OWASP Open Web Application Security Project | Standard

Mobile AppSec Verification

Version 1.1

(German Translation)

Project leaders: Sven Schleier and Jeroen Willemsen

Inhaltsverzeichnis

Security Requirements

Appendix

Changelog

This document is automatically generated at Sat Dec 29 2018 08:52:45 GMT+0100 (Midden-Europese standaardtijd)

WIP - international release

- Added thank you note for buyers of the book.
- Added missing authentication link & updated broken authentication link.
- Fixed swap of 4.7 and 4.8.
- First international release!
- Fixes in Spanish translation. Translation is now in sync with English (1.1.2).
- Fixes in Russian translation. Translation is now in sync with English (1.1.2).
- Added first release of German, French and Japanese!
- Simplified document for ease of translation.
- Added automation steps for automated releases.

1.1.0 14 July 2018:

The following changes are part of release 1.1:

- Requirement 2.6 "The clipboard is deactivated on text fields that may contain sensitive data." was removed.
- Requirement 2.2 "No sensitive data should be stored outside of the app container or system credential storage facilities." was added.
- Requirement 2.1 was reworded to "System credential storage facilities are used appropriately to store sensitive data, such as PII, user credentials or cryptographic keys.".

1.0 12 January 2018:

The following changes are part of release 1.0:

- Delete 8.9 as the same as 8.12
- Made 4.6 more generic
- Minor fixes (typos etc.)

Vorwort

Von Bernhard Mueller, OWASP Mobile Project.

Technologierevolutionen können sehr schnell gehen. Noch vor 10 Jahren waren Smartphones klobige Geräte mit kleinen Tastaturen - teure Spielgeräte für technikverliebte Geschäftsleute. Heute sind Smartphones ein wesentlicher Bestandteil unseres Lebens. Wir hängen inzwischen von ihnen ab und nutzen sie zur Information, Navigation und Kommunikation. Sie sind allgegenwärtig - in der Geschäftswelt und im privaten Umfeld.

Jede neue Technologie bringt neue Sicherheitsrisiken mit sich. Eine der größten Herausforderungen für alle die im Sicherheitsumfeld arbeiten, ist Schritt zu halten mit der fortlaufenden Weiterentwicklung. Das blaue Team (Verteidigerseite) liegt dabei immer einige Schritte hinter dem roten Team (Angreiferseite) zurück. Viele unterlagen deshalb zu Beginn dem Trugschluss, altbekannte Vorgehensweisen direkt auf das mobile Umfeld anzuwenden im Sinne von: "Smartphones sind im Prinzip nur kleine PCs und mobile Apps sind wie klassische Software - folglich gelten die gleichen Sicherheitsanforderungen". Jedoch funktioniert das so nicht. Mobile Betriebssysteme unterscheiden sich von Desktop-Betriebssystemen und mobile Apps sich von Webanwendungen. Zum Beispiel macht die klassische Methode Virenprüfungen auf Basis von Signaturen durchzuführen in modernen mobilen Betriebssystemumgebungen keinen Sinn. Es ist nicht nur inkompatibel zur Art und Weise wie mobile Apps verteilt werden sondern darüber hinaus aufgrund des Betriebssystemdesigns (Sandbox-Isolation) von Apps technisch überhaupt nicht umsetzbar. Bei gewöhnlichen 08/15-Apps sind zudem eine Reihe von Schwachstellen wie Buffer overflows und XSS weniger relevant als in Webanwendungen und Desktopanwendungen. (Ausnahmen möglich) Über die Zeit hat sich das gewandelt und die Securitybranche hat nun eine bessere Vorstellung für die mobile Bedrohungslage entwickelt.

Der Datenschutz steht im Mittelpunkt mobiler Sicherheit. Apps speichern persönliche Informationen, Fotos, Aufnahmen, Notizen, Kontodaten, Geschäftsdaten, Standortdaten und viele mehr. Sie dienen uns als tägliche Helfer um uns mit Daten- und Kommunikationsdiensten zu verbinden, die jede einzelne der Nachrichten, die wir mit anderen austauschen, verarbeiten. Übernimm die Kontrolle über ein Smartphone und Du erhälst ungefilterten Zugriff auf das Leben der betreffenden Person. Wenn man sich vorstellt, dass Mobilgeräte durchaus verloren gehen oder gestohlen werden können und sich mobile Schadsoftware im Aufwärtstrend befindet wird der hohe Bedarf an Datenschutz nochmals mehr deutlich. Ein Sicherheitsstandard für mobile Apps muss deshalb klar im Fokus haben, wie mobile Apps mit sensiblen Daten umgehen, sie verarbeiten und speichern.

Moderne mobile Betriebssysteme wie iOS und Android bieten gute APIs für sichere Datenspeicherung und Kommunikation - Diese müssen allerdings auch implementiert und dabei korrekt genutzt werden um effektiv zu sein. Datenspeicherung, Interprozesskommunikation, korrekte Nutzung kryptographischer APIs und sichere Netzwerkkommunikation sind nur ein paar der Aspekte die eine sorgfältige Betrachtung erfordern.

Eine wichtige Frage die man sich stellen muss, ist, wie weit und wie umfangreich Schutzmaßnahmen umzusetzen sind und wo die Grenze liegt. Zum Beispiel würden die meisten zustimmen, dass eine mobile App das Serverzertifikat bei einem TLS-Handshake prüfen muss. Was ist jedoch mit SSL-Zertifikat-Pinning ? Führt es direkt zu einer Schwachstelle, es nicht umzusetzen? Sollte dies eine Anforderung sein, wenn eine App sensible Daten verarbeitet oder ist es eventuell sogar contra-produktiv? Müssen wir Daten in einer SQLite-Datenbank verschlüsseln, wenn doch das Betriebssystem bereits ein Sandbox-Konzept mitbringt? Was für die eine App angemessen ist, kann für eine andere App unrealistisch sein. Der MASVS ist ein Versuch, diese Anforderungen zu standardisieren und Prüf-Level einzuführen die zu unterschiedlichen Bedrohungszenarien passen.

Darüber hinaus hat das Erscheinen von Root-Malware und Remote-Administrations-Tools gezeigt, dass mobile Betriebssysteme selbst ausnutzbare Schwachstellen haben. Um Client-seitige Manipulation zu verhindern und den Schutz sensibler Daten zu erhöhen, kommen vermehrt zusätzliche Container-basierte Isolations-Technologien zum Einsatz. An dieser Stelle wird es kompliziert. Es gibt Hardware-basierte Sicherheitsmechanismen und

Betriebssystemseitige Container-Lösungen wie Android for Work und Samsung Knox aber nicht durchgängig für alle Geräte. Als Notlösung sind Software-basierte Schutzmechanismen möglich jedoch existieren leider keine Standards sowie Testprozesse zu deren Prüfung.

Als Folge davon existieren zahlreiche mobile Sicherheitstestberichte bei denen die Tester fehlende Obfuskierung oder fehlende Root-Erkennung in einer Android-App als "Sicherheitslücke" werten. Auf der anderen Seite werden Maßnahmen wie Verschlüsselung von Zeichenketten, Erkennung von Debuggern oder Kontrollfluss-Obfuskierung nicht als verpflichtend erachtet. Letztlich ergibt dieses "schwarz-weiss Denken" keinen Sinn, denn Resilienz ist keine binäre Eigenschaft sondern hängt davon ab gegen welche konkreten Client-seitigen Bedrohungen man sich schützen möchte. Software-Schutzmaßnahmen zur Erhöhung der Resilienz bringen gewissen Nutzen jedoch können sie definitiv umgangen werden und dürfen deshalb niemals als Ersatz für Sicherheitsmaßnahmen dienen.

Das übergeordnete Ziel des MASVS ist es mit dem MASVS-L1 ein solides Basis-Sicherheitsniveau, mit dem MASVS-L2 ein erhöhtes Sicherheitsniveau sowie kontextbezogen weitere Defense-in-Depth-Maßnahmen gegen Client-seitige Bedrohungen (MASVS-R) für mobile App-Sicherheit anzubieten.

Folgendes soll mit dem MASVS erreicht werden:

- Bereitstellen von Sicherheitsanforderungen für Softwarearchitekten und Entwickler um sichere mobile Applikationen zu entwickeln;
- Definition eines Branchenstandards gegen den mobile Apps in Sicherheits-Reviews getestet werden können;
- Schärfung und Klarstellung der Rolle von Schutzmechanismen in der mobilen Sicherheit und Definition von Anforderungen um deren Effektivität zu prüfen;
- Aussprechen von individuellen Empfehlungen zur Anwendung von mobilen Sicherheitsmechanismen für verschiedene Anwendungsfälle.

