MASVS

Mobile Application Security Verification Standard

v2.0.0

Sven Schleier
Bernhard Mueller
Jeroen Beckers
Carlos Holguera
Jeroen Willemsen

OWASP

OWASP MAS
Mobile Application Security Project

OWASP Mobile Application Security Verification Standard (MASVS)

v2.0.0 released April 1, 2023

Release Notes: https://github.com/OWASP/owasp-masvs/releases/tag/v2.0.0

The OWASP MASVS, available online at https://mas.owasp.org/MASVS, is part of the OWASP Mobile Application Security (MAS) Project which also provides the OWASP Mobile Application Security Testing Guide (MASTG) v1.5.0

https://mas.owasp.org

Cover design by Carlos Holguera

Contents

Foreword

Technological revolutions can happen quickly. Less than a decade ago, smartphones were clunky devices with little keyboards - expensive playthings for tech-savvy business users. Today, smartphones are an essential part of our lives. We've come to rely on them for information, navigation and communication, and they are ubiquitous both in business and in our social lives.

Every new technology introduces new security risks, and keeping up with those changes is one of the main challenges the security industry faces. The defensive side is always a few steps behind. For example, the default reflex for many was to apply old ways of doing things: Smartphones are like small computers, and mobile apps are just like classic software, so surely the security requirements are similar? But it doesn't work like that. Smartphone operating systems are different from desktop operating systems, and mobile apps are different from web apps. For example, the classical method of signature-based virus scanning doesn't make sense in modern mobile OS environments: Not only is it incompatible with the mobile app distribution model, it's also technically impossible due to sandboxing restrictions. Also, some vulnerability classes, such as buffer overflows and XSS issues, are less relevant in the context of run-of-the-mill mobile apps than in, say, desktop apps and web applications (exceptions apply).

Over time, our industry has gotten a better grip on the mobile threat landscape. As it turns out, mobile security is all about data protection: Apps store our personal information, pictures, recordings, notes, account data, business information, location and much more. They act as clients that connect us to services we use on a daily basis, and as communications hubs that processes each and every message we exchange with others. Compromise a person's smartphone and you get unfiltered access to that person's life. When we consider that mobile devices are more readily lost or stolen and mobile malware is on the rise, the need for data protection becomes even more apparent.

A security standard for mobile apps must therefore focus on how mobile apps handle, store and protect sensitive information. Even though modern mobile operating systems like iOS and Android offer mature APIs for secure data storage and communication, those have to be implemented and used correctly in order to be effective. Data storage, inter-app communication, proper usage of cryptographic APIs and secure network communication are only some of the aspects that require careful consideration.

An important question in need of industry consensus is how far exactly one should go in protecting the confidentiality and integrity of data. For example, most of us would agree that a mobile app should verify the server certificate in a TLS exchange. But what about certificate or public key pinning? Does not doing it result in a vulnerability? Should this be a requirement if an app handles sensitive data, or is it maybe even counter-productive? Do we need to encrypt data stored in SQLite databases, even though the OS sandboxes the app? What is appropriate for one app might be unrealistic for another. The MASVS is an attempt to standardize these requirements using profiles that fit different threat scenarios.

Furthermore, the appearance of root malware and remote administration tools has created awareness of the fact that mobile operating systems themselves have exploitable flaws, so containerization strategies are increasingly used to afford additional protection to sensitive data and prevent client-side tampering. This is where things get complicated. Hardware- backed security features and OS-level containerization solutions, such as Android Enterprise and Samsung Knox, do exist, but they aren't consistently available across different devices. As a band aid, it is possible to implement software-based protection measures - but unfortunately, there are no standards or testing processes for verifying these kinds of protections.

As a result, mobile app security testing reports are all over the place: For example, some testers report a lack of obfuscation or root detection in an Android app as "security flaw". On the other hand, measures like string encryption, debugger detection or control flow obfuscation aren't considered mandatory. However, this binary way of looking at things doesn't make sense because resilience is not a binary proposition: It depends on the particular client-side threats one aims to defend against. Software protections are not

useless, but they can ultimately be bypassed, so they must never be used as a replacement for security controls.

The overall goal of the MASVS is to offer a baseline for mobile application security, while also allowing for the inclusion of defense-in-depth measures and protections against client-side threats. The MASVS is meant to achieve the following:

- Provide requirements for software architects and developers seeking to develop secure mobile applications;
- Offer an industry standard that can be tested against in mobile app security reviews;
- Clarify the role of software protection mechanisms in mobile security and provide requirements to verify their effectiveness;
- Provide specific recommendations as to what level of security is recommended for different use-cases.

