاده کنید. همه کاری که شما باید انجام دهید رمزنگاری داده‌ها با استفاده از کلید عمومی فرد است و سپس او می‌تواند از کلید خصوصی خود استفاده کند. بنابراین، یک کلید برای رمزنگاری (کلید عمومی) و دیگری برای رمزگشایی (کلید خصوصی) است.
Cryptanalysis	مطالعه شکستن متن رمزشده بدون دانش قبلی نسبت به کلید یا الگوریتم است.

٤-٣ رمزنگاری در برنامه های کاربردی موبایل

به نظر می‌رسد پیاده سازی PKI در برنامه‌های کاربردی به کار نمی‌آیند مخصوصاً هنگامی‌که برنامه مقداری پیچیده شود و منابع محدود باشد. باید سعی شود از روش‌های رمزنگاری مناسب برای منابع محدود استفاده کرد همان‌طور که در فصل قبل اشاره شد، حفاظت از اطلاعات کاربر بسیار مهمی است. در مثال فصل ۲ از الگوریتم استاندارد رمزنگاری پیشرفته(AES[1]) استفاده شد. AES یک الگوریتم کلیدی متقارن است زیرا تنها یک کلید برای رمزنگاری و رمزگشایی بکار رفته است.

٤-٣-١ الگوریتم‌های کلیدی متقارن

AES یک الگوریتم کلید متقارن یا رمز بلوکی است. همان‌طور که مشاهده شد، این بدان معنی است که تنها یک کلید در رمزنگاری و رمزگشایی استفاده می‌شود. الگوریتم‌ها برای رمزنگاری یا رمزگشایی اطلاعات به کار می‌روند. چگونگی پردازش داده‌ها منجر به تقسیم الگوریتم‌های متقارن می‌شود. برای مثال،می‌توان تعداد ثابتی از داده‌های بیتی را که به‌عنوان یک بلوک داده شناخته می‌شوند را در یک‌زمان پردازش کرد یا می‌توان داده‌های تک‌بیتی که به‌عنوان جریان شناخته می‌شوند را در یک‌زمان پردازش نمود. این تمایز، بلوک رمز شده و جریان رمز شده را می‌دهد. به‌طورکلی، AES یک الگوریتم بلوکی است که روی گروهی از داده‌های ۱۲۸بیتی کار می‌کند. رمزنگاری یک بلوک متن ۱۲۸بیتی با AES یک بلوک رمز شده با همان طول است. AES اجازه می‌دهد که اندازه کلید از صفرتا ۲۵٦بیت باشد. در مثال، از بیشترین اندازه کلید استفاده شده است. تعدادی دیگر از روش‌های رمزنگاری بلوکی در جدول ٤-۲ آورده شده است. برای به کار بردن دیگر روش‌های رمزنگاری به‌سادگی می‌توان نام الگوریتم را در متد KeyGenerator.getInstance() از AES به یکی از رمزهای بلوک در جدول زیر تغییر داد.

جدول ٤-۲ دیگر روش‌های رمزنگاری بلوکی

نام	API Level
AES	1+
Blowfish	10+
DES[2]	1+

[1] Advanced Encryption Standard

[2] Data Encryption Standard

۲-۳-٤ تولید کلید

کلید، بخش جدایی‌ناپذیر رمزنگاری است. بیشتر الگوریتم‌های مدرن رمزنگاری با کلید کار می‌کنند. در مثال فصل ۲، از یک تعداد مولد شبه تصادفی (PRNG) برای تولید کلیدهای رمزنگاری استفاده کردیم. همواره بهتر است بزرگ‌ترین اندازه کلید برای یک الگوریتم رمزنگاری انتخاب شود. اگر برنامه به‌کندی کار می‌کند بهتر است اندازه کلید به بزرگترین کلید قبلی کاهش داده شود. علت استفاده از بزرگ‌ترین اندازه کلید ممکن آن است که انجام حملات brute-force [1] که به کلید حمله می‌کنند را دشوارتر می‌کند. فرض کنید که شما یک کلید با اندازه ۱٦بیت را انتخاب می‌کنید. این به این معنی است که مهاجم باید ترکیبی ۱٦ بیتی از ۱ و ۰ ها را حداکثر٦٥٥۳٦ بار بیازماید. اگر اندازه کلید ۲٥٦بیتی انتخاب شود، پس مهاجم باید 2^{256} یا 11.6^{77} بار تلاش کند تا کلید را بشکند که چند سال برای او زمان می‌برد. البته این زمان می‌تواند به‌وسیله پیشرفت قدرت محاسباتی کاهش یابد، اما در همه زمینه‌های تحلیل رمز درست است. بنابراین، اندازه بزرگ کلید و الگوریتم‌های قوی تضمین می‌کند که یک مهاجم نمی‌تواند به‌راحتی متن رمز شده شما را بشکند. در اغلب موارد داده‌های رمز شده حکم عامل بازدارنده برای مهاجم رادارند. به‌جای گذراندن وقت روی شکستن رمزنگاری، آن‌ها به‌آسانی برای حمله به برنامه کاربردی مفروض بعدی حرکت می‌کنند.

توجه: یک حمله brute-force روی یک کلید یا رمز زمانی رخ می‌دهد که مهاجم به‌طور مداوم تلاش کند تا رمز صحیح را با تلاش های مکرر حدس بزند و رمزهای مبتنی بر ترکیب کاراکترهای مختلف مثل A-Z,a-z,0-9 و کاراکترهای خاص را امتحان کند. درنهایت، با آزمون همه ترکیبات ممکن، احتمالاً رمز صحیح را حدس می‌زند. یک کلید رمزنگاری با یک رمز(پسورد) معادل نیست. در مثال تولید کلیدمان، از یک کلید ۲٥٦بیتی تصادفی استفاده کردیم. به‌طورکلی روال رمزنگاری در پشت‌صحنه اتفاق می‌افتد، اگرچه می‌توان رمزهای کاربری را به کلیدها تبدیل کرد ولی این کار به‌هیچ‌وجه توصیه نمی شود. یک علت برای جلوگیری از این کار این است که رمزهای کاربری تقریباً همواره بیشتر از ۱۰تا ۱۲ بایت نیستند و این حتی نیمی از کلید با طول ۳۲ بایت (۳۲=۲٥٦/۸) را پوشش نمی‌دهد. با توجه به آنچه در مورد حملات brute-force گفته

[1] حمله ای که مهاجم تلاش می‌کند تا با آزمودن نام کاربری و رمزهای عبور گوناگون، به سیستم دسترسی پیدا کند و در آن تمام حالات ممکن تا رسیدن به جواب بررسی می‌گردد.

شـد بهتر اسـت که حداکثر طول کلید قابل‌قبول انتخاب شـود. همانطور که در کد ٤-١ مشـاهده می‌شـود از الگوریتم رمزنگاری AES ا ستفاده شده ا ست که عدد تصادفی از طریق الگوریتم SHA1 انتخاب می شود و همچنین کلیدی تصادفی تولید می‌شود.

کد ٤-١: الگوریتم تولید کلید

```
public static byte[] generateKey(byte[] randomNumberSeed) {
    SecretKey sKey=null;
    try  {
        KeyGenerator keyGen = KeyGenerator. getinstance( "AES");
        SecureRandom random= SecureRandom. getinstance( "SHA1PRNG");
        random.setSeed(randomNumberSeed);
        keyGen.init(256,random);
        sKey=keyGen.generateKey();
    } catch (NoSuchAlgorithmException e)  {
        Log.e(TAG, "No such algorithm exception");
    }
    return sKey.getEncoded();
}
```

٤-٣-٣ لایه گذاری داده‌ها

تاکنون در مورد پردازش الگوریتم‌های متقارن با یک اندازه ثابت بلوک داده‌ها صحبت شد. بااین‌حال، در مواردی که اطلاعات موجود کمتر از اندازه بلوک ورودی مورد نیاز توسـط الگوریتم اسـت چه باید کرد؟ **تصـویر ٤-٢** را در نظر بگیرید. دو بلوک اطلاعات وجود دارد اما تنها یکی از آن‌ها شـامل اندازه بلاک کامل هست (از یک اندازه بلوک ٨ بایتی برای سادگی کارها ا ستفاده خواهد شد) دومی تنها شامل ٤بیت هست. اگر این بلاک آخری با الگوریتم AES پردازش شـود، شـکسـت می‌خورد. برای رفع موقعیت‌هایی همچون این، گزینه‌های لایه گذاری مختلفی وجود دارد.

Offset	00 01 02 03 04 05 06 07	00 01 02 03 04 05 06 07
Data	41 42 43 44 45 46 47 48	49 50 51 52

تصویر ٤-٢: دو بلوک داده بدون تنظیم مناسب

زمانی که با موقعیت **تصویر ٤-٢** مواجه می‌شوید احتمالاً اولین لایه گذاری که به ذهن شما می‌رسد لایه گذاری ٤بیت باقی‌مانده با صـفر اسـت. به‌عنوان لایه گذاری صـفر شـناخته می‌شـود. گزینه‌های مختلف

لایه‌گذاری نیز وجود دارد. نمی‌خواهیم وارد جزئیات شویم اما باید به خاطر سپرد که نمی‌توان به‌راحتی یک متن را برداشت و بوسیله یک روش رمزنگاری بلوکی پردازش نمود. روش‌های رمزنگاری بلوکی همواره با یک ورودی ثابت کار می‌کنند و همواره یک اندازه خروجی ثابت دارند. تصویر ٤-٣ و تصویر ٤-٤ مثال‌های لایه گذاری صفر و لایه گذاری PKCS5/7[1] را نشان می‌دهد.

Offset	00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07
Data	41	42	43	44	45	46	47	48	49	50	51	52	**00**	**00**	**00**	**00**

تصویر ٤-٣: بلوک‌های داده با لایه گذاری صفر

Offset	00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07
Data	41	42	43	44	45	46	47	48	49	50	51	52	**04**	**04**	**04**	**04**

تصویر ٤-٤: بلوک‌های داده با لایه گذاری PKCS5/7

نکته:

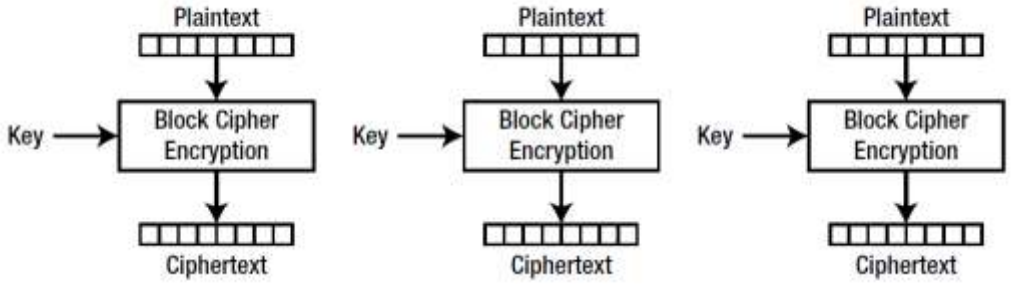
در لایه گذاری PKCS5/7 مقدار عددی طول بیت‌های باقی‌مانده را به‌عنوان یک‌لایه استفاده می‌کند. برای مثال، اگر ۱۰بیت برای لایه گذاری نیاز باشد، پس بیت‌لایه گذاری 0A است (که در هگزادسیمال به‌صورت 10 است). به‌طور مشابه، اگر ۲۸بیت برای لایه گذاری وجود داشت آنگاه بیت لایه گذاری 1C می‌بود.در مثال فصل ۲ هیچ لایه گذاری تعیین نشده است. به‌صورت پیش‌فرض اندروید از لایه گذاری PKCS5 استفاده می‌کند.

٤-٣-٤ حالت عملیات و رمزهای بلاکی

روش های رمزنگاری بلاکی، روش های رمزنگاری و رمزگشایی متنوعی دارند. ساده‌ترین فرم رمزنگاری زمانی ا ست که یک بلوک متن برای ایجاد یک بلوک متن رمز شده رمزنگاری می شود. بلوک‌های بعدی متن به همین ترتیب ادامه می‌یابد. این به‌عنوان حالت کتاب کد الکترونیکی (ECB[2]) شناخته می شود. تصویر ٤-٥ یک ارائه بصری از رمزنگاری ECB را نشان می‌دهد.

[1] Public Key Cryptography Standards

[2] Electronic Codebook



تصویر ٤-٥: رمزنگاری در حالت ECB

ECB هیچ حفاظتی در برابر حملات تشخیص الگو پیشنهاد نمی‌دهد. به این معنی که اگر پیام متنی شامل دو بلوک متنی یکسان باشد پس دو بلوک متن رمزی متناظر نیز وجود خواهد داشت. این‌یکی از روش‌هایی است که توسط تحلیل گران رمزنگاری برای تشخیص و تعیین الگوها بین متن رمزی استفاده می‌شود. پس‌ازاینکه الگوها شناسایی شدند، می‌توان به طرز دقیقی استنباط کرد که رمزنگاری ECB ا ستفاده شده ا ست. بنابراین، یک مهاجم تنها باید روی رمزگشایی یک بلوک خاص متن تمرکز کند. و نیازی به رمزگشایی کل پیام ندارد.

برای جلوگیری از این کار، تعدادی حالت مختلف عملیات برای رمزنگاری بلوکی وجود دارد:

- زنجیره بلوک- رمز(CBC)
- انتشار زنجیره سازی بلوک- رمز(PCBC)
- بازخورد رمزی(CFB)
- بازخورد خروجی (OFB)

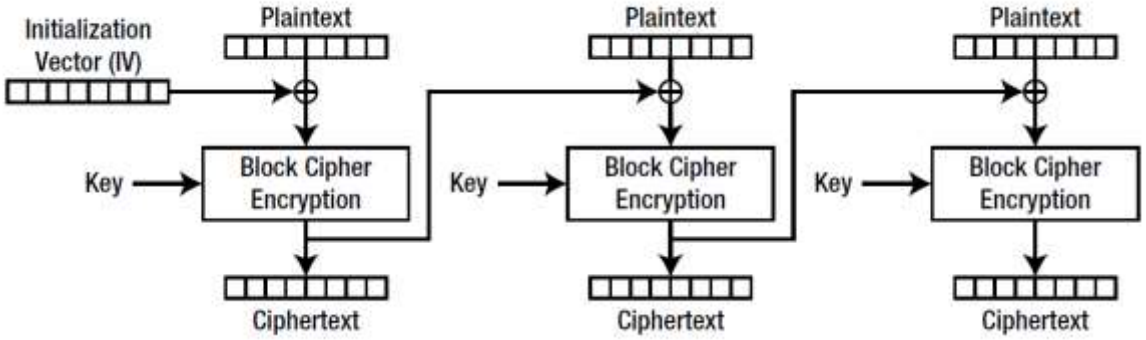
در این بخش توجه بیشتر بر روی فرایند رمزنگاریمعطوف شده است چرا که به‌سادگی می‌توان مراحل رمزگشایی را از معکوس فرایند رمزنگاری فهمید:

حالت CBC[1]: حالت زنجیره رمزی- بلوک (یا صفر- بلوک) (تصویر ٤-٦ را مشاهده کنید) از یک مقدار اضافی به‌عنوان یک بردار اولیه (IV[2]) استفاده می‌کند که برای انجام عملیات XOR روی بلوک اول متن

[1] Cipher Block Chaining

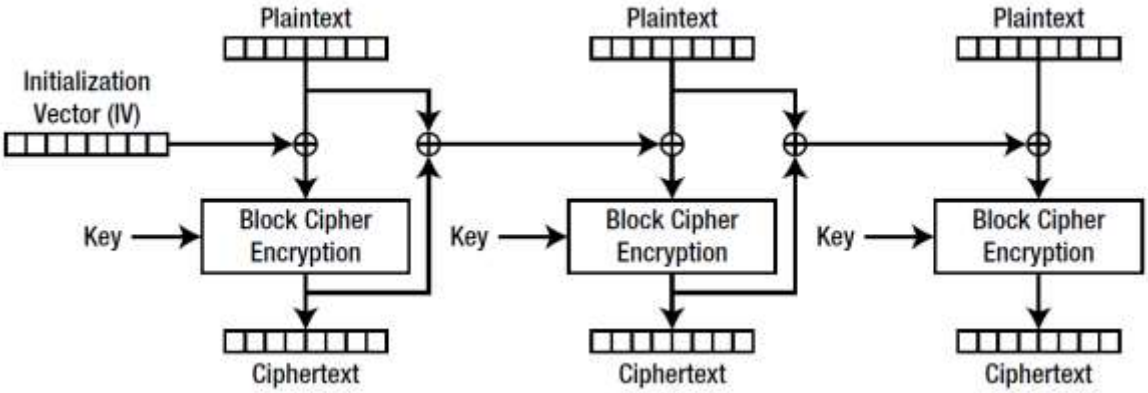
[2] Initialization Vector

استفاده می‌شود. پس‌ازاین، هر بلوک متن رمز شده با بلوک متن بعدی XOR می‌شود، و به همین ترتیب. این نوع حالت تضمین می‌کند که نتیجه هر بلوک متن رمز شده وابسته به بلوک متن واضح قبلی هست.



تصویر ٤-٦: رمزنگاری در حالت CBC

حالت PCBC[1]: انتشار حالت زنجیره‌بلوک- صفر(تصویر ٤-٧) بسیار مشابه حالت CBC است. تفاوت در این اســـت که به‌جای XOR کردن IV با اولین بلوک و XOR کردن متن رمز شـــده با بلوک‌های بعدی، در حالت PCBC مقدار IV و متن رمز شده برای بلوک اول XOR می شود و سپس نتیجه XOR متن واضح و متن رمز شـده را با بلوک متن بعدی XOR می‌کند. طراحی این حالت به‌گونه‌ای اسـت که یک تغییر کوچک در متن رمز شده از طریق فرایند رمزنگاری یا رمزگشایی انتشار می‌یابد.

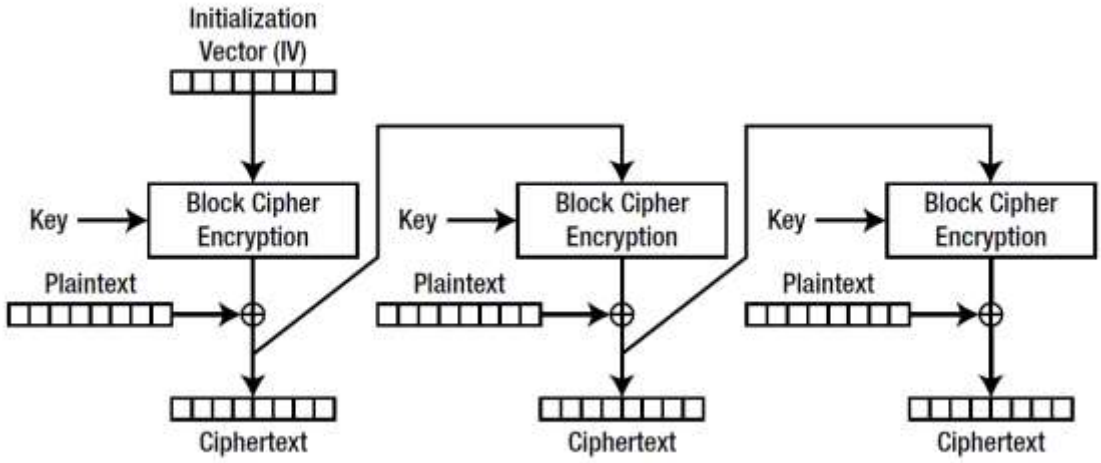


تصویر ٤-٧: رمزنگاری در حالت PCBC

حالت CFB: حالت بازخورد رمزی (تصویر ٤-٨ را مشاهده کنید) مکان IV و متن واضح در حالت CBC را

[1] Propagating Cipher Block Chaining

تغییر می‌دهد. یعنی به‌جای XOR کردن متن وا ضح و رمزنگاری آن، و سپس XOR کردن متن رمزی با متن وا ضح، در حالت CFB[1] ابتدا IV را رمزنگاری می‌کند سپس برای دریافت متن رمز شده آن را با متن وا ضح XOR می‌کند. سپس برای بلوک‌های بعدی، متن رمز شده دوباره رمزنگاری می شود و با متن وا ضح XOR می‌شود تا بلوک بعدی متن رمز شده را بدهد.

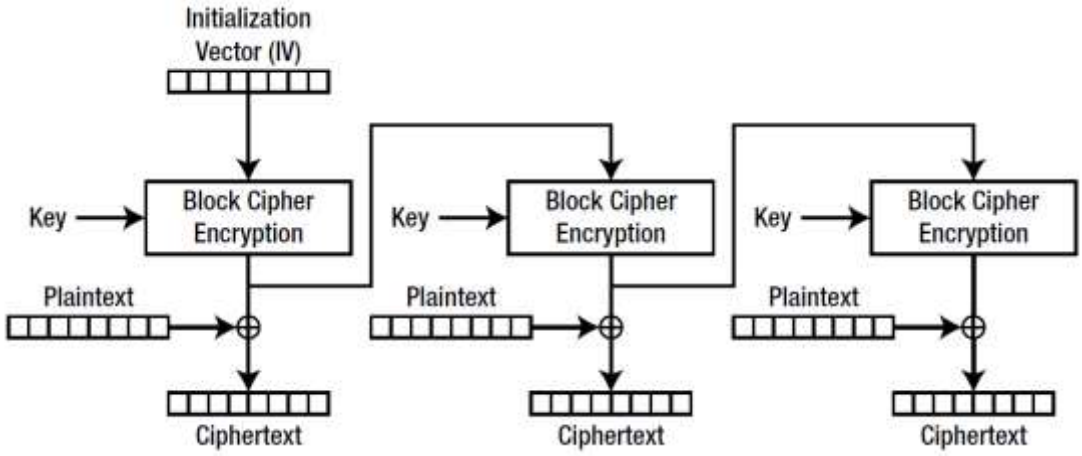


تصویر ٤-٨: رمزنگاری در حالت CFB

حالت OFB[2]:حالت بازخورد خروجی (تصویر ٤-٩ را مشاهده کنید) بسیار مشابه حالت CFB هست. تفاوت این ا ست که، به‌جای ا ستفاده از XOR شده‌ی IV رمز شده، متن وا ضح، از IV رمز شده ا ستفاده می‌کند. بنابراین،برای بلوک اول، IV با کلید رمزنگاری می‌شود و این به‌عنوان ورودی برای بلوک بعدی استفاده می شود. متن رمز شده از بلوک اول سپس با اولین بلوک از متن وا ضح XOR می شود. رمزنگاری بعدی از IV رمز شده از بلوک قبلی قبل از XOR کردن استفاده می‌کند.

[1] Cipher Feedback

[2] Output Feedback



تصویر ٤-٩: رمزنگاری در حالت OFB

اگر به مثال ا صلی دقت نمایید، م شاهده می‌کنید که از یک حالت خاص رمزنگاری ا ستفاده نمی شود. اندروید برای رمزنگاری و رمزگشایی به‌طور پیش‌فرض از حالت ECB اســتفاده می‌کند. انتخاب یک حالت رمزنگاری پیچیده‌تر مثل CBC یا CFB به خود شما به‌عنوان یک تو سعه‌دهنده بستگی دارد. حالا که شما از عملکرد داخلی الگوریتم متقارن AES بیشــتر آگاه هســتید، چگونگی تغییر لایه گذاری حالت عملکرد زمان رمزنگاری به شما نشان داده خواهد شد. برای خواندن فهرستی همانند کد ٤-٢، کد را تغییر دهید. به خطوط پررنگ کد توجه داشــته باشــید تنها چند تغییر انجام داده‌شــده اســت. ابتدا، AES به لایه گذاری AES/CBC/PKCS5 تغییر داده شد. سپس، بردار اولیه (IV) به متد init() ا ضافه گ شت. همان‌طور که قبلاً ا شاره شد، زمانی که رمزنگاری AES انتخاب می شودحالت پیش‌فرضی که اندروید ا ستفاده خواهد کرد لایه گذاری AES/ECB/ PKCS5 ا ست. شما می‌توانید با دو بار اجرای برنامه این کار را برر سی کنید یک‌بار با AES و یک‌بار با لایه گذاری AES/ECB/PKC5 هر دو متن رمز شده یکسانی را خواهند داد.

کد ٤-٢: روال رمزنگاری تغییر داده شده با حالت رمزنگاری CBC

```
private static byte[] encrypt(byte[]  key, byte[] data, byte[] iv){
    SecretKeySpec sKeySpec=new SecretKeySpec(key,"AES");
    Cipher cipher;
    byte[] ciphertext = null;
    try {
        cipher= Cipher. getinstance("AES/CBC/PKCSSPadding");
        IvParameterSpec ivspec=new IvParameterSpec(iv);
        cipher.init(Cipher.ENCRYPT_MODE, sKeySpec, ivspec);
        ciphertext=cipher.doFinal(data);
    } catch (NoSuchAlgorithmException e)  {
        Log.e(TAG,"NoSuchAlgorithmException");
    } catch (NoSuchPaddingException e)  {
        Log.e(TAG,"NoSuchPaddingException");
    } catch (IllegalBlockSizeException e) {
        Log.e(TAG,"IllegalBlockSizeException");
    } catch (BadPaddingException e) {
```

```
            Log.e(TAG,"BadPaddingException");
        } catch (InvalidKeyException e) {
            Log.e(TAG,"InvalidKeyException");
        }
        return ciphertext
}
```

فرض کنید که باید یک کلید مخفی را انتخاب می‌کردید. به‌جای استفاده از مولد عدد تصادفی برای تولید کلید مخفی می‌توانید یک‌روال مشابه کد ٤-٣ بنویسید. در این کد، stringKey کلید موردنظر برای رمزنگاری اطلاعات خواهد بود..

کد ٤-٣: نمونه تولید کلید با یک مقدار کلید ثابت

```
public static byte[] generateKey(String stringKey) {
    try {
        SecretKeySpec sks=new
SecretKeySpec(stringKey.getBytes(),"AES");

    } catch (NoSuchAlgorithmException e) {
        Log.e(TAG,"No such algorithm exception");
    }
    return sks.getEncoded();
}
```

٤-٤ ذخیره‌سازی داده‌ها در اندروید

رمزنگاری و ذخیره سازی داده‌ها مو ضوعاتی هستند که برای تولید یک برنامه امن‌تر به یکدیگر پیوند خورده‌اند. اندروید برنامه‌ها را در زمینه‌ی امنیتی جداگانه اجرا می‌کند. به این معنی که هر برنامه با UID و GID خود اجرا می‌شود. زمانی که یک برنامه داده‌ها را می‌نویسد دیگر برنامه‌ها قادر به خواندن داده‌ها نیستند. اگر می‌خواهید داده‌ها را بین برنامه‌ها به اشتراک بگذارید آنگاه به‌طور واضح نیازمند اشتراک‌گذاری با استفاده از یک ارائه‌دهنده محتوا ه ستید. سؤال :"اگر اندروید در حال حا ضر از اطلاعات محافظت می‌کند چرا باید تمام مباحث رمزنگاری را پوشش دهیم؟" همان‌طور که در آغاز این فصل اشاره شد فقط برای زمان‌های پیش‌بینی‌ نشده به هنگام آ سیب‌پذیری، ویروس یا زمانی که Trojan چهره ز شت خود را ن شان می‌دهد باید یک‌لایه امنیتی دیگر بر روی لایه امنیتی اندروید ایجاد کرد. اندروید به شما اجازه می‌دهد که توسط پنج گزینه مختلف اطلاعات را ذخیره نمایید (جدول ٤-٣ را مشاهده نمایید). بدیهی است که نیاز به تصمیم‌گیری برای مکان ذخیره برنامه‌های کاربردی خود بر اساس نیازمندی‌هایتان دارید.

