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
Mobile security

1. Can you do a Cross Site Scripting (XSS) attack in a native app?
   1. What if there is no webview?

2. Can you do a Cross-Site Request Forgery (CSRF) attack in a native app without a webview?
Mobile security?

- So CSRF and XSS do not easily apply.

- But path-traversals do...
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

- 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.
- Give a clear goal during implementation
- Give a clear goal during penetration testing
• Architecture & design
• Data storage & privacy
• Cryptography
• Authentication & Session management

• Network Communication
• Platform Interaction
• Code quality & build settings
• Resilience requirements
## V2: Data Storage and Privacy Requirements

<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>
## Current status MASVS

<table>
<thead>
<tr>
<th>Project Lead</th>
<th>Lead Author</th>
<th>Contributors and Reviewers</th>
</tr>
</thead>
</table>
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
- MSTG playground (External)
# 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>Bernhard Mueller</td>
<td>Carlos Holguera</td>
<td>Jeroen Beckers, Pawel Rzepa, Francesco Stillavato, Andreas Happe, Anthuk, Hoggle, Wen Bin Kong, Abdessamad Temmar, Bolot Kerimbaev, Cldiu André, Slawomir Kosowski</td>
<td>Sjoerd Langkemper, Anant Shrivastava, Jeroen Beckers</td>
<td>Heaven Hodges, Caitlin Andrews, Nick Epson, Anita Diamond, Anna Szkudlarek</td>
</tr>
<tr>
<td>Jeroen Willemsen (@jeroenwillemsen)</td>
<td>Romuald Szkudlarek</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)
MSTG Project status
MSTG Project status – allways more work

• Update to iOS 12/13 & Android Pie/Q
• Restructure MSTG
• Add missing testcases
• Automate MSTG playground & merge with crackmes
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
# Network Communication Requirements

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>L1</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Data is encrypted on the network using TLS. The secure channel is used</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>consistently throughout the app.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>The TLS settings are in line with current best practices, or as close as</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>possible if the mobile operating system does not support the recommended</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>standards.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>The app verifies the X.509 certificate of the remote endpoint when the</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>secure channel is established. Only certificates signed by a trusted CA are</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>accepted.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>The app either uses its own certificate store, or pins the endpoint</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>certificate or public key, and subsequently does not establish connections</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>with endpoints that offer a different certificate or key, even if signed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>by a trusted CA.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>The app doesn't rely on a single insecure communication channel (email or</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SMS) for critical operations, such as enrollments and account recovery.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>The app only depends on up-to-date connectivity and security libraries.</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Network Communication Requirements

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>L1</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Data is encrypted on the network using TLS. The secure channel is used consistently throughout the app.</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>5.2</td>
<td>The TLS settings are in line with current best practices, or as close as possible if the mobile operating system does not support the recommended standards.</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>5.3</td>
<td>The app verifies the X.509 certificate of the remote endpoint when the secure channel is established. Only certificates signed by a trusted CA are accepted.</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

- The MSTG will guide you on how to review the code & do dynamic analysis of
  - The usage of TLS in general
  - The settings of the TLS connection in general
  - Certificate validation (general, iOS and Android specific)
Network Communication Requirements

5.4 The app either uses its own certificate store, or pins the endpoint certificate or public key, and subsequently does not establish connections with endpoints that offer a different certificate or key, even if signed by a trusted CA.
SSL pinning

<table>
<thead>
<tr>
<th>Version</th>
<th>Certificate Serial Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate Algorithm Identifier for Certificate Issuer’s Signature</td>
<td></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 Information</td>
<td>Algorithm Identifier</td>
</tr>
<tr>
<td></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>OWASP</td>
</tr>
</tbody>
</table>
Network Communication Requirements

• The MSTG helps in finding ways to do pinning
  – In Android (OKHttp, WebView, networkSecurityConfig, using TrustManagers,
  – In iOS (NSURLConnection, TrustKit, AFNetworking, Alamofire)
  – Hybrid/multiplatform: Apache Cordova, Xamarin, Phonegap.

• But what about verifying it? Or bypassing it?
SSL Pinning – verify whether it is on

• Android:
  – Below Android 7: install your Burp/mitmproxy/Zap CA on the device,
  – Android 7 and above: rework networksecurityconfig.xml
  – Try to MiTM the application.

• iOS:
  – Install install your Burp/mitmproxy/Zap CA on the device
  – Try to MiTM the application.
SSL Pinning – bypassing it

• iOS: SSL Killswitch V2

• iOS: Frida & Objection

• Android: Xposed

• Android: Frida & Objection
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>#</th>
<th>Host</th>
<th>Method</th>
<th>URL</th>
<th>Params</th>
<th>Edited</th>
<th>Raw</th>
</tr>
</thead>
<tbody>
<tr>
<td>2608</td>
<td><a href="https://securemetrics.apple.com">https://securemetrics.apple.com</a></td>
<td>GET</td>
<td>/b/ss/applecn/global.applechome/a…</td>
<td></td>
<td></td>
<td>POST /search-services/suggestions/ HTTP/1.1</td>
</tr>
<tr>
<td>2606</td>
<td><a href="https://www.apple.com">https://www.apple.com</a></td>
<td>GET</td>
<td>/cn/shop/bag/status?apikey=5FX9Y…</td>
<td></td>
<td></td>
<td>User-Agent: Mozilla/5.0 (iPhone, CPU iPhone OS 10_3_3 like Mac OS X) AppleWebKit/603.3.8 (KHTML, like Gecko) Version/10.0 Mobile/14C60 Safari/602.1</td>
</tr>
</tbody>
</table>

Cookie: s_fid=2E61E7A6F5662F8E-24ECSEE849EA7008; s_pathLength=homepage%3D1%2C; s_v[1]=12DC14444052E5869-40000C3460000AAE[CE]

Accept-Encoding: gzip, deflate

Connection: close

Content-Type: application/json

Referer: https://www.apple.com/cn/

Accept-Language: en-sg

"query":"","src":"globalnav","id":"cfcf5f96-2936-a552-01b4-eb73956a3929","locale":"zh_CN"
SSL Pinning – SSL killswitch V2

Mobile substrate

SSL killswitch

MSHookFunction

Mobile app @ iOS 9

SSLHandshake, SSLSetSessionOption, SSLCreateContext

Mobile app @ iOS 10 / 11

tls_helper_create_peer_trust

Patch underlying SSL handshake implementation Used by NSURLConnection For all apps...
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!

FRIDA

OBJECTION
RUNTIME
MOBILE
EXPLORATION
GIT.IO/OBJECTION

OWASP Open Web Application Security Project
SSL pinning – non-jailbroken device
SSL Pinning – Objection

Patch underlying SSL handshake implementation
Used by NSURLConnection
For **one** app.

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

Let’s do similar runtime patching in Android...
SSL Pinning in Android - Xposed

Let’s pick it up from the rooted device again...
## Authentication requirements

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>L1</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>If the app provides users access to a remote service, some form of authentication, such as username/password authentication, is performed at the remote endpoint.</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>4.2</td>
<td>If stateful session management is used, the remote endpoint uses randomly generated session identifiers to authenticate client requests without sending the user's credentials.</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>4.3</td>
<td>If stateless token-based authentication is used, the server provides a token that has been signed using a secure algorithm.</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>4.4</td>
<td>The remote endpoint terminates the existing session when the user logs out.</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>4.5</td>
<td>A password policy exists and is enforced at the remote endpoint.</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>4.6</td>
<td>The remote endpoint implements a mechanism to protect against the submission of credentials an excessive number of times.</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>4.7</td>
<td>Biometric authentication, if any, is not event-bound (i.e. using an API that simply returns &quot;true&quot; or &quot;false&quot;). Instead, it is based on unlocking the keychain/keystore.</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>4.8</td>
<td>Sessions are invalidated at the remote endpoint after a predefined period of inactivity and access tokens expire.</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>4.9</td>
<td>A second factor of authentication exists at the remote endpoint and the 2FA requirement is consistently enforced.</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>4.10</td>
<td>Sensitive transactions require step-up authentication.</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>4.11</td>
<td>The app informs the user of all login activities with their account. Users are able view a list of devices used to access the account, and to block specific devices.</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>

### References

There are 2 ways to use TouchID:

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

2. Use the LocalAuthenticationContext:

```
LocalAuthenticationContext.evaluatePolicy(.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 is set to true (E.g.: call success methods)

Data storage & privacy requirements

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>System credential storage facilities are used appropriately to store sensitive data, such as PII, user credentials or cryptographic keys.</td>
</tr>
<tr>
<td>2.2</td>
<td>No sensitive data should be stored outside of the app container or system credential storage facilities.</td>
</tr>
<tr>
<td>2.3</td>
<td>No sensitive data is written to application logs.</td>
</tr>
<tr>
<td>2.4</td>
<td>No sensitive data is shared with third parties unless it is a necessary part of the architecture.</td>
</tr>
<tr>
<td>2.5</td>
<td>The keyboard cache is disabled on text inputs that process sensitive data.</td>
</tr>
<tr>
<td>2.6</td>
<td>No sensitive data is exposed via IPC mechanisms.</td>
</tr>
<tr>
<td>2.7</td>
<td>No sensitive data, such as passwords or pins, is exposed through the user interface.</td>
</tr>
<tr>
<td>2.8</td>
<td>No sensitive data is included in backups generated by the mobile operating system.</td>
</tr>
<tr>
<td>2.9</td>
<td>The app removes sensitive data from views when backgrounded.</td>
</tr>
<tr>
<td>2.10</td>
<td>The app does not hold sensitive data in memory longer than necessary, and memory is cleared explicitly after use.</td>
</tr>
<tr>
<td>2.11</td>
<td>The app enforces a minimum device-level security policy, such as requiring the user to set a device passcode.</td>
</tr>
<tr>
<td>2.12</td>
<td>The app educates the user about the types of personally identifiable information processed, as well as security best practices the user should follow in using the app.</td>
</tr>
</tbody>
</table>
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
  - Platform interaction
  - 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
Agenda

• Introduction into the current state of the MSTG.
  – Issues
  – Milestones
  – Project Page
• Release process.
• Contribution guidelines.
• Outline of the activities planned for this week.
• How to get started
• Notes for contributors & reviewers
How to get started

1. Fork the repo you want to work on:
   – https://github.com/OWASP/owasp-mstg
   – https://github.com/OWASP/owasp-masvs

2. Setup local git at your system (preferrably with ssh keys)

3. Clone the repo to your system

4. Add the upstream repo (MASVS/MSTG) to your repo configuration

5. Create a branch, start your work, commit and push when ready

6. Pull request and ask our attention to speed it up 😊.

7. Review feedback? Parse it as soon as you can, so you can move forward and add your stuff.
Notes for contributors

• For any tool: focus on the installation, basics and guide towards its own (online) help

• For every feature of a platform: focus on its working, best practices, pitfalls and insecurities
Notes for reviewers

- Really bad PR? Ask to get in touch and work together
- Ok-ish PR with big errors: comments
- Small issues: try to comment
- In parallel: PR for your own fixes, but keep it to a ### level per PR to cause less conflicts
FINAL NOTES:

• ALL EVENING sessions are in villa 708!