HOW NOT TO BUILD ANDROID APPS

Jack Mannino

Jack@nvisiumsecurity.com
http://twitter.com/jack_mannino
Overview

- Are Things REALLY That Bad?
- Android Crash Course
- App Smackdown
- Q&A
Company co-founder

https://www.nvisiumsecurity.com

OWASP Mobile Security Project Leader

LIKES: mobile security, baseball

DISLIKES: people who can’t drive

For more information, see:

http://lmgtfy.com/?q=jack+mannino
Are Things REALLY That Bad?

- Depends on *who* you ask

[Image of a monkey cartoon and a cat with a crown reading, "IT'S A CONSPIRACY!"
LOLCATS.COM]
Are Things REALLY That Bad?

- Android is “open’
- Ridiculous amount of malware
- Ecosystem is very fragmented
- Lots of W.T.F. in apps
Are Things REALLY That Bad?

- Ice Cream Sandwich (4.0) solves some problems
- FINALLY adds Address Space Layout Randomization (ASLR)
- Full device encryption, keychain API
“One More Thing”

Unless you root….

Or get a new device….

Or have a relatively new device….

You probably won’t get an update.

= 😞
### Fragmentation!

**ANDROID & iPHONE UPDATE HISTORY**

Includes every iPhone & Android model released in the US before July 2010.

Data as of the end of October 2011.

<table>
<thead>
<tr>
<th>Phone</th>
<th>1st Year After Release</th>
<th>2nd Year After Release</th>
<th>3rd Year After Release</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2007 iPhone</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple iPhone</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6/29/07 AT&amp;T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple iPhone 3G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3/15/08 AT&amp;T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple iPhone 3GS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/19/09 AT&amp;T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple iPhone 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6/28/10 AT&amp;T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2008 Android</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC G1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11/22/08 T-Mobile</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2009 Android</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC MyTouch 3G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6/5/09 T-Mobile</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC Hero</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12/11/09 Sprint</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Samsung Moment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11/19/09 Sprint</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motorola Cliq</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11/3/09 T-Mobile</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motorola Droid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11/5/09 Verizon</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2010 DECA Android</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC Nexus One</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/23/10 T-Mobile</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motorola Devour</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3/23/10 Verizon</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motorola Backflip</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3/7/10 AT&amp;T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motorola Cliq XT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3/12/10 T-Mobile</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC Droid Incredible</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/29/10 Verizon</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG Ally</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5/24/10 Verizon</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC MyTouch 3G Slide</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6/5/10 T-Mobile</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC Evo 4G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6/4/10 Sprint</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Garmin iQue Slide</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6/8/10 T-Mobile</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC Aria</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6/28/10 AT&amp;T</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* After $100 mail-in rebate ($50 on MyTouch 3G Slide & Garmin iQue)
Does This Affect Development?

- Yeah! Developers have 3 options:
  1. Limit their potential users by requiring the latest and greatest OS
  2. Distribute multiple versions of an app
  3. Ignore those fancy new security features (*most likely*)
Android Crash Course

- Dalvik VM ≠ security
- Security model is largely user driven
- Applications are granted permissions during installation
- Apps are self-signed (very different than iOS)
## Top Mobile Risks

### OWASP Mobile Top 10 Risks

<table>
<thead>
<tr>
<th>M1 - Insecure Data Storage</th>
<th>M6 - Improper Session Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2 - Weak Server Side Controls</td>
<td>M7 - Security Decisions Via Untrusted Inputs</td>
</tr>
<tr>
<td>M3 - Insufficient Transport Layer Protection</td>
<td>M8 - Side Channel Data Leakage</td>
</tr>
<tr>
<td>M4 - Client Side Injection</td>
<td>M9 - Broken Cryptography</td>
</tr>
<tr>
<td>M5 - Poor Authorization and Authentication</td>
<td>M10 - Sensitive Information Disclosure</td>
</tr>
</tbody>
</table>

*Source: nVisium SECURITY*
Android Architecture
Android App Anatomy

- APK = ZIP format
- classes.dex = app binary
- AndroidManifest.xml = configuration

```
$ ls
AndroidManifest.xml
Facebook for Android.apk
META-INF
assets
classes.dex
org
res
resources.arsc
```
AndroidManifest.xml

- First thing you should look at
- This is where components and permissions are declared
How components talk to each other

Explicit or Implicit

Intent Filters match actions, data type, categories, schemes, etc.