جدول ٤-٣: روش‌های ذخیره داده در اندروید

روش ذخیره‌سازی	شخصی‌سازی داده‌ها	توضیحات
Shared preferences	چهار حالت شخصی‌سازی: MODE_PRIVATE, MODE_WORLD_ READABLE, MODE_WORLD_WRITABLE, MODE-MULTI- PROCESS.	به شـــما اجازه می‌دهد که انواع داده‌های اولیه را ذخیره کنید(برای مثال int, Boolean, float, long, and String) که باید در برابر دسـتگاه محافظت شـــوند حتی اگر برنامه در حال اجرا نباشد داده‌های شما حفظ خواهد شد مگر اینکه دستگاه دوباره شروع به کار کند.
حافظه داخلی	سه حالت شخصی‌سازی: MODE_PRIVATE, MODE_WORLD_READABLE, MODE_WORLD_WRITABLE.	به شـما اجازه می‌دهد که داده‌هایتان را در حافظه داخلی دســتگاه ذخیره نمایید. به‌طورکلی این داده‌ها توسط دیگر برنامه‌ها یا حتی کاربران نهایی قابل‌دســتیابی نیســتند. یک مکان ذخیره‌ســازی شخصی است. داده‌های ذخیره‌شده در اینجا حتی بعد از شـروع به کار کردن دسـتگاه نیز محافظت می‌شـوند. زمانی که کاربر نهایی برنامه شـما رو حذف می‌کند اندروید نیز داده‌های شـمارا حذف خواهد کرد.
حافظه خارجی	حالت پیش‌فرض: MODE PRIVATE داده‌ها به‌طور پیش‌فرض توسط همه قابل خوانا هست.	داده‌هایی که اینجا ذخیره می‌شـوند قابل خواندن توســط همه می‌باشـند. کاربر دســتگاه و دیگر برنامه‌ها می‌توانند این داده‌ها را بخوانند اصلاح و یا حذف کنند. حافظه خارجی باکارت SD یا دسـتگاه حافظه داخلی (که غیرقابل‌حذف اسـت) در ارتباط است .
پایگاه داده SQLite	پایگاه داده‌هایی که ایجاد می‌کنید توسط هر کلاسی بین برنامه قابل‌دسترس هستند. برنامه‌های خارجی هیچ دسترسی به این پایگاه داده ندارند.	اگر نیازمند ایجاد یک پایگاه داده برای برنامه خود برای کسب مزیت جست‌وجوی SQLite و توانایی مدیریت داده‌ها هســتید از مکانیســم ذخیره‌سازی پایگاه داده SQLite استفاده کنید.
اتصال شبکه	بر اساس تنظیمات سرویس وب	شما می‌توانید داده‌ها را از طریق سرور وب از راه دور ذخیره و بازیابی کنید. می‌توانید در رابطه با

		آن در فصل ۵ بیشتر بخوانید.

انتخاب روش مناسب برای ذخیره داده‌ها تا حد زیادی به نیاز شما وابسته است. به برنامه Proxim در فصل ۲ دقت نمایید می‌توانیم داده‌هایمان را در یک پایگاه داده SQL ذخیره نماییم زیرا این کار ما را از انتخاب یک ساختار داده غیرضروری در امان می‌دارد. بیایید به مثال‌ها نگاهی بیندازیم که چگونه می‌توان داده‌ها را با استفاده از روش‌های متنوع ذخیره یا بازیابی نمود.

٤-٤-١ تنظیمات مشترک

برای ذخیره تنظیمات برنامه‌ها، «تنظیمات مشترک[1]» بسیار مفید است. همان‌طور که نام آن بیان می‌کند، این روش ذخیره‌سازی برای نگهداری تنظیمات کاربر یک برنامه بسیار مناسب است. در این مثال باید اطلاعات مربوط به یک سرور ایمیل که برنامه باید داده‌ها را از آن بازیابی کند را ذخیره کنیم. باید hostname و پورت سرور ایمیل و اینکه آیا سرور ایمیل از SSL استفاده می‌کند را ذخیره کنیم. کد پایه برای ذخیره (کد ٤-٤) و بازیابی (کد ٤-٥) داده‌ها برای SharedPreferences داده شده است. کلاس ذخیره سازی مثال یک در کد ٤-٦ و خروجی برنامه در تصویر ٤-١٠ نشان داده‌شده است.

کد ٤-٤:کد ذخیره داده در sharedpreferences

```
package android.secure.programming;
import java.util.Hashtable;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.preference.PreferenceManager;

public class StoreData {

    public static boolean storeData(Hashtable data, Context ctx) {
        SharedPreferences
prefs=PreferenceManager.getDefaultSharedPreferences(ctx);
        String hostname= (String) data.get( "hostname");
        int port=(Integer) data.get("port");
        boolean useSSL=(Boolean) data.get("ssl");
        Editor ed = prefs. edit ();
        ed.putString("hostname", hostname);
        ed.putlnt("port", port);
        ed.putBoolean("ssl", useSSL);
        returned.commit();
    }
}
```

[1] SharedPreferences

کد ٤-٥:کد بازیابی داده از SharedPreferences

```
package android.secure.programming;

import java.util.Hashtable;
import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;

public class RetrieveData {
    public static Hashtable get(Context  ctx) {
        String hostname = "hostname";
        String port= "port";
        String ssl = "ssl";

        Hashtable data=new Hashtable();
        SharedPreferences prefs=
PreferenceManager.getDefaultSharedPreferences(ctx);
        data.put(hostname, prefs.getString(hostname, null));
        data.put(port, prefs.getint(port, o));
        data.put(ssl, prefs.getBoolean(ssl, true));
        return data;
    }
}
```

کد ٤-٦: StorageExample 1, Main Class

```
package android.secure.programming;

import java.util.Hashtable;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.widget.EditText;

public class StorageExample1Activity extends AppCompatActivity {

    @Override
    public void onCreate(Bundle savedinstanceState) {
        super.onCreate(savedinstanceState);
        setContentView(R.layout.main);
        Context cntxt=getApplicationContext();
        Hashtable data=new Hashtable();
        data. put ( "hostname", "smtp. gmail. com");
        data.put("port", 587);
        data.put("ssl", true);

        if (StoreData.storeData(data, cntxt)) Log.i("SE", "Successfully
wrote data");
        else
        Log.e("SE", "Failed to write data to Shared Prefs");
        EditText ed=(EditText) findViewByid(R.id.editText1);
        ed.setText(RetrieveData.get(cntxt).toString());
    }
}
```

تصویر ٤-١٠: خروجی برنامه StorageExample1

٤-٤-٢ حافظه داخلی

همان‌طور که مشاهده شد، تنظیمات مشترک برای ذخیره انواع داده‌ها با جفت key-value بسیار مناسب است. این تا حدی مشابه جدول هش یا حتی شیء Properties در جاوا هست. محدودیت روش SharedPreferences این است که تنها می‌توان انواع داده های اولیه را ذخیره نمود و امکان ذخیر هانواع داده‌های پیچیده‌تر همچون بردارها یا جداول هش فراهم نیست. ماگر بخوایهم این نوع داده‌ها را نیز ذخیره نماییم باید از حافظه داخلی بهره برد.

در این روش ذخیره‌سازی شما داده‌هایتان را با یک OutputStream در حافظه داخلی می‌نویسید. بنابراین، تنها شیء ای می‌تواند در حافظه داخلی نوشته شود که به یک‌رشته از بایت‌ها ردیف شود. با ایجاد کلاس StorageExample2 (کد ٤-٧) شروع می کنیم. همانند گذشته، ماژول‌های ذخیره و بازیابی در کدهای جداگانه نشان داده‌شده است (

کد ٤-٨ و کد ٤-٩ را به ترتیب ببیند). تصویر ٤-١١ خروجی را نشان می‌دهد.

کد ٤-٧: StorageExample2, Main Class

```
package android.secure.programming;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.widget.EditText;
public class StorageExample2Activity extends AppCompatActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Context ctx = getApplicationContext();
        // Store data
        Contact contact = new Contact();
        contact.setFirstName("Android");
        contact.setLastName("Programming");
        contact.setEmail("android@android.ir");
        contact.setPhone(" + 12120031337");
        StoreData.storeData(contact.getBytes(), ctx);
        // Retrieve data
        EditText ed = (EditText) findViewById(R.id.editText1);
        ed.setText(new String(RetrieveData.get(ctx)));
    }
}
```

کد ٤-٨: استفاده از StoreData.java برای ذخیره داده در حافظه داخلی

```
package android.secure.programming;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import android.content.Context;
import android.util.Log;
public class StoreData {
    public static final String file = "contacts";
    public static void storeData(byte[] data, Context ctx) {
        try {
            FileOutputStream fos = ctx.openFileOutput(file,
ctx.MODE_PRIVATE);
            fos.write(data);
            fos.close();
        } catch (FileNotFoundException e) {
            Log.e("SE2", "Exception: " + e.getMessage());
        } catch (IOException e) {
            Log.e("SE2", "Exception: " + e.getMessage());
        }
    }
}
```

کد ٤-٩: استفاده از RetrieveData.java برای بازیابی داده از حافظه داخلی

```
package android.secure.programming;
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import android.content.Context;
import android.util.Log;
public class RetrieveData {
    public static final String file = "contacts";
    public static byte[] get(Context ctx) {
        byte[] data = null;
        try {
            int bytesRead = 0;
            FileInputStream fis = ctx.openFileInput(file);
            ByteArrayOutputStream bos = new ByteArrayOutputStream();
            byte[] b = new byte[1024];
            while ((bytesRead = fis.read(b)) ! = -1) {
                bos.write(b, 0, bytesRead);
            }
            data = bos.toByteArray();
        } catch (FileNotFoundException e) {
            Log.e("SE2", "Exception: " + e.getMessage());
        } catch (IOException e) {
            Log.e("SE2", "Exception: " + e.getMessage());
        }
        return data;
    }
}
```

```
android|programming|null|null|android@android.ir|+12120031337|
```

تصویر ٤-١١: خروجی برنامه StorageExample2

توجه داشته باشید که کد ٤-٧ برای ذخیره‌سازی داده‌ها از شیء Contact مثال Proxim استفاده می‌کند.

٤-٤-٣ پایگاه داده‌های SQLite

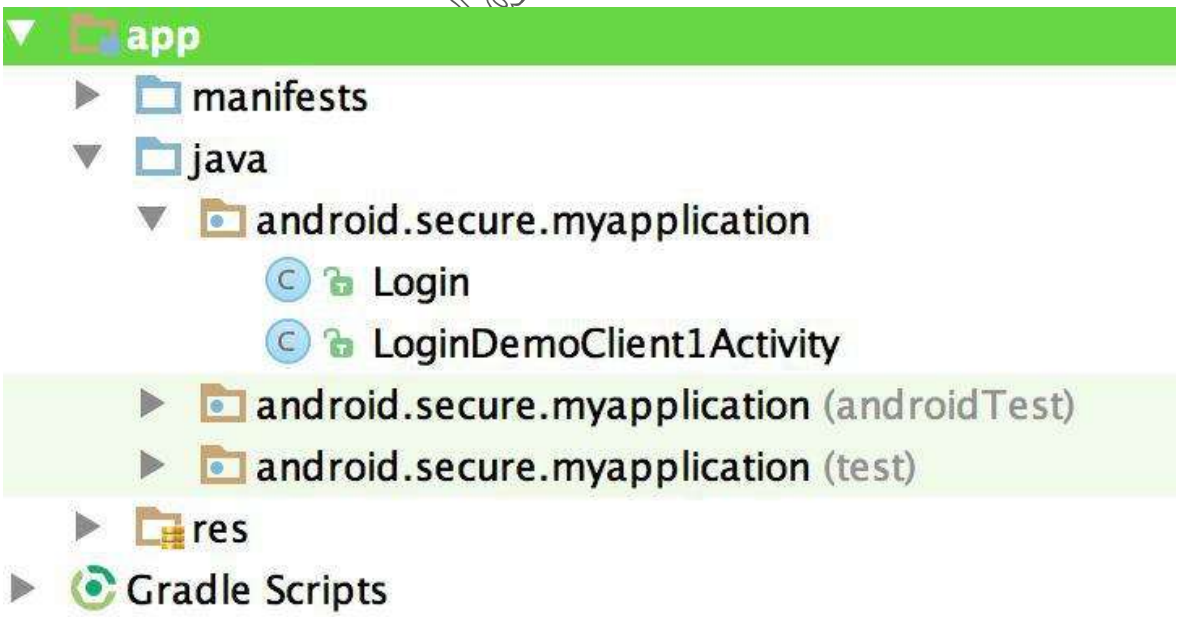
از مثال ذخیره‌سازی خارجی چشم‌پوشی نمودیم، چرا که قبلاً با چگونگی ذخیره‌سازی داده‌ها به صورت خارجی آ شنا شدیم. در عوض، روی چگونگی ایجاد، ذخیره و بازیابی داده‌ها با ا ستفاده از پایگاه داده SQLite اندروید متمرکز خواهیم شد. یک پایگاه داده ایجاد خواهیم کرد که می‌توان از آن برای ذخیره شیء Contact در برنامه Proxim استفاده نمود. جدول ٤-٤ طرح‌بندی جدول را نشان می‌دهد. به منظور سادگی، همه ستون‌ها را به‌صورت متن در نظر گرفته‌ایم . زمانی که می‌خواهید جدول خود را ایجاد کنید

اطمینان یابید که نوع داده‌ای ستون‌ها (شماره، رشته، یا زمان) بر اساس داده موردنظرتان باشد.

جدول ٤-٤: The Contacts Table Inside the ContactsDB SQLite Database

Column Name	Column Data Type
FIRSTNAME	TEXT
LASTNAME	TEXT
EMAIL	TEXT
PHONE	TEXT
ADDRESS1	TEXT
ADDRESS2	TEXT

یک پروژه جدید به نام StorageExample3 با ساختار همان داده‌شده در تصویر ٤-١٢ در محیط توسعه خود ایجاد کنید. اگر به شیء Contact نیاز دارید آن را از مثال Proxim کپی کنید.



تصویر ٤-١٢: The StorageExample3 project structure

کلاس StorageExample3، کلاس اصلی برای کار با پایگاه داده SQLite و ایجادکننده شیء Contact با داده‌های آن را نشان می‌دهد (کد ٤-١٠ را ببینید).

کد ٤-١١ یک کلاس کمکی را نشان می‌دهد که می‌توانید برای دست‌کاری پایگاه داده SQLite از آن استفاده نمایید.

کد ٤-١٢چگونگی ا ستفاده از این کلاس کمکی برای نو شتن داده‌ها از شیء Contact به پایگاه داده و تصویر ٤-١٣ چگونگی واکشی داده‌ها از پایگاه داده SQLite را نشان می‌دهد و یک شیء Contact را برمی‌گرداند. شما می‌توانید با دنبال کردن این کدها، درک بهتری از مطالب ذکر شده، پیدا خواهید نمود.

کد ٤-١٠:StorageExample3

```
package android.secure.programming;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.EditText;
public class StorageExample3Activity extends AppCompatActivity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // Store data
        Contact contact = new Contact();
        contact.setFirstName("android");
        contact.setLastName("programming");
        contact.setEmail("android@android.ir");
        contact.setPhone(" + 12120031337");
        ContactsDb db = new
        ContactsDb(getApplicationContext(),"ContactsDb",null,1);
        Log.i("SE3",String.valueOf(StoreData.store(db, contact)));
        Contact c = RetrieveData.get(db);
        db.close();
        EditText ed = (EditText)findViewById(R.id.editText1);
        ed.setText(c.toString());
    }
}
```

کد ٤-١١:کلاس ContactsDB Helper برای کار با پایگاه داده sqlite

```
package android.secure.programming;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
public class ContactsDb extends SQLiteOpenHelper {
    public static final String tblName = "Contacts";
    public ContactsDb(Context context, String name, CursorFactory
factory,
    int version) {
        super(context, name, factory, version);
    }
    @Override
```

```
    public void onCreate(SQLiteDatabase db) {
        String createSQL = "CREATE TABLE " + tblName
        + " ( FIRSTNAME TEXT, LASTNAME TEXT, EMAIL TEXT,"
        + " PHONE TEXT, ADDRESS1 TEXT, ADDRESS2 TEXT);";
        db.execSQL(createSQL);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        // Use this to handle upgraded versions of your database
    }
}
```

کد ٤-١٢: کلاس StoreData داده‌ها را از شیء contact در پایگاه داده می‌نویسد

```
package android.secure.programming;
import android.content.ContentValues;
import android.database.sqlite.SQLiteDatabase;
public class StoreData {
    public static long store(ContactsDb db, Contact contact) {
        // Prepare values
        ContentValues values = new ContentValues();
        values.put("FIRSTNAME", contact.getFirstName());
        values.put("LASTNAME", contact.getLastName());
        values.put("EMAIL", contact.getEmail());
        values.put("PHONE", contact.getPhone());
        values.put("ADDRESS1", contact.getAddress1());
        values.put("ADDRESS2", contact.getAddress2());
        SQLiteDatabase wdb = db.getWritableDatabase();
        return wdb.insert(db.tblName, null, values);
    }
}
```

کد ٤-١٣: کلاس RetrieveData برای بازیابی اطلاعات از پایگاه داده و برگرداندن یک شیء contact

```
package android.secure.programming;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
public class RetrieveData {
    public static Contact get(ContactsDb db) {
        SQLiteDatabase rdb = db.getReadableDatabase();
        String[] cols = { "FIRSTNAME", "LASTNAME", "EMAIL", "PHONE" };
        Cursor results = rdb.query(db.tblName, cols, "", null, "", "",
"");
        Contact c = new Contact();
        results.moveToLast();
        c.setFirstName(results.getString(0));
        c.setLastName(results.getString(1));
        c.setEmail(results.getString(2));
        c.setPhone(results.getString(3));
        return c;
    }
```

```
}
```

بسیار نادر است که از یک فایل مسطح برای ذخیره داده‌ها استفاده شود. مگر آن‌نکه با داده‌های خالص دودویی سر و کار داشته باشیم (مثل عکس،ویدیو یا موسیقی) بیشتر داده‌هایی که ذخیره می‌شوند یا به‌عنوان جفت key-value اســـتفاده می‌شـــوند یا در یک پایگاه داده SQLite ذخیره می‌گردند. بنابراین، می‌توان از SharedPreference اندروید یا SQLiteDatsbase برای انجام این کار استفاده نمود. هر دو روش، قابلیت‌های مدیریتی خوبی را ارائه می‌دهند. اگر تاکنون با پایگاه داده SQLite کارنکرده‌اید می‌توانید کمی بیشـــتر آن را بررســـی کنیـد. در عوض بیشـــتر ســـیســـتم‌عـامـل‌هـای موبـایـل همچون ios اپـل و RIM برای BlackBerrySmartphoneOS از پایگاه داده‌های SQLite پشـتیبانی می‌کنند. نکته مثبت این اســت که پایگاه داده‌های SQLite بسیار قابل‌حمل هستند و شما می‌توانید یک پایگاه داده SQLite را روی هر سیستم‌عاملی مثل مک و لینوکس و ویندوز ایجاد و اصلاح کنید و بخوانید.

در ادامه، کد پروژه StorageExample3 تحلیل می‌شود. کد ٤-١٠ کلاس اصلی است و یک شیء Contact را با داده‌های درون آن ایجاد می‌کند.

```
Contact contact = new Contact();
contact.setFirstName("android");
contact.setLastName("programming");
contact.setEmail("android@android.ir");
contact.setPhone(" + 12120031337");
```

پس از کلاس ContactsDb (

کد ٤-١١) از زیر کلاس‌های SQLiteOpenHelper استفاده می‌کند.

```
ContactsDb db=new
ContactsDb(getApplicationContext(),"ContactsDb",null,1);
```

بنابراین اگر بخواهید پایگاه داده خود را ایجاد کنید از زیر کلاس‌های SQLiteOpenHelper استفاده نمایید. آنگاه این کد از متدهای store() (

کد ٤-١٢) برای حفظ شـیء ایجادشـده Contact اسـتفاده می‌کند.متد store() را فراخوانی می‌کنیم و پایگاه داده SQLite جدیدمان و شـیء Contact را به آن می‌دهیم. StoreData شـیء Contact را به اشـیاء ContentValues تقسیم خواهد کرد.

```
ContentValues values = new ContentValues();
values.put("FIRSTNAME", contact.getFirstName());
values.put("LASTNAME", contact.getLastName());
values.put("EMAIL", contact.getEmail());
values.put("PHONE", contact.getPhone());
values.put("ADDRESS1", contact.getAddress1());
```

```
values.put("ADDRESS2", contact.getAddress2());
```

نکته: اگر در حال ایجاد اشیاء داده خود هستید و می‌دانید که قصد دارید از روش پایگاه داده برای ذخیره داده‌ها استفاده کنید باید ContentValueها را برای اشیاء داده خود در نظر بگیرید. زیرا کار با داده‌ها را در زمان ذخیره و بازیابی آن‌ها بسیار راحت‌تر خواهد ساخت.

سپس مقادیر را در جداول پایگاه داده خود می‌نویسیم. شیء SQLiteOpenHelper می‌تواند یک WritableDatabase یا ReadableDatabase را بازیابی کند بنابراین باید از منا سب‌ترین گزینه برای زمان جایگذاری یا پرس‌وجوی داده‌ها از جدول استفاده کنیم.

```
SQLiteDatabase wdb = db.getWritableDatabase();
return wdb.insert(db.tblName, null, values);
```

کلاس RetrieveData بازیابی داده‌ها از پایگاه داده را بر عهده می‌گیرد.اینجا تنها به مقادیر آخرین ردیف قرار داده‌شده توجه داریم. در یک برنامه برای واکشی هر ردیف اطلاعات باید Cursor خود را تکرار کنیم.

```
SQLiteDatabase rdb = db.getReadableDatabase();
String[] cols = { "FIRSTNAME", "LASTNAME", "EMAIL", "PHONE" };
Cursor results = rdb.query(db.tblName, cols, "", null, "", "", "");
```

پس از واکشی داده‌ها از جدول یک شیء Contact را دوباره ایجاد می‌کنیم تا اطلاعات آن ردیف بازگردانیم:

```
Contact c = new Contact();
results.moveToLast();
c.setFirstName(results.getString(0));
c.setLastName(results.getString(1));
c.setEmail(results.getString(2));
c.setPhone(results.getString(3));
return c;
```

خروجی احتمالاً همانند مثال قبلی است.(تصویر ٤-١٣ را مشاهده نمایید)

```
android|programming|null|null|android@android.ir|+12120031337|
```

تصویر ٤-١٣: خروجی برنامه StorageExample3

٤-٥ ترکیب ذخیره‌سازی داده‌ها با رمزنگاری

دو نکته بسیار مهم به‌طور مجزا در این فصل پو شش داده شد. داده ذخیره شده درون حافظه‌ای که انتخاب شده‌است قرار خواهدگرفت. برای ضمانت این‌که داده‌هاتوسط برنامه‌های غیرمجاز خوانده نمی شود می‌توان به اندروید اعتماد کرد اما اگر یک ویروس هفته بعد منتشر شود چه‌کاری باید انجام شود؟ این ویروس تنها روی تلفن‌های اندروید تأثیر می‌گذارد و قادر به دور زدن مجوزهای پایگاه داده SQLite برای خواندن تمام پایگاه داده‌های موجود روی دستگاه هست. اکنون تنها امید شما به منظور حفظ اطلاعات شخصی به خطر

افتاده است و امکان کپی همه داده‌ها از دستگاه شما مقدور است.

چنین حملاتی در فصل قبل مورد بررسی قرار گرفت و در گروه حملات غیرمستقیم واقع شدند. این ویروس غیرمستقیم است زیرا به‌طور مستقیم با برنامه شما کار نمی‌کند بلکه هدف آن سیستم‌عامل اندروید است. هدف کپی همه داده های پایگاه داده‌های SQLite است به این امید که حمله‌کننده بتواند هر اطلاعات حساسی که آنجا ذخیره‌شده است را کپی کند. اگر یک‌لایه محافظت دیگر اضافه نمایید؛ پس‌ازآن، حمله‌کننده اطلاعات را به صورت درهم مشاهده می‌کند. بیایید یک کتابخانه رمزنگاری دائمی بسازیم که بتوانیم در همه برنامه‌ها استفاده کنیم. ابتدا با ایجاد یک مجموعه مختصری از ویژگی‌ها شروع می‌کنیم:

- استفاده از الگوریتم های متقارن: کتابخانه از یک الگوریتم متقارن یا رمزنگاری بلوکی، برای رمزنگاری و رمزگشایی داده‌ها که قادر به تغییرات در الگوریتم انتخابی است، استفاده خواهد نمود.
- استفاده از یک کلید ثابت: باید قادر به دربرگیری یک کلید که می‌توان در دستگاه ذخیره کرد باشیم که قادر خواهد بود داده‌ها را رمزنگاری و رمزگشایی کند.
- کلید ذخیره شده روی دستگاه: کلید بر روی دستگاه قرار خواهد گرفت. درحالی‌که از منظر حملات مستقیم برای برنامه یک خطر محسوب می‌شود، ولی در مقابله با حملات غیرمستقیم کفایت می‌کند.

اجازه دهید که با ماژول مدیریت کلید آغاز کنیم (کد ٤-١٤ را مشاهده نمایید). زیرا قصد داریم از یک کلید ثابت استفاده نماییم. دیگر از یک کلید تصادفی که در مثال‌های قبل بکار برده شد استفاده نخواهد شد. KeyManager وظایف زیر را انجام خواهد داد:

١: یک کلید را به‌عنوان یک پارامتر می‌پذیرد (تابع setId(byte[] data))

٢: یک بردار اولیه را به‌عنوان یک پارامتر می‌پذیرد (تابع setIv(byte[] data))

٣: کلید را درون یک فایل در حافظه داخلی ذخیره کند.

٤: کلید را از یک فایل در حافظه داخلی بازیابی کند (تابع getId(byte[] data))

٥: IV را از یک فایل در حافظه داخلی بازیابی کند (تابع getIv(byte[] data))

کد ٤-١٤: ماژول KeyManager

```
package android.secure.programming;
```

```
import java.io.ByteArrayOutputStream;
import java.io.FileinputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import android.content.Context;
import android.util.Log;

public class KeyManager {
    private static final String TAG="KeyManager";
    private static final String filel ="id_value";
    private static final String file2 = "iv_value";

    private static Context ctx;

    public KeyManager(Context cntx) {
        ctx = cntx;
    }

    public void setid(byte[] data) {
        writer(data, file1);
    }
    public void set!v(byte[] data) {
        writer(data, file2);
    }
    public byte[] get!d()  {
        return reader(file1);
    }

    public byte[] get!v() {
        return reader(file2);
    }

    public byte[] reader(String file) {
        byte[] data= null;
        try {
            int bytesRead= O;
            FileinputStream fis=ctx.openFileinput(file);
            ByteArrayOutputStream bos=new ByteArrayOutputStream();
            byte[] b=new byte[1024];
            while ((bytesRead=fis.read(b)) !   =-1) {
                bos.write(b, o, bytesRead);
            }
            data=bos.toByteArray();
        } catch (FileNotFoundException e) {
            Log.e(TAG, "File not found  in getid()");
        } catch (IOException e) {
            Log.e(TAG, "IOException in setid(): "+e.getMessage());
        }
        return data;
    }

    public void writer(byte[] data, String file) {
```

```
        try {
            FileOutputStream fos=ctx.openFileOutput(file,
Context.MODE_PRIVATE);
            fos.write(data);
            fas.flush();
            fos. close();
        } catch (FileNotFoundException e) {
            Log.e(TAG, "File not found  in setid()");
        } catch (IOException e) {
            Log.e(TAG, "IOException in setid(): "+e.getMessage());
        }
    }

}
```

حال به بررسی ماژول Crypto می‌پردازیم (کد ٤-١٥ را مشاهده نمایید). این ماژول از رمزنگاری و رمزگشایی مراقبت می‌کند. یک متد armorEncrypt () و armorDecrypt () را به ماژول اضـــافه خواهیم‌کرد تا کار تبدیل داده‌های بایتی به داده‌های قابل‌مشاهده در مبنای ٦٤ و برعکس را راحت‌تر کند.

کد ٤-١٥: ماژول Cryptographic

```
package android.secure.programming;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import android.content.Context;
import android.util.Base64;
public class Crypto {
    private static final String engine = "AES";
    private static final String crypto = "AES/CBC/PKCS5Padding";
    private static Context ctx;
    public Crypto(Context cntx) {
        ctx = cntx;
    }
    public byte[] cipher(byte[] data, int mode)
    throws NoSuchAlgorithmException, NoSuchPaddingException,
    InvalidKeyException, IllegalBlockSizeException,
    BadPaddingException, InvalidAlgorithmParameterException {
        KeyManager km = new KeyManager(ctx);
        SecretKeySpec sks = new SecretKeySpec(km.getId(), engine);
        IvParameterSpec iv = new IvParameterSpec(km.getIv());
        Cipher c = Cipher.getInstance(crypto);
        c.init(mode, sks, iv);
        return c.doFinal(data);
    }
    public byte[] encrypt(byte[] data) throws InvalidKeyException,
    NoSuchAlgorithmException, NoSuchPaddingException,
```

```
    IllegalBlockSizeException, BadPaddingException,
    InvalidAlgorithmParameterException {
        return cipher(data, Cipher.ENCRYPT_MODE);
    }
    public byte[] decrypt(byte[] data) throws InvalidKeyException,
    NoSuchAlgorithmException, NoSuchPaddingException,
    IllegalBlockSizeException, BadPaddingException,
    InvalidAlgorithmParameterException {
        return cipher(data, Cipher.DECRYPT_MODE);
    }
    public String armorEncrypt(byte[] data) throws InvalidKeyException,
    NoSuchAlgorithmException, NoSuchPaddingException,
    IllegalBlockSizeException, BadPaddingException,
    InvalidAlgorithmParameterException {
        return Base64.encodeToString(encrypt(data), Base64.DEFAULT);
    }
    public String armorDecrypt(String data) throws InvalidKeyException,
    NoSuchAlgorithmException, NoSuchPaddingException,
    IllegalBlockSizeException, BadPaddingException,
    InvalidAlgorithmParameterException {
        return new String(decrypt(Base64.decode(data,
Base64.DEFAULT)));
    }
}
```

این دو فایل در هر برنامه‌ای که نیازمند ذخیره‌سازی داده به صورت رمز شده باشد، قابلیت استفاده را دارد. ابتدا اطمینان یابید که یک مقدار برای کلید خود و بردار اولیه دارید، سپس قبل از ذخیره داده متد رمزنگاری یا رمزگشایی را روی آن‌ها فراخوانی کنید. کد ١٦-٤ تغییرات موردنیاز برای کلاس StorageExample3 را نشان می‌دهد. به‌علاوه، کد ١٧-٤ و کد ١٨-٤ تغییرات موردنیاز برای فایل‌های StoreData و RetrieveData را به ترتیب نشان می‌دهد.

