بسمه تعالی

عنوان مستند

برنامه‌نویسی امن اندروید

مرکز ماهر

تابستان ۹۵

فهرست مطالب

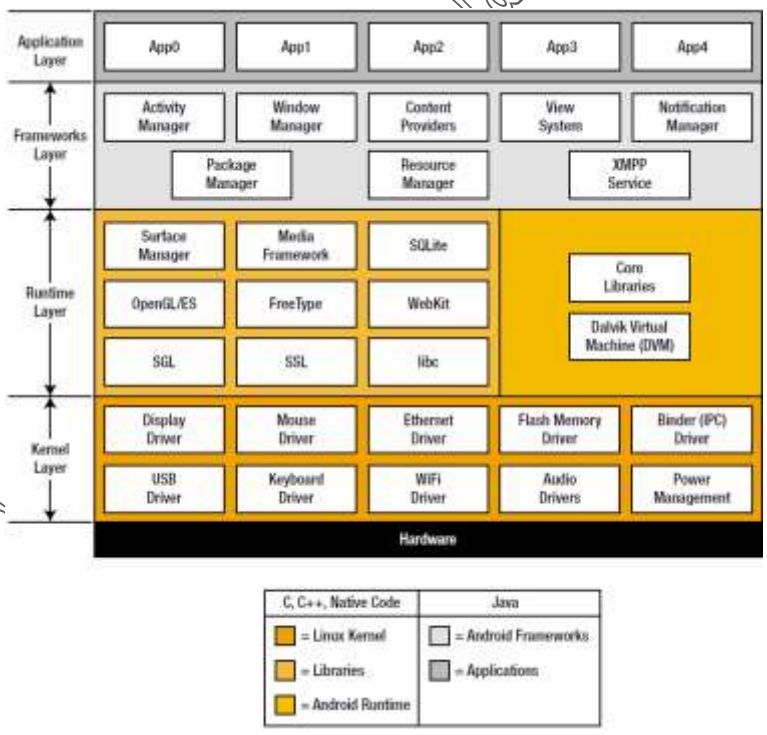
۱ معماری اندروید

یکی از ویژگی‌های منحصربه‌فرد سیستم‌عامل اندروید که موجب رشد سریع آن شده است؛ کدهای باینری و منبع آن است که به‌صورت متن‌باز ارائه‌شده‌اند. هرکسی می‌تواند کد منبع آن را دانلود نماید و برای دستگاه موبایل خود سفارشی نماید.

۱-۱ مؤلفه‌های اندروید

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

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



تصویر ۱-۱ معماری اندروید

[۱] Kernel

[۲] Application

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. اندروید از نسخه های هسته گوناگونی استفاده کرده است چنان که از نسخه ۲,٦,۲۵ در اندروید ۱,۰ به کار برده است. هسته، اولین لایه از نرم‌افزار است که با سخت‌افزار دستگاه در ارتباط است. هسته اندروید مراقب مدیریت نیرو و حافظه، درایوهای دستگاه، مدیریت فرایند، شبکه و امنیت خواهد بود. هسته اندروید در سایت https://android.googlesource.com موجود است.

اصلاح و ساخت یک هسته جدید چیزی نیست که شما بخواهید به‌عنوان یک توسعه‌دهندهٔ نرم‌افزار انجام دهید. به‌طورکلی، فقط تولیدکنندگان سخت‌افزار یا دستگاه می‌خواهند هسته را تغییر دهند تا مطمئن شوند که سیستم‌عامل بر روی سخت‌افزارِ خاص آن‌ها کار می‌کند.

۱-۱-۲ کتابخانه‌ها

کتابخانه‌ها فضایشان را با مؤلفه زمان اجرا[۱] به اشتراک می‌گذارند. کتابخانه‌ها به‌عنوان یک‌لایه ترجمه[۲]، بین هسته و چارچوب برنامه عمل می‌کنند. کتابخانه‌ها در C/C++ نوشته‌شده‌اند اما از طریق یک API جاوا به توسعه‌دهندگان، ارائه‌شده است. 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[۳]: 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

[۱] Runtime component

[۲] translation layer

[۳] Media libraries

بخش مهمی از سیستم‌عامل اندروید است و برنامه‌های بخش ثالث[1] و سیستم را اجرا می‌کند.

۳-۱-۱ ماشین‌مجازی Dalvik

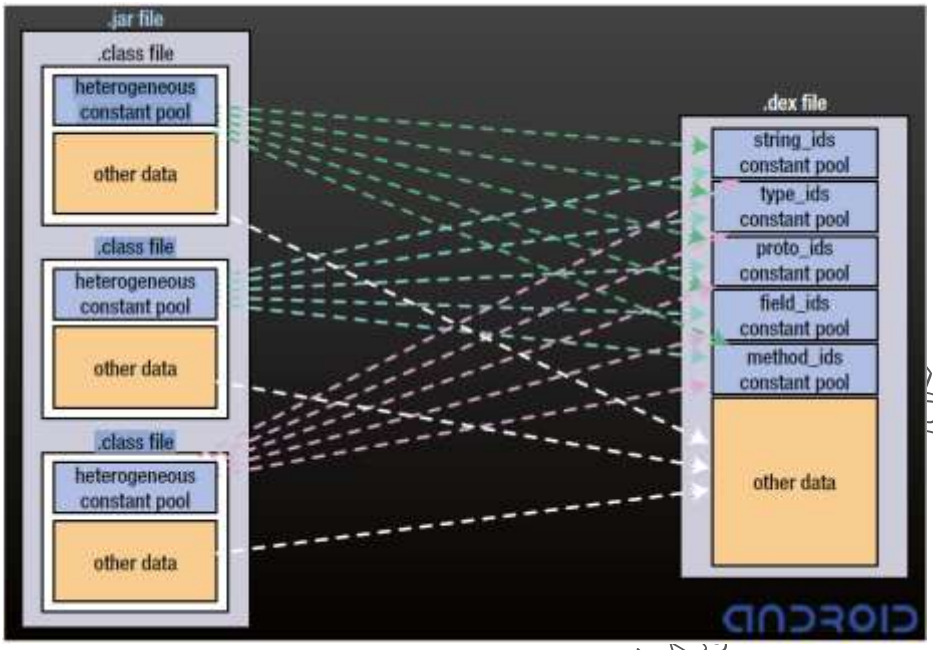
دن‌بورن اشتاین ماشین‌مجازی Dalvik را نوشته است. در درجه اول ماشین‌مجازی Dalvik برای اجرای برنامه‌ها روی دستگاه‌هایی با منابع محدود نوشته‌شده بود که تلفن‌های همراه در این دسته قرار می‌گیرند چراکه آن‌ها نسبت به سایر دستگاه های کامپیوتری از قدرت پردازش، مقدار حافظه در دسترس و عمر باطری کوتاهی برخوردارند.

ماشین‌مجازی چیست؟

ماشین‌مجازی یک سیستم‌عامل میهمان مجزا شده است که در داخل سیستم‌عامل میزبان اجرا می‌شود. یک ماشین‌مجازی برنامه‌های کاربردی را که در حال اجرا بر روی ماشین فیزیکی است اجرا خواهد 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.

ماشین‌مجازی Dalvik فایل‌های dex. را اجرا می‌کند. یک فایل dex. از فایل‌های class. یا jar. کامپایل شده ایجاد می‌شود و همه ثابت‌ها و داده‌های درون هر فایل با پسوند class. در یک مجموعه ثابتِ مشترک جمع می‌شوند.(تصویر۱-۲ را ببینید). ابزار dx در SDK اندروید این تبدیل را انجام خواهد داد. اندازهٔ فایل‌های dex. پس از تبدیل به میزان قابل‌توجهی کوچک می‌شود، همان‌طور که در جدول ۱-۱ می‌بینید.

[1] Third-party



تصویر ۱-۲. تبدیل فایل jar. به فایل dex.

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

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%

۱-۱-٤ چارچوب برنامه کاربردی

چارچوب برنامه کاربردی، یکی از بخش‌های معماری اندروید برای برنامه‌های کاربردی سیستم‌ها یا 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 هشدارها(برای آگاهی صاحبان دستگاه از رخدادها) و مدیر فعالیت (برای مدیریت چرخه‌ی عمر برنامه کاربردی) را برای یک توسعه‌دهنده تدارک می‌بیند.

[۱] Button

[۲] Text box

[۳] content provider

یک توسعه‌دهنده از API ها برای دسترسی به کتابخانه ها استفاده می کند. کد ۱-۱، یک نمونه نمایش از یک API است که نحوه‌ی استفاده از چارچوب برنامه کاربردی برای خواندن فایل‌های تصویری را نشان داده است. عبارت import ، اجازه می دهد تا به هسته‌ی کتابخانه‌های C/C++ از طریق API جاوا دسترسی داشت.

کد ۱-۱: نسخه نمایشی اجرای فایل تصویری

```
/*
* Copyright (C) 2009 The Android Open Source Project
*
* Licensed under the Apache License, Version 2.0 (the "License");
* you may not use this file except in compliance with the License.
* You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package com.example.android.apis.media;
import com.example.android.apis.R;
import android.app.Activity;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.Toast;
import android.widget.VideoView;
public class VideoViewDemo extends Activity {
    /**
    * TODO: Set the path variable to a streaming video URL or a local
media
    * file path.
    */
    private String path = "";
    private VideoView mVideoView;
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.videoview);
        mVideoView = (VideoView) findViewById(R.id.surface_view);
        if (path == "") {
            // Tell the user to provide a media file URL/path.
            Toast.makeText(
            VideoViewDemo.this,
            "Please edit VideoViewDemo Activity, and set path"
            + " variable to your media file URL/path",
            Toast.LENGTH_LONG).show();
        } else {
```

```
            /*
            * Alternatively,for streaming media you can use
            * mVideoView.setVideoURI(Uri.parse(URLstring));
            */
            mVideoView.setVideoPath(path);
            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 دارید.

۱-۲ درباره برنامه‌نویسی امن

حال که شما یک درک کلی از معماری اندروید دارید، 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 می‌شود، 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 می‌شود که شما در حال حاضر محیط توسعه اندروید خود را با استفاده از محیط توسعه Eclipse (IDE) راه‌اندازی کنید. در اینجا، بر این تمرکز می‌شود که شما چگونه می‌توانید برنامه 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 آن را دارد. 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 خود مسئولیت‌پذیر می‌کند. در اینجا به‌منظور درک بهتر مفاهیم امنیتی در رابطه با برنامه‌های کاربردی مثال‌های متنوعی ارائه می‌شود.

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 کامپیوتر رومیزی[1] کار می‌کنند، دارند. در کنار احتمال بالای از دست دادن یا سرقت دستگاه، کاربرانِ دستگاه تلفن همراه درخطر از دست دادن داده‌های حساس و حریم خصوصی خود قرار می‌گیرند. چرا اهمیت این خطرات برای کاربران کامپیوتر رومیزی متفاوت باشد؟ اول، کیفیت داده‌های ذخیره‌شده روی تلفن همراه کاربر، 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 کامپیوتر رومیزی نیز قرار داشته باشند ولی این را در نظر بگیرید: داده‌های روی دستگاه تلفن همراه ارزش بیشتری نسبت به داده‌های روی کامپیوتر رومیزی دارد، زیرا کاربر در تمام مدت آن را همراه خود حمل می‌کند. گوشی هوشمند یک بستر کامل از هردوی تلفن همراه و کامپیوتر رومیزی است که مجموعه‌ای غنی‌تر از اطلاعات شخصی را شامل است.

ازآنجاکه کاربر با گوشی هوشمند خود تعامل مداوم و زیادی دارد؛ داده‌های روی گوشی همیشه جدیدتر از داده‌های روی کامپیوتر رومیزی است. حتی اگر داده های گوشی هوشمند با یک مکان از راه دور یا سرویس

[1] Desktop computer

ابری به طور بلادرنگ همگام‌سازی شده باشد؛ این همگام‌سازی فقط کاربر را در برابر از دست دادن داده‌ها کمک می‌کند و در برابر نقض حریم خصوصی حفاظت نمی کند.

در نظر بگیرید فرمت داده‌های ذخیره‌شده بر روی دستگاه تلفن همراه، ثابت است. هر تلفن همراهی SMS/MMS، تماس‌ها و پست صوتی دارد. تلفن‌هایی که خیلی قوی هستند عکس‌ها، فیلم‌ها، مکان‌های GPS و ایمیل رادارند، اما همه آن‌ها معمولاً بدون سیستم‌عامل هستند. 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 ذخیره‌شده 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 ذخیره‌شده بر روی تلفن همراه فکر شود؛ در اکثر موارد پی برده می‌شود که این داده‌ها از خود دستگاه باارزش‌ترند. خطرناک‌ترین نوع حمله، آن است که در سکوت و از راه دور انجام می‌شود؛ یک مهاجم نیاز به دسترسی فیزیکی به گوشی ندارد. این نوع حملات می‌تواند در هرزمانی اتفاق بیافتد و اغلب به علت ضعف امنیتی در جای دیگر، روی دستگاه اتفاق می‌افتد. این لغزش 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 از حمله‌کننده، حفظ کند.

۳-۱ معماری امنیتی اندروید

همان‌طور که قبلاً بحث کرده‌ایم، اندروید روی کرنل لینوکس ۲,٦ اجرا می‌شود. ما همچنین یاد گرفتیم که کرنل لینوکس اندروید، مدیریت امنیت را برای سیستم‌عامل کنترل می‌کند. در ادامه معماری امنیتی اندروید به‌صورت مختصر بررسی می‌شود.

۱-۳-۱ جداسازی امتیاز

کرنل اندروید وقتی بخواهد نرم‌افزاری را اجرا کند؛ یک مدل جداسازی امتیاز را پیاده‌سازی می‌کند. این یعنی، مانند یک سیستم UNIX، سیستم‌عامل اندروید برای اجراشدن 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[۱] چیست؟

جداسازی امتیازی یک ویژگی مهم امنیتی است، چون با یکی از معمول‌ترین انواع حملات مقابله می‌کند. در بسیاری از موارد، حمله کننده با اولین حمله نمی تواند به تمام اهداف خود برسد 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 می‌شود. 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.

جداسازی امتیازی یکی از ویژگی‌های اصلی طراحی اندروید هست که در هسته پیاده‌سازی شده است. فلسفه‌ی پشت این طراحی این است که اطمینان حاصل شود که هیچ نرم‌افزاری نمی‌تواند کد یا داده‌های برنامه‌های کاربردی دیگر، اطلاعات کاربر یا سیستم‌عامل را بخواند و تغییر دهد. بنابراین، یک برنامه‌ی کاربردی نباید به‌صورت خودسرانه 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 boxهای خود اجرا شدند، درخواست صریح اجازه دسترسی به داده‌های یکدیگر، تنها راهی است که می‌توانند با یکدیگر ارتباط برقرار کنند.

[۱] Privilege separation

۲-۳-۱ مجوزها

اجازه دهید یک مثال ساده بزنم، 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 اضافه کند.

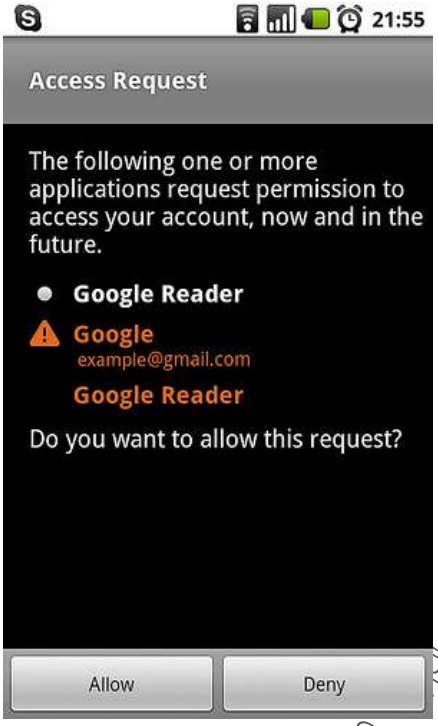
این به نرم‌افزار اجازه می‌دهد تا درخواست مجوز را به بخشی از سیستم ارسال کند که میکروفون را کنترل می‌کند. اما چه کسی درخواست را رد یا قبول می‌کند؟ اندروید به کاربر نهایی امکان می‌دهد تا زمانی که کاربر برنامه کاربردی را نصب می‌کند این تصمیم نهایی را بگیرد ، همانند آنچه در تصویر ۱-۳ مشاهده می‌نمایید. البته در اندروید ٦ ، اندروید مجوز دسترسی برای چنین درخواست‌هایی را در زمان اجرای برنامه نیز از کاربر می‌پرسد.

اگر توسعه‌دهنده صراحتاً نیاز خود را برای مجوز ضبط‌صوت 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 سناریویی که مجوزی توسط گرفتن خطای مناسب داده نمی‌شود را کنترل کند. برای درخواست مجوز باید تگ زیر در فایل پروژه AndroidManifest.xml قرار بگیرد:

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

لیست کامل مجوزها در پایگاه وب Android Developer[۱] موجود است.

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



تصویر ۱-۳ صفحه درخواست مجوزهای اندروید

۳-۳-۱ امضای کد برنامه

هر برنامه‌ای برای اجرا بر روی سیستم‌عامل اندروید باید امضا شود. اندروید از گواهی توسعه‌دهندگان فردی به‌منظور شناسایی آن‌ها و ایجاد روابط مطمئن در میان برنامه‌های مختلف در حال اجرا بر روی سیستم‌عامل استفاده می‌کند. سیستم‌عامل به یک برنامه‌ی بدون امضا اجازه‌ی اجرا شدن را نمی‌دهد. برای امضای گواهی نیازی به استفاده از صدور گواهی نیست، و اندروید به‌راحتی هر برنامه که با گواهی خود امضاشده باشد را اجرا می‌کند.

مانند بررسی مجوزها ، بررسی گواهی‌نامه‌ها فقط در طول نصب برنامه کاربردی انجام می‌شود. بنابراین اگر گواهی توسعه‌دهنده بعدازاینکه نرم‌افزار بر روی دستگاه نصب‌شده، منقضی شود، نرم‌افزار به اجرای خود ادامه می‌دهد. تنها نکته در این مرحله این خواهد بود که قبل از اینکه هر برنامه کاربردی جدیدی تولید شود توسعه‌دهنده به ایجاد یک گواهی جدید نیاز خواهد داشت. اندروید به دو گواهی‌نامه مجزا برای عیب‌یابی نسخه‌های نرم‌افزار و انتشار نسخه‌های یک نرم‌افزار نیاز دارد. به‌طورکلی، محیط Eclipse ابزارهای توسعه اندروید (ADT) را اجرا می‌کند که در حال حاضر برای کمک به توسعه‌دهندگان برای تولید کلیدها و نصب گواهی ارائه‌شده است، به‌طوری‌که قادر به بسته‌بندی و امضا شدن خودکار برنامه‌ها می‌باشند. شبیه‌ساز اندروید مثل دستگاه فیزیکی کار می‌کند. شبیه به دستگاه فیزیکی، فقط برنامه‌های کاربردی امضاشده را اجرا

می‌کند.

خلاصه

همان‌طور که تاکنون دیده‌ایم، اندروید به لطف توجه گوگل، پیشرفت فوق‌العاده داشته است. این مراقبت و توجه، به‌سرعت رشد اندروید در مقایسه با سایر سیستم‌عامل‌های هوشمند در حال رشد کمک کرده است، همچنین دلیل دیگر این نرخ رشد امکان استفاده از سخت‌افزارهای متفاوت برای تولید تلفن‌های همراه توسط تولیدکنندگان 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 تغییر داده می‌شود.

لایه بعدی که در اطراف کرنل 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. زمانی که ماشین‌مجازی Dalvik برنامه‌های کاربردی را می‌خواهد ایمن و کارآمد در یک محیط با منابع محدود اجرا کند، به همین منظور دارای برخی ویژگی‌های منحصربه‌فرد است.

لایه‌های بالایی بعدی که باید اضافه بشوند به ترتیب لایه چارچوب 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 جاوا در آن قرار دارند. هر کتابخانه‌ای که شما مایل به واردکردن آن در برنامه‌تان هستید در آن واردشده است. لایه برنامه‌های کاربردی جایی است که برنامه‌های کاربردی شما درنهایت قرار می‌گیرند و کار خواهند کرد. 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 یک نگاهی به خطرات امنیتی کردیم، و اینکه چگونه می‌توانید مسئولیت امنیت کاربر نهایی خود را داشته باشید ، و همچنین برخی از راه‌هایی که اندروید برای امنیت ارائه داده است را بررسی کردیم . سه حوزه‌ای که ما به آن‌ها نگاه کردیم، جداسازی امتیازی، مجوز، و امضای کد برنامه بود. در فصل

بعد درباره چگونگی استفاده از این ویژگی‌ها توضیح داده می‌شود.

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 مهم‌تر 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 می‌شود و بر روی اینکه چگونه آن‌ها را باید حفظ کرد، تمرکز می‌شود. در این فصل، بر آنچه در هنگام ذخیره‌سازی اطلاعات اتفاق می‌افتد تمرکز می‌شود.

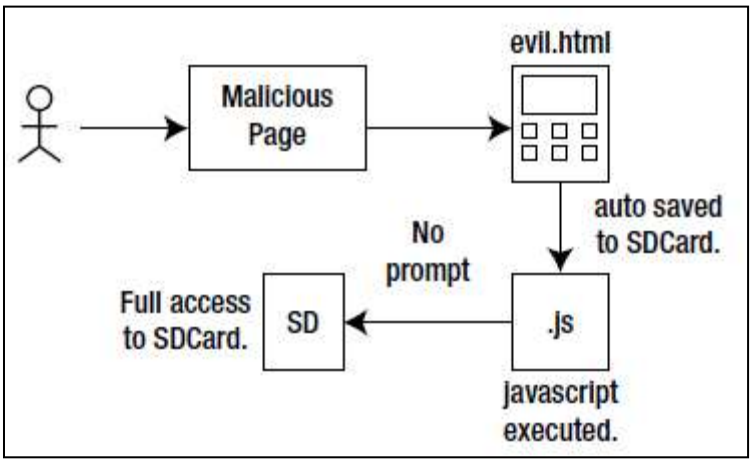
۲-۱ امنیت برنامه کاربردی در برابر حملات

هنگامی‌که داده‌ها ایجاد یا دریافت می‌شوند نیاز دارند تا درجایی ذخیره شوند. 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 می‌شود، نزدیک باشد. 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[۱]

در اواخر سال ۲۰۱۰ ، دو آسیب‌پذیری به ترتیب در نسخه‌های ۲,۲ و ۲,۳ اندروید کشف شد. این آسیب‌پذیری همانی است که در آن‌یک مهاجم می‌تواند هر فایلی که بر روی کارت SD 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.

[۱] Indirect Attacks



تصویر ۲-۱ حمله غیرمستقیم

موارد زیر نکات قابل تأمل هستند:

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

۲. با توجه به یک بخش از آسیب‌پذیری، فایل ***evil.html*** بدون اثر 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. برای بار دیگر، آمادگی برای کاربر نهایی وجود ندارد.

٤. با توجه به بخش آخر آسیب‌پذیری، کد جاوا اسکریپت اجراشده مرحله قبلی، به خاطر اینکه به‌صورت محلی روی دستگاه اجراشده، دسترسی کامل به بارگذاری فایل‌های ذخیره‌شده 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 تحت دایرکتوری خود را بر کارت SD بنویسد. چون آسیب‌پذیری موردبحث است، اطلاعات استفاده‌شده توسط برنامه کاربردی درخطر دزدیده شدن است. هر دستگاه اندروید که برنامه و نسخه نرم‌افزار آسیب‌پذیر 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 شما چه اندازه به معماری و جنبه‌های امنیتی برنامه قبل از شروع به نوشتن کد توجه دارید.

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، برنامه کاربردی تلفن‌بانک؛ یک مهاجم ممکن است وارد برنامه کاربردی تلفن همراهی که متعلق به یک بانک خاص است 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 امن نشده باشد– مهاجم می‌تواند حملات خاصی را دنبال کند که هدفش سوءاستفاده از این ضعف‌ها باشد. این یک 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 درزمانی 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 مایلی از محل کار و خانه باشد برنامه sms برای کاربر فعال می‌شود. به‌این‌ترتیب، همسرش می‌داند که او نزدیک به خانه و محل کارش است. روال ذخیره داده در کد نشان‌داده‌شده است.

کد ۲-۱ روال ذخیره، SaveController.java

```
package net.zenconsult.android.controller;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
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) {
```

[۱] Direct Attacks

[۲] Clear text

```
        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 قبل راه‌اندازی می‌شود. کلاس‌های موقعیت در کد ۲-۲ و تماس در کد ۲-۳ با جزئیات بیشتر ارائه‌شده است. البته می‌توان یک روال 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 net.zenconsult.android.model;
publicclass Location {
    private String identifier;
```

```
    privatedouble latitude;
    privatedouble longitude;
    public Location() {
    }
    publicdouble getLatitude() {
            return latitude;
    }
    publicvoid setLatitude(double latitude) {
            this.latitude = latitude;
    }
    publicdouble getLongitude() {
            return longitude;
    }
    publicvoid setLongitude(double longitude) {
            this.longitude = longitude;
    }
    publicvoid 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();
    }
    publicbyte[] getBytes() {
            return toString().getBytes();
    }
}
```


کد ۲-۳ کلاس تماس، Contact.java

```
package net.zenconsult.android.model;
publicclass Contact {
    private String firstName;
    private String lastName;
    private String address1;
    private String address2;
    private String email;
    private String phone;
    public Contact() {
    }
    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();
    }
    publicbyte[] 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() هستند که تمام محتویات کلاس را به‌صورت یک‌رشته 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، استفاده شود.

کد ۲-٤ کد اضافه نمودن شیء Contact جدید

```
final Contact contact = new Contact();
contact.setFirstName("Sheran");
contact.setLastName("Gunasekera");
contact.setAddress1("");
contact.setAddress2("");
contact.setEmail("sheran@zenconsult.net");
contact.setPhone("12120031337");
```

لحظه‌ای را فرض کنید، زمانی که یک کاربر اطلاعات یک مخاطب جدید را به برنامه کاربردی اضافه می‌کند، کد ٤-۲ فراخوانی شده است. البته می توان به جای استفاده از مقادیر تعریف شده، از متد getText() برای خواندن مقادیر از اشیای EditText که در view اصلی تعریف شده اند، 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: همیشه تمرین خوبی است که از متد getExternalStorageDirectory() برای پیدا کردن موقعیت کارت SD بر روی دستگاه اندروید استفاده کنیم چون اندروید می‌تواند بر روی تعداد زیادی از دستگاه‌ها با مشخصات متفاوت اجرا شود. موقعیت دایرکتوری کارت SD امکان ندارد همیشه در sdcard/ باشد. متد getExternalStorageDirectory() از سیستم‌عامل برای موقعیت درست کارت SD درخواست می‌کند و موقعیت را برای شما برمی‌گرداند.