Wir sind uns bewußt dass ein 100%-iger Konsens innerhalb der Sicherheitsbranche unmöglich zu erreichen ist. Nichtsdestotrotz hoffen wir dass der MASVS eine nützliche Hilfestellung für alle Phasen mobiler Softwareentwicklung und Tests darstellt. Als offener Standard soll der MASVS sich über die Zeit weiterentwickeln und wir begrüßen jede Unterstützung und Vorschläge.

Mobile Application Security Verification Standard

Über den Standard

Willkommen beim Mobile Application Security Verification Standard (MASVS) 1.1. Der MASVS ist eine Community-Initiative mit dem Ziel ein Rahmenwerk von Security-Anforderungen für Design, Entwicklung und Test von mobilen Apps unter iOS und Android zu etablieren.

Der MASVS vereint Community Engagement und Feedback aus der Praxis. Wir gehen davon aus dass der Standard sich über die Zeit weiter entwickelt und begrüßen jedes Feedback aus der Community. Der beste Weg mit uns in Kontakt zu treten ist der OWASP Mobile Project Slack Channel: https://owasp.slack.com/messages/.

Nutzerkonten können unter folgender URL angelegt werden: https://owasp.slack.com/join.

Copyright and License

Project Lead	Lead Author	Mitwirkende and Reviewer	Übersetzung ins deutsche
Sven Schleier & Jeroen Willemsen	Bernhard Mueller	Alexander Antukh, Mesheryakov Aleksey, Bachevsky Artem, Jeroen Beckers, Vladislav Chelnokov, Ben Cheney, Stephen Corbiaux, Manuel Delgado, Ratchenko Denis, Ryan Dewhurst, Tereshin Dmitry, Christian Dong, Oprya Egor, Ben Gardiner, Rocco Gränitz, Henry Hu, Sjoerd Langkemper, Vinícius Henrique Marangoni, Martin Marsicano, Roberto Martelloni, Gall Maxim, Rio Okada, Abhinav Sejpal, Stefaan Seys, Yogesh Shamrma, Prabhant Singh, Nikhil Soni, Anant Shrivastava, Francesco Stillavato, Romuald SZKUDLAREK, Abdessamad Temmar, Koki Takeyama, Chelnokov Vladislav	Rocco Gränitz, Sven Schleier (Review)

Dieses Dokument basiert auf einem Fork des OWASP Application Security Verification Standard verfasst von Jim Manico.

Sponsoren

Obwohl beide Projekte, der MASVS und der MSTG, auf freiwilliger Basis im Rahmen der Community erarbeitet wurden und weiter gepflegt werden, ist manchmal ein wenig Hilfe von außen nötig. Wir danken deshalb den Sponsoren mit deren Hilfe wir technische Editoren aquirieren können. Der Inhalt des MASVS oder MSTG ist in keinster Weise durch etwaige Sponsoren beeinflusst. Eine nähere Beschreibung der Sponsoren-Pakete befindet sich sich im OWASP Projekt Wiki.

Honourable Benefactor

Als nächstes möchten wir uns beim OWASP Bay Area Chapter für das Sponsoring bedanken. Zum Schluss möchten wir uns bei allen bedanken, die das Buch von Leanpub gekauft und uns auf diese Weise gesponsert haben.

Der Mobile Application Security Verification Standard

Der MASVS kann genutzt werden um das Sicherheitsniveau von mobilen Apps nachweisen zu können. Die Anforderungen wurden auf Basis folgender Ziele entwickelt:

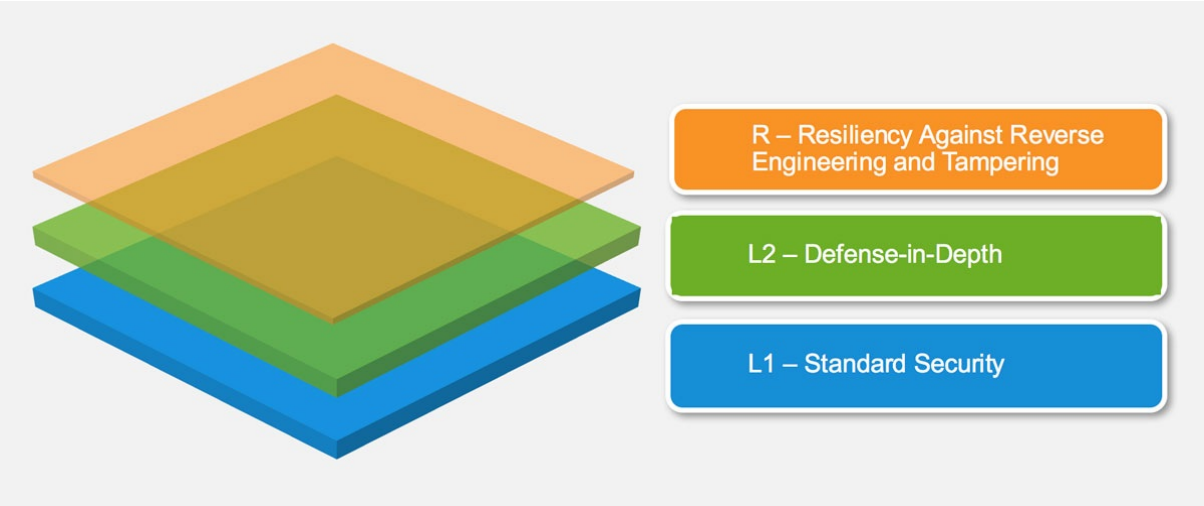
- Nutzung als Metrik - Um Entwicklern und Applikations-Verantwortlichen einen Security Standard anzubieten gegen den existierende mobile Apps verglichen werden können;
- Nutzung als Hilfestellung - Um Hilfe zu bieten in allen Phasen mobiler App-Entwicklung und Tests.
- Nutzung bei Beschaffung/Kauf von mobilen Apps - Um als Baseline zur Prüfung der Security von mobilen Apps zu dienen.

Das Mobile AppSec Model

Der MASVS definiert zwei strikte Prüf-Level (L1 and L2), sowie eine Reihe von Sicherheitsmaßnahmen zur Robustheit gegen Reverse Engineering (MASVS-R). Diese sind flexibel, d.h. adaptierbar an ein App-spezifisches Threat Model. MASVS-L1 and MASVS-L2 enthalten generische Sicherheitsanforderungen und werden für alle mobilen Apps (L1) und Apps die hochsensible Daten verarbeiten (L2) empfohlen. MASVS-R deckt zusätzliche Schutzmaßnahmen ab die anzuwenden sind um Client-seitigen Threats vorzubeugen.

Eine App die alle Anforderungen aus MASVS-L1 erfüllt, folgt Security Best Practices und vermeidet damit typische Schwachstellen. MASVS-L2 fügt weitere Defense-In-Depth Maßnahmen wie SSL-Pinning hinzu um ein erhöhtes Schutzniveau gegen komplexere Angriffe zu bieten. Dabei gilt die Annahme, dass das mobile Betriebssystem intakt und der Endnutzer nicht als potentieller Angreifer betrachtet wird. Die Anforderungen aus dem MASVS-R ganz oder teilweise zu erfüllen, hilft dabei spezifische client-seitige Threats (böswilliger Nutzer o. kompromittiertes Betriebssystem) zu verhindern bzw. zu erschweren.

Achtung - Die Schutzmaßnahmen enthalten in MASVS-R und beschrieben im OWASP Mobile Testing Guide können letztlich alle umgangen werden und dürfen nicht als Ersatz für Sicherheitsmaßnahmen (L1/L2) genutzt werden. Stattdessen sind sie dazu gedacht, um zusätzliche Threat-spezifische Schutzmaßnahmen zu Apps hinzuzufügen, die bereits die MASVS Anforderungen L1 oder L2 erfüllen.



Dokumentstruktur

Der erste Teil des MASVS enthält eine Beschreibung des Security Modells und der Prüf-Level, gefolgt von Empfehlungen zur Nutzung des Standards in der Praxis. Die detaillierten Sicherheitsanforderungen und ihre Einordnung in die Prüf-Level befinden sich im zweiten Teil. Die Anforderungen wurden in acht Kategorien (V1 bis V8), basierend auf technischen Zielen/Scopes, eingeteilt. Die folgende Nomenklatur wird durchgehend im MASVS und MSTG genutzt:

- *Anforderungs Kategorie:* MASVS-Vx, z.B. MASVS-V2: Datenspeicherung und Datenschutz
- *Anforderung:* MASVS-Vx.y, z.B. MASVS-V2.2: "Keine sensiblen Daten werden in Applikations Logs geschrieben."