کد ١٦-٤: StorageExample3 جدید با رمزنگاری

```
package android.secure.programming;
import net.zenconsult.android.crypto.Crypto;
import net.zenconsult.android.crypto.KeyManager;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.EditText;
public class StorageExample3Activity extends AppCompatActivity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        String key = "12345678909876543212345678909876";
        String iv = "1234567890987654";
        KeyManager km = new KeyManager(getApplicationContext());
        km.setIv(iv.getBytes());
        km.setId(key.getBytes());
        // Store data
```

```
        Contact contact = new Contact();
        contact.setFirstName("Sheran");
        contact.setLastName("Gunasekera");
        contact.setEmail("sheran@zenconsult.net");
        contact.setPhone(" + 12120031337");
        ContactsDb db = new ContactsDb(getApplicationContext(),
"ContactsDb",
        null, 1);
        Log.i("SE3", String.valueOf(StoreData.store(new Crypto(
        getApplicationContext()), db, contact)));
        Contact c = RetrieveData.get(new
Crypto(getApplicationContext()), db);
        db.close();
        EditText ed = (EditText) findViewById(R.id.editText1);
        ed.setText(c.toString());
    }
}
```

کد ٤-١٧: کلاس StoreData تغییر داده‌شده

```
package android.secure.programming;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import net.zenconsult.android.crypto.Crypto;
import android.content.ContentValues;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
public class StoreData {
    public static long store(Crypto crypto, ContactsDb db, Contact
contact) {
        // Prepare values
        ContentValues values = new ContentValues();
        try {
            values.put("FIRSTNAME",
crypto.armorEncrypt(contact.getFirstName()
            .getBytes()));
            values.put("LASTNAME",
crypto.armorEncrypt(contact.getLastName()
            .getBytes()));
            values.put("EMAIL", crypto.armorEncrypt(contact.getEmail()
            .getBytes()));
            values.put("PHONE", crypto.armorEncrypt(contact.getPhone()
            .getBytes()));
            values.put("ADDRESS1", contact.getAddress1());
            values.put("ADDRESS2", contact.getAddress2());
        } catch (InvalidKeyException e) {
            Log.e("SE3", "Exception in StoreData: " + e.getMessage());
        } catch (NoSuchAlgorithmException e) {
            Log.e("SE3", "Exception in StoreData: " + e.getMessage());
        } catch (NoSuchPaddingException e) {
            Log.e("SE3", "Exception in StoreData: " + e.getMessage());
        } catch (IllegalBlockSizeException e) {
```

```
            Log.e("SE3", "Exception in StoreData: " + e.getMessage());
        } catch (BadPaddingException e) {
            Log.e("SE3", "Exception in StoreData: " + e.getMessage());
        } catch (InvalidAlgorithmParameterException e) {
            Log.e("SE3", "Exception in StoreData: " + e.getMessage());
        }
        SQLiteDatabase wdb = db.getWritableDatabase();
        return wdb.insert(ContactsDb.tblName, null, values);
    }
}
```

کد ٤-١٨: کلاس **RetrieveData** تغییر داده‌شده

```
package android.secure.programming;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import net.zenconsult.android.crypto.Crypto;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
public class RetrieveData {
    public static Contact get(Crypto crypto, ContactsDb db) {
        SQLiteDatabase rdb = db.getReadableDatabase();
        String[] cols = { "FIRSTNAME", "LASTNAME", "EMAIL", "PHONE" };
        Cursor results = rdb.query(ContactsDb.tblName, cols, "", null,
"", "",
        "");
        Contact c = new Contact();
        results.moveToLast();
        try {
            c.setFirstName(crypto.armorDecrypt(results.getString(0)));
            c.setLastName(crypto.armorDecrypt(results.getString(1)));
            c.setEmail(crypto.armorDecrypt(results.getString(2)));
            c.setPhone(crypto.armorDecrypt(results.getString(3)));
        } catch (InvalidKeyException e) {
            Log.e("SE3", "Exception in RetrieveData: " +
e.getMessage());
        } catch (NoSuchAlgorithmException e) {
            Log.e("SE3", "Exception in RetrieveData: " +
e.getMessage());
        } catch (NoSuchPaddingException e) {
            Log.e("SE3", "Exception in RetrieveData: " +
e.getMessage());
        } catch (IllegalBlockSizeException e) {
            Log.e("SE3", "Exception in RetrieveData: " +
e.getMessage());
        } catch (BadPaddingException e) {
            Log.e("SE3", "Exception in RetrieveData: " +
e.getMessage());
        } catch (InvalidAlgorithmParameterException e) {
```

```
            Log.e("SE3", "Exception in RetrieveData: " +
e.getMessage());
        }
        return c;
    }
}
```

تصویر ٤-١٤ تلاش برای دستیابی به پایگاه داده SQLite بدون رمزگشایی را نشان می‌دهد.

تصویر ٤-١٤: داده‌ها بدون رمزنگاری نمایش داده‌شده‌اند

خلاصه

در این فصل اصول اولیه رمزنگاری پوشش داده شد. مفاهیم PKI و شخص ثالث قابل‌اعتماد و همچنین ساختار PKI با توجه به اهداف بررسی گشت. سپس به سازوکار ساده رمزنگاری داده‌ها و اصطلاحات پرداخته شد. نکته قابل توجه آن است که رمزنگاری به‌سادگی انتخاب یک الگوریتم متقارن نیست و باید به جنبه‌های مختلفی مثل لایه گذاری و حالت‌های عملیات دقت نمود. سپس در رابطه با سازوکارهای مختلف ذخیره سازی داده‌ها روی اندروید مثال‌هایی مورد برر سی قرار گرفت و پایگاه داده‌های SQLite و SharedPreferences به دلیل توانایی در ذخیره انواع داده‌های برنامه برگزیده شدند. در انتها یک کتابخانه همه‌منظوره برای رمزنگاری و رمزگشایی آورده شد. این کتابخانه می‌تواند در هر یک از برنامه‌های

آینده که نیاز به ذخیره‌سازی داده‌ها دارد به کار رود.

۵ برنامه‌های وب

برنامه های کاربردی باید با برنامهٔ تحت وب ارتباط برقرار کنند. خواه برنامه در حال ارتباط با یک RESTful API[۱] باشـــد یا در حال تبادل اطلاعات با یک برنامهٔ تحت وب. به‌طور طبیعی، به‌عنوان یک توسعه‌دهنده، وظیفهٔ شماست که اطمینان حاصل کنید که تبادل اطلاعات به صورتی است که مهاجمان نتوانند به داده‌های خصوصی کاربر نهایی دسترسی پیدا کنند یا آن‌ها را تغییر دهند. فصل‌های قبل، بیشتر بر روی بررسی "داده‌های ثابت" و داده‌های ذخیره شده و رمزنگاری شده معطوف شدیم. در این فصل، "داده‌های در حال انتقال" پوشش داده خواهند شد.

معمولاً، SSL یا TLS امنیت داده‌های در حال انتقال را بر عهده‌دارند. اما کشف امکان نفوذ به یکی از منبع گواهی‌ها یا CA[۲] در هلند که DigiNotar نامیده می شود باعث شد تا دوباره این روش را موردبررسی قرار دهیم (برای اطلاعات بیشتر به سایت http://en.wikipedia.org/wiki/DigiNotar مراجعه کنید.) در پایان، تصمیم‌گیری برای امنیت داده‌های در حال انتقال به شما به‌عنوان توسعه‌دهنده واگذار خواهدشد. اما به‌طور واضح، این حمله باعث شده که حتی گواهی SSL نیز همیشه بهترین گزینه نباشد. بدین ترتیب، برخی از موضـــوعات مربوط به امنیت برنامه‌نویســـی تحت‌وب پوشـــش داده خواهند شـــد و اینکه چگونه نرم‌افزار موبایل باید با برنامه‌های کاربردی تحت وب تعامل برقرار کند. همچنین، به‌طور مختصر پروژه‌ی امنیتیِ برنامه تحت وب‌باز بررسی خواهد گردید. این‌یک مثال مناسب به منظورِ امن کردن برنامهٔ تحت وب است.

در نظر بگیرید که چگونه کد ۵-۱ امن شده است. حالا از خود بپرسید که برای امن‌تر شدن کد چه باید کرد؟ (در پایان فصـــل، نتایج را بررســـی کنید و کد خودتان را مقایســـه نمایید و ببینید که راهِ درســـت را رفته‌اید یا نه.)

کد ۵-۱: ورود مشتری

```
package android.secure.programming;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
```

[۱] Rest یک معماری برای ارتباط برنامه های کاربردی تحت شبکه می‌باشد در صورتی که SOAP یک پروتکل برای ارتباط برنامه های کاربردی تحت شبکه می‌باشد.

[۲] Certificate Authority

```
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import android.util.Log;
public class Login {
    private final String TAG = "HttpPost";
    public Login() {
    }
    public HttpResponse execute() {
        HttpClient client = new DefaultHttpClient();
        HttpPost post = new
HttpPost("http://logindemo1.appspot.com/logindemo");
        HttpResponse response = null;

        List < NameValuePair > nvPairs = new ArrayList < NameValuePair
> (2);
        nvPairs.add(new BasicNameValuePair("username", "android"));
        nvPairs.add(new BasicNameValuePair("password", "programming"));

        try {
            UrlEncodedFormEntity params = new
UrlEncodedFormEntity(nvPairs);
            post.setEntity(params);
            response = client.execute(post);
        } catch (UnsupportedEncodingException e) {
            Log.e(TAG, "Unsupported Encoding used");
        } catch (ClientProtocolException e) {
            Log.e(TAG, "Client Protocol Exception");
        } catch (IOException e) {
            Log.e(TAG, "IOException in HttpPost");
        }
        return response;
    }
}
```


۵-۱ آماده‌سازی محیط

ابتدا باید محیط آزمون را راه اندازی نمود. بدیهی اســـت به یک زیرســـاخت برای برنامه کاربردی که تحت وب ساخته شده با شد و همچنین به کدنویسی صورت گرفته در back-end[1] نیاز خواهد شد. بدین منظور، یک برنامهٔ کاربردی کوچک نوشـــته می‌شـــود. یک پروژه با نام LogiDemo و پکیجی با عنوان android.secure.programming ایجاد کنید درون پکیج نام‌گذاری شده یک فایل به نام LoginDemoServlet بسازید که فایل حاوی کدهای نشان داده شده در کد ۵-۲ است. این برنامه عملکرد ساده‌ای دارد؛ کد منتظر یک درخواست HTTP GET می‌ماند و سپس با متن سادهٔ "Hello, world" پاسخ می‌دهد.

[1] توسعه یک برنامه کاربردی به دو بخش back-end و front-end تقسیم می‌شود. Front-end بخشی از برنامه کاربردی می‌باشد که کاربر نهایی با آن تعامل دارد و back-end بخشی از برنامه کابردی است که برنامه سمت سرور و پایگاه داده در آن قرار دارد.

کد ۵-۲: پکیج برنامهٔ کاربردی کوچک به‌صورت پیش‌فرض، android.secure.programming

```
import java.io.IOException;
import javax.servlet.http.*;
@SuppressWarnings("serial")
public class LoginDemoServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws IOException {
        resp.setContentType("text/plain");
        resp.getWriter().println("Hello, world");
    }
}
```

با ابزار tomcat می‌توان کد را به‌صورت local اجرا و مشاهده نمود. زمانی که آدرس http://localhost:8080/logindemo بارگذاری شود، پاسخ "Hello, World" نمایش داده می‌شود. (تصویر ۵-۱ را ببینید).

تصویر ۵-۱: دسترسی به logindemoServlet

در حال حا ضر یک برنامهٔ کاربردی وب وجود دارد که می‌توان از آن را برای هر آنچه نیاز ا ست ا ستفاده کرد. اکنون نگاهی مختصر به تئوری‌های مربوط به برنامهٔ کاربردی تحت وب خواهیم انداخت.

۵-۲ HTML، برنامهٔ کاربردی تحت وب و خدمات وب

هر تو سعه‌دهنده‌ی وب می‌داند که HTML چی ست. HTML سازنده‌ی ا صلی هر وب سایت مدرن است. زبان نشانه‌گذاری ابرمتن (html) به‌صورت طرحی تفصیلی در سال ۱۹۹۱ حیات خود را شروع کرد؛ این زبان بسیار ساده به‌عنوان اهرمی برای ایجاد صفحات وبِ پایه استفاده می‌شود و در سال ۲۰۰۸ زمانی که طرحی برای نسخه‌ی HTML5 منتشر شد به سرعت ر شد کرد. صفحات HTML خالص به‌عنوان صفحات استاتیک شناخته می‌شوند. به‌عبارت‌دیگر، آن‌ها بر روی مرورگر کاربر نهایی ارائه می‌شوند و آنجا می‌مانند تا کاربر صفحهٔ دیگری را بازنماید.

یک برنامهٔ وب یک بخش از برنامهٔ کاربردی اســت که در آن کاربران نهایی به بیش از یک شــبکه دسترسی دارند–درست مثل صفحات وب. یک برنامهٔ وب، شامل عناصر پویای بیشتری نسبت به HTML

ساده است. برای مثال، برنامه‌های وب مدرن دارای تعداد زیادی زبان‌های سمت سرور هستند. این زبان‌ها(مانند: PHP، JSP، ASP و ...) بر اساس ورودی کاربر نهایی یک صفحه html ایستا به‌صورت بلادرنگ تولید می‌کنند. برنامهٔ وب روی یک وب سرور اجرا می‌شود و روی یک سخت‌افزار میزبانی می‌شود که کاربر نهایی می‌تواند از یک شبکه مثل اینترنت به آن دسترسی داشته باشد. چارچوب برنامهٔ سمت سرور از رابط کاربری، منطق برنامه‌ای(برای مثال جست جو، محاسبه یا هر فرایند دیگری)، و ذخیره سازی داده یا توابع بازیابی مراقبت می‌کند. ازآنجا که تمام پردازش‌های پیچیده، در back end یا سمت سرور قرارگرفته‌اند یک مرورگر وب ساده چیزی بیشتر از یک تعامل با رابط کاربری نیست[1].

برنامه‌های تحت وب مزایایی را به توسعه‌دهندگان ارائه می‌دهند. یکی از بزرگ‌ترین مزایای آن توانایی انجام بروزرسانی‌ها یا تعمیرات سرور است. ازاین‌رو نباید نگران بروزرسانی صدها یا هزاران نفر از مشتریان بود. یکی دیگر از مزایای بزرگ برنامه‌های کاربردی وب این است که کاربران نهایی به یک سرویس‌گیرنده‌ی سبک–مرورگر وب– نیاز دارند. بدین ترتیب، نه‌تنها می‌توانید تعداد زیادی از کاربران رایانه‌های شخصی را پشتیبانی کنید، بلکه می‌توانید کاربران موبایل را نیز پشتیبانی نمایید.

یک سرویس وب شبیه به برنامهٔ وب است که می‌توان به آن از طریق یک شبکه راه دور دسترسی داشت. تفاوت اصلی در این است که کاربران دسترسی تعاملی مستقیم با سرویس ندارند. یک وب‌سرویس در اکثر موارد، قادر به توصیفِ سرویس‌هایی که ارائه می‌دهد است و اینکه چگونه دیگر برنامه‌های کاربردی می‌توانند به آن‌ها دسترسی داشته باشند. بدین منظور سرویس‌وب از فایل زبان توصیف سرویس وب (WSDL[2]) استفاده می‌کند. برنامه‌های کاربردی دیگر با استفاده از فایل WSDL که منتشر شده است می‌توانند چگونگی کار با سرویس وب را درک کنند. به‌طورکلی، سرویس وب از فرمت xml[3] خاص برای تبادل اطلاعات استفاده می‌کند. یکی از پروتکل‌های معروف، پروتکل دسترسی آسان به اشیاء (SOAP[4]) است. پروتکل SOAP از فایل‌های xml مختلف بر اساس برنامه‌های کاربردی خاص ساخته‌شده است. نمونه‌ای از یک پیام SOAP در کد ۵-۳ نشان داده‌شده است.

کد ۵-۳: نمونه‌ای از پیام SOAP(گرفته‌شده از ویکی‌پدیا)

```
POST /InStock HTTP/1.1
Host: www.example.org
```

[1] در سال های اخیر با معرفی Angular کمی این روند تغییر یافت که در کتاب فعلی مورد بحث قرار نمی‌گیرد.

[2] Web Service Definition Language

[3] eXtensible Markup Language

[4] Simple Object Access Protocol

```
Content-Type: application/soap + xml; charset = utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"
<?xml version = "1.0"?>
<soap:Envelope xmlns:soap = "http://www.w3.org/2003/05/soap-envelope">
<soap:Body>
      <m:GetStockPrice xmlns:m = "http://www.example.org/stock">
            <m:StockName > IBM</m:StockName>
      </m:GetStockPrice>
</soap:Body>
</soap:Envelope>
```

راه دیگری که وب‌سرویس‌ها می‌توانند کار کنند، ارائه RESTful API است. معماری REST[1]، یک معماری بنیادی و stateless است که از پروتکل Client-Server برای ارتباط با سرویس‌های وب استفاده می‌کند. REST به‌جای تکیه‌بر پروتکل‌های پیچیده (مانندSOAP که از یک URI و پارامترهای متعدد ا ستفاده می‌کند)، از یک امکان بسیار ساده‌و قابل‌دسترس (مانند HTTP) با URI های جداگانه برای هر یک از منابع استفاده می‌کند.

شما می‌توانید اطلاعات بیشتر در مورد REST را در رسالهٔ Roy Fielding در سایت www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm یا ویکی‌پدیا به نشانی http://en.wikipedia.org/wiki/Representational_state_transfer مطالعه کنید. اگر چه استفاده از وب‌سرویس RESTful ساده است، توانایی انجام وظایفی مشابه‌وظایف SOAP را داراست. به مثال SOAP در کد ۳-۵ نگاه کنید. اگر سرویس وب بخواهد همانند این کد را به‌عنوان RESTful API ارائه دهد، باید برای دسترسی به منبع موردنظر (در اینجا لیست قیمت کالاهای IBM) از URI زیر استفاده نمود:

http://www.example.com/stocks/price/IBM

گاهی اوقات، سرورها می‌توانند داده‌ها را به روش‌های مختلف برگردانند. برای مثال، اگر از سرور خروجی xml درخواست شود، می‌توان پسوند xml را به URI اضافه کرد و همین‌طور اگر خروجی JSON[2] را درخواست شود باید پسوند json را اضافه شود، همانند زیر:

http://www.example.com/stocks/price/IBM.xml
http://www.example.com/stocks/price/IBM.json

اکنون زمان خوبی است که در موردHTTP (پروتکل انتقال ابرمتن) صحبت شود. HTTP پروتکلی

[1] Representational State Transfer

[2] JavaScript Object Notation

است که وب را گسترش و رشد داد. قبلاً ابرمتن به HTML ساده و قدیمی اشاره می‌کرد، ولی در حال حاضر توانسته است تا پشتیبانی از xml (زبان نشانه‌گذاری توسعه‌پذیر) را شامل شود. Xml از قواعد HTTP پیروی می‌کند؛ ولی شامل تگ‌های سفارشیHTML (یا کلمات کلیدی) است که می‌تواند مورداستفاده قرار بگیرد. HTTP مانند پروتکل درخواست و پاسخ بین Client-Server عمل می‌کند. Client، یا عامل کاربر(مرورگر وب) یک درخواست به وب سرور می‌دهد و پاسخ را به صورت HTML یا xml دریافت می‌کند.

درخواست‌های HTTP انواع درخواست‌ها را شامل می شوند، درحالی‌که چندین روش درخواست وجود دارد، پرکاربردترین آنها، GET و POST است. درخواست GET برای بازیابی داده‌ها و درخواست POSTبرای ارائه‌ی داده‌ها استفاده می شوند. در حال پر کردن فرم ثبت‌نام اگر دکمهٔ ثبت را در صفحه‌ای که درمرورگر نمایش داده شده است را کلیک کنید داده به سرور ارسال می‌گردد. با توجه به کد ۵-۱ در آغاز فصل، خط زیر مشاهده خواهد شد:

```
HttpPost post = new HttpPost("http://logindemo1.appspot.com/logindemo");
```

قطعه کد بالا ایجاد یک درخواست HTTP POST به یک آدرس خاص است. همان‌طور که احتمالاً می‌دانید، یکURL[1] که در سمت مرورگر فراخوانی می شود به سرور می‌گوید که یک منبع خاص را برایش بازیابی نماید. منابع می‌توانند فایل‌ها، اسناد یا اشیای ذخیره‌شده بر روی سرور از راه دور باشند. درخواست‌ها و پاسخ‌های HTTP هر دو ساختار مشابهی دارند. هردو حاوی سرآیند و محتوا هستند. به منظور توضیحات بیشتر در مورد HTTP به سایت www.w3.org مراجعه نمایید.

۵-۲-۱ اجزاء یک برنامهٔ کاربردی تحت وب

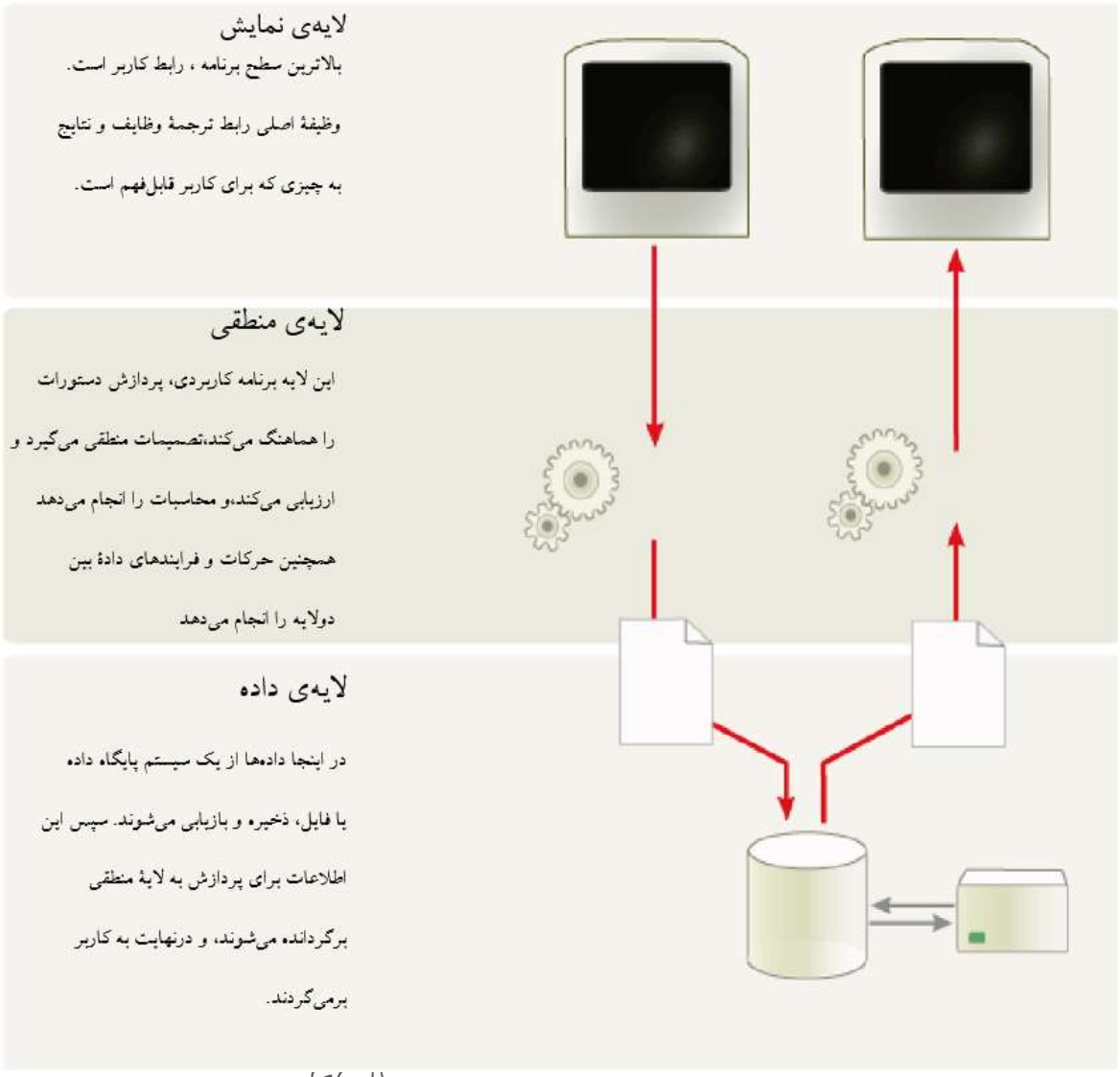
برنامه‌های کاربردی وب از لایه‌های مختلف تشکیل شده‌اند. برنامه‌های کاربردی وب معمولا سه لایه دارند(تصویر ۵-۲ را ببینید): لایه‌ی نمایش[2]، لایه‌ی منطقی[3] و لایه‌ی داده[4]. بر اساس الزامات و پیچیدگی یک برنامه کاربردی، تعداد لایه‌ها می‌توانند افزایش یابند.

[1] Uniform Resource Locator

[2] Presentation Tier

[3] Application Tier

[4] Data Tier



تصویر ۵-۲: برنامه وب سه لایه

مزایای زیادی برای داشتن برنامه‌های کاربردی چندلایه وجود دارد: یکی از آن‌ها این است که صاحبان سیستم می‌توانند پیکربندی سخت‌افزار را مستقل از دیگر لایه‌ها انجام دهند. سناریویی را در نظر بگیرید که یک شـــرکت نیاز دارد تا مقدار ذخیره‌ســـازی داده را افزایش دهد. بخش IT می‌تواند این لایه را بدون ایجاد تغییراتِ قابل‌توجهی در لایه‌های دیگر ارتقاء دهد. مزیت بعدی این است که گروه‌های امنیتی می‌توانند در هر لایه کنترل مجزا داشته باشند. هر لایه عملکردهای مختلفی دارد، و درنتیجه مجموعه‌ای متفاوت از الزامات و کنترل‌های امنیتی مرتبط مورد نیاز اسـت. برنامه کاربردی چندلایه به صـاحبش اجازه می‌دهد تا کنترل مجزای بیشتری بر لایه‌ها برای مدیریت اشکالات داشته باشد. بنابراین بر اساس معماری سه لایه، یک برنامهٔ تحت وب موارد زیر را شامل می‌شود:

- یک وب سرور برای داده‌های موجود
- یک سرور برنامه کاربردی برای کنترل تمام درخواست‌ها به منظور تبادل داده‌ها
- یک سرور پایگاه داده که داده‌ها را ذخیره و بازیابی می‌کند.

اجازه دهید با در نظر گرفتن یک مثال، چگونگیِ ارتباط بین لایه‌ها را ببینیم.

فرایند ورود

احراز هویت کاربر که در یک نشست[1] مشتری با سرور رخ می دهد، چیزی شبیه به آنچه در زیر آمده است:

- مشتری صفحه ورود را از وب سرور درخواست می‌کند.[وب سرور/لایهٔ نمایش]
- مشتری گواهی‌ها را به وب سرور ارسال می‌کند.[وب سرور/لایهٔ نمایش]
- سرور داده را دریافت می‌کند و بررسی می‌کند که آیا با قوانین اعتبار سنجی مطابقت دارد یا نه[سرور برنامهٔ کاربردی/لایهٔ منطقی]
- اگر داده‌ها خوب باشند، سرور برنامه کاربردی، از پایگاه داده پرس‌وجو می‌کند تا از وجود یا عدم وجود گواهی مطلع گردد. [سرور برنامهٔ کاربردی/لایهٔ منطقی]
- سرور پایگاه داده به سرور برنامهٔ کاربردی پاسخ موفقیت یا شکست را می‌دهد[سرور پایگاه داده/لایهٔ داده]
- سرور برنامهٔ کاربردی با وب سرور گفت گو می‌کند، تا در صورتی‌که گواهی درست بود پورتال را به مشتری ارائه دهد یا اگر گواهی مطابقت نداشت پیام خطا دهد.[سرور برنامهٔ کاربردی/لایهٔ منطقی]
- وب سرور پیامی از سرور برنامهٔ کاربردی نمایش می‌دهد[وب سرور/لایهٔ نمایش]

این یک مثال ساده است و چگونگی فرایند درخواست و پاسخ را از بیرون به داخل و بالعکس نشان می‌دهد.