اجازه دهید با متد سازنده SaveContact شروع کنیم:

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

قطعه کد قبل منتظر یک شیء Context و یک شیء contactاست. 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 در مورد موقعیت مسیر کارت SD است. برای دسترسی به آن، شما باید متد Context.getExternalFilesDir() را فراخوانی کنید. بعدازآن که متد مقادیر موردنظر را دریافت کرد؛ با متد isReadWrite() بررسی می‌کند که کارتSD روی دستگاه نصب و قابل‌نوشتن هست یا نه؟

```
File outputFile = new
```

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

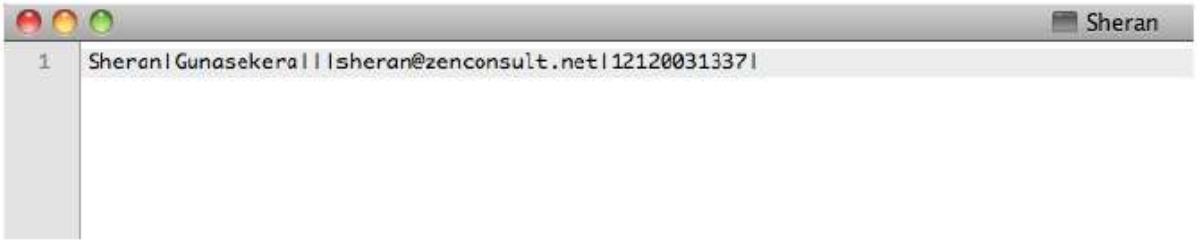
این کد یک شیء فایل ایجاد می‌کند که به موقعیت دایرکتوری کارتSD اشاره می‌کند وازنام شیء Context به‌عنوان اسم فایل استفاده می‌شود.

```
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 بانام "Sheran" وجود داشته باشد که در مسیر Android/data/net.zenconsult.android/files بر کارت SD دستگاه نوشته‌شده باشد.

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



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 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 داشت. اما زمانی که برنامه‌نویسان سعی می‌کنند یک خلاصه مختصر از پروژه را بنویسند موجب می‌شود تا جلوتر از زمان فکر کنند. هرچند که به نظر می‌رسد این نکته واضحی است؛ توسعه‌دهندگان زیادی وجود دارند که موفق به انجام این مرحله ساده نشده‌اند. یکی دیگر از مواردی که باید دقت کرد، توجه به اطلاعات یا داده‌هایی است که توسط برنامه کنترل خواهد شد. برای مثال، در جدول ۲-۱ برخی از انواع داده‌هایی که توسط یک برنامه کنترل خواهد شد نمایش داده‌شده است. 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 شما از نزدیک به جدول طبقه‌بندی داده در جدول ۲-۱ نگاه کنید، متوجه خواهید شد که بعضی از عنوان‌ها بسیار انتزاعی هستند. افراد مختلف نظرات متفاوتی روی‌داده 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[۱] : 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؟

۲-۳-۱ اطلاعات شخصی چیست؟

اطلاعات شخصی می‌تواند 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

[۱] Data Type

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 باکسی 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 کارتSD 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 نیز زیرساخت کلید عمومی و رمزنگاری بیشتر بحث می‌شود. بنابراین ، هدف از این تمرین این است که با مثال‌های بسیار ساده از رمزنگاری، رمزنگاری AES[1] توضیح داده شود. رمزنگاری با کلید عمومی یا رمزنگاری نامتقارن یکی از روش‌های رمزنگاری یا مبهم کردن داده‌ها با استفاده از دو نوع متفاوت از کلیدها هست. هر کاربر دو کلید دارد، یک کلید عمومی و یک کلید خصوصی. کلید خصوصی، تنها داده‌هایی را می‌تواند رمزگشایی کند که توسط کلید عمومی رمزگذاری شده باشد. کلید عمومی، کلیدی است که آزادانه در اختیار دیگران قرارگرفته می‌شود و کاربران دیگر از آن برای رمزنگاری داده‌ها استفاده خواهند کرد.

[1] AES

۲-٤-۱ رمزگذاری

با توجه به مباحث مطرح‌شده 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 AES استفاده نمود؛ این الگوریتم متن را با یک رمز عبور یا یک کلید، رمز می‌کند. یک کلید برای هر رمزنگاری و رمزگشایی موردنیاز است. این روش به‌عنوان رمزنگاری کلید متقارن شناخته می‌شود. برخلاف رمزنگاری کلید عمومی، در این روش فقط یک کلید برای رمزنگاری و رمزگشایی داده‌ها استفاده می‌شود. این کلید باید به‌صورت ایمن ذخیره بشود. کد ۲-۵ روال عادی رمزنگاری را نشان می‌دهد.

کد ۲-۵ روال عادی رمزنگاری

```
privatestaticbyte[] 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;
}
```

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. خط بعدیِ کد، داده‌ی متن آشکار را رمزنگاری می‌کند و محتوای رمزنگاری‌شده را در آرایه بیتی ciphertext ذخیره می‌کند.

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

طبق روال عادی قبل برای این کار، باید یک کلید رمزنگاری‌شده داشته باشد. همچنین مهم است که از کلید یکسان برای روال رمزنگاری استفاده شود. در غیر این صورت، رمزنگاری شکست خواهد خورد. درنهایت بهتر است که مولد کلید را 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 استفاده‌شده است.

کد ۲-۶ الگوریتم تولید کلید

```
publicstaticbyte[] 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 بررسی کد بالا متوجه می‌شوید که این دو خط کد زیر کلاس KeyGenerator را مقداردهی اولیه می‌کند که می‌تواند کلید خاص AES را تولید کند، و سپس دستگاه مولد اعداد تصادفی را مقداردهی اولیه کند که می‌تواند اعداد تصادفی را تولید کند:

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

این اعداد تصادفی با استفاده از روش SHA1 کدگذاری شده‌اند. SHA1[1]، که یک تابع درهم‌سازی رمزنگاری است. این الگوریتم روی قطعه‌ای از اطلاعات عمل می‌کند که دارای یک طول دلخواه است و یک‌رشته‌ی کوتاه با طول ثابت‌شده را تولید خواهد کرد. اگر هر مقدار از اطلاعات درهم‌سازی شده متغیر باشد، پس نتیجه‌ی درهم‌سازی متفاوت خواهد بود. این نشان‌دهنده این است که یک مقدار از اطلاعات دستکاری‌شده است.

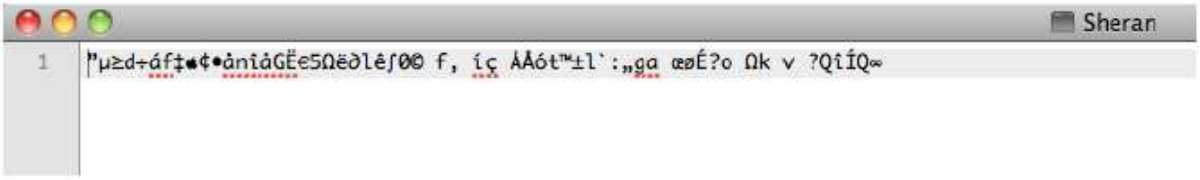
قطعه کد زیر از عدد تصادفی استفاده می‌کند که یک کلید ۲۵۶ بیتی را با استفاده از عدد تصادفی تولید می‌کند:

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

به‌سادگی یک‌بار الگوریتم تولیدکلید را اجرا کنید 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 غیرقابل‌خواندن است.



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).

کد ۲-۷ متد SaveController.java

```
package net.zenconsult.android.controller;
```

[1] Secure Hash Algorithm 1

```
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 یا دیگر داده‌های خواندنی روی دستگاه تلفن همراه اجتناب کند. اگرچه برنامه‌ی کاربردی می‌تواند خودش امن نوشته شود؛ بااین‌حال یک حمله‌ی غیرمستقیم که از یک جای کاملاً متفاوت آغازشده ، می‌تواند اطلاعات حساس یا شخصی نوشته‌شده 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 طول اجرای برنامه‌ی کاربردی هستند، رفتار کرد.

۲. مجموعه روال‌های رمزنگاری را 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 کند. یک الگوریتم تولید کلید خوب نوشته شود که کلیدهای رمزی طولانی و غیرقابل‌پیش‌بینی را تولید کند.

٤. داده‌ها باید در هنگام ایجاد یا ذخیره‌سازی رمزنگاری شود.

۳ معماری امن اندروید

در فصل دوم مثال‌های ساده‌ای از چگونگی استفاده رمزنگاری در حفاظت از اطلاعات طرح شد. اما مثال‌های گفته‌شده در مورد معماری امن اندروید و امنیت داخلی اندروید نبوده است. در این بخش نگاهی می‌شود به مواردی که اندروید برای حفاظت از اطلاعات در اختیار برنامه‌نویس‌ها می‌گذارد و بر حملات مستقیمی که بر روی برنامه‌ها رخ می‌دهد و همچنین به چگونگی جلوگیری از آن‌ها برای مراقبت بیشتر از اطلاعات شخصی افراد، پرداخته می‌شود.

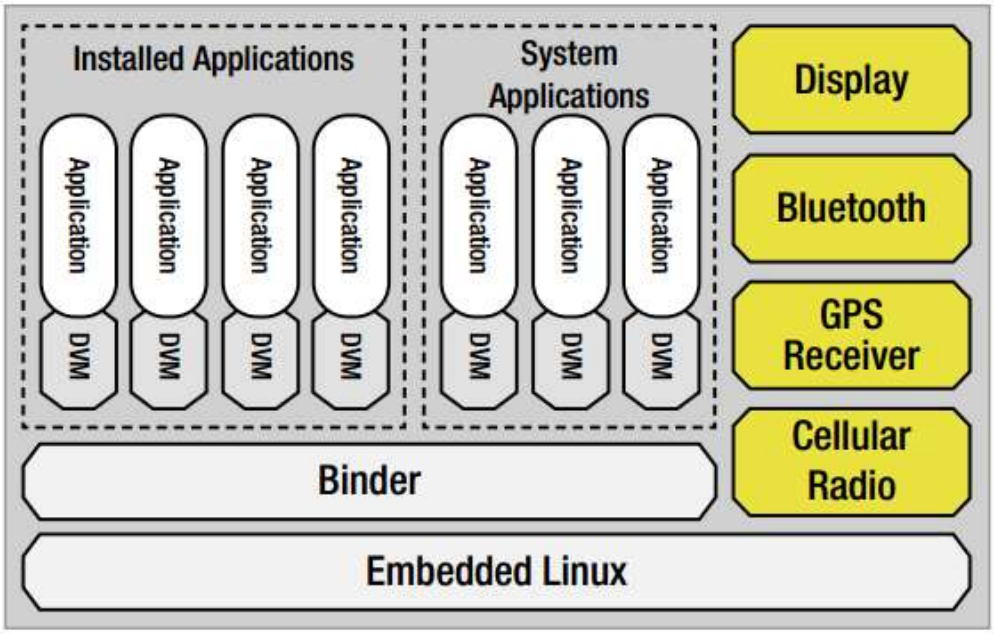
محیط اندروید مکانیسم‌هایی را برای کنترل امنیت سیستم و برنامه‌ها دارد. هر فرایند در اندروید با مجموعه‌ای از امتیازات مخصوص به خود اجرا می‌شود. بنابراین دیگر برنامه‌ها نمی‌توانند بدون مجوز از طرف کاربر به این برنامه یا اطلاعات آن دسترسی داشته باشند.

اندروید APIهای زیادی را در اختیار برنامه‌نویس قرار می‌دهد اما برای استفاده برنامه از همه آن‌ها نیاز است تا کاربر نهایی این دسترسی‌ها را در زمان نصب برنامه، 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.



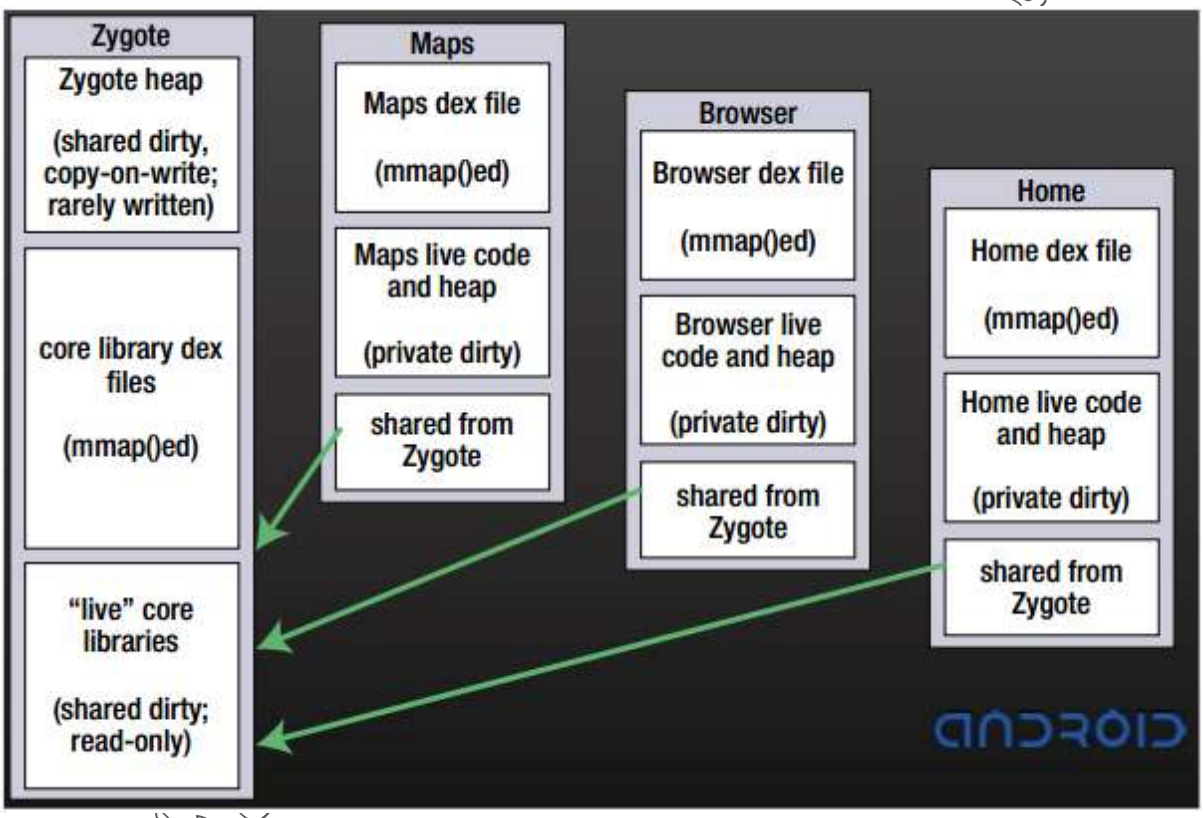
تصویر ۳-۱ معماری اندروید با تمرکز بر برنامه‌ها

تصویر ۱-۳ نمونه ساده‌تری از معماری اندروید را به نمایش می‌گذارد که از نمونه ارائه‌شده در فصل ۲ ساده‌تر است. این تصویر بیشتر بر روی خود برنامه تمرکز دارد. همان‌طور که مشاهده کردید برنامه‌های اندروید بر روی ماشین‌مجازی Dalvik [۱] اجرا می‌شوند. جایی که اساسی‌ترین بلوک‌های کد بر روی آن اجرا می‌شوند، مشابه ماشین‌مجازی جاوا هست که امروزه بر روی کامپیوترهای شخصی و سرورها موجود است. همان‌طور که در تصویر ۱-۳ نشان داده‌شده است؛ هر برنامه پیوسته‌ای در ماشین‌مجازی Dalvik اجرا می‌شود، به‌بیان‌دیگر هر برنامه‌ای بدون تعامل خارجی در طول برنامه دیگر اجرا می‌شود مگر مواردی که مجوز داشته باشند. زمانی که ماشین‌های مجازی مستقل شروع به کار کنند، موجب می‌شوند 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 بعدی را انجام می‌دهد: ۱- در مرحله اول به‌عنوان یک lunch pad برای برنامه‌های جدید کار می‌کند و ۲- 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 کتابخانه که ماشین‌مجازی به آن‌ها احتیاج دارد را بارگذاری و مقداردهی اولیه می‌نماید و سپس منتظر می‌ماند تا سیگنال

[۱] Dalvic virtual machine

راه‌اندازی یک برنامه را دریافت کند. zygote در زمان بوت راه‌اندازی می‌شود و مشابه یک 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؛ او نمونه‌ای از یک ماشین‌مجازی را که جزئی از zygote است را صدا می‌زند. اگر زمانی این نمونه برای راه‌اندازی برنامه‌ها بکار گرفته شود، یک نمونه دیگر از آن مشتق می‌شود تا جای آن را پر کند. برنامه‌ی جدید از نمونه جدید استفاده می‌کند؛ برنامه دیگر از نمونه بعدی و همین‌طور این روند ادامه می‌یابد. بخش مخزن zygote همیشه مجموعه‌ای از کتابخانه‌های هسته را برای برنامه‌ها در طول چرخه زندگی‌شان آماده می‌کند. تصویر ۳-۲ نمایانگر این مطلب است.



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

۳-۲ فهم معماری مجوزها

همان‌طور که در فصل اول بحث کردیم، برنامه‌ها با مجموعه کاربران و گروه‌های هویتی خودشان بر روی سیستم‌عامل اندروید اجرا می‌شوند. روش اجباری در اجرای هر برنامه به برنامه‌ها اجازه‌ی خواندن و نوشتن داده‌های دیگر برنامه‌ها را نمی‌دهد. در راستای تسهیل اشتراک اطلاعات و انجام ارتباطات در طول برنامه‌ها، 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 ندارد. نهایتاً یک برنامه هرگز اجازه ندارد تا اطلاعات یک کاربر را بخواند یا در آن بنویسد. این وظیفه را هسته لینوکس انجام می‌دهد.

برای یک برنامه در صورت نیاز به دسترسی به APIهای ویژه یا داده‌های کاربر لازم است تا از کاربر نهایی اجازه داشته باشد. 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 کنید. پس زمانی که برای اولین بار برنامه توسط کاربر نصب می‌شود و از او در مورد پذیرفتن یا رد کردن مجوزها سؤال می‌شود بهتر است برنامه جوری 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، 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

ارائه‌دهندگان محتوا تقریباً همان ذخیره کنندگان 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 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 imageهای ذخیره‌شده بر روی دستگاه 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 به‌صورت 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

intentها نوعی پیام هستند که یک برنامه برای کنترل وظیفه‌ها یا انتقال داده‌ها به برنامه دیگر می‌فرستد. intent ها با سه مؤلفه از برنامه‌ها کار می‌کنند: فعالیت (activity) ، سرویس (service) و گیرنده پخشی (broadcast receiver). بعضی از اجزای اصلی یک شیء intent شامل intent action و intent data است.

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

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

برای فراخوانی این intent، کد زیر فراخوانی می‌شود:

```
startActivity(intent);
```

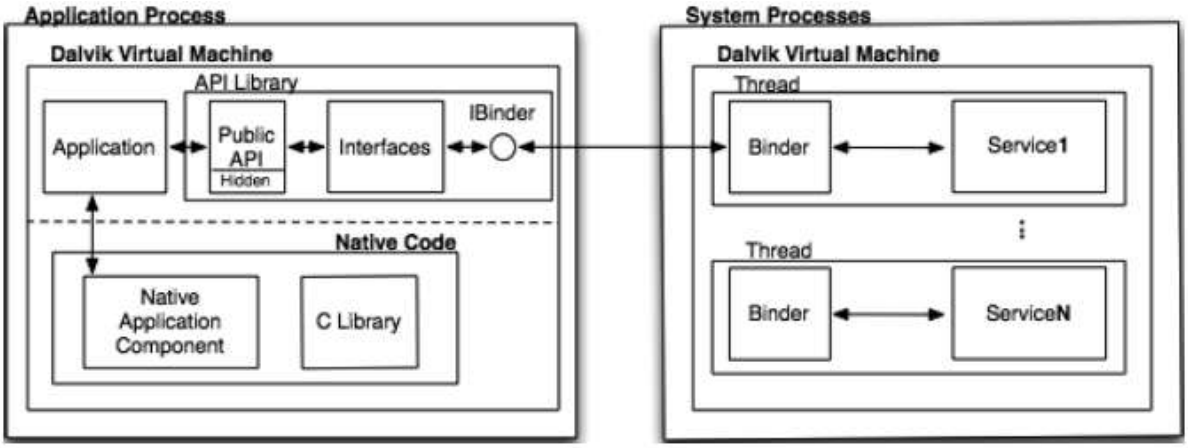
برای کنترل این‌که کدام برنامه intentها را دریافت می‌کنند باید یک مجوز باید به intent قبلی اضافه شود تا آن را هدایت کنند.

۳-۳ بررسی مجوزها

ارائه‌دهندگان محتوا و intent ها به‌طور کامل توضیح دادیم. 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 شما 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 یا SecurityException رخ می‌دهد.

فراخوانی APIها در سه مرحله انجام می‌پذیرد: در ابتدا کتابخانه مربوطه خواسته 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.



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.

شما باید تگ‌ها و مشخصه‌های خاصی را در androidmanifest.xml تعریف نمایید:

کد ۳-۱: androidmanifest.xml

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

ابتدا نام مجوز خود را در android:name تعریف می‌کنید. به android:label و android:description هم نیاز می‌شود که اشاره‌گر به رشته‌ای هستند که در فایل AndroidManifest.xml تعریف می‌کنید. 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 نیاز است.

این امکان را هم خواهید داشت تا از android:permissionGroup استفاده کنید؛ تا مجوز شما در گروه‌های دستگاهی یا گروهی که خودتان تعریف کرده‌اید، قرار گیرد. گروه کردن یک مجوز سفارشی با یک گروه مجوزهای موجود بهترین راه است تا مجوزها را به‌صورت گروهی به کاربر نشان دهید تا بتواند راحت‌تر مجوزها را مرور کند.

برای مثال برای اضافه کردن مجوز purgeDatabase به گروهی که به SD card دسترسی دارد شما باید این مشخصه را به فایل AndroidManifest.xml در بخش تعریف مجوز موردنظر اضافه کنید.

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

نکته قابل‌توجه این است که باید قبل از هرگونه برنامه وابسته‌ای، برنامه شما بر روی دستگاه نصب شود. این مورد معمولی ابتدا ممکن است قابل‌تحمل باشد اما بعد از مدتی در حین پیاده‌سازی حتماً به مشکل بر خواهید خورد.

۲-۳-۳ سطوح محافظت

یک مجوز با مشخصه سطح حفاظت خطرناک یا بالاتر به‌طور خودکار برنامه وادار می‌کند تا برای کاربر پیام هشدار نمایش دهد. این رفتار وجود دارد تا به کاربر درباره یک خطر احتمالی آگاهی دهد و همچنین به کاربر پیشنهاد اعطا یا رد مجوز درخواستی برنامه را 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.

۳-۳-۳ کدهای نمونه برای مجوزهای سفارشی

کد ۳-۲: کدهای نمونه برای مجوزهای سفارشی