Prüf-Level im Detail

MASVS-L1: Standard Security

Eine mobile App die MASVS-L1 entspricht, erfüllt Mobile Application Best Practices. Sie erfüllt Basis-Anforderungen in Bezug auf Codequalität, Umgang mit sensiblen Daten und Interaktion mit dem mobilen Umfeld. Es gibt einen Testprozess um die Effektivität der Security Maßnahmen zu prüfen. Dieser Level eignet sich für alle mobilen Applikationen.

MASVS-L2: Defense-in-Depth

MASVS-L2 führt erweiterte Sicherheitsmaßnahmen ein, die über die Standard Anforderungen hinausgehen. Um L2 zu erfüllen, muss ein Threat Model existieren und Security muss ein integraler Teil der App-Architektur sein. Dieser Level ist angedacht für alls Apps die sensible Daten verarbeiten wie z.B. Apps für mobiles Bezahlen.

MASVS-R: Resilienz gegen Reverse Engineering and Manipulation

Die App hat state-of-the-art Security und ist resilient (robust) gegen spezifische, klar definierte Client-seitige Angriffe wie Manipulation, Modifizierung oder Reverse Engineering um sensiblen Quellcode oder Daten zu extrahieren. Solch eine App nutzt Hardware Security Funktionen oder ausreichend starke und überprüfbare Software-Schutzmaßnahmen. MASVS-R ist geeignet für Apps die hoch sensible Daten verarbeiten, um geistiges Eigentum zu schützen oder um eine App manipulationssicher zu machen.

Empfohlene Nutzung

Nach vorheriger Bewertung des Risikos und des erforderlichen Sicherheitsniveaus können Apps gegen MASVS L1 oder L2 geprüft werden. L1 ist geeignet für alle mobilen Apps, während L2 für Apps mit mehr sensiblen Daten oder sensibler Funktionalität empfohlen wird. MASVS-R (oder Teile davon) können genutzt werden um *zusätzlich zu bereits implementierten Sicherheitsfunktionen* die Resilienz der App gegen spezifische Threats wie Repackaging oder Extraktion von sensiblen Daten zu prüfen.

Insgesamt gibt es folgende Prüfkombinationen:

- MASVS-L1
- MASVS-L1+R
- MASVS-L2
- MASVS-L2+R

Die Kombinationen stellen verschiedene Graduierungen der Security und Resilienz dar. Das Ziel ist Flexibilität: Eine Gaming-App z.B. wird aus Usability-Gesichtspunkten darauf verzichten, MASVS-L2 Maßnahmen wie 2-Faktor Authentifizierung zu nutzen hat jedoch aus Business-Sicht hohen Schutzbedarf gegen Code-Manipulation (Tampering).

Welche Prüfvariante wählen?

Die Anforderungen aus MASVS L2 zu implementieren erhöht die Sicherheit - dies kann sich jedoch negativ auf die Endnutzerakzeptanz auswirken (klassischer Kompromiss). Zeitgleich steigen die Entwicklungskosten.

Beispiele

MASVS-L1

- Geeignet für alle mobilen Apps. MASVS-L1 benennt Security Best Practices mit akzeptablen Auswirkungen auf Entwicklungskosten und Nutzererlebnis. Sollte für jede App genutzt werden die keinen höheren Level erfordert.

MASVS-L2

- Gesundheitswesen: Mobile Apps die personenbezogene Daten speichern die für Identitätsdiebstahl, Zahlungsbetrug und eine Reihe anderer Betrugsvorhaben genutzt werden können. Zu den für den US-Gesundheitsbereich geltenden Compliance-Regeln gehören der Health Insurance Portability and Accountability Act (HIPAA), Datenschutz, Security und Vorgaben/Regeln zur Benachrichtigung bei Datenpannen sowie Patientensicherheit.
- Banken/Finanzwesen: Apps mit Zugriff auf hochsensible Daten wie Kreditkarten, personenbezogene Daten oder die Zahlungsflüsse erlauben. Diese Apps erfordern erweiterte Security Maßnahmen um Betrug zu vermeiden. Finanz-Apps müssen Compliance-Anforderungen aus dem Payment Card Industry Data Security Standard (PCI DSS), Gramm Leech Bliley Act und Sarbanes-Oxley Act (SOX) erfüllen.

MASVS L1+R

- Mobile Apps für die der Schutz des geistigen Eigentums geschäftskritisch ist. Die Resilienz-Maßnahmen aus MASVS-R dienen dazu, für Angreifer den Aufwand zu erhöhen um an den Original-Quellcode zu gelangen und Manipulation der Apps (Tampering/Cracking) zu erschweren.
- Gaming Industrie: Spiele mit hohem Bedarf Modding und Cheating zu verhindern, wie Online-Games. Cheating ist ein großes Problem in Online-Spielen, da eine hohe Anzahl von Cheatern die regulären Nutzer verärgern und ein erfolgreiches Game letztlich zum Scheitern bringen kann. MASVS-R bietet Basis-Maßnahmen um den Aufwand für Cheater zu erhöhen.

MASVS L2+R

- Finanzwesen: Online Banking Apps die Nutzern Geldtransfers erlauben und bei denen Techniken wie Code-Injektion oder Instrumentierung auf kompromittierten Geräten Risiken darstellen. In diesem Fall können Maßnahmen aus dem MASVS-R genutzt werden um Manipulationen zu erschweren und den Aufwand für Malware Autoren zu erhöhen.
- Alle mobilen Apps die sensible Daten auf dem mobilen Gerät speichern und gleichzeitig eine hohe Kompatibilität zu diversen Geräten und Betriebssystemversionen bieten müssen. In diesem Fall können Resilienz-Maßnahmen als defense-in-depth genutzt werden um den Aufwand für Angreifer, die sensible Daten extrahieren wollen, zu erhöhen.

Bewertung and Zertifizierung

OWASP's Standpunkt zu MASVS Zertifizierungen und Gütesiegeln

OWASP ist eine herstellerunabhängige Non-Profit-Organisation und führt keine Zertifizierungen von Herstellern, Prüfern oder Software durch.

OWASP distanziert sich von derartigen Bescheinigungen, Qualitätssiegeln oder Zertifikaten. Diese sind in keinem Fall durch OWASP freigegeben, registriert oder zertifiziert. Eine Organisation/Unternehmen sollte vorsichtig und misstrauisch gegenüber derartigen Gütesiegeln sein.

Dies sollte Organisationen nicht davon abhalten Dienstleistungen auf Basis des (M)ASVS Standard anzubieten, solange keine offiziellen OWASP Zertifikate ausgestellt werden.

Anleitung zur Zertifizierung von mobilen Apps

Der empfohlene Weg um eine mobile App gegen den MASVS zu prüfen ist das offene Review-Format. Dabei erhalten Tester Zugriff auf Schlüsselressourcen wie Architekten und App-Entwickler, Projekt-Dokumentation, Quellcode und authentifizierten Zugriff auf API-Endpunkte (mindestens ein Nutzeraccount für jede Rolle)

Es ist wichtig zu erwähnen, dass der MASVS nur die Security auf der client-seitigen mobilen App und der Netzwerkkommunikation zwischen App und API-Endpunkt(en) sowie einige Basisanforderungen in Bezug auf Nutzerauthentifizierung und Session Management abdeckt. Er enthält darüber hinaus keine spezifischen Security-Anforderungen für remote Services z.B. Webservices die von der App genutzt werden. Jedoch schreibt MASVS V1 vor, dass API-Endpunkte von einem übergreifenden Threat Model berücksichtigt und gegen entsprechende Standards wie ASVS geprüft werden müssen.

Eine Zertifizierungsstelle muss in jedem Report den Scope der Überprüfung (insbesondere wenn eine Schlüsselkomponente out of scope ist), eine Zusammenfassung der Prüfungs-Findings, inklusive einer Auflistung aller erfolgreich durchgeführten sowie gescheiterten Tests sowie klare Lösungshinweise zu allen gescheiterten Tests aufführen. Eine Sammlung detaillierter Arbeitspapiere, Screenshots oder Demo-Videos, Skripte um eine gefundene Schwachstelle zuverlässig und wiederholt zu exploiten aber auch elektronische Aufzeichnungen/Logs wie Traffic-Mitschnitte durch einen Proxy und zugehörige Notizen wie z.B. eine ToDo-Liste oder Cleanup-Liste aufzubewahren gilt als "Best Practice". Es ist unzureichend einfach nur ein Tool zu starten und die Findings zu berichten. Stattdessen ist es wichtig Beweise zu liefern, dass alle geforderten Themenpunkte gründlich getestet wurden. Für den Fall von Streitigkeiten sollten ausreichend Nachweise vorliegen um darlegen zu können, dass jede zu prüfende Anforderung tatsächlich getestet wurde.