۲-۲-۵ فناوری برنامهٔ وب

فناوری های متعددی وجود دارند که می‌توان برای هر لایه از برنامهٔ وب استفاده کرد. می‌توان چارچوب برنامه، سرورِ برنامهٔ کاربردی، زبان‌های برنامه‌نویسی سمت سرور و سرورهای پایگاه داده را انتخاب کرد. معیارهای انتخاب به عواملی مانند برنامهٔ کاربردی موردنیاز، بودجه، و قابلیت پشتیبانی از

[1] Session

فناوری بـستگی دارد. ازآنجاکه تو سعهٔ اندروید عمدتاً روی جاوا انجام می شود، برای برنامهٔ کاربردی وب از جاوا بهره خواهیم برد. به‌غیراز جاوا، می‌توان از دیگر فناوری‌های سمت سرور استفاده کرد که برخی از آن‌ها در زیر آورده شده:

PHP: www.php.net
Python: www.python.org
Django: www.djangoproject.com
Perl: www.perl.org (معمولا کمتر استفاده می‌شود)
Cold Fusion: www.adobe.com/product/coldfusion-family.html
ASP.NET: www.asp.net
Ruby on Rails: www.rubyonrails.org

به‌طور مشـــابه‌بســـته به نیاز، می‌توان از تعداد زیادی از پایگاه داده‌های محبوب برای برنامه در لایهٔ داده بهره جست. پایگاه داده‌های رایگان و تجاری زیادی وجود دارد. این‌یکی از تصمیمات عمده‌ای است که شما یا معماری برنامهٔ شما باید ابتدا ایجاد کند. اینجا تعداد کمی از پایگاه داده‌های محبوب نام برده شده است که می‌توانید بیشتر در مورد آن‌ها اطلاعات کسب نمایید:

Oracle: www.oracle.com
Microsoft SQL Server: www.microsoft.com/sqlserver
MySQL: www.mysql.com
PostgreSQL: www.postgresql.org
CouchDB: http://couchdb.apache.org
MongoDB: www.mongodb.org

اکنون به تکمیل برنامهٔ وب خواهیم پرداخت، این برنامه از فرایند برر سی رمز عبورِ ابتدایی پ شتیبانی می‌کند. توجه داشته باشید که مثال‌ها بسیار ساده‌اند. روال احراز هویت برای برنامه‌های وب واقعی، پیچیده‌تر خواهد بود. کد ٥-٤ مشاهده شود.

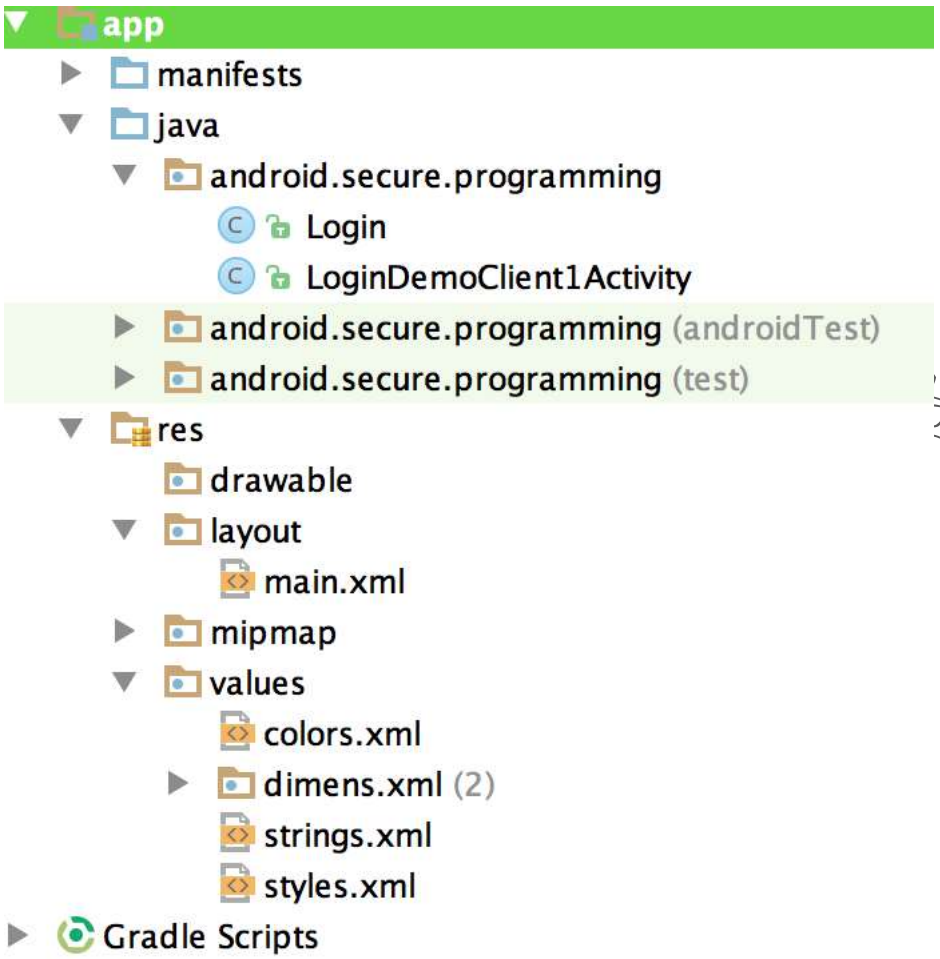
کد ٥-٤: کد تائید گواهی جدید

```
package android.secure.programming;
import java.io.IOException;
import javax.servlet.http.*;
@SuppressWarnings("serial")
public class LoginDemoServlet extends HttpServlet {
    private String username = "sheran";
    private String password = "s3kr3tc0dez"; // Hardcoded here intended
to
    // simulate a database fetch
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws IOException {
        resp.setContentType("text/plain");
        resp.getWriter().println("Hello, world");
    }
    public void doPost(HttpServletRequest req, HttpServletResponse
resp)
    throws IOException {
        String user = req.getParameter("username"); // No user input
```

```
validation
        // here!
        String pass = req.getParameter("password"); // No user input
validation
        // here!
        resp.setContentType("text/plain");
        if (user.equals(username) && pass.equals(password)) {
            resp.getWriter().println("Login success!!");
        } else {
            resp.getWriter().println("Login failed!!");
        }
    }
}
```

در گام بعد کدها باید در localhost اجرا شـوند. حال یک پروژه‌ی جدید اندروید که احراز هویت را کنترل می‌کند ایجاد خواهیم کرد(تصویر ۵-۳ را که ساختار پروژه را نمایش می‌دهد ببینید). کد ۵-۵، کد ۵-۶، کد ۵-۷ و کد ۵-۸ به ترتیب شـــامل کدLogin، LoginDemoClient1Activity ، strings.xml و فایل main.xml است. همچنین برای اتصال به وب سرور نیاز به دسترسی به اینترنت خواهید داشت به همین منظور از اضافه شدن کد زیر به فایل AndroidManifest.xml اطمینان حاصل نمایید:

```
<uses-permission android:name = "android.permission.INTERNET" > </uses-permission>
```



تصویر ۵-۳: ساختار پروژه

کد ۵-۵: کلاس Login

```
package android.secure.programming;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import android.util.Log;
public class Login {
    private final String TAG = "HttpPost";
    private String username;
    private String password;
    public Login(String user, String pass) {
        username = user;
```

```
        password = pass;
    }
    public HttpResponse execute() {
        Log.i(TAG, "Execute Called");
        HttpClient client = new DefaultHttpClient();
        HttpPost post = new HttpPost("http://[ip
server]:8080/logindemo");
        HttpResponse response = null;
        // Post data with number of parameters
        List < NameValuePair > nvPairs = new ArrayList < NameValuePair
> (2);
        nvPairs.add(new BasicNameValuePair("username", username));
        nvPairs.add(new BasicNameValuePair("password", password));
        // Add post data to http post
        try {
            UrlEncodedFormEntity params = new
UrlEncodedFormEntity(nvPairs);
            post.setEntity(params);
            response = client.execute(post);
            Log.i(TAG, "After client.execute()");
        } catch (UnsupportedEncodingException e) {
            Log.e(TAG, "Unsupported Encoding used");
        } catch (ClientProtocolException e) {
            Log.e(TAG, "Client Protocol Exception");
        } catch (IOException e) {
            Log.e(TAG, "IOException in HttpPost");
        }
        return response;
    }
}
```

کد ۵-۵ روال Login را شامل می‌شود. ساختار کلاس Login، دو پارامتر می‌گیرد که شامل username و password هستند. متد ()execute از این پارامترها برای ساختن یک درخواست HTTP POST برای کاربر استفاده می‌کند.

کد ۵-۶: کلاس LoginDemoClient1Activity

```
package android.secure.programming;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class LoginDemoClient1Activity extends AppCompatActivity
implements
OnClickListener {
    private final String TAG = "LoginDemo1";
    private HttpResponse response;
    private Login login;
    /** Called when the activity is first created. */
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.login);
        button.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        String u = ((EditText) findViewById(R.id.username)).toString();
        String p = ((EditText) findViewById(R.id.password)).toString();
        login = new Login(u, p);
        String msg = "";
        EditText text = (EditText) findViewById(R.id.editText1);
        text.setText(msg);
        response = login.execute();
        Log.i(TAG, "After login.execute()");
        if (response ! = null) {
            if (response.getStatusLine().getStatusCode() ==
HttpStatus.SC_OK) {
                try {
                    BufferedReader reader = new BufferedReader(
                    new InputStreamReader(response.getEntity()
                    .getContent()));
                    StringBuilder sb = new StringBuilder();
                    String line;
                    while ((line = reader.readLine()) ! = null) {
                        sb.append(line);
                    }
                    msg = sb.toString();
                } catch (IOException e) {
                    Log.e(TAG, "IO Exception in reading from stream.");
                }
            } else {
                msg = "Status code other than HTTP 200 received";
            }
        } else {
            msg = "Response is null";
        }
        text.setText(msg);
    }
}
```

کد ۵-۶یک فعالیت اندروید ا ستاندارد ا ست. این را می‌توان به‌عنوان ورودی برنامه یا نقطه شروع در نظر گرفت.

کد ۵-۷: فایل strings.xml

```
<?xml version = "1.0" encoding = "utf-8"?>
<resources>
<string name = "hello" > Web Application response:</string>
<string name = "app_name" > LoginDemoClient1</string>
<string name = "username" > Username</string>
<string name = "password" > Password</string>
<string name = "login" > Login</string>
</resources>
```

کد ۵-۸: فایل main.xml

```
<?xml version = "1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android =
"http://schemas.android.com/apk/res/android"
android:orientation = "vertical"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:weightSum = "1">
<TextView android:textAppearance = "?android:attr/textAppearanceLarge"
android:id = "@ + id/textView1" android:layout_height = "wrap_content"
android:layout_width = "wrap_content" android:text =
"@string/username">
</TextView>
<EditText android:layout_height = "wrap_content"
android:layout_width = "match_parent" android:id = "@ + id/username">
</EditText>
<TextView android:textAppearance = "?android:attr/textAppearanceLarge"
android:id = "@ + id/textView2" android:layout_height = "wrap_content"
android:layout_width = "wrap_content" android:text =
"@string/password">
</TextView>
<EditText android:layout_height = "wrap_content"
android:layout_width = "match_parent" android:inputType =
"textPassword"
android:id = "@ + id/password">
</EditText>
<Button android:text = "@string/login" android:layout_height =
"wrap_content"
android:layout_width = "166dp" android:id = "@ + id/login">
</Button>
<TextView android:text = "@string/hello" android:layout_height =
"wrap_content"
android:layout_width = "fill_parent">
</TextView>
<EditText android:id = "@ + id/editText1" android:layout_height =
"wrap_content"
android:layout_width = "match_parent" android:inputType =
"textMultiLine"
android:layout_weight = "0.13">
<requestFocus > </requestFocus>
</EditText>
</LinearLayout>
```

فایل‌های string.xml و main.xml به ترتیب شامل مجموعه‌ای از ثابت‌های رشتهٔ تعریف شده و لایهٔ گرافیکی از برنامه می‌باشند.

برنامهٔ را اجرا کنید و نام کاربری و رمز عبورهای متعددی را وارد نمایید. باید دو پیام پاسخ مجزا نمایش داده شود، یکی برای موفقیت و یکی برای عدم موفقیت رمز عبور(تصویر ۵-٤). اکنون هردو برنامه‌ی کلاینت و سرور login نوشته شده است.

تصویر ۵-۶ عدم موفقیت لاگین

۵-۳ OWASP و حملات وب

OWASP[1] سازمانی است که دانش، فن‌ها و دستورالعمل‌هایی را برای آزمون و تأمین امنیت برنامه‌های کاربردیِ تحت وب و سایر بسترها را فراهم می‌سازد. OWASP در دسامبر سال ۲۰۰۱ تأسیس شد. "بهبود و ارتقا امنیت نرم‌افزارها" به‌عنوان هدف اصلی در فهرست آن‌ها ثبت شده است. یک منبع بزرگ برای یادگیری است و همچنین روش‌های امن سازی برنامهٔ کاربردی را ارائه می‌دهد. از سال ۲۰۰۴ پروژه ده خطر برتر OWASP یک پروژهٔ فرعی از پایهٔ OWASP بوده است. OWASP top ten ده مورد از مهم‌ترین آسیب‌پذیری‌های امنیتی را فهرست می‌کند. این آسیب‌پذیری‌ها توسط یک اجتماع گسترده از مدیران پروژه و کارشناسان امنیتی برنامه‌های کاربردیِ تحت وب در سطح جهان فهرست شده‌اند. لیست ده خطر برتر (top ten list) توسط تعداد زیادی از سازمان‌های تجاری استفاده و تصویب شده است، و به یک استاندارد برای امنیتِ برنامهٔ کاربردیِ وب تبدیل گشت. OWASP top ten 2013 آخرین به‌روزرسانی است که می‌توان آن را در سایت https://www.owasp.org/index.php/Top_10_2013-Top_10 دید.

موضوعات در مورد OWASP top ten 2013 در زیر لیست شده‌اند:

[1] Open Web Application Security Project

- A1 Injection
- A2 Broken Authentication and Session Management
- A3 Cross-Site Scripting (XSS)
- A4 Insecure Direct Object References
- A5 Security Misconfiguration
- A6 Sensitive Data Exposure
- A7 Missing Function Level Access Control
- A8 Cross-Site Request Forgery (CSRF)
- A9 Using Components with Known Vulnerabilities
- A10 Unvalidated Redirects and Forwards

لیست بالا یک لیست از راهنمایی‌های عملی در وب سایت است که به توسعه‌دهندگان وب کمک بسیار زیادی می‌کند. یکی از جدیدترین پروژه‌های OWASP، Mobile top ten است، که بخشی از پروژه امنیتی موبایل OWASP است. این پروژه در فصل ۷ بررسی می‌گردد. Mobile top ten بسیاری از موضوعات مطرح‌شده در این فصل را شامل می‌شود و همچنین فن‌ها و اصول زیادی را انتقال می‌دهد. در زیر موضوعات پوشش داده‌شده لیست شده است:

- شناسایی و محافظت از داده‌های حساس بر روی دستگاه موبایل
- رسیدگی به گواهی رمز عبور امنیتی روی دستگاه
- اطمینان از اینکه داده‌های در حال انتقال محافظت می‌شوند
- پیاده‌سازی درستِ گواهی یا مجوز کاربر و مدیریت نشست
- امن نگه‌داشتن API های back end (سرویس‌ها) و پلتفرم (سرور)
- یکپارچه کردن داده‌ها با سرویس‌ها یا برنامه‌های شخص ثالث به‌صورت امن
- توجه خاص کردن به جمع‌آوری و ذخیره‌سازی موفق برای جمع‌آوری و استفاده از داده‌های کاربر
- انجام کنترل برای جلوگیری از دسترسی غیرمجاز به منابع مالی (به‌عنوان‌مثال، کیف پول، sms، تماس‌های تلفنی)
- اطمینان از امنیتِ توزیع یا تأمین برنامه‌های کاربردیِ موبایل
- با دقت بررسی کردن همه‌ی توضیحات خطای زمان اجرایِ کد

٤-٥ تکنیک های احراز هویت

احراز هویت، ویژگیِ مهم برنامه‌های کاربردی موبایل است که برای ارتباط با برنامه‌های کاربردیِ وب نیاز است. تقریباً تمام برنامه‌های امروزی برای اجازهٔ دسترسی به داده‌هایشان، وابسته به ترکیب نام کاربری و

رمز عبور یا پین هستند. نام کاربری و رمز عبور بر روی سرور ذخیره شده است، و هر زمان برنامه کاربردی بخواهد یک کاربر نهایی را احراز هویت کند، یک مقایسه انجام خواهد داد. اگر به کد ۵-۱ دوباره نگاه بیندازید، می‌بینید که این کار دقیقاً انجام‌شده است. کدهای زیر شامل نام کاربری و رمز عبور برای برنامهٔ کاربردی وب است:

```
nvPairs.add(new BasicNameValuePair("username", "android"));
nvPairs.add(new BasicNameValuePair("password", "programming"));
```

در این مثال، اطلاعات به صورت قوی رمز شده‌اند، اما هر وقت که کاربر بخواهد وارد شود به‌راحتی می‌تواند از روی دستگاه بازیابی شود. اما چه اتفاقی میافتد اگر ارتباط ما در زمان انتقال شنود شود؟ شاید بگویید در ارتباط از پروتکل SSL[1] استفاده‌شده است، اما به نظر نمی‌رسد که از آن در مثال ذکر شده استفاده شده باشد، زیرا درخواست POST به یک پورت به‌غیراز SSL/TLS[2] می‌رود:

```
HttpPost post = new HttpPost("http://logindemo1.appspot.com/logindemo");
```

اجازه دهید در نظر بگیریم که امنیت ترافیک SSL نقض شده است. مهاجم که در حال استراق سمع روی گفت‌وگوی ما با برنامهٔ کاربردی وب است، حالا می‌تواند به اطلاعات کاربری دسترسی داشته باشد و تمام‌کاری که او باید انجام دهد، استفاده از آن اطلاعات روی برنامهٔ تحت وب یا دستگاه موبایل دیگر است. در این صورت او می‌تواند کنترل کاملی بر روی پروفایل کاربری داشته باشد. تاکنون، با خطراتی که ممکن است هنگام تصدیق هویتِ راه دور رخ دهد، آشنا شده‌ایم. هرچند ممکن است اطلاعات از یک کانال امن بگذرد، اما هنوز در معرض حملات قرار دارد. و حتماً نباید حملهٔ شدیدی مانند DigiNotar با شد، که یک مهاجم می‌تواند گواهی خودش را صادر کند. برای مثال، می‌تواند حملهٔ SSL man-in-the-middle باشد.

نمی‌توان همیشه به SSL اعتماد کرد. به‌طورکلی، کاربران نهایی فکر می‌کنند که SSL به معنی این است که آن‌ها در امنیت هستند. یک آیکون قفل و نوار آدرس روی مرورگر، شاخص‌هایی هستند که می‌گویند در حال بازدید از یک سایت به‌صورت امن هستید. این لزوماً درست نیست، بااین‌حال، به برخی از مفاهیم SSL خواهیم پرداخت.

SSL (لایهٔ سوکت امن) یک پروتکل انتقال است که داده‌هایی که بین دو رایانه در حال انتقال است را رمزنگاری می‌کند. یک استراق سمع کننده نمی‌تواند حداقل بدون انجام بعضی از تلاش‌ها داده‌های رمزنگاری‌شده را ره‌گیری کند. بنابراین، SSL تضمین می‌کند که داده‌ها بین کلاینت و سرور محرمانه می‌ماند. SSL یک پروتکل قدیمی است. بسیاری از افراد انتقال داده با HTTP رمز شده بین کلاینت سرور را نسبت

[1] Secure Sockets Layer

[2] Transport Layer Security

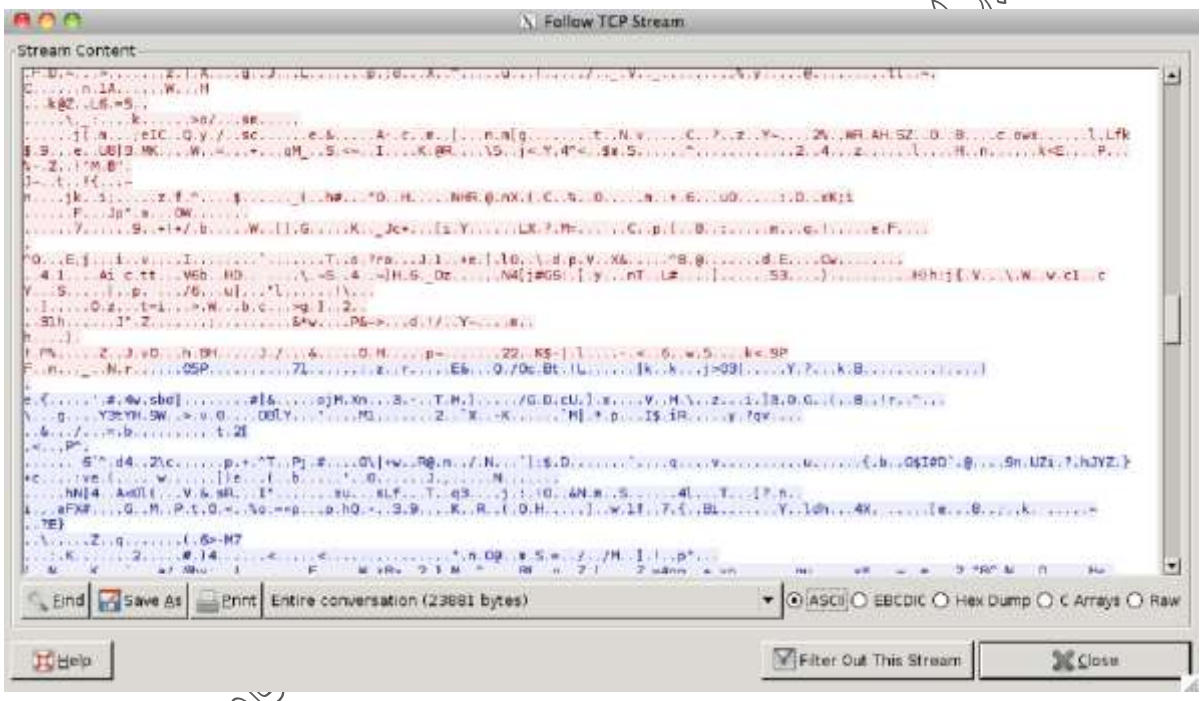
به SSL ترجیح می‌دهند. جدیدترین نسخه SSL، پروتکل TLS[1](امنیت لایهٔ انتقال) است. بخش جدایی‌ناپذیر از SSL و TLS گواهیِ X.509 است. X.509 استانداردِ زیرساخت کلید عمومی (PKI) است که به‌طور خلاصه در فصل ۳ پوشش داده شد. معمولاً، کاربران گواهی سرور X.509 را به‌عنوان گواهی SSL می‌شناسند. این گواهی جزءِ بسیار مهم SSL است. تصویر ۵-۵ راه‌اندازی مرورگر یک نشست SSL را نشان می‌دهد.

[1] پروتکل TLS به عنوان نسخه سوم SSL معرفی شده است.

تصویر ۵-۵: راه‌اندازی نشست SSL/TLS

TLSو SSL ترکیبی از فن‌های رمزنگاری را برای اطمینان از انتقال داده‌های امن اســـتفاده می‌کنند. اکنون اجازه دهید به راه‌اندازی نشست بپردازیم. نیازی به طرح جزییات نیست، زیرا نیازی به اختراع دوباره چرخ و بازنویسی الگوریتم TLS نیست. در عوض، در این بخش با ایده‌هایی از چگونگی رمزنگاری در طول نشست TLS آشنا خواهیم شد. ابتدا، کلاینت با یک وب سرور تماس می‌گیرد و بعضی اطلاعات را به آن می‌فرستد. اطلاعات شامل جزئیاتی از نسخه TLS است که شامل یک فهرستی از الگوریتم‌های رمزنگاری است که توسط نسخه TLS استفاده شده در client پشتیبانی می شود. این‌ها مجموعه رمز نامیده می‌شوند و آن‌ها شـامل الگوریتم‌های پشـتیبانی شـده برای کارهای مختلفی مانند تعویض کلید، احراز هویت و رمزهای انبوه است. سپس، سرور پس از انتخاب مجموعه رمز خاص که پشتیبانی می‌کند و بالاترین نسخهٔ رایج TLS که توسط کلاینت و سرور پشتیبانی می‌شود، پاسخ می‌دهد. سرور گواهیSSL خود را به کلاینت می‌فرستد. کلاینت، از کلید عمومی سـرور اسـتفاده می‌کند تا یک کلید موقت را رمزنگاری و تبادل کند، کلیدی که یک کلید اصلی را تولید می‌کند. هنگامی‌که کلید موقت مبادله می شود، کلاینت و سرور از مقادیر تصادفی و کلید موقت برای تولید یک کلید اصـــلیِ نهایی اســـتفاده می‌کنند. این کلید اصـــلی روی کلاینت و ســـرور ذخیره

می‌شود. سرور و کلاینت همه داده‌ها را با کلید اصلی رمز و ارسال می‌کنند. مجموعه رمز انتخاب‌شده و کلید اصلی متقارن در هر دو طرف برای رمزنگاری و رمزگشایی داده‌ها استفاده می‌شود. تصویر ۵-۶ نشان می‌دهد که اگر شما قادر به ایجاد یک نشست رمزنگاری شده بین کلاینت و سرور باشید چه چیزی را خواهید دید. تصویر ۵-۷ فرایند دستکانی و دیگر جزئیات مربوطه که در هنگام استفاده از openssl مشاهده می‌شود، نشان داده است. در یک نگاه سریع به شما می‌گوید که داده‌ها به‌هیچ‌وجه برای یک مهاجم قابل‌استفاده نیست. پس، این برای شما به‌عنوان یک توسعه‌دهنده به چه معنی است؟ آیا شما باید از SSL استفاده کنید و نباید هرگز نسبت به چشم‌های کنجکاو زمانی که داده‌های حساس را بین کلاینت و سرور تبادل می‌کنید نگران باشید؟ ابتدا اجازه دهید به جزئیات نگاهی بیاندازیم و سپس پاسخ آن را خواهیم یافت.



