Fast forwarding Mobile Security with the MSTG

Jeroen Willemsen – OWASP Benelux days
About me

Jeroen Willemsen
@commjoenie
jeroen.willemsen@owasp.org
“Security architect”
“Full-stack developer”
“Mobile security”

@OWASP_MSTG
Agenda

• Introduction into the MASVS

• Introduction into the MSTG

• Some examples
The MSTG: mobile security?

QUESTION:

Can you do a CSRF or XSS attack on a native mobile app without a webview?

Answer:

XSS: No,
CSRF: No. Even with deeplinks it is not the same.
The MSTG: mobile security?

• So CSRF and XSS do not easily apply.

• But path-traversals do...
The MSTG: mobile security?

• So CSRF and XSS do not easily apply.
• But path-traversals do...
• And then there is... Data leakage
  – through logging,
  – through insecure storage,
  – Through IPC.
• What about weak authentication mechanisms?
• What about reverse engineering?
How do we fix this?

Mobile Application Security Verification Standard
https://github.com/OWASP/owasp-masvs

Mobile Security Testing Guide
https://github.com/OWASP/owasp-mstg

Mobile Appsec Checklist
OWASP Mobile AppSec Verification Standard (MASVS)

• Started as a fork of the OWASP ASVS
• Formalizes best practices and other security requirements
• Mobile-specific, high-level, OS-agnostic

• Why?
  • Shift left: give security requirements a-priori
OWASP Mobile AppSec Verification Standard (MASVS)
<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>L1</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>System credential storage facilities are used appropriately to store sensitive data, such as user credentials or cryptographic keys.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>No sensitive data is written to application logs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>No sensitive data is shared with third parties unless it is a necessary part of the architecture.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>The keyboard cache is disabled on text inputs that process sensitive data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>The clipboard is deactivated on text fields that may contain sensitive data.</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2.6</td>
<td>No sensitive data is exposed via IPC mechanisms.</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2.7</td>
<td>No sensitive data, such as passwords or pins, is exposed through the user interface.</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2.8</td>
<td>No sensitive data is included in backups generated by the mobile operating system.</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
How to use the MASVS?

During early stages of development:
• Basis for (future) design decisions and enhancements
• Helps building internal baselines for Mobile Security and Coding Guidelines
• To determine security requirements early on. For example:

1.3 Security controls are never enforced only on the client side, but on the respective remote endpoints.

While Implementing:
• Track the security requirements during development
• Redefine security requirements when business requirements are changing

During Penetration Test:
• Share the status of your security requirements with the tester
Current status MASVS

• Current release: 1.1 (English)
• Translations:
  – Released: Spanish, Russian
  – Ready: French, German, Japanese
  – In progress: Chinese (ZHTW)
  – Started: Persian
Current status MASVS

- Current release: 1.1
- Translations
- Lab-project status!
Current status MASVS

- Current release: 1.1
- Translations
- Lab-project status!
- NIST 800-163, revision 1
## Current status MASVS

<table>
<thead>
<tr>
<th>Project Lead</th>
<th>Lead Author</th>
<th>Contributors and Reviewers</th>
</tr>
</thead>
</table>
Future plans for the MASVS

- Ongoing: Integration with SKF
- Ongoing: Automate & simplify releases
- Ongoing conversations with the Cloud Security Alliance.
- Revisit Location & Connectivity requirements
- Re-evaluate the need for payload encryption
- Add more translations
Your turn!

- https://github.com/OWASP/owasp-masvs
- https://mobile-security.gitbook.io/masvs/

- Download it
- Read it
- Use it
- Give Feedback! Create an issue or a PR
- Tweet about it (@OWASP_MSTG)
Agenda

• Introduction into the MASVS

• Introduction into the MSTG

• Some examples
OWASP Mobile Security Testing Guide (MSTG)

- Manual for testing security maturity of iOS and Android (mostly) native apps.
- Maps on MASVS requirements.

Why?
- Educate developers and penetration testers.
- Provide a baseline for automated checks
OWASP Mobile Security Testing Guide (MSTG)

- General testing guide
- Android Testing guide
- iOS Testing guide
OWASP Mobile Security Testing Guide (MSTG)

• General testing guide
• Android Testing guide
• iOS Testing guide
• Crackme’s & Challenges

Kudos to Bernhard Mueller @bernhardm for his hard work!
OWASP Mobile Security Testing Guide (MSTG)

- General testing guide
- Android Testing guide
- iOS Testing guide
- Crackme’s & Challenges
- Mobile Appsec Checklist
OWASP Mobile Security Testing Guide (MSTG)

- General testing guide
- Android Testing guide
- iOS Testing guide
- Crackme’s & Challenges
- Mobile Appsec Checklist
- MSTG playground (External)
Current status MSTG

- We JUST released 1.1.0 TODAY!!!
- Lab-project & Mentioned in NIST 800-163, revision 1, 3K+ stars
- Automation: Simplified Crackme maintenance & document generation
# Current status MSTG

<table>
<thead>
<tr>
<th>Authors</th>
<th>Co-Authors</th>
<th>Top Contributors</th>
<th>Reviewers</th>
<th>Editors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeroen Willemsen (@jeroenwillemsen)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sven Schleier (@sushi2k)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The full list of contributors is available on GitHub:  
[https://github.com/OWASP/owasp-mstg/graphs/contributors](https://github.com/OWASP/owasp-mstg/graphs/contributors)
Ongoing work for MSTG

- Adding code samples in Swift and Kotlin
- Adding Android 8/9 & iOS 12 updates (ongoing for 1.2)
- Translation to Japanese & Russian (ongoing)
- Getting hardcopies available
Future plans MSTG

• Migrate crackmes and MSTG playground to one repository and develop more bad/good examples
• Restructure the MSTG to align with the MASVS
• Consider MDM write-ups (version 1.3)?
• Add more crackme exercises for iOS
• Seek collaboration with Apple / Google to speed up?
• Collaborate with standardization bodies
Your turn!

