What’s Old Is New Again: An Overview Of Mobile Application Security

Zach Lanier
Mike Zusman
Agenda

- Who are we?
  - Mike Zusman
  - Zach Lanier
- What do we do?
  - Black box Mobile App Assessments...
  - Mobile Application Themes
    - Broad Observations
- Platform Security
  - Quick Overview of Common Platforms
  - Testing Techniques
  - Dynamic vs Static, current challenges
- Case Studies
  - What we’ve seen, and how we’ve found it
Other than carbon-based multi-cellular life forms
How susceptible is your organization to phishing attacks?

Using PhishMe.com, you can emulate real phishing attacks against your organization within minutes. Focus your training on the most susceptible by providing immediate feedback to anyone that falls victim. Learn More

Our prices are insane!
Broad Observations
The Web pushed content to the browser
- Centralization of apps & data
- Always a push for MORE (ActiveX, applets, flash)

Now, everyone gets their own app!
- Code (not HTML) gets pushed to the endpoint
- XKCD Viewer
AuthC/AuthZ

- **Carrier Applications**
  - “we trust you because you’re on our network”

- **Third-party Applications**
  - SOMETIMES better than carrier apps
  - Incomplete support of open standards

- **Client-side data trust issues**
  - admin=1
Broad Observations

- HyperGlobalMegaCloudDataMeshStore
  - Many Apps for syncing data between device and CLOUD
  - Full AuthC, AuthZ bugs
Platform Security
Platform Security

- Many disparate platforms
  - Android, RIM, iPhoneOS, WinMo, Brew, Symbian, WebOS...

- Controls
  - Network Hardening
  - Application Sandboxing
  - Managed Code
  - Code Signing
- Concerns
  - Shared User Accounts
  - Native Code (Obj-C, JNI)
  - Certificate Validation
    - SSL, Code Signing
  - Support for Emerging Technologies
    - Flash, WebKit, HTML5
Testing Techniques
Testing Techniques

- White Box Source Code Review
  - Sometimes, it’s trivial to get app source code
- Black Box
  - Acquiring Application Binaries
  - Reverse Engineering
    - Disassembly/Decompilation
  - Network Analysis
    - Protocol Analysis, fuzzing
    - MITM
Testing Techniques (cont’d)

- undx, dex2jar, coddec, JAD
  - decompilation
- Smali / baksmali
  - (dis)assembly, patching
- Native Code?
  - IDA with ARM support
  - Strings!
Testing Techniques (cont’d)

```
oishi$ adb pull /system/app/com.amazon.mp3.apk
1241 KB/s (552871 bytes in 0.434s)
oishi$ unzip com.amazon.mp3.apk
Archive: com.amazon.mp3.apk
  inflating: META-INF/MANIFEST.MF
  inflating: META-INF/MP3TEAMS.SF
  inflating: META-INF/MP3TEAMS.RSA
  inflating: res/drawable/album_detail_info_background.xml
  inflating: res/drawable/album_track_toggle_active_background.xml
  inflating: res/drawable/album_track_toggle_inactive_background.xml
  extracting: res/drawable/artwork_placeholder.png
  extracting: res/drawable/artwork_placeholder_small.png
  extracting: res/drawable/buy_button_hot_opaque.png
```
Testing Techniques (cont’d)

```java
# direct methods
.method static constructor <clinit>()V
    .locals 1
    .prologue
    .const/4 v0, 6x6
    .line 35
    sput-object v0, Lcom/amazon/mp3/net/RestClient;->sSocketFactoryFallback:Ljava/org/apache/http/conn/ssl/SSLSocketFactory;
    .line 36
    sput-object v0, Lcom/amazon/mp3/net/RestClient;->sSocketFactory:Lorg/apache/http/conn/ssl/SSLSocketFactory;
    .line 34
    return-void
.end method

.method public constructor <init>(Ljava/io/InputStream;Ljava/lang/String;)V
    .locals 5
    .parameter "keyStoreStream"
    .parameter "keyStorePassword"
```
public class RestClient {
    public RestClient(InputStream arg0, String arg1) {
        /*<invalid signature>*//*java.lang.Object local = com/amazon/mp3/net/RestClient;
        local;
        JVM INSTR monitorenter;
        Object obj1 = sSocketFactoryFallback;
        if(obj1 == null) goto _L2; else goto _L1
        _L1:
        obj1 = sSocketFactory;
        if(obj1 == null) goto _L2; else goto _L3
        _L3:
        return;
        _L2:
        Object obj = null;
        obj1 = KeyStore.getDefaultType();
        KeyStore keystore = KeyStore.getInstance(((String) {obj1}));
        char ac[] = arg1.toCharArray();
        keystore.load(arg0, ac);
        ac = JVM INSTR new #52 <Class SSLSocketFactory>;
        ac.SSLSocketFactory(keystore, arg1, keystore);
        sSocketFactoryFallback = ac;
        ac = JVM INSTR new #52 <Class SSLSocketFactory>;
        KeyStore keystore1 = null;
        ac.SSLSocketFactory(keystore, arg1, keystore1);
        sSocketFactory = ac;
        IoUtility.close(arg0);
Not everyone can be a Binary RE ninja
  • ...and project timelines don’t allow for on-the-job training :-)
• Sometimes the easiest way to understand an application is to look at its TRAFFIC
• You need to be come the MITM
  • Just like WAPT, and Burp, WebScarab, etc.
Testing Techniques (cont’d)

