Mobile Application Assessment By The Numbers – A Whole-istic View

SESSION ID: MBS-F02

Dan Cornell
CTO
Denim Group
@danielcornell
Agenda

- Background
  - Mobile Application Threat Model
  - Assessment Methodology
  - Data Collected
- Findings
  - Types of Vulnerabilities Identified
  - Where Vulnerabilities Were Identified
  - How Vulnerabilities Were Identified
Background
Introduction

- Data comes from:
  - 61 Assessments
  - 20 Applications

- What we found:
  - 957 Vulnerabilities

- Assessment with the most vulnerabilities: 3 assessments had 10 Critical vulnerabilities

- Assessments with the least vulnerabilities: only three assessments had one vulnerability (all others had more)
Research Background

- Mobile application threat model
- Assessment methodology
  - Static versus dynamic testing
  - Automated versus manual testing
- Why CWE?
- Assessment data
Mobile Application Threat Model

- More complicated than a “typical” web application threat model
- Not just about code running on the device
- Main components:
  - Mobile application
  - Enterprise web services
  - 3rd party web services
Assessment Methodology

- Testing activities
  - Combination of both static and dynamic activities
  - Combination of automated tools, manual review of automated test results and manual testing
  - Tools include Fortify SCA, IBM Rational AppScan, Portswigger BurpSuite
- Scope can include:
  - Code running on the device itself
  - Enterprise services
  - 3rd party supporting services
Determining Severity

Based on customized DREAD model:
- Damage potential
- Reproducibility
- Exploitability
- Affected users
- Discoverability

Each factor ranked 1-3

Collapsed to single dimension:
- Critical: > 2.6
- High: 2.3 – 2.6
- Medium: 2.0 – 2.3
- Low: < 2
Why CWE?

- Vulnerability taxonomy used was MITRE’s Common Weakness Enumeration (CWE)
  - [http://cwe.mitre.org/](http://cwe.mitre.org/)
- Every tool has its own “spin” on naming vulnerabilities
- OWASP Top 10 / WASC 24 are helpful but not comprehensive
- CWE is exhaustive (though a bit sprawling at times)
- Reasonably well-adopted standard
- Many tools have mappings to CWE for their results
Assessment Data

- Subset of mobile assessments
- Mostly customer-facing applications from financial services organizations
- Primarily iOS and Android applications
  - Some WAP, Windows Phone 7
What Did We Find?
Types of Vulnerabilities Found

- Top 10 Most Prevalent CWEs – Overall
- Top 10 Most Prevalent CWEs – Critical/High Risk
Top 10 Most Prevalent CWEs – Overall

<table>
<thead>
<tr>
<th>CWE Description</th>
<th>Risk Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use of a Broken or Risky Cryptographic Algorithm - LOW RISK</td>
<td></td>
</tr>
<tr>
<td>Information Exposure Through an Error Message - LOW RISK</td>
<td></td>
</tr>
<tr>
<td>Cross-Site Request Forgery (CSRF) - LOW RISK</td>
<td></td>
</tr>
<tr>
<td>Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection') - CRITICAL</td>
<td></td>
</tr>
<tr>
<td>Improper Input Validation - LOW RISK</td>
<td></td>
</tr>
<tr>
<td>External Control of System or Configuration Setting - LOW RISK</td>
<td></td>
</tr>
<tr>
<td>Information Leak Through Debug Information - LOW RISK</td>
<td></td>
</tr>
<tr>
<td>Cleartext Transmission of Sensitive Information - LOW RISK</td>
<td></td>
</tr>
<tr>
<td>Information Leak Through Log Files - LOW RISK</td>
<td></td>
</tr>
<tr>
<td>Information Exposure - LOW RISK</td>
<td></td>
</tr>
</tbody>
</table>

- Information Leak Through Log Files - LOW RISK: 284
- Information Exposure - LOW RISK: 271
- Cleartext Transmission of Sensitive Information - LOW RISK: 26
- Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection') - CRITICAL: 22
- Improper Input Validation - LOW RISK: 21
- External Control of System or Configuration Setting - LOW RISK: 21
- Information Leak Through Debug Information - LOW RISK: 20
- Cross-Site Request Forgery (CSRF) - LOW RISK: 16
- Information Exposure Through an Error Message - LOW RISK: 14
- Use of a Broken or Risky Cryptographic Algorithm - LOW RISK: 14
Top 10 Most Prevalent CWEs – Critical/High Risk

1. Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection') - CRITICAL
2. Information Leak Through Caching - HIGH
3. Access Control Bypass Through User-Controlled Key - CRITICAL
5. Exposure of Access Control List Files to an Unauthorized Control Sphere - CRITICAL
6. Incorrect User Management - CRITICAL
7. Uncontrolled Resource Consumption ('Resource Exhaustion') - CRITICAL
8. Missing XML Validation - CRITICAL
9. Failure to Preserve Web Page Structure ('Cross-Site Scripting') - CRITICAL
10. Uncontrolled Resource Consumption ('Resource Exhaustion') - CRITICAL

Graph showing the prevalence of each CWE with labels and values:
- Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection') - CRITICAL: 22
- Information Leak Through Caching - HIGH: 6
- Access Control Bypass Through User-Controlled Key - CRITICAL: 6
- Access Control (Authorization) Issues - CRITICAL: 4
- Exposure of Access Control List Files to an Unauthorized Control Sphere - CRITICAL: 3
- Incorrect User Management - CRITICAL: 3
- Uncontrolled Resource Consumption ('Resource Exhaustion') - CRITICAL: 3
- Missing XML Validation - CRITICAL: 2
- Failure to Preserve Web Page Structure ('Cross-Site Scripting') - CRITICAL: 1
- Uncontrolled Resource Consumption ('Resource Exhaustion') - CRITICAL: 1
Similar to the OWASP Top 10 Web Application Risks, but targeted at mobile applications (obviously)

Top risks to mobile applications:


Work in progress to update this based on industry-contributed data
OWASP Top 10 Mobile Risks

M1: Insecure Data Storage
M2: Weak Server Side Controls
M3: Insufficient Transport Layer Protection
M4: Client Side Injection
M5: Poor Authorization and Authentication

M6: Improper Session Handling
M7: Security Decisions Via Untrusted Inputs
M8: Side Channel Data Leakage
M9: Broken Cryptography
M10: Sensitive Information Disclosure
Compare to OWASP Top 10 Mobile Risks

Strong Overlap
- Weak server-side controls
- Poor authentication and authorization
- Security decisions via untrusted inputs
- Sensitive information disclosure

Overlap
- Insecure data storage
- Insufficient transport layer data protection
- Improper session handling
- Side channel data leakage
- Broken cryptography

Weak Overlap
- Client-side injection
Where Did We Find Overall Vulnerabilities?

- Device: 342 (36%)
- Corporate Web Service: 591 (62%)
- Third-Party Web Service: 24 (2%)
Where Did We Find Critical/High Risk Vulnerabilities?

- Corporate Web Service: 41 (70%)
- Device: 15 (25%)
- Third Party Web Service: 3 (5%)
Analysis of “Where” Data

- Mobile security is about more than the code running on the device
- The things we really care about (Critical, High) are most frequently found on corporate web services
- Then on the device
- Then on 3rd party web services
- Reflects the “scale” benefits of finding web services vulnerabilities
How Did We Find Vulnerabilities?

- Static vs. dynamic testing
- Automated vs. manual testing
- What techniques identified the most vulnerabilities?
- What techniques identified the most serious vulnerabilities?
Static vs. Dynamic Method of Finding Vulnerabilities

- **Static**
  - Low Risk: 599
  - Medium Risk: 206
  - High Risk: 14
  - Critical: 33

- **Dynamic**
  - Low Risk: 206
  - Medium Risk: 84
  - High Risk: 14
  - Critical: 10
Static vs. Dynamic Method of Finding Vulnerabilities

**Static**
- Low Risk: 93%
- Critical: 5%
- High Risk: 0%
- Medium Risk: 2%

**Dynamic**
- Low Risk: 66%
- Critical: 3%
- High Risk: 4%
- Medium Risk: 27%
Critical and High Risk Vulnerabilities

- Static testing was more effective when finding serious (Critical and High) vulnerabilities.
- But it also found a lot of lower-risk vulnerabilities (as well as results that had to be filtered out).

