Application Security Verification Standard 4.0

Finale Version

März 2019

Frontispiece

Der ASVS Standard

Der Application Security Verification Standard ist eine Liste von Anwendungssicherheits-Anforderungen und -Tests, die von Architekten, Entwicklern, Testern, Sicherheitsexperten, Werkzeugherstellern und Verbrauchern verwendet werden können, um sichere Anwendungen zu konzipieren, zu implementieren, zu testen und zu verifizieren.

Copyright und Lizenz

Version 4.0.1, März 2019, deutsche Übersetzung, März 2020

Projektleiter

- Andrew van der Stock
- Daniel Cuthbert
- Jim Manico
- Josh C Grossman
- Mark Burnett

Beitragende und Rezensenten

- Osama Elnaggar
- Erlend Oftedal
- Serg Belkommen
- David Johansson
- Tonimir Kisasondi
- Ron Perris
- Jason Axley

- Abhay Bhargav
- Benedikt Bauer
- Elar Lang
- ScriptingXSS
- Philippe De Ryck
- Grog's Axle
- Marco Schnüriger
- Jacob Salassi
- Glenn ten Cate
- Anthony Weems
- bschach
- javixeneize
- Dan Cornell
- hello7s
- Lewis Ardern
- Jim Newman
- Stuart Gunter
- Geoff Baskwill
- Talargoni
- Ståle Pettersen
- Kelby Ludwig
- Jason Morrow
- Rogan Dawes
- Daniël Geerts

Der Application Security Verification Standard baut auf den Vorarbeiten der Beteiligten von ASVS 1.0 im Jahr 2008 bis 3.0 im Jahr 2016 auf. Viele der Struktur- und Verifizierungselemente, die heute noch in der ASVS enthalten sind, wurden ursprünglich von Mike Boberski, Jeff Williams und Dave Wichers geschrieben, aber es gibt darüber hinaus noch viele weitere Mitwirkende. Vielen Dank an alle, die sich bisher beteiligt haben. Für eine umfassende Liste all jener, die zu früheren Versionen beigetragen haben, konsultieren Sie bitte die jeweiligen Vorversionen.

Wenn eine Danksagung in der obigen 4.0-Gutschriftenliste fehlt, wenden Sie sich bitte an vanderaj@owasp.org oder stellen Sie ein Ticket bei GitHub ein, um in zukünftigen 4.x-Updates anerkannt zu werden.

Vorwort

Willkommen beim Application Security Verification Standard (ASVS) Version 4.0. Der ASVS ist ein von der Community getriebenes Vorhaben mit dem Ziel, ein Rahmenwerk von Sicherheitsanforderungen und -kontrollen zu schaffen. Dabei soll sich dieses auf die Definition der funktionalen und nicht-funktionalen Sicherheitskontrollen konzentrieren, die beim Entwurf, der Entwicklung und dem Testen moderner Webanwendungen und Webservices erforderlich sind.

ASVS v4.0 ist das Ergebnis seitens der Bemühungen der Community und des Feedbacks aus der Industrie des letzten Jahrzehnts. Wir haben versucht, die Einführung des ASVS zu erleichtern um dies für eine Vielzahl von

verschiedenen Anwendungsfällen während des gesamten Lebenszyklus der sicheren Softwareentwicklung zu zu ermöglichen.

Wir wissen, dass es höchstwahrscheinlich niemals einen 100%igen Konsens über den Inhalt eines Webanwendungsstandards, einschließlich des ASVS, geben wird. Die Risikoanalyse ist immer bis zu einem gewissen Grad subjektiv, was eine Herausforderung darstellt, wenn man versucht, alles in einem Standard zu verallgemeinern. Wir hoffen jedoch, dass die letzten Aktualisierungen in dieser Version ein Schritt in die richtige Richtung sind und die in diesem wichtigen Industriestandard eingeführten Konzepte weiter verbessern.