```
package net.zenconsult.libs;
public class Mofest {
    public Mofest(){
    }
    public class permission {
        public permission(){
            final String PURGE_DATABASE =⊃
            "net.zenconsult.libs.Mofest.permission.PURGE_DATABASE";
        }
    }
}
package net.zenconsult.libs;
import android.app.Activity;
import android.os.Bundle;
public class ZenLibraryActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

```
    }
}
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.zenconsult.libs"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk android:minSdkVersion="10" />
    <permission

android:name="net.zenconsult.libs.Mofest.permission.PURGE_DATABASE"
        android:protectionLevel="dangerous"
        android:label="@string/label_purgeDatabase"
        android:description="@string/description_purgeDatabase"
        android:permissionGroup="android.permission-group.COST_MONEY"/>
    <uses-permission
android:name="net.zenconsult.libs.Mofest.permission.PURGE_DATABASE" />
    <uses-permission android:name="android.permission.SET_WALLPAPER" />
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity
            android:name=".ZenLibraryActivity"

android:permission="net.zenconsult.libs.Mofest.permission.PURGE_DATABASE"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

خلاصه:

در این فصل نگاهی بر مجوزهای استاندارد و تعریف‌شده توسط کاربر داشتیم. همچنین طور intent ها و ارائه‌دهندگان محتوا را دبر جزئیات بیشتر موردبررسی قراردادیم، نکات مهمی که به دست آمد شامل موارد زیر است:

- اندروید مجموعه‌ای از مکانیسم‌ها دارد 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 مجوز از کاربر، اجازه تبادل داده‌ها را باهم ندارند.
- ارائه‌دهندگان محتوا ذخیره و دسترسی به داده‌ها را ممکن می‌کنند، آن‌ها تقریباً شبیه به پایگاه داده‌ها

هستند.

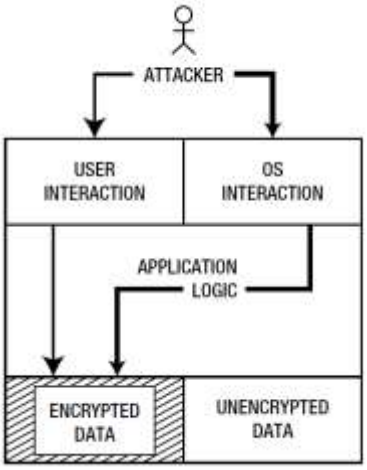
- Intent ها پیام‌هایی هستند که بین برنامه‌ها برای فراخوانی یا قطع یک سرویس یا برنامه دیگر ردوبدل می‌شوند.
- دسترسی به APIهای خاص با استفاده از مجوزها کنترل می‌شود. مجوزها به چهار گروه تقسیم می‌شوند که مجوزهای سطح یک، دو و سه همیشه به کاربر هشدار می‌دهند تا زمانی که این مجوزها احتمال تأثیر مخرب بر اطلاعات کاربر دارند برای تائید نهایی به کاربر اعلان می‌شوند.

مجوزهای سفارشی برای محافظت از منابع برنامه‌های مستقل ایجاد می‌شود. برنامه شما برای استفاده از منابع از 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 ٤ رمزنگاری را به‌طور مختصر بررسی کردیم. در این فصل روی اهمیت استفاده از رمزنگاری به‌منظور مبهم و ایمن کردن داده‌هایی که کاربر ذخیره خواهد کرد یا انتقال خواهد داد بیشتر متمرکز می‌شویم. ابتدا اساس رمزنگاری و چگونگی اعمال آن‌ها در مفهوم برنامه‌ها را پوشش می‌دهیم. سپس به روش‌های مختلف ذخیره‌سازی داده روی پلتفرم‌های اندروید می‌پردازیم. در بین راه برای چگونگی ذخیره و بازیابی داده‌ها از روش‌های مختلف نمونه‌هایی را شرح می‌دهیم و هر تابعی که به‌طور ایدئال برای پلت فرم مناسب است را به‌طور مختصر بیان می‌کنیم.

نکته بسیار مهمی که باید به خاطر داشته باشید این است که هیچ‌گاه تلاش نکنید که روال رمزنگاری را یادداشت کنید مگر اینکه با موضوعات رمزنگاری آشنایی داشته باشید. بسیاری از توسعه‌دهندگان را دیده‌ایم که تلاش می‌کنند تا این کار را انجام دهند و درنهایت هم در دستگاه‌های موبایل و هم در برنامه های کاربردی وب با برنامه های کاربردی آسیب‌پذیر مواجه می‌شوند. رمزنگاری موضوع بسیاری وسیعی هست و برای گروهی که زندگی‌شان را به این موضوع اختصاص داده‌اند بهترین مورد هست. به‌عنوان یک توسعه‌دهنده برنامه کاربردی شما تنها می‌توانید به یک زیرمجموعه خاص از این موضوعات رمزنگاری علاقه‌مند باشید. تاریخچه رمزنگاری را پوشش نخواهیم داد. شما تنها باید این نکته را به خاطر بسپارید: داده‌های کاربری حساس خود را برای کاربران غیرمجاز غیرقابل خواندن کنید. اگر یک مهاجم با یک حمله مستقیم یا غیرمستقیم به برنامه کاربردی شما حمله کند، آنگاه لایه اضافی رمزنگاری شما (تصویر ٤-١ را مشاهده کنید) اجازه دسترسی به داده‌های کاربری حساس را به او نمی‌دهد. حمله‌کننده باید ابتدا از لایه اضافی عبور کند تا حمله موفقی داشته باشد. این قوانین مشابه اصول تضمین اطلاعات دفاع در عمق است که آژانس امنیت ملی در ایالات‌متحده توسعه داده است.



تصویر ٤-١: مثالی از دفاع در عمق

٤-١ پیدایش کلید عمومی

ازآنجایی‌که بحث رمزنگاری مطرح است؛ کمی پاگیری در رابطه با پیدایش کلید عمومی (PKI) ارزشمند است. PKI بر اساس اصول شناسایی و تائید اطمینان مبتنی بر شخص ثالث مورد اعتماد هست. 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 را عمیقاً بررسی خواهیم کرد.

آقای Krabs صاحب Krusty Krab، یکی از مشهورترین رستوران‌های غذای فوری هست. او مشهورترین Krabby Patty (یکی از خوشمزه‌ترین برگرها) با دلیلی برای شهرتش را ایجاد کرد. هیچ‌کس به‌جز آقای Krabb دستورالعمل فوق سری این برگر خوشمزه را نمی‌داند. با توجه به شهرتش، او اخیراً امتیاز رستورانش را می‌فروشد. همان‌طور که بیشتر زیرشاخه‌های تحت امتیاز او ازنظر جغرافیایی دور می‌باشند، آقای کربز تصمیم دارد که راز دستورالعمل خود را با پیک به آن‌ها انتقال دهد. تنها مشکل این روش این است که رقیب آقای کربز، آقای Sheldon James اخیراً سعی داشت که راز این دستورالعمل را برباید و احتمالاً دوباره تلاش خواهد کرد.

ما غذا را خیلی دوست داریم مخصوصاً برگر بنابراین تصمیم به بازگشایی یک رستوران Krustry Krab در شهر خود را گرفته‌ایم. با آقای کربز تماس می‌گیریم و همراه با برگه مربوطه، او سندی را انتقال می‌دهد که شامل چگونگی دریافت و حفظ کردن راز دستورالعمل Krabby Patty هست. در اینجا دستورالعمل‌هایی که انجام می‌دهیم به این صورت هست:

- در نزدیک‌ترین اداره پلیس تحت برنامه KK از طریق بخش IV ثبت‌نام کنید.
- یک قفل با یک کلید دریافت کنید که قفل دریافت شده از بخش IV اداره پلیس را باز می‌کند.
- قفل را به اداره پلیس تحویل بدهید.
- از کلید محافظت کنید.
- جعبه فلزی که با پیک برای ما فرستاده می‌شود را دریافت و بازکنید.

مطمئناً زمانی که این مراحل را تکمیل کردیم، یک بسته با پست می‌رسد. به طرز عجیبی، قسمت خارجی مقوای بسته دستکاری‌شده است، اما قفل یا جعبه فلزی جامدی درون آن دست‌کاری نشده است. کلید قفل را به‌راحتی باز می‌کند. از دستورالعمل Krabby Patty راداریم. بعدها، از آقای کربز می‌شنویم که آقای پلانکتون سعی بر دزدی ماشین پست و باز کردن جعبه فلزی را داشته است اما موفق نشده است. و این نشان می‌دهد که مقوای بیرونی دستکاری‌شده است همان‌طور که متوجه شدیم. برای روشن‌تر شدن موضوع، ویژگی‌ها و موضوعات مرتبط با عناصر PKI در این داستان را شرح می‌دهیم.

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

آقای پلانکتون: 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 استفاده کند تا یک پیام را رمزنگاری کند یا بگشاید.

از اینکه قفلمان یا کلید عمومی را به هرکسی بدهیم واهمه‌ای نداریم زیرا تنها ما می‌توانیم پیام را بازکنیم. می‌توانیم برای اختصاص دادن به هرکس که می‌خواهد یک پیام ایمن ارسال کند تعداد نامحدودی قفل داشته باشیم.

کلید قفل : این کلید شخصی است، شخصی است زیرا هیچ‌کس مثل آن را ندارد. تنها ما با این کلید قادر به باز کردن قفل خود هستیم. ما 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 پیدا می‌کند.

اداره پلیس: یک مرکز صدور گواهی است (CA). یکی از عناصر اساسی یک PKI، CA معادل یک شخص ثالث مورد اعتماد است.هر دو آقای کربز 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 نمی‌شناسیم یا تاکنون ندیده‌ایم بخواهد به ما یک پیام امن ارسال کند، لازم نیست که در مورداطمینان به‌فرد نگران باشیم. تنها باید به قانون و حقی اعتماد کنیم که می‌گوید او فردی که خود می‌گوید است.

برنامه KK: برای داستان این محدوده CA است. برای مثال اداره پلیس یا CA ممکن است قادر باشد که برای بسیاری از سناریوهای مختلف به‌عنوان شخص ثالث عمل کند. دامنه CA اطمینان خواهد داد که تمام تراکنش‌ها درزمینهٔ یکسانی اتفاق خواهد افتاد. بنابراین برنامه KK تنها برای پرداختن به امتیاز آقای کربز هست.

اداره IV: RA 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 خود را با یک سند رسمی صادرشده مثل کارت شناسایی ملی یا گذرنامه ثابت کنید. 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 برای شما پیش بیاید این 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 کربز و ما برای اطمینان از جلوگیری ارسال یا دریافت پیام به شبه،هر دو از یک شخص ثالث مورداطمینان استفاده می‌کنیم. بنابراین در مورد این سیستم چطور؟دو روش کلیدی برای حمله به این سیستم وجود دارد: ۱- مهاجم می‌تواند درروند ثبت‌نام فریب ایجاد کند و هویت یک کاربر مشروع را برباید،۲- مهاجم می‌تواند یک حمله فیزیکی به پیام رمز شده در حال انتقال انجام دهد.

فایده این زیرساخت این است که اگر آقای پلانکتون تلاش کند که خود را به‌جای آقای کربز 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 را به کار می‌گیرد. این قفل‌ها الگوریتم‌های رمزنگاری هست که استفاده می‌شوند.

٤-٢ اصطلاحات مورداستفاده در رمزنگاری

یادگیری اصطلاحات به‌صورت درست بسیار ضروری است بدون یادگیری اصطلاحات درست شما می‌توانید رمزنگاری کنید اما احتمالاً با سرعت کمتری. جدول ٤-١ اصطلاحات مورداستفاده در رمزنگاری را درزمینه‌ی نوشتن و ایمن‌سازی برنامه های کاربردی خود فهرست می‌کند.

جدول ٤-١ اصطلاحاتی که در رمزنگاری استفاده می‌شود

اصطلاح	توضیحات
Plaintext/cleartext	این پیام شما است، فایل متنی است که شما می‌نویسید، اطلاعات کاربری که شما ذخیره می‌کنید یا پیام سطری که شما تمایل دارید در برابر دیگران محافظت شود عموماً توسط هرکسی قابل خواندن است.
Encryption	این فرایند برای گرفتن متن واضح و تبدیل آن به یک متن مبهم یا غیرقابل خوانا استفاده می‌شود.
Ciphertext	این نتیجه رمزنگاری متن واضح است. پیام رمزشده هست.
Cryptographic algorithm/ algorithm/cipher	یک نوع تابع خاص ریاضی است که برای رمزنگاری و رمزگشایی متن واضح استفاده می‌شود.
Decryption	معکوس رمزنگاری است. فرایندی است که توسط آن متن رمزی مبهم به متن واضح قابل‌خواندن تبدیل می‌شود.
Key	این مقدار به‌طور واحد روی الگوریتم رمزنگاری و رمزگشایی تأثیر می‌گذارد. آنجا 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 باشد. می‌توانید این نوع کلید را برای رمزنگاری داده‌ها برای فرد خاصی استفاده 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	این به مطالعه شکستن متن رمزی بدون دانش قبلی کلید یا الگوریتم برمی‌گردد.

٤-٣ رمزنگاری در برنامه های کاربردی موبایل

به نظر می‌رسد پیاده‌سازی PKI در برنامه های 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 کنیم. بنابراین اجازه دهید بررسی کنیم که چه رمزنگاری را متناسب با برنامه‌هایمان می‌خواهیم؟ همان‌طور که در فصل قبل اشاره کردیم، حفاظت از اطلاعات کاربر بسیار مهم است. در مثال فصل ۲ از الگوریتم استاندارد رمزنگاری پیشرفته(AES) استفاده کرده‌ایم. این‌یک الگوریتم کلیدی متقارن است زیرا تنها یک کلید برای رمزنگاری و رمزگشایی وجود دارد.

٤-٣-١ الگوریتم‌های کلیدی متقارن

AES یک الگوریتم کلیدی متقارن یا رمز بلوکی است. همان‌طور که مشاهده می‌کنیم، 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،می‌توانیم تعداد ثابتی از داده‌های بیتی را که به‌عنوان یک بلوک داده شناخته می‌شوند را در یک‌زمان پردازش کنیم یا می‌توانیم داده‌های تک‌بیتی که به‌عنوان جریان شناخته می‌شوند را در یک‌زمان پردازش کنیم. این تمایز، بلوک

رمز شده و جریان رمز شده را به ما 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 کرده‌ایم. 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	1+

٤-۳-۲ تولید کلید

کلید بخش جدایی‌ناپذیر رمزنگاری است. بیشتر الگوریتم‌های مدرن رمزنگاری برای صحیح کار کردن نیازمند یک کلید می‌باشند. در مثال فصل ۲، از یک تعداد مولد شبه تصادفی (PRNG) برای تولید کلیدهای رمزنگاری استفاده کردیم (را ببینید). یک قانون خوب این است‌که همواره بزرگ‌ترین اندازه کلید یک الگوریتم را همواره انتخاب کنیم. اگر برنامه به‌کندی کار می‌کند و پاسخ می‌دهد پس بهتر است اندازه کلید را به کوچک‌ترین کلید بعدی کاهش دهیم. در رمزنگاری، شما همیشه می‌خواهید بزرگ‌ترین اندازه کلید ممکن را برای الگوریتم استفاده کنید. علت آن این است که انجام brute-force که به کلید شما حمله می‌کند را دشوارتر می‌کند. برای نشان دادن، فرض کنید که شما یک کلید با اندازه ۱٦بیت را انتخاب می‌کنید. این به این معنی است که مهاجم باید ترکیبی از یک 1s و 0s به‌صورت کلی٦٥٥۳٦ بار را آزمون کند. اگر شما اندازه کلید ۲۵٦بیتی را انتخاب کنید، پس مهاجم باید 2^{256} یا 11.6^{77} بار تلاش کند تا کلید شمارا 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 با متن رمزی شما کار کند. در اغلب موارد داده‌های رمز شده حکم عامل بازدارنده برای مهاجم رادارند. به‌جای گذراندن وقت روی شکستن رمزنگاری شما، آن‌ها به‌آسانی برای حمله به برنامه کاربردیبرنامه 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 ایجاد 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 ۲۵۶بیتی تصادفی استفاده کردیم. به‌طورکلی روال رمزنگاری همه در پشت‌صحنه اتفاق می‌افتد؛ اگرچه می‌توان رمزهای کاربری را به کلیدها تبدیل کرد ولی این کار به‌هیچ‌وجه توصیه نمی‌شود. 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 (۲۵۶/۸=۳۲) را پوشش نمی‌دهد. با توجه به آنچه در مورد حملات brute-force می‌دانیم بهتر است که حداکثر طول کلید قابل‌قبول را انتخاب کنیم.

کد٤-١: 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 کرده‌ایم. بااین‌حال در مواردی که اطلاعات شما کمتر از اندازه بلوک ورودی موردنیاز توسط الگوریتم است چه باید کرد؟ مورد تصویر ٤-٢ را در نظر بگیرید. دو بلوک اطلاعات راداریم اما تنها یکی از آن‌ها شامل اندازه بلاک کامل هست (از یک اندازه بلوک ۸بایتی برای سادگی کارها استفاده خواهیم کرد) دومی تنها شامل ٤بیت هست. اگر این بلاک آخری را با الگوریتم AES پردازش کنیم، شکست 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 صفر است. به‌عنوان لایه گذاری صفر شناخته می‌شود. گزینه‌های مختلف لایه گذاری دیگری نیز وجود دارد. نمی‌خواهیم وارد جزئیات شویم اما باید به خاطر بسپارید که نمی‌توانید به‌راحتی یک متن را پردازید و از طریق یک روش رمزنگاری بلوکی پردازش کنید. 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 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:

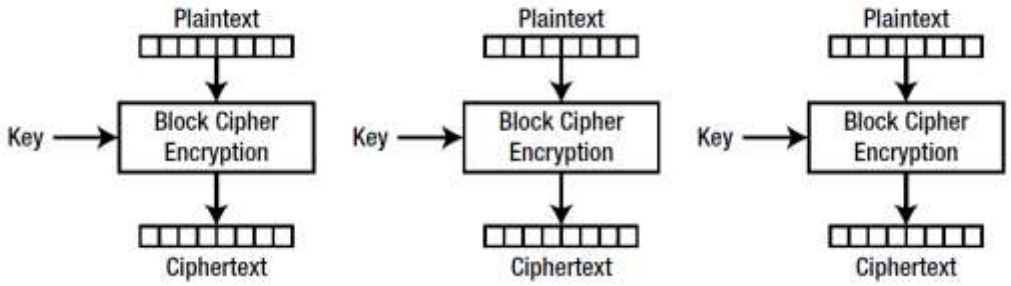
در لایه گذاری PKCS5/7 مقدار عددی طول بیت‌های باقی‌مانده را به‌عنوان یک لایه استفاده می‌کند. برای مثال، اگر ١٠بیت برای لایه گذاری نیاز باشد، پس بیت لایه گذاری 0A است (که در هگزادسیمال به‌صورت 10 است). به‌طور مشابه، اگر ٢٨بیت برای لایه گذاری وجود داشت آنگاه بیت لایه گذاری 1C می‌بود.

در مثال فصل ٢ هیچ لایه گذاری تعیین نشده است. به‌صورت پیش‌فرض اندروید از لایه گذاری PKCS5 استفاده می‌کند.

٤-٣-٤ حالت عملیات و رمزهای بلوکی

رمزنگاری بلاکی روش‌های رمزنگاری و رمزگشایی متنوعی دارند. ساده‌ترین فرم رمزنگاری زمانی است که یک بلوک متن برای ایجاد یک بلوک متن رمز شده رمزنگاری می‌شود. بلوک بعدی متن برای دادن بلوک بعدی متن رمزی رمزنگاری می‌شود و به همین ترتیب ادامه می‌یابد. این به‌عنوان حالت کتاب کد الکترونیکی

(ECB) شناخته می‌شود. تصویر ٤-٥ یک ارائه بصری از رمزنگاری ECB را نشان می‌دهد.



تصویر ٤-٥: رمزنگاری در حالت ECB

باوجود سادگی حالت 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 تحلیل‌گران رمزنگاری برای تشخیص و تعیین الگوها بین متن رمزی استفاده می‌شود. پس‌ازاینکه الگوها شناسایی شدند، می‌توان به طرز دقیقی استنباط کرد که رمزنگاری ECB استفاده‌شده است. بنابراین، یک مهاجم تنها باید روی رمزگشایی یک بلوک خاص متن تمرکز کند. و نیازی به رمزگشایی کل پیام ندارد.

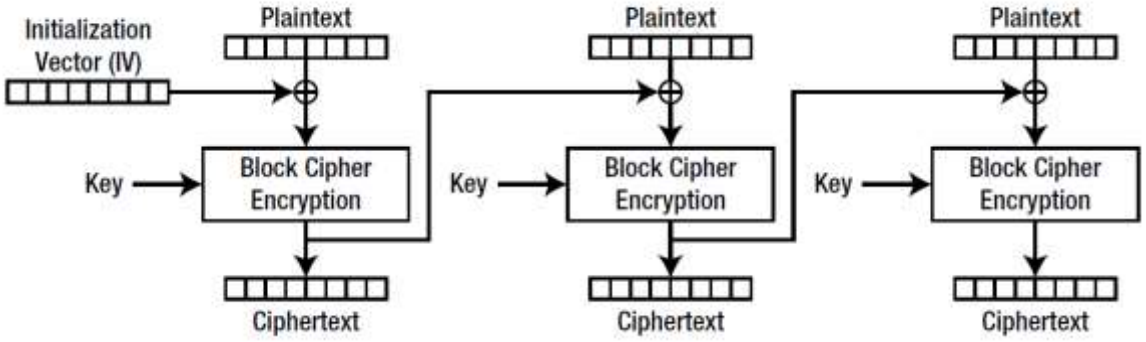
برای جلوگیری از این کار، تعدادی حالت مختلف عملیات برای رمزنگاری بلوکی وجود دارد:

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

تنها فرایند رمزنگاری را در این بخش شرح می‌دهیم (به‌سادگی می‌توان مراحل رمزگشایی را از معکوس فرایند رمزنگاری فهمید):

حالت CBC: حالت زنجیره رمزی- بلوک (یا صفر- بلوک) (تصویر ٤-٦ را مشاهده کنید) از یک مقدار اضافی به‌عنوان یک بردار اولیه (IV) استفاده می‌کند که برای انجام عملیات XOR روی بلوک اول متن استفاده می‌شود.پس‌ازاین، هر بلوک متن رمز شده با بلوک متن بعدی XOR می‌شود، و به همین ترتیب. این

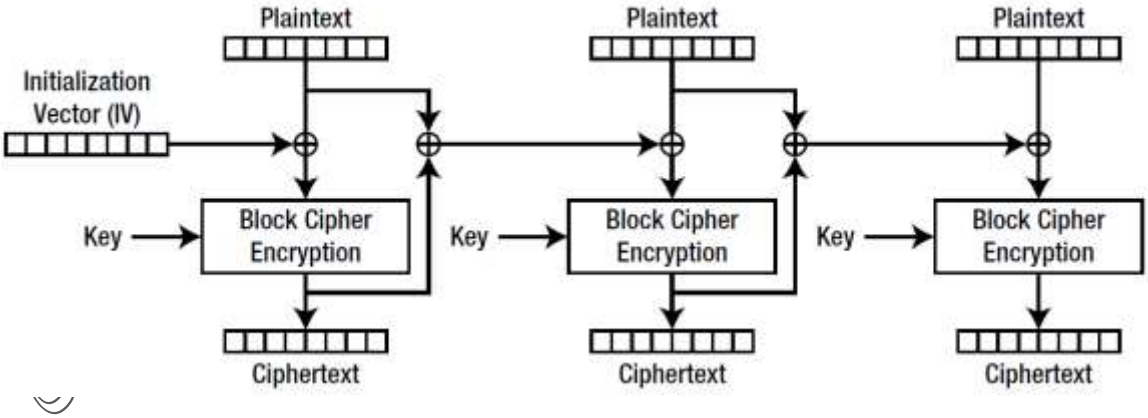
نوع حالت تضمین می‌کند که نتیجه هر بلوک متن رمز شده وابسته به بلوک متن واضح قبلی هست.



تصویر ٤-٦: رمزنگاری در حالت CBC

حالت PCBC

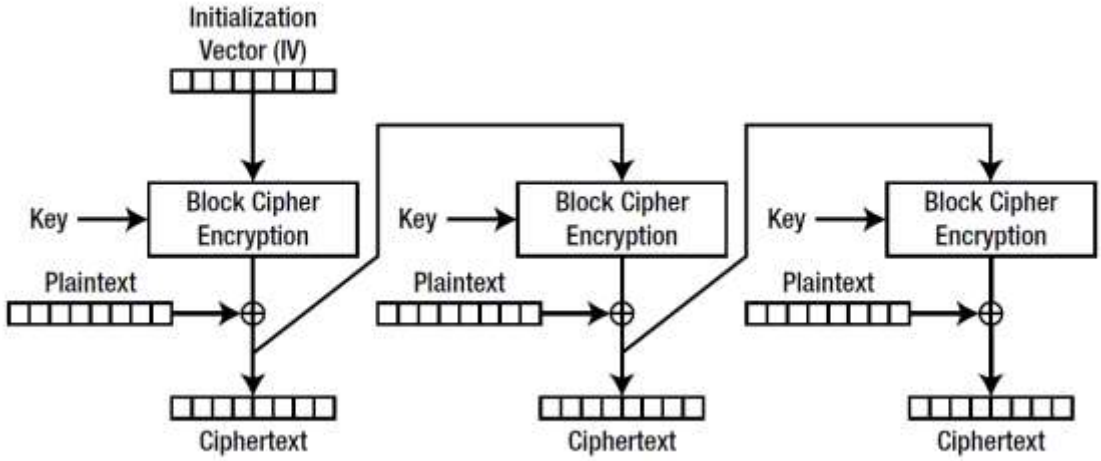
انتشار حالت زنجیره بلوک- صفر(تصویر ٤-٧ را مشاهده کنید) بسیار مشابه حالت CBC است. تفاوت در این است که به‌جای XOR کردن IV با اولین بلوک و XOR کردن متن رمز شده با بلوک‌های بعدی، در حالت PCBC مقدار IV و متن رمز شده برای بلوک اول XOR می‌شود و سپس نتیجه XOR متن واضح و متن رمز شده را با بلوک متن بعدی XOR می‌کند. طراحی این حالت به‌گونه‌ای است که یک تغییر کوچک در متن رمز شده از طریق فرایند رمزنگاری یا رمزگشایی انتشار می‌یابد.



تصویر ٤-٧: رمزنگاری در حالت PCBC

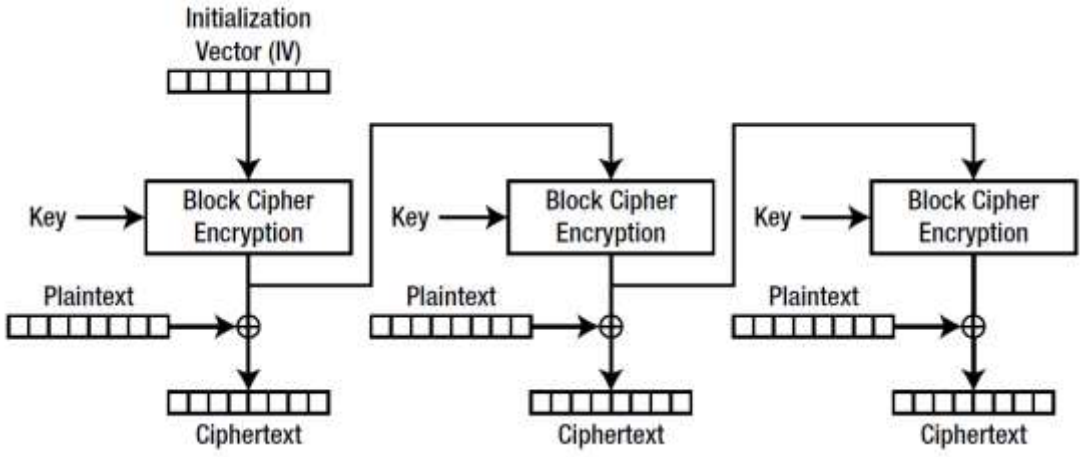
حالت CFB: حالت بازخورد رمزی (تصویر ٤-٨ را مشاهده کنید) مکان IV و متن واضح در حالت CBC را تغییر می‌دهد. یعنی به‌جای XOR کردن متن واضح و رمزنگاری آن، و سپس XOR کردن متن رمزی با متن واضح، در حالت CFB ابتدا IV را رمزنگاری می‌کند سپس برای دریافت متن رمز شده آن را با متن واضح

XOR می‌کند. سپس برای بلوک‌های بعدی، متن رمز شده دوباره رمزنگاری می‌شود و با متن واضح XOR می‌شود تا بلوک بعدی متن رمز شده را بدهد.



تصویر ٤-٨: رمزنگاری در حالت CFB

حالت OFB:حالت بازخورد خروجی (تصویر ٤-٩ را مشاهده کنید) بسیار مشابه حالت CFB هست. تفاوت این است که، به‌جای استفاده از XORشده‌ی IV رمز شده و متن واضح، از IV رمز شده استفاده می‌کند. بنابراین،برای بلوک اول، IV با کلید رمزنگاری می‌شود و این به‌عنوان ورودی برای بلوک بعدی استفاده می‌شود. متن رمز شده از بلوک اول سپس با اولین بلوک از متن واضح XOR می‌شود. رمزنگاری بعدی از IV رمز شده از بلوک قبلی قبل از XOR کردن استفاده می‌کند.



تصویر ٤-٩: رمزنگاری در حالت OFB

اگر به مثال اصلی دقت نمایید، مشاهده می‌کنید که از یک حالت خاص رمزنگاری استفاده نمی‌کنیم . اندروید برای رمزنگاری و رمزگشایی به‌طور پیش‌فرض از حالت ECB استفاده می‌کند. انتخاب یک حالت رمزنگاری

پیچیده‌تر مثل CBC یا CFB به خود شما به‌عنوان یک توسعه‌دهنده بستگی دارد. حالا که شما از عملکرد داخلی الگوریتم متقارن AES بیشتر آگاه هستید، چگونگی تغییر لایه گذاری حالت عملکرد را زمان رمزنگاری به شما نشان خواهیم داد. به مثال اصلی‌مان بازمی‌گردیم، برای خواندن فهرستی همانند کد ٤-٢، کد را تغییر دهید. به خطوط پررنگ کد توجه داشته باشید تنها چند تغییر انجام داده‌ایم. ابتدا، AES را به لایه گذاری AES/CBC/PKCS5 تغییر داده‌ایم. سپس، بردار اولیه (IV) را به متد init() اضافه کرده‌ایم. همانطور که قبلاً اشاره کردیم، زمانی که شما رمزنگاری AES را مشخص می‌کنید حالت پیش‌فرضی که اندروید استفاده خواهد کرد لایه گذاری AES/ECB/ PKCS5 است. شما می‌توانید با دو بار اجرای برنامه این کار را بررسی کنید یک‌بار با AES و یک‌بار با لایه گذاری AES/ECB/PKC5 هر دو متن رمزی 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 را بنویسید. در این کد، stringKey کلید موردنظر شما برای رمزنگاری اطلاعات هست.

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 ذخیره‌سازی داده‌ها موضوعاتی هستند که می‌توان برای تولید یک برنامه امن‌تر این دو را به یکدیگر ارتباط داد. اندروید برنامه‌ها را درزمینه‌ی 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 هستید. سؤال :"اگر اندروید در حال حاضر از اطلاعات محافظت می‌کند چرا باید تمام مباحث رمزنگاری را پوشش دهیم؟" همان‌طور که در آغاز این فصل اشاره کردیم فقط برای زمان‌های پیش‌بینی‌نشده به هنگام آسیب‌پذیری، ویروس یا زمانی که Trojan چهره زشت خود را نشان می‌دهد می‌توانیم یک‌لایه امنیتی دیگر بر روی لایه امنیتی اندروید ایجاد کنیم. 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 شما اجازه می‌دهد که انواع داده‌های اولیه را ذخیره کنید(برای مثال int, Boolean, float, long, and String) که باید در برابر دستگاه محافظت شوند حتی اگر برنامه در حال اجرا نباشد داده‌های شما حفظ خواهند شد مگر اینکه دستگاه دوباره شروع به کار کند.

حافظه داخلی	سه حالت شخصی‌سازی: MODE_PRIVATE, MODE_WORLD_READABLE, MODE_WORLD_WRITABLE.	به شما اجازه می‌دهد که داده‌هایتان را در حافظه داخلی دستگاه ذخیره نمایید به‌طورکلی این داده‌ها توسط دیگر برنامه‌ها یا حتی کاربران نهایی قابل‌دستیابی نیستند. یک مکان ذخیره‌سازی شخصی است. داده‌های ذخیره‌شده در اینجا حتی بعد از شروع به کار کردن دستگاه نیز محافظت می‌شوند. زمانی که کاربر نهایی برنامه شما رو حذف می‌کند اندروید نیز داده‌های شمارا حذف خواهد کرد.
حافظه خارجی	حالت پیش‌فرض: MODE PRIVATE داده‌ها به‌طور پیش‌فرض توسط همه قابل خوانا هست.	داده‌هایی که اینجا ذخیره می‌شوند قابل خوانا توسط همه می‌باشند. کاربر دستگاه و دیگر برنامه‌ها می‌توانند این داده‌ها را بخوانند اصلاح کنند و حذف کنند. حافظه خارجی باکارت SD یا دستگاه حافظه داخلی (که غیرقابل‌حذف است) 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 هستید از مکانیسم 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 داده ساختار غیرضروری در امان می‌دارد. بیایید به مثال‌ها نگاهی بیندازیم که چگونه داده‌ها را با استفاده از هر روش ذخیره یا بازیابی کنیم.

٤-٤-١ تنظیمات مشترک

برای ذخیره تنظیمات برنامه‌ها که تا زمان راه‌اندازی مجدد دستگاه صورت می‌گیرد، «تنظیمات مشترک[1]» بسیار مفید است. همان‌طور که نام آن بیان می‌کند، این روش ذخیره‌سازی برای نگهداری تنظیمات کاربر برای 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 داده‌شده است. کلاس ذخیره‌سازی 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 net.zenconsult.android;
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.putInt("port", port);
        ed.putBoolean("ssl", useSSL);
        returned.commit();
    }
}
```

کد ٤-٥:کد بازیابی داده از SharedPreferences

```
package net.zenconsult.android;
```

[1] SharedPreferences

```
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 ۶-۲: StorageExample 1, Main Class

```
package net.zenconsult.android;

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 Activity {
    /**  Called when the activity is first created. */
    @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 کردیم،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 شما تنها می‌توانید انواع داده‌های اولیه را ذخیره کنید. شما نمی‌توانید انواع داده‌های پیچیده‌تر همچون بردارها یا جداول هش را ذخیره نمایید. اگر بخواهید این داده‌ها را به‌جای داده‌های اولیه ذخیره نمایید می‌توانید به حافظه داخلی توجه داشته باشید.

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. اجازه دهید که با ایجاد کلاس StorageExample2 (کد ٤-٧ را ببینید) شروع کنیم. همانند گذشته، ماژول‌های ذخیره و بازیابی در کدهای جداگانه نشان داده‌شده است (کد ٤-٨ و کد ٤-٩ را به ترتیب ببینید). تصویر ٤-١١ خروجی را نشان می‌دهد.