### Critical/High Risk Vulnerabilities Found

- Found with Dynamic Testing: 24 (41%)
- Found with Static Testing: 35 (59%)
Automated vs. Manual Method of Finding Vulnerabilities

**Manual**
- Critical, 10
- High Risk, 15
- Medium Risk, 89

**Automatic**
- Critical, 33
- High Risk, 1
- Medium Risk, 4

**Results**
- Low Risk, 279
- Medium Risk, 89
- Critical, 10
- High Risk, 15
- Low Risk, 526
- Medium Risk, 4
- Critical, 33
- High Risk, 1
Automated vs. Manual Method of Finding Vulnerabilities

**Automatic**
- Low Risk: 93%
- Critical: 6%
- High Risk: 0%
- Medium Risk: 1%

**Manual**
- Low Risk: 71%
- Critical: 2%
- High Risk: 4%
- Medium Risk: 23%
Automated vs. Manual Method of Finding Vulnerabilities (Critical and High)

- Automated testing was more effective when finding serious (Critical and High) vulnerabilities

![Critical/High Risk Vulnerabilities Found](chart)

- Found with Automated Testing: 34 (58%)
- Found with Manual Testing: 25 (42%)
Automated vs. Manual, Static vs. Dynamic Methods

<table>
<thead>
<tr>
<th></th>
<th>Automatic / Static</th>
<th>Manual / Dynamic</th>
<th>Manual / Static</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Risk</td>
<td>526</td>
<td>206</td>
<td>5</td>
</tr>
<tr>
<td>Medium Risk</td>
<td>4</td>
<td>84</td>
<td>73</td>
</tr>
<tr>
<td>High Risk</td>
<td>1</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>Critical</td>
<td>33</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
Automated vs. Manual, Static vs. Dynamic Methods

<table>
<thead>
<tr>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>79</td>
</tr>
<tr>
<td>Automatic</td>
<td>564</td>
</tr>
<tr>
<td>Manual</td>
<td>314</td>
</tr>
<tr>
<td>Automatic</td>
<td>0</td>
</tr>
</tbody>
</table>
Automated vs. Manual, Static vs. Dynamic for Critical and High Vulnerabilities

<table>
<thead>
<tr>
<th></th>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>Automatic</td>
<td>34</td>
<td>0</td>
</tr>
</tbody>
</table>
Analysis of “How” Data

- A comprehensive mobile application security assessment program must incorporate a significant manual testing component
- Automated tools for testing mobile applications are not as mature as those for testing web applications
- Web services can be challenging to test in an automated manner
## On-Device Vulnerabilities By Platform

<table>
<thead>
<tr>
<th>Platforms</th>
<th>Number of Assessments on Device</th>
<th>Number of Total Vulnerabilities on Device</th>
<th>Average Number of Vulnerabilities Found per Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>39</td>
<td>252</td>
<td>6.5</td>
</tr>
<tr>
<td>Android</td>
<td>19</td>
<td>84</td>
<td>4.4</td>
</tr>
<tr>
<td>Windows Phone 7</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>WAP</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Other Observations

- We also include “other observations” as part of our assessments
- These reflect:
  - Application weaknesses
  - Coding flaws or behavior that are not “best practice” but do not reflect an immediate, exploitable vulnerability
- We had 1,948 “other observations”
  - Roughly twice as many as actual vulnerabilities
Other Observations – Where Were They Found?

- **Device**: 1892, 97%
- **Third-Party Web Service**: 1, 0%
- **Corporate Web Service**: 55, 3%
What Does This Mean?

- Most of these “other observations” are about code on the device
  - Mobile application developers need help building better code
  - AND automated code scanning tools need to be better about filtering less valuable results

- Something that is not a problem today could be later on
  - Identification of new platform vulnerabilities
  - Changes coming along with a new application release
Conclusions

- **What To Test?**
  - Mobile “apps” are not standalone applications
  - They are systems of applications
  - Serious vulnerabilities can exist in any system component

- **How To Test?**
  - Mobile application testing does benefit from automation
  - Manual review and testing is required to find the most serious issues
  - A combination of static and dynamic testing is required for coverage
Recommendations

- Plan your mobile application assessment strategy with coverage in mind
- Evaluate the value of automation for your testing
  - More “cost” than simply licensing – deployment time and results culling
- Look for opportunities to streamline
  - Fast application release cycles can require frequent assessments
- Control scope:
  - Assess application changes (versus entire applications)
  - Manage cost of reporting
Next Steps (For Us)

- Incorporate more assessment data
- Possible collaboration with OWASP Top 10 Mobile Risks
  - Currently being reworked based on data sets such as ours
- Better analysis of applications over time