تصویر ۵-۶: ضبط ترافیک از یک نشست SSL

```
azazel:~ sheran$ openssl s_client -connect mail.google.com:443
CONNECTED(00000003)
depth=1 /C=ZA/O=Thawte Consulting (Pty) Ltd./CN=Thawte SGC CA
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=mail.google.com
   i:/C=ZA/O=Thawte Consulting (Pty) Ltd./CN=Thawte SGC CA
 1 s:/C=ZA/O=Thawte Consulting (Pty) Ltd./CN=Thawte SGC CA
   i:/C=US/O=VeriSign, Inc./OU=Class 3 Public Primary Certification Authority
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIDIjCCAougAwIBAgIQHxn23jXdY6FCkYrVLMCrEjANBgkqhkiG9w0BAQUFADBM
MQswCQYDVQQGEwJaQTElMCMGA1UEChMcVGhhd3RlIENvbnN1bHRpbmcgKFB0eSkg
THRkLjEWMBQGA1UEAxMNVGhhdJRlIFNHQyBDQTAeFw0wOTEyMTgwMDAwMDBaFw0x
MTEyMTgyMzU5NTlaMGkxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlh
MRYwFAYDVQQHFA1Nb3VudGFpbiBWaWV3MRMwEQYDVQQKFApHb29nbGUgSW5jMRgw
FgYDVQQDFA9tYWlsLmdvb2dsZS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJ
AoGBANknyBHye+RFyUa2Y3WDaXd+P0GJgDjxRSegPNnoqABL2QfQut7t9CymrNwn
E+wWWaaZF0LmjSfSgRSwS4L6ssXQuyBZYiijlrVh9nbBbUbS/brGDz3RyXeaWDP2
BnYyrVFfKV9u+BKLrebFCDmzQOOpW5Edl+PPUd9lWY6NgKtTAgMBAAGjgecwgeQw
DAYDVR0TAQH/BAIwADA2BgNVHR8ELzAtMCugKaAnhiVodHRwOi8vY3JsLnRoYXd0
ZS5jb20vVGhhd3RlU0dDQ0EuY3JsMCgGA1UdJQQhMB8GCCsGAQUFBwMBBggrBgEF
BQcDAgYJYIZIAYb4QgQBMHIGCCsGAQUFBwEBBGYwZDAiBggrBgEFBQcwAYYWaHR0
cDovL29jc3AudGhhd3RlLmNvbTA+BggrBgEFBQcwAoYyaHR0cDovL3d3dy50aGF3
dGUuY29tL3J1cG9zaXRvcnkvVGhhd3RlX1NHQ19DQS5jcnQwDQYJKoZIhvcNAQEF
BQADgYEAicju7fexy+yRP2drx57Tcqo+BElRlCiHNZlnhPmS9QSZaudDA8jy25IP
VWvjEgaq13Hro0Hg32ZNVK53qcXwjWtnCAReojvNwj6/x1Ciq5B6D7E6eiYDSfXJ
8/a2vR5IbgY89nq+wuHaA6vspH6vNR848xO3z1PQ7BrIjnYQ1A0=
-----END CERTIFICATE-----
subject=/C=US/ST=California/L=Mountain View/O=Google Inc/CN=mail.google.com
issuer=/C=ZA/O=Thawte Consulting (Pty) Ltd./CN=Thawte SGC CA
---
No client certificate CA names sent
---
SSL handshake has read 1773 bytes and written 316 bytes
---
New, TLSv1/SSLv3, Cipher is RC4-SHA
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1
    Cipher    : RC4-SHA
    Session-ID: 66EE866203EE2F764591D20357C4C1556DA9C5CB8476A04A5E4BFC84C3FC10F1
    Session-ID-ctx:
    Master-Key: E08CE3405A61A2F0F53EDCCCC0673F01FAE0E9DB7FE1D5F56156A88C8A35537BA8E8B5C163EB94506D1F1A374E1E2B96
    Key-Arg   : None
    Start Time: 1319528177
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
```

تصویر ۵-۷: فرایند دستکانی SSL که در زمان استفاده از گزینهٔ s_client در فرمان openssl مشاهده می‌شود

SSL مرتبط با امنیت است و X.509 مرتبط با اعتماد است. یک گواهی SSL بر اساس معیارهای خاص برای یک فرد یا شرکت صادر می‌شود. CA به‌عنوان مرکز اعتبارسنجی مسئول صدور گواهی و احراز هویت مشتری است. برای مثال، هر کسی نمی‌تواند یک گواهی برای www.google.com را درخواست دهد بدون اینکه ثابت کند به نحوی به آن وابسته یا به‌عنوان نماینده شرکت گوگل ا ست. اگر CA این مجوزها را بررسی نکند، بنابراین هرکسی می‌تواند از مجوز SSL دیگران استفاده نماید و بر روی وب سرور خود نصب کند.

با فریب دادن کاربر نهایی ، باور به اینکه سرور شما یک سرورِ google.com است، می‌توانید یک حمله مرد میانی (MitM) را انجام دهید و همه داده‌های کاربر را ره‌گیری کنید. در آینده به‌صورت خلاصه به حملهٔ MitM نگاه خواهیم کرد، اما ابتدا، به موضوع جالب و مهم دیگری تحت عنوان گواهی خود امضاشده می‌پردازیم.

نکته: یک CA برای مشتریان گواهی SSL صادر کرده است. تا زمانی که گواهی صادر می شود، CA نیز همچنین گواهی SSL را با گواهی اصلی خودش امضا می‌کند. این امضا نشان‌دهنده‌ی این است که CA گواهی SSL صادر شده را تائید می‌کند. یک مرورگر می‌تواند گواهی SSL را در نگاه اول با امضای CA بررسی کند که آیا امضا با گواهی اصلی مطابقت دارد یا نه.

تعداد زیادی ازCA های اصلی شناخته‌شده در سراسر جهان وجود دارد. به‌طورکلی، گواهی‌های اصلی CA بسته‌بندی‌شده و داخل مرورگر وب شما آمده است. این به‌مرورگر شما اجازه می‌دهد تا گواهی‌های SSL صادرشده را به‌وسیلهٔCA های مختلف بررسی نماید.

برای مثال، فرض کنید از VeriSign یک گواهی برای دامنه example.com درخواست کرده‌اید. VeriSign اول بررسی می‌کند که شما صاحب واقعی این دامنه هستید یا نه و سپس یک گواهی برای وب سرور شما صادر می‌کند. او این گواهی‌ها را با گواهی اصلی خودش امضا[1] می‌نماید. پس‌ازاینکه شما گواهیSSL تان را دریافت کردید، آن را روی وب سرورتان نصب و وب‌سایت خود را راه‌اندازی می‌کنید. حالا، زمانی که وب سایت شما بازدید می‌شود، مرورگر کلاینت (بازدید کننده) اول گواهی SSL شمارا بررسی می‌کند، و سپس تلاش می‌کند تا بررسی کند که آیا گواهی شما به‌درستی توسط یک CA معتمد صادرشده[2] یا نه. برای انجام این کار، مرورگر به مخزن گواهی‌های اصلیِ مورد اعتماد داخلی‌اش نگاه می‌اندازد تا تعیین کند که آیا امضای گواهی اصلی VeriSign با امضای روی گواهی مطابقت دارد یا نه. اگر مطابق باشد، بنابراین می‌تواند با خیالی آسوده به دیدن سایت شما ادامه دهد. بااین‌حال، اگر مشکلی در بررسی گواهی شما وجود داشته باشد، مرورگر هشدار می‌دهد که قادر به تائید گواهی نیست. توجه داشته باشید که مرورگر قبل از چراغ سبز دادن، تعدادی از جزئیاتِ دیگرِ گواهی را بررسی خواهد نمود.

۱-٤-۵ گواهی خود امضاشده

در زمان توسعه و آزمایش پروژه‌ها، توسعه‌دهندگان گاهی اوقات از یک گواهی خود امضا شده روی وب سایت‌شان استفاده می‌کنند. این نوع از گواهی‌نامه‌ها، در تمام جهات با گواهینامهٔ SSL صادر شده توسط CAیکسان است. بااین‌حال تفاوت اصلی این است که، امضای روی این گواهی از یک CA معتمد نیست. در عوض، توسعه‌دهنده گواهی را خودش امضا می‌کند. زمانی که یک مرورگر با یک گواهی SSL خود امضاشده

[1] منظور از امضا، Hash کردن پیام رمز شده می‌باشد.

[2] باتوجه به مفاهیمی که آموخته‌ایم CA باید احراز هویت شود تا پیام های مبادله شده شنود نشوند.

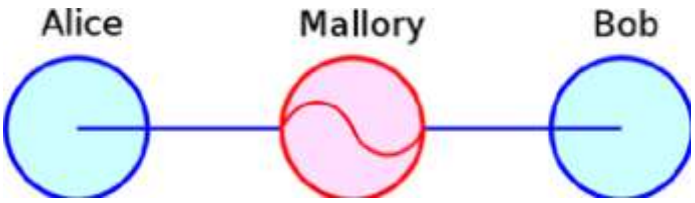
به سایتی متصل می‌شود، هیچ راهی برای تائید گواهی امضاشده وجود ندارد. چراکه گواهی شخصی که آن را امضا کرده در مخزن گواهی‌های معتمد داخلی مرورگر، وجود ندارد. بنابراین مرورگر با یک خطایی شبیه به آنچه در تصویر ۵-۸ نشان داده‌شده است به کاربر هشدار می‌دهد.

تصویر ۵-۸: خطا برای یک گواهی غیرقابل اطمینان یا خود امضاشده

این مرحله‌ی تأیید خطا در مرورگر بسیار مهم است. وجود آن باعث می شود که یک مهاجم به‌راحتی نتواند خودش یک گواهی متعلق به www.google.com را صادر کند و به کاربران حقه بزند. همینطور یک مرورگر اگر نتواند گواهی SSL را تائید کند به کاربر هشدار می‌دهد و به همین دلیل پیام untrusted در مرورگر ظاهر می‌شود.

۵-٤-۲ حمله مرد میانی (MITM)

یک حملهٔ MitM روشی است که یک مهاجم می‌تواند ترافیک شبکه یا داده‌های در حال جریانِ بین دو بخش را استراق سمع کند. مهاجم در موقعیتی قرار می‌گیرد تا قادر به رهگیری ترافیک ارسال‌کننده و گیرنده باشد و به‌طور مؤثر خودش را در بین این دو قرار می‌دهد(تصویر ۵-۹). در این موقعیت، مهاجم قادر به رهگیری و انتقال اطلاعات بین دو طرف است. اگر این کار به‌درستی انجام شود، کاربران در هر دو طرف گفت‌وگو نمی‌توانند بفهمند که یک مهاجم ترافیک آن‌ها را شنود و رهگیری می‌کند.



تصویر ۵-۹: مالوری در بین آلیس و باب(تصویر از ویکی‌پدیا)

آنچه در زیر آمده مثالی از حملهٔ MitM با استفاده از تصویر ۵-۹ به‌عنوان منبع است:

- آلیس به باب جعلی: سلام باب ، آلیس هستم کلیدت را بده
- آلیس جعلی : سلام باب، آلیس هستم کلیدت را بده
- باب به آلیس جعلی: کلید باب
- باب جعلی به آلیس: کلید جعلی باب
- آلیس به باب جعلی : یک پیام رمزی‌شده با کلید جعلی
- آلیس جعلی به باب : یک پیام تغییر داده و رمز شده با کلید باب

اکثر اوقات، حملات بر روی گواهی‌های خود امضا یا فریب مرورگرها با گواهیِ مهاجم به‌ظاهر معتبر متمرکز شده است. تا قبل از حمله DigiNotar ، مهاجمان اطلاعات بسیار کمی در مورد امنیت CA به دست می‌آوردند، و حوادثی به‌مراتب کمتر در سمت CA ها وجود داشت. به‌هرحال، این موضوع تا ژوئن ۲۰۱۱ درست بود.

در تئوری، حمله به یک CA برای به دست آوردن کلیدهای اصلی و گواهی‌های SSL نیز یک گزینه است. مهاجمان خیلی زیادی به این گزینه فکر نمی‌کنند زیرا آن‌ها به‌طور بدیهی انتظار دارند که درجهٔ امنیت برای ورود به CA ها بالا باشد. ولی اشتباه است! در ژوئن ۲۰۱۱، یک CA بنام DigiNotar موردحمله قرار گرفت. مهاجم بیش از ۵۰۰ گواهی SSL جعلی امضاشده توسط DigiNotar برای خودش صادر کرد. به‌عنوان یک CA مطمئن، DigiNotar دارای گواهی اصلی در تمام مرورگرهای مدرن است. این بدین معنی است که مهاجم گواهی‌های SSL قانونی را دارد که او می‌تواند برای انجام حملات MitM از آن استفاده کند. ازآنجاکه مرورگرها در حال حاضر به گواهی اصلی DigiNotar اعتماد کرده‌اند، پس آن‌ها این گواهی SSL جعلی را معتبر می‌دانند، و کاربران نهایی هرگز نمی‌فهمند که مهاجم داده‌هایشان را می‌رباید. چرا این اتفاق افتاد؟ DigiNotar کنترل‌های امنیتی بسیار سهل‌انگارانه‌ای را در زیر ساختارش داشت. بنابراین مهاجم قادر به ارتباط راه دور با سرورهای آن‌ها بود و به بسیاری از سیستم‌هایی که مسئول صدور گواهی قانونی هستند دسترسی پیدا کرد. بعدازاین، برای مهاجم صدور گواهی برای خودش، یک کار نسبتاً ساده‌ای است. برخی از

وب‌سایت‌های مشهور که گواهی جعلی برای آن‌ها ایجاد شد عبارت‌اند از:

*.google.com

(این به معنی این است که شامل تمام زیر دامنه‌های google.com است مثلاً

(mail.google.com, docs.google.com, plus.google.com , ...

*.android.com
*.microsoft.com
*.mozilla.org
*.wordpress.org
www.facebook.com
www.mossad.gov.il
www.sis.gov.uk

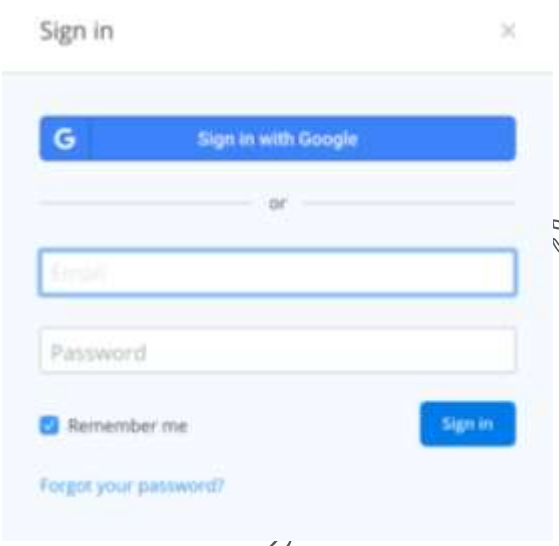
همهٔ مرورگرهای وب، گواهی اصلی DigiNotar را در فهرستشان ثبت کرده‌اند و DigiNotar به شیوه‌های قاعده‌مند شروع به لغو تمام گواهی‌های جعلی کرده است. بااین‌حال DigiNotar اعتماد هزاران کاربر را در سراسر جهان ازدست‌داد. اگر چنین CA ی بزرگی، می‌تواند از چنین نقض امنیتی بزرگ رنج ببرد، که صدها عدد از گواهی‌های SSL به خطر بیافتد، پس آیا ما واقعاً می‌توانیم در تمام مدت فقط وابسته به SSL باشیم؟ درواقع، بله ما می‌توانیم. رویدادهایی مانند DigiNotar اغلب به‌ندرت اتفاق میافتد، بنابراین تمایل داریم اعتماد SSL را انتخاب کنیم. بااین‌حال بهتر است تا یک‌لایهٔ رمزنگاری داده بین برنامه‌ی موبایل و سرور اضافه نماییم. پس، اگر لایهٔ SSL به هر روشی نقض شود، مهاجم هنوز با دیگر لایه‌های رمزنگاری مواجه هست. در اغلب موارد، این لایهٔ اضافی به‌عنوان یک عامل بازدارنده عمل خواهد کرد، و ممکن است مهاجم از ادامه حمله به برنامهٔ شما منصرف شود.

آیا راهی وجود دارد که از جا سو سی مهاجمان روی گواهی‌های SSL بتوان جلوگیری کرد؟ در واقع بله! بیایید به دو راه نگاه کنیم که حتی زمانی که کانال‌های انتقال امن شکست‌خورده باشند از به خطر افتادن گواهی‌هایمان جلوگیری کنیم. اولین آن OAuth و Challenge/Response دومین آن است.

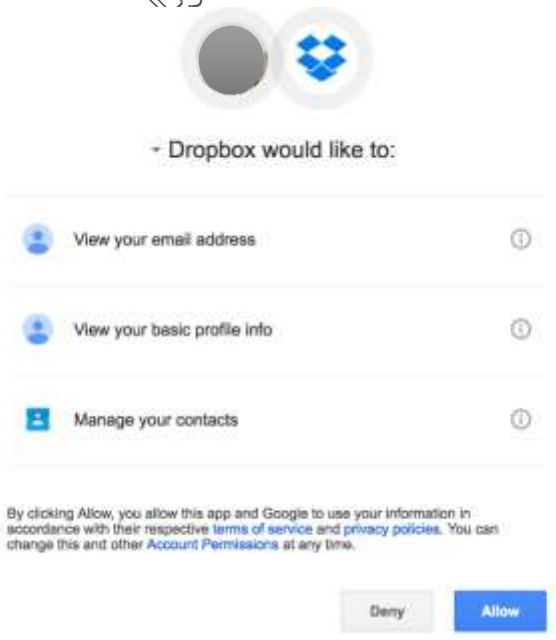
۳-٤-۵ OAuth

پروتکل OAuth به سایر سایت‌ها یا برنامه‌های کاربردی اجازه می‌دهد (به‌عنوان مصرف‌کننده) از اطلاعات کاربر که بر روی یک برنامه‌ی وب دیگر وجود دارد، (که یک ارائه‌دهنده‌ی سرویس نامیده می‌شود) استفاده کند. به‌عنوان‌مثال dropbox یک سرویس‌دهنده فضای ابری است که کاربران می‌توانند فایل‌های خود را در این فضای ابری ذخیره کنند. قبل ازOAuth ، یک کاربر برای ورود به dropbox.com باید نام کاربری و رمز عبور را وارد می‌کرد تا بتواند به فایل‌های خود در این سایت دسترسی داشته باشد. مشکل این روش آن است که کاربر اطلاعات هویتی خود را در dropbox ذخیره نموده و سطح افشای اطلاعات هویتی‌اش افزایش

پیدا کرده است. اگر چنین سناریویی با OAuth پیاده‌سازی شود، دیگر کاربر نباید برای ثبت‌نام یا ورود مجدد به سایت اطلاعات هویتی خود را وارد کند. در عوض، dropbox (مصرف‌کننده) کاربر را به سایت گوگل (ارائه‌دهنده‌ی سرویس) هدایت می‌کند(تصویر ۵-۱۰ را ببینید) و از او درخواست می‌کند که دسترسی به اطلاعات کاربری خود در گوگل را تائید یا رد کند(تصویر ۵-۱۱ را ببینید). بدین ترتیب dropbox هویت کاربر را از طریق حساب کاربری گوگل احراز می‌کند.



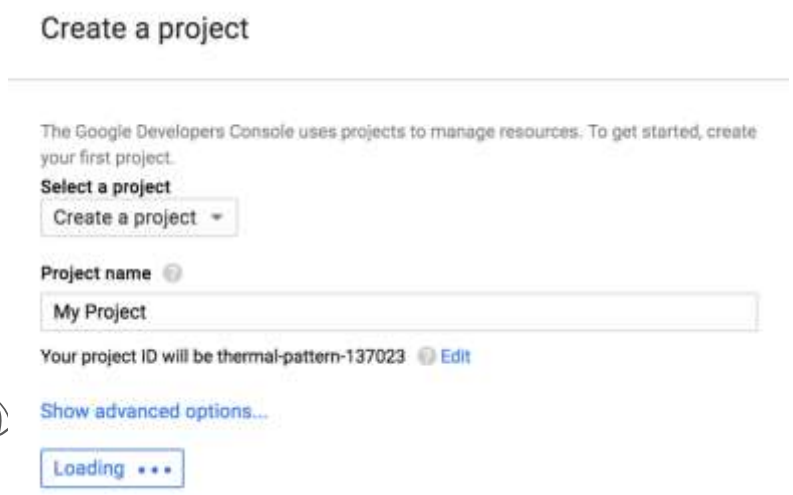
تصویر ۵-۱۰: ورود به dropbox با استفاده از حساب کاربری google



تصویر ۵-۱۱ dropbox برای استفاده از اطلاعات حساب کاربری گوگل درخواست مجوز می‌کند

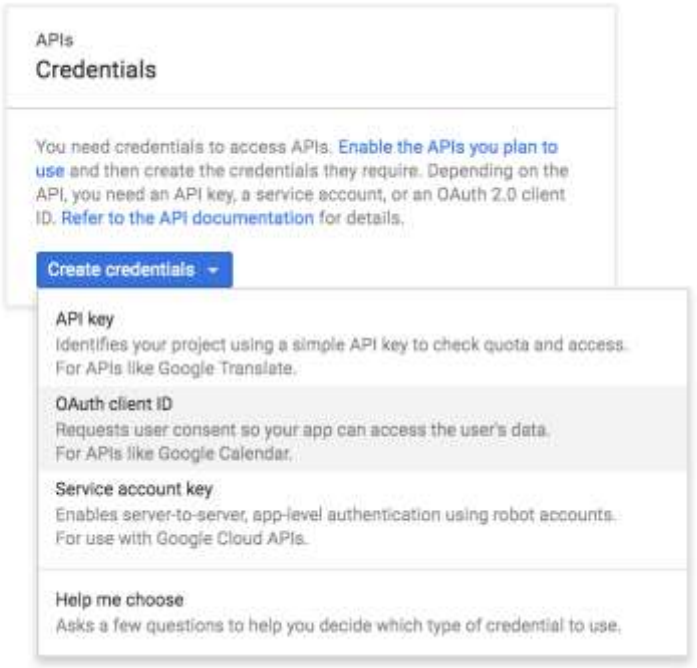
OAuth با استفاده از نشانه (توکن) درخواست[1] کار می‌کند. سایت‌هایی که می‌خواهند به داده‌ها در یک برنامهٔ وب دیگر دسترسی داشته باشند، قبل از اینکه بتوانند به داده‌ها را دسترسی داشته باشند باید نخست، از چنین برنامه‌هایی یک توکن در اختیار داشته باشند.

اجازه دهید به چگونگی کار کردن OAuth برای dropbox نگاه کنیم. فرض کنید که یک برنامهٔ اندروید نوشته‌اید که برای احراز هویت از اطلاعات حساب کاربری گوگل استفاده می‌کند. نرم‌افزار اندروید برای انجام این کار نیاز به دسترسی به اطلاعات حساب کاربری گوگل را دارد. برنامهٔ اندروید(مصرف‌کننده)، گوگل (ارائه‌دهندهٔ سرویس) و کاربر نهایی سه نقش اصلی در این فرآیند بشمار می‌روند. در OAuth نیاز است که اول برنامهٔ مصرف‌کننده در سایت ارائه دهنده سرویس که قصد احراز هویت بوسیله آن را دارد، ثبت نام نماید. این کار لازم است، چون با این عمل برنامه مصرف کننده یک شناسهٔ برنامه دریافت خواهید کرد که برای استفاده در کد مورد نیاز خواهد بود. برای ثبت برنامه مصرف کننده مورد مثال، باید سایت https://console.developers.google.com/apis/library رویت شود(تصویر ۵-۱۲). کتابخانه OAuth 2.0 credentials جستجو شود. در یک پروژه جدید، یک ID مشتری OAuth ایجاد گردد(تصویر ۵-۱۳، تصویر ۵-۱۴، تصویر ۵-۱۵ و تصویر ۵-۱۶).

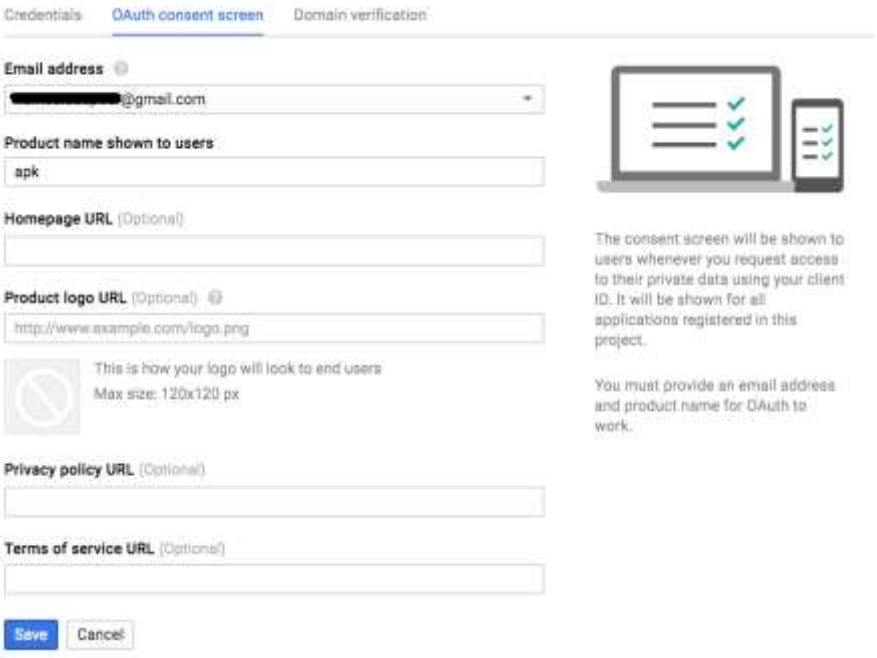


تصویر ۵-۱۲: ایجاد یک پروژه جدید روی گوگل

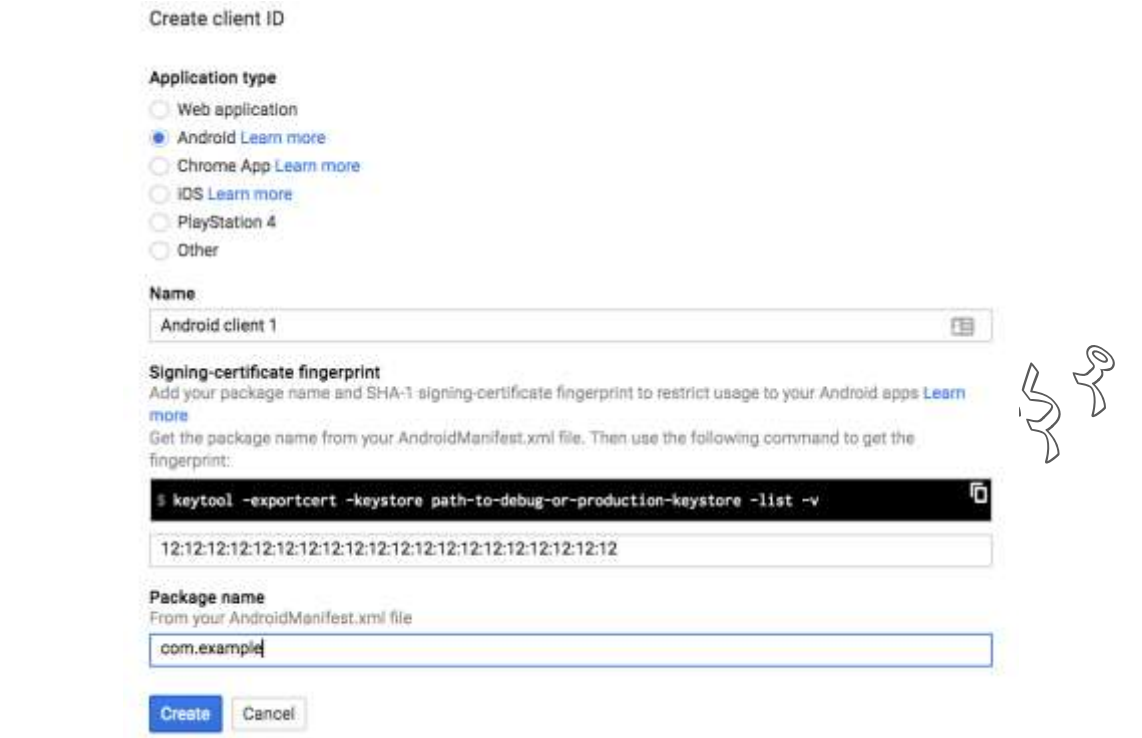
[1] Request token



تصویر ۵-۱۳: ایجاد یک ID مشتری جدید



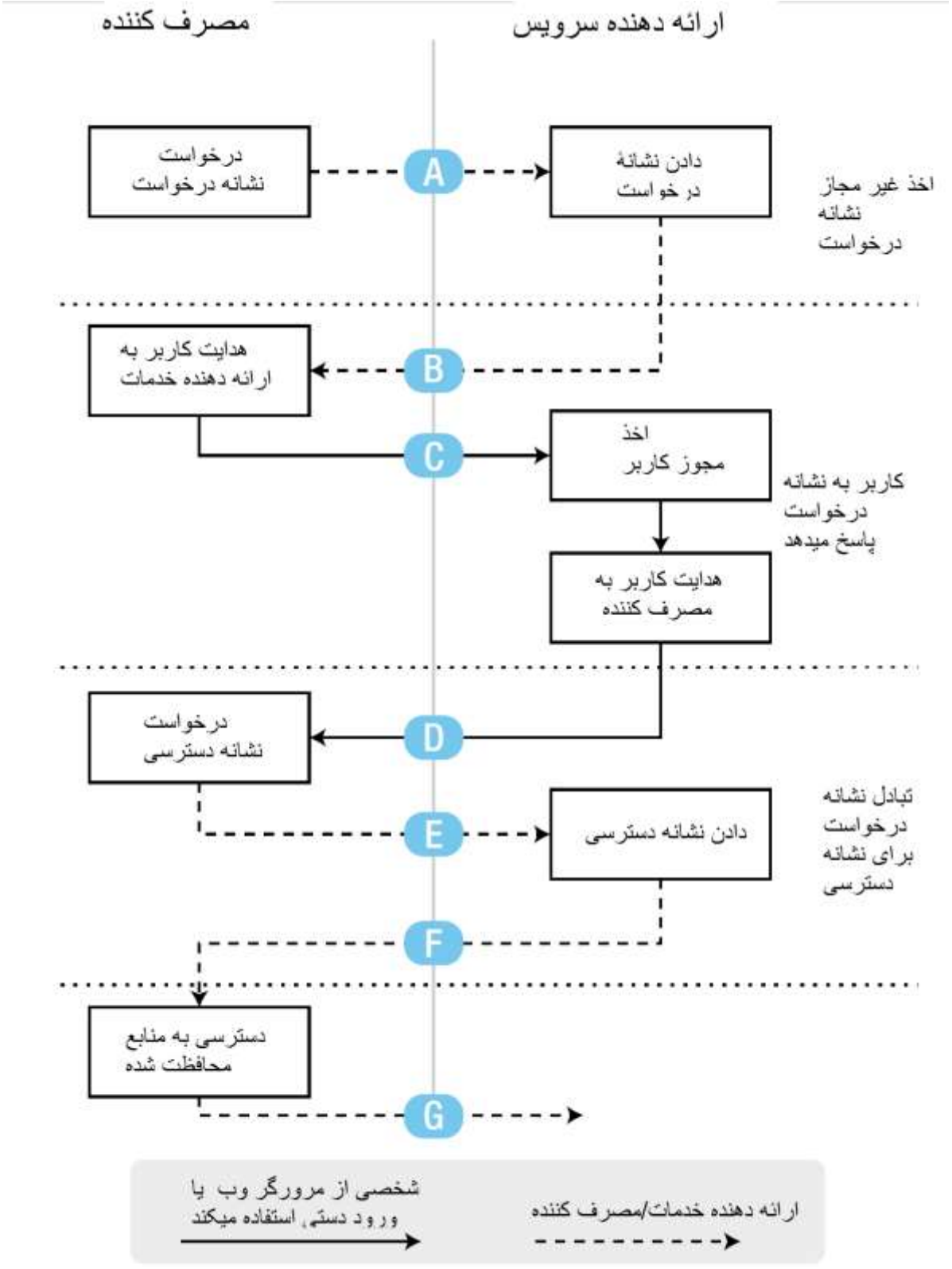
تصویر ۵-۱٤: تکمیل جزئیات صفحه درخواست OAuth



تصویر ۵-۱۵: انتخاب نوع برنامهٔ کاربردی و تکمیل اطلاعات آن

تصویر ۵-۱۶: ID مشتری و رمز مشتری اکنون ایجادشده

حالا که ID مشتری OAuth بدست آمد، به جریان احراز هویت یک برنامهٔ کاربردی با OAuth نگاه خواهیم کرد(تصویر ۵-۱۷).