کد ٤-٧: StorageExample2, Main Class

```
package net.zenconsult.android;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.widget.EditText;
public class StorageExample2Activity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Context ctx = getApplicationContext();
        // Store data
        Contact contact = new Contact();
        contact.setFirstName("Sheran");
        contact.setLastName("Gunasekera");
        contact.setEmail("sheran@zenconsult.net");
        contact.setPhone(" + 12120031337");
        StoreData.storeData(contact.getBytes(), ctx);
        // Retrieve data
        EditText ed = (EditText) findViewById(R.id.editText1);
        ed.setText(new String(RetrieveData.get(ctx)));
    }
}
```

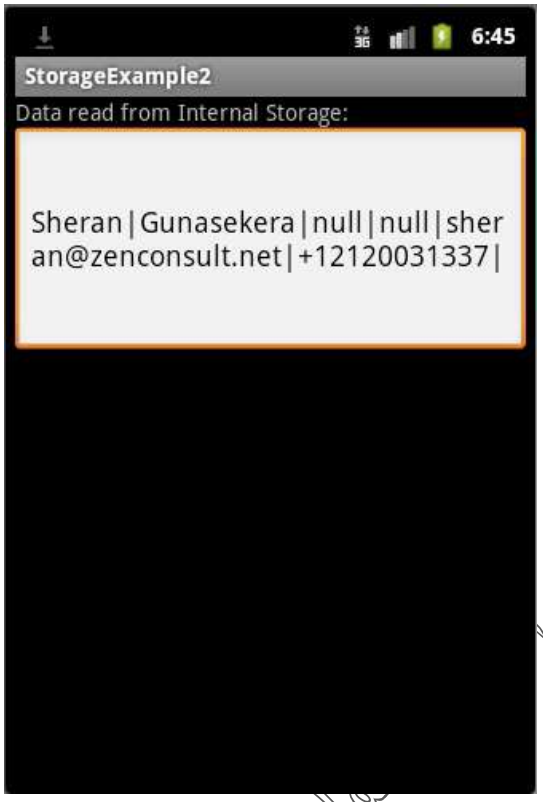
کد ٤-٨: استفاده از StoreData.java برای ذخیره داده در حافظه داخلی

```
package net.zenconsult.android;
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 net.zenconsult.android;
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;
    }
}
```



تصویر ٤-١١: خروجی برنامه StorageExample2

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

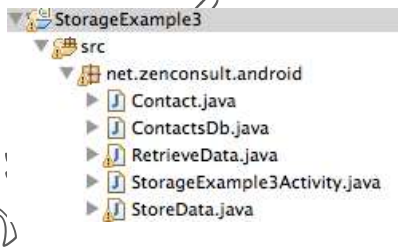
٤-٤-٣ پایگاه داده‌های SQLite

قصد داریم که از مثال ذخیره‌سازی خارجی عبور کنیم زیرا شما قبلاً با چگونگی ذخیره‌سازی داده‌ها به‌صورت خارجی آشنا شدید. در عوض، روی چگونگی ایجاد، ذخیره، و بازیابی داده‌ها با استفاده از پایگاه داده SQLite اندروید متمرکز می‌شویم. یک پایگاه داده ایجاد خواهیم کرد که می‌توانیم برای ذخیره شیء Contact در برنامه Proxim استفاده کنیم. جدول ٤-٤ طرح‌بندی جدول را نشان می‌دهد. روشی اسناد را در پیش‌گرفته‌ایم و 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 نشان 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 استفاده کنید و کد ٤-١٢چگونگی استفاده از این کلاس کمکی برای نوشتن داده‌ها از شیء Contact به پایگاه داده را نشان می‌دهد. درنهایت، 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، دید نزدیک‌تری به هر قسمت از کد داشته باشید تا درک کنید هر قسمت چه‌کاری انجام می‌دهد و چگونه انجام می‌دهد.

کد ٤-١٠:StorageExample3

```
package net.zenconsult.android;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.EditText;
public class StorageExample3Activity extends Activity {
    /** 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("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(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 net.zenconsult.android;
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 net.zenconsult.android;
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 net.zenconsult.android;
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;
    }
}
```

با توجه به تجربه ، بسیار نادر است که از یک فایل مسطح برای ذخیره داده‌ها استفاده کنیم. مگر اینکه با داده‌های خالص دودویی کارکنیم (برای مثال عکس،ویدیو یا موسیقی) بیشتر داده‌هایی که ذخیره می‌کنیم یا به‌عنوان جفت key-value استفاده می‌شوند یا در یک پایگاه داده SQLite ذخیره می‌شوند. بنابراین، می‌توانیم از SharedPreference اندروید یا SQLiteDatsbase برای انجام این کار استفاده کنیم. هر دو روش قابلیت‌های مدیریتی خوبی را ارائه می‌دهند. اگر تاکنون با پایگاه داده‌های SQLite کارنکرده‌اید می‌توانید کمی بیشتر آن را بررسی کنید. در عوض،بیشتر سیستم‌عامل‌های موبایل همچون ios اپل و RIM برای

BlackBerrySmartphoneOS از پایگاه داده‌های SQLite پشتیبانی می‌کنند. نکته مثبت این است 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.

اجازه دهید که کد پروژه StorageExample3 تحلیل کنیم. کد ٤-١٠ کلاس اصلی است و یک شیء Contact را با داده‌های درون آن ایجاد می‌کند.

```
Contact contact = new Contact();
contact.setFirstName("Sheran");
contact.setLastName("Gunasekera");
contact.setEmail("sheran@zenconsult.net");
contact.setPhone(" + 12120031337");
```

سپس 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 کنید. آنگاه این کد از متدهای ()store (کد ٤-١٣) برای حفظ شیء ایجادشده Contact استفاده می‌کند.متد ()store را فراخوانی می‌کنیم و پایگاه داده SQLite جدیدمان و شیء Contact را به آن می‌دهیم. StoreData شیء Contact را به اشیاء 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 شما در حال ایجاد اشیاء داده خود هستید و می‌دانید که قصد دارید از روش پایگاه داده برای ذخیره داده‌ها استفاده کنید شما باید ContentValueها برای اشیاء داده خود را در نظر بگیرید زیرا کار با داده‌ها در زمان ذخیره و بازیابی آن‌ها را بسیار راحت‌تر می‌سازد.

سپس مقادیر را در جداول پایگاه داده خود می‌نویسیم. شیء SQLiteOpenHelper می‌تواند یک WritableDatabase یا ReadableDatabase را بازیابی کند بنابراین باید از مناسب‌ترین 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

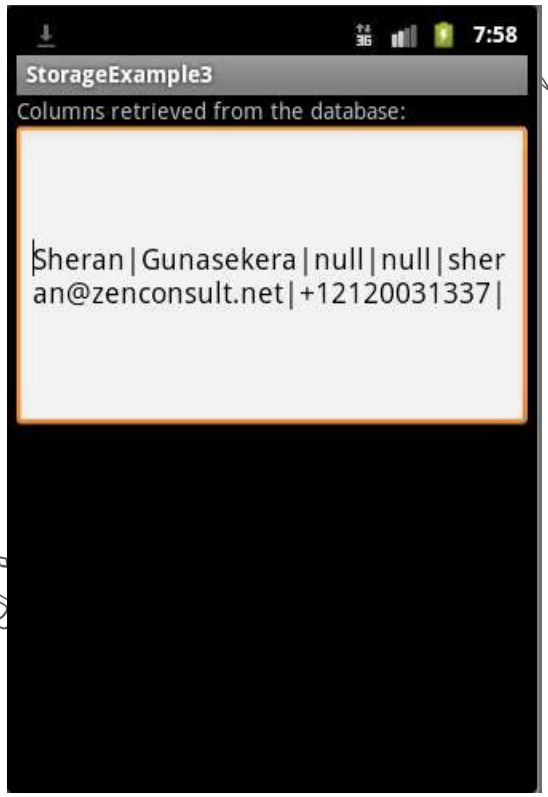
داده‌شده توجه داریم. در یک برنامه برای واکشی هر ردیف اطلاعات باید Cursor خود را تکرار کنیم.

```
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)



تصویر ٤-١٣: خروجی برنامه StorageExample3

٤-٥ ترکیب ذخیره‌سازی داده‌ها با رمزنگاری

دو نکته بسیار مهم را به‌طور جداگانه در این فصل پوشش دادیم. می‌توانیم به‌طور واضح مشاهده کنیم که هر آنچه داده ذخیره کنیم درون حافظه‌ای که انتخاب کرده‌ایم قرار می‌گیرد. برای ضمانت این‌که داده‌های توسط برنامه‌های غیرمجاز خوانده نمی‌شود می‌توانیم به اندروید اعتماد کنیم اما اگر یک ویروس هفته بعد منتشر شود چه‌کاری باید انجام دهیم؟ 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 حفظ اطلاعات شخصی خود به خطر افتاده است و همه داده‌ها می‌تواند از دستگاه شما کپی شود.

چنین حملاتی را در فصل قبل بررسی کردیم و آن‌ها را به‌عنوان حملات غیرمستقیم دسته‌بندی کردیم. این ویروس غیرمستقیم است زیرا به‌طور مستقیم با برنامه شما کار نمی‌کند بلکه هدف آن سیستم‌عامل اندروید است. هدف کپی همه پایگاه داده‌های SQLite است به این امید که حمله‌کننده بتواند هر اطلاعات حساسی که آنجا ذخیره‌شده است را کپی کند. اگر شما یک‌لایه محافظت دیگر را اضافه کنید؛ پس‌ازآن، حمله‌کننده اطلاعات را به‌صورت 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 الگوریتم‌های متقارن: کتابخانه ما از یک الگوریتم متقارن یا رمزنگاری بلوکی، برای رمزنگاری و رمزگشایی داده‌هایمان استفاده خواهد کرد. اگرچه باید بعداً بتوانیم قادر به تغییرات در الگوریتم انتخابی خود باشیم در AES پایدار خواهیم ماند.
- استفاده از یک کلید ثابت: باید قادر به دربرگیری یک کلید که می‌توانیم در دستگاه ذخیره کنیم باشیم که قادر خواهد بود داده‌ها را رمزنگاری و رمزگشایی کند.
- کلید ذخیره‌شده روی دستگاه: کلید بر روی دستگاه قرار خواهد گرفت. درحالی‌که از منظر حملات مستقیم برای برنامه یک خطر محسوب می‌شود، آن در مقابله با حملات غیرمستقیم برای ما کافی است.

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

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

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

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

کد ٤-١٤: ماژول KeyManager

```
package net.zenconsult.android.crypto;

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 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 اضافه می‌کنیم تا کار تبدیل داده‌های بایتی به داده‌های قابل‌مشاهده در مبنای ٦٤ و برعکس را راحت‌تر کند.

کد ٤-١٥: ماژول Cryptographic

```
package net.zenconsult.android.crypto;
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)));
    }
}
```

شما می‌توانید این دو فایل را در هریک از برنامه‌ها که نیازمند ذخیره‌سازی داده برای رمزنگاری می‌باشند جای دهید. ابتدا اطمینان یابید که شما یک مقدار برای کلید خود و بردار اولیه دارید سپس قبل از ذخیره داده خود متد رمزنگاری یا رمزگشایی را روی آن‌ها فراخوانی کنید. کد ۱۶-۴ تغییرات موردنیاز برای کلاس StorageExample3 را نشان می‌دهد. به‌علاوه، کد ۱۷-۴ و کد ۱۸-۴ تغییرات موردنیاز برای فایلهای StoreData و RetrieveData را به ترتیب نشان می‌دهد.

کد ٤-١٦: StorageExample3 جدید با رمزنگاری

```
package net.zenconsult.android;
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 Activity {
    /** 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 net.zenconsult.android;
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 net.zenconsult.android;
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 را پوشش دادیم چگونگی PKI و شخص ثالث قابل اعتماد و همچنین ساختار PKI را با توجه به اهدافمان بررسی کردیم. سپس به سازوکار ساده رمزنگاری داده‌ها و اصطلاحات توجه کردیم. دیدیم که رمزنگاری به‌سادگی انتخاب یک الگوریتم متقارن نیست و باید به جنبه‌های مختلفی مثل لایه گذاری و حالت‌های عملیات دقت کرد. سپس سازوکارهای مختلف ذخیره‌سازی داده‌ها روی اندروید را بحث کردیم و نمونه‌های هر یک را پوشش دادیم و پایگاه داده‌های SQLite و SharedPreferences را به دلیل توانایی در ذخیره انواع داده‌های برنامه برگزیدیم. سپس بررسی کردیم که چگونه می‌توان داده‌هایمان را رمز کنیم و یک کتابخانه همه‌منظوره برای رمزنگاری و رمزگشایی ساختیم. این کتابخانه می‌تواند در هر یک از برنامه‌های آینده که نیاز به ذخیره‌سازی داده‌ها دارد به کار رود.

۵ برنامه‌های وب

از برخی نکته نظرات، شما باید با برنامهٔ تحت وب ارتباط برقرار کنید. خواه برنامه شما در حال ارتباط با یکRESTful API هست یا در حال تبادل اطلاعات با برنامهٔ تحت وب خود، برنامهٔ موبایل شما برای تعامل با دیگر برنامه‌های کاربردی نیاز دارد تا باز باشد. به‌طور طبیعی، به‌عنوان یک توسعه‌دهنده‌ی، 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. در فصل‌های قبل،ما بیشتر زمان را صرف بررسی "داده‌های ثابت" و داده‌های ذخیره‌شده و رمزنگاری‌شده کردیم. در این فصل، "داده‌های در حال انتقال" را پوشش می‌دهیم.

در اصل، قصد نداریم زمان زیادی را در مورد محاسن رمزنگاری داده‌های در حال انتقال بحث کنیم. معمولاً، SSLیا TLS امنیت داده‌های در حال انتقال را بر عهده‌دارند. اما امکان نفوذ به منبع گواهی‌ها یا CA در هلند که DigiNotar نامیده می‌شود باعث شده است 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 را به شما به‌عنوان توسعه‌دهنده واگذار می‌کنیم; اما به‌طور واضح، این حمل باعث شده که حتی گواهی SSL نیز همیشه بهترین گزینه نباشد. بدین ترتیب، برخی از موضوعات مربوط به امنیت برنامه‌نویسی تحت وب را پوشش خواهیم داد و اینکه چگونه نرم‌افزار موبایل شما باید با برنامه‌های کاربردی تحت وب تعامل برقرار کند. همچنین، به‌طور مختصر پروژه‌ی امنیتیِ برنامه تحت وب‌باز را پوشش خواهیم داد. این‌یک منبع بسیار خوبی است که می‌توانید برنامهٔ تحت‌وب خود را امن کنید.

در نظر بگیرید که چگونه سورس کد ۵-۱ امن شده است. حالا از خود بپرسید که برای امن‌تر شدن کد چه باید کرد؟(در پایان فصل، نتایج را بررسی کنید و کد خودتان را مقایسه کنید و ببینید که راه درست را رفته‌اید یا نه.)

کد ۵-۱: ورود مشتری

```
package net.zenconsult.android.examples;
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";
    public Login() {
    }
    public HttpResponse execute() {
        HttpClient client = new DefaultHttpClient();
        HttpPost post = new
HttpPost("http://logindemo1.appspot.com/logindemo");
        HttpResponse response = null;
        // Post data with number of parameters
        List < NameValuePair > nvPairs = new ArrayList < NameValuePair >
(2);
        nvPairs.add(new BasicNameValuePair("username", "sheral"));
        nvPairs.add(new BasicNameValuePair("password", "s3kr3tc0dez"));
        // Add post data to http post
        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

اجازه دهید با راه‌اندازی محیط آزمونمان شروع کنیم. بدیهی است که ما به یک زیرساخت برنامه کاربردی نیاز داریم که تحت وب ساخته‌شده باشد.

ما هنوز قصد نوشتن کد back-end را نداریم. اول اجازه دهید یک برنامهٔ کاربردی کوچک بنویسیم که می‌توانیم آن را بسازیم و با آن شروع کنیم. یک پروژه با نام LogiDemo و پکیجی با عنوان net.zenconsult.gapps.logindemo ایجاد کنید درون پکیج نام‌گذاری شده یک فایل با نام LoginDemoServlet بسازید که فایل حاوی کدهای نشان داده‌شده 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.

کد ۵-۲: پکیج برنامهٔ کاربردی کوچک به‌صورت پیش‌فرض، net.zenconsult.gapps.logindemo

```
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 می‌توانید این کد را به‌صورت local در کامپیوتر خود اجرا و مشاهده کنید. زمانی که به آدرس http://localhost:8080/logindemo می‌رویم، پیام پاسخ "Hello, World" را می‌بینیم (تصویر ۵-۱ را ببینید).

تصویر ۵-۱: دسترسی به logindemoServlet

ما در حال حاضر یک برنامهٔ کاربردی وب‌داریم که می‌توانیم آن را برای هر آنچه بخواهیم استفاده کنیم. اجازه دهید یک مقدار از تئوری‌های مربوط به برنامهٔ کاربردی وب را بررسی کنیم.

۵-۲ HTML، برنامهٔ کاربردی وب و خدمات وب

هر توسعه‌دهنده‌ی وب می‌داند که HTML چیست؟ HTML سازنده‌ی اصلی هر وب‌سایت مدرن است .زبان نشانه‌گذاری ابرمتن (html) به‌صورت طرحی تفصیلی در سال 1991 حیات خود را شروع کرد. این زبان بسیار ساده به‌عنوان اهرمی برای ایجاد صفحات وبِ پایه استفاده می‌شود و تا سال 2008 زمانی که طرحی برای نسخه‌ی HTML5 منتشر شد به‌سرعت رشد 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 است که در آن کاربران نهایی به بیش از یک شبکه دسترسی دارند–درست مثل صفحات وب. یک برنامهٔ وب، شامل عناصر پویای بیشتر نسبت به HTML ساده است.

برای مثال، برنامه‌های وب مدرن دارای تعداد زیادی زبان‌های سمت سرور هستند. این زبان‌ها(مانند: PHP ، ASP،JSP و ...) بر اساس ورودی کاربر نهایی یک صفحه html ایستا به‌صورت بلادرنگ تولید می‌کنند. برنامهٔ وب روی یک وب سرور نصب‌شده است و روی یک سخت‌افزار میزبانی می‌شود که یک 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 ذخیره‌سازی داده یا توابع بازیابی مراقبت می‌کند. ازآنجاکه تمام پردازش‌های پیچیده، در back end یا سمت سرور قرارگرفته‌اند یک مرورگر وب ساده چیزی بیشتر از یک تعامل با رابط کاربری نیست.

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

یک سرویس وب شبیه به برنامهٔ وب است که می‌توان به آن از طریق یک شبکه راه دور دسترسی داشت. همچنین شبیه به این است که برخی از انواع برنامه‌ی کاربردی سرور را اجرا کند. تفاوت اصلی در این است که کاربران دسترسی تعاملی مستقیم با سرویس ندارند. در اکثر موارد، سرویس‌های وب با clientها یا دیگر برنامه‌های کاربردی سرور در تعامل‌اند. 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 سرویس وب از فایل زبان توصیف سرویس وب (WSDL) استفاده می‌کند. برنامه‌های کاربردی دیگر با استفاده از فایل WSDL که منتشرشده است می‌توانند چگونگی کار با سرویس وب را درک کنند. به‌طورکلی، سرویس وب از فرمت xml خاص برای تبادل اطلاعات استفاده می‌کند. یکی از پروتکل‌های معروف، پروتکل دسترسی آسان به اشیاء (SOAP) است SOAP .از فایل‌های xml مختلف براساس 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
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 .REST یا epresentational State Transfer، 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 استفاده می‌کند)، از یک محیط بسیار ساده‌تر 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 در سایت http://en.wikipedia.org/wiki/Representational_state_transfer مطالعه کنید. اگرچه استفاده از وب‌سرویس RESTful ساده است، بازهم می‌تواند وظایفی مشابه وظایف SOAP انجام دهد. به مثال SOAP در کد ۵-۳ نگاه کنید. اگر سرویس وب بخواهد همانند این کد را به‌عنوان RESTful API به ما ارائه دهد، پس 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 ما مجبور باشیم از سرور خروجی xml درخواست کنیم، ما می‌توانیم پسوند xml را به URI خود اضافه کنیم و همین‌طور اگر خروجی JSON را بخواهیم باید پسوند json را اضافه کنیم، 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 کنیم. HTTP پروتکلی است که وب را گسترش و رشد داد. قبلاً ابرمتن به HTML ساده و قدیمی اشاره می‌کند، ولی در حال حاضر توانسته است تا xml (زبان نشانه‌گذاری توسعه‌پذیر) 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 از HTML یا xml دریافت می‌کند.

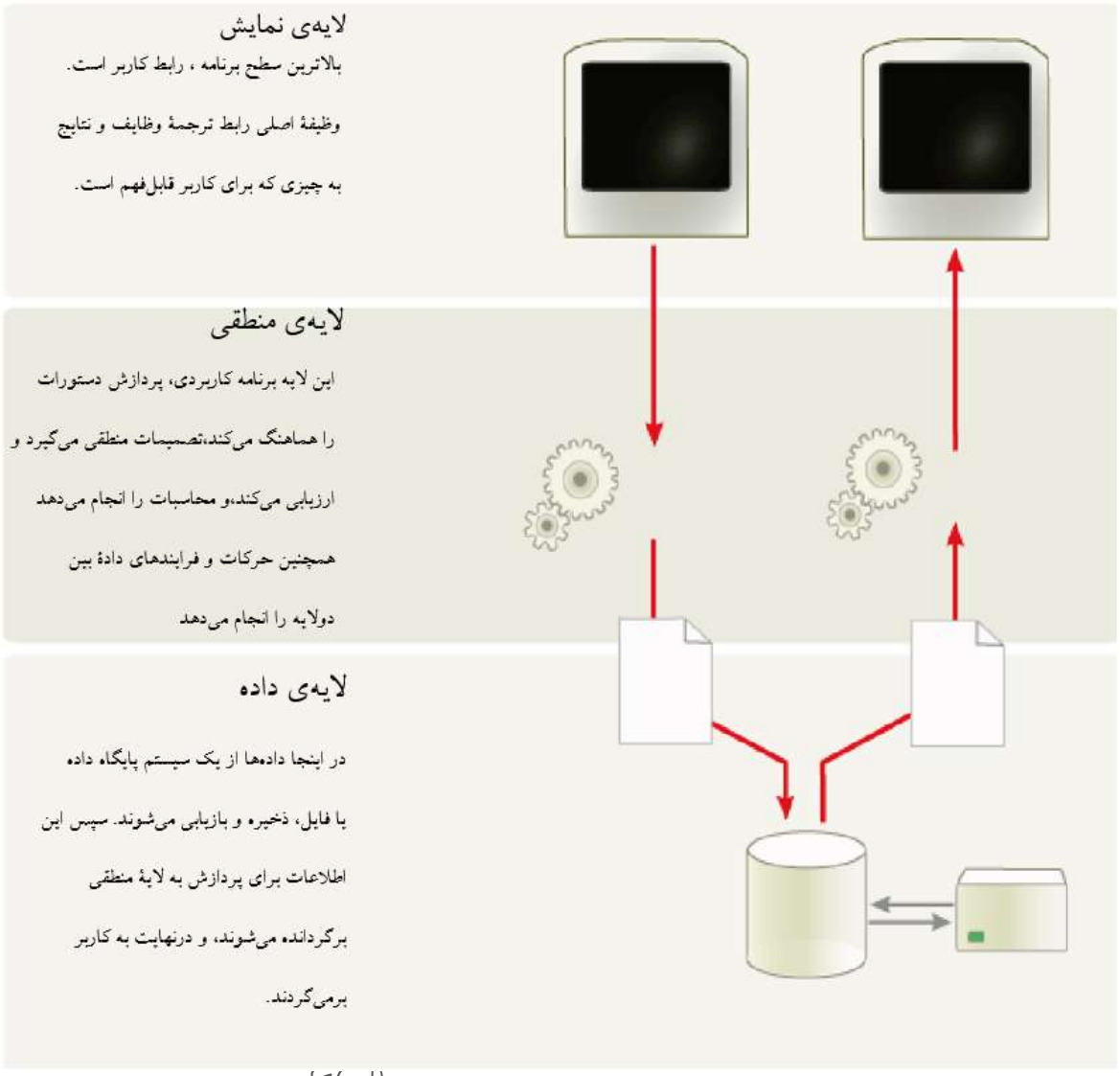
درخواست‌های HTTP انواع درخواست‌ها را شامل می‌شوند، درحالی‌که چندین روش درخواست وجود دارد، یکی از بهترین آن‌ها که استفاده می‌شود، GET و POST است. درخواست‌های GET برای بازیابی داده‌ها استفاده می‌شوند، و درخواست‌های POST برای ارائه‌ی داده‌ها استفاده می‌شوند. اگر شما در حال پر کردن فرم ثبت‌نام باشید، دکمهٔ ثبت را برای فعالیت مرورگر کلیک کنید تا داده‌ی شما به سرور فرستاده شود. اگر شما به کد ۵-۱ در آغاز فصل مراجعه کنید، شما خط زیر را خواهید دید:

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

این ایجاد یک درخواست HTTP POST به یک آدرس خاص است. همان‌طور که شما احتمالاً میدانید، یک URL یک نوع درخواست است که به عامل کاربر می‌گوید که در آن، یک منبع خاص را بازیابی می‌کند. 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. شما می‌توانید اطلاعات اضافه را در مورد HTTP در سایت www.w3.org پیدا کنید.

۵-۲-۱ اجزاء در یک برنامهٔ کاربردی وب

برنامه‌های کاربردی وب از لایه‌های مختلف تشکیل‌شده‌اند.برنامه‌های کاربردی وبِ معمولی سه لایه دارند(تصویر ۵-۲ را ببینید):لایه‌ی نمایش، لایه‌ی منطقی و لایه‌ی داده. بر اساس الزامات و پیچیدگی یک برنامه کاربردی، تعداد لایه‌ها می‌توانند افزایش یابند.



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 شرکت نیاز دارد تا مقدار ذخیره‌سازی 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 مختلف را دارد، و درنتیجه مجموعه‌ای متفاوت از الزامات و کنترل‌های امنیتی مرتبط دارند. برنامه‌های کاربردی چندلایه به صاحبانش اجازه می‌دهد تا کنترل مجزای بیشتر بر لایه‌های فردی نسبت به اشکالات داشته باشند، زیرا همهٔ این سه لایه بر روی یک سیستم قرار دارند.

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 برای 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

جلسه احرازِ هویتِ کاربر که یک مشتری با یک سرور می‌سازد، 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 درصورتی‌که 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]

این‌یک مثال ساده است و چگونگی فرایند درخواست‌ها و پاسخ‌ها را از بیرون به داخل و بالعکس نشان می‌دهد.

۲-۲-۵ فناوری برنامهٔ وب

فناوری‌های متعددی وجود دارند که شما می‌توانید برای هر لایه از برنامهٔ وب استفاده کنید. شما می‌توانید از وب سرورهای بسیاری، چارچوب برنامه، سرورِ برنامهٔ کاربردی، زبان‌های برنامه‌نویسی سمت سرور و سرورهای پایگاه داده را انتخاب کنید. معیارهای انتخاب شما به عواملی مانند برنامهٔ کاربردی موردنیاز،

بودجه، و قابلیت پشتیبانی از فناوری شما بستگی دارد.

ازآنجاکه توسعهٔ اندروید عمدتاً روی جاوا انجام می‌شود، تصمیم داریم بـرای برنامـهٔ کـاربردی وب از جـاوا بهره ببریم. به‌غیراز جاوا، شما می‌توانید از دیگر فناوری‌های سمت سرور استفاده کنید که برخی از آن‌هـا در زیر آورده شده:

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

به‌طور مشابه بسته به نیازتان شما می‌توانید از تعداد زیادی از پایگاه داده‌های محبوب برای برنامه خود بـرای لایهٔ داده استفاده کنید. پایگاه داده‌های رایگان و تجاری زیادی وجـود دارد.این‌یکـی از تصـمیمات عمـده‌ای است که شما یا معماری برنامهٔ شما باید ابتدا ایجاد کند. اینجا تعداد کمی از پایگاه داده‌های محبـوب وجـود دارد که شما می‌توانید بیشتر در مورد آن یادبگیرید:

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

الآن اجازه دهید به کامل کردن برنامهٔ وبمان بپردازیم، این برنامه از فرایند بررسی رمز عبورِ ابتدایی پشـتیبانی می‌کند. توجه داشته باشید که مثال‌ها بسیار ساده‌اند. روال احراز هویت برای برنامه‌های وب واقعی، پیچیده‌تر خواهد بود. کد ٥-٤ را بررسی کنید.

کد ٥-٤: کد تائید گواهی جدید

```
package net.zenconsult.gapps.logindemo;
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!!");
        }
    }
}
```

گام بعدی کدهای خود را همان‌گونه که گفته شد در localhost اجرا کنید. حال یک پروژه‌ی جدید اندروید که احراز هویت را کنترل می‌کند ایجاد کنید(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 شامل کدLogin، LoginDemoClient1Activity ، strings.xml و فایل‌های main.xml است. همچنین شما برای اتصال به وب سرورتان نیاز به دسترسی به اینترنت دارید به همین منظور از اضافه شدن کد زیر به فایل AndroidManifest.xml اطمینان حاصل کنید:

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

تصویر ۵-۳: ساختار پروژه

کد ۵-۵: کلاس Login

```
package net.zenconsult.android.examples;
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 کلاسLogin ، دو پارامتر می‌گیرد که شامل username و passwordهستند. متد execute() از این پارامترها برای ساختن یک درخواست HTTP POST برای کاربر استفاده می‌کند.