We are aware that 100% industry consensus is impossible to achieve. Nevertheless, we hope that the MASVS is useful in providing guidance throughout all phases of mobile app development and testing. As an open source standard, the MASVS will evolve over time, and we welcome any contributions and suggestions.

By Bernhard Mueller

About the Standard

OWASP MAS
Mobile Application Security Project

The OWASP Mobile Application Security Verification Standard (MASVS) is the industry standard for mobile application security. It provides a comprehensive set of security controls that can be used to assess the security of mobile apps across various platforms (e.g., Android, iOS) and deployment scenarios (e.g., consumer, enterprise). The standard covers the key components of the mobile app attack surface including storage, cryptography, authentication and authorization, network communication, interaction with the mobile platform, code quality and resilience against reverse engineering and tampering.

The OWASP MASVS is the result of years of community effort and industry feedback. We thank all the contributors who have helped shape this standard. We welcome your feedback on the OWASP MASVS at any time, especially as you apply it to your own organization and mobile app development projects. Getting inputs from a variety of mobile app developers will help us improve and update the standard which is revised periodically based on your inputs and feedback.

You can provide feedback using GitHub Discussions in the OWASP MASVS repo https://github.com/OWASP/owasp-masvs/discussions, or contact the project leads directly https://mas.owasp.org/contact/.

The OWASP MASVS and MASTG are trusted by the following platform providers and standardization, governmental and educational institutions. Learn more.

android CREST NIST National Institute of Standards and Technology U.S. Department of Commerce Bundesamt für Sicherheit in der Informationstechnik ioXt internet of secure things ...

Authors

Sven Schleier

Sven is specialised in penetration testing and application security and has guided numerous projects to build security in from the start. He strongly believes in knowledge sharing and is speaking worldwide at meetups and conferences, is an adjunct professor and is conducting hands-on workshops about mobile app security to penetration testers, developers and students.

Carlos Holguera

Carlos is a mobile security research engineer with many years of hands-on experience in security testing for mobile apps and embedded systems such as automotive control units and IoT devices. He is passionate about reverse engineering and dynamic instrumentation of mobile apps and is continuously learning and sharing his knowledge.

Jeroen Beckers

Jeroen is a mobile security lead responsible for quality assurance on mobile security projects and for R&D on all things mobile. Ever since his master's thesis on Android security, Jeroen has been interested in mobile devices and their (in)security. He loves sharing his knowledge with other people, as is demonstrated by his many talks & trainings at colleges, universities, clients and conferences.

Bernhard Mueller

Bernhard is a cyber security specialist with a talent for hacking systems of all kinds. During more than a decade in the industry, he has published many zero-day exploits for software. BlackHat USA commended his pioneering work in mobile security with a Pwnie Award for Best Research.

Jeroen Willemsen

Jeroen is a principal security architect with a passion for mobile security and risk management. He has supported companies as a security coach, a security engineer and as a full-stack developer. He loves explaining technical subjects: from security issues to programming challenges.

Contributors

All of our contributors are listed in the Contributing section of the OWASP MAS website:

https://mas.owasp.org/contributing/

Donators

While both the MASVS and the MASTG are created and maintained by the community on a voluntary basis, sometimes outside help is required. We therefore thank our donators for providing the funds to be able to hire technical editors. Note that their donation does not influence the content of the MASVS or MASTG in any way. The Donation Packages are described on the OWASP MAS Website.

Changelog

All our Changelogs are available online at the OWASP MASVS GitHub repository, see the Releases page:

https://github.com/OWASP/owasp-masvs/releases

Copyright and License

The Mobile Application Security Verification Standard

The Mobile Application Security Verification Standard (MASVS) is a comprehensive security standard developed by the Open Worldwide Application Security Project (OWASP). This framework provides a clear and concise set of guidelines and best practices for assessing and enhancing the security of mobile applications. The MASVS is designed to be used as a metric, guidance, and baseline for mobile app security verification, serving as a valuable resource for developers, application owners, and security professionals.

The objective of the MASVS is to establish a high level of confidence in the security of mobile apps by providing a set of controls that address the most common mobile application security issues. These controls were developed with a focus on providing guidance during all phases of mobile app development and testing, and to be used as a baseline for mobile app security verification during procurement.