- MAPT MITM Challenges!
  - Run the app in an emulator (boring)
  - Connect the phone to your own WAP
    - Uplink your WAP to your laptop with Internet sharing enabled
    - Run Wireshark
  - WiFi not always an option
    - Handset might not support WiFi
    - Application might require carrier network
      - Modification of Application
        - Change server.carrier.com to testsite.com
MAPT MITM Challenges!

- WireShark lets you see traffic
  - SYN TCP 80? Easy.
  - SYN TCP 443? A little harder.
  - SYN TCP 9999? Ok...
    - Binary data?! Huh?
  - UDP DST Port 4717?!?
    - I quit!
Case Studies
Case Study: Foursquared

- Foursquare client for Android
- Originally written in Java, like most Android applications
  - Source available under Apache 2.0 license
Case Study: Foursquared

- Foursquare API supports Basic Auth and OAuth...
  - OAuth includes signatures for transactions, helps prevent replay attacks, etc.
  - Guess which one foursquared uses
Case Study: Foursquared

- That’s right. HTTP Basic Auth...over plaintext transport

14:54:35.510013 IP (tos 0x0, ttl 64, id 38148, offset 0, flags [DF], proto TCP (6), length 250) 25.33.4073.200000 win 2920
E..#####.r##!#Xx..l...P.#.#####.P..h7##.GET /v1/user?mayor=0&badges=0 HTTP/1.1
User-Agent: com.joelapenna.foursquared 2010011401
Host: api.foursquare.com
Connection: Keep-Alive
Authorization: Basic ZXZpbHNxWFrUdBaK6oZXJLJm9yZzpnb29kdmlzaW9u

- There’s a CWE for that!
  - CWE-311: Missing Encryption of Sensitive Data (including credentials) 😊
Case Study: Foursquared

- Why is this a problem?
  - EVERYONE uses Foursquare
    - Well, maybe not you, but everyone else!
  - Most applications “prefer” WiFi to cell radio
    => trivial interception of creds
- Funny enough, Foursquared has OAuth support
  - But it’s not actually used
Case Study: Storage Application

- Multi-platform application for storing and retrieving music, videos, documents, and more
  - Android, BREW, Blackberry, and fat web browser
- Developed by third party, branded for a major carrier
- Proprietary, binary-only
Case Study: Storage Application

- Simple crash in storage quota viewer
  - Divide-by-zero error leads to DoS
  - Attacker must successfully intercept and modify server response for this to happen
    - A bit more difficult since this tends to occur over the carrier’s network, but WiFi is still an option
Case Study: Storage Application

E/AndroidRuntime( 261): Uncaught handler: thread main exiting due to
E/AndroidRuntime( 261): java.lang.ArithmeticException: divide by zero
E/AndroidRuntime( 261): at com. _______. gui.activities.Sto
Case Study: Storage Application

- Diddling with “Digital Rights Management”
  - App supports sharing of video, audio, image content with your contacts
  - Enforces “DRM” on “protected” files
    - Often copyrighted or premium content
  - Enforcement occurs based on the value of an attribute in the file’s XML manifest
    - Yes, Virginia, that is under the user’s control
Case Study: Storage Application