کد ۵-۶: کلاس LoginDemoClient1Activity

```
package net.zenconsult.android.examples;
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 Activity 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 استاندارد است. این را می‌توان به‌عنوان ورودی برنامه یـا نقطـه شـروع در نظـر 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>
```

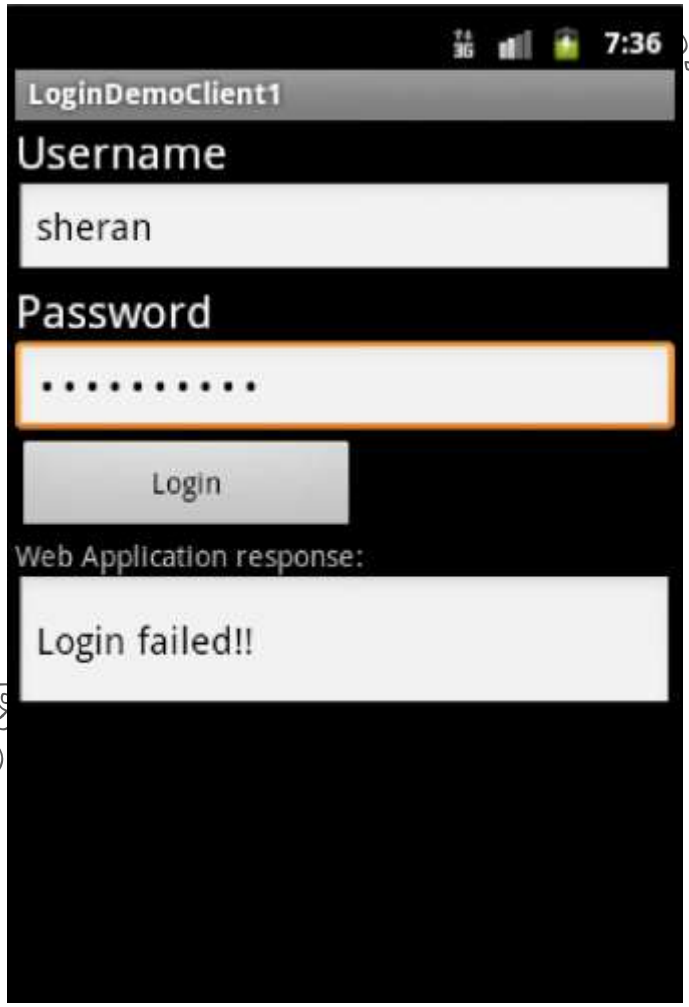
کد ۵-۸: فایل main.xml

```
<?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 تعریف‌شده و لایهٔ گرافیکی از برنامه می‌باشند.

برنامهٔ خود را اجرا کنید و نام کاربری و رمز عبورهای متعددی را وارد کنید. شما باید دو پیام پاسخ مجزا را ببینید، یکی برای موفقیت و یکی برای عدم موفقیت رمز عبور(تصویر ۵-٤را ببینید). شما هردو برنامه‌ی کلاینت و سرور login را نوشتید.



تصویر ۵-٤: عدم موفقیت لاگین

۵-۳ OWASP و حملات وب

OWASP سازمانی است که دانش، فن‌ها و دستورالعمل‌هایی را برای آزمون و تأمین امنیت برنامه‌های

کاربردیِ وب و سایر بسترها فراهم می‌کند. Owasp در دسامبر سال 2001 تأسیس شد. "بهبود و ارتقا امنیت نرم‌افزارها" به‌عنوان هدف اصلی در فهرست آن‌ها ثبت‌شده است. یک منبع بزرگ برای یادگیری است و همچنین روش‌های امن سازی برنامهٔ کاربردی شمارا ارائه می‌دهد.

از سال ۲۰۰٤ 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 وب در سطح جهان فهرست شده‌اند. لیست ده خطر برتر (top ten list) توسط تعداد زیادی از سازمان‌های تجاری استفاده و تصویب‌شده است، و به یک استاندارد برای امنیت برنامهٔ کاربردیِ وب تبدیل‌شده است.

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:

- 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

یک لیست از راهنمایی‌های عملی در وب‌سایت است که به شما به‌عنوان توسعه‌دهندهٔ وب کمک بسیار زیادی می‌کند. یکی از جدیدترین پروژه‌هایOWASP ، 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 به شما 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، به‌نوعی وابسته به ترکیب نام کاربری و رمز عبور یا پین است. نام کاربری و رمز عبور بر روی سرور ذخیره‌شده‌اند، و هر زمان برنامه کاربردی بخواهد یک کاربر نهایی را احراز هویت کند، یک مقایسه انجام خواهد داد. اگر شما 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", "sheran"));
nvPairs.add(new BasicNameValuePair("password", "s3kretc0dez"));
```

در این مثال، اطلاعات به‌صورت 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 استفاده‌شده است ، اما به نظر نمی‌رسد که ما آن را در مثالمان استفاده کرده باشیم، چون درخواستPOST مان به یک پورت به‌غیراز SSL/TLS می‌رود:

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

اجازه دهید به‌طورجدی در نظر بگیریم که ترافیکSSL مان نقض شده است. مهاجم که در حال استراق سمع روی گفت‌وگوی ما با برنامهٔ کاربردی وب است، حالا می‌تواند به اطلاعات کاربری ما 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 ما داشته باشد.

تاکنون، ما خطراتی که هنگام تصدیق هویتِ راه دور با آن مواجه می‌شویم را میدانیم. هرچند ممکن است اطلاعات ما از یک کانال امن بگذرد، اما هنوز در معرض حملات قرار دارد. و حتماً نباید حملهٔ شدیدی مانند DigiNotar باشد، که یک مهاجم می‌تواند گواهی خودش را صادر کند. برای مثال، می‌تواند حمله SSL man-in-the-middle باشد.

شما نمی‌توانید همیشه به SSL اعتماد کنید. 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 به شما میگویند که شما در حال بازدید از یک سایت به‌صورت امن هستید. این لزوماً درست نیست، بااین‌حال، قصد داریم یک‌لحظه به برخی از مفاهیم SSL بپردازیم.

SSL (لایهٔ سوکت امن) یک پروتکل انتقال است که داده‌هایی که بین دو کامپیوتر 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 به SSL ترجیح می‌دهند. جدیدترین پروتکل TLS (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 آن را در فصل ۳ پوشش دادیم. معمولاً، کاربران گواهی سرور X.509 را به‌عنوان گواهی SSL می‌شناسند. این گواهی جزءِ بسیار مهم SSL است. تصویر ۵-۵ راه‌اندازی مرورگر یک تست SSL را نشان می‌دهد.

تصویر ۵-۵: راه‌اندازی نشست SSL/TLS

TLSو SSL ترکیبی از فن‌های رمزنگاری را برای اطمینان از انتقال داده‌های امن استفاده می‌کنند. اکنون اجازه دهید راه‌اندازی نشست را ببینیم. نیازی طرح جزییات نیست ، چون شما نمی‌خواهید که الگوریتم مذاکره TLS را خودتان بنویسید. در عوض، در این بخش با ایده‌هایی از چگونگی رمزنگاری در طول نشست TLS آشنا می‌شویم.

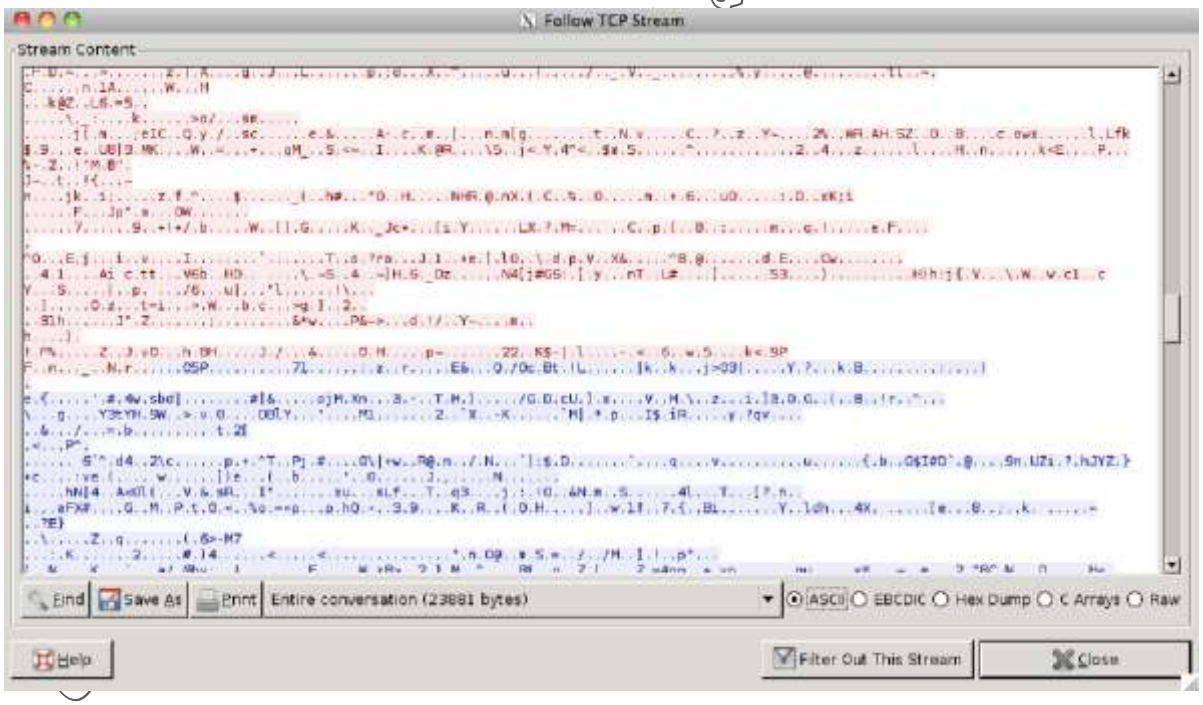
اول، کلاینت با یک وب سرور تماس می‌گیرد و بعضی اطلاعات را به آن می‌فرستد. اطلاعات شامل جزئیاتی از نسخه TLS است که شامل یک فهرستی از الگوریتم‌های رمزنگاری است که توسط نسخه TLS استفاده‌شده در client پشتیبانی می‌شود. این‌ها مجموعه رمز نامیده می‌شوند، و آن‌ها شامل الگوریتم‌های پشتیبانی شده برای کارهای مختلفی مانند تعویض کلید، احراز هویت و رمزهای انبوه است.

بعد، سرور پس از انتخاب مجموعه رمز خاص که پشتیبانی می‌کند و بالاترین نسخهٔ رایج TLS که هم کلاینت و هم سرور پشتیبانی می‌کنند، پاسخ می‌دهد. سپس سرور گواهیSSL اش را به کلاینت می‌فرستد. سپس کلاینت، از کلید عمومی سرور استفاده می‌کند تا یک کلید موقت را رمزنگاری و تبادل کند، کلیدی که

یک کلید اصلی را تولید می‌کند.

هنگامی‌که کلید موقت مبادله می‌شود، کلاینت و سرور از مقادیر تصادفی و کلید موقت برای تولید یک کلید اصلیِ نهایی استفاده می‌کنند. این کلید اصلی روی کلاینت و سرور ذخیره می‌شوند.

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 ثبت یک نشست رمزنگاری‌شده 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 معنی‌ای 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 کمی نگاه کنیم و بعداً ما برای جواب شما برمی‌گردیم.



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
THRkLjEWMBQGA1UEAxMNVGhhdGJRlIFNHQyBDQTAeFw0wOTEyMTgwMDAwMDBaFw0x
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 درباره امنیت است و X.509 درباره اعتماد است. یک گواهی SSL بر اساس معیارهای خاص برای یک فرد یا شرکت صادر می‌شود. CA به‌عنوان مرکز اعتبارسنجی مسئول صدور گواهی است و مسئول احراز هویت مشتری است. برای مثال، شما نمی‌توانید یک گواهی www.google.com را درخواست دهید بدون اینکه ثابت کنید شما به نحوی به آن وابسته هستید یا به‌عنوان نماینده شرکت گوگل هستید. این مهم است زیرا، اگر CA این مجوزها را بررسی نکند، بنابراین هرکسی می‌تواند از مجوز SSL دیگران استفاده کند و بر روی وب سرور خود نصب کند.

با فریب دادن کاربر نهایی شما، به باور به اینکه سرور شما یک سرورِ google.com است، شما می‌توانید یک حمله مردمیانی (MitM) را انجام دهید و همه داده‌های کاربر را ره‌گیری کنید. ما به‌طور خلاصه به حملهٔ MitM نگاه می‌کنیم; اما اول، می‌خواهیم موضوع دیگری را که ممکن است شما در مورد آن بدانید پوشش دهیم، و آن گواهی خود امضاشده است.

نکته: یک CA برای مشتریان گواهی SSL صادر کرده است. تا زمانی که گواهی صادر می‌شود، CA نیز همچنین گواهی SSL را با گواهی اصلی خودش امضا می‌کند. این امضا نشان‌دهنده‌ی این است که CA گواهی SSL صادرشده 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 به‌مرور گر شما اجازه می‌دهد تا گواهی‌های SSL صادرشده را به‌وسیلهٔCA های مختلف بررسی کند.

برای مثال، فرض کنید شما از VeriSign برای یک گواهی برای دامنه‌تان 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 کرده. پس‌ازاینکه شما گواهی SSLتان را دریافت کردید، آن را روی وب سرورتان نصب می‌کنید و وب‌سایتتان را راه‌اندازی می‌کنید. حالا، زمانی که وب‌سایت شمارا بازدید می‌کنیم، مرورگر ما اول گواهی SSL شمارا بررسی می‌کند، و سپس تلاش می‌کند تا بررسی کند که آیا گواهی شما به‌درستی توسط یک CA معتمد صادرشده یا نه. برای انجام این کار، مرورگر ما به انبار گواهی‌های اصلیِ اعتماد شدهٔ داخلی‌اش نگاه می‌کند تا تعیین کند که آیا امضای گواهی اصلی VeriSign با امضای روی گواهی شما مطابقت دارد یا نه.اگر مطابق باشد، بنابراین می‌توانم به دیدن سایت شما ادامه دهم. بااین‌حال، اگر مشکلی در بررسی گواهی شما وجود داشته باشد، مرورگر به ما هشدار می‌دهد که قادر به تائید گواهی نیست.

توجه داشته باشید که مرورگر شما قبل از چراغ سبز دادن، تعدادی از جزئیاتِ دیگرِ گواهی را بررسی خواهد کرد.

۵-٤-۱ گواهی خود امضاشده

در زمان توسعه و آزمایش پروژه‌ها، توسعه‌دهندگان گاهی اوقات از یک گواهی خود امضاشده روی وب‌سایتشان استفاده می‌کنند. این نوع از گواهی‌نامه در تمام جهات با گواهینامهٔ SSL صادرشده توسط CA یکسان 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 به سایتی متصل می‌شود، هیچ راهی برای تائید گواهی امضاشده وجود ندارد. چراکه گواهی

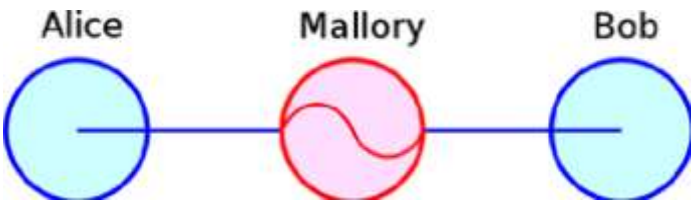
شخصی که آن را امضا کرده در انبار گواهی‌های معتمد داخلی مرورگر، وجود ندارد. بنابراین مرورگر با یک خطایی شبیه به آنچه در تصویر ۵-۸ نشان داده‌شده است به کاربر هشدار می‌دهد.

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

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

۵-٤-۲ حملهٔ مردمیانی (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 خودش در موقعیتی قرار می‌گیرد تا قادر به ره‌گیری ترافیک از دو فرستنده و گیرنده باشد و به‌طور مؤثر خودش را در بین این دو قرار می‌دهد(تصویر ۵-۹را ببینید). 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، حملاتی که ما می‌بینیم روی گواهی‌های خود امضا یا فریب مرورگرها با گواهیِ مهاجم به‌ظاهر معتبر متمرکزشده است. تا قبل از حمله DigiNour ، مهاجمان اطلاعات بسیار کمی در مورد امنیت CA به دست می‌آوردند، و حوادثِ 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 بانام 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 می‌دزدد.

چرا این اتفاق افتاد؟ DigiNotar کنترل‌های امنیتی بسیار سهل‌انگارانه‌ای را در زیر ساختارش دارد. بنابراین مهاجم قادر به ارتباط راه دور با سرورهای آن‌ها بود و به بسیاری از سیستم‌هایی که مسئول صدور گواهی قانونی آن‌ها هستند دسترسی پیدا کرد. بعدازاین، یک کار نسبتاً ساده برای مهاجم است که گواهی‌های صادرشده را برای خودش برای هر زمان که بخواهد نگه دارد. 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 و ...)

*.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 ازدست‌داده است.

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، تمایل داریم تا یک لایهٔ رمزنگاری داده‌ها بین برنامه‌ی موبایل و سرور را اضافه کنیم. پس، اگر لایهٔ SSLبه 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 برنامهٔ کاربردی شمارا رها کند.

راهی وجود دارد که از جاسوسی مهاجمان روی گواهی‌های ssl جلوگیری کند؟ درواقع بله! بیایید به دو راه نگاه کنیم که حتی زمانی که کانال‌های انتقال امنِ ما شکست‌خورده باشند ما می‌توانیم از به خطر افتادن گواهی‌هایمان جلوگیری کنیم. اولین آن OAuth و Challenge/Response دومین آن است.

۳-٤-۵ OAuth

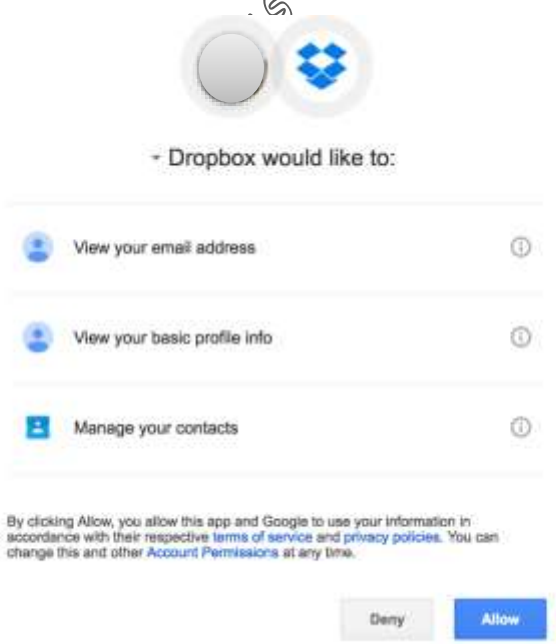
پروتکل OAuth به سایر سایت‌ها یا برنامه‌های کاربردی اجازه می‌دهد تا به‌عنوان مصرف‌کننده از اطلاعات کاربر روی یک برنامه‌ی وب که یک ارائه‌دهنده‌ی سرویس نامیده می‌شود استفاده کند. کاربر نهایی می‌تواند کنترل کند تا چه مقدار دسترسی به این برنامه‌ها بدهد و این بدون نیاز به فاش و یا ذخیره‌ی اطلاعات برنامه‌ی وبِ موجود انجام می‌شود.

به‌عنوان‌مثال dropbox یک سرویس‌دهنده فضای ابری است که کاربران می‌توانند فایل‌های خود را در این فضای ابری ذخیره کنند. قبل از OAuth ، یک کاربر برای ورود به dropbox.com باید نام کاربری و رمز عبور را وارد می‌کرد تا بتواند به فایل‌های خود را در این سایت بارگذاری کند. مشکل این روش این است که در حال حاضر کاربر اطلاعات هویتی خود را در dropbox ذخیره یا استفاده کرده است و سطح افشای اطلاعات هویتی‌اش افزایش پیداکرده 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

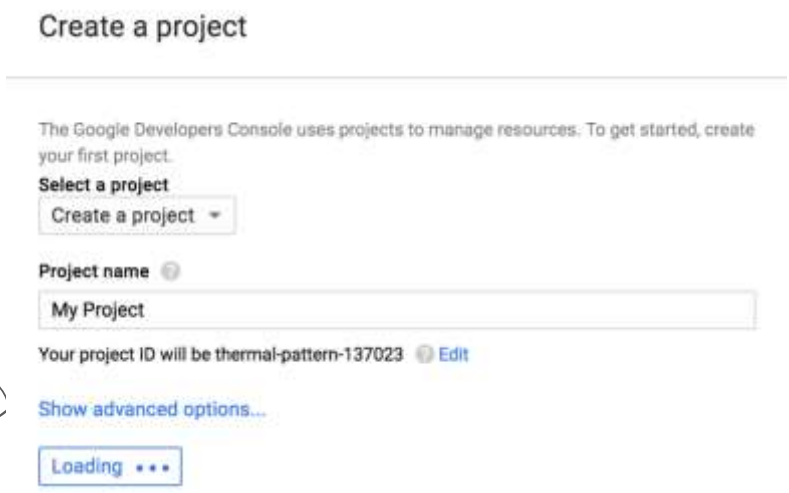


تصویر ۵-۱۱ dropbox برای استفاده از اطلاعات حساب کاربری گوگل درخواست مجوز می‌کند

OAuthبا استفاده از نشانه‌های درخواست[1] کار می‌کند. سایت‌هایی که می‌خواهند به داده‌ها در یک برنامهٔ وب دسترسی داشته باشند، قبل از اینکه آن‌ها بتوانند دسترسی به داده‌ها را شروع کنند نیاز هست که از چنین برنامه‌هایی یک نشانه داشته باشند.

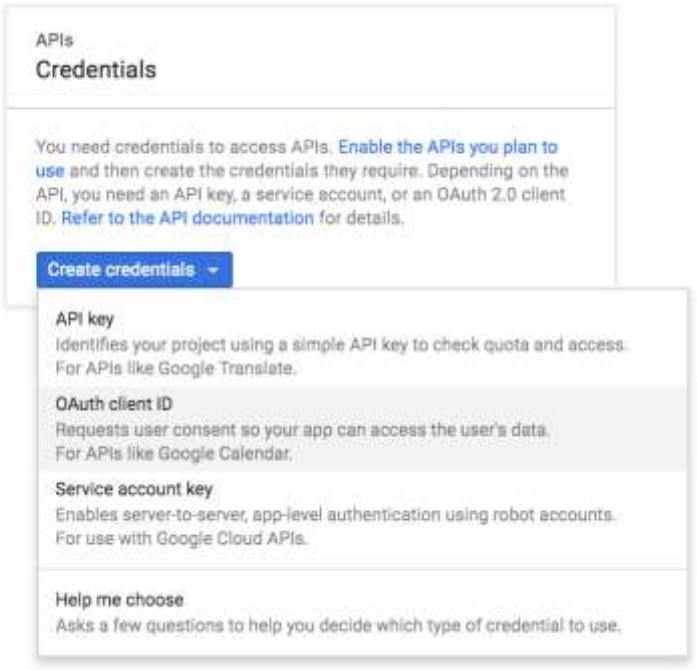
ابتدا اجازه دهید به چگونگی کار کردن OAuth برای dropbox نگاه کنیم. به‌عنوان‌مثال، فرض کنید که شما یک برنامهٔ اندروید نوشته‌اید برای احراز هویت از اطلاعات حساب کاربری گوگل استفاده می‌کند. نرم‌افزار اندروید شما برای انجام این کار نیاز به دسترسی به اطلاعات حساب کاربری گوگل را دارد. در این مورد، بازیگران (برنامهٔ اندروید(مصرف‌کننده) ، گوگل (ارائه‌دهندهٔ سرویس) و کاربر نهایی هست.

OAuth نیاز دارد که شما اول برنامهٔ مصرف‌کننده‌تان را روی سایتی که شما در حال احراز هویت آن هستید ثبت کنید. این کار لازم است، چون شما یک شناسهٔ برنامه دریافت خواهید کرد که شما برای استفاده در کدتان نیاز خواهید داشت. برای ثبت برنامه‌تان، شما باید سایت https://console.developers.google.com/apis/library را ببینید(تصویر ۵-۱۲را ببینید) کتابخانه OAuth 2.0 credentials را جستجو کنید یک پروژه ایجاد کنید، و یک ID مشتری OAuth ایجاد کنید(تصویر ۵-۱۳، تصویر ۵-۱٤، تصویر ۵-۱۵ و تصویر ۵-۱٦ را ببینید).