Filters DO NOT provide security the way you’d think
Intents

```java
Intent intent = new Intent(DoCheckin.this,
    ViewCheckin.class);
Bundle bundle = new Bundle();
bundle.putString("checkinID", checkinInfo.get("checkinID"));
bundle.putString("venueName", checkinInfo.get("venueName"));
bundle.putString("venueWebsite",
    checkinInfo.get("venueWebsite"));
bundle.putString("dateTime", checkinInfo.get("dateTime"));
bundle.putString("latitude", latitude);
bundle.putString("longitude", longitude);
intent.putExtras(bundle);
CheckinDBHelper db = new CheckinDBHelper(context);
checkinInfo.put("latitude", latitude);
checkinInfo.put("longitude", longitude);
db.insertCheckin(checkinInfo);
db.close();
startActivity(intent);
```
Components

- Activities
- Broadcast Receivers
- Content Providers (won’t cover it)
- Services (won’t cover it)
Activities

- Single screen with a UI
- Life cycle includes `onCreate` -> `onDestroy`
Broadcast Receivers

- Receive intents sent with `sendBroadcast(intent)` or `sendOrderedBroadcast(intent)`

- Can require a specific permission by the sender

- However, not perfect
Now, Let’s Have Fun

- A few real-world examples
- Sanitized real-world examples, to protect the “innocent”
Droid Commander

- Allows you to remotely control a device via SMS
- Interesting permissions
- BIND_DEVICE_ADMIN
- READ_SMS
Droid Commander

Good Afternoon,

Your application has significant security issues. For starters, the following:

1) Weak pin numbers can be selected.

2) The pin can be brute forced via SMS, with no resulting lockouts.

3) You are using the Device ID + pin as the authenticator for the web application side. The Device ID is a compromised value, as this is sent pretty much everywhere. Hence, you are only offering a single factor of authentication.

I'd be happy to elaborate further on these issues, and provide you with proof of concept exploits demonstrating how someone could attack users.

Jack A. Mannino
CEO/Founder
nVisium Security Inc.
Hi Jack,

Thanks for sharing your thoughts and taking a look at Droid Commander android app from the security angle. I am aware of most of the issues you raised. Since I am still working towards the final version, I do have plans to tighten the security aspects on the web server. For now, it is still a work in progress so that's why it is offered for free.

As far as PIN no. is up to the user how big or elaborate the PIN could be. Final version will also accept characters as part of the PIN so it will be more difficult to brute forced.

I'm not sure which version did you try. But the latest version 3.1.20 keeps the SMS commands hidden so there is no way to know what the PIN is after receiving commands.

I'll let you know once the final version is available and we could revisit your points again.

BTW your website looks very good.

Best Regards,

David
Heh...”Handy”
Abuse Cases

- Remotely trigger phone calls to a number of your choice
- Eavesdropping
- Take pictures, record calls
- **REMTELY WIPE A DEVICE**
We Need a Pin

Agent Disabled - Configure your pin first in Settings tab
Sync PIN with your web portal
false
false

Agent Not Available - Must exit and re-login after setting your pin for the first time
Suggestions

- 1
- 11
- 666
- 14
How It Works

- Send an SMS message to the device
- Syntax: pin:command:data
  - 666:Call:9765551234
  - 666:wipe:omglulz
Lessons Learned

- Don’t use this app
- SMS is a terrible way to enable single factor authentication
- Weak authentication pin makes it worse
- Malicious apps can abuse this too
A Russian app that aggregates multiple Russian banking services
public Cursor getAllRowsCursor()
{
    SQLiteDatabase localSQLiteDatabase = mBankDBAdapter.this.db;
    String[] arrayOfString = new String[13];
    arrayOfString[0] = "_id";
    arrayOfString[1] = "id_bank";
    arrayOfString[2] = "bank";
    arrayOfString[3] = "caption";
    arrayOfString[4] = "card_icon";
    arrayOfString[5] = "card_type";
    arrayOfString[6] = "card_number";
    arrayOfString[7] = "serv_number";
    arrayOfString[8] = "balances";
    arrayOfString[9] = "valut";
    arrayOfString[10] = "ki";
    arrayOfString[11] = "is_payee";
    arrayOfString[12] = "cr_limit";
    return localSQLiteDatabase.query("cards", arrayOfString, null, null, null, null, null);
}
Can It Get Worse?

```java
public void backupDb(Activity paramActivity) {
    this.activityContext = paramActivity;
    if (Util.isExternalStorageAvail())
        new ExportDatabaseFileTask(null).execute(new Void[0]);
    while (true)
    {
        return;
        Toast.makeText(paramActivity, 2131165438, 0).show();
    }
}
private class ExportDatabaseFileTask extends AsyncTask<Void, Void, Boolean>
{
    private final ProgressDialog dialog = new ProgressDialog(Mbank26Adapter.this.activityContext);

    private ExportDatabaseFileTask()
    {
    }

    protected Boolean doInBackground(Void[] params)
    {
        String str = Mbank26Adapter.this.context.getPackageName();
        File localFile1 = new File(Environment.getDataDirectory() + "/data/" + str + "/databases/" + "mobileBank.db");
        File localFile2 = new File(Mbank26Adapter.PATH_TO_BACKUP);
        if (!localFile2.exists())
        {
            localFile2.mkdirs();
            localFile2 = new File(localFile2, localFile1.getName());
            try
            {
                localFile2.createNewFile();
                Utility.copyFile(localFile1, localFile2);
                Boolean localBoolean2 = Boolean.valueOf(true);
                localBoolean1 = localBoolean2;
            }
        }
        return localBoolean1;
    }
public class MBankDBAdapter{

private static final String DATABASE_NAME = "mobileBank.db";
private static final int DATABASE_VERSION = 7;
private static final int NO_SELECT_ITEM = -1;
private static final String PATH_TO_BACKUP = Const.PATH_TO_CARD + "/backup";
private Activity activityContext;
private final Context context;
private SQLiteDatabase db;
private MBankDBsceHelper dbHelper;
public TableCards tableCards;
public TableGroups tableGroups;
public TableJournal tableJournal;
public TableLogSMS tableLogSMS;
public TableTemplates tableTemplates;

Will this end well?
Tip: Don’t Do This