Nutzung des OWASP Mobile Security Testing Guide (MSTG)

Der OWASP MSTG ist eine Anleitung zum Testen der Sicherheit von mobilen Apps. Der Guide beschreibt den technischen Prozess zur Überprüfung der Anforderungen aus dem MASVS. Der MSTG enthält eine Liste von Testfällen. Jeder Testfall referenziert eine einzelne Anforderung im MASVS. Während der MASVS eher grobe und generische Anforderungen enthält, bietet der MSTG detaillierte Empfehlungen und Testprozeduren je mobilem Betriebssystem.

Die Rolle von Werkzeugen für automatisierte Security Tests

Die Nutzung von Quellcode-Scannern und Black-Box-Test-Werkzeugen erhöht die Effizienz und wird klar empfohlen. Allerdings ist eine vollständige MASVS Überprüfung allein mit automatisierten Werkzeugen unmöglich. Jede mobile App ist unterschiedlich und ein Verständnis der Gesamtarchitektur, der Geschäftslogik und technischer Fallstricke genutzter Technologien und Frameworks ist eine zwingende Voraussetzung um die Sicherheit einer App zu prüfen.

Andere Anwendungsfälle

Als detaillierter Security-Architektur-Guide

Der MASVS ist eine wertvolle Ressource für Security Architekten. Den 2 wichtigsten Security-Architektur-Frameworks SABSA und TOGAF fehlen wesentliche Teile um Security-Architektur-Reviews für mobile Apps durchführen zu können. Der MASVS kann genutzt werden um diese Lücken zu schließen und ist eine wertvolle Hilfe bei der Auswahl geeigneter Security Maßnahmen im Bereich mobiler Apps.

Als Ersatz für pauschale Secure Coding Checklisten

Viele Organisationen können davon profitieren, den MASVS zu adaptieren indem Sie einen geeigneten Level wählen oder den MASVS als Ausgangsbasis nutzen (Forking) und die Inhalte entsprechend zum Risikoprofil und Domänen-Kontext der jeweiligen App anpassen. Dieser Weg des Forkens wird empfohlen wobei darauf zu achten ist, die Nachverfolgbarkeit der einzelnen Themenpunkte zu gewährleisten. So sollte z.B. Anforderung 4.1 auch nach zukünftigen Änderungen die gleiche semantische Bedeutung behalten.

Als Basis für Security Tests

Eine gute Testmethodik für Security Tests mobiler Apps sollte alle Anforderungen aus dem MASVS abdecken. Der OWASP MSTG beschreibt Black-box und White-box Testfälle für jede Anforderungen aus dem MASVS.

Als Vorgabe für automatisierte Unit- und Integrationstests

Um die Anzahl von Findings in der Vorproduktions-Phase zu reduzieren sollten bereits während der Entwicklung automatische Unit-, Integrations- und Akzeptanztests durchgeführt werden. Um dies zu unterstützen wurde der MASVS mit starkem Fokus auf Testbarkeit entwickelt. Eine Ausnahme bilden die Anforderungen aus dem Bereich Architektur. Teams die die Anforderungen aus dem ASVS (über automatisierte Tests) umsetzen erhöhen dadurch nicht nur die Qualität der entwickelten Apps sondern verbessern auch die Security Awareness der Entwickler.

Zum Training für sichere Software-Entwicklung

Der MASVS kann darüber hinaus genutzt werden um die Merkmale zu beschreiben, die eine sichere mobile App haben sollte. Viele Kurse für "sichere Softwareentwicklung" sind simple Kurse über Hacking-Techniken und bieten kaum wertvolle Tipps für Entwickler. Kurse die auf Basis des MASVS proaktive Schutz-Maßnahmen in den Vordergrund stellen, anstatt z.B. die Top 10 Schwachstellen zu behandeln, bieten wesentlichen Mehrwert für Entwickler.

V1: Anforderungen an Architektur, Design und Bedrohungsanalysen

Zielsetzung

In einer perfekten Welt würde Security über alle Phasen der Softwareentwicklung berücksichtigt werden. In der Realität wird Security jedoch meist erst spät im Entwicklungsprozess berücksichtigt. Neben den technischen Maßnahmen fordert der MASVS auch organisatorische Maßnahmen die sicherstellen, dass Security in der Planungsphase sowie beim Design der mobilen App entsprechend Berücksichtigung gefunden hat. Für jede Komponente der App muss der fachliche Funktionsumfang und Sicherheitsfunktionen klar definiert und bekannt sein. Die meisten Apps kommunizieren mit entfernten Schnittstellen (API-Endpunkten). Deshalb müssen auch für diese API-Endpunkte angemessene Security Standards umgesetzt sein. Der isolierte Test einer mobilen App ohne die Endpunkte ist unzureichend!

Die Kategorie "V1" enthält Sicherheitsanforderungen an die Architektur und das Design einer App. Aufgrund dessen ist dies die einzige Kategorie die nicht auf technische Testfälle in den OWASP Mobile Testing Guide referenziert. Um Themen wie Bedrohungsanalyse, sichere Softwareentwicklungsprozesse und Schlüssel-Management abzudecken, sollten Anwender des MASVS die jeweiligen OWASP Projekte und/oder Standards wie die unten verlinkten Dokumente berücksichtigen.

Security Anforderungen

Die Anforderungen für MASVS-L1 und MASVS-L2 sind nachfolgend aufgelistet.

#	Beschreibung	L1	L2
1.1	Alle Komponenten der mobilen App sind identifiziert und für den Betrieb der App erforderlich.	✓	✓
1.2	Sicherheitsfunktionen sind niemals nur auf Client-Seite implementiert sondern immer auch im entsprechenden entfernten API-Endpunkt.	✓	✓
1.3	Es existiert eine Architekturübersicht über die mobile App und alle verbundenen API-Endpunkte und Security wurde in der Gesamtarchitektur berücksichtigt.	✓	✓
1.4	Alle sensiblen Daten im Kontext der mobilen App wurden klar identifiziert.	✓	✓
1.5	Für jede Komponente der App ist der angebotene fachliche Funktionsumfang und/oder Sicherheitsfunktionen/-mechanismen klar definiert.		✓
1.6	Für die mobile App und die genutzten API-Endpunkte wurde eine Bedrohungsanalyse durchgeführt und potentielle Bedrohungen und Gegenmaßnahmen identifiziert.		✓
1.7	Alle Sicherheitsfunktionen wurden in Form von zentralen Komponenten implementiert.		✓
1.8	Eine dedizierte Richtlinie zum Management von kryptographischen Schlüsseln (falls in der App genutzt) beschreibt den sicheren Umgang mit Schlüsseln über den gesamten Lebenszyklus, idealerweise basierend auf Standards wie NIST SP 800-57.		✓
1.9	Es gibt einen Mechanismus in der mobilen App um App-Aktualisierungen zu erzwingen.		✓
1.10	Security wird in allen Teilen des Softwareentwicklungszyklus berücksichtigt.		✓

Referenzen

Für weitere Informationen:

- OWASP Mobile Top 10: M10 - Extraneous Functionality: https://www.owasp.org/index.php/Mobile_Top_10_2016-M10-Extraneous_Functionality
- OWASP Security Architecture cheat sheet: https://www.owasp.org/index.php/Application_Security_Architecture_Cheat_Sheet
- OWASP Thread modelling: https://www.owasp.org/index.php/Application_Threat_Modeling
- OWASP Secure SDLC Cheat Sheet: https://www.owasp.org/index.php/Secure_SDLC_Cheat_Sheet
- Microsoft SDL: https://www.microsoft.com/en-us/sdl/
- NIST SP 800-57: https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-4/final

V2: Anforderungen an Datenspeicherung und Datenschutz

Zielsetzung

Der Schutz sensibler Daten wie Nutzer-Anmeldedaten und privaten Informationen ist ein Schwerpunkt im Bereich mobiler Sicherheit. Zum Beispiel können sensible Daten ungewollt anderen Apps auf dem mobilen Gerät zur Verfügung gestellt werden falls Betriebssystem-Mechanismen wie Interprozesskommunikation auf unsichere Weise genutzt werden. Datenlecks können ungewollt im Bereich Cloud-Daten-Speicherung, Backups oder dem Tastatur-Cache auftreten. Darüber hinaus können mobile Geräte leichter verloren gehen oder gestohlen werden. Die Wahrscheinlichkeit für physische Angriffe ist höher als bei anderen Geräten. In diesem Fall können zusätzliche Schutzmaßnahmen implementiert werden um den Zugriff auf sensible Daten zu erschweren. Im MASVS steht die App im Mittelpunkt und nicht das mobile Gerät. Security-Lösungen für mobiles Gerätemanagement (MDM) werden aus OWASP Sicht zwar zur Umsetzung von Security-Richtlinien im Unternehmens-Umfeld empfohlen, werden im MASVS jedoch nicht berücksichtigt.