Video Info

Name: [redacted]
Date: 07/04/2010 9:37 am
Copyrighted: Yes
Content: Permission is set to "VIEW"
Resolution: -
Duration: -
Size: 1 MB
Format: 3gp
Case Study: Storage Application

here's that movie you wanted

File name:
File size: 1058kB
Case Study: Storage Application

File could not be sent

File is protected by DRM
Case Study: Storage Application

Becomes...

```xml
<extension>
  <fileAttribute name="ContentPermissions">SHARE</fileAttribute>
  <fileAttribute name="CI_COMPLETE">true</fileAttribute>
  <fileAttribute name="Duration">90530</fileAttribute>
  <fileAttribute name="Height">144</fileAttribute>
  <fileAttribute name="Width">176</fileAttribute>
  <fileAttribute name="TranscodingStatus">Success</fileAttribute>
  <fileAttribute name="ContentPermissionsDetail">PENDING</fileAttribute>
  <fileAttribute name="BitRate">95.0</fileAttribute>
</extension>
```
Case Study: Storage Application

Video Info

Name: [Redacted]
Date: 07/04/2010 9:37 am

**Copyrighted: No**

Resolution: -
Duration: -
Size: 1 MB
Format: 3gp

After changing ContentPermission to "SHARE"

Intrepidus Group, Inc. © 2010
Case Study: Storage Application

- The “DRM” is basically enforced within the client, predicated on the response from the server
  - And that response can be intercepted and modified => “DRM” bypass
- CWE-807: Reliance on Untrusted Inputs in a Security Decision
  - I like CWE, btw
Case Study: Widget Framework

- Cross-platform framework for HTML/JS widgets
  - WinMo, Android, etc.
Case Study: Widget Framework

- Custom permissions restricted us from sending messages (Intents) to the widget runtime

```xml
<uses-sdk android:minSdkVersion="7" android:targetSdkVersion="7" />
<permission android:name="@string/permname_access" android:protectionLevel="signatureOrSystem" android:description="@string/permdesc_access" platform_provider="platform provider" />
```

```bash
# am start -a "android.intent.action.VIEW" -t "application/widget" -d "file://sdc
Starting: Intent { act=android.intent.action.VIEW dat=file://sdc:/../ui.manager.WidgetInstallActivity
java.lang.SecurityException: Permission Denial: starting Intent { act=android.int
0/ gt typ=application/widget flg=0x1000000 cmp=/../ui.manager.WidgetInstallActivity INFO
(pid=-1, uid=-1) requires com.
```
But, other (malicious) apps can clobber widget content!

- CWE-276: Incorrect Default Permissions
- So we wrote a malicious app to do just that 😊
Case Study: Widget Framework

```
# ls -l /data/misc
drwxrwx---  bluetooth  bluetooth  2010-07-12 16:55 bluetoothd
drwx------  keystore   keystore    2010-07-12 16:55 keystore
drwxrwx---  system     system      2010-07-12 16:55 vpn
drwxrwxrwx  root       root        2010-07-23 15:34  widgets
drwxrwx---  wifi       wifi        2010-07-12 16:55 wifi
# ls -l /data/misc/widgets
drwxrwxrwx  app_24    app_24      2010-07-23 15:39 chess
```

Intrepidus Group, Inc. © 2010
Case Study: Widget Framework
Case Study: Embedded Device #1

- Mix of HTTP and HTTPS external content
  - Strict SSL Validation
  - MITM’d HTTP traffic
    - injected HTML to enable hidden admin functionality
- Ultimate Pwn: CWE-78
  - Command Injection in wireless SSID
  - Enable Telnetd, change root pass
  - game over
Case Study: Embedded Device #2

- Excellent input validation throughout the web app....
  - except for the Command injection vuln in one parameter
    - CWE 78
  - Enable telnetd
  - toor/<blank>
  - web server runs as root
  - game over
Case Study: BREW Picture Upload

- BREW application designed to back up data from the phone
- BREW != Smart Phone
  - No WiFi
- Application Directed SMS
  - SMS Clients can parse messages, and identify specific control messages for distinct applications
  - Debug code: SMS to change target server
Case Study: BREW Picture Upload

- Configured the client to talk to our MITM on the Internet
  - Traffic was plaintext HTTP/SOAP
- Somethings were immediately clear
  - Authentication Requests
  - Static Authentication Token
  - Session Cookies
- However, we were missing something....
We began MITM’ing AFTER our account was provisioned

Re-provisioning the account through our MITM revealed the following comms:

- Client: “Here is a TON of data”
- Server: “OK, that data looks good, here is your static authentication token, which you will use FOREVER!”
Authentication token was an MD5 hash, created on the server