By adhering to the controls outlined in the OWASP MASVS, organizations can ensure that their mobile applications are built with security in mind, reducing the risk of security breaches and protecting sensitive user data. Whether used as a metric, guidance, or baseline, the OWASP MASVS is an invaluable tool for enhancing the security of mobile applications.

The OWASP MASVS is a living document and is regularly updated to reflect the changing threat landscape and new attack vectors. As such, it's important to stay up-to-date with the latest version of the standard and adapt security measures accordingly.

Mobile Application Security Model

The standard is divided into various groups that represent the most critical areas of the mobile attack surface. These control groups, labeled **MASVS-XXXXX**, provide guidance and standards for the following areas:

- **MASVS-STORAGE:** Secure storage of sensitive data on a device (data-at-rest).
- **MASVS-CRYPTO:** Cryptographic functionality used to protect sensitive data.
- **MASVS-AUTH:** Authentication and authorization mechanisms used by the mobile app.
- **MASVS-NETWORK:** Secure network communication between the mobile app and remote endpoints (data-in-transit).
- **MASVS-PLATFORM:** Secure interaction with the underlying mobile platform and other installed apps.
- **MASVS-CODE:** Security best practices for data processing and keeping the app up-to-date.
- **MASVS-RESILIENCE:** Resilience to reverse engineering and tampering attempts.

Each of these control groups contains individual controls labeled **MASVS-XXXXX-Y**, which provide specific guidance on the particular security measures that need to be implemented to meet the standard.

Mobile Application Security Profiles

The MAS project has traditionally provided three verification levels (L1, L2 and R), which were revisited during the MASVS refactoring in 2023, and have been reworked as "security testing profiles" and moved over to the OWASP MASTG. These profiles are now aligned with the NIST OSCAL (Open Security Controls Assessment Language) standard, which is a comprehensive catalog of security controls that can be used to secure information systems.

By aligning with OSCAL, the MASVS provides a more flexible and comprehensive approach to security testing. OSCAL provides a standard format for security control information, which allows for easier sharing and

reuse of security controls across different systems and organizations. This allows for a more efficient use of resources and a more targeted approach to mobile app security testing.

However, it is important to note that implementing these profiles fully or partially should be a risk-based decision made in consultation with business owners. The profiles should be tailored to the specific security risks and requirements of the mobile application being developed, and any deviations from the recommended controls should be carefully justified and documented.

Assumptions

When using the MASVS, it's important to keep in mind the following assumptions:

- The MASVS is not a substitute for following secure development best practices, such as secure coding or secure SDLC. These practices should be followed holistically in your development process and the MASVS complements them specifically for mobile apps.
- The MASVS assumes that you've followed the relevant standards of your industry and country for all elements of your app's ecosystem, such as backend servers, IoT, and other companion devices.
- The MASVS is designed to evaluate the security of mobile apps that can be analyzed statically by obtaining the app package, dynamically by running it on a potentially compromised device, and also considers any network-based attacks such as MITM.

While the OWASP MASVS is an invaluable tool for enhancing the security of mobile applications, it cannot guarantee absolute security. It should be used as a baseline for security requirements, but additional security measures should also be implemented as appropriate to address specific risks and threats to the mobile app.

Security Architecture, Design and Threat Modeling for Mobile Apps

> The OWASP MASVS assumes that best practices for secure architecture, design, and threat modeling have been followed as a foundation.

Security must be a top priority throughout all stages of mobile app development, from the initial planning and design phase to deployment and ongoing maintenance. Developers need to follow secure development best practices and ensure that security measures are prioritized to protect sensitive data, comply with policies and regulations, and identify and address security issues that can be targeted by attackers.

While the MASVS and MASTG focuses on controls and technical test cases for app security assessments, non-technical aspects such as following best practices laid out by OWASP Software Assurance Maturity Model (SAMM) or NIST.SP.800-218 Secure Software Development Framework (SSDF) for secure architecture, design, and threat modeling are still important. The MASVS can also be used as reference and input for a threat model to raise awareness of potential attacks.

To ensure that these practices are followed, developers can provide documentation or evidence of adherence to these standards, such as design documents, threat models, and security architecture diagrams. Additionally, interviews can be conducted to collect information on adherence to these practices and provide an understanding of the level of compliance with these standards.

Secure App Ecosystem

> The OWASP MASVS assumes other relevant security standards are also leveraged to ensure that all systems involved in the app's operation meet their applicable requirements.