```java
public final class Const
{

    public static final boolean DEBUG = false;
    public static final int DEF_INTERNAL_FILTER_JOURNAL = 60;
    public static final String LAST_UPDATE_IDCARD = "LAST_UPDATE_IDCARD";
    public static final int MY_REQUESTCODE_ADD_TO_JOURNAL = 32006;
    public static final int MY_REQUESTCODE_CONTACT_TEL = 32002;
    public static final int MY_REQUESTCODE_SELECT_CARD = 32003;
    public static final int MY_REQUESTCODE_SET_GPS_ON = 32005;
    public static final int MY_REQUESTCODE_SET_OPERATION_LOCATION = 32004;
    public static final int MY_REQUESTCODE_SET_USES_BANKS = 32007;
    public static final int MY_REQUESTCODE_SHOW_LOGO = 32001;
    public static final int MY_REQUESTCODE_SHOW_PATH_TO_CARD = 32000;
    public static final String PATH_TO_CARD = "SMSBanking";
    public static final int TRACE;
    public static final SimpleDateFormat format_date;
    public static final DecimalFormat format_money = new DecimalFormat("###,###0.00");

    static
    {
        format_date = new SimpleDateFormat("dd.MM.yy HH:mm:ss")
        PATH_TO_CARD = Environment.getExternalStorageDirectory() + "/SMSBanking";
    }
}
```
Lessons Learned

- Storing sensitive stuff in the clear is bad
- Backing it up to external storage is even worse
- Other apps can access this
- Lose phone, game over
Device ID Authentication

- Requires READ_PHONE_STATE permission
- Tons of apps have this permission
- Ad networks track this stuff too
Device ID Authentication

- Some apps that have this permission:
  - 3D Sexy Girls
  - Adult Sexy Wallpapers
  - Asian Sexy Girl
  - Hot Sex Tips
  - Male and Female Sex Movies
Device ID Authentication

- Let’s assume that you trust the makers of those apps
- Do you trust their ability to not get owned?
Device ID Authentication

- Demo of a weak implementation
- Pulled from the OWASP GoatDroid Project (I wrote it)
- GoatDroid is a playground for learning about Android security
Authentication Tips

- Never use device ID, IMEI, IMSI, etc as sole authenticators
- Out of band measures fail on a single device, for the most part
- Contextual information (ie- location) can help, but can be spoofed
XSS On Steroids

- XSS...you know, it makes pretty alert boxes
- Arbitrary script execution is fun
- Hybrid native/web apps
Android allows Java code to be called via WebViews using JavaScript

Powerful, but dangerous

```java
webview.getSettings().setJavaScriptEnabled(true);
webview.addJavascriptInterface(new SmsJSInterface(this), "smsJSInterface");
```
XSS On Steroids

Another demo
Do’s and Don’ts

- Do: contextually output encode untrusted data within a WebView

- Don’t: tie sensitive actions to untrusted data within JavaScript interfaces
Activity Lifecycle Badness

- Other apps can start your app’s activities
- Permitted when you export an Activity
- Also permitted if your Activity uses Intent Filters
Activity Lifecycle Badness

- Scenario: your Activity executes something within the `onCreate()` method

- It also receives data within a Bundle

- Demo time
Activity Lifecycle Badness

- Another thing you shouldn’t do: 
  depend on onDestroy() to execute

- NOT GUARANTEED TO BE CALLED

- Often used to “clean up” sensitive data

- Solution: don’t do this
Nothing I presented today requires a prohibitively high skill level.

These types of issues are extremely common within real-world apps.

Common sense and judgement are equally as important as technical stuff.
Thanks For Listening!

Follow my daily rants @
http://twitter.com/jack_mannino

Check out the OWASP Mobile Security Project