- Assumption was that the hash was based on data sent by the device in the previous request
- After a few attempts, we reproduced the hash. The plaintext was:

  !78![username]$[identifier]$1234

- This is where it turned into a WAPT
Once we understood how the hash was created, we knew we could hijack other user accounts

- We just needed the data
- Malicious client development
  - Probed the web service to find where it leaked data to unauthenticated users
  - You could send a mobile number to the system, and get back all data required for the auth hash
Case Study: BREW Picture Upload

- **Post-mortem**
  - No SSL (duh!)
  - No real authentication scheme
    - Carriers still trust the network
    - “If they are on our network, we can identify the user by MDN”
    - Without well architected identity management systems, authentication for arbitrary mobile applications is hard
  - No authorization controls on the server
  - Sounds like a WAPT to me :-)

Intrepidus Group, Inc. © 2010
Case Study: RIM Picture Upload

- Third-Party application, to be deployed by a carrier
- Designed to upload and save pictures from the device, and view/share on the web
- Devices were given to us with the application already installed
Case Study: RIM Picture Upload

- Tested using the emulator and a proxy
  - Extracted files using JavaLoader.exe
Main application is Application .COD
- Actually just a .ZIP!
- COD (.ZIP) was extracted to reveal multiple .COD files (compiled classes)
- Used CODDEC to decompile to Java byte code
aload_2
ldc literal_528:"4.5.0.66"
invokenonvirtual_lib java.lang.String.startsWith // pc = 2
ifne Label47
aload_2
ldc literal_528:"4.5.0.77"
invokenonvirtual_lib java.lang.String.startsWith // pc = 2
ifne Label50
Label47:
iconst_1
putstatic isRIMSSLissue // UploaderMain
goto Label52
Label50:
iconst_0
putstatic isRIMSSLissue // UploaderMainLabel52:
Case Study: RIM Picture Upload

- Hard coded TripleDES Key in Java byte code

```java
ldc literal_876:"Q1VtT0JoVmY2N2E="
invokestatic byte[] decode( java.lang.String ) // Base 64
invokespecial_lib java.lang.String.<init> // pc=2
astore 8

nullbyte:base64 mikezusman$ ./base64decode.py Q1VtT0JoVmY2N2E=BUmOBhvF67a
nullbyte:base64 mikezusman$
```
Case Study: RIM Picture Upload

invokenonvirtual_liv java.lang.String.getBytes // pc=1
astore_6
aload_0
aload_7
aload_6

...

private final byte[] com.picuploader.BizProcess.SendRequest.routine_12998
  (com.picuploader.BizProcess.SendRequest, byte[], byte[] );
{
  enter
  new_lib net.rim.device.api.crypto.TripleDESKey
dup
  aload_1
  invokespecial_lib net.rim.device.api.crypto.TripleDESKey.<init> // pc=2
Case Study: RIM Picture Upload

- Every encrypted image sent on the wire was prefixed with an auth header
Case Study: RIM Picture Upload

- WebAppSec, injection flaws abound!

```xml
Line 131: <compilation debug="true" />
Line 132: <authentication mode="Windows" />
Line 133: <identity impersonate="true" userName="FileMover" password="#rs:"
Line 134: <customErrors mode="Off" />
Line 135: <webServices>
```

Please provide the username for the Facebook account you would like to connect:

Username: mike@test.com

Intrepidus Group, Inc. © 2010
Case Study: RIM Picture Upload

- Lax permissions
  - What ever happened to least privilege?
Case Study: RIM Picture Upload

- **Post-Mortem**
  - Broken Crypto
    - hardcoded, shared keys
  - Lack of input validation
  - Lax perms, no defense in depth
  - Authentication and Authorization in web application worked well
Conclusion

- Lack of guidance, standards, practices makes developers reinvent the wheel
  - Or just make them think they need to
- Neglecting the security lessons learned with “traditional” and web applications
  - Client-side trust
  - Access control issues
  - ...and all of the other “basic” problems and mistakes of yore
Conclusion

- OWASP Mobile Security Project
  - Aims to provide resources for mobile app developers to build & deploy secure mobile apps
  - ...just like OWASP does for web applications
- Mailing list:

  https://lists.owasp.org/mailman/subscribe/owasp-mobile-security-project
Conclusion

- [http://intrepidusgroup.com/insight](http://intrepidusgroup.com/insight)
- Questions?