Mobile apps often interact with multiple systems, including backend servers, third-party APIs, Bluetooth devices, cars, IoT devices, and more. Each of these systems may introduce their own security risks that must be considered as part of the mobile app's security design and threat modeling. For example, when interacting with a backend server, the OWASP Application Security Verification Standard (ASVS) should be used to ensure that the server is secure and meets the required security standards. In the case of Bluetooth devices, the app should be designed to prevent unauthorized access, while for cars, the app should be designed to protect the user's data and ensure that there are no safety issues with the car's operation.

Security Knowledge and Expertise

> The OWASP MASVS assumes a certain level of security knowledge and expertise among developers and security professionals using the standard. It's important to have a good understanding of mobile app security concepts, as well as the relevant tools and techniques used for mobile app security testing and assessment. To support this, the OWASP MAS project also provides the OWASP Mobile Application Security Testing Guide (MASTG), which provides in-depth guidance on mobile app security testing and assessment.

Mobile app development is a rapidly evolving field, with new technologies, programming languages, and frameworks constantly emerging. It's essential for developers and security professionals to stay current with these developments, as well as to have a solid foundation in fundamental security principles.

OWASP SAMM provides a dedicated "Education & Guidance" domain which aims to ensure that all stakeholders involved in the software development lifecycle are aware of the software security risks and are equipped with the knowledge and skills to mitigate these risks. This includes developers, testers, architects, project managers, executives, and other personnel involved in software development and deployment.

Applicability of the MASVS

By adhering to the MASVS, businesses and developers can ensure that their mobile app are secure and meet industry-standard security requirements, regardless of the development approach used. This is the case for downloadable apps, as the project was traditionally focused on, but the MAS resources and guidelines are also applicable to other areas of the business such as preloaded applications and SDKs.

Native Apps

Native apps are written in platform-specific languages, such as Java/Kotlin for Android or Objective-C/Swift for iOS.

Cross-Platform and Hybrid Apps

Apps based on cross-platform (Flutter, React Native, Xamarin, Ionic, etc.) and hybrid (Cordova, PhoneGap, Framework7, Onsen UI, etc.) frameworks may be susceptible to platform-specific vulnerabilities that don't exist in native apps. For example, some JavaScript frameworks may introduce new security issues that don't exist in other programming languages. It is therefore essential to follow the security best practices of the used frameworks.

The MASVS is agnostic to the type of mobile application being developed. This means that the guidelines and best practices outlined in the MASVS can be applied to all types of mobile apps, including cross-platform and hybrid apps.

Preloads

Preloaded apps are apps that are installed on a user's device at factory time and may have elevated privileges that leave users vulnerable to exploitative business practices. Given the large number of preloaded apps on an average user's device, it's important to measure their risk in a quantifiable way.

There are hundreds of preloads that may ship on a device, and as a result, automation is critical. A subset of MAS criteria that is automation-friendly may be a good basis.

SDKs

SDKs play a vital role in the mobile app value chain, supplying code developers need to build faster, smarter, and more profitably. Developers rely on them heavily, with the average mobile app using 30 SDKs, and 90% of code sourced from third parties. While this widespread use delivers significant benefits to developers, it also propagates safety and security issues.

SDKs offer a variety of functionality, and should be regarded as an individual project. You should evaluate how the MASVS applies to the used SDKs to ensure the highest possible security testing coverage.

Assessment and Certification

OWASP's Stance on MASVS Certifications and Trust Marks

OWASP, as a vendor-neutral not-for-profit organization, does not certify any vendors, verifiers or software.

All such assurance assertions, trust marks, or certifications are not officially vetted, registered, or certified by OWASP, so an organization relying upon such a view needs to be cautious of the trust placed in any third party or trust mark claiming (M)ASVS certification.

This should not inhibit organizations from offering such assurance services, as long as they do not claim official OWASP certification.

Guidance for Certifying Mobile Apps

The recommended way of verifying compliance of a mobile app with the MASVS is by performing an "open book" review, meaning that the testers are granted access to key resources such as architects and developers of the app, project documentation, source code, and authenticated access to endpoints, including access to at least one user account for each role.

It is important to note that the MASVS only covers the security of the mobile app (client-side). It does not contain specific controls for the remote endpoints (e.g. web services) associated with the app and they should be verified against appropriate standards, such as the OWASP ASVS.

