Mobile Security chess board - Attacks & Defense

Who Am I?

- Hemil Shah – hemil@espheresecurity.net
- Past experience – HBO, KPMG, IL&FS, Net Square
- Interest – Application security research (Web & Mobile)
- Contribution – One of the contributors for OWASP Mobile Top 10
- Global Speaker & Trainer – HITB, OWASP – EU, OWASP – India, SyScan, DeepSec, BreakPoint, HackCon, NullCon
- Published research – Articles / Papers – Packstroem, etc.
- Tools – iAppliScan, FSDroid, DumpDroid, w5Scanner, scanweb2.0, AppMap, AppCodeScan, AppPrint etc.

Mobile Apps

Market Share

Mobile Top 10 - OWASP

- Weak Server Side Controls
- Insecure Data Storage
- Insufficient Transport Layer Protection
- Unintended Data Leakage
- Poor Authorization and Authentication
- Broken Cryptography
- Client Side Injection
- Security Decisions Via Untrusted Inputs
- Improper Session Handling
- Lack of Binary Protections

Enterprise Mobile Cases
E-commerce

- Typical application making server side calls
- Security issues and hacks
  - Credit card and Private data storage with poor crypto
  - SQLite hacks
  - SQL injection over JSON
  - Ajax driven XSS
  - Several XSS with Blog component
  - Several information leaks through JSON fuzzing
- Server side scan with tools/products failed

Banking Application

- Scanning application for vulnerabilities
- Typical banking running with middleware
- Vulnerabilities – Mobile interface
  - Poor encoding to store SSN and PII information locally
  - Very sensitive transaction information stored locally
  - Default OS Behavior leaking information
  - Credentials submitted in GET request
  - Keys/session stored in keychain file

Social Application

- Social Application on multiple platforms
  - Application leverages browser component as part of the mobile
  - Common code base for all platforms
  - Vulnerable
    - Bypass Profile validation (Logical) and unique device installation
    - Screenshot revealing sensitive information
    - Default OS Behavior leaking information
    - Presentation layer (XSS and CSRF)
    - Unencrypted Communication channel

Postmortem

- One pattern in all the reviews - SOME INFORMATION WAS STORED LOCALLY
- More than 99% of the application review has the LOCAL STORAGE issue as we saw in stats.
- Server side and logical issues are still hard to find but have biggest impact.

Mobile Threats and Risk

Attacks on Mobile

iOS Weaknesses Allow Attacks Via Trojan Chargers

Skype security flaw allows users to bypass Android lock screen

Bug In iOS 7 Beta Lets Anyone Bypass iPhone Lockscreen To Access Photos

32.8 Million Android Devices Infected in 2012

And more than 13 million devices were infected in the first quarter of 2013, according to Kaspersky.
### Why should I worry?

- We have MDM in place
- We do not allow any JailBreak or rooted device in our environment with MDM
- We have strict policy enforced and all our devices are forced to have password lock
- May or may not have BYOD
- OS provides encryption

### Mobile Attacks

- So What attacks are we talking about?
- Privacy becomes important along with the Security in mobile space
- It is MOBILE so chances of losing device or someone getting physical access to it is MUCH MUCH higher than the other devices

### Exploitation

- Physical Theft
- Temporary physical access
- Malware
- Malicious Applications
- Lack of standardize security review process
- JailBreak/Rooted devices

### What can be done???

- Information found in local storage with default OS behavior –
- Changing OS behavior –
- Server side exploitation –
- XSS in Mobile Hybrid application –

### Technology Trends

#### Mobile Infrastructure

- Internet
- Exchange
- firewall
- Other Office
- Dial-up
- VPN
- DMZ
- Intranet
- Database
- RAS
Mobile App Environment

Mobile Architecture

Game is complex – Chess

Challenges

- Different code base
- Achieve things with single click
- Vendor review process - Not transparent – Can we rely on it???
- Decrease transaction time
- Competition
- Rapid business requirement results in high frequency of updates

Frequency of updates

- Very High compare to Web Applications
- Usually, 4-5 updates in a year for web applications or even less at times
- Usually, 10-12 updates in mobile applications or even more in some cases
- We all have accepted that application needs to be reviewed before going to production – DID WE???
## Frequency of Updates

<table>
<thead>
<tr>
<th>Application Name</th>
<th>Number of Releases in iOS</th>
<th>Number of Releases in Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>19</td>
<td>34</td>
</tr>
<tr>
<td>Twitter</td>
<td>22</td>
<td>25</td>
</tr>
<tr>
<td>Chase Bank</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>eBay</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Amazon</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Temple Run 2</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>FB Messenger</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>WhatsApp</td>
<td>4</td>
<td>154</td>
</tr>
<tr>
<td>skype</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>