- [https://github.com/OWASP/owasp-mstg](https://github.com/OWASP/owasp-mstg)
- [https://mobile-security.gitbook.io/mstg/](https://mobile-security.gitbook.io/mstg/)

- Download it
- Read it
- Use it
- Give Feedback (file an issue)
- **Fix issues: send in your Pull Requests!**
- Tweet about it (@OWASP_MSTG)
Agenda

- Introduction into the MASVS
- Introduction into the MSTG
- Some examples
Let’s not repeat ourselves!

This happened yesterday:

Training 3 - Android security workshop by Jeroen Beckers & Stephanie Vanroelen

Let’s give some love to iOS!
SSL pinning

<table>
<thead>
<tr>
<th>Version</th>
<th>Certificate Serial Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate Algorithm</td>
<td>Identity for Certificate Issuer’s Signature</td>
</tr>
<tr>
<td>Issuer</td>
<td></td>
</tr>
<tr>
<td>Validity Period</td>
<td></td>
</tr>
<tr>
<td>Subject</td>
<td></td>
</tr>
<tr>
<td>Subject Public-Key</td>
<td>Algorithm Identifier</td>
</tr>
<tr>
<td>Information</td>
<td>Public-key Value</td>
</tr>
<tr>
<td>Issuer Unique Identifier</td>
<td></td>
</tr>
<tr>
<td>Subject Unique Identifier</td>
<td></td>
</tr>
<tr>
<td>Extensions</td>
<td></td>
</tr>
<tr>
<td>Certification Authority's Digital Signature</td>
<td></td>
</tr>
</tbody>
</table>
SSL pinning – SSL killswitch V2

Two easy ways to break most pinners:

1. Jailbreak ➔ use Cydia & SSL Killswitch V2

2. Do dynamic instrumentation on a non-jailbroken device

See https://github.com/OWASP/owasp-mstg/blob/master/Document/0x04f-Testing-Network-Communication.md
and https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06g-Testing-Network-Communication.md
<table>
<thead>
<tr>
<th>ID</th>
<th>Host</th>
<th>Method</th>
<th>URL</th>
<th>Parameters</th>
<th>Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1KB</td>
<td><a href="https://www.apple.com">https://www.apple.com</a></td>
<td>POST</td>
<td>/search-services/suggestions/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2KB</td>
<td><a href="https://www.apple.com">https://www.apple.com</a></td>
<td>GET</td>
<td>/search-services/suggestions/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3KB</td>
<td><a href="https://www.apple.com">https://www.apple.com</a></td>
<td>GET</td>
<td>/search-services/suggestions/</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Request Headers:

- Content-Type: application/json
- Host: www.apple.com
- User-Agent: Mozilla/5.0 (iPhone, CPU iPhone OS 13_3 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Mobile/15E148
- Connection: close
- Accept-Language: zh-CN
SSL pinning – SSL killswitch V2

Mobile substrate

SSL killswitch

Patch underlying SSL handshake implementation
Used by NSURLconnection For all apps...

Mobile app @ iOS 9
SSLHandshake, SSLSetSessionOption, SSLCreateContext

Mobile app @ iOS 10 / 11
tls_helper_create_peer_trust
What if you don’t want to jailbreak?

• Jailbroken devices require maintenance
• Jailbreaks are getting harder to find
• What about jailbreak protection of the app?
• Let’s patch the app itself!
SSL pinning – non-jailbroken device
SSL pinning – Objection

1. Frida server in Gadget waits
2. Objection connects to server with explore REPL
3. Objection calls script that patches underlying SSL handshake implementation

Patch underlying SSL handshake implementation Used by NSURLConnection For **one** app.
TouchID the wrong way: using LAContext

There are 2 ways to use TouchID:

1. Protect an entry in the keychain and unlock it via TouchID

2. Use the LocalAuthenticationContext:
   
   ```
   LocalAuthenticationContext.evaluationPolicy(.deviceOwnerAuthenticationWithBiometrics, localizedReason: reasonString) {
       success, evaluateError in {
           if success {
               successmethods()  
           } else {
               ....
           }
   }
   ```

   What if we call the `successmethods()` directly?
Bypassing Touch-ID

• With **needle**

• With **Objection**

• Both cases: use Frida to hook onto
  `evaluatePolicy:localizedReason:reply`
  – Ensures that when *evaluatePolicy* is called that the reply its success is set to true (E.g.: call success methods)

There is much more!

- Reverse Engineering
  - Root / Jailbreak Detection
  - Anti-Debugging
  - Detecting Reverse Engineering Tools
  - Emulator Detection / Anti-Emulation
  - File and Memory Integrity Checks
  - Device Binding
  - Obfuscation
There is much more!

- Reverse Engineering
- Analysis & best practices for
  - Storage
  - Cryptography
  - Local Authentication
  - Network Communication
  - Code quality & build settings
I WANT YOU

TO HELP US TO
THE NEXT LEVEL!
QUESTIONS?
@OWASP_MSTG
jeroen.willemsen@owasp.org
THANK YOU!

@OWASP_MSTG
jeroen.willemsen@owasp.org