A certifying organization must include in any report the scope of the verification (particularly if a key component is out of scope), a summary of verification findings, including passed and failed tests, with clear indications of how to resolve the failed tests. Keeping detailed work papers, screenshots or recording, scripts to reliably and repeatedly exploit an issue, and electronic records of testing, such as intercepting proxy logs and associated notes such as a cleanup list, is considered standard industry practice. It is not sufficient to simply run a tool and report on the failures; this does not provide sufficient evidence that all issues at a certifying level have been tested and tested thoroughly. In case of dispute, there should be sufficient supportive evidence to demonstrate that every verified control has indeed been tested.

Using the OWASP Mobile Application Security Testing Guide (MASTG)

The OWASP MASTG is a manual for testing the security of mobile apps. It describes the technical processes for verifying the controls listed in the MASVS. The MASTG includes a list of test cases, each of which map to a control in the MASVS. While the MASVS controls are high-level and generic, the MASTG provides in-depth recommendations and testing procedures on a per-mobile-OS basis.

Testing the app's remote endpoints is not covered in the MASTG. The OWASP Web Security Testing Guide (WSTG) is a comprehensive guide with detailed technical explanation and guidance for testing the security of web applications and web services holistically and can be used in addition to other relevant resources to complement the mobile app security testing exercise.

The Role of Automated Security Testing Tools

The use of source code scanners and black-box testing tools is encouraged in order to increase efficiency whenever possible. It is however not possible to complete MASVS verification using automated tools alone, since every mobile app is different. In order to fully verify the security of the app it is essential to understand

the overall architecture, business logic, and technical pitfalls of the specific technologies and frameworks being used.

Other Uses

As Detailed Security Architecture Guidance

One of the more common uses for the Mobile Application Security Verification Standard is as a resource for security architects. The two major security architecture frameworks, SABSA or TOGAF, are missing a great deal of information that is necessary to complete mobile application security architecture reviews. MASVS can be used to fill in those gaps by allowing security architects to choose better controls for issues common to mobile apps.

As a Replacement for Off-the-shelf Secure Coding Checklists

Many organizations can benefit from adopting the MASVS, by choosing one of the two levels, or by forking MASVS and changing what is required for each application's risk level in a domain-specific way. We encourage this type of forking as long as traceability is maintained, so that if an app has passed control 4.1, this means the same thing for forked copies as the standard evolves.

As a Basis for Security Testing Methodologies

A good mobile app security testing methodology should cover all controls listed in the MASVS. The OWASP Mobile Application Security Testing Guide (MASTG) describes black-box and white-box test cases for each verification control.

As a Guide for Automated Unit and Integration Tests

The MASVS is designed to be highly testable, with the sole exception of architectural controls. Automated unit, integration and acceptance testing based on the MASVS controls can be integrated in the continuous development lifecycle. This not only increases developer security awareness, but also improves the overall quality of the resulting apps, and reduces the amount of findings during security testing in the pre-release phase.

For Secure Development Training

MASVS can also be used to define characteristics of secure mobile apps. Many "secure coding" courses are simply ethical hacking courses with a light smear of coding tips. This does not help developers. Instead, secure development courses can use the MASVS, with a strong focus on the proactive controls documented in the MASVS, rather than e.g. the Top 10 code security issues.

MASVS-STORAGE: Storage

Mobile applications handle a wide variety of sensitive data, such as personally identifiable information (PII), cryptographic material, secrets, and API keys, that often need to be stored locally. This sensitive data may be stored in private locations, such as the app's internal storage, or in public folders that are accessible by the user or other apps installed on the device. However, sensitive data can also be unintentionally stored or exposed to publicly accessible locations, typically as a side-effect of using certain APIs or system capabilities such as backups or logs.

This category is designed to help developers ensure that any sensitive data intentionally stored by the app is properly protected, regardless of the target location. It also covers unintentional leaks that can occur due to improper use of APIs or system capabilities.

Controls

ID	Control
MASVS-STORAGE-1	The app securely stores sensitive data.
MASVS-STORAGE-2	The app prevents leakage of sensitive data.

MASVS-STORAGE-1

Control

The app securely stores sensitive data.

Description

Apps handle sensitive data coming from many sources such as the user, the backend, system services or other apps on the device and usually need to store it locally. The storage locations may be private to the app (e.g. its internal storage) or be public and therefore accessible by the user or other installed apps (e.g. public folders such as Downloads). This control ensures that any sensitive data that is intentionally stored by the app is properly protected independently of the target location.

MASVS-STORAGE-2

Control

The app prevents leakage of sensitive data.