## Mobile Attacks

- So What attacks are we talking about?
- Privacy becomes important along with the Security in mobile space
- It is MOBILE so chances of losing device or someone getting physical access to it is MUCH MUCH higher than the other devices

## Mobile Top 10 - OWASP

- Weak Server Side Controls
- Insecure Data Storage
- Insufficient Transport Layer Protection
- Unintended Data Leakage
- Poor Authorization and Authentication
- Broken Cryptography
- Client Side Injection
- Security Decisions Via Untrusted Inputs
- Improper Session Handling
- Lack of Binary Protections

## Top 5 vulnerability

- From the stats of eSphere data -

### Mobile Attacks

- Weak Server Side Controls

### Weak Server Side Controls

- SQL Injection over JSON or other streams
- Cross Site Scripting
- Copy/Paste enabled in sensitive fields - Privacy Issue
- Sensitive Information stored in Logs/Default OS Behaviour
- Local Storage
Server Side Issues

- Most Application makes server side calls to either web services or some other component. Security of server side component is equally important as client side.
- Controls to be tested on the server side – Security Control Categories for Server Side Application – Authentication, Access Controls/Authorization, API misuse, Path traversal, Sensitive information leakage,

Insecure Storage

- How attacker can gain access
  - Wifi
  - Default password after jail breaking (alpine)
  - Adb over wifi
  - Physical Theft
  - Temporary access to device

What

- What information
  - Authentication Credentials
  - Authorization tokens
  - Financial Statements
  - Credit card numbers
  - Owner’s Information – Physical Address, Name, Phone number
  - Social Engineering Sites profile/habbits
  - SQL Queries

Insufficient Transport Layer Protection
Insecure Network Channel

- Easy to perform MiM attacks as Mobile devices uses untrusted network i.e open/Public WiFi, HotSpot, Carrier’s Network
- Application deals with sensitive data i.e.
  - Authentication credentials
  - Authorization token
  - PII Information (Privacy Violation) (Owner Name, Phone number, UDID)

Insecure Network Channel

- Can sniff the traffic to get an access to sensitive data
- SSL is the best way to secure communication channel
- Common Issues
  - Does not deprecate HTTP requests
  - Allowing invalid certificates
  - Sensitive information in GET requests

Session token

Unintended Data Leakage

Unintended Data Leakage

- Platform issues – sandboxing or disable controls
  - Cache
  - Logs, Keystrokes, screenshots etc.
  - Temp files
- 3rd Party libs (AD networks and analytics)

Data Leakage

- Default OS behavior after iOS 4.0 to cache all the URLs (Request/Response) in the local storage in file named cache.db file
- Cache.db file is not encrypted
- By default, application takes last screenshot and saves it in to file system when user presses home button
Poor Authorization and Authentication

- No password complexity specially on mobile
- Hidden/No Logout button
- Long session time out
- No account lock out
- Authorization flags or based on the local storage

Broken Cryptography

- Broken implementation
- Hash/Encoding used in place of encryption
- Client side script in place of SSL

Client Side Injection

- Most Mobile platforms uses SQLite as database to store information on the device
- Using any SQLite Database Browser, it is possible to access database logs which has queries and other sensitive database information
- In case application is not filtering input, SQL Injection on local database is possible
Security Decisions Via Untrusted Inputs

Untrusted Source

- Any input from client side which can be modified
- Mainly authentication and authorization decisions based on the untrusted input
- Easiest way for developer to solve complex issues/functionality
- Attacker can get this information by either reverse engineering application or by checking local storage

KeyChain Dumper

- Easy as running a command
- Upload on to server in /var directory
- Give execute permission
  - Chmod +x /var/keychain_dumper
- Get all the keys
  - ./keychain_dumper

Improper Session Handling

Improper Session

- Session is key for any application for authorization
- Session is stored in binary format but can be easily reversible
- Application is sending sensitive information in GET request (Be it on HTTP or HTTPS)

Lack of Binary protection
Lack of Binary Protection

- Apple signs and encrypts all the binaries
- Still strings can be retrieved from the binary
- Storing Encryption and Decryption keys in the client side is still a problem

Automation in Application Reviews

Manual Review

- Looking for information in local storage manually is really –
  - Time Consuming
  - Tedious
  - Prone to be false negatives (how accurately you can check files more than once in an hour and file formats are different)

Manual Review - iOS

What do we need

- Automation!!!
- Automation!!!
- Automation!!!
- Automation!!!
- Automation!!!
- Unfortunately no complete automation is available today BUT some of the tools which can be handy are -

Snoop-it

- The only tool today to automate iOS application reviews
- Very handy and gives perfect pointer where to look for
- A long way to go for automation like web
Snoop-it (Cont...)