تصویر ۵-۱۷: جریان احراز هویت OAuth

OAuth یک فرایند چندمرحله‌ای ا ست که سه بخش تعاملی ا صلی دارد. م صرف‌کننده یک نرم‌افزاری

است که خواستار دسترسی به اطلاعات یک ارائه‌دهنده‌ی سرویس است و این فقط زمانی اتفاق می‌افتد که کاربر به‌صراحت به مصرف‌کننده اجازه دهد. به جزئیات بیشتر این مراحل نگاهی می‌اندازیم:

۱. جریان A: نرم‌افزار مصرف‌کننده(برنامه‌ی اندروید) یک نشانهٔ درخواست از ارائه‌دهنده‌ی سرویس(google) درخواست می‌کند.
۲. جریان B: گوگل به برنامهٔ شما می‌گوید که کاربر را به صفحهٔ وب برای درخواست مجوز هدایت نماید. پس برنامهٔ شما یک صفحهٔ وب‌باز می‌کند که کاربر نهایی را به URL خاص هدایت خواهد نمود.
۳. جریان C: کاربر نهایی گواهی خودش را در این صفحه وارد می‌کند. لازم به ذکر است که او باید در وب سایت ارائه‌دهندهٔ سرویس (google) وارد شود و اجازهٔ دسترسی به برنامهٔ شمارا بدهد. کاربر گواهی‌اش را به وب‌سایت ارسال می‌کند و آن را در هیچ جای دستگاه ذخیره نمی‌نماید.
۴. جریان D: گوگل اجازهٔ دسترسی به برنامهٔ را می‌دهد. در این مرحله، نرم‌افزار شما باید این جواب را تشخیص دهد و بر این اساس عمل کند. فرض کنیم مجوز داده‌شده است، بنابراین نرم‌افزار شما در حال حاضر یک نشانهٔ درخواست مجاز دارد.
۵. جریان E: با استفاده از این نشانهٔ درخواستِ مجاز، برنامهٔ شما درخواست دیگری را برای ارائه‌دهندهٔ سرویس ایجاد می‌کند.
۶. جریان F: سپس ارائه‌دهنده‌ی سرویس نشانهٔ درخواست را برای دسترسی به نشانه و ارسال‌هایی که در پاسخ برگشت شده‌اند مبادله می‌کند.
۷. جریان G: در حال حاضر برنامهٔ شما از این دسترسیِ به نشانه برای دسترسی به هر منبع محافظت شده(در این مورد، اطلاعات حساب کاربری گوگل) استفاده می‌کند تا زمانی که این نشانه منقضی شود.

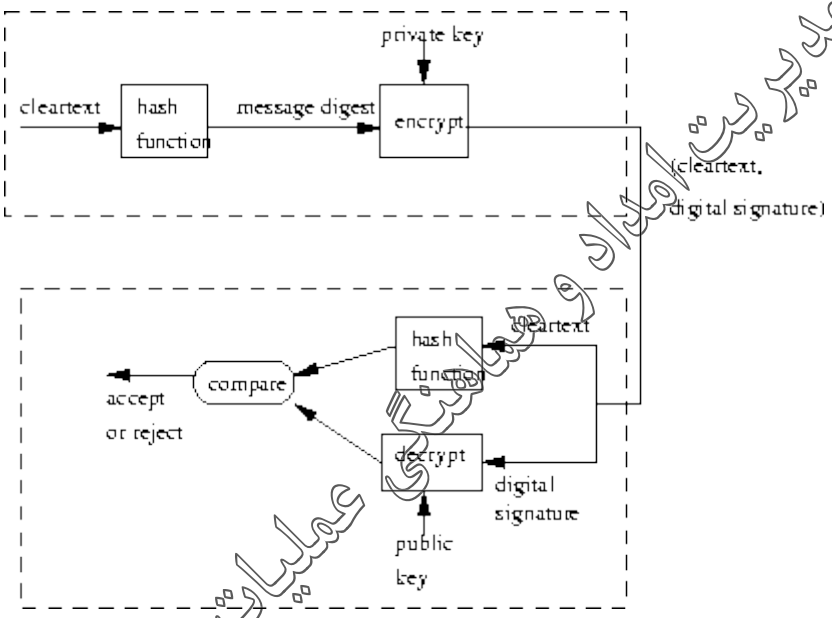
برنامه، دسترسی به‌حساب کاربری گوگل را بدون نیاز به ذخیرهٔ گواهی کاربر نهایی با موفقیت به دست آورد. حتی اگر تلفن همراه کاربر به خطر بیافتد و یک مهاجم تمام داده‌های برنامه را کپی کند، او نام کاربری و رمز عبور google را در داده‌های برنامهٔ شما پیدا نخواهد کرد. در این روش، شما مطمئن هستید که ضرورتاً برنامهٔ شما داده‌های حساس را درز نمی‌دهد.

در اینجا از google صرفاً به‌عنوان یک چارچوب مرجع استفاده کرده‌ایم. هدف نهایی ما ایجاد یک سیستم احراز هویت OAuth برای برنامه‌های کاربردی back-end است. بنابراین، به‌جای google ، برنامهٔ

وبِ back-end شما، ارائه‌دهندهٔ سرویس OAuth خواهد بود.

کاربر نهایی باید با یک مرورگر وب وارد برنامه‌ی کاربردی وب شود و صراحتاً دسترسی به منابع را اجازه دهد. سپس برنامهٔ موبایل شما و برنامهٔ وب back-end با استفاده از درخواست و نشانه‌های دسترسی ارتباط برقرار خواهند کرد. مهم‌تر از همه، برنامهٔ اندروید نام کاربری و رمز عبور برنامهٔ وب را ذخیره نمی‌کند.

٤-٤-٥ چالش/پاسخ با رمزنگاری



مکانیسم دوم برای حفاظت از گواهی کاربر نهایی در هنگام انتقال در اینترنت، استفاده از تکنیک چالش/پاسخ است. این تکنیک در بسیاری از جهات شبیه به OAuth است که هیچ گواهی دراین ارتباط ارسال نمی شود. در عوض، یک‌طرف، طرفِ دیگر را برای چالش می‌طلبد. سپس طرف دیگر با توجه به یک الگوریتم خاص انتخاب‌شده و تابع رمزنگاری، یک‌رشته تصادفی از اطلاعات را رمز می‌کند. کلید استفاده‌شده برای رمزنگاری این اطلاعات، رمز عبور کاربر ا ست. این داده‌های رمز شده به‌طرف به چالش کشیده شده ارسال می‌شود، که پس‌ازآن همان رشته از اطلاعات را با استفاده از رمز عبور ذخیره‌شده در پایگاه آن رمزنگاری می‌کند. سپس متن رمزی شده سنجیده می شود; اگر منطبق با شد، به کاربر اجازه‌ی د سترسی داده می‌شود.

خلاصه

این فصل، بیشتر روی چگونگی انتقال داده به‌صورت امن از برنامه‌ی موبایل به برنامهٔ وب متمرکز شده بود. پروتکل‌ها و روش‌هایی برای امن کردن داده در زمان انتقال پو شش داده شد. آموختیم ، در برخی

موارد نمی‌توان به برخی از پروتکل‌ها اعتماد کرد. در چنین مواردی، روش‌هایی برر سی شد که می‌توانست در محافظت از گواهی کاربر نهایی در برابر دزدیده شدن یا استراق سمع کمک کند.

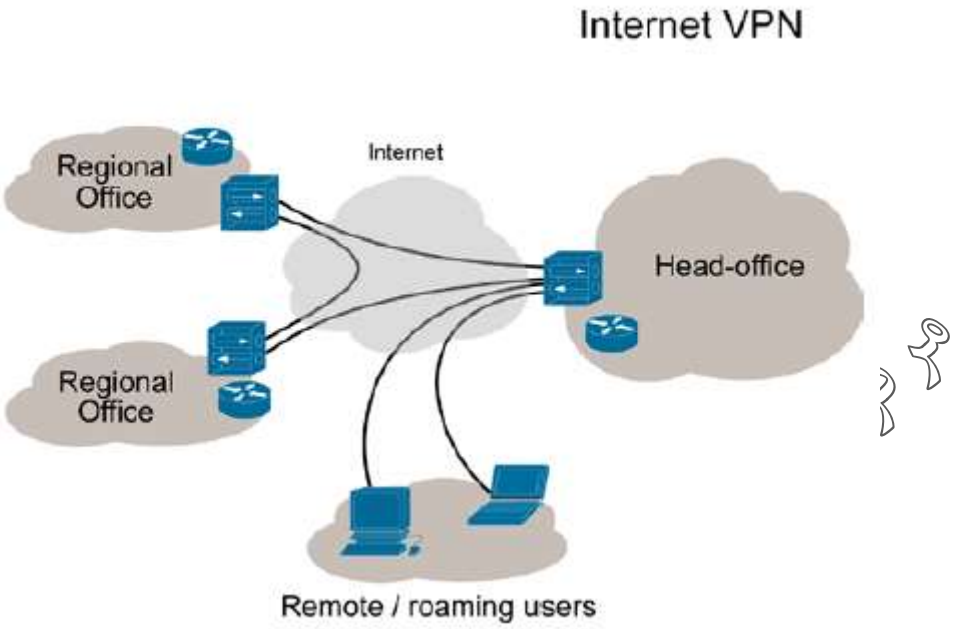
همچنین امنیت برنامه‌های تحت وب بررسی گردید. با توجه به اینکه بیشتر برنامه‌های موبایل در بعضی از شکل‌ها با یک برنامه‌ی وب ارتباط برقرار می‌کند، بهتر است که بدانیم چگونه این فناوری کار می‌کند. درنهایت، به برخی از نمونه‌های واقعی برای محافظت از گواهی‌های کاربر در زمان انتقال اشاره شد.

٦ امنیت در برنامه‌های تجاری

تا به اینجا امنیت در اندروید را از دید یک برنامه نویس بررسی کرده ایم. اما بهتر است کمی بر روی توسعه‌دهندگان بزرگ و تجاری تمرکز نماییم. ممکن است شما خود را هم سطح چنین توسعه دهندگانی نبینید. مسلما ابعاد کار آنها از توان یک برنامه نویس خارج است، اما باید به این نکته توجه کنید که این توسعه دهندگان به نگرش جدیدی روی آورده اند که کارهای خود را برون سپاری نمایند. بسیاری از شرکت‌های توسعه دهنده بزرگ تجاری، کارهای موبایلی خود را به شرکت‌های کوچک واگذار می‌کنند و به این ترتیب دغدغه‌ای برای مدیریت تیم توسعه دهنده ندارند. به همین دلیل ممکن است یک شرکت بزرگ تجاری، برنامه موبایلی را به شما سفارش دهد. در چنین موقعیتی شما با چالشی فراتر از از دست رفتن اطلاعات شخصی کاربر روبرو هستید. در یک محیط بزرگ تجاری با اطلاعات حساسی مانند اطلاعات مالی و سرور ها سر و کار دارید.

٦-١ ارتباط

امروزه، ارتباط با محل شرکت از راه دور به یک امر متداول تبدیل‌شده است. ارتباط از راه دور، پشتیبانی از راه دور و برون سپاری، نیاز مهمی را بوجود آورده است و آن، اجازه ورود تنها به کاربران مجاز می‌باشد. در اینجا اتصالات به درون شبکه موضوع اصلی کنترل امنیت است. برای ایجاد اطمینان، سازمان ها معمولا از VPN استفاده می کنند. به این ترتیب به کاربران اجازه اتصال به شبکه را می دهند.



تصویر ٦-١ شبکه خصوصی مجازی (vpn)

VPN، مانند پلی است که بین یک شبکه عمومی مانند اینترنت و یک شبکه خصوصی مانند شبکه داخلی یک سازمان، عمل می‌کند(تصویر ٦-١). کاربران راه دور قادرند از طریق VPN، مانند یک کاربر محلی به منابع شبکه خصوصی موردنظر خود، دسترسی داشته باشند. VPNها به تدریج در فضای موبایلی در حال رشد هستند و این قابلیت را دارند تا به شبکه شرکت ها متصل شوند و به صورت امن تبادل اطلاعات کنند. وقتی که یک برنامه تجاری طراحی می کنید باید به نیاز احتمالی مدیر شبکه برای اتصال به شبکه سازمان، از طریق VPN توجه داشته باشید.

اگر سازمانی، از VPN استفاده نمی‌کند، لازم است داده بین کاربر و سرور به صورت رمزنگاری مبادله شود. اگر اطلاعاتی که تبادل می‌شوند دارای اهمیت خاصی هستند، این کار پرهزینه ای است. در واقع، نیاز به فشرده سازی داده‌ها مطرح می‌گردد.

اگر تجربه کار با شرکت های تجاری را نداشته‌اید، مفهوم سیستم‌های تجاری ممکن است برای شما گنگ باشد. این سیستم ها انواع گوناگونی دارند. اما در اینجا پیرامون برنامه ریزی منابع (ERP) [1] نرم افزارها صحبت خواهیم کرد زیرا طیف گسترده ای را در برنامه های تجاری شامل می شود. برنامه های ERP معمولا یکی از این موارد را پوشش می دهند:

- مدیریت زنجیره تأمین

[1] Enterprise Resource Planning

- مدیریت ارتباط با مشتری
- تولید انبوه منابع انسانی
- حسابداری و مالی
- مدیریت پروژه

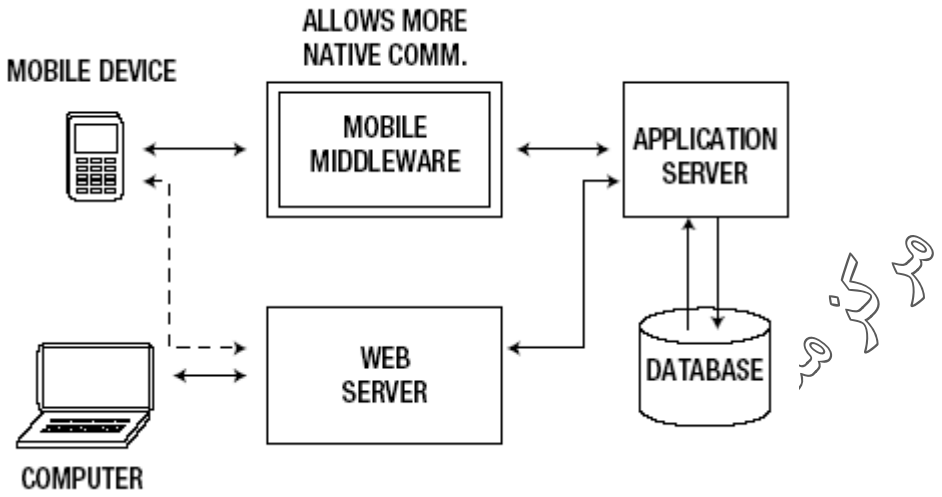
۲-۶ میان‌افزار موبایل

برنامه‌های ERP که با آنها کار می‌کنید به خوبی تعریف شده و رشد کرده‌اند. همچنین به عنوان یک برنامه نویس، باید به گونه ای برنامه‌های خود را ایجاد کنید تا با سیستم های موجود نیز تعامل داشته باشد. این مسئله ممکن است گاهی آزار دهنده باشد. زیرا نیاز است بعضی از عملکردها را بر روی موبایل پیاده سازی نمایید. یکی از بهترین روش هایی که می توان برای رفع این مشکل استفاده کرد، میان افزار[1]ها می باشند. بهتر است زمانی را صرف ایجاد میان افزار مختص به برنامه موبایل خود نمایید. به این ترتیب بجای ارتباط با برنامه های قدیمی تجاری، با میان افزار خودکار می کنید. در حقیقت میان‌افزار موبایل ارتباط بین برنامه‌های موبایل و سیستم تجاری را فراهم می کند. هدف این است که برنامه موبایل بدون اینکه نیاز باشد تا با سیستم های پردازشی تعامل کند و از محدودیت منابع موبایل متأثرشود، به اطلاعات دسترسی پیدا نماید. برای مثال در یک برنامه موبایل بانک، یک میان افزار به عنوان تبدیل کننده درخواست های موبایل ایجاد گردیده است که در سمت سرور قرار دارد. این میان افزار اطلاعات را از نرم‌افزار بانک دریافت می‌کند و در قالب فرمت موبایلی بر می‌گرداند.

همانطور که در تصویر ۶-۲ مشخص شده است، موبایل از طریق میان افزار به اطلاعات دسترسی پیدا می‌کند. موبایل می‌تواند از طریق مرورگر به برنامه سمت سرور، دسترسی پیدا نماید، اما به احتمال زیاد تجربه کاربری مطلوبی نخواهد داشت. با استفاده از میان افزار موبایل، ارتباط بین برنامه ها استاندارد سازی می‌شود.

[1] Mobile Middleware

به این ترتیب عمده ارتباطات با برنامه سیستم بر روی یک سخت افزار قوی اجرا می گردد.



تصویر ٦-٢ مثال میان‌افزار موبایل

لذا برای ایجاد برنامه های بزرگ تجاری موبایلی نیاز است تا به چند مبحث کلیدی توجه کنیم. در این فصل به دو مبحث دسترسی به پایگاه داده [١] و نمایش اطلاعات [٢] می پردازیم.

٦-٢-١ دسترسی به پایگاه داده

وقتی صحبت از دسترسی به پایگاه داده به میان می آید اولین روشی که به ذهن می رسد این است که اطلاعات مورد نیاز برای دسترسی به پایگاه داده درون برنامه ذخیره شود و در موقع نیاز اتصال ایجاد شود و تراکنش مورد نظر انجام شود. اما این روش کاملا ناامن است. زیرا نیاز است تا این اطلاعات در زمان کدنویسی درون برنامه تعبیه شوند. وقتی سورس برنامه بوسیله شخص نفوذگر خوانده شود این اطلاعات بسیار حساس در اختیار وی قرار می گیرد و تمام سیستم شما در معرض خطر قرار می‌گیرد. برای مثال در کد ٦-١ برنامه نویسان از یک کلاس برای نگهداری اطلاعات پایگاه داده استفاده کرده‌اند.

کد ٦-١ یک کلاس نمونه برای ذخیره اطلاعات پایگاه داده

```
public class JDBCHandler {
    private String username;
    private String password;
    private String url;
    private String port;
    private String db;
}
```

[١] Database Access

[٢] Data Presentation

همانطور که مشخص است این روش در همان مرحله اول نا امن است و به ادامه این روش نمی پردازیم.
برای رفع این مشکل بهتر است از REST Api [۱] استفاده شود. به این ترتیب با استفاده از پروتکل http درخواست به میان افزار ارسال و نتیجه را دریافت می کند. تمام اتصالات و تراکنش ها با پایگاه داده توسط میان افزار انجام می شود و تنها اطلاعاتی که برای تراکنش نیاز است توسط موبایل ارسال می‌گردد. از آنجایی که اطلاعات ارسال شده از موبایل به صورت پویا است و به استفاده کاربر بستگی دارد، امنیت برنامه در سطح بالاتری نسبت به روش قبل تامین می گردد. برای افزایش امنیت ارتباط، باید ابزاری که برای ارتباط با میان افزار انتخاب می‌شود نسبت به باقی ابزارها امن تر باشد.
در این کتاب از کتابخانه Retrofit 2.0 [۲] استفاده می شود. در روش های قدیمی از HttpClient و کلاس‌های موجود در کتابخانه Apache برای اتصال به سرور ها استفاده می شد اما به علت کند بودن نسبت به کتابخانه‌های جدید و داشتن امکانات کمتر کنار گذاشته شد. از مزیت‌های استفاده از Retrofit 2.0 می توان به موارد زیر اشاره کرد:

- نسبت به سایر کتابخانه‌ها(Async Tast ، Volley و غیره) سریع‌تر است و در bench mark test رتبه بهتری دارد.
- از java.io [۳] و java.nio [۴] پشتیبانی می کند که باعث آسان تر شدن دسترسی، ذخیره و پردازش داده ها می شود.
- با استفاده بهتر از بافر و ByteString، میزان مصرف Cpu و حافظه را بهینه می‌کند.

برای استفاده از Retrofit 2.0 نیاز است تکه کد ذیل، در Maven (کد ۶-۲) و در gradle (کد ۶-۳) به برنامه اضافه نماییم.

کد ٦-٢ برای Maven

```
<dependency>
  <groupId>com.squareup.retrofit2</groupId>
  <artifactId>retrofit</artifactId>
  <version>2.1.0</version>
</dependency>
```

کد ٦-٣: فایل build.gradle

```
compile 'com.squareup.retrofit2:retrofit:2.1.0'
```

یادآوری می شود، همان گونه که در فصل های قبل اشاره شد تمرکز این کتاب بر روی آموزش برنامه نویسی اندروید نیست و هدف آن آموزش روش های برنامه نویسی امن در اندروید است. برای اینکه درک بهتری

[۱] Representation state transfer
[۲] https://github.com/square/retrofit
[۳] https://docs.oracle.com/javase/8/docs/api/java/io/package-summary.html
[۴] https://docs.oracle.com/javase/8/docs/api/java/nio/package-summary.html

نسبت به میان افزار ایجاد شود، قصد داریم تا به یک میان افزار آماده درخواست دهیم اطلاعاتی در مورد پایگاه داده را به ما ارائه دهد.

میان افزار انتخابی ما https://api.github.com است و قصد داریم در یک برنامه مخازنی که توسط کاربران به صورت عمومی ایجاد شده است را دریافت کنیم. برای مثال وقتی به https://api.github.com/users/Square درخواستی ارسال کنیم مشخصات آن را بر می گرداند (کد ۶-۴).

کد ٦-٤ اطلاعات ارسالی از میان افزار github

```
{
  "login": "square",
  "id": 82592,
  "avatar_url": "https://avatars.githubusercontent.com/u/82592?v=3",
  "gravatar_id": "",
  "url": "https://api.github.com/users/square",
  "html_url": "https://github.com/square",
  "followers_url": "https://api.github.com/users/square/followers",
  "following_url":
"https://api.github.com/users/square/following{/other_user}",
  "gists_url": "https://api.github.com/users/square/gists{/gist_id}",
  "starred_url":
"https://api.github.com/users/square/starred{/owner}{/repo}",
  "subscriptions_url":
"https://api.github.com/users/square/subscriptions",
  "organizations_url": "https://api.github.com/users/square/orgs",
  "repos_url": "https://api.github.com/users/square/repos",
  "events_url": "https://api.github.com/users/square/events{/privacy}",
  "received_events_url":
"https://api.github.com/users/square/received_events",
  "type": "Organization",
  "site admin": false,
  "name": "Square",
  "company": null,
  "blog": "http://square.github.io",
  "location": null,
  "email": null,
  "hireable": null,
  "bio": null,
  "public_repos": 182,
  "public_gists": 5,
  "followers": 0,
  "following": 0,
  "created_at": "2009-05-08T23:28:44Z",
  "updated_at": "2017-02-11T11:07:14Z"
}
```

توجه داشته باشید که فقط اطلاعات مشخص شده در میان افزار برای ما ارسال می گردد و به اطلاعات دیگری دسترسی نداریم. ابتدا مجوز دسترسی به اینترنت را به AndroidManifest.xml اضافه می کنیم. اکنون یک ظاهر برای برنامه تعریف می‌شود تا با وارد کردن نام کاربری بتوانیم اطلاعات github آن شخص را مشاهده کرد. برای این کار در فایل activity_main.xml محیط جستجو و لیست نتیجه طراحی شده است.

کد ٦-٥: محتوای activity_main

```
<?"xml version="1.0" encoding="utf-8?>
android.support.constraint.ConstraintLayout >
"xmlns:android="http://schemas.android.com/apk/res/android
"xmlns:app="http://schemas.android.com/apk/res-auto
"xmlns:tools="http://schemas.android.com/tools
"android:id="@+id/activity_main
"android:layout_width="match_parent
"android:layout_height="match_parent
<"tools:context="android.secure.retclient.MainActivity
EditText>
"android:layout_width="0dp
"android:layout_height="wrap_content
"android:inputType="textPersonName
"android:hint="@string/username
"android:ems="10
"android:id="@+id/searchBar
"android:layout_marginStart="8dp
"app:layout_constraintLeft_toLeftOf="parent
"android:layout_marginLeft="8dp
"android:layout_marginTop="8dp
"app:layout_constraintTop_toTopOf="parent
"app:layout_constraintRight_toLeftOf="@+id/searchButton
"android:layout_marginEnd="8dp
</ "android:layout_marginRight="8dp
ImageView>
"android:layout_width="wrap_content
"android:layout_height="wrap_content
"app:srcCompat="@drawable/ic_search_black_24dp
"android:id="@+id/searchButton
"android:layout_marginEnd="8dp
"app:layout_constraintRight_toRightOf="parent
"android:layout_marginRight="8dp
"app:layout_constraintTop_toTopOf="@+id/searchBar
"app:layout_constraintBottom_toBottomOf="@+id/searchBar
</ "android:contentDescription="@string/search_button
android.support.v7.widget.CardView>
"android:layout_height="wrap_content
"android:layout_marginEnd="16dp
"app:layout_constraintRight_toRightOf="parent
"android:layout_marginRight="16dp
```

```
"android:layout_marginStart="16dp
"app:layout_constraintLeft_toLeftOf="parent
"android:layout_marginLeft="16dp
"android:layout_marginTop="24dp
"app:layout_constraintTop_toBottomOf="@+id/searchBar
"app:cardUseCompatPadding="true
"android:layout_width="0dp
<"app:cardElevation="6dp
android.support.constraint.ConstraintLayout>
"android:layout_height="wrap_content
<"android:layout_width="match_parent
ImageView>
"android:layout_width="100dp
"app:srcCompat="@mipmap/ic_launcher
"android:id="@+id/avatar
"android:layout_height="100dp
"app:layout_constraintTop_toTopOf="parent
"app:layout_constraintBottom_toBottomOf="parent
"app:layout_constraintLeft_toLeftOf="parent
"android:layout_marginStart="8dp
"android:layout_marginLeft="8dp
"android:layout_marginTop="8dp
</ "android:layout_marginBottom="8dp
TextView>
"android:text="name
"android:layout_width="0dp
"android:layout_height="wrap_content
"android:id="@+id/name
"app:layout_constraintTop_toTopOf="parent
"android:layout_marginStart="8dp
"app:layout_constraintLeft_toRightOf="@+id/avatar
"android:layout_marginLeft="8dp
"app:layout_constraintRight_toRightOf="parent
"android:layout_marginTop="8dp
"android:textAppearance="@style/TextAppearance.AppCompat.Large
"android:layout_marginEnd="8dp
</ "android:layout_marginRight="8dp
TextView>
"android:text="type
"android:layout_width="wrap_content
"android:layout_height="wrap_content
```

```
"android:id="@+id/type
"android:layout_marginTop="16dp
"app:layout_constraintTop_toBottomOf="@+id/name
"android:layout_marginStart="8dp
"app:layout_constraintLeft_toRightOf="@+id/avatar
</ "android:layout_marginLeft="8dp
TextView>
"android:text="repository count
"android:layout_width="wrap_content
"android:layout_height="wrap_content
"android:id="@+id/repo
"app:layout_constraintRight_toRightOf="parent
"app:layout_constraintBottom_toBottomOf="parent
"android:layout_marginBottom="8dp
"android:layout_marginEnd="8dp
"android:layout_marginRight="8dp

</ "android:textAppearance="@style/TextAppearance.AppCompat.Medium
<android.support.constraint.ConstraintLayout/>
<android.support.v7.widget.CardView>
<android.support.constraint.ConstraintLayout/>
```

و نتیجه آن به صورت شکل ۶-۱ خواهد بود.