Description

There are cases when sensitive data is unintentionally stored or exposed to publicly accessible locations; typically as a side-effect of using certain APIs, system capabilities such as backups or logs. This control covers this kind of unintentional leaks where the developer actually has a way to prevent it.

MASVS-CRYPTO: Cryptography

Cryptography is essential for mobile apps because mobile devices are highly portable and can be easily lost or stolen. This means that an attacker who gains physical access to a device can potentially access all the sensitive data stored on it, including passwords, financial information, and personally identifiable information. Cryptography provides a means of protecting this sensitive data by encrypting it so that it cannot be easily read or accessed by an unauthorized user.

The purpose of the controls in this category is to ensure that the verified app uses cryptography according to industry best practices, which are typically defined in external standards such as NIST.SP.800-175B and NIST.SP.800-57. This category also focuses on the management of cryptographic keys throughout their lifecycle, including key generation, storage, and protection. Poor key management can compromise even the strongest cryptography, so it is crucial for developers to follow the recommended best practices to ensure the security of their users' sensitive data.

Controls

ID	Control
MASVS-CRYPTO-1	The app employs current strong cryptography and uses it according to industry best practices.
MASVS-CRYPTO-2	The app performs key management according to industry best practices.

MASVS-CRYPTO-1

Control

The app employs current strong cryptography and uses it according to industry best practices.

Description

Cryptography plays an especially important role in securing the user's data - even more so in a mobile environment, where attackers having physical access to the user's device is a likely scenario. This control covers general cryptography best practices, which are typically defined in external standards.

MASVS-CRYPTO-2

Control

The app performs key management according to industry best practices.

Description

Even the strongest cryptography would be compromised by poor key management. This control covers the management of cryptographic keys throughout their lifecycle, including key generation, storage and protection.

MASVS-AUTH: Authentication and Authorization

Authentication and authorization are essential components of most mobile apps, especially those that connect to a remote service. These mechanisms provide an added layer of security and help prevent unauthorized access to sensitive user data. Although the enforcement of these mechanisms must be on the remote endpoint, it is equally important for the app to follow relevant best practices to ensure the secure use of the involved protocols.

Mobile apps often use different forms of authentication, such as biometrics, PIN, or multi-factor authentication code generators, to validate user identity. These mechanisms must be implemented correctly to ensure their effectiveness in preventing unauthorized access. Additionally, some apps may rely solely on local app authentication and may not have a remote endpoint. In such cases, it is critical to ensure that local authentication mechanisms are secure and implemented following industry best practices.

The controls in this category aim to ensure that the app implements authentication and authorization mechanisms securely, protecting sensitive user information and preventing unauthorized access. It is important to note that the security of the remote endpoint should also be validated using industry standards such as the OWASP Application Security Verification Standard (ASVS).

Controls

ID	Control
MASVS-AUTH-1	The app uses secure authentication and authorization protocols and follows the relevant best practices.
MASVS-AUTH-2	The app performs local authentication securely according to the platform best practices.
MASVS-AUTH-3	The app secures sensitive operations with additional authentication.

MASVS-AUTH-1

Control

The app uses secure authentication and authorization protocols and follows the relevant best practices.

Description

Most apps connecting to a remote endpoint require user authentication and also enforce some kind of authorization. While the enforcement of these mechanisms must be on the remote endpoint, the apps also have to ensure that it follows all the relevant best practices to ensure a secure use of the involved protocols.

MASVS-AUTH-2

Control

The app performs local authentication securely according to the platform best practices.

Description

Many apps allow users to authenticate via biometrics or a local PIN code. These authentication mechanisms need to be correctly implemented. Additionally, some apps might not have a remote endpoint, and rely fully on local app authentication.

MASVS-AUTH-3

Control

The app secures sensitive operations with additional authentication.

Description

Some additional form of authentication is often desirable for sensitive actions inside the app. This can be done in different ways (biometric, pin, MFA code generator, email, deep links, etc) and they all need to be implemented securely.

MASVS-NETWORK: Network Communication

Secure networking is a critical aspect of mobile app security, particularly for apps that communicate over the network. In order to ensure the confidentiality and integrity of data in transit, developers typically rely on encryption and authentication of the remote endpoint, such as through the use of TLS. However, there are numerous ways in which a developer may accidentally disable the platform secure defaults or bypass them entirely by utilizing low-level APIs or third-party libraries.