Definition sensibler Daten

Sensible Daten im Kontext des MASVS sind jegliche Nutzer-Anmeldedaten sowie alle übrigen Daten die entsprechend ihrem Kontext sensibel sind:

- Personenbezogene Daten die für Identitätsdiebstahl genutzt werden können: Sozialversicherungsnummern, Kreditkarteninformationen, Bankdaten und Gesundheitsdaten;
- Hoch sensible Daten deren Kompromittierung zu hohem Reputationsverlust oder finanziellen Aufwänden führen würde: Vertragsinformationen, Informationen die durch Verschwiegenheitserklärungen geschützt sind, Management Informationen;
- Alle Daten die durch gesetzliche Bestimmungen oder aufgrund regulatorischer Vorgaben (Compliance) zu schützen sind.

Anforderungen

Ein Großteil von Datenpannen kann bereits durch Einhaltung einfacher Regeln vermieden werden. Die meisten der Sicherheitsmaßnahmen in dieser Kategorie sind deshalb für alle Prüf-Levels zwingend erforderlich.

#	Beschreibung	L1	L2
2.1	Die App speichert sensible Daten wie personenbezogene Daten, Anmeldedaten oder kryptographische Schlüssel unter Nutzung der vom jeweiligen Betriebssystem angebotenen sicheren Speichermechanismen.	✓	✓
2.2	Es werden keine sensiblen Daten außerhalb des App-Containers oder außerhalb des vom jeweiligen Betriebssystem angebotenen sicheren Speichermechanismus abgelegt.	✓	✓
2.3	Es werden keine sensiblen Daten in die Logfiles der App geschrieben.	✓	✓
2.4	Es werden keine sensiblen Daten mit Dritten geteilt - es sei denn dies wurde in der App-Architektur definiert und ist zur Erfüllung des Zwecks der App erforderlich.	✓	✓
2.5	Der Tastatur-Cache ist für alle sensiblen Texteingaben deaktiviert.	✓	✓
2.6	Es werden keine sensiblen Daten über Interprozesskommunikation zur Verfügung gestellt.	✓	✓
2.7	Es werden keine sensiblen Daten wie Passwörter oder Pins über die Benutzeroberfläche exponiert.	✓	✓
2.8	Es ist sichergestellt, dass betriebssystemgesteuerte Backups keine sensiblen App-Daten enthalten.		✓
2.9	Die App entfernt sensible Daten aus der aktuellen Ansicht wenn der Hintergrundmodus aktiviert wird.		✓
2.10	Die App hält sensible Daten nur solange wie nötig im Speicher und betroffene Speicherbereiche werden nach Nutzung explizit gelöscht.		✓
2.11	Die App erzwingt ein Minimum an Geräteschutz-Richtlinien wie das Definieren eines Gerätepassworts.		✓
2.12	Die App klärt den Nutzer über die Art und Weise der verarbeiteten personenbezogenen Daten auf und gibt dem Nutzer Security-Best-Practice-Empfehlungen zum Umgang mit der App.		✓

Referenzen

Der OWASP Mobile Security Testing Guide bietet detaillierte Anleitungen um die Anforderungen aus dieser Kategorie zu überprüfen.

- Für Android - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05d-Testing-Data-Storage.md
- Für iOS - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06d-Testing-Data-Storage.md

Weitere Informationen unter:

- OWASP Mobile Top 10: M2 - Insecure Data Storage: https://www.owasp.org/index.php/Mobile_Top_10_2016-M2-Insecure_Data_Storage
- CWE: https://cwe.mitre.org/data/definitions/922.html

V3: Anforderungen an Kryptographie

Zielsetzung

Kryptographie ist eine wesentlicher Eckpfeiler zum Schutz von Daten, die auf mobilen Geräten gespeichert werden. Es ist aber auch eine Kategorie bei der vieles falsch gemacht werden kann, besonders wenn man keine Standard-Konventionen einhält. Die Kategorie soll sicherstellen, dass eine überprüfte App Kryptographie-Best-Practices nach dem Stand der Technik nutzt:

- Nutzung bewährter Krypto-Bibliotheken,
- Richtige Auswahl und Konfiguration kryptographischer Primitive,
- Nutzung eines geeigneten Zufallszahlengenerators wenn kryptographisch sichere Zufallszahlen erforderlich sind.

Anforderungen

#	Beschreibung	L1	L2
3.1	Verschlüsselung in der App basiert nicht nur auf symmetrischer Kryptographie mit hart-codierten Schlüsseln.	✓	✓
3.2	Die App nutzt bewährte Implementierungen zur Umsetzung kryptographischer Primitive.	✓	✓
3.3	Die App nutzt kryptographische Primitive passend zum spezifischen Anwendungsfall, konfiguriert nach Best-Practice Vorgaben dem Stand der Technik entsprechend.	✓	✓
3.4	Die App nutzt keine kryptographischen Protokolle oder Algorithmen die allgemein als veraltet und unsicher gelten.	✓	✓
3.5	Die App verwendet einen kryptographischen Schlüssel für genau einen Zweck und nicht für mehrere Zwecke.	✓	✓
3.6	Alle Zufallswerte werden über einen ausreichend sicheren kryptographischen Zufallszahlengenerator erzeugt.	✓	✓

Referenzen

Der OWASP Mobile Security Testing Guide bietet detaillierte Anleitungen um die Anforderungen aus dieser Kategorie zu überprüfen.

- Für Android - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05e-Testing-Cryptography.md
- Für iOS - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06e-Testing-Cryptography.md

Weitere Informationen unter:

- OWASP Mobile Top 10: M5 - Insufficient Cryptography: https://www.owasp.org/index.php/Mobile_Top_10_2016-M5-Insufficient_Cryptography
- CWE: https://cwe.mitre.org/data/definitions/310.html

V4: Anforderungen an Authentifizierung und Session Management

Zielsetzung

Ein integraler Teil der Architektur einer mobilen App ist der Login eines Nutzers an einem entfernten Service. Obwohl sich ein Großteil der Logik an dem Endpunkt abspielt, definiert der MASVS eine Reihe von Basisanforderungen an Nutzerkonten und Session-Management.

Anforderungen

#	Beschreibung	L1	L2
4.1	Falls die App Nutzern Zugriff auf entfernte Service APIs bietet wird am API-Endpunkt eine Authentifizierung z.B. mit Nutzername/Passwort durchgeführt.		
4.2	Kommt Session-Management am API-Endpunkt zum Einsatz, so werden zufällig generierte Session-IDs erzeugt um Client-Anfragen zu authentifizieren und keine Nutzer-Anmeldedaten versandt.	✓	✓
4.3	Kommt statuslose Token-basierte Authentifizierung zum Einsatz, so werden die Token am Server mit einem sicheren Algorithmus signiert.	✓	✓
4.4	Der API-Endpunkt beendet die existierende Nutzersitzung sobald sich der Nutzer abmeldet.	✓	✓
4.5	Es existiert eine Passwort-Richtlinie die am entfernten API-Endpunkt erzwungen wird.	✓	✓
4.6	Der entfernte API-Endpunkt implementiert einen Mechanismus um sich gegen eine exzessive Anzahl von Login-Versuchen zu schützen.	✓	✓
4.7	Nach einer definierten Inaktivitätsdauer werden Nutzersitzungen am entfernten API-Endpunkt beendet und Zugriffs-Tokens werden nach Ablauf der Gültigkeitsdauer abgelehnt.	✓	✓
4.8	Biometrische Authentifizierung basiert auf dem Betriebssystem-basierten Entsperren des Keystores (Android)/der Keychain (iOS) und nicht auf einer z.B. event-basierten API die einfach "true" oder "false" zurück liefert.		✓
4.9	Es gibt einen 2. Authentifizierungsfaktor und er wird am entfernten API-Endpunkt konsistent erzwungen.		✓
4.10	Sensible Transaktionen erfordern eine zusätzliche Authentifizierung (durch einen weiteren Authentifizierungsfaktor).		✓
4.11	Die App informiert den Nutzer über alle Anmelde-Vorgänge am Nutzerkonto. Nutzer können eine Liste aller mit dem Konto verbundenen Geräte sehen und ausgewählte Geräte blockieren.		✓

Referenzen

Der OWASP Mobile Security Testing Guide bietet detaillierte Anleitungen um die Anforderungen aus dieser Kategorie zu überprüfen.