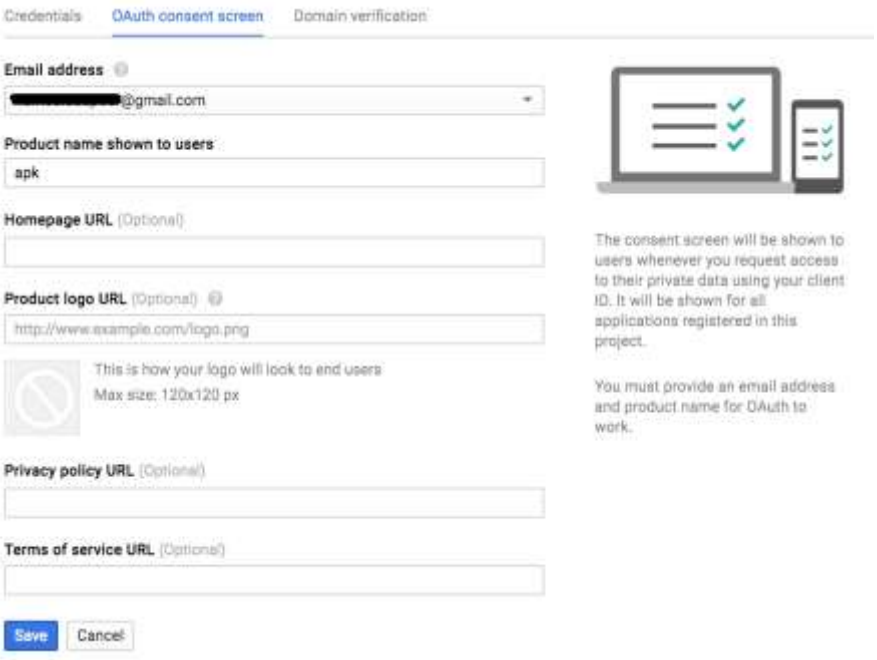


تصویر ۵-۱۲: ایجاد یک پروژه جدید روی گوگل

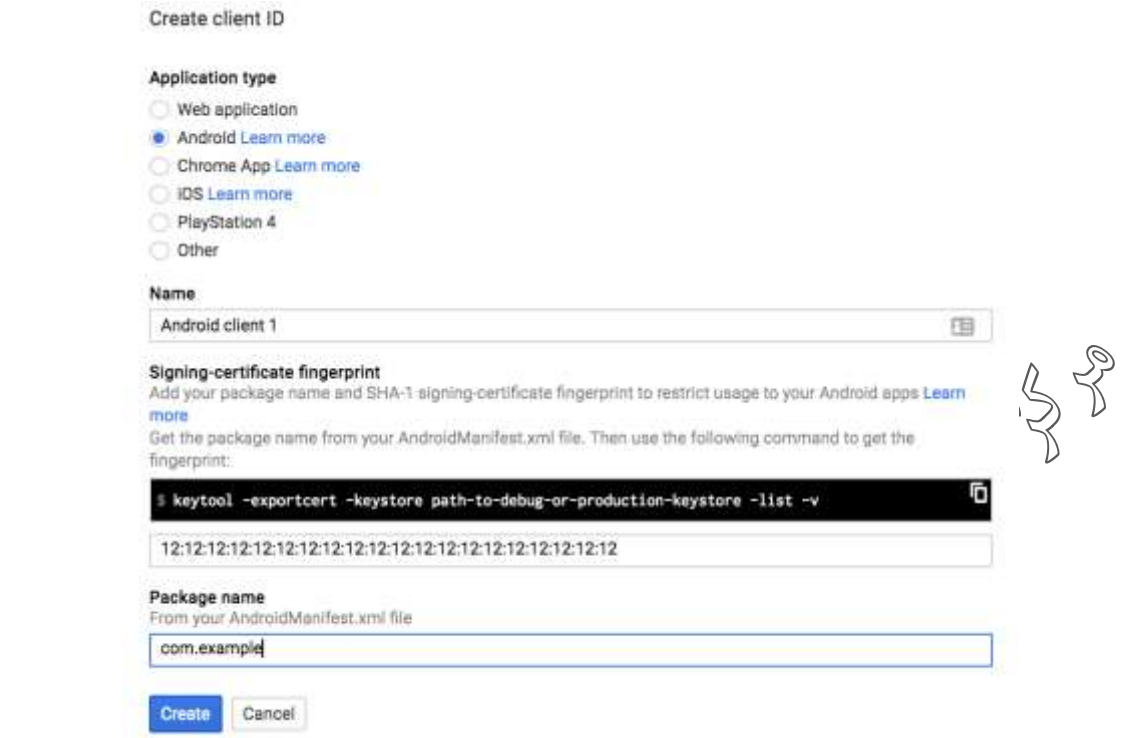
[1] Request token



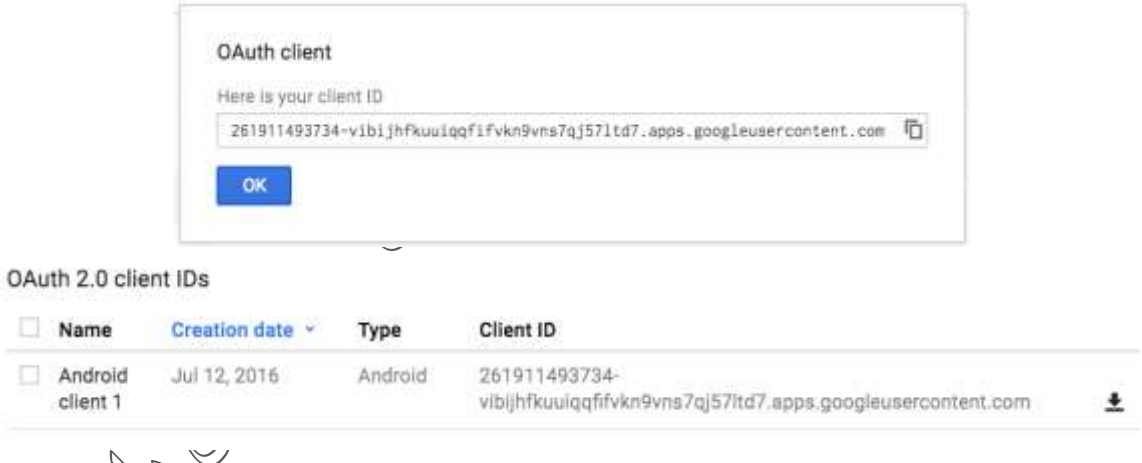
تصویر ۵-۱۳: ایجاد یک ID مشتری جدید



تصویر ۵-۱۴: تکمیل جزئیات صفحه درخواست OAuth

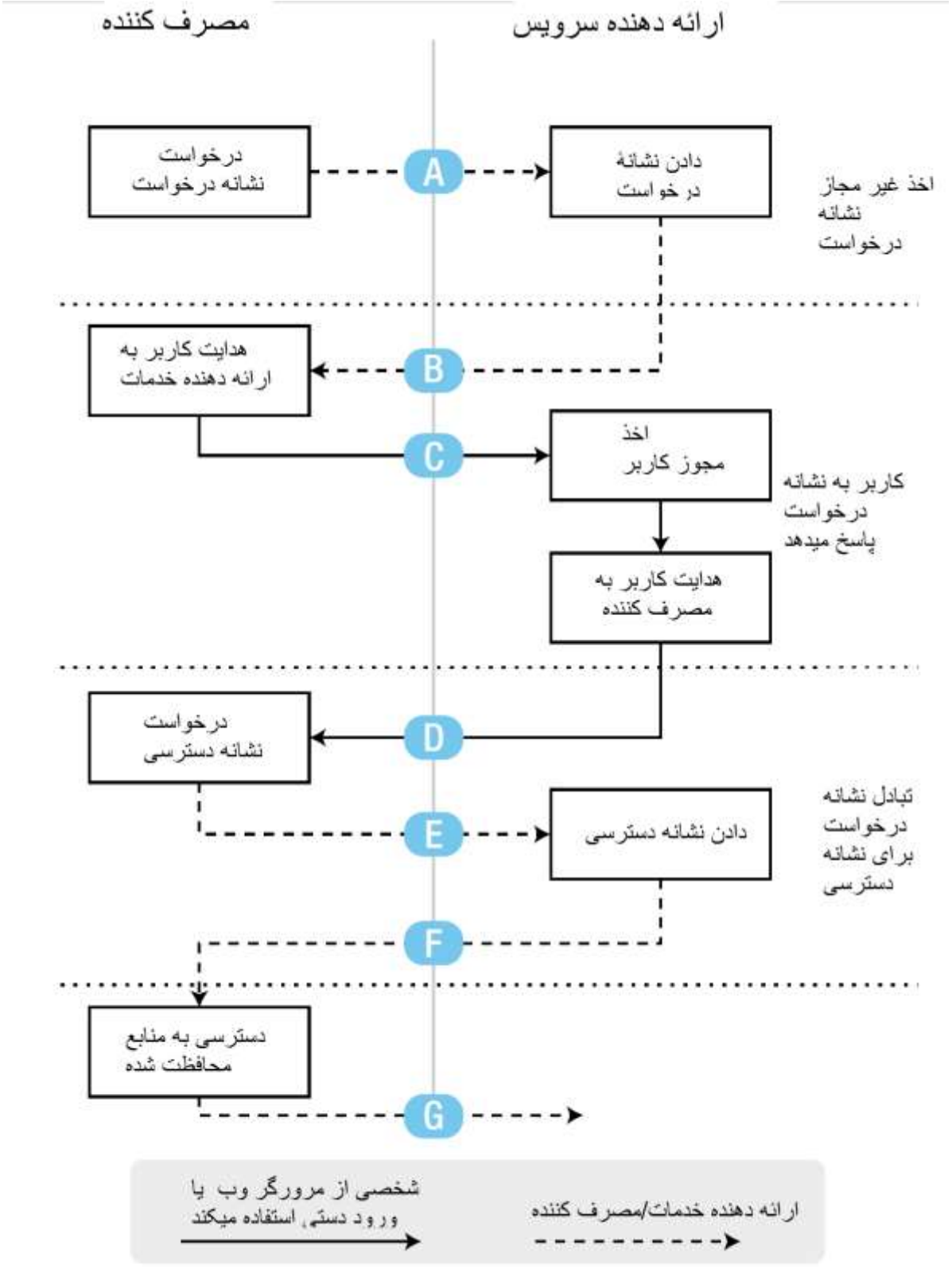


تصویر ۵-۱۵: انتخاب نوع برنامهٔ کاربردی و تکمیل اطلاعات آن



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

حالا که شما ID مشتریOAuth را گرفتید، اجازه دهید به جریان احراز هویت یک برنامهٔ کاربردی با OAuth نگاه کنیم(تصویر ۵-۱۷ را ببینید).



تصویر ۵-۱۷: جریان احراز هویت OAuth

OAuthیک فرایند چندمرحله‌ای است که سه بخش تعاملی اصلی دارد. مصرف‌کننده یک نرم‌افـزاری اسـت

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

۱. جریان A: نرم‌افزار مصرف‌کننده(برنامه‌ی اندروید شما) 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 مجدد کند. پس برنامهٔ شما یک صفحهٔ وب‌باز می‌کند که کاربر نهایی را به URL خاص هدایت خواهد کرد.
۳. جریان C: کاربر نهایی گواهی خودش را در این صفحه وارد می‌کند. لازم به ذکر است که او باید در وب‌سایت 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 نمی‌کرد.
۴. جریان D: گوگل اجازهٔ دسترسی به برنامهٔ شمارا می‌دهد و با یک جواب به نشانهٔ اینکه آیا به نشانهٔ درخواست مجوز داده‌شده است پاسخ 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 دستیابی به هر منبع محافظت‌شده(در این مورد، اطلاعات حساب کاربری گوگل) استفاده می‌کند تا زمانی که این نشانه منقضی شود.

در حال حاضر برنامهٔ شما، دسترسی به‌حساب کاربری گوگل را بدون نیاز به ذخیرهٔ گواهی کاربر نهایی با موفقیت به دست آورده است. حتی اگر تلفن 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 شما باید با یک مرورگر وب وارد برنامه‌ی کاربردی وب شود و صراحتاً دسترسی به منابع را اجازه دهد. بعد، برنامهٔ موبایل شما و برنامهٔ وب back-end با استفاده از درخواست و نشانه‌های دسترسی ارتباط برقرار خواهند کرد. مهم‌تر از همه، برنامهٔ اندروید شما نام کاربری و رمز عبور برنامهٔ وب شمارا 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 دراین‌ارتباط ارسال نمی‌شود. در عوض، یک‌طرف، طرفِ دیگر را برای چالش می‌طلبد. سپس طرف دیگر با توجه به یک الگوریتم خاص انتخاب‌شده و تابع رمزنگاری، یک‌رشته نصادفی از اطلاعات را رمز می‌کند. کلید استفاده‌شده برای رمزنگاری این اطلاعات، رمز عبور کاربر است. 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 پایان آن رمزنگاری می‌کند. سپس متن رمزی‌شده سنجیده می‌شود; اگر منطبق باشد، به کاربر اجازه‌ی دسترسی داده می‌شود.

خلاصه

در این فصل، بیشتر روی چگونگی انتقال داده‌هایمان به‌صورت امن از برنامه‌ی موبایل به برنامهٔ وب تمرکز شده بود. همچنین پروتکل‌ها و روش‌هایی برای امن کردن داده‌هایمان در زمان انتقال را پوشش دادیم. همچنین فهمیدیم که، در برخی موارد، نمی‌توانیم به برخی از پروتکل‌ها اعتماد کنیم. در چنین مواردی ، ما روش‌هایی را بررسی کردیم که می‌توانست در محافظت از گواهی کاربر نهایی در برابر دزدیده شدن یا استراق سمع کمک کند.

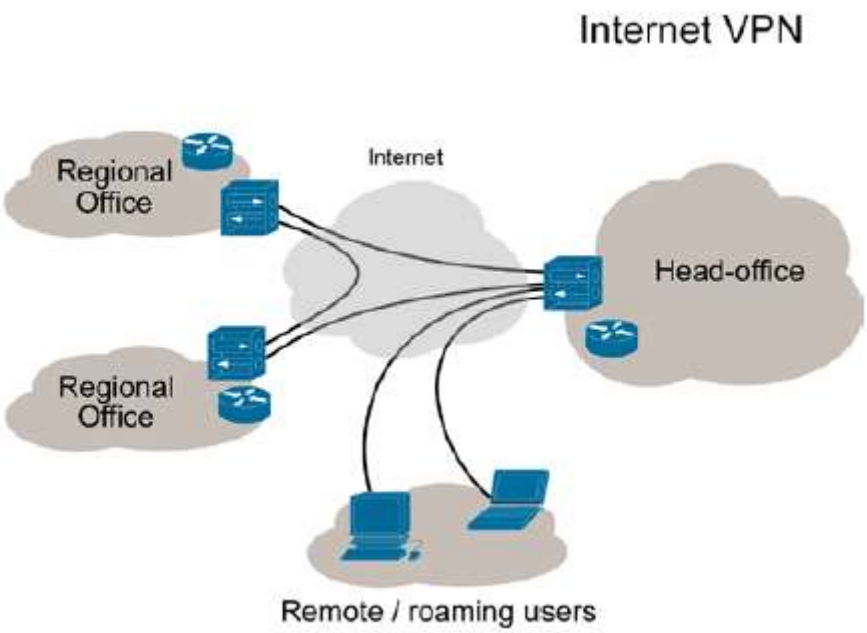
ما همچنین موضوعاتی را که شامل امنیت برنامه‌های تحت وب می‌شد را پوشش دادیم. با توجه به اینکه بیشتر برنامه‌های موبایل در بعضی از شکل‌ها با یک برنامه‌ی وب ارتباط برقرار می‌کند، بهتر است که بدانیم چگونه این فناوری کار می‌کند. درنهایت، ما به بعضی از منابع مفید نگاه کردیم که در امن کردن برنامه‌های وبمان به ما کمک می‌کرد، و همچنین برخی از نمونه‌های واقعی برای محافظت از گواهی‌های کاربر درزمانی که در حال انتقال‌اند را دیدیم.

٦ امنیت در کسب‌وکار

تاکنون از جنبه توسعه‌دهندگان مستقل به برنامه‌های کاربردی موبایل توجه شده است. اکثر توسعه‌دهندگان تجاری و بزرگ با برون‌سپاری، کارهایشان را به توسعه‌دهندگان کوچک واگذار می‌کنند و لذا نیازی به مدیریت یک گروه کاری در شرکت ندارند.

٦-١ ارتباط

ارتباط با محل شرکت از راه دور، امروزه به یک امر متداول تبدیل‌شده است. شبکه، مخابرات و برون‌سپاری و تائید هویت کاربران، جهت ارتباط راه دور، نیاز است. شبکه VPN ، یک ابزاری مناسب برای این کار است.



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 کسب‌وکار، عمل می‌کند. کاربران راه دور قادرند از طریق VPN، مانند یک کاربر محلی به منابع شبکه خصوصی موردنظر خود، دسترسی داشته باشند. در فضای موبایلی هم VPN، کم‌کم همین نقش را بازی می‌کند. کسب‌وکارها مایل هستند که طراحی به شکلی انجام شود که از طریق VPN، بتوانند به منابع خودشان دسترسی داشته باشند.

اگر کسب‌وکاری از VPN، استفاده نمی‌کند، لازم است داده بین کاربر و سرور به‌صورت رمزنگاری مبادله

شوند. البته اگر حجم داده زیاد باشد این کار باعث هزینه زیادی می‌شود که بایستی از فشرده‌سازی نیز استفاده شود. همه این‌ها منجر به استفاده از منبع پردازنده می‌شود.

۶-۲ برنامه‌های کاربردی تجاری

برنامه‌های کاربردی کسب‌وکار (تجاری) عمدتاً در حوزه برنامه‌ریزی منابع کسب‌وکار (enterprise resource planning: ERP)، هستند. برنامه‌های ERP ، شامل موارد زیر نیز هست:

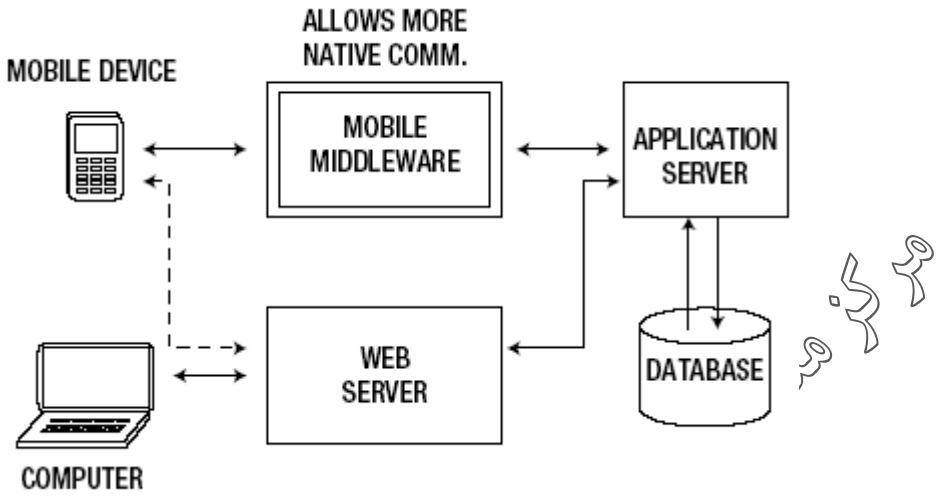
- مدیریت زنجیره پشتیبانی
- مدیریت ارتباط با مشتری
- تولید صنعتی
- منابع انسانی
- حسابداری و مالی
- مدیریت پروژه

برنامه‌های ERP، باید خوب کار کند و هر برنامه جدید باید با برخی دستگاه‌های قدیمی موجود نیز کار کند. یکی از بهترین روش‌ها برای انجام این عمل، به‌کارگیری میان‌افزار (middleware) موبایلی هست.

۶-۳ میان‌افزار موبایل

بهتر است که برنامه موبایل خودتان را طوری بسازید که با برنامه‌های موجود کسب‌وکار، تعامل و کار کند. بسترهای میان‌افزار موبایل بین برنامه‌های موبایل و دستگاه‌های کسب‌وکار موجود عمل می‌کنند. هدف این است که برنامه موبایل بتواند با داده‌های کاربرد کسب‌وکار، تعامل و کار کند.

در یک برنامه موبایل بانکی، میان‌افزار موبایل، عمل تجمیع با یک برنامه بسته و خاص را انجام می‌دهد. تولیدکننده یک میان‌افزار موبایلی از نوع یک مترجم صفحه‌نمایش ایجاد می‌کند که یک کاربرد طرف سرور هست و وب‌سایت را از کاربرد بانکی بازیابی کرده و آن را در یک صفحه خاص کپی کرده و درنهایت آن‌ها را به فرمت موبایلی درمی‌آورد. مطابق تصویر ۶-۲ یک برنامه کاربردی موبایل قادر است که با یک سیستم میان‌افزار ارتباط برقرار کند که تجرد داده و واسط کاربری برنامه کاربردی کسب‌وکار را فراهم می‌کند. البته یک کاربر موبایل می‌تواند از طریق یک مرورگر موبایلی با برنامه کاربردی کسب‌وکار نیز ارتباط مستقیم داشته باشد که تجربه نشان داده این روش مناسب نیست.



تصویر ٦-٢ مثال میان‌افزار موبایل

لذا دو سناریو کلیدی در هنگام تولید برنامه‌های موبایلی کسب‌وکار لازم است موردتوجه قرار گیرد. دسترسی به پایگاه داده (database access) و نمایش داده (data representation) دو موضوع چالشی هستند که لازم است موردتوجه دقیق قرار گیرند.

٦-٣-١ دسترسی به پایگاه داده

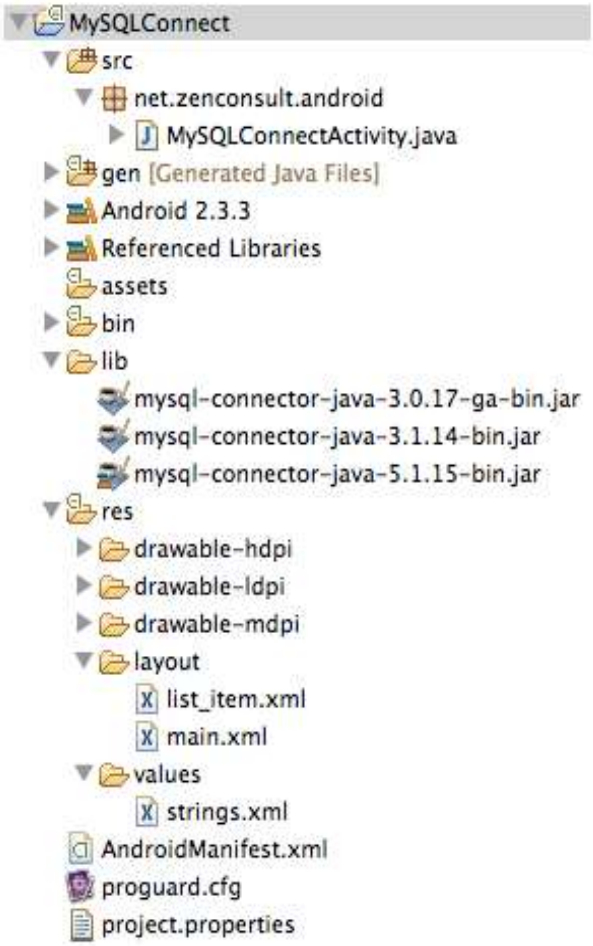
اندروید دو بسته javax.sql و java.sql را جهت دسترسی به سرورهای پایگاه داده فراهم کرده است. در اینجا و در ابتدا به یک روش ناامن (insecure) ولی ساده اشاره می‌شود. البته بعداً روش‌های بهتری معرفی خواهد شد. برای آنکه متوجه شوید که چرا یک روش به‌صورت ناامن است این روش ساده توضیح داده‌شده است و این باعث می‌شود که روش‌های امن (secure) را بهتر متوجه شوید.

یک برنامه کاربردی به پایگاه داده MySQL متصل می‌شود و داده‌هایی را از جدول apress پایگاه داده می‌خواند. جهت عملکرد صحیح آن باید هر دو دستگاه اندروید و سرور پایگاه داده در یک شبکه باشند. لازم است که سرور پایگاه داده، طوری تنظیم‌شده باشد که به آدرس عمومی (public IP address) گوش بدهد که برای این کار بایستی فایل my.cnf در سرور MySQL اصلاح و ویرایش شود. کد زیر نمایی از پایگاه داده مربوطه را نمایش می‌دهد. مطمئن باشید که پایگاه داده‌ای بانام android first ایجادشده است و بعدازآن جدول مربوطه نیز ایجادشده و تعدادی رکورد اطلاعات در آن نیز قرار گیرد تا برنامه کاربردی در موقع ارتباط با آن بتواند داده را بازیابی کند.

کد ٦-١

```
CREATE TABLE `apress` (
`id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(50) NOT NULL DEFAULT '',
`email` varchar(50) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE = MyISAM AUTO_INCREMENT = 4 DEFAULT CHARSET = latin1;
```

حالا کار تولید برنامه کاربردی شروع می‌شود. یک project بانام MySQLConnect ایجاد شود و در شاخه (فولدر) پروژه، یک فولدر جدید بانام lib ایجاد گردد. پس‌ازآن آخرین نسخه MySQL Connector/J را از www.mysql.com/products/connector/ دانلود شود و سپس آن archive، را بازکرده و فایل jar. را به دایرکتوری lib کپی شود. آن فایل باید به شکل mysql-connector-java-5.1.15-bin.jar باشد. اگر از Eclipse جهت تولید برنامه کاربردی استفاده می‌شود آنگاه project layout چیزی مشابه شکل بعدی خواهد بود.



تصویر ٦-٣ ساختار پروژه اتصال mysql

در این مثال یک layout بانام ListView ایجاد می‌شود. این‌یک لیست کامل تمام صفحه از داده‌ای که از پایگاه داده بازیابی می‌شود، هست. ListView شامل آیتم‌هایی هست که باید هر آیتم در فایل جداگانه‌ای تعریف شود. برای این کار یک فایل XML جدید بانام list_item.xml ایجادشده که شامل کد نشان داده‌شده در کد ٦-٢ هست و این فایل در پوشه layout ذخیره می‌شود.

کد ٦-٢: محتوای فایل list_item.xml

```
<?xml version = "1.0" encoding = "utf-8"?>
<TextView xmlns:android = "http://schemas.android.com/apk/res/android"
android:layout_width = "fill_parent"
android:layout_height = "fill_parent"
android:padding = "10dp"
android:textSize = "16sp" >
</TextView>
```

این بیان می‌کند که هر آیتم لیست از نوع متن هست و جزئیات بیشتری در مورد اندازه فونت و لایه گذاری لیست می‌دهد. در ادامه کد فایل MySQLConnectActivity.java هست.

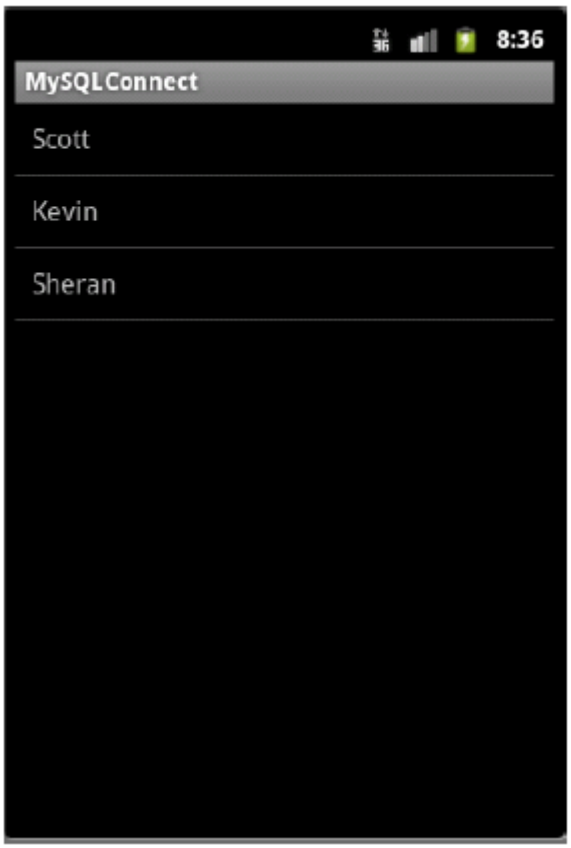
کد ٦-٣: کد فایل MySQLConnectActivity.java

```
package net.zenconsult.android;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Enumeration;
import java.util.Hashtable;
import android.app.ListActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
public class MySQLConnectActivity extends ListActivity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Connection conn = null;
        String host = "192.168.3.105";
        int port = 3306;
        String db = "android";
        String user = "sheran";
```

```
        String pass = "P@ssw0rd";
        String url = "jdbc:mysql://" + host + ":" + port + "/" + db +
"?user = "
        + user + "&password = " + pass;
        String sql = "SELECT * FROM apress";
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            conn = DriverManager.getConnection(url);
            PreparedStatement stmt = conn.prepareStatement(sql);
            ResultSet rs = stmt.executeQuery();
            Hashtable < String, String > details = new Hashtable <
String, String > ();
            while (rs.next()) {
                details.put(rs.getString("name"), rs.getString("email"));
            }
            String[] names = new String[details.keySet().size()];
            int x = 0;
            for (Enumeration < String > e = details.keys();
e.hasMoreElements();) {
                names[x] = e.nextElement();
                x++;
            }
            conn.close();
            this.setListAdapter(new ArrayAdapter < String > (this,
            R.layout.list_item, names));
            ListView lv = getListView();
            lv.setTextFilterEnabled(true);
            lv.setOnItemClickListener(new OnItemClickListener() {
                public void onItemClick(AdapterView < ? > parent, View
view,
                int position, long id) {
                    Toast.makeText(getApplicationContext(),
                    ((TextView) view).getText(),
Toast.LENGTH_SHORT).show();
                }
            });
        } catch (ClassNotFoundException e) {
            Log.e("MYSQL", "Class not found!");
        } catch (SQLException e) {
            Log.e("MYSQL", "SQL Exception " + e.getMessage());
        } catch (InstantiationException e) {
            Log.e("MYSQL", "Instantiation error " + e.getMessage());
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

به دلیل آنکه دسترسی به شبکه انجام می‌شود، برنامه کاربردی بایستی دارای مجموعه مجوز android.permission.INTERNET در فایل AndroidManifest.xml باشد.

آنگاه پروژه لازم است ذخیره شود و در شبیه‌ساز اندروید (Android simulator) اجرا شود. سپس برنامه کاربردی، شروع می شود و با پایگاه داده ارتباط برقرار می‌کند و داده را بازیابی می نماید و خروجی برنامه (تصویر ٦-٤) نشان داده می شود.



تصویر ٦-٤ خروجی برنامه

چنانچه مشاهده شد، باآنکه خواندن داده به‌طور مستقیم از پایگاه داده امکان دارد، ولی به نظر می‌رسد با سختی‌هایی روبه‌رو هستیم؛ بنابراین به کتابخانه‌های JDBC driver برای برنامه کاربردی، نیاز است.

در بعضی موارد اگر با پایگاه داده بدون JDBC drivers ارتباط برقرار کنید، با دشواری‌هایی روبه‌رو می‌شوید. اگر به نکات امنیتی توجه شود، ملاحظه می‌شود که سرور پایگاه داده بایستی متصل به اینترنت یا VPN باشد زیرا هر دو دستگاه موبایل و سرور پایگاه داده، لازم است که باهم تعامل داشته باشند. سرانجام مشاهده می‌شود که اطلاعات حساب (credentials) سرور پایگاه داده نیز لازم است در برنامه کاربردی ذخیره شود.

بخش بعدی کد را در ادامه ملاحظه کنید:

کد ٦-٤