This category is designed to ensure that the mobile app sets up secure connections under any circumstances. Specifically, it focuses on verifying that the app establishes a secure, encrypted channel for network communication. Additionally, this category covers situations where a developer may choose to trust only specific Certificate Authorities (CAs), which is commonly referred to as certificate pinning or public key pinning.

Controls

ID	Control
MASVS-NETWORK-1	The app secures all network traffic according to the current best practices.
MASVS-NETWORK-2	The app performs identity pinning for all remote endpoints under the developer's control.

MASVS-NETWORK-1

Control

The app secures all network traffic according to the current best practices.

Description

Ensuring data privacy and integrity of any data in transit is critical for any app that communicates over the network. This is typically done by encrypting data and authenticating the remote endpoint, as TLS does. However, there are many ways for a developer to disable the platform secure defaults, or bypass them completely by using low-level APIs or third-party libraries. This control ensures that the app is in fact setting up secure connections in any situation.

MASVS-NETWORK-2

Control

The app performs identity pinning for all remote endpoints under the developer's control.

Description

Instead of trusting all the default root CAs of the framework or device, this control will make sure that only very specific CAs are trusted. This practice is typically called certificate pinning or public key pinning.

MASVS-PLATFORM: Platform Interaction

The security of mobile apps heavily depends on their interaction with the mobile platform, which often involves exposing data or functionality intentionally through the use of platform-provided inter-process communication (IPC) mechanisms and WebViews to enhance the user experience. However, these mechanisms can also be exploited by attackers or other installed apps, potentially compromising the app's security.

Furthermore, sensitive data, such as passwords, credit card details, and one-time passwords in notifications, is often displayed in the app's user interface. It is essential to ensure that this data is not unintentionally leaked through platform mechanisms such as auto-generated screenshots or accidental disclosure through shoulder surfing or device sharing.

This category comprises controls that ensure the app's interactions with the mobile platform occur securely. These controls cover the secure use of platform-provided IPC mechanisms, WebView configurations to prevent sensitive data leakage and functionality exposure, and secure display of sensitive data in the app's user interface. By implementing these controls, mobile app developers can safeguard sensitive user information and prevent unauthorized access by attackers.

Controls

ID	Control
MASVS-PLATFORM-1	The app uses IPC mechanisms securely.
MASVS-PLATFORM-2	The app uses WebViews securely.
MASVS-PLATFORM-3	The app uses the user interface securely.

MASVS-PLATFORM-1

Control

The app uses IPC mechanisms securely.

Description

Apps typically use platform provided IPC mechanisms to intentionally expose data or functionality. Both installed apps and the user are able to interact with the app in many different ways. This control ensures that all interactions involving IPC mechanisms happen securely.

MASVS-PLATFORM-2

Control

The app uses WebViews securely.

Description

WebViews are typically used by apps that have a need for increased control over the UI. This control ensures that WebViews are configured securely to prevent sensitive data leakage as well as sensitive functionality exposure (e.g. via JavaScript bridges to native code).

MASVS-PLATFORM-3

Control

The app uses the user interface securely.

Description

Sensitive data has to be displayed in the UI in many situations (e.g. passwords, credit card details, OTP codes in notifications). This control ensures that this data doesn't end up being unintentionally leaked due to platform mechanisms such as auto-generated screenshots or accidentally disclosed via e.g. shoulder surfing or sharing the device with another person.

MASVS-CODE: Code Quality

Mobile apps have many data entry points, including the UI, IPC, network, and file system, which might receive data that has been inadvertently modified by untrusted actors. By treating this data as untrusted input and properly verifying and sanitizing it before use, developers can prevent classical injection attacks, such as SQL injection, XSS, or insecure deserialization. However, other common coding vulnerabilities, such as memory corruption flaws, are hard to detect in penetration testing but easy to prevent with secure architecture and coding practices. Developers should follow best practices such as the OWASP Software Assurance Maturity Model (SAMM) and NIST.SP.800-218 Secure Software Development Framework (SSDF) to avoid introducing these flaws in the first place.

This category covers coding vulnerabilities that arise from external sources such as app data entry points, the OS, and third-party software components. Developers should verify and sanitize all incoming data to prevent injection attacks and bypass of security checks. They should also enforce app updates and ensure that the app runs up-to-date platforms to protect users from known vulnerabilities.

Controls

ID	Control
MASVS-CODE-1	The app requires an up-to-date platform version.
MASVS-CODE-2	The app has a mechanism for enforcing app updates.
MASVS-CODE-3	The app only uses software components without known vulnerabilities.
MASVS-CODE-4	The app validates and sanitizes all untrusted inputs.