- Am General - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x04e-Testing-Authentication-and-Session-Management.md
- Für Android - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05f-Testing-Local-Authentication.md
- Für iOS - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06f-Testing-Local-Authentication.md

Weitere Informationen unter:

- OWASP Mobile Top 10: M4 - Insecure Authentication: https://www.owasp.org/index.php/Mobile_Top_10_2016-M4-Insecure_Authentication
- OWASP Mobile Top 10: M6 - Insecure Authorization: https://www.owasp.org/index.php/Mobile_Top_10_2016-M6-Insecure_Authorization
- CWE: https://cwe.mitre.org/data/definitions/287.html

V5: Anforderungen an Netzwerkkommunikation

Zielsetzung

Der Zweck dieser Kategorie ist es die Vertraulichkeit und Integrität übertragener Daten zwischen mobiler App und entferntem Server zu gewährleisten. Dazu muss eine mobile App einen sicheren, verschlüsselten Kanal zur Netzwerkkommunikation unter Nutzung des TLS-Protokolls mit adäquaten TLS-Einstellungen aufbauen. Level 2 benennt zusätzliche Defense-in-Depth Maßnahmen wie SSL-Pinning.

Anforderungen

#	Beschreibung	L1	L2
5.1	Daten werden zur Netzwerkkommunikation innerhalb der App durchgängig mit TLS verschlüsselt.	✓	✓
5.2	Die TLS-Einstellungen entsprechen aktuellen Best Practices, oder erfüllen die Best Practices so gut wie möglich falls das mobile Betriebssystem die Empfehlung nicht unterstützt.	✓	✓
5.3	Die App überprüft das X.509-Zertifikat des Servers beim TLS-Handshake. Die App akzeptiert nur Zertifikate die von einer vertrauenswürdigen Zertifizierungsstelle signiert wurden.	✓	✓
5.4	Die App nutzt entweder ihren eigenen Zertifikatspeicher oder nutzt Public Key-/ Zertifikats-Pinning und akzeptiert deshalb keine Verbindungen zu Endpunkten die andere Zertifikate/Schlüssel präsentieren - selbst wenn die Zertifikate von einer vertrauenswürdigen Zertifizierungstelle signiert sind.		✓
5.5	Die App nutzt keine unsicheren Kommunikationskanäle wie Email oder SMS für kritische Operationen wie Konten-Registrierung oder Konten-Reaktivierung.		✓
5.6	Die App nutzt aktuelle Bibliotheken zum Aufbau von sicheren Kommunikationsverbindungen.		✓

Referenzen

Der OWASP Mobile Security Testing Guide bietet detaillierte Anleitungen um die Anforderungen aus dieser Kategorie zu überprüfen.

- Für Android - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05g-Testing-Network-Communication.md
- Für iOS - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06g-Testing-Network-Communication.md

Weitere Informationen unter:

- OWASP Mobile Top 10: M3 - Insecure Communication: https://www.owasp.org/index.php/Mobile_Top_10_2016-M3-Insecure_Communication
- CWE: https://cwe.mitre.org/data/definitions/319.html
- CWE: https://cwe.mitre.org/data/definitions/295.html

V6: Anforderungen zur Plattform-Interaktion

Zielsetzung

Die Anforderungen aus dieser Kategorie sollen sicherstellen, dass Plattform-Komponenten und Standard-Komponenten von der App auf sichere Weise genutzt werden. Zusätzlich decken die Anforderungen auch die Kommunikation (IPC) zwischen Apps ab.

Anforderungen

#	Beschreibung	L1	L2
6.1	Die App fordert vom Nutzer nur die unbedingt erforderlichen App-Berechtigungen.	✓	✓
6.2	Alle Eingaben von externen Quellen und dem Nutzer werden validiert und falls erforderlich bereinigt. Dazu gehören Daten aus der GUI, IPC Mechanismen wie Intents oder spezifische URL-Schemas und Netzwerk-Daten.	✓	✓
6.3	Die App bietet keine sensible Funktionalität über App-eigene URL-Schemas an - wenn doch, ist der Mechanismus angemessen geschützt.	✓	✓
6.4	Die App bietet keine sensible Funktionalität über Interprozesskommunikation (IPC) an - wenn doch, ist der Mechanismus angemessen geschützt.	✓	✓
6.5	JavaScript ist in WebViews deaktiviert wenn es nicht unbedingt erforderlich ist.	✓	✓
6.6	WebViews sind so konfiguriert, dass nur die minimal nötigen Protokoll-Handler erlaubt sind (Idealerweise nur HTTPS). Potentiell gefährliche Handler wie file, tel und app-id sind deaktiviert.	✓	✓
6.7	Wenn eine WebView über Javascript Zugriff auf native Methoden einer App bekommt, ist sichergestellt, dass die WebView nur vorgegebenes Javascript aus der App rendert und kein Javascript aus externen Quellen.	✓	✓
6.8	Objektserialisierung wird, falls vorhanden, über eine sichere Serialisierungs-API implementiert.	✓	✓

Referenzen

Der OWASP Mobile Security Testing Guide bietet detaillierte Anleitungen um die Anforderungen aus dieser Kategorie zu überprüfen.

- Für Android - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05h-Testing-Platform-Interaction.md
- Für iOS - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06h-Testing-Platform-Interaction.md

Weitere Informationen unter:

- OWASP Mobile Top 10: M1 - Improper Platform Usage: https://www.owasp.org/index.php/Mobile_Top_10_2016-M1-Improper_Platform_Usage
- CWE: https://cwe.mitre.org/data/definitions/20.html
- CWE: https://cwe.mitre.org/data/definitions/749.html

V7: Anforderungen zu Code Qualität und Build-Einstellungen

Zielsetzung

Das Ziel dieser Kategorie ist, sicherzustellen, dass bei der App-Entwicklung Basis-Security-Praktiken eingehalten werden und die enthaltenen Sicherheits-Funktionen des Compilers aktiviert sind.

Anforderungen

#	Beschreibung	L1	L2
7.1	Die App ist signiert und mit einem gültigen Zertifikat provisioniert dessen privater Schlüssel angemessen geschützt ist.	✓	✓
7.2	Die App wurde im Release-Modus gebaut und mit passenden Release-Einstellungen (kein Debugging).	✓	✓
7.3	Debugging Symbole wurden von nativen Binärdateien entfernt.	✓	✓
7.4	Debugging Code wurde entfernt und die App-Logdateien enthalten keine ausführlichen Fehler oder Debug-Meldungen.	✓	✓
7.5	Bibliotheken und Frameworks von Drittanbietern die die App nutzt wurden auf Schwachstellen geprüft.	✓	✓
7.6	Die App führt eine sichere Fehlerbehandlung durch indem sie Exceptions abfängt und kontrolliert behandelt.	✓	✓
7.7	Treten in Sicherheitsfunktionen Fehler auf lehnt die App-Fehlerbehandlung die Zugriffe standardmäßig ab.	✓	✓
7.8	Das Speichermanagement (Allokation und Freigabe von Speicher) erfolgt in unmanaged code auf sichere Weise.	✓	✓
7.9	Angebotene Sicherheitsfunktionen der Entwicklungsumgebung wie Byte-Code Minimierung, Stack-Protection, PIE-Support und automatisches Reference-Counting sind aktiviert.	✓	✓

Referenzen

Der OWASP Mobile Security Testing Guide bietet detaillierte Anleitungen um die Anforderungen aus dieser Kategorie zu überprüfen.

- Für Android - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05i-Testing-Code-Quality-and-Build-Settings.md
- Für iOS - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06i-Testing-Code-Quality-and-Build-Settings.md

Weitere Informationen unter:

- OWASP Mobile Top 10: M7 - Poor Code Quality: https://www.owasp.org/index.php/Mobile_Top_10_2016-M7-Poor_Code_Quality
- CWE: https://cwe.mitre.org/data/definitions/119.html
- CWE: https://cwe.mitre.org/data/definitions/89.html
- CWE: https://cwe.mitre.org/data/definitions/388.html
- CWE: https://cwe.mitre.org/data/definitions/489.html

V8: Anforderungen an Manipulationssicherheit/Resilienz

Zielsetzung

Diese Kategorie behandelt Defense-in-Depth-Maßnahmen empfohlen für Apps die Zugriff auf sensible Daten oder sensible Funktionalitäten beinhalten. Sind diese Maßnahmen nicht umgesetzt, führt dies nicht unmittelbar zu einer Schwachstelle - jedoch erhöhen die Maßnahmen die Robustheit der App gegen Client-seitige Angriffe und Reverse Engineering.