```
Connection conn = null;
String host = "192.168.3.105";
int port = 3306;
String db = "android";
String user = "sheran";
String pass = "P@ssw0rd";
String url = "jdbc:mysql://" + host + ":" + port + "/" + db + "?user = "
+ user + "&password = " + pass;
String sql = "SELECT * FROM apress";
```

خطوط بالا با string user , string pass شروع می‌شود و نشان می‌دهد که چگونه گواهی پایگاه داده در کد قرارگرفته است. اگر موبایل در معرض خطر قرار گیرد یک هکر می‌تواند گواهی دسترسی به پایگاه داده را از کد بخواند و از طریق یک کامپیوتر دیگر با دسترسی به پایگاه داده مستقیما به آن حمله کند. بنابراین بهترین روش، استفاده از JDBC اتصال محلی (native)، به پایگاه داده در یک برنامه کاربردی نیست. بهتر این است که با یک ماژول میان‌افزار موبایلی، دسترسی به پایگاه داده با روش امن‌تر و بهتر انجام شود.

چگونه می‌توان دسترسی به پایگاه داده را بهبود بخشید؟ ساده‌ترین و متداول‌ترین روش استفاده از http است. با کمک http، می‌توان دسترسی به پایگاه داده را امن‌تر کرد و بر خلاف روش قبل به اطلاعات حساب پایگاه داده نیاز نیست. همچنین با کمک ssl، و در صورت نیاز یک لایه رمزنگاری اضافی، می‌توان امنیت را نیز بهتر کرد.

اما چگونه از http برای درخواست داده از پایگاه داده استفاده می‌شود؟ می‌توان از وب‌سرویس، جهت بازیابی داده استفاده کرد. به‌جای آنکه آن را خیلی پیچیده کرد می‌شود از REST (representational state transfer) جهت ارتباط استفاده کرد. به‌کارگیری یک RESTful API، باعث ساده شدن دسترسی به داده خواهد شد، مانند:

https://192.168.3.105/apress/members

با استفاده از این نوع درخواست، می‌شود مجموعه داده یکسان از پایگاه داده، مانند مثال قبلی از MySQLConnect ، بازیابی کرد. این در مقایسه با یک درخواست http خیلی ساده‌تر هست. به دلیل آنکه از http استفاده‌شده است بایستی برای پاسخ نیز از http، استفاده کرد که باعث موضوع نمایش داده می‌شود که در بخش بعدی به آن پرداخته خواهد شد.

می‌توان چندین کتابخانه برای کارهای مختلف ایجاد کرد که یکی از آن‌ها برای ارتباط با پایگاه داده باشد.

این روش باعث سرعت بخشیدن به موضوع تولید برنامه کاربردی می‌شود. حالا شما لازم است یک میان‌افزار برای توسعه برنامه‌های خود ایجاد کنید.

۶-۳-۲ نمایش داده

منظور از نمایش داده این است که چگونه برنامه کاربردی شما، داده را از یک برنامه کاربردی وب دریافت می‌کند. در این حالت ما می‌خواهیم نحوه دریافت داده توسط برنامه کاربردی را به شکل استاندارد درآوریم. پراستفاده‌ترین شیوه‌های مورداستفاده امروزی XML و JSON(JavaScript Object Notation) می‌باشند. علاوه بر این بسیاری از کتابخانه‌های متن‌باز دیگر نیز موجود هستند که می‌شود از آن‌ها استفاده کرد یا اصلاح کرد تا مناسب برنامه کاربردی شما شوند.

به درخواست RESTful API برگردیم و دو پاسخ ممکن که از میان‌افزار موبایل قابل دریافت هست، در نظر بگیریم.

کد ۵-۶

```
XML
<?xml version = "1.0" encoding = "UTF-8"?>
<apress>
<users>
<user name = "Sheran" email = "sheranapress@gmail.com" />
<user name = "Kevin" email = "kevin@example.com" />
<user name = "Scott" email = "scottm@example.com" />
</users>
</apress>
JSON
{
users:{
user:[
        {
name:'Sheran',
email:'sheranapress@gmail.com'
        },
        {
name:'Kevin',
email:'kevin@example.com'
        },
        {
name:'Scott',
email:'scottm@example.com'
        }
        ]
    }
```

```
}
```

مزیت آن این است که شما لازم نیست کدی بنویسید که داده فرمت JSON , XML را بخواند. اندروید دارای کتابخانه‌هایی برای هر دو مورد هست. بیایید مجدد به کد برگردیم. یک پروژه ایجاد کرده و نام آن را RESTFetch بگذارید. یک فایل list_item.xml مشابه مثال قبلی ایجاد کرده و سپس مجوز android.permission.INTERNET را به آن بدهید. لیست بعدی شامل کدی است که یک درخواست ایجاد کرده، نمایش XML را پردازش کرده و نتایج را در لیست قرار می‌دهد. و تصویر ۶-۵ خروجی آن هست.

کد ۶-۶

```
package net.zenconsult.android;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.StringReader;
import java.net.URI;
import java.net.URISyntaxException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import android.app.ListActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
public class RESTFetchActivity extends ListActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        BufferedReader in = null;
        try {
            HttpClient client = new DefaultHttpClient();
            HttpGet request = new HttpGet();
            request.setURI(new
URI("http://192.168.3.105/apress/members"));
            HttpResponse response = client.execute(request);
            in = new BufferedReader(new
```

```
InputStreamReader(response.getEntity()
            .getContent()));
            StringBuffer sb = new StringBuffer("");
            String line = "";
            String newLine = System.getProperty("line.separator");
            while ((line = in.readLine()) ! = null) {
                sb.append(line + newLine);
            }
            in.close();
            Document doc = null;
            DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            InputSource is = new InputSource();
            is.setCharacterStream(new StringReader(sb.toString()));
            doc = db.parse(is);
            NodeList nodes = doc.getElementsByTagName("user");
            String[] names = new String[nodes.getLength()];
            for (int k = 0; k < nodes.getLength(); ++k) {
                names[k] =
nodes.item(k).getAttributes().getNamedItem("name")
                .getNodeValue();
            }
            this.setListAdapter(new ArrayAdapter < String > (this,
            R.layout.list_item, names));
            ListView lv = getListView();
            lv.setTextFilterEnabled(true);
            lv.setOnItemClickListener(new OnItemClickListener() {
                public void onItemClick(AdapterView < ? > parent, View
view,
                int position, long id) {
                    Toast.makeText(getApplicationContext(),
                    ((TextView) view).getText(), Toast.LENGTH_SHORT)
                    .show();
                }
            });
        } catch (IOException e) {
            Log.e("REST", "IOException " + e.getMessage());
        } catch (URISyntaxException e) {
            Log.e("REST", "Incorret URI Syntax " + e.getMessage());
        } catch (ParserConfigurationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SAXException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

تصویر ٦-٥: خروجی درخواست RESTful API با XML Response

کد درخواست/ پاسخ JSON و خروجی آن را کد زیر و تصویر ٦-٦ مشاهده کنید.

کد ٦-٧

```
package net.zenconsult.android;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import java.net.URISyntaxException;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import android.app.ListActivity;
import android.os.Bundle;
import android.util.Log;
```

```
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
public class RESTJSONActivity extends ListActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        BufferedReader in = null;
        try {
            HttpClient client = new DefaultHttpClient();
            HttpGet request = new HttpGet();
            request.setURI(new
URI("http://192.168.3.105/apress/members.json"));
            HttpResponse response = client.execute(request);
            in = new BufferedReader(new
InputStreamReader(response.getEntity()
            .getContent()));
            StringBuffer sb = new StringBuffer("");
            String line = "";
            while ((line = in.readLine()) ! = null) {
                sb.append(line);
            }
            in.close();
            JSONObject users = new JSONObject(sb.toString())
            .getJSONObject("users");
            JSONArray jArray = users.getJSONArray("user");
            String[] names = new String[jArray.length()];
            for (int i = 0; i < jArray.length(); i++) {
                JSONObject jsonObject = jArray.getJSONObject(i);
                names[i] = jsonObject.getString("name");
            }
            this.setListAdapter(new ArrayAdapter < String > (this,
            R.layout.list_item, names));
            ListView lv = getListView();
            lv.setTextFilterEnabled(true);
            lv.setOnItemClickListener(new OnItemClickListener() {
                public void onItemClick(AdapterView < ? > parent, View
view,
                int position, long id) {
                    Toast.makeText(getApplicationContext(),
                    ((TextView) view).getText(), Toast.LENGTH_SHORT)
                    .show();
                }
            });
        } catch (IOException e) {
            Log.e("RESTJSON", "IOException " + e.getMessage());
        } catch (URISyntaxException e) {
            Log.e("RESTJSON", "Incorret URI Syntax " + e.getMessage());
        } catch (JSONException e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }
    }
}
```

تصویر ٦-٦: خروجی از RESTful API query با یک JSON response

چنانچه لازم باشد می‌توان هر دو XML و SJON را باهم ترکیب کرد. جهت تشخیص نوع پاسخ لازم هست نوع فایل نیز ضمیمه شود. بنابراین برای پاسخ XML، باید http://192.168.3.105/apress/members.xml و برای پاسخ SJON، باید http://192.168.3.105/apress/members.json فراخوانی شود.

دوباره می‌شود مثال را چنان اصلاح کرد تا اینکه داده پاسخ را بررسی کرده تا ساختار را خودکار کشف کند.

۷ پروژه امنیت برنامه‌های کاربردی موبایل

در سال ۲۰۱۳ آمارهایی از آسیب‌پذیری‌های جدید برنامه‌های کاربردی موبایل جمع‌آوری شد؛ آنچه اینجا مشاهده می‌شود نتیجه‌ای از این اطلاعات است.

تصویر ۷-۱ ده خطر برتر امنیتی موبایل

۷-۱ M1- کنترل‌های ضعیف سمت سرور

این مربوط به آسیب‌پذیری‌ها در سمت سرور و برنامه‌های وب است که OWASP به‌طور جداگانه آن را در پروژه‌های OWASP Web Top Ten یا Cloud Top Ten بررسی نموده است.

۷-۲ M2- ذخیره ناامن داده‌ها

۷-۲-۱ عوامل تهدید

دزدیده شدن یا گم‌شدن موبایل ؛ یک بدافزار یا برنامه دوباره بسته‌بندی‌شده[۱] برای دزدیدن اطلاعات

[۱] Repackaged app

۷-۲-۲ نحوه حمله

با دسترسی فیزیکی به موبایل می‌توان آن را به یک کامپیوتر متصل نمود و با کمک ابزارهـایی آزاد اطلاعـات برنامه‌های نصب‌شده را استخراج نمود این اطلاعات معمول شامل اطلاعـات هـویتی و یـا دیگـر اطلاعـات حساس هست. یا با نصب یک بدافزار یا نرم‌افزار معتبر دستکاری‌شده می‌توان اطلاعات را دزدید.

۷-۲-۳ علت ضعف امنیتی

گروه‌های برنامه‌نویس فرض می‌کنند که کاربران یا بـدافزارها بـه سیسـتم فایـل موبایـل دسترسـی ندارنـد و ازاین‌رو نمی‌توانند داده‌های حساس ذخیره‌شده در حافظه را ببینند. سیسـتم فایل‌هـا بـه‌راحتی قابل‌دسـترس است. به همین دلیل همگان باید منتظر یک کاربر خرابکار یا بدافزاری باشند که قصـد دزدیـدن اطلاعـات را دارد. روش‌های حفاظت رمزنگاری در موبایل‌های روت شده یا قفل‌شکسته به‌راحتی دور زده می‌شود. زمانی که داده‌ای موردحفاظت نباشد می‌توان از ابزارهایی برای مشاهده اطلاعات برنامه‌ها استفاده کرد.

۷-۲-٤ تأثیرات فنی

در بهترین حالت اطلاعات یک نفر و در بدترین حالت اطلاعات افراد بی‌شماری قابل‌دسترسـی اسـت. ایـن اطلاعات می‌تواند شامل :نـام کـاربری، تـوکن هـا ی احـراز هویـت، رمزهـا، کوکی‌هـا، اطلاعـات مکـانی، UDID/EMEI، نام موبایل، نام ارتباط شبکه‌ای، اطلاعات شخصی نظیر DOB و آدرس و روابط اجتمـاعی و اطلاعات کارت اعتباری، اطلاعات برنامه‌ها نظیـر لاگ هـای ذخیره‌شده و اطلاعـات عیب‌یـابی و پیام‌هـای ذخیره‌شده برنامه و تاریخچه مبادلات باشد.

۷-۲-۵ تأثیرات تجاری

این نوع آسیب‌پذیری‌ها معمولاً خطرات جدی برای تجارت‌ها دارند ازجملـه سـرقت شناسـه، کلاه‌بـرداری، صدمه به اعتبار، نقض سیاست خارجی و از دست رفتن اسناد.

۷-۲-٦ تشخیص آسیب‌پذیری

برنامه‌ها برای ذخیره اطلاعات باید از APIهایی استفاده کنند کـه داده‌هـا را به‌صـورت امـن ذخیـره می‌کنـد. OWASP مشاهده کرده است که اغلب داده‌هـا بـا ایـن روش‌هـا نـاامن ذخیره‌شـده‌اند ازجملـه پایگـاه داده SQLite، فایل‌های لاگ، فایل Plist، فایل اظهارنامه و ذخیره داده‌های XML، ذخیره داده‌هـای بـاینری،ذخیره کوکی‌ها، کارت حافظه SD، همسان‌سازی ابری. همچنـین بـاوجود ذخیـره اطلاعـات به‌صـورت رمـز شـده

حمله‌کننده‌ها می‌توانند با انجام یک حمله باینری بر روی برنامه کلیدهای رمزنگاری را به دست آورند.

۷-۲-۷ جلوگیری از آسیب‌پذیری

قاعده اصلی برنامه‌های موبایل فقط اطلاعاتی را ذخیره کنند که لازم است. درصورتی‌که به‌عنوان یک برنامه‌نویس باید بدانیم که اطلاعات با یک لمس در موبایل از بین می‌رود و از طرفی باید احتمال سرقت اطلاعات با یک سوءاستفاده ریشه را بررسی نمود و اگر قابلیت استفاده در برابر امنیت بسیار مهم‌تر بود پیشنهاد می‌شود که APIهای امن ذخیره داده به‌طور دقیق بررسی و به طرز صحیح استفاده شود. در اندروید می‌توان از روش‌های زیر استفاده نمود:

- استفاده از متد setStorageEncryption برای ذخیره رمز شده داده‌های محلی
- استفاده از توابع کتابخانه javax.crypto یا کلیدهای متقارن AES128 برای ذخیره رمز شده داده‌ها بر کارت حافظه SD
- اطمینان از MODE_WORLD_READABLE نبودن اشیاء Shared Preferences مگر برای اشتراک اطلاعات بین برنامه‌ها
- اجتناب از وابستگی فقط به کلیدهای رمزنگاری کد شده در هنگام ذخیره اطلاعات حساس
- رسیدگی برای فراهم آوردن یک‌لایه رمزنگاری اضافی در بالای هر روش رمزنگاری پیش‌فرض در سیستم

۷-۳ M3- حفاظت ناکافی در لایه انتقال

۷-۳-۱ عوامل تهدید

معمولاً برنامه‌های موبایل، داده‌ها را در یک مد مشتری-سرویس‌دهنده تبادل می‌کنند. برای انتقال داده‌ها از شبکه اینترنت و دیگر بسترهای انتقال داده در موبایل استفاده می‌شود. حمله‌کنندگان می‌توانند با سوءاستفاده از آسیب‌پذیری‌های موجود در این بسترها داده‌های حساس را شنود کنند. از عوامل تهدیدات می‌توان دسترسی متخاصم به شبکه محلی برای آگاهی از ترافیک شبکه ، دستگاه‌های شبکه نظیر روتر و... و بدافزارهای موبایلی نام برد.

۷-۳-۲ نحوه حمله

حمله‌کننده با نظارت بر ترافیک شبکه به‌خصوص ترافیک غیر رمز شده می‌تواند به اطلاعات دلخواه برسد که

معمولاً برای حمله‌کنندگان آسان است.

۷-۳-۳ علت ضعف امنیتی

برنامه‌ها نمی‌توانند در هرجایی از SSL استفاده نمایند لذا به‌طور مداوم از ترافیک شبکه خود محافظت نمی‌کنند. در برخی موارد هم پیاده‌سازی امنیت انتقال در برنامه‌ها به‌درستی انجام نمی‌شود. البته نقص‌های اساسی را می‌توان با نظارت بر ترافیک شبکه مشاهده کرد. رفع آن‌ها با بررسی برنامه و پیکربندی آن حاصل می‌شود.

۷-۳-٤ تأثیرات فنی

نمایش آشکار اطلاعات کاربری می‌تواند موجب سرقت حساب کاربری شود و در مواردی که آن حساب کاربری اصلی باشد موجب حمله به کل سایت می‌شود. پیکربندی ضعیف SSL می‌تواند منجر به حملات مردمیانی و فیشینگ شود.

۷-۳-۵ تأثیرات تجاری

مهم‌ترین مشکل نقض حریم شخصی است که موجب سرقت هویت، کلاه‌برداری و صدمه به اعتبار می‌شود.

۷-۳-٦ تشخیص آسیب‌پذیری

با مشاهده ترافیک برنامه از طریق یک واسط و با پاسخ به این سؤالات می‌توان به وجود آسیب‌پذیری پی برد:

- آیا همه ارتباطات رمز شده‌اند؟
- آیا گواهی‌های SSL در اطلاعات موجود هستند؟
- آیا گواهی‌های SSL با امضای خودشان هستند؟
- آیا SSL از یک طول رمز کافی استفاده می‌کند؟
- آیا برنامه موردنظر گواهی‌نامه‌های معتبر پذیرفته‌شده کاربر را به‌عنوان منابع موثق می‌پذیرد؟

۷-۳-۷ جلوگیری از آسیب‌پذیری

- برنامه‌نویسان باید به‌طورکلی لایه شبکه را ناامن فرض کنند و به خطر شنود توجه کنند؛

- برای انتقال داده‌های حساس حتماً باید از SSL استفاده نمود که کلید رمز استاندارد قوی با طول مناسب را به کار می‌برد و گواهی‌نامه‌های آن توسط یک مرکز معتبر امضاشده باشد.
- هیچ‌گاه نباید از گواهینامه‌های خود امضاشده استفاده نمود.
- باید از نشست‌های SSL تودرتو به دلیل امکان فاش شدن توکن نشست اجتناب شود.
- یک اتصال امن باید بعد از احراز هویت سرور مقصد برقرار شود و فوراً از طریق پیامی تشخیص نامعتبر بودن گواهی‌نامه را به کاربر اطلاع دهد.
- هیچ‌گاه اطلاعات حساس از طریق کانال‌های غیر امن مانند پیامک و... نباید ارسال شود.
- به دلیل امکان شنود اطلاعات در دستگاه قبل از رمز شدن توسط SSL و وجود آسیب‌پذیری در این پروتکل بهتر است از یک‌لایه امنیتی جداگانه قبل از SSL برای اطمینان بیشتر استفاده شود.
- در اندروید باید بعد از پایان توسعه نرم‌افزار کدهای مربوط به پذیرش همه گواهینامه‌ها را پاک نمود. مانند:

org.apache.http.conn.ssl.AllowAllHostnameVerifier
SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER

در صورت استفاده از کلاس SSLSocketFactory باید مطمئن شد که بعدازآن متدهای CheckServerTrust برای بررسی گواهی‌نامه سرور پیاده‌سازی شده است.

۸-۳-۷ سناریوهای متخصصان آزمون نفوذ

- عدم بررسی گواهی‌نامه

برنامه با سرور یک ارتباط امن مبتنی بر SSL ایجاد می‌کند ولی برنامه هر گواهی‌نامه ارائه‌شده از سمت سرور را بدون بررسی می‌پذیرد و این تصدیق اعتبار دوطرفه را از بین می‌برد و برنامه نسبت به حمله مردمیانی از طریق یک پراکسی SSL مستعد است.

- مذاکره ضعیف در دست تکانی

بخشی از مرحله دست تکانی مذاکره برای تعیین رشته رمز است اگر این رشته رمز کوتاه تعیین شود رمزنگاری ضعیف انجام می‌شود و درنتیجه به‌آسانی توسط یک متخاصم رمزگشایی می‌شود.

- نشت اطلاعات حریم خصوصی

انتقال اطلاعات شخصی بین برنامه و سرور از طریق کانال ناامن به‌جای SSL، قابلیت اعتماد به این اطلاعات را به خطر می‌اندازد.

۷-٤ M4- نشت ناخواسته اطلاعات

۷-٤-۱ عوامل تهدید

متخاصم با عامل‌هایی همچون یک بدافزار موبایلی یا یک برنامه دستکاری‌شده یا با دسترسی مستقیم می‌تواند از آسیب‌پذیری‌های در این زمینه سوءاستفاده کند.

۷-٤-۲ نحوه حمله

با دسترسی مستقیم به موبایل و با ابزارهای آزاد جرم یابی می‌توان یک حمله انجام داد. همچنین متخاصم می‌تواند با اجرای یک کد مخرب که APIهای مجاز را فراخوانی می‌کند یک حمله را هدایت کند.

۷-٤-۳ علت ضعف امنیتی

معمولاً نشت ناخواسته اطلاعات زمانی اتفاق می‌افتد که برنامه‌نویس اطلاعات حساس برنامه را در مکانی ذخیره می‌کند که توسط سایر برنامه‌ها به‌آسانی قابل‌دسترسی است و یا در هنگام پردازش برنامه، اطلاعات واکشی شده توسط سیستم‌عامل در مکانی قرار می‌گیرد که در دسترس برنامه‌های دیگر است. این‌ها از بی‌اطلاعی برنامه‌نویس درباره نحوه ذخیره و پردازش اطلاعات توسط سیستم‌عامل ناشی می‌شود. البته با بررسی مکان‌های در دسترس همه برنامه‌ها می‌توان به وجود این آسیب‌پذیری پی برد.

۷-٤-٤ تأثیرات فنی

این آسیب‌پذیری با استخراج اطلاعات حساس ضربه شدیدی به کاربر می‌زند.

۷-٤-٥ تأثیرات تجاری

اطلاعات دزدیده‌شده می‌تواند موجب نقض حریم خصوصی و صدمه به اعتبار و کلاه‌برداری شود.

۷-٤-٦ تشخیص آسیب‌پذیری

آسیب‌پذیری‌های موجود در سیستم‌عامل ، محیط کامپایلر، فریمورک ها و سخت‌افزارهای جدید و... به همراه عدم دانش کافی برنامه‌نویسان می‌تواند نشت ناخواسته اطلاعات را به دنبال داشته باشد. این آسیب‌پذیری در فرایندهای داخلی بیشتر دیده‌شده است:

- روش سیستم‌عامل برای ذخیره کردن اطلاعات، ضرب کلیدها، لاگ ها، بافرها در حافظه نهان

- روش فریمورک توسعه برای ذخیره کردن این اطلاعات در حافظه نهان
- روش فریمورک های کسب‌وکار ، اجتماعی، آنالیز و تبلیغاتی برای ذخیره کردن اطلاعات در حافظه نهان

۷-٤-۷ جلوگیری از آسیب‌پذیری

در ابتدا مدل تهدید برای سیستم‌عامل و فریمورک را بررسی می‌کنیم تا نحوه اداره هر یک از موارد زیر را دریابیم:

- URL Caching (Both request and response)
- Keyboard Press Caching
- Copy/Paste buffer Caching
- Application backgrounding
- Logging
- HTML5 data storage
- Browser cookie objects
- Analytics data sent to 3rd parties

آشنایی با چگونگی کار پیش‌فرض سیستم‌عامل و فریمورک برای تعیین و اجرای کنترل‌ها لازم است.

۷-۵ M5- ضعف در احراز هویت و اعطای مجوز

۷-۵-۱ عوامل تهدید

متخاصم با ابزارهای موجود می‌تواند حملات خودکاری برای سوءاستفاده از آسیب‌پذیری‌های فرایندهای احراز هویت و اعطای مجوز اجرا نماید.

۷-۵-۲ نحوه حمله

متخاصم فقط یک‌بار لازم است فرایند احراز هویت را بررسی نماید و چگونگی آسیب‌پذیری را کشف کند. یک بدافزار یا باتنت با ارسال پیام‌های درخواست سرویس به سرویس‌دهنده پشتیبان برنامه فرایند احراز هویت را دور می‌زند و هرگونه ارتباط مستقیم بین برنامه و سرویس‌دهنده را جعل می‌کند.

۷-۵-۳ علت ضعف امنیتی

یک متخاصم با دور زدن فرایند احراز هویت ضعیف می‌تواند به‌طور گمنام قابلیت‌هایی را در سرویس‌دهنده پشتیبان یا برنامه کاربردی اجرا کند. احراز هویت ضعیف بیشتر به دلیل شکل کلمه عبور است که بر مبنای پین‌های ٤ رقمی است.

نیازمندی‌های احراز هویت در برنامه‌های کاربردی با احراز هویت در وب کاملاً متفاوت است. در وب این کار به‌صورت آنلاین و بلادرنگ انجام می‌شود به همین دلیل دسترسی به اینترنت لازم و همیشگی است ولی در برنامه‌های کاربردی کاربر آن همیشه آنلاین نیستند و از طرفی اتصال اینترنت در موبایل قابل‌اتکا و همیشگی نیست ازاین‌رو احراز هویت آفلاین انجام می‌شود. در هنگام پیاده‌سازی فرایند احراز هویت در برنامه‌های کاربردی باید موارد بسیاری را در نظر گرفت.

برای تشخیص احراز هویت ضعیف باید با حمله باینری سعی کرد با دور زدن احراز هویت آفلاین، قابلیت‌های برنامه را اجرا کرد. بعلاوه با از بین بردن هرگونه توکن نشست از درخواست‌های POST/GET باید سعی کرد تا قابلیت‌های سرویس‌دهنده پشتیبان را اجرا کرد.

برای تشخیص ضعف در فرایند اعطای مجوز باید بتوان با حمله باینری قابلیت‌هایی از برنامه را اجرا کرد که فقط کاربران ویژه مجوز اجرای آن را دارند. به‌علاوه متخصصان باید سعی کنند هر قابلیت ممتازی را در سمت سرویس‌دهنده پشتیبان اجرا کنند برای این منظور از توکن نشست کم امتیاز در درخواست‌های POST/GET استفاده می‌شود.

۷-۵-٤ تأثیرات فنی

وجود این آسیب‌پذیری سبب می‌شود هویت کاربر را نتوان احراز کرد و ازاین‌رو راه‌حل‌های شناسایی کاربری که درخواست قابلیت یا خدمتی را می‌دهد و همچنین راه‌حل‌های ثبت اطلاعات و بازرسی کاربر دیگر کارایی نداشته باشد. به همین دلیل در هنگام رخداد یک حمله نمی‌توان منبع آن و ماهیت سوءاستفاده و چگونگی مقابله با آن را تشخیص داد.

زمانی کنترل‌های احراز هویت شکسته می‌شود که هویت کاربر غیرقابل تشخیص باشد. هویت کاربر به نقش کاربر در سیستم و مجوزهایی مرتبط است که حمله‌کننده می‌تواند با جعل هویت کدهایی را اجرا کند و سیستم قادر به اعتبارسنجی مجوزهای کاربر نیست؛ بنابراین هم کنترل‌های احراز هویت و هم کنترل‌های اعطای مجوز شکسته می‌شود. شکسته شدن کنترل‌های اعطای مجوز می‌تواند موجب مشکل امتیاز بیش‌ازحد شود.

۷-۵-۵ تأثیرات تجاری

حداقل نتیجه احراز هویت ضعیف صدمه به اعتبار است و درواقع یک کاربر به‌طور گم نام یا مشخص با شکستن کنترل‌های اعطای مجوز قابلیت‌هایی را با امتیاز بالا در سیستم‌عامل اجرا کند که می‌تواند موجب

صدمه به اعتبار و کلاه‌برداری و سرقت اطلاعات شود.

۷-۵-۶ تشخیص آسیب‌پذیری

باید از طراحی الگوی های احراز هویت ناامن زیر در برنامه‌های موبایل اجتناب شود:

- در هنگام ایجاد برنامه موبایلی از یک برنامه وب نباید عوامل احراز هویت آن کمتر از عوامل احراز هویت برنامه وب باشد.
- احراز هویت محلی می‌تواند موجب آسیب‌پذیری‌های در سمت مشتری شود. برنامه‌ها به دلیل ذخیره محلی داده‌ها برای نیازهای تجاری فوری باید احراز هویت کاربر را آفلاین انجام دهند ولی این نوع احراز هویت می‌تواند با دست‌کاری در زمان اجرای برنامه یا تغییرات باینری در دستگاه‌های روت شده دور زده شود.
- در صورت امکان باید همه درخواست‌های احراز هویت در سمت سرور رسیدگی شود و همچنین باید مطمئن شد که داده‌ها بعد از تائید هویت کاربر برای برنامه مشتری ارسال شود.
- اگر در برنامه‌ای ذخیره محلی داده‌ها نیاز است به‌منظور اطمینان از در دسترس بودن داده‌ها فقط با ارائه گواهی‌نامه معتبر باید داده‌ها توسط یک کلید رمزنگاری مشتق شده از گواهی‌نامه ورود کاربر رمز شوند. البته این روش در برابر حمله باینری آسیب‌پذیر است و داده‌ها قابل رمزگشایی است.
- قابلیت "یادآوری مشخصات[1]" در برنامه‌ها برای احراز هویت پایا نباید هرگز کلمه عبور را در دستگاه ذخیره کند.
- برای اطمینان از کاهش خطر دسترسی به برنامه در دستگاه‌های دزدیده‌شده باید برنامه موبایل از یک توکن احراز هویت خاص دستگاه و قابل‌ابطال توسط کاربر استفاده کند.
- نباید از مقادیر قابل جعل همچون شناسه‌های دستگاه یا مکان جغرافیایی برای احراز هویت یک کاربر استفاده نمود.
- در صورت امکان باید قواعدی برای انتخاب کلمه عبور تعیین نمود تا برای نمونه نتوانند کلمات عبور ٤ رقمی انتخاب نمایند.