تصویر ٦-٣ محیط طراحی شده

اکنون نیاز است تا ساختار اطلاعاتی را که می خواهیم از میان افزار دریافت کنیم، مشخص شود. کلاس UserModel.java مشخص کننده ساختار اطلاعات است.

کد ٦-٦: محتوای کلاس UserModel.java

```
package android.secure.retclient;
import com.google.gson.annotations.SerializedName;
import java.io.Serializable;
public class UserModel implements Serializable {
    @SerializedName("name")
    private String name;
    @SerializedName("avatar_url")
    private String profilePic;
    @SerializedName("type")
    private String type;
    @SerializedName("public_repos")
    private int repositoriesCount;
    public String getName() {
        return name;
    }
    public String getProfilePic() {
        return profilePic;
    }
    public String getType() {
        return type;
    }
    public int getRepositoriesCount() {
```

```
        return repositoriesCount;
    }
                                                                    }
```

در کلاس MainActivity.java که به activity_main.xml متصل است، متد دسترسی به میان افزار را می‌نویسیم.

کد ٦-٧: محتوای کلاس MainActivity.java

```
package android.secure.retclient;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import java.io.IOException;
import retrofit2.Call;
import retrofit2.Retrofit;
import retrofit2.http.GET;
import retrofit2.http.Path;
public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
    private ImageView searchButton;
    private EditText username;
    private TextView repository;
    private TextView type;
    private TextView name;
    private ImageView avatar;
    Retrofit retrofit;
    private static final String API_URL = "https://api.github.com/";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        /// init UI
        initUI();
        /// Retrofit Config
        retrofitConfig();
    }
    private void retrofitConfig() {
        retrofit = new Retrofit.Builder()
                .baseUrl(API_URL)
                .build();
    }
    private void initUI() {
        searchButton = (ImageView) findViewById(R.id.searchButton);
        username = (EditText) findViewById(R.id.searchBar);
        repository = (TextView) findViewById(R.id.repo);
        type = (TextView) findViewById(R.id.type);
        name = (TextView) findViewById(R.id.name);
        avatar = (ImageView) findViewById(R.id.avatar);
        /// button click listener
        searchButton.setOnClickListener(this);
    }
    /// search click
    @Override
```

```
    public void onClick(View view) {
        try {
            search(username.getText().toString());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
private void search(String username) throws IOException {
        GithubConnection                    connection                    =
retrofit.create(GithubConnection.class);
        Call<UserModel> requestModel = connection.user(username);
        requestModel.enqueue(new Callback<UserModel>()) {
            @Override
            public      void      onResponse(Call<UserModel>      call,
Response<UserModel> response) {

            }
            @Override
            public void onFailure(Call<UserModel> call, Throwable t) {
            }
        });
    }
    private interface GithubConnection{
        @GET("/users/{username}")
        retrofit2.Call<UserModel> user(
                @Path("username") String username
        );
    }
                                                                        }
```

همانطور که در کد مشخص شده است، درخواست به میان افزار ارسال می‌شود. در این ارتباط هیچ یک از اطلاعات مربوط به پایگاه داده و یا اطلاعات حساس سرور قرار ندارند. به صورت کاملا پویا به میان افزار مورد نظر متصل می شود و اطلاعات را دریافت می کند و حتی با خواندن کد، شیوه پردازش سرور مشخص نمی شود. اکنون یکی از مباحث مهم که ارتباط امن میان سرور و موبایل بود را بررسی کردیم. اما باید روشی نیز برای نمایش اطلاعات دریافتی بکار رود.

۶-۲-۲ نمایش داده

وقتی درخواستی به سرور ارسال می گردد، نیاز است تا با یک روش استاندارد پاسخ ارسال شود. پاسخ ها باید در قالب مشخصی باشند تا از طریق تمام زبان ها و تمام platform ها قابل دسترسی باشند. پروژه های تجاری بزرگ همیشه در حال رشد می باشند و بخش های جدیدی به آنها اضافه می شود. به این ترتیب اگر برای هر زبان و platform از یک ساختار جداگانه استفاده شود، سرور باید پردازش زیادی را صرفا جهت ارسال یک داده مشخص انجام دهد. برای مثال اگر هم نرم افزار دسکتاپی و هم نرم افزار موبایلی برای یک پروژه تجاری تعریف شده باشد، و درخواست دریافت مشخصات کاربر به سرور ارسال شود، علاوه بر اینکه با پایگاه داده ارتباط برقرار می‌کند، باید نوع درخواست دهنده نیز مشخص شود و متناسب با آن پاسخ ارسال گردد. این کار باعث یک حجم زیادی از پردازش ها می شود. برای جلوگیری از این مشکل از یک روش استاندارد

به نام JSON [1] استفاده می شود. تمام زبان های برنامه‌نویسی می توانند با این روش ارتباط برقرار کنند. در حقیقت JSON یک رشته با ساختار استاندارد است. کد ۴-۶ یک نمونه از پاسخ سرور در قالب JSON است. برای خواندن JSON در اندروید بهتر است از کتابخانه GSON[2] استفاده شود. مزیت این کتابخانه این است که JSON را با مدل کلاس ساختار منطبق می سازد و دیگر نیاز نیست تا با استفاده از دستور های شرطی و تکرار مقادیر را از رشته JSON دریافت کنیم.

Retrofit 2.0 در مبحث نمایش داده نیز مزیتی نسبت به سایر کتابخانه ها دارد. می توان کتابخانه ای که نیاز است تا با آن JSON پردازش شود را در Retrofit 2.0 مشخص کرد و نتیجه را با استفاده از GSON در قالب کلاس مدل برگرداند. برای استفاده از GSON در Retrofit 2.0 نیاز است تا کتابخانه مربوط به آن را به Maven و یا gradle اضافه نماییم.

کد ٦-٨ برای Maven

```
<dependency>
  <groupId>com.squareup.retrofit2</groupId>
  <artifactId> converter-gson</artifactId>
  <version>2.1.0</version>
</dependency>
```

کد ٦-٩: فایل buil.gradle

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

نیاز است تا متد retrofitConfig را در کد ۶-۷ تغییر دهیم و استفاده از GSON را در آن مشخص کنیم.

کد ٦-١٠: تغییرات اعمالی بر روی retroficConfig

```
private void retrofitConfig() {
        retrofit = new Retrofit.Builder()
                .baseUrl(API_URL)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
}
```

اکنون قصد داریم تا نتیجه دریافتی از میان افزار را نمایش دهیم. برای دریافت تصاویر ذخیره شده در سرور بهتر است از کتابخانه Picasso [3]استفاده شود، زیرا تنها با وارد کردن URL تصویر را در View مورد نظر قرار می دهد و نکته مهم‌تر آن است که این کار با استفاده از یک Thread مجزا از Thread اصلی برنامه انجام می‌پذیرد. برای استفاده از این کتابخانه باید آن را به Maven و یا gradle اضافه کرد.(کدهای ٦-١١و ٦-١٢ و

[1] JavaScript Object Notation

[2] https://github.com/google/gson

[3] http://square.github.io/picasso/

(۶-۱۳

کد ۶-۱۱ برای Maven

```
<dependency>
  <groupId>com.squareup.picasso</groupId>
  <artifactId> picasso</artifactId>
  <version>2.5.2</version>
</dependency>
```

کد ۶-۱۲: فایل buil.gradle

```
compile 'com.squareup.picasso:picasso:2.5.2'
```


کد ۶-۱۳: تغییرات اعمالی برای نمایش پاسخ دریافت شده

```
private void search(String username) throws IOException {
        GithubConnection                connection                =
retrofit.create(GithubConnection.class);
        Call<UserModel> requestModel = connection.user(username);
        wait(true);
        requestModel.enqueue(new Callback<UserModel>() {
            @Override
            public      void      onResponse(Call<UserModel>      call,
Response<UserModel> response) {
                showUI(response.body());
                MainActivity.this.wait(false);
            }
            @Override
            public void onFailure(Call<UserModel> call, Throwable t) {
            }
        });
    }
    private void wait(boolean b) {
        if (b) {
            progressDialog.setMessage("در حال برقراری ارتباط");
            progressDialog.show();
        } else {
            progressDialog.hide();
        }
    }
    private void showUI(UserModel userModel) {
        name.setText(userModel.getName());
        type.setText(userModel.getType());

repository.setText(String.valueOf(userModel.getRepositoriesCount()));

Picasso.with(this).load(userModel.getProfilePic()).into(avatar);
    }
```

خروجی کار به صورت شکل ۶-۲ است.

تصویر ٦-٤ نتیجه ارسال درخواست به میان افزار github

خلاصه

در این فصل پروژه های بزرگ تجاری در اندروید بررسی شد. سازمان ها شبکه های مختص به خود را دارند و برای ارتباط از راه دور از VPN ها استفاده می کنند. علت این کار هم برقراری یک ارتباط امن در یک بستر عمومی مانند اینترنت است. تا به حال تضمین اطلاعات کاربران بر روی موبایل هایشان اهمیت داشت اما در پروژه‌های تجاری باید نکات مهم‌تری نیز مد نظر قرار گیرد. رعایت امنیت در پروژه‌های تجاری بسیار حائز اهمیت است. چون شکست در یک قسمت پروژه ممکن است تمام پروژه را به خطر بیاندازد. یکی از مهم ترین قسمت های برنامه نویسی پروژه‌های تجاری ارتباط با پایگاه داده است. دو دیدگاه در ارتباط با پایگاه داده وجود دارد :

١- مشخصات پایگاه داده را در سورس نگهداری شود و در صورت نیاز اتصال برقرار شود.

٢- استفاده از میان افزار

در روش اول چون اطلاعات حساسی مانند نام کاربری و رمز عبور در سورس نگهداری می شود، در صورتی که سورس توسط مهاجم خوانده شود یک تهدید بزرگ برای پروژه است. استفاده از میان افزار باعث

می شود تا برنامه موبایل بدون نیاز به دانستن پردازش های سمت سرور، اطلاعات مورد نیاز خود را دریافت نماید. همچنین جهت برقراری ارتباط ایمن بین موبایل و میان‌افزار باید از کتابخانه‌های مناسب بهره برد. یکی از این کتابخانه‌ها Retrofit 2.0 است که مزیت‌های بیشتری نسبت به سایر کتابخانه‌ها دارد و دارای bench mark بهتری نیز می‌باشد.

علاوه بر ایجاد امنیت، باید یک ساختار مشخص برای ارسال و دریافت اطلاعات مشخص شود تا بوسیله تمام زبان‌ها و platformها قابل استفاده باشد. برای این مهم قالب JSON بکار می‌رود. درخواست ها از طریق http ارسال می‌شوند و پاسخ‌ها در قالب JSON از میان‌افزار به موبایل ارسال می‌گردد.

۷ پروژه امنیت برنامه‌های کاربردی موبایل

در سال ۲۰۱۶[۱] آمارهایی از آسیب‌پذیری‌های جدید برنامه‌های کاربردی موبایل جمع‌آوری شد؛ آنچه اینجا مشاهده می‌شود نتیجه‌ای از این اطلاعات است.

M1 – استفاده نامناسب از پلتفرم[۲]

M2 – ذخیره ناامن داده[۳]

M3 – ارتباط ناامن[۴]

M4 – احراز هویت ناامن[۵]

M5 – رمزنگاری ناکافی[۶]

M6 – مجوز ناامن[۷]

M7 – کیفیت کد کلاینت[۸]

M8 – دستکاری کد[۹]

M9 – مهندسی معکوس[۱۰]

M10 – عملکرد فرعی[۱۱]

---

[۱] OWASP Mobile Top 10 – 2016
[۲] Improper Platform Usage
[۳] Insecure Data Storage
[۴] Insecure Connection
[۵] Insecure Authentication
[۶] Insufficient Cryptography
[۷] Insecure Authorization
[۸] Client Code Quality
[۹] Code Tampering
[۱۰] Reverse Engineering
[۱۱] Extraneous Functionality

۷-۱ M1- استفاده نامناسب از پلتفرم

۷-۱-۱ عوامل تهدید

به منظور امن شدن پلتفرم اصول و قواعد وضع شده است. هر زمان که این اصول و موارد نقض شوند، باعث بروز این گونه تهدیدات امنیتی خواهند شد. برای مثال intent ها، مجوزها و غیره که اگر به صورت امن انجام نشوند می‌توانند مخاطره انگیز شوند.

۷-۱-۲ جلوگیری از جمله استفاده نامناسب از پلتفرم

کدنویسی امن و تنظیمات درست باعث جلوگیری از این نوع حملات می‌شود. برای مثال اطلاعات حساس نباید از طریق راه‌های ارتباطی قابل شنود منتقل شود و این نوع امن سازی باید در سمت سرور حتما صورت پذیرد.

۷-۱-۳ نمونه ای از استفاده نامناسب از پلتفرم

به دلیل اینکه چندین پلتفرم وجود دارد و هرکدام از این پلتفرم‌ها API مخصوص به خود را دارند، مثال در نظر گرفته شده می‌تواند قابل تطبیق در شرایط مختلف باشد. بعنوان مثال، استفاده از ذخیره سازی محلی بجای Keychain در IOS را در نظر بگیرید. Keychain در IOS محلی امن برای ذخیره سازی داده‌ها است. داده‌های حساس باید درون Keychain ذخیره شوند و به هیچ وجه نباید درون حافظه محلی ذخیره شوند. هنگامی که اطلاعات حساس درون حافظه‌ی محلی ذخیره شوند به دلیل اینکه از پلتفرم استفاده نامناسب شده است، امکان استخراج اطلاعات حساس وجود دارد که این اطلاعات ممکن است برای حملات مورد استفاده واقع شوند.

۷-۲ M2- ذخیره ناامن داده

۷-۲-۱ عوامل تهدید

این نقص امنیتی هنگامی رخ می‌دهد که اطلاعات کاربر به صورت ناامن بر روی د ستگاه ذخیره شده با شد. در صورت سرقت د ستگاه و در اختیار قرار گرفتن آن در د ست افراد سودجو، با لو رفتن اطلاعات محرمانه، امنیت اشخاص در معرض مخاطره قرار می‌گیرد.

۷-۲-۲ جلوگیری از ذخیره ناامن داده

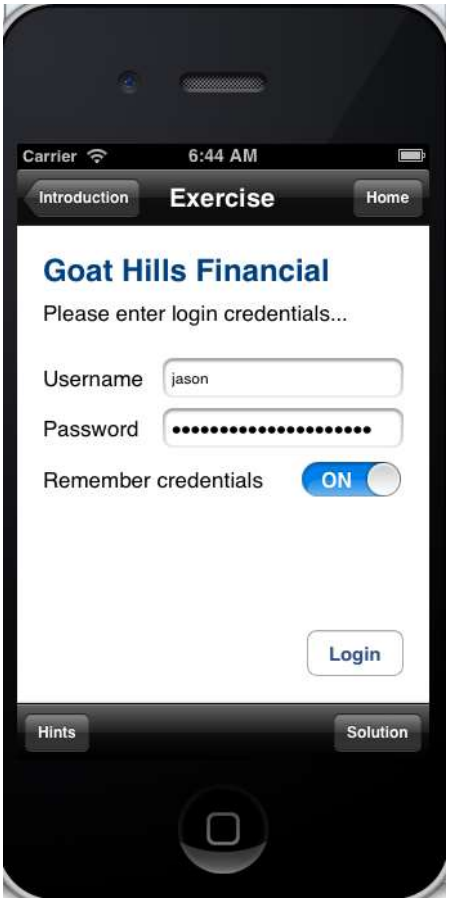
بسیار مهم است هنگام برنامه نویسی، به تهدیدات برنامه، سیستم عامل و فریم ورک[1] توجه شود و طریقه‌ی انجام عملیات مختلف بر روی API ها بررسی و یاد گرفته شود. برای مثال نحوه ذخیره‌سازی URL ها، نحوه‌ی ذخیره‌سازی دکمه‌های فشرده شده بر روی صفحه کلید، نحوه‌ی ذخیره‌سازی عملیات copy/paste، نحوه‌ی ذخیره‌سازی در HTML5، نحوه‌ی ذخیره‌سازی در مرورگرها و مواردی از این قبیل باید آموخته شود.

۷-۲-۳ نمونه ای از ذخیره ناامن داده

برای مثال به پروژه‌ی زیر دقت کنید. در این پروژه در صفحه‌ی بانک، اطلاعات باید در مکان مناسب و امنی ذخیره شوند اما در تصاویر زیر اطلاعات درون حافظه ی تلفن همراه و به صورت فایل credentials.sqlite ذخیره می‌شوند و امکان به سرقت رفتن آن به راحتی توسط مهاجمین مقدور است. به‌عنوان مثال متن (Jason:pleasedontstoremebro!) به صورت یک متن بدون رمزنگاری نمایش داده می شود. همانند آنچه در شکل ۷-۱ و شکل ۷-۲ مشاهده می شود در صورتی که اطلاعات در مکان امن ذخیره نشوند و به صورت متن آشکار و بدون رمزنگاری در مکان ناامن ذخیره شوند امکان سرقت اطلاعات پدید می‌آید.(شکل ۷-۳)

[1] Framework

شکل ۷-۱ محل ذخیره سازی داده



شکل ۷-۲ دریافت ورودی از کاربر

```
mac:Documents haddix$ strings credentials.sqlite
SQLite format 3
Ytablesqlite_sequencesqlite_sequence
CREATE TABLE sqlite_sequence(name,seq)n
;tablecredscreds
CREATE TABLE creds (id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT, password TEXT)
'jasonpleasedontstoremebro!
```

شکل ۷-۳ نمایش اطلاعات محرمانه

۷-۳ M3- ارتباط ناامن

۷-۳-۱ عوامل تهدید

در هنگام تولید یک برنامه کاربردی تبادل اطلاعات بین کلاینت و سرور، باید به این موضوع توجه ویژه ای داشت که دو طرف ارتباط و مسیر ارتباط باید به صورت کامل امن باشند. عوامل تهدید می‌توانند در شبکه محلی، شبکه‌های بزرگتر و حتی درون دستگاه تلفن همراه وجود داشته باشند.

۷-۳-۲ جلوگیری از ارتباط ناامن

برای جلوگیری از ارتباط ناامن، می‌توان از موارد گوناگونی استفاده کرد. به عنوان مثال، استفاده از SSL و TLS به منظور امن سازی شبکه، استفاده از session id، استفاده از رمزنگاری با طول کلید زیاد، استفاده از گواهینامه‌های معتبر و عدم استفاده از راه‌های ارتباطی ناامن همانند SMS.

۷-۳-۳ نمونه ای از ارتباط ناامن

یک نمونه از حمله ارتباط ناامن می‌تواند بدین گونه باشد که در زمان انجام عمل TLS Handshake بین کلاینت و سرور، حمله Man in the middle صورت پذیرد و کلاینت توسط گواهینامهی تقلبی فریب بخورد و اطلاعات را از مسیر ناامن عبور دهد.

۷-٤ M4- احراز هویت ناامن

۷-٤-۱ عوامل تهدید

در این نوع از حملات، هویت اشخاص در زمان احراز هویت دزدیده می‌شود و اطلاعات در اختیار فرد فاقد صلاحیت قرار می‌گیرد.

۷-٤-۲ جلوگیری از احراز هویت ناامن

برای جلوگیری از احراز هویت ناامن باید همواره یک نکته را مدنظر داشت، هرزمانی که امکان انجام عملیات احراز هویت در سرور مقدور است از انجام این عملیات در سمت کلاینت خودداری نماییم.

۷-٤-۳ نمونه ای از احراز هویت ناامن

باتوجه به کدی که برنامه‌نویس نوشته است، تنها کسانی می‌توانند از سرویس استفاده کنند که احراز هویت شده باشند اما برنامه نویس از این موضوع غافل شده است که باید در درخواست های بعدی نیز چک نماید که آیا این درخواست مربوط به یک کاربر شناخته شده است یا خیر. به همین سادگی امکان دریافت و پذیرش درخواستی که عملیات احرازهویت بر روی آن انجام نشده است، پدید می‌آید

۷-۵ M5- رمزنگاری ناکافی

۷-۵-۱ عوامل تهدید

در صورت استفاده از روش‌های رمزنگاری نامطمئن و ناکافی، در صورت شنود پیام‌ها بوسیله افراد غیرمجاز، احتمال شکستن رمز و دسترسی به اطلاعات محرمانه بسیار بالا است.

۷-۵-۲ جلوگیری از رمزنگاری ناکافی

به منظور جلوگیری از این نوع حملات، حتما باید از استانداردهای رمزنگاری پیروی نمود و همچنین از ذخیره سازی اطلاعات حساس بر روی تلفن همراه به شدت اجتناب ورزید.

۷-۶ M6- مجوز ناامن

۷-۶-۱ عوامل تهدید

در این روش عوامل تهدید مواردی می‌باشد که به صورت اتوماتیک نفوذپذیری ها را پیدا می‌کنند. به عنوان مثال، برنامه نویس با استفاده از قابلیت‌هایی که در اختیار دارد ابزارهایی به منظور خودکارسازی تست نفوذ ایجاد می‌کند تا با پیدا کردن آسیب پذیری‌ها بتواند به مجوزها دسترسی پیدا نماید.

۷-۶-۲ جلوگیری از مجوز ناامن

برای جلوگیری از این نوع حملات هیچگاه به اطلاعاتی که از سمت کاربر ارسال می‌شود اطمینان حاصل نکنید و ملاک عملیات‌های احراز هویت و مجوزدهی خود را بر اساس اطلاعات کاربر قرار ندهید. هنگام اعطای مجوز به داده‌هایی که از کاربر در سمت سرور دارید اکتفا کنید و ملاک ارزیابی قرار دهید.

۷-۶-۳ نمونه ای از مجوز ناامن

کاربر درخواستی را برای سرور می‌فرستد که شامل actorID و توکن است. actorID از نظر سرور اشتباه ولی توکن کاملا درست و مطابق با داده‌های ذخیره شده در پایگاه داده است. بنابراین سرور برای بار دوم از کاربر درخواست می‌کند که پارامتر actorID را ارسال نماید. اکنون کاربر می‌تواند actorID مربوط به کاربر دیگری را ارسال کند و به اطلاعات آن کاربر دسترسی پیدا نماید. در این معماری ناامن کاربران می‌توانند اطلاعات یکدیگر را سرقت کنند.

۷-۷ M7- کیفیت کد کلاینت

۷-۷-۱ عوامل تهدید

عوامل تهدید در این روش هنگامی رخ می‌دهد که برای ورودی های اشتباه به متدهای کد کلاینت داده شود. هنگامی که کد کلاینت فاقد کیفیت لازم باشد می‌تواند زمینه ساز استفاده بدافزارها در سمت کلاینت شود.

۷-۷-۲ جلوگیری از حمله کیفیت کد کلاینت

به منظور جلوگیری از هرگونه حملات و افزایش کیفیت کدنویسی، کدها باید مطابق با استاندارد و Design Pattern های شناخته شده نوشته شوند. هنگامی که از بافر استفاده می‌شود توجه شود تا سایز ورودی از سایز بافر بیشتر نشود.

۷-۷-۳ نمونه ای از حمله کیفیت کد کلاینت

در این مثال نمونه ای آورده شده است که باعث پر شدن بافر می‌شود و باید از این کار در برنامه‌های کاربردی اجتناب شود. همانند آنچه در شکل ۷-٤ نمایش داده شد در این برنامه اطلاعات از یک آرایه به عنوان ورودی دریافت می‌شود و در یک بافر کپی می‌شود. سایز بافر ۸ کاراکتر می‌باشد بنابراین هنگامی که مقدار اطلاعات از سایز بافر بیشتر شود کاربر از برنامه کاربردی خارج می‌شود.

```
include <stdio.h>

int main(int argc, char **argv)
   {
   char buf[8]; // buffer for eight characters
   gets(buf); // read from stdio (sensitive function!)
   printf("%s\n", buf); // print out data stored in buf
   return 0; // 0 as return value
   }
```

شکل ۷-٤ پرشدن بافر در برنامه ی کاربردی

۷-۸ M8- دستکاری کد

۷-۸-۱ عوامل تهدید

در این روش مهاجم از طریق دستکاری کد، یک برنامه کاربردی مخرب تحویل کاربران می‌دهد و هنگامی که کاربران از این برنامه استفاده می‌کنند اطلاعات محرمانه آنها نیز برای مهاجم ارسال خواهد شد. همچنین مهاجم امکان دسترسی به تلفن همراه کاربران را می‌تواند داشته باشد.

۷-۸-۲ جلوگیری از حمله دستکاری کد

روش های متفاوتی برای جلوگیری از حمله دستکاری کد وجود دارد. برای مثال در build.prop چک شود که آیا خط ro.build.tags=test-keys توسط توسعه دهنده قرار گرفته است. گواهینامه های OTA چک شود. فایل های SU که به صورت باینری می‌باشند چک شود. فایل build.prop یک فایل سیستمی برای تنظیمات و مشخصات یک برنامه کاربردی اندروید است. OTA مکانیزمی برای بروزرسانی و نصب آن به صورت over-the-air می‌باشد. همچنین برای تبدیل فایل‌های سیستمی به فایل‌های اجرایی، فایل های باینری su به کار می‌رود.

۷-۸-۳ نمونه ای از حمله دستکاری کد

بازی ها معمولا هدف این‌گونه حملات قرار می‌گیرند. برای مثال کاربران یک بازی علاقه ای به پرداخت پول برای تعدادی از ویژگی‌های بازی را ندارند و یک پکیج از بازی را نصب می‌کنند تا بدون پرداخت پول به تمامی قابلیت ها دسترسی پیدا نمایند. بنابراین امکان وجود دارد که توسط پکیج جدید که کدها در آن دستکاری شده‌اند مورد نفوذ قرار گیرند.

۷-۹ M9- مهندسی معکوس

۷-۹-۱ عوامل تهدید

در این روش مهاجم، برنامه‌ی کاربردی مورد نظر را دریافت می‌کند و از طریق ابزارهای مربوطه شروع به بررسی و آنالیز برنامه می‌نماید.

۷-۹-۲ جلوگیری از مهندسی معکوس

برای جلوگیری از مهندسی معکوس باید از ابزارهایی استفاده شود تا کدهای برنامه را بهم بریزد و اصطلاحا کدهای برنامه را تاریک نماید. همچنین برای مقابله با این روش ابزارهایی برای سرهم کردن کد وجود دارد. برای اینکه از عملکرد ابزاری که برای بهم ریختن کد استفاده نموده‌اید، اطمینان حاصل نمایید، ابتدا یک بار کدهای خود را درهم سازی کنید و سپس با استفاده از ابزارهای آماده اقدام به سرهم کردن کد نمایید. ابزارهایی نظیر IDA Pro و Hopper برای سرهم کردن کدهای بهم ریخته بکار می‌روند.

۷-۹-۳ نمونه ای از مهندسی معکوس

در این روش ابتدا مهاجم شروع به دریافت فایل برنامه‌ی کاربردی می‌کند. سپس از طریق ابزارهای آماده کدهای برنامه کاربردی را استخراج می‌کند و به فایل‌هایی همانند manifest، منابع، assets دسترسی پیدا می‌نماید، سپس آنها را به فایل jar تبدیل می‌کند تا به منابع و کدهایی که توسط برنامه‌نویس ایجاد شده است دسترسی نماید.

۷-۱۰ M10- عملکرد فرعی

۷-۱۰-۱ عوامل تهدید

در این روش مهاجم به دنبال درک قابلیت‌های غیراصلی در یک برنامه کاربردی می‌باشد و معمولا بدون دست داشتن کاربر نهایی به صورت مستقیم به بهره برداری از قابلیت های غیراصلی برنامه می‌پردازد. برای مثال برنامه نویس هنگام توسعه یک برنامه ی کاربردی اقدام به قراردادن درپشتی[۱] در برنامه می‌کند یا اقدام به نوشتن پسورد به صورت کامنت شده در کدهای برنامه می‌کند. بنابراین این‌گونه عملکردها می‌توانند زمینه‌ساز عوامل تهدید در این نوع حملات شوند.