MASVS-CODE-1

Control

The app requires an up-to-date platform version.

Description

Every release of the mobile OS includes security patches and new security features. By supporting older versions, apps stay vulnerable to well-known threats. This control ensures that the app is running on an up-to-date platform version so that users have the latest security protections.

MASVS-CODE-2

Control

The app has a mechanism for enforcing app updates.

Description

Sometimes critical vulnerabilities are discovered in the app when it is already in production. This control ensures that there is a mechanism to force the users to update the app before they can continue using it.

MASVS-CODE-3

Control

The app only uses software components without known vulnerabilities.

Description

To be truly secure, a full whitebox assessment should have been performed on all app components. However, as it usually happens with e.g. for third-party components this is not always feasible and not typically part of a penetration test. This control covers “low-hanging fruit” cases, such as those that can be detected just by scanning libraries for known vulnerabilities.

MASVS-CODE-4

Control

The app validates and sanitizes all untrusted inputs.

Description

Apps have many data entry points including the UI, IPC, the network, the file system, etc. This incoming data might have been inadvertently modified by untrusted actors and may lead to bypass of critical security checks as well as classical injection attacks such as SQL injection, XSS or insecure deserialization. This control ensures that this data is treated as untrusted input and is properly verified and sanitized before it's used.

MASVS-RESILIENCE: Resilience Against Reverse Engineering and Tampering

Defense-in-depth measures such as code obfuscation, anti-debugging, anti-tampering, etc. are important to increase app resilience against reverse engineering and specific client-side attacks. They add multiple layers of security controls to the app, making it more difficult for attackers to successfully reverse engineer and extract valuable intellectual property or sensitive data from it, which could result in:

- The theft or compromise of valuable business assets such as proprietary algorithms, trade secrets, or customer data
- Significant financial losses due to loss of revenue or legal action
- Legal and reputational damage due to breach of contracts or regulations
- Damage to brand reputation due to negative publicity or customer dissatisfaction

The controls in this category aim to ensure that the app is running on a trusted platform, prevent tampering at runtime and ensure the integrity of the app's intended functionality. Additionally, the controls impede comprehension by making it difficult to figure out how the app works using static analysis and prevent dynamic analysis and instrumentation that could allow an attacker to modify the code at runtime.

However, note that the lack of any of these measures does not necessarily cause vulnerabilities - instead, they add threat-specific additional protection to apps which must also fulfil the rest of the OWASP MASVS security controls according to their specific threat models.

Controls

ID	Control
MASVS-RESILIENCE-1	The app validates the integrity of the platform.
MASVS-RESILIENCE-2	The app implements anti-tampering mechanisms.
MASVS-RESILIENCE-3	The app implements anti-static analysis mechanisms.
MASVS-RESILIENCE-4	The app implements anti-dynamic analysis techniques.

MASVS-RESILIENCE-1

Control

The app validates the integrity of the platform.

Description

Running on a platform that has been tampered with can be very dangerous for apps, as this may disable certain security features, putting the data of the app at risk. Trusting the platform is essential for many of the MASVS controls relying on the platform being secure (e.g. secure storage, biometrics, sandboxing, etc.). This control tries to validate that the OS has not been compromised and its security features can thus be trusted.

MASVS-RESILIENCE-2

Control

The app implements anti-tampering mechanisms.

Description

Apps run on a user-controlled device, and without proper protections it's relatively easy to run a modified version locally (e.g. to cheat in a game, or enable premium features without paying), or upload a back-doored version of it to third-party app stores. This control tries to ensure the integrity of the app's intended functionality by preventing modifications to the original code and resources.

MASVS-RESILIENCE-3

Control

The app implements anti-static analysis mechanisms.

Description

Understanding the internals of an app is typically the first step towards tampering with it (either dynamically, or statically). This control tries to impede comprehension by making it as difficult as possible to figure out how an app works using static analysis.

MASVS-RESILIENCE-4

Control

The app implements anti-dynamic analysis techniques.

Description

Sometimes pure static analysis is very difficult and time consuming so it typically goes hand in hand with dynamic analysis. Observing and manipulating an app during runtime makes it much easier to decipher its behavior. This control aims to make it as difficult as possible to perform dynamic analysis, as well as prevent dynamic instrumentation which could allow an attacker to modify the code at runtime.