Was ist neu in 4.0

Die größte Änderung in dieser Version ist die Aufnahme der NIST 800-63-3 Richtlinien zur digitalen Identität, die moderne, stichhaltige und erweiterte Authentifizierungskontrollen einführt. Obwohl wir einen gewissen Rückschritt bei der Angleichung an einen fortgeschrittenen Authentifizierungsstandard feststellen, sind wir der Meinung, dass eine Angleichung der Standards unerlässlich ist, vor allem dann, wenn ein anderer angesehener evidenzbasierter Anwendungssicherheitsstandard bereits vorhanden ist.

Standards der Informationssicherheit sollten stets versuchen, die Anzahl der besonderen Anforderungen zu minimieren, so dass sich Unternehmen, die die Anforderungen erfüllen möchten, nicht über konkurrierende oder inkompatible Mechanismen entscheiden müssen. Sowohl die OWASP Top 10 2017 als auch nun auch der OWASP Application Security Verification Standard haben sich hinsichtlich Authentifizierung und Sitzungsmanagement an NIST 800-63 angelehnt. Wir ermutigen andere standardsetzende Gremien, mit uns, dem NIST und anderen zusammenzuarbeiten, um zu einem allgemein akzeptierten Satz von Anwendungssicherheitsmechanismen zu kommen, um die Sicherheit zu maximieren und gleichzeitig die Kosten für die Einhaltung von Vorschriften zu minimieren.

Alle Kapitel des ASVS 4.0 wurden von Anfang bis Ende vollständig neu durchnummeriert. Das neue Nummerierungsschema ermöglichte es uns, Lücken aus längst überholten Kapiteln zu schließen und längere Kapitel aufzuteilen, um die Anzahl der Kontrollen, die ein Entwickler oder ein Team einhalten muss, zu minimieren. Wenn eine Anwendung beispielsweise keine JSON Web tokens (JWT) verwendet, ist der gesamte Abschnitt über JWT im Sitzungsmanagement nicht relevant.

Neu in Version 4.0 ist eine umfassende Abbildung der Common Weakness Enumeration (CWE), eine der am häufigsten gewünschten Erweiterungen, die wir in den letzten zehn Jahren hatten. Die CWE-Zuordnung ermöglicht es Werkzeugherstellern und Anwendern von Software für Schwachstellenmanagement, die Ergebnisse anderer Werkzeuge und früherer ASVS-Versionen mit der Version 4.0 bzw. neueren Versionen abzugleichen. Um Platz für den CWE-Eintrag zu schaffen, mussten wir die Spalte "Seit" entfernen Da wird alle Kapitel neu durchnummeriert haben ist diese Angabe weniger sinnvoll als in früheren Versionen des ASVS. Nicht jeder Eintrag in der ASVS hat eine zugehörige CWE, und da die CWE sehr viele Dubletten aufweist, haben wir versucht, den am häufigsten verwendeten und nicht unbedingt den nächstliegenden Eintrag zu verwenden. Verifizierungskontrollen lassen sich nicht immer eindeutig auf entsprechende Schwachstellen abbilden. Wir begrüßen jedoch die laufende Diskussion über die Schließung dieser Lücke mit der CWE-Community und der Informationssicherheit im Allgemeinen.

Wir haben daran gearbeitet, die Anforderungen der OWASP Top 10 2017 und der OWASP Proactive Controls 2018 weitestgehend zu erfüllen oder gar zu übertreffen. Da die OWASP Top 10 2017 das absolute Minimum zur Vermeidung von Nachlässigkeiten darstellt, haben wir bewusst alle Top 10 Anforderungen, bis auf spezifische Anforderungen an die Protokollierung, als Kontrollen der Stufe 1 eingeführt. Damit wollen wir es den OWASP Top 10-Anwendern erleichtern, einen tatsächlichen Sicherheitsstandard zu erreichen.

Wir wollten sicherstellen, dass die Stufe 1 des ASVS 4.0 eine umfassende Obermenge der Abschnitte 6.5 des PCI DSS 3.2.1 für Anwendungsdesign, Kodierung, Tests, Codereviews für Sicherheit sowie Penetrationstests darstellt. Dies erforderte zusätzlich zu den bestehenden branchenführenden Anforderungen an die Anwendungs- und Web-Service-Verifikation auch die Abdeckung von Bufferoverflows und unsicheren Speicheroperationen in V5 und unsichere speicherbezogene Compilerflags in V14.