- Snoop-it helps you monitor –
  - File system access
  - Keychain access
  - HTTP(S) connections
  - Access to sensitive API
  - Debug outputs
  - Tracing App internals

Snoop-it (Cont...)

- Along with Monitoring, snoop-it allows to -
  - Fake hardware identifier
  - Fake location/GPS data
  - Explore and force display of available ViewController
  - List custom URL schemes
  - List available Objective-C classes, objects and methods
  - Bypass basic jailbreak detection mechanisms

iAppliScan

- iAppliScan allows you to automate iOS application review.
- Interesting features –
  - Look for sensitive information in files/directories
  - Find whether particular file exist or not
  - Download file for further analysis
  - Run external command

iAppliScan

Review without JailBreak
Reviewing without jailbreaking

• Is it really possible to review application without jailbreaking?
  • “YES”
  • “YES”
  • “YES”
  • “YES”

Reviewing without jailbreaking

• Plenty of tools available (Specially for Forensic) to brows the application directory without jailbreaking.
• iFunBox allows to view files on the device without jailbreak
• Displays application’s permissions
• Browse the installed application directory

Reviewing without jailbreaking

• Copy the entire application directory multiple times
• Look for sensitive information in the files
• Use Proxy on non-jailbreak device to check all server side attacks.

Reviewing with iFunbox

Automation in Android

Manual Review - Android
FSDroid

• Leverages SDK Class – No hacks in here!!!
• FSDroid can –
  – Monitor file system
  – Can write filter to monitor particular directory
  – Can save last 5 reports for future use
  – Does not need mobile device – can run on Emulator smoothly
  – Easy to run (As easy as giving directory name and pressing start button)

File System Monitoring Demo

Looking in to Code

Static Code Analysis

• Introduce in Mac OS X v10.6, XCode 3.2, Clang analyzer merged into XCode.
• Memory leakage warning
• Run from Build->Analyze
• Innovative shows you complete flow of object start to end
• Configure as a automatic analysis during build process

Static Code Analysis

Potential Memory Leak

Dead store – variable never used
Code Analysis with AppCodeScan

- Semi automated tool
- Ability to expand with custom rules
- Simple tracing utility to verify and track vulnerabilities
- Simple HTML reporting which can be converted to PDF

AppCodeScan

- Sophisticated tool consist of two components
  - Code Scanning
  - Code Tracer
- Allows you to trace back the variable
- AppCodeScan is not complete automated static code analyzer.
  - It only relies on regex and lets you find SOURCE of the SINK

Rules in AppCodeScan

- Writing rules is very straight forward
- In an XML file which is loaded at run time
- This release has rules for iOS and Android for
  - Local Storage, Unsafe APIs, SQL Injection, Network Connection, SSL Certificate Handling, Client Side Exploitation, URL Handlers, Logging, Credential Management and Accessing PII.

Sample Rules - Android

```xml
<Check>
  <Description>Credential Management</Description>
  <Pattern>.*AccountManager.*|.*checkSignature.*</Pattern>
  <FileTypes>ALL TEXT</FileTypes>
</Check>```

Sample Rules - iOS

```xml
<Check>
  <Description>Local Storage</Description>
  <Pattern>.*Keychain.*|.*SecAttrAccessibleWhenUnlocked.*|.*SecAttrAccessibleAfterFirstUnlock.*|.*SecItemAdd.*|.*SecItemTable.*|.*NSDataWritingFileProtectionComplete.*</Pattern>
  <FileTypes>ALL TEXT</FileTypes>
</Check>```
iOS DEMO

Debuggable flag in Android
- One of the key attributes in the Android manifest file
- Under “application” section
- Describes debugging in enabled
- If “Debuggable” attribute is set to true, the application will try to connect to a local UNIX socket “@jdwp-control”
- Using JDWP, it is possible to gain full access to the Java process and execute arbitrary code in the context of the debuggable application

CheckDebuggable Script
- Checks in APK whether debuggable is enabled
- Script can be found at – http://www.espheresecurity.com/resources/tools.html
- Paper can be found at - http://www.espheresecurity.com/CheckDebuggable.pdf

Thank you

eSphere Security Solutions pvt Ltd
Tel: +91 99790 55100, +1 201 203 7008
Email – contact@espheresecurity.net
Web – http://espheresecurity.com