۷-۵-۷ جلوگیری از آسیب‌پذیری

توسعه‌دهندگان باید فرض کنند هر فرایند احراز هویت و اعطای مجوز در سمت مشتری مانند برنامه‌های

[1] Remember me

موبایلی قابل دور زدن است و در صورت امکان باید همه کنترل‌ها را در سمت سرویس‌دهنده وب نیز به‌منظور بررسی بیشتر، اعمال کنند ولی از طرفی لازمه برخی برنامه‌های موبایل احراز هویت و اعطای مجوز در داخل برنامه و به‌صورت آفلاین است در این صورت باید هرگونه تغییر کدهای غیرمجاز با ابزارهای بررسی صحت جاسازی‌شده در کد محلی را تشخیص داد.

۷-۵-۸ سناریوهای نمونه

توسعه‌دهندگان به‌اشتباه فرض می‌کنند که فقط کاربران محرز شده می‌توانند درخواست‌های یک سرویس را به سمت سرور ارسال کنند. از طرفی سرور نیز در طول پردازش یک درخواست، اعتبار کاربر درخواست دهنده را بررسی نمی‌کند؛ ازاین‌رو متخاصم می‌تواند با ایجاد و ارسال یک درخواست جعلی سرویس، یک قابلیت مجاز برای کاربران معتبر را اجرا کنند.

توسعه‌دهندگان به‌اشتباه فرض می‌کنند فقط کاربران مجاز می‌توانند از وجود یک تابع خاص در برنامه موبایل اطلاع یابند. ازاین‌رو آن‌ها انتظار دارند که فقط کاربران مجاز بتوانند از برنامه درخواستی را برای یک سرویس صادر کنند. ولی کد پشتیبان درخواست را بدون توجه به بررسی هویت کاربر درخواست دهنده پردازش می‌کند و از این طریق متخاصم می‌تواند یک قابلیت خاص را با کاربر کم امتیاز نیز اجرا کند.

توسعه‌دهندگان به‌منظور سهولت استفاده از برنامه ، به کاربران اجازه می‌دهند تا کلمه عبور با طول ٤ رقمی انتخاب کنند. سرور این کلمات عبور را به‌صورت درهم شده ذخیره می‌کند. به دلیل طول کوتاه این کلمه عبور، متخاصم قادر خواهد بود به‌راحتی با کمک جداول درهم کلمه عبور اصلی را پیدا کند. حال اگر فایل کلمات عبور در سرور به هر دلیلی در دسترس قرار گیرد کلمات عبور کاربران به‌راحتی قابل بازیابی است.

۷-٦ M6- رمزنگاری شکننده

۷-٦-۱ عوامل تهدید

با دسترسی مستقیم به دستگاه یا با یک بدافزار می‌توان به داده‌هایی که به گونه نادرست رمز شده‌اند دست‌یافت.

۷-٦-۲ نحوه حمله

متخاصم با دسترسی مستقیم به دستگاه یا با کمک یک بدافزار یا ثبت ترافیک شبکه می‌تواند داده‌های رمز شده را مشاهده و رمزگشایی نماید.

۷-۶-۳ علت ضعف امنیتی

متخاصم به دلیل استفاده برنامه از الگوریتم‌های رمزنگاری ضعیف یا نقص‌های فرایند رمزنگاری می‌تواند داده‌های رمز شده را رمزگشایی نماید.

۷-۶-٤ تأثیرات فنی

این آسیب‌پذیری موجب بازیابی غیرمجاز داده‌های حساس رمز شده در موبایل می‌شود.

۷-۶-۵ تأثیرات تجاری

ازجمله نتایج رمزنگاری شکننده می‌توان به نقص حریم خصوصی ، سرقت کد و اطلاعات و خسارت اعتباری اشاره کرد.

۷-۶-٦ تشخیص آسیب‌پذیری

معمولاً برنامه‌ها به دو شیوه از رمزنگاری شکننده استفاده می‌کنند: اول، برنامه‌ها از یک فرایند زمینه برای رمزنگاری و رمزگشایی استفاده می‌کنند که عیب‌یابی دارد و قابل سوءاستفاده است.دوم،برنامه یک الگوریتم رمزنگاری و رمزگشایی پیاده‌سازی می‌کند که ذاتاً ضعیف است و توسط متخاصم قابل رمزگشایی است. این دو شیوه در سناریوهای زیر تشریح شده است:

- تکیه به فرایندهای رمزنگاری تعبیه‌شده در برنامه

مدل امنیتی iOS برنامه‌ها را وادار می‌کند تا برای اجرا و همچنین مقابله با مهندسی معکوس، کد خود را رمز و امضا کنند. iOS برنامه را در حافظه رمزگشایی می‌کند و بعد از بررسی صحت امضا کد را اجرا می‌کند. ولی با ابزارهای در دسترس همچون GBD و ClutchMod می‌توان برنامه رمز شده را دانلود نمود و روی یک دستگاه قفل‌شکسته شده اجرا کرد و بعد از رمزگشایی برنامه در حافظه و قبل از اجرا یک دامپ از حافظه گرفته می‌شود و با کمک IDA Pro یا Hopper به‌راحتی می‌توان تحلیل ایستا و پویا روی برنامه انجام داد و یک حمله باینری را اجرا کرد. همیشه باید فرض کرد که یک متخاصم می‌تواند هرگونه رمزنگاری کد فراهم‌شده توسط سیستم‌عامل دستگاه را دور بزند.

- فرایندهای مدیریت کلید ضعیف

اگر کلیدها به‌درستی مدیریت نشوند بهترین الگوریتم‌ها هم نمی‌توانند امنیت را حفظ کنند. برخی خطاها در استفاده صحیح از الگوریتم رمزنگاری وجود دارد برای نمونه کلیدها در مکانی قابل‌دسترس ذخیره شوند و یا

کد نمودن کلیدها در باینری فراموش شود که موجب آسیب‌پذیری در برابر حمله باینری می‌شود.

- ایجاد و استفاده از پروتکل‌های رمزنگاری سفارشی

استفاده از پروتکل‌ها و الگوریتم‌های رمزنگاری سفارشی و خودساخته ساده‌ترین و بدترین روش رمزنگاری است. همیشه باید از پروتکل‌ها و الگوریتم‌های مورد تائید متخصصان استفاده نمود و در صورت امکان باید APIهای جدیدترین فناوری‌های رمزنگاری را به کار ببرد. یک متخاصم با حمله باینری می‌تواند کتابخانه‌های معمول رمزنگاری به همراه کلیدهای کد شده را بیابد. نیازمندی‌های امنیتی فراوان پیرامون رمزنگاری، استفاده از رمزنگاری جعبه سفید[1] را پراهمیت می‌کند. این فناوری خاص به‌گونه‌ای رمزنگاری را انجام می‌دهد که هیچ قسمتی از اطلاعات حساس مانند کلیدهای رمزنگاری فاش نشود.

- استفاده از الگوریتم‌های رمزنگاری ناامن

بسیاری از الگوریتم‌ها نقص‌های قابل‌توجهی دارند و یا همه نیازمندی‌های امنیتی جدید را برطرف نمی‌کنند.ازجمله : RC2، MD4، MD5، SHA1.

۷-۷ M7- تزریق در سمت مشتری

۷-۷-۱ عوامل تهدید

تزریق داده‌های غیرقابل اطمینان به برنامه‌ها از طریق کاربران خارجی، کاربران داخلی ، خود برنامه و دیگر برنامه‌های مخرب امکان‌پذیر است.

۷-۷-۲ نحوه حمله

متخاصم حملات متنی ساده‌ای را اجرا می‌کند که از نحو مفسر در برنامه هدف سوءاستفاده می‌کند و بردار تزریق تقریباً می‌تواند هر منبع داده‌ای شامل فایل یا خود برنامه باشد.

۷-۷-۳ علت ضعف امنیتی

تزریق در سمت مشتری موجب اجرای کد مخرب توسط یک برنامه در دستگاه موبایل می‌شود. این کد مخرب به‌عنوان داده‌ی ورودی به یک برنامه تزریق می‌شود و این داده مانند سایر داده‌ها توسط فریمورک

[1] White Box Cryptography

پشتیبان کننده برنامه پردازش می‌شود ولی در طول پردازش این داده خاص، وضعیت[1] برنامه تغییر داده می‌شود و فریمورک آن را به‌عنوان یک کد اجرایی تفسیر می‌نماید. این کد در بهترین حالت در همان محدوده و مجوزهای برنامه عمل می‌کند ولی در بدترین حالت می‌تواند با مجوزهای بالاتر و در محدوده بیشتری اجرا شود که آسیب آن بیشتر از حالت قبلی است.

روش دیگر ، تزریق باینری در یک حمله باینری است که این حمله می‌تواند حتی خطرناک‌تر از تزریق داده به برنامه عمل کند.

۷-۷-٤ تأثیرات فنی

برای تشخیص صحیح تأثیرات فنی یک برنامه باید مدل تهدید آن را ایجاد کنیم. در هنگامی‌که برنامه با بیش از یک کاربر در یک دستگاه یا با یک دستگاه اشتراکی یا با پرداخت برای محتوا سروکار داشته باشد حمله تزریق می‌تواند سخت باشد. نکته دیگر، تزریق برای سرریز مؤلفه‌های برنامه است که به دلیل کد مدیریت حفاظت از زبان برنامه، احتمالاً اثرات فنی کمتری دارد.

۷-۷-۵ تأثیرات تجاری

نتایج و اثرات تجاری این آسیب‌پذیری به ماهیت کد مخرب بستگی دارد. معمولاً این کدها اطلاعات حساس مانند رمزهای عبور، کوکی‌های نشست و اطلاعات شناسایی را می‌دزدند و ازاین‌رو اثرات تجاری کلاه‌برداری و نقض محرمانگی را در پی‌دارند.

۷-۷-٦ تشخیص آسیب‌پذیری

بهترین روش برای تشخیص ، شناسایی راه‌های ورود اطلاعات به برنامه و بررسی درستی داده‌های ارائه‌شده کاربر و برنامه‌ها است. سریع‌ترین و دقیق‌ترین روش برای اطمینان از کنترل صحیح برنامه بر داده‌ها، بررسی کد برنامه است. تحلیل گران امنیتی با کمک ابزارهای تحلیلی می‌توانند کاربرد مفسرها را پیدا کنند و همچنین جریان داده‌ها در برنامه را ردیابی نمایند. متخصصان آزمون نفوذ با سوءاستفاده ماهرانه از این آسیب‌پذیری‌ها، وجود این مشکلات را تائید می‌کنند.

به این دلیل که داده‌های ورودی یک برنامه از منابع زیادی می‌آیند؛ لیست کردن این منابع در مشخص کردن

[1] Context

هدف کارشان اهمیت دارد. به‌طورکلی حملات تزریق شامل انواع زیر است:

- تزریق SQL: پایگاه داده SQLite می‌تواند مانند برنامه‌های وب در معرض این حمله قرار گیرد. آسیب‌پذیری نسبت به این حمله و درنهایت مشاهده اطلاعات با این روش می‌تواند بسیار خطرناک باشد.
- فایل‌های محلی: اداره فایل‌ها بر روی دستگاه می‌تواند همان خطرات بالا را داشته باشد مگر این‌که این خواندن فایل‌ها فقط مربوط به برنامه‌ای باشد و فایل‌ها در مسیر آن برنامه ذخیره‌شده باشد.
- تزریق JavaScript(XSS) : مرورگرهای موبایل در معرض این حمله هستند و این مرورگرهای به کوکی‌های برنامه‌های موبایل نیز دسترسی دارند که می‌تواند منجر به سرقت نشست شود.
- رابط‌های کاربری برنامه‌ها و توابع می‌توانند داده‌هایی را بپذیرند که در یک آزمون فازینگ منجر به شکست برنامه می‌شود. البته به دلیل مکانیسم‌های حفاظتی سیستم‌عامل موبایل این نواقص نمی‌تواند منجر به سرریز شود بااین‌حال چند نمونه به‌عنوان آسیب‌پذیری "Userland" در زنجیره آسیب‌پذیری‌ها برای دستگاه‌های قفل‌شکسته یا روت شده وجود دارد.
- بدافزارها می‌توانند یک حمله باینری علیه لایه "نمایش" (html,css,javascript) یا علیه باینری اجرایی برنامه انجام دهند. این تزریق کد باینری یا با چارچوب برنامه موبایل یا در زمان اجرای برنامه انجام می‌شود.

۷-۷-۷ جلوگیری از آسیب‌پذیری

به‌طورکلی برای جلوگیری از تزریق کد به برنامه نیاز هست تا همه راه‌های ورودی برنامه را پیداکرده و برای آن‌ها نوعی اعتبارسنجی ورودی قرار داده شود.

در اندروید:

- هنگام سروکار داشتن با پرس‌وجوهای پویا و مؤلفه‌های "فراهم‌کننده محتوا" باید از پرس‌وجوهای پارامتری یا Prepared statement استفاده کرد.
- باید جاوا اسکریپت و پلاگین پشتیبان کننده آن به‌طور پیش‌فرض برای هر webview غیرفعال باشد.
- برای هر WebView (webview.getSettings().setAllowFileAccess(false);) دسترسی به سیستم فایل باید غیرفعال باشد.
- برای همه مؤلفه‌های "فعالیت" باید داده‌ها و اقدامات با یک فیلتر intent تائید اعتبار شود.

جلوگیری در تزریق باینری: در ادامه به‌طور کامل تشریح می‌شود.

۷-۷-۸ سناریوهای نمونه

اگر داده بازیابی شده از سرور برنامه شامل داده‌های مخرب باشد این داده‌ها به پایگاه داده محلی موبایل تزریق می‌شود که می‌تواند منجر به حمله تزریق SQL شود.

Intent های مخرب از دیگر برنامه‌ها می‌تواند منجر به سرریز بافر شود و درنتیجه کدهای مخرب اجرا شود.[1]

تغییرات Html از طریق بدافزارها یا دیگر برنامه‌ها می‌تواند منجر به اجرای کد جاوا اسکریپت مخرب در لایه نمایش شود.[2]

۷-۸ M8- تصمیم‌گیری‌های امنیتی بر اساس ورودی‌های نامعتبر

۷-۸-۱ عوامل تهدید

کاربران و بدافزارها و برنامه‌های آسیب‌پذیر می‌توانند داده‌های نامعتبر را به متدهای حساس ارسال کنند.

۷-۸-۲ نحوه حمله

سوءاستفاده از این آسیب‌پذیری آسان است؛ حمله‌کننده می‌تواند با دسترسی به یک برنامه فرمان‌ها را شنود کند و پارامترهای آن را تغییر دهد.

۷-۸-۳ علت ضعف امنیتی

معمولاً برنامه‌نویسان برای تمایز بین کاربران سطح بالا و پایین از پارامترها و مقادیر پنهان و عملکردهای پنهان استفاده می‌کنند. پیاده‌سازی ضعیف این عملکردها امکان شنود و تغییر آن‌ها توسط حمله‌کننده‌ها را در پی دارد که موجب رفتار نامناسب برنامه و حتی اعطای مجوزهای بیشتر به حمله‌کننده می‌شود.

[1] CrossApplication Scripting Attacks

[2] CrossSite Script Attacks

۷-۸-٤ تأثیرات فنی

این آسیب‌پذیری باعث ارتقای سطح دسترسی برای حمله‌کننده می‌شود و حتی می‌تواند مکانیسم‌های امنیتی پیاده‌سازی شده در برنامه را دور بزند که باعث از بین رفتن قابلیت اعتماد و صحت می‌شود.

۷-۸-۵ تأثیرات تجاری

این آسیب‌پذیری موجب از دست رفتن اعتبار می‌شود و همچنین صحت و قابلیت اعتماد را از بین می‌برد.

۷-۸-٦ تشخیص آسیب‌پذیری

به‌طورکلی برنامه‌ها می‌توانند داده‌ها را از منابع مختلف دریافت کنند و در بیشتر موارد از مکانیسم ارتباط بین فرایندی (IPC) برای دریافت داده استفاده می‌کنند.

به‌طورکلی به الگوهای طراحی IPC زیر باید پایبند بود:

- اگر نیازمندی‌های تجاری برای ارتباطات IPC وجود داشته باشد برنامه باید با ایجاد یک لیست سفید این ارتباطات را محدود کنند.
- تعامل کاربر برای انجام هرگونه اقدام حساسی که از طریق رابط‌های IPC فعال می‌شوند ضروری است.
- برای جلوگیری از حملات مبتنی بر ورودی باید ورودی‌های دریافت شده از رابط‌های IPC اعتبارسنجی شوند.
- به دلیل خطر شنود اطلاعات حساس توسط برنامه‌های دیگر، به‌هیچ‌وجه نباید این داده‌ها از طریق مکانیسم‌های IPC ارسال شوند.

۷-۸-۷ جلوگیری از آسیب‌پذیری

در اندروید:

هنگام استفاده از Activity هایی که تنها همان برنامه از آن استفاده میکند؛ خطر این وجود دارد که یک نرم افزار سوم در این بین بتواند یکintent که برای شروع Activity استفاده میشود را بخواند و آن Activity را با یک intent جعلی فراخوانی کند.

نکات ساخت یک Activity:

۱. taskAffinity را مشخص نکنید.

۲. lounchMode را مشخص نکنید.

۳. گزینه ی experted را false تنظیم کنید.

٤. intent دریافتی را با دقت و امنیت کنترل کنید حتی اگر آن intent از همان برنامه ارسال شده باشد. هنگام پردازش داده ی Intent دریافتی اولین کاری که باید بکنید بررسی اعتبار ورودی است .

۷-۹ M9- اداره نادرست نشست

۷-۹-۱ عوامل تهدید

هر کاربر یا برنامه‌ای می‌تواند به ترافیک شبکه و کوکی‌ها و ... دسترسی داشته باشد.

۷-۹-۲ نحوه حمله

یک متخاصم با دسترسی فیزیکی یا یک بدافزار می‌تواند ترافیک شبکه را ثبت کند.

۷-۹-۳ علت ضعف امنیتی

به‌منظور تسهیل در تراکنش پایدار بین برنامه و سرور پشتیبان ، برنامه‌ها از کوکی نشست استفاده می‌کنند که وضعیت را بر روی پروتکل‌های ناپایداری همچون HTTPS و SOAP حفظ می‌کند. برای حفظ وضعیت سرور پشتیبان پس از احراز هویت کاربر برنامه ، یک کوکی نشست برای برنامه ارسال می‌کند تا در ارتباطات بعدی و تراکنش‌های سرویس از این کوکی استفاده کند و این به سرور اجازه می‌دهد هر درخواست سرویس از سمت برنامه را به‌راحتی احراز هویت کند و مجوزهای را لازم را تجویز کند.اداره نادرست نشست زمانی اتفاق می‌افتد که توکن نشست با یک متخاصم به اشتراک گذارده شود.

۷-۹-٤ تأثیرات فنی

یک متخاصم با دسترسی به توکن های نشست‌ها می‌تواند هویت کاربر را جعل کند و با ارسال آن به سمت سرور پشتیبان، یک سرویس حساس را درخواست کند. بنابراین خطرات این آسیب‌پذیری به نوع کاربر جعل‌شده و سرویس‌های درخواستی آن بستگی دارد. متخاصم در بدترین حالت با جعل هویت کاربر مدیر می‌تواند قابلیت‌هایی حساسی را درخواست کند و در حالت معمولی کاربران کنترل حساب خود را از دست می‌دهند.

۷-۹-۵ تأثیرات تجاری

صدمات تجاری ناشی از سوءاستفاده از این آسیب‌پذیری می‌تواند شامل کلاه‌برداری، سرقت اطلاعات و توقف کسب‌وکار باشد.

۷-۹-۶ تشخیص آسیب‌پذیری

معمولاً نتایج این آسیب‌پذیری مشابه آسیب‌پذیری احراز هویت ضعیف است. کاربر برنامه فقط یک‌بار در طول یک نشست و آن‌هم در ابتدای آن احراز هویت می‌شود بنابراین کد برنامه باید به‌دقت از نشست‌های کاربر محافظت کند.

در زیر به نمونه‌هایی از نحوه اداره نامناسب یک نشست اشاره‌شده است:

- غفلت از باطل کردن نشست‌ها در سمت سرور: برنامه‌نویسان معمولاً نشست‌ها را در سمت موبایل ابطال می‌کنند و درحالی‌که در سمت سرور آن نشست برقرار است و این موجب سوءاستفاده متخاصمین از این موقعیت با کمک ابزارهای دست‌کاری HTTP می‌شود.
- عدم محافظت با مهلت زمانی مناسب: برنامه‌ها می‌توانند با تعیین مهلت زمانی برای نشست‌ها، از آن‌ها در برابر سوءاستفاده متخاصمین محافظت کنند بنابراین متخاصمین دیگر نمی‌توانند با دسترسی به یک نشست موجود بوده هویت کاربر آن نشست را جعل کنند. این مهلت زمانی با توجه به حساسیت برنامه و مشخصات خطر متعلق به آن برنامه و ماهیت ارتباطی کاربر با برنامه تعیین می‌شود معمولاً کاربران یک‌دفعه کارهای زیادی از برنامه‌های موبایلی خود می‌خواهند و به‌علاوه وقفه‌ها بین ارتباطات کاربر با برنامه زیاد است ازاین‌رو این‌ها پیش‌بینی مهلت زمانی را نسبت به برنامه‌های وب سخت‌تر می‌کند و تعیین مهلت زمانی طولانی‌تر خطر دزدیدن نشست را بیشتر می‌کند؛ بااین‌حال معمولاً این مهلت زمانی، زمان‌های ۱۵ دقیقه یا ۳۰ دقیقه یا یک ساعت است.
- غفلت از تعویض کوکی‌ها : در طول تغییرات وضعیت احراز هویت باید کوکی‌ها به‌طور مناسب باز تنظیم شوند. وقایعی موجب تغییر وضعیت احراز هویت می‌شوند که ازجمله تعویض از یک کاربر گمنام به یک کاربر واردشده یا تعویض از هر کاربر واردشده به کاربر واردشده دیگر یا تعویض از یک کاربر معمولی به یک کاربر ویژه یا مهلت‌های زمانی هستند. برای هرکدام از وقایع گفته‌شده نشست‌ها در سمت سرور باطل می‌شوند و دیگر کوکی‌های آن نشست‌ها نباید پذیرفته شود. در حالت ایدئال باید برنامه این‌گونه از کوکی‌ها را تشخیص دهد.

- ساخت توکن غیر ایمن: تولید توکن های مناسب سخت است. برنامه‌نویسان باید از الگوریتم‌های رمزنگاری و روش‌های استاندارد آزموده شده برای ایجاد توکن استفاده کنند به‌طوری‌که به دلیل پیچیده و طولانی و شبه تصادفی بودن آن قابل حدس زدن نباشد.

۷-۹-۷ جلوگیری از آسیب‌پذیری

باید مطمئن شد که برنامه در طول چرخه حیات یک نشست متعلق به آن برنامه توکن های نشست را به‌طور مناسب ایجاد، حفظ و ابطال می‌کند.

۷-۱۰ M10- نبود حفاظت‌های باینری

۷-۱۰-۱ عوامل تهدید

معمولاً یک متخاصم کد برنامه را تجزیه‌وتحلیل و مهندسی معکوس می‌کند و سپس با انجام تغییراتی در آن موجب اجرای برخی قابلیت‌های پنهان می‌شود.

۷-۱۰-۲ نحوه حمله

تجزیه‌وتحلیل و مهندسی معکوس معمولاً توسط ابزارهای خودکار انجام می‌گردد.

۷-۱۰-۳ علت ضعف امنیتی

عدم محافظت باینری می‌تواند برنامه و دارنده آن را در معرض خطرات فنی و تجاری بسیاری قرار دهد. البته یک برنامه با محافظت باینری نیز می‌تواند مهندسی معکوس شود و در معرض خطر باشد ولی این محافظت باینری روند عملیات را کند می‌کند. معمولاً برنامه‌ها بدون محافظت باینری ایجادشده‌اند. تشخیص مهندسی معکوس کد یک برنامه توسط متخاصم دشوار است و معمولاً زمانی مالک برنامه می‌فهمد که کد برنامه‌اش در برنامه دیگر به‌کاررفته باشد و این نحوه تشخیص بسیار تصادفی است. همچنین برنامه باید در صورت بروز تغییرات و تزریق در زمان اجرا و پاسخ تشخیص دهد و با روش‌های مختلف واکنش نشان دهد که این واکنش‌های از پیش تعریف‌شده می‌تواند یا تلاش برای خنثی کردن حمله یا شکست حمله با روش ماهرانه باشد.

۷-۱۰-٤ تأثیرات فنی

عمده برنامه‌های موبایل از حفاظت باینری در برابر مهندسی معکوس و تغییرات در کد باینری بی‌بهره‌اند و توسعه‌دهندگان باید برای مقابله با این موارد حفاظت باینری را در برنامه‌ها بگنجانند.

این نوع محافظت می‌تواند روند مهندسی معکوس را دچار تأخیر کند ولی نمی‌تواند از انجام این حمله جلوگیری کند. در بیشتر موارد متخاصم کد برنامه را می‌دزدد و درحالی‌که مالک برنامه بی‌خبر است در یک برنامه دیگر به کار می‌برد و آن برنامه را در اَپ استورها به فروش می‌گذارد.

همچنین این حفاظت می‌تواند روند تغییر کد باینری برنامه برای اجرا یا غیرفعال کردن قابلیت‌های برنامه را کند. معمولاً این تغییرات در برنامه‌ای محتمل است که اطلاعات حساس مانند رمزها و کارت‌های اعتباری را ذخیره و پردازش می‌کنند. تغییر کد معمولاً نوعی از بازبسته بندی یا انضمام بدافزار به برنامه است.

۷-۱۰-۵ تأثیرات تجاری

نبود حفاظت باینری موجب برخی نتایج درزمینهٔ تجارت می‌شود که ازجمله : سرقت اطلاعات محرمانه ، کلاه‌برداری و دسترسی غیرمجاز ، صدمه به اعتبار، دزدی و از دست رفتن درآمد، سرقت مالکیت معنوی

۷-۱۰-٦ تشخیص آسیب‌پذیری

درصورتی‌که کد برنامه در یک محیط غیرقابل‌اعتماد میزبانی شود، آن برنامه در معرض خطر است. محیط غیرقابل‌اعتماد محیطی است که سازمان توسعه‌دهنده، دسترسی فیزیکی به آن نداشته باشد مانند: موبایل‌ها، ابرها، مراکز داده و ... .

در موارد زیر می‌توان به آسیب‌پذیر بودن برنامه را پی برد:

- درصورتی‌که بتوان یک برنامه را با ابزارهای آزاد رمزگشایی کرد برای نمونه با ابزارهای ClutchMod یا به‌طور دستی با GDB می‌توان برنامه‌های IPhone را رمزگشایی کرد.
- اگر با کمک ابزارهایی نظیر IDA Pro و Hopper بتوان کنترل جریان برنامه را ترسیم نمود و کلیه برنامه را استخراج کرد.
- اگر بتوان لایه ارائه برنامه (html,css,..) در داخل تلفن تغییر داد و جاوا اسکریپت دلخواه را اجرا نمود.

- اگر با کمک یک ابزار ویرایش hex بتوان کد باینری برنامه را تغییر داد و کنترل‌های امنیتی را دور زد.

۷-۱۰-۷ جلوگیری از آسیب‌پذیری

برای جلوگیری باید از مؤلفه‌های امنیتی برای بررسی موارد زیر در برنامه استفاده شود:

- کنترل تشخیص شکسته شدن قفل سیستم‌عامل موبایل
- کنترل checksum
- کنترل‌های گواهی‌نامه
- کنترل تشخیص عیب‌یابی

برنامه باید بتواند با مؤلفه‌های بالا دو خطر عمده را کاهش دهد:

- جلوگیری از تجزیه‌وتحلیل و مهندسی معکوس برنامه با کمک روش‌های تجزیه‌وتحلیل ایستا و پویا توسط متخاصم
- تشخیص تغییرات کد و یا کدهای اضافه‌شده در زمان اجرا و واکنش نشان دادن نسبت به این نقض صحت کد.

۷-۱۰-۸ سناریوهای نمونه

در اینجا به برخی آسیب‌پذیری‌ها مربوط به عدم محافظت باینری در برنامه‌ها و ابزارهای تشخیص آن‌ها اشاره‌شده است:

در اندروید: تبدیل بایت کد(apktool; dex2jar)، تجزیه‌وتحلیل در زمان اجرا(ADB) ، مهندسی معکوس(IDA Pro; Hopper) ، تبدیل به اسمبلی (baksmali))، تزریق کد(Mobile Substrate)