Smart Phones Dumb Apps

Trinity University

Monday February 14th, 2011
Agenda

• Generic Smartphone Threat Model
• Sample Application
• What an Attacker Sees (Android Edition)
• What About iPhones/iPads?
• Closing Thoughts
• Questions
Smart Phones, Dumb Apps

• Lots of media focus on device and platform security
  – *Important because successful attacks give tremendous attacker leverage*

• Most organizations:
  – *Accept realities of device and platform security*
  – *Concerned about the security of their custom applications*
  – *Concerned about sensitive data on the device because of their apps*
  – *Concerned about network-available resources that support their apps*

• Who has smartphone application deployed for customers?

• Who has had smartphone applications deployed without their knowledge?
  – *$!%#$ marketing department…*
Generic Smartphone Threat Model
Some Assumptions for Developers

- Smartphone applications are essentially thick-client applications
  - That people carry in their pockets
  - And drop in toilets
  - And put on eBay when the new iPhone comes out
  - And leave on airplanes
  - And so on…

- Attackers will be able to access:
  - Target user (victim) devices
  - Your application binaries

- What else should you assume they know or will find out?
A Sample Application

- Attach to your brokerage account
- Pull stock quotes
- Make stock purchases

- (Apologies to anyone with any sense of UI design)

- This is intentionally nasty, but is it unrealistic?
So What Does a Bad Guy See? (Android Edition)

• Install the application onto a device
• Root the device
• Pull the application’s APK file onto a workstation for analysis

• APK files are ZIP files
• They contain:
  – AndroidManifest.xml
  – Other binary XML files in res/
  – classes.dex DEX binary code
What’s Up With My XML Files?

- Binary encoding

- Use axml2xml.pl to convert them to text

http://code.google.com/p/android-random/downloads/detail?name=axml2xml.pl
Much Better

- Now we see:
  - Screens in application
  - Permissions required by the application
Do the Same Thing With the Rest of Them

- Recurse through the res/ subdirectory
- UI layouts, other resources
What About the Code?

• All of it is stuffed in classes.dex

• Android phones use DEX rather than Java bytecodes
  – Register-based virtual machine rather than stack-based virtual machine

• Options:
  – Look at DEX assembly via de-dexing
  – Convert to Java bytecode and then to Java source code
De-Dex to See DEX Assembly

- DEX bytecode \(\approx\) Java bytecode
- All code goes in one file
- Disassemble to DEX assembly with dedexer

http://dedexer.sourceforge.net/
Lots of Information

• Like the fun-fun world of Java disassembly and decompilation
  – *(We’ll get to the DEX decompilation in a moment)*

• LOTS of information available
But Can I Decompile to Java?

- Yes
- We
- Can

• Convert to Java bytecodes with dex2jar
• Convert to Java source code with your favorite Java decompiler
DEX Assembly Versus Java Source Code

- De-DEXing works pretty reliably
- DEX assembly is easy to parse with grep
- DEX assembly is reasonably easy to manually analyze
- Java decompilation works most of the time
- Java source code can be tricky to parse with grep
- Java source code is very easy to manually analyze

Verdict:
- Do both!
- Grep through DEX assembly
- Analyze Java source
So What Did We Learn?

- Look at the string constants
  - URLs, hostnames, web paths
- Look at the de-DEXed assembly
  - Method calls
  - Data flow
- Developers: BAD NEWS
  - The bad guys have all your code
  - They might understand your app better than you
Is There Sensitive Data On the Device?

- Look at the code
- Grep for “File”
What About Java Source Code?