۷-۱۰-۲ جلوگیری از حمله عملکرد فرعی

برای جلوگیری از آسیب پذیری در مقابل حمله عملکرد فرعی لازم است هنگام نوشتن برنامه کاربردی تست‌هایی در زمینه امنیت بر روی کدهای برنامه کاربردی صورت پذیرد. برای مثال تنظیمات برنامه‌ی کاربردی برای پیداکردن در مخفی بررسی شود یا در انتهای توسعه برنامه کاربردی توسط ابزارهای آماده تست‌های

[۱] Backdoor

امنیتی بر روی کدها انجام شود. همچنین می‌توان نقاطی که اطلاعات به سمت API ها ارسال می‌شود را از نظر امنیتی بررسی نمود تا قابلیت نفوذ از مهاجمین سلب شود. درنهایت به منظور اطمینان خاطر، می‌توان هنگام انتشار برنامه کاربردی تمامی Log ها را غیرفعال کرد تا مهاجمین از نحوه عملکرد برنامه مطلع نشوند.

۳-۱۰-۷ نمونه ای از عملکرد فرعی

در این روش مهاجم اقدام به تغییر حالت دیباگ برنامه کاربردی می‌نماید و از طریق Log هایی که به مهاجم نمایش داده می‌شود، مهاجم درک درستی از عملکرد برنامه کاربردی پیدا می‌کند و بعد از آن امکان کشف آسیب پذیری‌ها برای او میسر می‌گردد.

خلاصه

در این فصل ۱۰ روش برتر نفوذ در حوزه ی موبایل بررسی شد و عوامل تهدید، جلوگیری از حملات و سناریو نمونه هرکدام بیان شد. باتوجه به اهمیت امنیت سایبری رعایت تمام نکاتی که گفته شد ضروری می باشد.

۸ امنیت در برنامه‌نویسی اندروید

برنامه نویســـان اندروید عمدتا به منظور تولید برنامه‌هایشـــان از زبان جاوا بهره می‌برند. در نتیجه نیاز است تا یکسری از خطاها برنامه‌نویسی رایج در کد نویسی جاوا مورد بررسی قرار گیرد. این خطا ها ممکن است به دلیل بی توجهی و یا عدم اطلاع کافی برنامه نویس پدید آید. این نوع خطاها شرایطی را برای مهاجم فراهم می‌ســـازند تا بتواند با تغییر در کد برنامه، رفتار برنامه را در جهت منافع خویش تغییر دهد. مســـلما هر رفتار دور از انتظار در عملکرد برنامه یک تهدید برای آن محسوب می‌شود.

روش ها و ابزارهای مورد اســـتفاده در هر برنامه، متناســـب با ویژگی‌های آن برنامه می‌تواند متفاوت باشد. استفاده گسترده از این روش‌ها و ابزارها، ممکن است گستردگی اشتباهات برنامه‌نویس‌ها در استفاده از آنها را نیز به همراه دارد. در این فصل قصد داریم تا اشتباهات رایج میان برنامه‌نویسان را دسته‌بندی و بررسی نماییم..

۸-۱ کدهای ناشناس

معمولا برنامه‌نویسان برای افزایش سرعت ایجاد پروژه از کد های آماده در محیط هایی مانند github استفاده می‌کنند. یک مولفه[۱] خارجی که توسط یک برنامه نویس دیگر نوشته شده است ممکن است حاوی یکسری کد مخرب و تهدیدهای بالقوه باشد. اما نمی توان استفاده از کتابخانه های خارجی در پروژه را به طور کامل رد کرد. در اندروید کتابخانه‌های خارجی، دامنه گسترده تری دارند و تقریبا اجتناب از استفاده آنها غیر ممکن اســـت. گاهی ایجاد یک ابزار ممکن اســـت عمده زمان پروژه را به خود اختصـــاص دهد. بدین منظور بهتر است کدها و کتابخانه- ها توسط برنامه نویسان و توسعه دهندگان بررسی شود و همچنین از کتابخانه‌هایی استفاده شود که از منابع مطمئنی منتشر می شوند. بهتر است این کتابخانه‌ها در محیط آزمایشی به صورت کامل بررسی شوند تا از صحت کارکرد آنها اطمینان حاصل شود. استفاده از این نوع ابزارها در پروژه های تجاری بزرگ تا حدودی می‌تواند ضـامن امنیت این کتابخانهها باشـد. اما بهتر اسـت این معیار به عنوان آخرین معیار یک برنامه‌نویس باشد.

همانطور که اشاره شد ممکن است خطاهای برنامه‌نویسی به علت بی توجهی برنامه نویس باشد. یکی از این خطاهای بسـیار خطرناک وجود اطلاعات حسـاس در کد برنامه اسـت. گاهی ممکن اسـت برنامه در مرحله تست باشد و برنامه‌نویس به منظور افزایش سرعت آزمایش، یکسری اطلاعات را که نیاز است کاربر وارد

[۱] Component

کند را به صـــورت ایســـتا در کد قرار دهد. در پایان اگر فراموش کند تا این اطلاعات را حذف نماید ممکن است یک تهدید جدی برای برنامه خود ایجاد کرده باشد. در صورتی که در آزمایش‌ها از این روش استفاده می‌کنید، حتما مطمئن شوید که بعد از آزمایش این اطلاعات را حذف کرده‌اید و یا بهتر است از روش Mock در اندروید استفاده نمایید. ممکن است در آزمایش یک قسمت از پروژه، به خروجی بخش های دیگری نیاز باشد. با استفاده از این روش، می‌توان رفتار بخش های دیگر را طوری شبیه سازی کرد تا مقادیر مورد نیاز بخش در حال آزمایش، تأمین شود. برای مثال برای شبیه سازی رفتار یک صفحه ورود [۱]، بدون اینکه نیاز باشد تا سروری پیاده سازی شود و اطلاعات ورودی به آن ارسال شود و نتیجه برگردانده شود، پاسخ سرور به ازای ورودی صحیح و نادرست را می‌توان با استفاده از روش Mock مشخص کرد. Mockito یکی از ابزار Mocking [۲] در اندروید است.

SQL Injection و ارسال فایل های مخرب دو حمله عمده و مهم در بحث امنیت هستند. علت بروز چنین حملاتی اطمینان برنامه‌نویسان به ورودی های کاربران است. وقتی یک دستور SQL توسط مهاجم به عنوان ورودی ارسال شود، باعث یک رخداد غیر منتظره می شود و ممکن است باعث تخریب بخشی و یا تمام اطلاعات شـــود. ســـازمان‌های فعال در بحث امنیت مانند Mitre [۳] معتقدند که " هیچ اطلاعات ورودی امن نیست". در برنامه‌های بزرگ تجاری، مدیریت تراکنش‌های پایگاه داده از طریق میان افزار انجام می شود. اما بهتر است به عنوان فرستنده اطلاعات نیز، امنیت در ورود اطلاعات نیز رعایت شود. راه حل پیشنهادی مشخص کردن ساختار اطلاعات دریافتی از کاربران است. برای مثال بهتر است از Spinner هایی استفاده کنید که امکان افزودن آیتم به آنها وجود دارد. هر آیتم ورودی را نیز می‌توان کنترل کرد که ساختار مشخصی داشته باشد. برای مثال وقتی یک لیست از انواع مواد غذایی ایجاد شده است تنها ورود حروف مجاز است و استفاده از نمادها مجاز نیست. در نهایت با استفاده از یک سیستم پیشنهاددهی می‌توان میزان اطلاعات ورودی نوشتاری کاربر را کم کرد و اطلاعات انتخابی را جایگزین آن نمود.

۸-۲ رمزنگاری

پس از ایجاد امنیت در ورود اطلاعات، باید امنیت اطلاعات در جابجایی نیز تأمین شـــود. کار با

[۱] Login Form

[۲] https://github.com/mockito/mockito

[۳] https://www.mitre.org/

اطلاعات ناامن همیشه در لیست مشکلات امنیتی گزارش شده توسط OSWAP [۱]، Mitre بوده است. به بیان ساده، در دنیای ارتباطات امروز اطلاعات رمز نگاری نشده، مورد تأیید نیستند. پیاده‌سازی رمزنگاری در پروژه باید متنا سب با آن، شامل یکسری ملزومات با شد. برای مثال رمزنگاری باید در برابر حملات brute-forece مقاوم باشد و لازمه آن بروز بودن روش‌های رمزنگاری است. برای ارسال اطلاعات از اندروید به سرور بهتر است از ابزاری استفاده شود که اطلاعات را به صورت رمزنگاری شده ارسال نماید. همچنین اگر در ذخیره سازی داده از فایل‌ها بهره می‌برید شود، حتما از رمز شدن محتویات آن اطمینان حاصل نمایید. چرا که دسترسی به این فایل ها معمولا کار چندان مشکلی نیست لذا باید داده‌هایی که درون آن‌ها قرار می‌گیرند رمز شده باشند.

۸-۳ بازخوانی و آزمایش کد

بازخوانی کد باعث می شود تا اگر نکته امنیتی از قلم افتاده و نقض شده ا ست، اعمال گردد و سطح امنیت برنامه را بالا رود.. علاوه بر بازخوانی برنامه‌نویس، بهتر است این عمل و آزمایش برنامه به وسیله یکسری ابزار اتوماتیک نیز انجام شود. به این ترتیب دامنه گسترده تری از بررسی ها صورت میپذیرد. Lint [۲] و CTS [۳] می‌توانند ابزار مناسبی برای این منظور باشند. برای مثال با استفاده از CTS یک لیست از پردازش‌هایی که توسط کاربر root انجام می‌شود مشخص خواهد شد.

۸-٤ امضا

امضا [٤] برنامه ها یکی از مهم ترین موارد امنیتی است که باید یک برنامه نویس اندروید به آن توجه کند. هر برنامه قابل اجرا در اندروید باید توسط یک کلید به صورت دیجیتال امضا شود. این کلید شامل یکسری از اطلاعات صاحب برنامه می‌باشد. وقتی یک برنامه امضا می شود، در واقع مثل آن است که برای برنامه اثر انگشت تعریف شده است. وقتی برنامهای امضا شد، در بروزرسانی‌های آتی، اندروید این اطمینان را دارد که هر بروز رسانی منتشر شده برای این برنامه، از سوی برنامه نویس معتبر آن صورت پذیرفته است. وقتی یک برنامه دارای یک ام ضا مختص به خود با شد، در تمام بروزر سانی های بعدی باید از همین ام ضا

[۱] https://www.owasp.org/

[۲] http://tools.android.com/tips/lint

[۳] https://source.android.com/compatibility/cts/index.html

[٤] https://developer.android.com/studio/publish/app-signing.html

استفاده نماید.

وقتی یک نسخه بروز شده برنامه قرار است که نصب شود، گواهی‌نامه و امضا آن باید با نسخه موجود یکسان باشد و تنها در صورت تطابق امضا ها اجازه بروزرسانی از طرف اندورید صادر می‌گردد. اگر امضا جدید با امضا قدیمی تطابق نداشته باشد، باید نام پکیج [1] که به برنامه اختصاص داده‌اید را تغییر دهید و به این ترتیب کاربر یک برنامه جدید را نصب می‌کند و بروزرسانی انجام نمی شود. اندروید مجوز اجرا بر پایه امضاها [2] را پشتیبانی می‌کند.. برنامه هایی که با یک کلید مشخص امضا شده اند می‌توانند اطلاعاتشان را به صورت امن تبادل کنند. تاریخ انقضایی که برای کلید مشخص می شود مشخص کننده طول عمر پشتیبانی برنامه است. وقتی کلید یک برنامه منقضی شود، آن برنامه دیگر قابل بروزرسانی نیست. اگر می‌خواهید برنامه ای را در Google Play قرار دهید، باید تاریخ انقضا آن از ۲۲ اکتبر ۲۰۳۳ بیشتر باشد.

از آنجایی که کلیدها بسیار حائز اهمیت هستند، نگهداری از آنها بسیار مهم است. اگر کلید ها و رمزهای عبور مرتبط به آن در یک فضای ناامن و در دسترس اشخاص دیگر قرار داشته باشد، هویت شما به عنوان برنامه‌نویس پروژه هایتان مخدوش شده است. اگر یک مهاجم به کلید ها و رمزهای مرتبط با آن دسترسی پیدا کند، می‌تواند برنامه مخربی را به نام نسخه بروزرسانی شده برنامه شما، ارائه دهد. گاهی ممکن است به عنوان یک برنامه نویس قصد داشته باشید تا کد پروژه را به اشتراک بگذارید. قبل از انتشار کد بهتر است اطلاعات حساس موجود در build.gradle را که شامل اطلاعات کلید است، خارج کنید. کد ۸-۱ نمونه ای از یک فایل build.gradle است که اطلاعات کلید در آن وجود دارد. برای رفع این مشکل، مانند کد ۸-۲ ، ابتدا باید متغیرهایی در فایلی مانند key.properties مشخص شود، سپس، مقادیر کلید با استفاده از متغیرها مشخص می‌شوند. (کد ۸-۳).

کد ۸-۱ فایل build.gradle

```
android {
    signingConfigs {
        config {
            keyAlias 'keyName'
            keyPassword 'password'
            storeFile file('location')
            storePassword 'password'
        }
}
```

[1] Package

[2] Signature-Base Permissions Enforcement

کد ۸-۲ فایل key.properties

```
store=password
key=password
alias=keyName
file=location
```

کد ۸-۳ فایل build.gradle

```
def keyStoreFile=rootProject.file("keystore.properties")
def properties = new Properties()
properties.load(new FileInputStream(keyStoreFile))
android {
    signingConfigs {


        config {
            keyAlias properties['keyName']
            keyPassword properties['key']
            storeFile file(properties['file'])
            storePassword properties['store']
        }
}
```

توجه: همواره مدنظر داشته باشید که به هنگام انتشار کد، فایل key.properties را به اشتراک نگذارید.

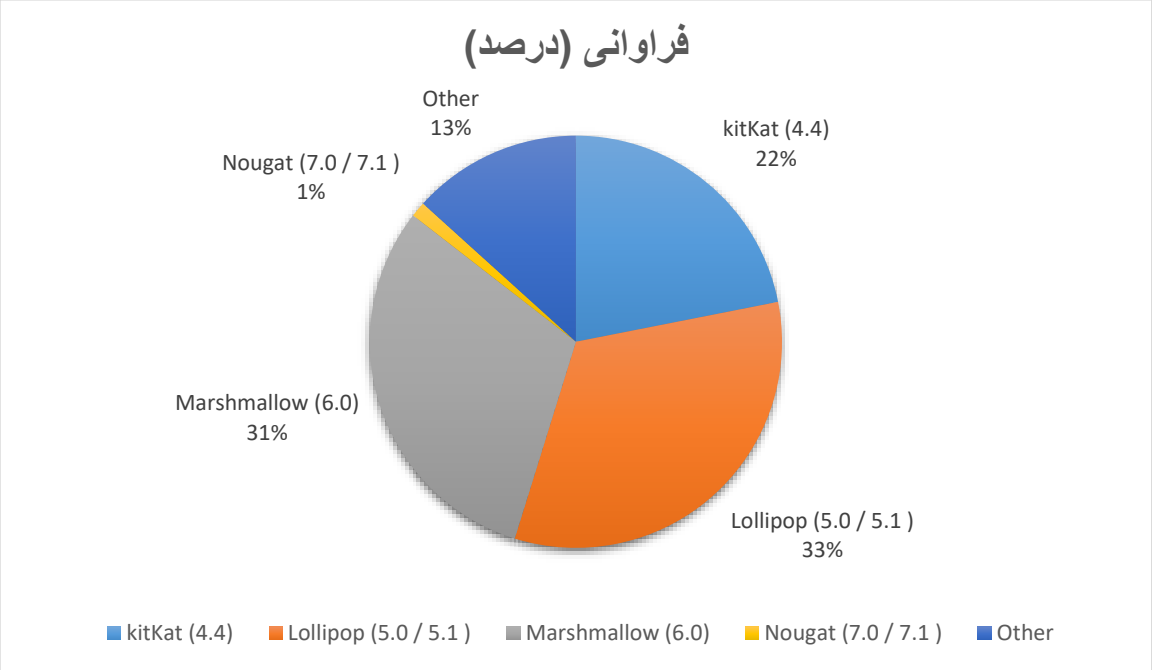
۸-۵ امنیت در نسخه های ارائه شده اندروید

در سیستم عامل اندروید از تدابیر امنیتی تعریف شده در کرنل لینوکس، استفاده شده است.مانند ارتباط امن ما بین پردازش‌ها [۱]، با این عمل برنامه‌های در حال اجرا در پردازش های متفاوت می‌توانند به صورت امن با یکدیگر ارتباط برقرار نمایند. اگر برنامه ای رفتار مخربی داشته باشد، با استفاده از این تدابیر، سیستم عامل آن را کنترل می‌کند تا به برنامه های دیگر و دستگاه آسیبی نرسد.

تا کنون ۱۱ نسخه رسمی از اندروید منتشر شده است. بر اساس آمار ۸۶,۷ درصد از گوشی های اندروید نسخه ٤,٤ به بعد را استفاده می‌کنند [۲] و شکل ۸-۱ نشان دهنده توزیع آن است. معمولا گوگل هر چند ماه یک بروز رسانی امنیتی برای نسخه های درحال پشتیبانی ارائه می‌کند.

[۱] Inter-Process Communication (IPC)

[۲] https://developer.android.com/about/dashboards/index.html#Platform



شکل ۸-۱ توزیع نسخه های اندروید

۸-٦ نسخه ٤,٤

نسخه ٤,٤ اندروید با نام kitkat [۱] در تاریخ ۳ سپتامبر ۲۰۱۳ عرضه شد. در این نسخه امکاناتی از جمله موارد زیر ارائه شد::

- تراکنش امن با استفاده از NFC [۲] و HCE [۳]
- امکان چاپ هر نوع محتوایی با استفاده از wifi و یا سرویس های ابری مانند Google Clound Print
- بهینه شدن مصرف حسگر [٤] های فعال

علاوه بر این امکانات، تدابیر امنیتی نیز بهبود یافت. از تدابیر امنیت اضافه شده به این نسخه می توان به موارد ذیل اشاره کرد :

- در این نسخه اندروید از SELinux در مود اجرایی [٥] استفاده شد.

[۱] https://www.android.com/versions/kit-kat-4-4/

[۲] Near Field Communication

[۳] Host Card Emulation

[٤] Sensor

[٥] Enforcing Mode

- VPN هر کاربر تنها برای خودش قابل استفاده است. در حالت چند کاربره، داده‌های یک کاربر بدون تأثیر بر ترافیک سایر کاربران، می‌تواند از طریق VPN ارسال شود.
- امکان استفاده از الگوریتم‌های ECDSA [۱] و DSA [۲] در کلیدها ، از این نسخه به اندروید اضافه شد.
- دستگاه های اندروید شامل یکسری گواهی‌های تأیید شده می‌باشند. گاهی ممکن است یک گواهی از طریق کاربر، یک مهاجم و یا یک سرویس جدید به دستگاه اضافه شود. در این نسخه اگر گواهی به دستگاه اضافه شود، برای کاربر یک پیغام هشدار صادر می‌شود. این عمل بدان علت است که یک مهاجم می‌تواند ترافیک های رمز شده را با یک گواهی نامعتبر مشاهده نماید.
- در این نسخه از FORTIFY_SOURCE [۳] سطح دو استفاده شده است. با استفاده از این روش از سرریز شدن بافر جلوگیری می‌شود تا باعث آسیب نشود.
- اندروید ٤,٤ گواهی‌های جعلی را شناسایی می‌کند و از استفاده آنها در SSL/TLS جلوگیری می‌کند.

۸-۷ نسخه ۵,۰

نسخه ۵,۰ اندروید با نام Lollipop [٤] در تاریخ ۲۵ ژوئن ۲۰۱٤ عرضه شد. در این نسخه امکاناتی همچون موارد ذیل ارائه شد: :

- در این نسخه ابزارهای مورد نیاز برای طراحی برنامه‌های اندروید به روش Material Design [٥]، در اختیار برنامه‌نویسان قرار گرفت. شرکت گوگل Material Design را به عنوان یک زبان طراحی معرفی کرده است. Material Design یک روش جامع برای طراحی است و استانداردهایی را برای استفاده از جلوه‌های ظاهری مانند پویانمایی [٦]، حرکت ها [۷] ، فاصله ها و سایه ها مشخص می‌کند.
- پردازش ها در این نسخه نسبت به نسخه‌های قبلی، بسیار سریع‌تر و بهینه‌تر انجام می‌شوند و کارایی دستگاه به طور قابل ملاحظه ای افزایش یافته است. با تغییر ظاهر اعلان ها [۸] دسترسی به آنها آسان

---

[۱] Elliptic Curve Digital Signature Algorithm

[۲] Digital Signature Algorithm

[۳] https://fedoraproject.org/wiki/Security_Features?rd=Security/Features

[٤] https://www.android.com/versions/lollipop-5-0/

[٥] http://material.io/

[٦] Animation

[۷] Transitions

[۸] Notifications

تر، قابلیت تنظیم و وضوح آنها بهتر شده است.

در این نسخه عمده تدابیر امنیتی اضافه شده در نسخه قبل بروزرسانی شده است و همچنین قسمت هایی نیز به آن اضافه گردید. موارد ذیل تدابیر اضافه شده به این نسخه است:

- رمزنگاری اطلاعات به صورت پیش فرض فعال است.
- رمز عبور کاربر با استفاده از Scrypt [۱] از حمله Brute-Force محافظت می شود. Scrypt یک روش رمزنگاری KDF [۲] بر پایه رمز عبور [۳] است. با استفاده از روش KDF می‌توان یک کلید را به یک کلید رمز شده طولانی‌تر تبدیل کرد.
- استفاده از SELinux در تمام حوزه ها
- استفاده از قفل هوشمند: با استفاده از این ویژگی می‌توان مکان امن یا یک دستگاه امن را تعریف کرد و اگر دستگاه اندرویدی در آن مکان قرار گرفت و یا دستگاه امن را پیدا کرد، بدون وارد کردن رمز، قفل صفحه باز شود.
- قابلیت استفاده چند کاربره و تعریف کاربر با وضعیت میهمان.
- WebView بدون وابستگی به بروزرسانی تمام سیستم می‌تواند به صورت مجزا بروز شود. در نسخه‌های قبل، WebView همراه با بروز رسانی سیستم عامل بروز می‌شد. اما از این نسخه به بعد، Web view به عنوان یک نرم افزار مستقل قابل بروزرسانی است.
- بروزرسانی رمز نگاری برای HTTPS وTLS/SSL: در این نسخه از TLSv1.2 ، TLSv1.1 استفاده شده است و از AES-GCM بجای MD5 و 3DES و دیگر ابزار رمزنگاری استفاده شده است.
- در این نسخه توابع جدیدی به FORTIFY_SOURCE اضافه شده است و از دستگاه در مقابل آسیب هایی مانند Memory Cruption محافظت می‌کنند. Memory Cruption وقتی رخ می‌دهد که خانه های حافظه به علت خطاهای برنامه‌نویسی، سهوا مقداردهی شده است و باعث می‌شود برنامه با خطا متوقف [۴] شود.

---

[۱] https://en.wikipedia.org/wiki/Scrypt

[۲] Key Derivation Function

[۳] Password Base

[۴] Crash

۸-۸ نسخه ٦,۰

نسخه ٦,۰ اندروید با نام Marshmallow [۱] در تاریخ 28 مه ۲۰۱۵ عرضه شد. در این نسخه امکاناتی از قبیل موارد ذیل به اندروید اضافه گشت:

- در خواست مجوز در هنگام اجرا [۲]
- در این نسخه حالت آماده به کار[۳] به اندروید اضافه شده است. برنامه‌هایی که در حال استفاده نیستند به‌حالت آماده به کار می‌روند. این عمل باعث می‌شود تا باتری بهینه‌تر مصرف شود.
- استفاده از BoringSSL [٤] بجای OpenSSL
- در این نسخه، سرویس‌های دوربین در پردازش های اولویت بالا [٥] قرار داده شده‌اند. این تغییر همراه با ارائه Camera2 API [٦] بوده اســـت. در نســـخه قدیمی Camera API [۷] اگر یک برنامه در حال استفاده از دوربین باشد و برنامه‌ای با اولویت بالاتر بخواهد از دوربین استفاده کند، باعث بروز خطا و فراخوانی متد onError() می‌شـــد. اما در بروز رســـانی جدید اگر این اتفاق رخ دهد، متد onDisconnected() فراخوانی و دوربین به ســـرویس بالاتر داده می‌شـــود و خطایی رخ نمی‌دهد. همچنین مدیریت سرویس درحال استفاده تنها از طریق کاربر آن امکان پذیر است. برای مثال کاربر مهمان نمی‌تواند پردازشی که توسط کاربر اصلی ایجاد شده است را لغو نماید.

همچنین موارد امنیتی ذیل برای افزایش سطح امنیت در نظر گرفته شده است :

- در نســـخه‌های قبل مجوزها در ابتدای نصـــب برنامه به کاربر نمایش داده می‌شـــد. اما در این نســـخه مجوزهای حساس به صورت پیش فرض غیر فعال هستند و در زمان استفاده از کاربر مجوز استفاده را درخواست می‌کنند. به این ترتیب کاربر می‌تواند یک مجوز را صادر و یا لغو نماید.
- دستگاه از زمان boot تا بالا آمدن سیستم عامل از منظر صحت روش های رمزنگاری بررسی می‌شود تا از صحت برنامه های اجرایی دستگاه اطمینان حاصل شود.
- احراز هویت با استفاده از اثر انگشت

---

[۱] https://www.android.com/versions/marshmallow-6-0/

[۲] Runtime Permission

[۳] Standby

[٤] https://boringssl.googlesource.com/boringssl/

[٥] High Priority Processes

[٦] https://developer.android.com/reference/android/hardware/camera2/package-summary.html

[۷] https://developer.android.com/reference/android/hardware/Camera.html

- افزودن لایه جداکننده سخت افزار [1] که در کاربردهای تشخیص اثر انگشت، قفل صفحه و رمزنگاری دستگاه استفاده می‌شود.
- ر سانه‌های قابل جدا سازی مانند حافظه جانبی می‌توانند به عنوان حافظه د ستگاه شناخته شوند و فضـــای داخلی دســـتگاه را افزایش دهند. البته این رســـانه‌ها به صـــورت Block-Level رمزنگاری می‌شوند.

برای دسترسی به فایل‌ها و فضای ذخیره سازی از طریق USB، به تأیید کاربر نیاز است.

۸-۹ نسخه ۷,۰

نسخه ۷,۰ اندروید با نام Nougat [2] در تاریخ ۹ مارچ ۲۰۱٦ عرضه شد. در این نسخه امکاناتی مانند :

- پشتیبانی از Multi-Window
- افزودن امکانات انتخابی برای اعلان ها [3]
- بهینه شدن پردازش های پس‌زمینه [4]
- استفاده از SurfaceView بجای TextureView برای بهینه شدن مصرف باتری

ائه گردید. موارد زیرروشهایی است که با بهبود نسخه‌های قبل در حوزه امنیت و همچنین اضافه شدن تدابیر جدید در این نسخه، برای دسترسی به سطح بالاتری از امنیت استفاده شده است :

- در این نسخه بجای رمز نگاری فضای ذخیره سازی، از رمزنگاری فایل ها ا ستفاده شده ا ست. در این روش اطلاعات کاربران بیشتر از یکدیگر تفکیک می‌شوند و محافظت بیشتری از آن ها صورت می‌گیرد.
- با استفاده از بوت تأیید شده [5] دستگاهی که در معرض خطر است، امکان بوت شدن را ندارد.
- بروز رسانی SELinux
- در این نسخه یکسری روش ها برای محافظت از حافظه مشخص شده ا ست. بخشی از حافظه که کرنل در آن قرار دارد، فقط قابل خواندن [6] است.

---

[1] Hardware Abstraction Layer (HAL)

[2] https://www.android.com/versions/nougat-7-0/

[3] Notifications

[4] Background Process

[5] Verified Boot

[6] Read-Only

- معرفی روش جدید برای امضا که باعث می‌شود تأیید امضا با سرعت بیشتر انجام شود.

خلاصه

معمولا برنامه‌نویسان از زبان جاوا برای تولید برنامه‌های اندروید استفاده می‌کنند. به این ترتیب نیاز است تا با اشتباهات رایج برنامه‌نویسان که منجر به بروز ضعف امنیتی در برنامه می شود، برر سی گردد و با استفاده از روش‌های مناسب، این ضعف ها پوشش داده شوند.

نسخه‌های سیستم عامل اندروید به صورت سالانه بروزرسانی می‌شوند. این بروزرسانی‌ها شامل بروزرسانی در ظاهر، سرویس‌ها و بهینه سازی سیستم عامل از جهات گوناگون می‌باشد. هر بروزرسانی سیستم‌عامل، همراه با بهبودهایی در امنیت سیستم عامل نیز می‌باشد. همچنین اندروید چندین بروز رسانی امنیتی در بازه زمانی چند ماهه پس از انتشار نسخه رسمی ارائه می‌کند.