Wir haben damit auch die Umstellung des ASVS weg von den monolithischen, lediglich serverseitigen Kontrollen hin zur Bereitstellung von Sicherheitskontrollen für alle modernen Anwendungen und APIs abgeschlossen. In Zeiten funktionaler Programmierung, von Serverless APIs, mobiler Anwendungen, der Cloud,

Containern, von CI/CD, DevSecOps, Federation und weiteren können wir die moderne Anwendungsarchitektur nicht länger ignorieren. Moderne Anwendungen sind ganz anders konzipiert als diejenigen, die bei der Veröffentlichung des ursprünglichen ASVS im Jahr 2009 umgesetzt wurden. Die ASVS muss immer weit in die Zukunft blicken, damit wir unserem primären Publikum - den Entwicklern - fundierte Ratschläge geben können. Wir haben alle Anforderungen, die davon ausgehen, dass Anwendungen auf Systemen einer einzigen Organisation ausgeführt werden, genau beschrieben oder entfernt.

Aufgrund der Größe des ASVS 4.0 und unseres Wunsches, die Basis für alle weiteren ASVS-Bemühungen zu werden, haben wir die mobile Sektion zugunsten des Mobile Application Security Verification Standard (MASVS) aufgegeben. Der Anhang zum Internet der Dinge (IoT) wird ebenfalls in einer zukünftigen IoT-ASVS Version erscheinen, welches vom OWASP IoT Projekts betreut wird. Eine erste Vorschau auf den IoT-ASVS haben wir in Anhang C aufgenommen. Wir danken sowohl dem OWASP Mobile Team als auch dem OWASP IoT-Projektteam für ihre Unterstützung des ASVS und freuen uns darauf, in Zukunft mit ihnen zusammenzuarbeiten, um ergänzende Standards zu schaffen.

Und schließlich haben wir weniger wirksame Kontrollen enttarnt und entfernt. Im Laufe der Zeit wurde die ASVS zu einem umfassenden Satz von Kontrollen, aber nicht alle Kontrollen sind gleichwertig hinsichtlich der Herstellung sicherer Software. Diese Bemühungen zur Beseitigung von Elementen mit geringer Auswirkung könnten noch weiter gehen. In einer künftigen Ausgabe des ASVS wird das Common Weakness Scoring System (CWSS) dazu beitragen, die wirklich wichtigen Kontrollen und die Kontrollen, die ausgemustert werden sollten, weiter zu priorisieren.

Ab der Version 4.0 wird sich der ASVS ausschließlich darauf konzentrieren, der führende Webanwendungs- und Servicestandard zu sein, der traditionelle und moderne Anwendungsarchitekturen sowie agile Sicherheitspraktiken und die DevSecOps-Kultur abdeckt.

Verwendung des ASVS

Der ASVS verfolgt zwei Hauptziele:

- Organisationen bei der Entwicklung und Wartung sicherer Anwendungen zu unterstützen.
- Sicherheitsdienstleistern, Anbietern von Sicherheitswerkzeugen und Verbrauchern die Möglichkeit zu geben, deren Anforderungen und Angebote dazu aufeinander abzustimmen.

Stufen des Application Security Verification Standards

Der Application Security Verification Standard definiert drei Verifizierungsstufen für Sicherheit, wobei jede Stufe immer weiter in die Tiefe geht.

- Die Stufe 1 ASVS ist für niedrige Sicherheitsstufen gedacht und ist vollständig durch Penetrationstests verifizierbar.
- Die Stufe 2 des ASVS ist für alle Anwendungen gedacht, die schützenwerte sensible Daten enthalten, und ist die empfohlene Stufe für die meisten Anwendungen.
- Die Stufe 3 des ASVS ist für die Anwendungen mit der höchsten Kritikalität gedacht - also Anwendungen, die Transaktionen von hohem Wert durchführen, sensible medizinische Daten enthalten bzw. alle Anwendungen, die ein Höchstmaß an Vertrauen erfordern.

Jede ASVS-Stufe enthält eine Liste von Sicherheitsanforderungen. Jede dieser Anforderungen kann auch auf sicherheitsspezifischen Features und Eigenschaften abgebildet werden, die von Entwicklern in die Software eingebaut werden müssen.