Die Schutzmechanismen in diesem Abschnitt sollten nach Bedarf angewendet werden. Anhand einer Risikoprüfung sollten Risiken wie unerlaubte Manipulation der App oder Reverse Engineering des Codes bewertet werden. Es wird empfohlen das OWASP Dokument "Technical Risks of Reverse Engineering and Unauthorized Code Modification Reverse Engineering and Code Modification Prevention" (siehe Referenzen) zu nutzen um eine Liste mit Geschäftsrisiken und zugeordneten technischen Bedrohungen zu identifizieren.

Um die hier aufgeführten Schutzmechanismen effektiv umsetzen zu können muss eine App mindestens alle Anforderungen aus MASVS-L1 sowie die jeweiligen Anforderungen aus Kategorie V8 in aufsteigender Reihenfolge umsetzen. So setzen die Schutzmaßnahmen unter "Nachvollziehbarkeit verhindern" voraus, dass auch die Anforderungen aus "Dynamische Analyse und Manipulation verhindern" und "Gerätebindung" umgesetzt werden.

Achtung: Diese Software-Schutzmaßnahmen sind kein Ersatz für die Umsetzung adäquater Sicherheitsmaßnahmen. Die Maßnahmen aus MASVR-R sind gedacht um auf Basis App-spezifischer konkreter Bedrohungen zusätzliche Schutzmaßnahmen über die bereits umgesetzten Sicherheitsmaßnahmen aus MASVS hinaus umzusetzen.

Folgende Eckpunkte gelten:

1. Eine Bedrohungsanalyse wurde durchgeführt. Darin werden wesentliche Client-seitige Bedrohungen und Gegenmaßnahmen sowie der erforderliche Schutzbedarf dargestellt. Ein Aspekt zum Schutzbedarf könnte beispielsweise sein den Aufwand zum manuellen Reverse Engineering einer App für Malware-Autoren, die die App instrumentieren wollen, signifikant zu erhöhen.
2. Die Bedrohungsanalyse muss realistisch und vernünftig erfolgen. Zum Beispiel nützt es nichts, einen kryptographischen Schlüssel in einer White-Box-Implementierung zu "verbergen" wenn doch der Angreifer ohne weiteres an den Code kommt.
3. Die Effektivität der Schutzmaßnahmen sollte immer von einem Experten mit ausgewiesener Erfahrung im Bereich Anti-Code-Manipulation und Code-Obfuskierung überprüft werden (siehe auch Kapitel "reverse engineering" and "assessing software protections" im Mobile Security Testing Guide).

Dynamische Analyse und Manipulation verhindern

#	Beschreibung	R
8.1	Die App erkennt und reagiert auf ein gerootetes Gerät oder Geräte mit jailbreak indem der Nutzer gewarnt oder die App beendet wird.	✓
8.2	Die App verhindert Debugging und/oder erkennt und reagiert auf einen Debugger. Alle verfügbaren Debugging-Protokolle müssen abgedeckt werden.	✓
8.3	Die App erkennt und reagiert auf Manipulationen an ausführbaren Dateien und kritischen Daten innerhalb der App-eigenen Sandbox.	✓
8.4	Die App erkennt und reagiert auf typische und allgemein bekannte Reverse Engineering Tools und Frameworks auf dem Gerät.	✓
8.5	Die App erkennt und reagiert darauf wenn sie in einem Emulator ausgeführt wird.	✓
8.6	Die App erkennt und reagiert auf Manipulation von Code und Daten im eigenen Arbeitsspeicherbereich.	✓
8.7	Die App realisiert mehrere Mechanismen in jeder Kategorie (8.1 bis 8.6). Die Resilienz steigt mit der Anzahl, Vielfalt und Originalität der genutzten Mechanismen.	✓
8.8	Die Erkennungsmechanismen der App lösen verschiedene Arten von Reaktionen aus z.B. verzögerte, versteckte oder heimliche Reaktionen.	✓
8.9	Programmatische Abwehrmaßnahmen werden durch Obfuskierung geschützt was wiederum De-Obfuskierung mittels dynamischer Analyse verhindert.	✓

Gerätebindung

#	Beschreibung	R
8.10	Die App implementiert einen Mechanismus zur Gerätebindung bei dem der Geräte-Fingerabdruck aus mehreren einzigartigen Geräteeigenschaften ermittelt wird.	✓

Nachvollziehbarkeit verhindern

#	Beschreibung	R
8.11	Alle ausführbaren Dateien und Bibliotheken der App sind entweder auf Dateiebene verschlüsselt und/oder wichtige Code- und Datenabsegmente in ausführbaren Dateien sind verschlüsselt oder durch Packing obfuskiert. Triviale statische Analyse offenbart keinen wichtigen Code oder Daten.	✓
8.12	Wenn das Ziel der Obfuskierung der Schutz sensibler Logik wie Algorithmen oder Berechnungen ist, so wird ein angemessener Obfuskierungsmechanismus, dem Stand der Technik entsprechend, genutzt der resilient gegen manuelle und automatisierte De-Obfuskierungsangriffe ist. Die Wirksamkeit der Obfuskierungsmethode muss durch manuelle Tests überprüft werden. Es ist zu beachten, dass hardware-basierte Isolations-Mechanismen softwarebasierter Obfuskierung vorzuziehen sind.	✓

Referenzen

Der OWASP Mobile Security Testing Guide bietet detaillierte Anleitungen um die Anforderungen aus dieser Kategorie zu überprüfen.

- Für Android - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05j-Testing-Resiliency-Against-Reverse-Engineering.md
- Für iOS - https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06j-Testing-Resiliency-Against-Reverse-Engineering.md

Weitere Informationen unter:

- OWASP Mobile Top 10: M8 - Code Tampering: https://www.owasp.org/index.php/Mobile_Top_10_2016-M8-Code_Tampering
- OWASP Mobile Top 10: M9 - Reverse Engineering: https://www.owasp.org/index.php/Mobile_Top_10_2016-M9-Reverse_Engineering
- WASP Reverse Engineering Threats - https://www.owasp.org/index.php/Technical_Risks_of_Reverse_Engineering_and_Unauthorized_Code_Modification
- OWASP Reverse Engineering and Code Modification Prevention - https://www.owasp.org/index.php/OWASP_Reverse_Engineering_and_Code_Modification_Prevention_Project

Appendix A: Glossar

2FA

Zwei-Faktor-Authentifizierung bedeutet es wird ein zusätzlicher Authentifizierungsfaktor z.B. eine PIN oder ein Einmalpasswort zur Anmeldung am Nutzerkonto verlangt.

Address Space Layout Randomization (ASLR)

Eine Technik um Angriffe auf Arbeitsspeicherbereiche zu erschweren.

Akzeptanztest (UAT)

Eine Testumgebung die sich ähnlich verhält wie die Produktivumgebung, in der Tests vor dem go-live ausgeführt werden.

Applikationssicherheit

Applikationssicherheit ist fokussiert auf Sicherheitsaspekte und Angriffe auf Anwendungsebene d.h. Applikationskomponenten und -funktionen korrespondierend zur Anwendungsschicht im Open Systems Interconnection Reference Model (OSI Modell). Der Fokus liegt nicht auf Betriebssystem- oder Netzwerkaspekten.

Authentifizierung

Die Überprüfung der angegebenen Identität eines Nutzers.

Automatisierte Prüfungen

Die Nutzung automatisierter Werkzeuge (dynamische/statische Analyse oder beides) die Schwachstellen anhand von Signaturen identifizieren.

Bedrohungsanalyse

Eine Methodik die dazu dient Sicherheitsschwachstellen im Design einer Anwendung zu identifizieren und Gegenmaßnahmen zu entwickeln um die Sicherheitsarchitektur zu verbessern. Dabei werden relevante Gruppen von Angreifern, Sicherheitszonen, Sicherheitsmechanismen sowie technische und fachliche Wertgegenstände identifiziert und einbezogen.

Black box Tests

Ist eine Testmethode bei der die Funktionalität einer Komponente oder Anwendung "von außen" ohne Wissen über interne Strukturen und Mechanismen getestet wird.

