MASA Lab Workshop

H1 2022

The following is a summary of the MASA Lab Workshop in which five Authorized Lab partners met to review and clarify the acceptance criteria for the OWASP MASVS and MSTG Level 1 requirements used for MASA validation. The clarifications below are intended to be supplemental to MSTG and do not replace existing guidelines. All MASA parties agree on the decisions and the scope provided in this document, as it defines the baseline for security testing and provides consistent results. This feedback was provided to OWASP for possible improvements in the upcoming MASVS 2.0 refactor. Developers may use this document as a reference when engaging with any of the MASA lab partners.

Architecture, Design and Threat Modeling

All Level 1 requirements have been removed as these requirements are difficult for typical pentesting style assessments. Further, it is expected that OWASP will be moving all requirements in this section to Level 2.

Data Storage and Privacy Requirements

MSTG-STORAGE-1

The primary focus of this specification is to protect the credentials and cryptographic keys used by the application, and thus all credentials and cryptographic keys shall use the system provided credential storage. Sensitive data does not need to be stored in system credential storage. This would include refresh tokens. For auth tokens that have a short TTL (i.e. 48 hours) can be stored within the sandbox using something like shared preferences.

MSTG-STORAGE-2

Whereas MSTG-STORAGE-1 is focused on protecting credentials and keys, MSTG-STORAGE-2 is focused on protecting sensitive data. As modern Android devices utilize file based encryption and provide isolation between applications through sandboxing, it is not required that sensitive data be further encrypted when stored in the sandbox. However, some developers may choose to encrypt some sensitive data. If sensitive data is stored in external storage on the device it shall always be encrypted using the keystore or a mechanism to decrypt the data with user supplied information.

MSTG-STORAGE-3

No further clarification needed

MSTG-STORAGE-5

No further clarification needed

MSTG-STORAGE-7

Passwords, pins, and credit card information must be properly masked. These fields may be displayed to the user in clear text through an explicit action, such as clicking a show password button.
Other sensitive data, such as bank account numbers and other PII which may be abused for identity theft, must also be protected. Displaying this type of sensitive data must require explicit user actions, but does not need to be masked after the action. For example, a banking application may have a default page which displays account balances. However, displaying the bank account number in clear text may be displayed only after the user clicks a show account information button.

MSTG-STORAGE-12

Over the last year, Google Play introduced Safety Labels as an upcoming platform requirement. This requirement has been updated to check for validity of developer disclosures on the label. Labs will perform passive observation for a limited set of data types collected during the active testing window. Passive observation will include looking for data types being transmitted over the network if possible. Results will be flagged in the validation report along the lines of:
Pass: We observed that the developer included a privacy policy in the app and had a data safety section disclosed in their Play Store listing. During our passive testing we did not observe any data type that was not declared being transmitted.
Fail: We observed that the developer included | did not include a privacy policy in the app and had a data safety section disclosed in their Play Store listing. During our passive testing we identified one or more data types that were being transmitted.

Cryptography Requirements

MSTG-CRYPTO-1

This requirement is looking to ensure that the app developer does not leverage hard coded keys to encrypt any sensitive data. This only applies to security-relevant code / use cases such as primary functionally that provides things like authentication or protection of sensitive data. A developer may use deprecated or weak crypto for non-security relevant code / use cases such as analytics data which doesn't contain sensitive data.

MSTG-CRYPTO-2

The objective of this requirement is for the developer to leverage a well known trusted implementation of cryptographic primitives within their app. If a developer provides their own crypto, they will be required to demonstrate evidence it meets a security threshold.

MSTG-CRYPTO-3

Where the previous requirement looked at this more holistically, this requirement specifies industry best practices for crypto primitives are defined as:
Key length
- Reference standards here: https://www.keylength.com/en/4/
- Hashing algorithms: SHA-224 or greater
- Public key encryption algorithms: Key length no less than 2048 bits
- Digital signature: Key length no less than 2048 bits
- Elliptic Curve: Key length no less than 224 bits

Weak key generation
- Suitable functions: PBKDF2, Argon2, bcrypt, scrypt, Catena, Lyra2, Makwa, yescrypt, Balloon
- NIST requires at least 10,000 iterations for PBKDF2. However, best practice recommends more.
- OWASP now recommends the following for PBKDF2 (based on IETF guidance current through March ‘22):
- The work factor for PBKDF2 is implemented through an iteration count, which should be set differently based on the internal hashing algorithm used.
- PBKDF2-HMAC-SHA1: 720,000 iterations
- PBKDF2-HMAC-SHA256: 310,000 iterations
- PBKDF2-HMAC-SHA512: 120,000 iterations

Use of ECB
- Don’t use ECB.

Static IV
- IV should be random and unpredictable
- Reiterating the above, IVs should never be reused"

MSTG-CRYPTO-4

App developers should not leverage any outdated or deprecated cryptographic protocols or algorithms.If deprecated protocols / algorithms are found they must not be used for a security relevant context. This excludes APK signing.

MSTG-CRYPTO-5

No further clarification needed

MSTG-CRYPTO-6

As with previous crypto requirements this test case should only focus on security sensitive functionality .

Authentication and Session Management Requirements

MSTG-AUTH-1

The intention of this requirement is for primary services offered by the app so this would not apply to things such as an analytics service. If remote services do not exist this should be marked as N/A.

MSTG-AUTH-2

No further clarification needed

MSTG-AUTH-3

When apps use OAuth 2.0 the scope should be limited to just the token. Regardless of which method of authentication is being used (token-based vs stateless), the method's best practices must be ensured so that the authentication follows the industry best practices so that it is protected against tampering, enveloping, replay, null cipher, and key substitution attacks.

MSTG-AUTH-4

Focus on clearing out from the mobile app side and ensure app meets requirements as outlined in AUTH-7

MSTG-AUTH-5

This is focused on password complexity requirements (i.e. 8 characters or more) and should not prevent the user from setting a more complex password. This should be enforced by the remote endpoint and not on the client side.

MSTG-AUTH-6

Captcha and any other out of band or 2FA counts as a delay mechanism.

MSTG-AUTH-7

For stateless this should be based on an appropriate TTL which should be 24-48 hours. For stateful use the document TTL but it's for the session token.

Network Communication Requirements

MSTG-NETWORK-1

No further clarification needed

MSTG-NETWORK-2

As long as the developer is using platform provided defaults and is on Target API level 29 or higher which defaults to TLS 1.3 the TLS settings are in line with current best practices.

MSTG-NETWORK-3
No further clarification needed

Platform Interaction Requirements

MSTG-PLATFORM-1
No further clarification needed

MSTG-PLATFORM-2
This requirement as written is a great best practice but is very large in scope and therefore we have aligned on a revised scope for testing:

- Verify the app can only load the minimum set of URLs needed for the app to function (e.g. the app can't load an arbitrary site unless it is a browser etc)
- Fragment Injection: If android:targetSdkVersion < 19 class must contain implementation of isValidFragment to protect against Fragment injection
- UI input: reviewer should take a sample approach if necessary to test the UI elements most likely to affect security if they accept data which is not validated
- IPC input: reviewer should take a sample approach if necessary to test the IPC entry points most likely to affect security if they accept data which is not validated
- App crashes should only be looked at from a security context (i.e. does the crash lead to sensitive data being logged or a VPN app disrupting its service)

MSTG-PLATFORM-3
No further clarification needed

MSTG-PLATFORM-4
No further clarification needed

Code Quality and Build Setting Requirements

MSTG-CODE-1

APK signing shall be performed using V2 or V3.

MSTG-CODE-2
No further clarification needed
MSTG-CODE-3
No further clarification needed

MSTG-CODE-4

Look for debug code path in the production app and excessive logging in logcat or app logs. If an app has some "beta" functionality that may be considered test code as long as it's documented it would pass.

Having Strict Mode enabled cannot be directly considered as FAIL, only in cases where this could be used by an attacker (i.e: by disclosing sensitive data in logs) this can be considered as FAIL.
In addition to this, other scenarios where debug code could be considered as security relevant when an non-expected (or non-documented) debug code disclose sensitive information or security mechanisms can be disabled. (i.e: certificate pinning)

MSTG-CODE-5

Developers shall provide the list of dependencies used in their application to the lab.

When a vulnerability affecting a third party library or component is identified, proper justification shall be provided by the developer including if the conditions for potential real exploitation are met and the reasons explaining why the component cannot be updated. (i.e: not available or affecting supply chain at a lower level)

NOTE: A real example could be a vulnerability affecting a component under specific configurations that are not applicable to the app.
NOTE 2: Even if an update exists for a specific component, there could be a potential risk associated with being an "early adopter" if the risk associated with a potential vulnerability is not relevant.

MSTG-CODE-9

Since most Android applications are Java-based, they are immune to buffer overflow vulnerabilities. Nevertheless, a buffer overflow vulnerability may still be applicable when you're using the Android NDK; therefore, consider secure compiler settings. When some security mechanisms are not in place for NDK libraries, justification for the decision shall be provided.

ADDITIONAL CONSIDERATIONS

Testing for apps with a hardware component

- Labs should focus on the existing scope of app security, authentication and connectivity to the app backend service. In the case a mobile application works with a piece of hardware, the authorized lab can ask the developer to self-attest the security implemented between the app and the hardware which can be included in the validation report, however that is out of scope for the MASA assessment.

Attendees


BISHOPFOX
DEKRA
leviathan
security group
nccgroup
NowSecure™