- Get the source code with JD-Gui
  - [http://java.decompiler.free.fr/](http://java.decompiler.free.fr/)
Look for Files With Bad Permissions

- Look for file open operations using
  - `Context.MODE_WORLD_READABLE`
  - (translates to “1”)

```java
try {
    String str4;
    FileOutputStream localFileOutputStream = getApplicationContext().openFileOutput("SecretFile.txt", 1);
    OutputStreamWriter localOutputStreamWriter = new OutputStreamWriter(localFileOutputStream);
    localObject = localStringBuilder.toString();
    localOutputStreamWriter.write((String)localObject);
    localOutputStreamWriter.close();
    String str5;
    for (localObject - 1; localObject - str5) {
        return localObject;
        localStringBuilder.append(str4);
        break;
        localException1 = localException1;
        localObject = new StringBuilder("Error while retrieving URL ").append(str1).append(" ");
        str5 = localException1.getMessage();
    }
}
```
Next: What Is On the Server-Side

- To access sensitive data on a device:
  - *Steal a device*
  - *Want more data?*
  - *Steal another device*

- To access sensitive data from web services
  - *Attack the web service*

- String constants for URLs, hostnames, paths

- Examples:
  - *3rd party web services*
  - *Enterprise web services*
So Now What?

- **3rd Party Web Services**
  - *Is data being treated as untrusted?*

- **Enterprise Web Services**
  - *Did you know these were deployed?*
Web Services Example

- Trumped up example, but based on real life
- Given a web services endpoint, what will a bad guy do?
What Is Wrong With the Example Application?

- Sensitive data stored on the device
- Trusts data from 3rd party web services
- Exposes enterprise web services
- Enterprise web services vulnerable to XSS attacks
- And so on…
What About iPhones/iPads?

• Objective-C compiled to ARMv6 machine code
  – Not as fun as Java compiled to DEX bytecode

• Apps from iTunes Store
  – Encrypted
  – Used to be “easy” (well, mechanical) to break encryption with a jailbroken phone and a debugger
  – Now trickier
  – But the default apps are not encrypted…
Run “strings” on the Binary

- Web services endpoints: URLs, hostnames, paths
- Objective-C calling conventions:

  \[\text{myThing doStuff a b c}]\;

  becomes

  \text{obj_msgsend(myThing, "doStuff:", a, b, c);}
Run “otool” on the Binary

- `otool -l <MyApp>`
  - View the load commands
  - Segment info, encryption info, libraries in use

- `otool -t -v <MyApp>`
  - Disassemble the text segment to ARMv6 assembly
  - If run on an encrypted application you get garbage

- And so on…
iPhone/iPad URL Schemes

- iPhone applications can be set up to “handle” certain URL schemes
- Defined in the application’s Info.plist
- Binary format: annoying
Decoding Files: Easy for iPhones/iPads Too

- `plutil -convert xml1 Info.plist`
- **Much nicer**
- **XPath: Look for:**
  
  
  /plist/dict/array/dict[key='CFBundleURLSchemes']/array/string

- **Now you know the URL Schemes the app handles**
Net Result for iPhone/iPad

• More obscure
  – *But does that mean more secure?*

• Can still retrieve a tremendous amount of information
So What Should Developers Do?

• Threat model your smartphone applications
  – More complicated architectures -> more opportunities for problems

• Watch what you store on the device
  – May have PCI, HIPAA implications

• Be careful consuming 3rd party services
  – Who do you love? Who do you trust?

• Be careful deploying enterprise web services
  – Very attractive target for bad guys
  – Often deployed “under the radar”
So What Should Security People Do?

• Find out about smartphone projects
  – *Not always done by your usual development teams*
  – *R&D, “Office of the CTO,” Marketing*

• Assess the security implications of smartphone applications
  – *What data is stored on the device?*
  – *What services are you consuming?*
  – *Are new enterprise services being deployed to support the application?*
Resources

• axml2xml.pl (Convert Android XML files to normal XML)
• Dedexer (Convert DEX bytecodes into DEX assembler)
  – http://dedexer.sourceforge.net/
• Dex2jar (Convert DEX bytecode in Java bytecode)
  – http://code.google.com/p/dex2jar/
• JD-GUI (Convert Java bytecode to Java source code)
  – http://java.decompiler.free.fr/
• otool (Get information about iPhone binaries)
Online

- Code, slides and video online: 

  www.smartphonesdumbapps.com
Questions?

Dan Cornell
dan@denimgroup.com
Twitter: @danielcornell

www.denimgroup.com
(210) 572-4400