Cross-Site Scripting (XSS)

Eine Sicherheitsschwachstelle die typischerweise in Web-Applikationen vorkommt die das Einschleusen von Client-seitigem Script-Code in den Seiteninhalt zulassen.

CWE

Common Weaknesses Enumeration - CWE ist eine Community-basierte Sammlung von allgemeinen Software Security Schwächen. Die Sammlung stellt eine Basis darf zur Identifikation, Mitigierung und zur Vermeidung von Schwachstellen und definiert so eine gemeinsame Sprache für alle Nutzer sowie für Bereitsteller von Security-Werkzeugen.

DAST

Dynamische Applikations-Security Tests (DAST) dienen der Erkennung von Sicherheitsschwachstellen einer Applikation zur Laufzeit.

Dynamische Prüfungen

Die Nutzung automatisierter Werkzeuge um zur Laufzeit einer Applikation Sicherheitsschwachstellen auf Basis von Signaturprüfungen zu finden.

Eingabe-Validierung

Überführung in eine standardisierte Form (Kanonisierung) und Prüfung von Eingabedaten denen nicht vertraut wird z.B. Nutzereingaben oder Request-Parameter.

Globally Unique Identifier(GUID)

Eine einzigartige Referenz-Nummer die als Identifikator in einer Software genutzt werden kann.

Hartcodierte Schlüssel

Kryptographische Schlüssel die auf unsichere Weise direkt im Quellcode oder der Anwendungskonfiguration hinterlegt sind.

Hyper Text Transfer Protocol (HTTP)

Ein Kommunikations-Protokoll für verteilte Informationssysteme auf Basis von Hypermedia und damit die Basis der Datenkommunikation im weltweiten Internet.

IPC

Interprozesskommunikation - Mit IPC kommunizieren Prozesse über Betriebssystem-Mechanismen mit dem Kernel und untereinander um Aktivitäten zu koordinieren oder Daten auszutauschen.

Java Bytecode

Java Bytecode ist der Befehlssatz der Java Virtual Machine (JVM). Ein Bytecode besteht aus einem oder in einigen Fällen zwei Bytes die einen Befehl (OP-Code) repräsentieren sowie optional weitere Bytes die als Parameter für den OP-Code dienen.

Komponente

Eine Zusammenfassung einzelner Code-Elemente zu einer eigenständigen Einheit mit Zugriffen auf Speicher- und Netzwerkschnittstellen um mit anderen Komponenten zu kommunizieren.

Kryptographisches Modul

Hardware, Software, und/oder Firmware die kryptographische Algorithmen und/oder erzeugte kryptographische Schlüssel nutzt.

Malicious Code

Bösartiger Code der während der Entwicklung, verborgen vor dem Applikationsverantwortlichen, in die Applikation eingebracht wird. Der eingeschleuste Code umgeht dabei gezielt Sicherheitsrichtlinien und ist dadurch nicht vergleichbar mit Malware wie einem Virus oder einem Wurm!

Malware

Ausführbarer Code der zur Laufzeit ohne Wissen des Nutzers oder Administrators in die Zielanwendung injiziert wird.

Open Web Application Security Project (OWASP)

Open Web Application Security Project (OWASP) ist eine weltweite freie, offene und herstellerunabhängige Community mit Fokus auf Verbesserung der Applikationssicherheit. Unsere Mission ist es Applikationssicherheit sichtbar zu machen so dass Einzelpersonen und Organisationen klare und bewußte Entscheidungen über Sicherheitsrisiken treffen können. Mehr unter: https://www.owasp.org/

Personenbezogene Daten

Personenbezogene Daten sind Daten die genutzt werden können um eine Person direkt oder indirekt zu identifizieren, kontaktieren oder lokalisieren bzw. eine Person in einem Zusammenhang zu identifizieren.

PIE

Position-independent executable (PIE) - Positionsunabhängiger Code ist Maschinencode der an einer beliebigen Stelle im primären Speicher ausgeführt werden kann. (unabhängig von der absoluten Speicheradresse)

PKI

Public Key Infrastruktur - PKI basiert darauf, dass öffentliche Schlüssel an eine Identität gebunden werden. Die Bindung erfolgt durch einen Registrierungsprozess und das Ausstellen eines Zertifikats durch eine Zertifizierungsstelle, in englisch Certificate Authority (CA).

Prüfer

Eine Person oder ein Team, dass eine Anwendung gegen den OWASP MASVS prüft.

Prüfung zur Applikationssicherheit

Die technische Prüfung einer Applikation gegen den OWASP MASVS.

Prüfbericht zur Applikationssicherheit

Ein Prüfbericht, für eine Applikation, der die Analyseschritte eines Prüfers sowie die Gesamtergebnisse dokumentiert.

SAST

Statische Applikations-Security-Tests (SAST) sind eine Reihe von Techniken, die dazu genutzt werden können potentielle Sicherheitsschwachstellen in Quellcode, Bytecode und Binärdateien zu identifizieren. SAST Lösungen analysieren eine Applikation typischerweise zur Entwicklungs- oder Buildzeit jedoch nicht zur Laufzeit.

SDLC

Software development lifecycle.

Sicherheitsarchitektur

Eine Abstraktion des Applikations-Designs einer Anwendung bei der dokumentiert wird an welchen Stellen und in welchem Maße Sicherheitsmechanismen genutzt werden. Darüber hinaus wird beschrieben an welchen Stellen im System sensible Nutzer- und Anwendungsdaten verarbeitet werden.

Sicherheitsarchitekturanalyse

Die technische Prüfung der Sicherheitsarchitektur einer Applikation.

Sicherheitskonfiguration

Die Laufzeitkonfiguration einer Anwendung; enthält Optionen, die die Sicherheitsfunktionen beeinflussen.

Sicherheitsmechanismus

Eine Sicherheitsfunktion oder Komponente die Sicherheitsprüfungen durchführt z.B. eine Autorisierungsprüfung oder das Erzeugen eines Eintrags im Audit-Log beim Login eines Administrators.

SQL Injection (SQLi)

Eine Technik um Code in datengetriebene Anwendungen einzuschleusen. Dabei werden schadhafte SQL-Anweisungen in Nutzereingaben eingeschleust und in der Datenbank zur Ausführung gebracht.

SSO Authentifizierung

Single Sign On(SSO) bedeutet, ein Nutzer muss sich an einer Applikation einloggen und ist dann automatisch an weiteren Anwendungen angemeldet. Damit können sich Nutzer u.U. über mehrere unterschiedliche Plattformen, Technologien und Domains anmelden. Zum Beispiel ist man bei Google nach der Nutzeranmeldung automatisch auch für Youtube, Google-Docs und Google-Mail angemeldet.

Transport Layer Security (TLS)

Kryptographisches Protokoll um die Vertraulichkeit, Integrität und Authentizität von Daten während der Übertragung im Internet abzusichern.

URI/URL/URL Fragmente

Eine URI (Uniform Resource Identifier) ist eine Zeichenfolge um einen Namen oder eine Webressource zu identifizieren. Ein URL (Uniform Resource Locator) wird oft als Referenz auf eine Webressource genutzt.

Whitelist

Eine Liste erlaubter Operationen zum Beispiel eine Liste erlaubter Buchstaben die zur Eingabe-Validierung genutzt werden soll.

X.509 Zertifikat

Ein X.509 Zertifikat ist ein digitales Zertifikat das eine international standardisierten PKI-Standard nutzt um nachzuweisen dass ein öffentlicher Schlüssel zu einem Nutzer, einem Computer oder einer Serviceidentität, aufgeführt in dem Zertifikat, gehört.

Appendix B: Referenzen

Die folgenden OWASP Projekte könnten für Anwender dieses Standards nützlich sein:

- OWASP Mobile Security Project - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
- OWASP Mobile Security Testing Guide - https://www.owasp.org/index.php/OWASP_Mobile_Security_Testing_Guide
- OWASP Mobile Top 10 Risks - https://www.owasp.org/index.php/Projects/OWASP*Mobile_Security_Project*-_Top_Ten_Mobile_Risks
- OWASP Reverse Engineering and Code Modification Prevention - https://www.owasp.org/index.php/OWASP_Reverse_Engineering_and_Code_Modification_Prevention_Project

Die folgenden Webseiten könnten ebenfalls für Anwender dieses Standards nützlich sein:

- MITRE Common Weakness Enumeration - http://cwe.mitre.org/
- PCI Security Standards Council - https://www.pcisecuritystandards.org
- PCI Data Security Standard (DSS) v3.0 Requirements and Security Assessment Procedures https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf