ZIMPERIUM®

How Zimperium Helps You Meet OWASP MASVS Resilience Requirements

What are MASVS Resilience Requirements?

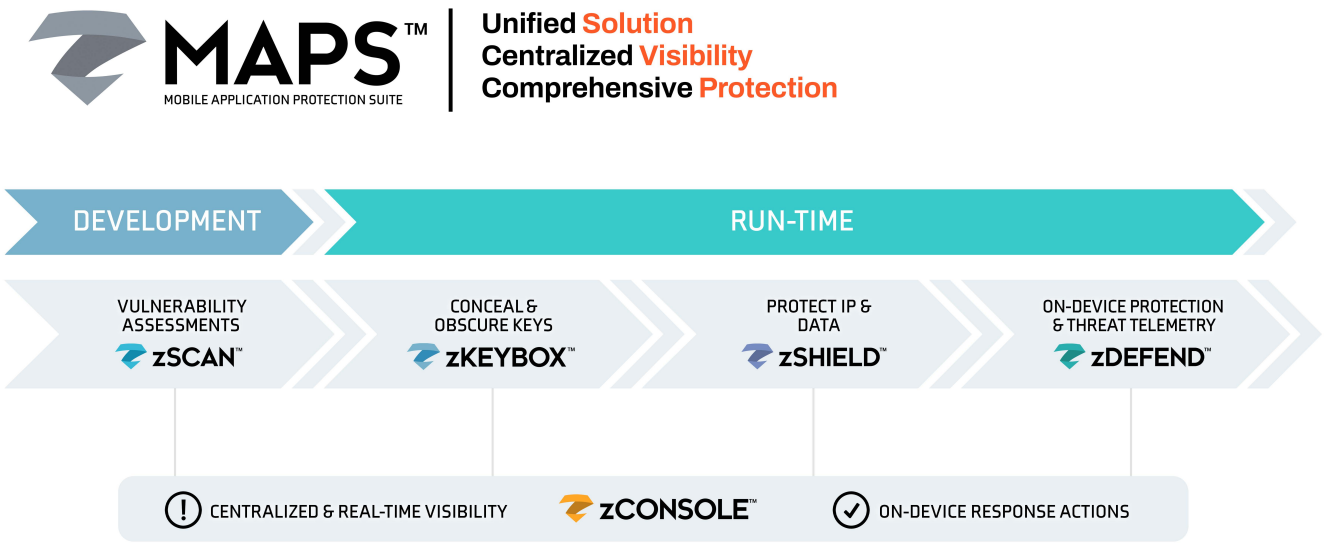
OWASP MASVS Resilience requirements are defense-in-depth measures recommended for apps that process or give access to sensitive data or functionality. They increase the app's resilience against reverse engineering, unauthorized tampering, and specific client-side attacks.

MSTG-RESILIENCE-1 The app detects and responds to the presence of a rooted or jailbroken device either by alerting the user or terminating the app.	**MSTG-RESILIENCE-2** The app prevents debugging and/or detects and responds to a debugger being attached. All available debugging protocols must be covered.	**MSTG-RESILIENCE-3** The app detects and responds to tampering with executable files and critical data within its own sandbox.	**MSTG-RESILIENCE-4** The app detects and responds to the presence of widely used reverse engineering tools and frameworks on the device.
MSTG-RESILIENCE-5 The app detects and responds to being run in an emulator.	**MSTG-RESILIENCE-6** The app detects and responds to tampering the code and data in its own memory space.	**MSTG-RESILIENCE-7** The app implements multiple mechanisms in each defense category (8.1 to 8.6).	**MSTG-RESILIENCE-8** The detection mechanisms trigger responses of different types, including delayed and stealthy responses.
MSTG-RESILIENCE-9 Obfuscation is applied to programmatic defenses, which in turn impede de-obfuscation via dynamic analysis.	**MSTG-RESILIENCE-10** The app implements a 'device binding' functionality using a device fingerprint derived from multiple properties unique to the device.	**MSTG-RESILIENCE-11** All executable files and libraries belonging to the app are either encrypted on the file level and/or important code and data segments inside the executables are encrypted or packed.	**MSTG-RESILIENCE-12** Obfuscation scheme is both appropriate for the particular task and robust against manual and automated de-obfuscation methods.
MSTG-RESILIENCE-13 Next to having solid hardening of the communicating parties, application-level payload encryption can be applied to further impede eavesdropping.			

What is Zimperium Mobile App Protection Suite (MAPS)?

Zimperium's Mobile Application Protection Suite consists of four products with a centralized dashboard to view threats and create response policies. It is the only unified platform that combines centralized visibility with comprehensive in-app protection, combining both inside-out and outside-in security approaches to help enterprises build and maintain secure mobile apps.

- **zScan:** Discover and fix compliance, privacy, and security issues within the development process before you publicly release your apps.
- **zKeyBox:** Protect confidential data by securing cryptographic keys with white-box cryptography so they cannot be discovered, extracted, or manipulated.
- **zShield:** Harden and protect the app with advanced obfuscation and anti-tampering functionality to protect the source code, intellectual property (IP), and data within the application.
- **zDefend:** Enable the mobile application to detect and proactively protect itself by taking actions on the end user's device, even without network connectivity.



How Does MAPS Help You Meet the Resilience Requirements?

The following table helps you better understand how Zimperium MAPS capabilities map to OWASP MASVS Resilience (MASVS-R) requirements and, in turn, aids you in achieving MASVS compliance.

MAPS Coverage Overview

ID	MASVS-R ID	zSCAN™	zDEFEND™	zSHIELD™	zKEYBOX™
8.1	MSTG-RESILIENCE-1	✅	✅	✅	
8.2	MSTG-RESILIENCE-2	✅	✅	✅	
8.3	MSTG-RESILIENCE-3	Manual	✅	✅	
8.4	MSTG-RESILIENCE-4	✅	✅	✅	
8.5	MSTG-RESILIENCE-5	✅	✅		
8.6	MSTG-RESILIENCE-6	Manual	✅		
8.7	MSTG-RESILIENCE-7	Manual	✅	✅	
8.8	MSTG-RESILIENCE-8	Manual	✅		
8.9	MSTG-RESILIENCE-9	Manual		✅	
8.10	MSTG-RESILIENCE-10	Manual	✅		
8.11	MSTG-RESILIENCE-11	Manual		✅	
8.12	MSTG-RESILIENCE-12	Manual	✅	✅	
8.13	MSTG-RESILIENCE-13	Manual	✅		✅

Detailed Summary

8.1	MSTG-RESILIENCE-1	The app detects and responds to the presence of a rooted or jailbroken device either by alerting the user or terminating the app.

zScan scans the app binary and identifies if jailbreak or root detection logic is present or missing.

zShield injects advanced detections and protections for rooted and jailbroken devices into mobile applications.

zDefend provides advanced behavioral machine learning (ML)-based protection, and updates can be made OTA without republishing the app.

The threats below, when detected, are reported to the centralized dashboard for alerting and analysis.

- Device Rooted/Jailbroken
- SE Linux Disabled
- System Tampering
- File System Modification

8.2	MSTG-RESILIENCE-2	The app prevents debugging and/or detects and responds to a debugger being attached. All available debugging protocols must be covered.

zScan reinforces coding best practices where:

- Debugger Check Enabled
- WebView Not Debuggable
- Debug App Disabled

zShield injects detections to identify when the debugger is attached to the application.

zDefend provides advanced behavioral ML-based detection, and updates can be made OTA without republishing the app.

The threats below, when detected, are reported to the centralized dashboard for alerting and analysis.

- App Debug Enabled
- USB Debugging Enabled
- Android Debug Bridge Apps Not Verified
- Device Attack / OS Tampering

8.3	MSTG-RESILIENCE-3	The app detects and responds to tampering with executable files and critical data within its own sandbox.

Insight: In order for attackers to tamper with files and critical data in its own sandbox, the device must be in the compromised state as per Android/iOS security design. zDefend SDK and zShield detects OS-level compromises, such as alerting the app to take response action.

zShield provides:

- Code and data integrity checks by inserting overlapping checkers
- App integrity verification against signatures/enterprise certificate of the binary to protect against re-package and re-signing attack

zDefend provides advanced behavioral ML-based detection of runtime tampering and injection of instrumentation frameworks.

The threats below, when detected, are reported to the centralized dashboard for alerting and analysis.

- App Tampering
- System Tampering
- Filesystem Changed
- SE Linux Disabled
- Elevation of Privileges

8.4	MSTG-RESILIENCE-4	The app detects and responds to the presence of widely used reverse engineering tools and frameworks on the device.
	zScan scans the app binary and identifies if best practices for preventing reverse engineering are present or missing • Static Data Exposure • Readable Method Names • No Code Obfuscation / Code Obfuscation Detected • Automatic Reference Counting (ARC) Disabled **zShield** • Provides real-time visibility into app tampering. • Provides code obfuscation/encryption techniques that make it difficult for static reverse engineering tools, such as Xposed and Frida. • Injects rule-based detection for reversing engineering tools and framework. **zDefend** detects the presence of reverse engineering or hooking tools by leveraging advanced behavioral ML-based runtime detection. Updates to keep up with new tools and techniques can be made OTA without needing to republish the app.	
8.5	MSTG-RESILIENCE-5	The app detects and responds to being run in an emulator.
	zScan scans the app binary and identifies if emulator detection logic is present or missing. **zDefend** detects popular emulators / device tampering, such as custom roms, by leveraging advanced behavioral ML-based runtime detection. Updates to keep up with new tools and techniques can be made OTA without needing to republish the application. The threats below, when detected, are reported to the centralized dashboard for alerting and analysis. • App Running on Emulator • Android Device - Possible Tampering • Android Device - Custom Rom	
8.6	MSTG-RESILIENCE-6	The app detects and responds to tampering the code and data in its own memory space.
	Insight: For attackers to tamper with the code in the app's own memory space, the device OS needs to be in a compromised state. **zDefend** SDK detects various indicators using our multi-layer machine learning detection engine. For more detail, see responses for 8.1, 8.3, and 8.4.	
8.7	MSTG-RESILIENCE-7	The app implements multiple mechanisms in each defense category (8.1 to 8.6). Note that resiliency scales with the amount, and diversity of the originality of the mechanisms used.
	See response to 8.1 - 8.6. **zShield** is recommended for MASVS - L1 + Resilience compliance. A combination of zShield and zDefend is recommended for MASVS L2 + Resilience compliance.	

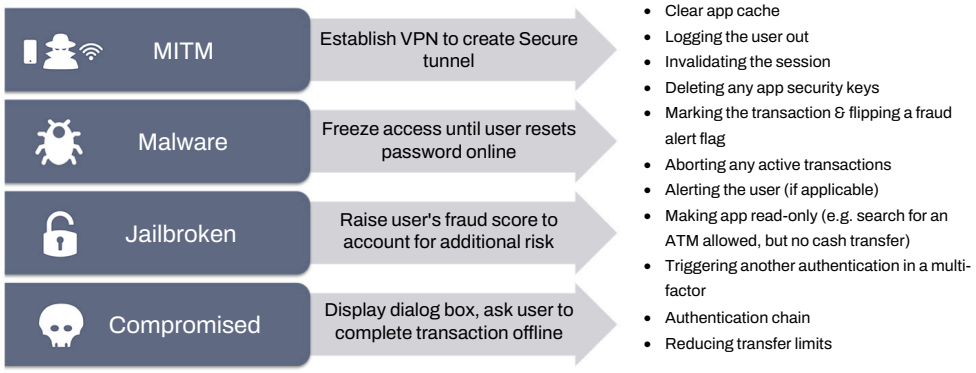
8.8	MSTG-RESILIENCE-8	The detection mechanisms trigger responses of different types, including delayed and stealthy responses.

zDefend provides the ability to detect and trigger an on-device response action to prevent the threat. The application developer builds the response action once the app owner and security team determine the response.

For example, when a device compromise is detected, the app can choose to alert the user and terminate the app.

The Zimperium team will also conduct a security workshop with the security and development team to offer best practices.

In-App Remediation Actions



8.9	MSTG-RESILIENCE-9	Obfuscation is applied to programmatic defenses, which in turn impede de-obfuscation via dynamic analysis.

zShield obfuscates the code during compile time. Here are key capabilities covered by the zShield solution. The bullet points below summarize the essential source code protection features from Zimperium.

- Android Obfuscation can be applied to both source (Java/Kotlin/Native) and binary level (apk, aab, aar)
- iOS Obfuscation can be applied to source code (Objective C, Swift, Native)

Android

- Code Obfuscation
- Integrity Protection
- Inlining of Static Functions
- String Literal Obfuscation
- Google Play Licensing Protection
- Integrity Protection of Android APK Packages
- Binary Packing
- Cross Checking of Shared Libraries
- Resource Encryption

iOS

- Code Obfuscation
- Integrity Protection
- Inlining of Static Functions
- String Literal Obfuscation
- Mach-O Signature Verification
- Objective-C Message Call Obfuscation
- Objective-C and Swift Metadata Encryption

8.10	MSTG-RESILIENCE-10	The app implements a 'device binding' functionality using a device fingerprint derived from multiple properties unique to the device.

Insight: Due to the recent OS changes on both iOS and Android, unique device ID is not exposed due to privacy concerns.

zDefend, when initialized, generates a UUID - unique device identifier. This is unique per application bundle install or reinstall. Zimperium SDK offers tracking IDs API, which allows applications to initialize and pass to the SDK a correlation ID that can tie the events reported on devices with user/transaction context.

The tracking IDs are made available on the management console so they can be used to bind multiple devices and filter events in the backend.

Tracking ID 1	6335797caa09f40016293cc2
Tracking ID 2	3B74E87A7F

8.11	MSTG-RESILIENCE-11	All executable files and libraries belonging to the app are either encrypted on the file level and/or important code and data segments inside the executables are encrypted or packed. Trivial static analysis does not reveal important code or data.

zShield provides encryption, obfuscation, and packing techniques of the application files, resources, and libraries. See 8.9 for more details.

8.12	MSTG-RESILIENCE-12	If the goal of obfuscation is to protect sensitive computations, an obfuscation scheme should be both appropriate for the particular task and robust against manual and automated de-obfuscation methods, considering currently published research. In addition, the effectiveness of the obfuscation scheme must be verified through manual testing. Note that hardware-based isolation features are preferred over obfuscation whenever possible.

Attackers can tackle reverse engineering and attack the application using online or offline techniques and tools. Zimperium **zDefend** and **zShield** cover both static and dynamic aspects of mobile in-app protection.

Static Attack (Offline)

Hackers download and transform the code into human readable format

- Code Logic / IP
- API Keys / Endpoints
- Remove / Inject Malicious Code
- Repackage the App

Dynamic Attack (Online)

Hacker "owns" the device and gathers app's behavior

- Root / Jailbreak / Compromise
- Network / MITM
- Privilege Escalation
- Hooking / Debugging
- Code Injection

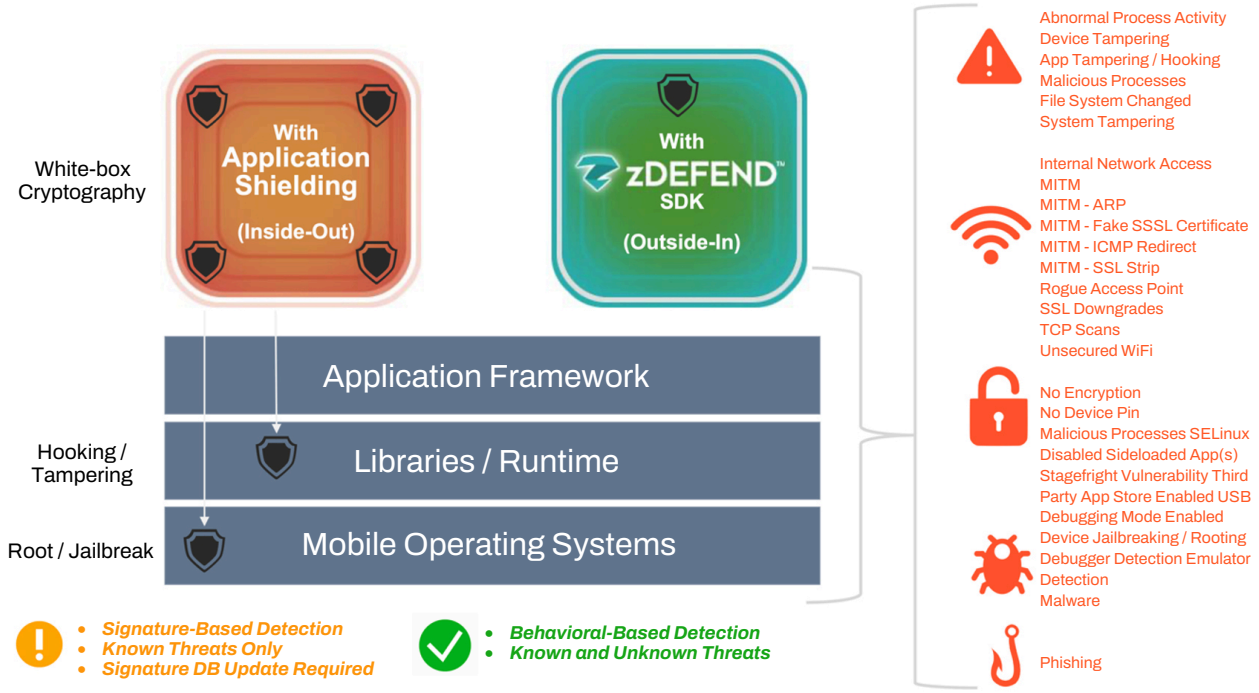
zShield provides strong code obfuscation and protects the integrity of apps and data, forcing the attacker to move to dynamic analysis.

zDefend SDK enables mobile apps to immediately detect when a user's device is compromised, when any network attacks are occurring, and even if malicious apps have been installed. Application development vendors can configure appropriate programmatic remedial actions when a threat is detected.

8.13	MSTG-RESILIENCE-13	Next to having solid hardening of the communicating parties, application-level payload encryption can be applied to further impede eavesdropping.
	zDefend covers both runtime detection of SSL / MITM attacks. Regardless of if the app has implemented certificate pinning, the attacker can always extract certificates from the app binary and perform an SSL bypass attack on the mobile device. zDefend protects the SSL pinning and detects network configuration and MITM SSL certificate threats on the device. **Note:** The network detections below require location and network access permission on iOS/Android. **zKeyBox** offers secure communication and storage of the payload. The solution leverages white-box cryptography to ensure that the cryptographic keys used to protect sensitive data are not exposed on a mobile device at rest, in memory, and in transit. zKeyBox supports a large set of cipher/algorithms and APIs to provide an in-depth defense approach to data security on mobile devices. • Native API • Java API • JavaScript API • TLS API (OpenSSL v3 Provider) • Secure Database API (SQLite) • DUKPT (Derived Unique Key Per Transaction) API • Secure PIN Entry Threats reported to centralized dashboard: • MITM • MITM Fake SSL Certificates • MITM ARP • Rogue Access Point	

Conclusion

Zimperium's Mobile Application Protection Suite (MAPS) helps you meet these OWASP MASVS Resilience requirements. It is the only unified platform that combines centralized visibility with comprehensive in-app protection, combining both inside-out and outside-in security approaches to help enterprises build and maintain secure mobile apps. To learn more about how Zimperium can help your organization deliver secure mobile apps, contact us.



Protected App

Injection + SDK

ZIMPERIUM

Learn more at: zimperium.com
Contact us at: 844.601.6760 | info@zimperium.com

Zimperium, Inc
4055 Valley View, Dallas, TX 75244