Agenda

- Introductions
- OWASP Summit Recap
- OWASP Mobile Top 10 v0.1 (DRAFT)
Who Are We
Other than carbon-based multi-cellular life forms
What is OWASP Mobile?

- Traditional OWASP projects do not cater to the specific risks faced by mobile apps
- Many mobile apps are effectively single-site browsers
- web application != your own web browser
- The OWASP Mobile Security Project aims to fill these knowledge gaps
Why? New Threats

- App to App Data Leakage
  - No concept of “Same Origin Policy”
- SMS Attacks
- Abuse of Client Side Paid Resources
- “AppSpoofing” (platform agnostic)
  - Recent Trojaned Android App’s
Why? <facepalm>

- Scary, scary mobile payment sample code

```java
final double amount = 848.64;
final String currency = "USD";
final String companyId = "685123";
final String companyPassword = "vRWoQu7Cw8sN3KbbLbEajQ==";
final String messageId = "12TG65H987BBQJ56742";
final String settlementId = "12345WXYZ6789";
final String cardHolderName = "John Doe";
final String accountNumber = "5466160123456789";
final String expiryMonth = "01";
```
Why? <facepalm #2>

OWASP Summit Recap

- Trapped on OWASP Island!
  - All OWASP, all the time

- 1 dynamic working session, and 1 official working group

- Key outcomes
  - Metric driven OWASP Top 10 initiative (Late Q2)
  - Secure Development Guidelines (Late Q3)
  - Testing Guide (eta?)
  - Target platforms: Android, iOS, Rim, Windows Phone 7
Mobile Attack Surface

- Mobile Platforms (operating systems, hardware, pay services)
- Native Applications (contacts, email, dialer)
- Third-party Client Applications <- OWASP Mobile
- Internet facing backend systems <- OWASP
- The “AppStore” (platform agnostic)
Top 10 Risks for Mobile

- OWASP Mobile Top 10

- Veracode Mobile Top 10
Top 10 Risks for Mobile

- Identify tactical solutions and guide strategic improvement

- Top 10 Mobile Risks (Veracode) (for testers?)
  - Malicious Behavior
  - Vulnerabilities

- OWASP Top 10
  - All vulnerabilities, all the time
  - Focus on what developers can control
Top Ten List (0.1)

1. Insecure or unnecessary client-side data storage
2. Lack of data protection in transit
3. Personal data leakage
4. Failure to protect resources with strong authentication
5. Failure to implement least privilege authorization policy
6. Client-side injection
7. Client-side DOS
8. Malicious third-party code
9. Client-side buffer overflow
10. Failure to apply server-side controls

Case Studies

News from the trenches
One: Client Side Storage

1. Insecure or unnecessary client-side data storage
   - Secret information saved on RIM client
   - Developers had to work around a platform bug affecting their use of SSL
One: RIM Data Back-up

- Secret information saved on RIM client
- Developers had to work around an SSL platform bug
- Once information is stored on device, it should be considered public

```java
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 // UploaderMain
Label52:
```
One: Client Side Storage

```java
ldc literal_876:"QIVtT0JoVmY2N2E="
invokestatic byte[] decode( java.lang.String ) // Base 64
invokespecial_lib java.lang.String.<init> // pc=2
astore 8
```

```bash
nullbyte:base64 mikezusman$ ./base64decode.py QIVtT0JoVmY2N2E=BUmOBhVf67a
nullbyte:base64 mikezusman$
```
One: Client Side Storage

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

...
One: Recommendations

- Do not store secret keys in client code
- Do not save password in easily accessible places
  - SD Card
- Be careful with verbose log messages in production
Two, Three, Four: CIAuth

- Two: Lack of Data Protection in Transit
  - Integrity

- Three: Personal Data Leakage
  - Confidentiality

- Four: Failure to Protect Resources with Strong Authentication
  - Authentication
Two, Three, Four: Picture Backup

- Confidentiality & Integrity Fail
- BREW Application designed to back up data from the device
  - BREW != Smart Phone
  - No WiFi, only radio network
- We became MITM
  - Details coming in a few slides (Risk #6)
- The server did not use SSL
  - Walled garden mentality
  - We could see all PII in plaintext
Two, Three, Four: Picture Backup

- Authentication Fail
- PII included authentication data
- Authentication was not straightforward, but it was not secure either
  - Convoluted device registration process
  - User was never engaged
  - No secrets exchanged
- We built our own client to masquerade as anyone
  - Not #winning
Two, Three, Four: Recommendations

- Use SSL, and validate certificates
- Encoding != Encryption
- Client Side Security != Guarantee
- Implement Strong Authentication controls
  - Use secrets
  - Out of band mechanisms
Five: Least Privilege Authorization Policy

- Applications & users only have access to necessary resources
- Authorization checks should be enforced by the server (see #10)
- Your application is in the hands of the attacker
  - Hidden functionality will be exposed
Custom permissions restricted us from sending messages (Intents) to the application runtime.
But, other (malicious) apps can clobber application content!

- CWE-276: Incorrect Default Permissions
- So we wrote a malicious app to do just that 😊
Six: Client Side Injection

- Some platforms make heavy use of HTML & JavaScript:
  - WebView, embedded browsers
  - Rich, easily developed UI

- Android Web Market XSS (3/7/2010):
  - Arbitrary app installation / RCE
Six: Client Side Injection

- Remember the BREW Data Backup application?

- We became MITM using SMS
  - SMS used to change the IP address of the configured server

- SMS used to “push” notifications to applications
Six: Recommendations

- Output encoding as a primary defense
  - Render user data safe

- Whitelisting
  - Use input validation when appropriate
Seven: BOOM!

- Seven: Client Side Denial of Service
  - Coding Bugs
  - Poor input validation
  - Fast development cycles lead to errors

```
E/AndroidRuntime( 261): Uncaught handler: thread main exiting due to
E/AndroidRuntime( 261): java.lang.ArithmeticException: divide by zero
E/AndroidRuntime( 261):     at com. example .gui.activities.Sto
```
Eight: Malicious 3rd Party code

- 3rd party code can leak your users data with your knowledge or their consent
- Don’t explicitly trust 3rd party code
  - Ad networks
  - Open Source libs
Nine: Client Side BoF

- General Memory Corruption Bugs
- Incorrect Assumptions on Input
- Lack of Data Validation
Nine: Recommendations

- Validate ALL user input
  - Correct Format
  - Correct Size
  - Etc
  - Sound familiar?

- Don’t write web apps in C
  - Use Managed Languages
  - Auto-magically protect against common bugs
Ten: Failure to Apply Server Side Controls

- PLEASE SEE 2010 OWASP TOP 10

Two Arrested in Massive iPad Hack Attack
Hackers Allegedly Uncovered Email Addresses to Bold Face Names Including Mayor Bloomberg

BY EAMON MCNIFF
Jan. 18, 2011

Two men described as "Internet trolls" have been charged today with allegedly hacking AT&T's servers to obtain the information of 120,000 iPad users, including some boldface names like Hollywood mogul Harvey Weinstein, New York Mayor Michael Bloomberg and President Obama's former chief of staff.
The OWASP Mobile Top 10 will begin to guide organizations down a secure path

Create Awareness and Make Resources Available
  - Help organizations develop & establish mobile secure development standards

Establish Initiatives with Platform Vendors

Educate and provide tools for testers
The end.

- http://intrepidusgroup.com/insight
- Questions?