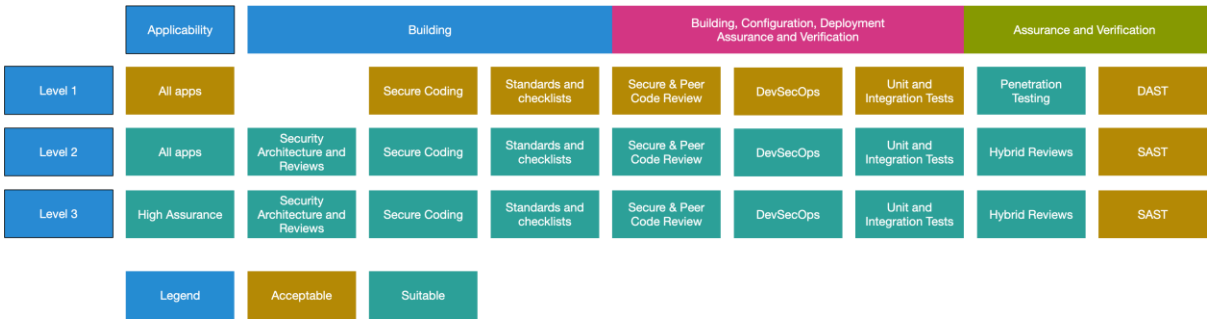

Abbildung 1 - Stufen des OWASP Application Security Verification Standard 4.0

Die Stufe 1 ist die einzige Stufe, die vollständig von Menschen mittels Penetrationtests prüfbar ist. Alle anderen Stufen erfordern zusätzlich Zugang zu Dokumentation, Quellcode, Konfiguration und den am Entwicklungsprozess beteiligten Personen. Aber selbst wenn die Stufe 1 "Black Box"-Tests (ohne Dokumentation und Quellcode) zulässt, ist dies keine wirksame Sicherungsmaßnahme und es sollte proaktiv von deren alleinigen Verwendung abgeraten werden. Böswillige Angreifer haben sehr viel Zeit zur Verfügung, während für die meisten Penetrationstests nur ein paar Wochen zur Verfügung stehen. Die Verteidiger müssen Sicherheitskontrollen einbauen, alle Schwachstellen schützen, finden und beheben und böswillige Akteure innerhalb einer angemessenen Zeit aufspüren und darauf reagieren. Böswillige Akteure haben im Wesentlichen unendlich viel Zeit und benötigen nur eine einzige undichte Verteidigung, eine einzige Schwachstelle oder eine fehlende Erkennung, um erfolgreich zu sein. Black-Box-Tests, die hektisch oft erst am Ende der Entwicklung oder womöglich gar nicht durchgeführt werden, können dieser Asymmetrie unmöglich mit Erfolg begegnen.

In den letzten 30 Jahren haben Black-Box-Tests immer wieder bewiesen, daß dabei kritische Sicherheitsprobleme übersehen werden, die dann zu immer massiveren Verletzungen der Datensicherheit führen. Wir befürworten nachdrücklich den Einsatz einer breiten Palette von Verfahren zur Sicherheitssicherung und Verifizierungsmaßnahmen. Dies umfasst die Ablösung klassischer Penetrationstests durch quellcodegeführte (hybride) Penetrationstests auf Stufe 1, mit vollem Zugang zu den Entwicklern und der Dokumentation während des gesamten Entwicklungsprozesses. Die Finanzaufsichtsbehörden tolerieren keine externen Finanzprüfungen ohne Zugang zu den Bilanzbüchern, Mustertransaktionen oder den Personen, welche die Kontrollen durchführen. Industrie und Regierungen müssen den gleichen Standard an Transparenz im Bereich der Softwareentwicklung fordern.

Wir empfehlen nachdrücklich die Verwendung von Sicherheitswerkzeugen innerhalb des Entwicklungsprozesses. DAST- und SAST-Werkzeuge können von der Build-Pipeline kontinuierlich verwendet werden, um leicht zu aufzuspürende Sicherheitsprobleme, die niemals vorhanden sein sollten, zu finden ("Low Hanging Fruits").

Automatisierte Werkzeuge und Online-Scans können nicht mehr als die Hälfte der ASVS ohne menschliche Hilfe abdecken. Wenn eine umfassende Testautomatisierung für jeden Build erforderlich ist, dann wird eine Kombination aus benutzerdefinierten Unit- und Integrationstests zusammen mit durch den Build initiierten Online-Scans verwendet. Tests der Geschäftslogik und der Zugriffskontrolle sind nur mit menschlicher Unterstützung möglich. Diese sollten in Unit- und Integrationstests umgesetzt werden.

Wie man diesen Standard verwendet

die beste Möglichkeit, den Application Security Verification Standard zu nutzen, ist diesen als Blaupause zur Erstellung einer Checkliste für die sichere Entwicklung zu verwenden, die speziell auf Ihre Anwendung, Plattform oder Organisation zugeschnitten ist. Durch die Anpassung des ASVS an Ihre Anwendungsfälle wird der Schwerpunkt auf die Sicherheitsanforderungen gelegt, die für Ihre Projekte und Umgebungen am wichtigsten sind.

Stufe 1 - Erste Schritte - automatisiert und Ausgangspunkt für weitere Stufen

Eine Anwendung erreicht ASVS Stufe 1, wenn sie sich angemessen gegen einfach zu entdeckende Sicherheitslücken der Anwendung verteidigt, die in den OWASP Top 10 und anderen ähnlichen Checklisten beschrieben sind.

Die Stufe 1 ist das absolute Minimum, das alle Anwendungen anstreben sollten. Es ist auch hilfreich als erster Schritt innerhalb eines mehrstufigen Prozesses oder wenn Anwendungen keine sensiblen Daten speichern bzw. verarbeiten und daher nicht die strengeren Kontrollen der Stufen 2 oder 3 benötigen. Kontrollen der Stufe 1 können entweder automatisch durch Werkzeuge oder ohne großen Aufwand manuell ohne Zugriff auf den Quellcode überprüft werden. Wir betrachten Stufe 1 als das für alle Anwendungen erforderliche Minimum.

Bedrohungen für die Anwendung werden höchstwahrscheinlich von Angreifern ausgehen, die einfache wenig aufwendige Techniken verwenden, um schnell zu findende und leicht ausnutzbare Schwachstellen zu identifizieren. Dies steht im Gegensatz zu einem entschlossenen Angreifer, der seine Energie gezielt für die Anwendung einsetzt. Wenn die von Ihrer Anwendung verarbeiteten Daten einen hohen Wert haben, wird man nur selten mit einer Überprüfung der Stufe 1 aufhören.

Stufe 2 - Der Standard für die meisten Anwendungen

Eine Anwendung erreicht ASVS Stufe 2 (d.h. den Standard), wenn sie die meisten Risiken, die heutzutage mit Software verbunden sind, angemessen abwehrt.

Stufe 2 stellt sicher, dass wirksame Sicherheitskontrollen vorhanden sind und innerhalb der Anwendung verwendet werden. Stufe 2 ist in der Regel für Anwendungen geeignet, die signifikante Business-zu-Business Transaktionen abwickeln. Einschließlich solcher, die Informationen aus dem Gesundheitswesen verarbeiten, geschäftskritische oder sensible Funktionen implementieren oder andere sensible Vermögenswerte verarbeiten, oder für Branchen, in denen die Integrität ein kritischer Aspekt zum Schutz ihres Geschäfts ist, wie z.B. die Spielindustrie, um Betrüger und Spiel-Hacks zu vereiteln.

Bedrohungen von Anwendungen der Stufe 2 sind in der Regel qualifizierte und motivierte Angreifer, die sich auf bestimmte Ziele konzentrieren und dabei Werkzeuge und Techniken einsetzen, die gut trainiert und wirksam sind, um Schwachstellen in Anwendungen zu entdecken und auszunutzen.

Stufe 3 - Anwendungen mit sehr hohen Sicherheitsanforderungen

ASVS Stufe 3 ist die höchste Verifizierungsstufe innerhalb des ASVS. Diese Stufe ist in der Regel Anwendungen vorbehalten, die ein erhebliches Maß an Sicherheitsverifikation erfordern, wie z.B. solche, die in Bereichen des Militärs, der Gesundheit und Safety, kritischen Infrastrukturen usw. zu finden sind.

Für Organisationen kann die ASVS-Stufe 3 nötig werden, wenn deren Anwendungen kritische Funktionen ausführen, bei denen ein Ausfall die Operationen der Organisation und sogar ihre Überlebensfähigkeit erheblich beeinträchtigen könnte. Nachstehend finden Sie ein Beispiel für die Anwendung von ASVS Stufe 3. Eine Anwendung erreicht ASVS-Stufe 3 (Fortgeschrittene Stufe), wenn sie sich angemessen gegen fortgeschrittene Sicherheitslücken einer Anwendung wehrt und auch die Prinzipien eines guten Sicherheitsdesigns verfolgt.

Eine Anwendung auf ASVS-Stufe 3 erfordert eine gründlichere Analyse der Architektur, der Codierung und des Testprozesses als alle anderen Stufen. Eine sichere Anwendung wird auf sinnvolle Weise modularisiert (um die Robustheit, Skalierbarkeit und vor allem den Aufbau von Sicherheitsschichten zu erleichtern), und jedes Modul (getrennt nach Netzwerkverbindung und/oder physischer Instanz) kümmert sich um seine eigenen Sicherheitsverantwortlichkeiten (*Defense in Depth*), die angemessen dokumentiert werden müssen. Zu den Verantwortlichkeiten gehören Kontrollen zur Gewährleistung der Vertraulichkeit (z.B. Verschlüsselung), Integrität (z.B. Transaktionen, Eingabevalidierung), Verfügbarkeit (z.B. entsprechende Lastverteilung), Authentifizierung (auch zwischen Systemen), Unabstreitbarkeit, Autorisierung und Auditierung (Protokollierung).

Anwendung des ASVS in der Praxis

Verschiedene Bedrohungen haben ihren Ursprung in unterschiedlichen Motivationen. Einige Branchen verfügen über einzigartige Informations- und Technologievorteile und bereichsspezifische regulatorische Anforderungen.

Organisationen wird empfohlen, ihre spezifischen Risikomerkmale auf der Grundlage ihrer Geschäftsart eingehend zu prüfen und auf Grundlage dieser Risiken und Geschäftsanforderungen die dafür geeignete ASVS-Ebene zu bestimmen.

Prüfung und Zertifizierung

Die Haltung der OWASP zu ASVS-Zertifizierungen und Vertrauenssiegeln

OWASP als herstellerneutrale, gemeinnützige Organisation zertifiziert derzeit keine Anbieter, Verifizierer oder Software für ASVS.

Alle derartigen Zusicherungen, Vertrauenssiegel oder Zertifizierungen werden von OWASP nicht offiziell überprüft, registriert oder zertifiziert, so dass eine Organisation, die sich auf eine solche Prüfung verlässt, vorsichtig sein muss, was das Vertrauen in eine dritte Partei oder ein Vertrauenssiegel betrifft, das eine ASVS-Zertifizierung für sich beansprucht.

Dies sollte aber Organisationen auch nicht daran hindern, derartige Dienstleistungen anzubieten, solange sie keine offizielle OWASP-Zertifizierung beanspruchen.

Leitfaden für zertifizierende Organisationen

Der Application Security Verification Standard kann für volltransparente Prüfungen von Anwendungen verwendet werden, einschließlich des offenen und ungehinderten Zugangs zu Schlüsselressourcen wie Architekten und Entwickler, Projektdokumentation, Quellcode, authentifizierter Zugang zu Testsystemen (inklusive des Zugangs zu Konten in jeder Rolle), insbesondere für Verifikationen der Stufen 2 und 3.

In der Vergangenheit haben Penetrationstests und Überprüfungen von sicherem Code ausnahmslos Probleme beinhaltet - das heißt, dass nur fehlgeschlagene Tests im Abschlussbericht erscheinen. Eine zertifizierende Organisation muss in jedem Bericht folgende Angaben mit aufnehmen:

- Den Umfang der Verifizierung (insbesondere, wenn eine Schlüsselkomponente außerhalb des Umfangs liegt, wie z.B. die SSO-Authentifizierung)
- Eine Zusammenfassung der Verifizierungsergebnisse, einschließlich der bestandenen und nicht bestandenen Tests, mit klaren Angaben zur Lösung der fehlgeschlagenen Tests.

Bestimmte Prüfungsanforderungen sind möglicherweise nicht auf die zu prüfende Anwendung anwendbar. Wenn Sie beispielsweise Ihren Kunden eine zustandslose Service-Schicht API ohne eine zugehörige Client-Implementierung zur Verfügung stellen, sind viele der Anforderungen in *V3 Session Management* nicht direkt anwendbar. In solchen Fällen kann eine zertifizierende Organisation immer noch die volle ASVS-Konformität beanspruchen, muss aber in jedem Bericht klar einen Grund für die Nichtanwendbarkeit solcher ausgeschlossener Prüfungsanforderungen angeben.

Die Aufbewahrung von detaillierten Arbeitsaufzeichnungen, Screenshots oder Videos, Skripts zur verlässlichen und wiederholten Ausnutzung eines Problems und elektronischen Aufzeichnungen von Tests, wie z.B. das Abfangen von Proxy-Protokollen und zugehörigen Notizen, wie z.B. einer Bereinigungsliste, gilt als Industriestandard und kann für die kritischsten Entwickler als Beweis für die Ergebnisse nützlich sein. Es reicht nicht aus, einfach ein Tool laufen zu lassen und über die Fehler zu berichten; dies liefert (überhaupt) keinen ausreichenden Beweis dafür, dass alle Probleme auf einer Zertifizierungsebene gründlich getestet und geprüft worden sind. Im Falle von Streitigkeiten sollte es genügend Sicherheitsbeweise geben, um nachzuweisen, dass jede einzelne verifizierte Anforderung tatsächlich auch getestet wurde.

Testmethoden

Zertifizierende Organisationen können die geeignete(n) Prüfmethode(n) frei wählen, sollten diese jedoch in einem Bericht aufführen.

Je nach der zu prüfenden Anwendung und den Prüfungsanforderungen können unterschiedliche Prüfmethoden verwendet werden, um ein hinreichendes Vertrauen in die Ergebnisse zu gewinnen. Die Überprüfung der Wirksamkeit von Eingabevalidierungsmechanismen einer Anwendung kann beispielsweise entweder mittels eines manuellen Penetrationstests oder mit Hilfe von Quellcodeanalysen durchgeführt werden.

Die Rolle automatisierter Sicherheitswerkzeuge

Der Einsatz von Werkzeugen für automatisierte Penetrationstests wird empfohlen, um eine möglichst hohe Testabdeckung zu erreichen.

Es ist nicht möglich, die vollständige ASVS-Prüfung nur mit automatisierten Penetrationstesttools alleine abzudecken. Während eine große Anzahl der Anforderungen in Stufe 1 mit automatisierten Tests überprüft werden kann, ist die überwiegende Mehrheit der Anforderungen nicht für automatisierte Penetrationstests geeignet.

Bitte beachten Sie, dass die Grenzen zwischen automatisierten und manuellen Tests mit zunehmender Reife in der Anwendungssicherheitsindustrie immer mehr verschwimmen. Automatisierte Werkzeuge werden häufig von Experten manuell optimiert, und manuelle Tester nutzen oft eine Vielzahl automatisierter Werkzeuge.

Die Rolle von Penetrationstests

In Version 4.0 haben wir uns entschieden, die Stufe 1 vollständig penetrationstestbar zu machen, ohne Zugriff auf Quellcode, Dokumentation oder Entwickler zu erfordern. Zwei Protokollierungselemente, die zur Einhaltung der OWASP Top 10 2017 A10 erforderlich sind, erfordern Interviews, Screenshots oder eine andere Sammlung von Beweisen, wie sie auch in der OWASP Top 10 2017 erforderlich sind. Das Testen ohne Zugang zu den notwendigen Informationen ist jedoch keine ideale Methode der Sicherheitsüberprüfung, da die Möglichkeit versäumt wird, ein Code-Review zu etablieren, Bedrohungen und fehlende Kontrollen zu identifizieren und einen weitaus gründlicheren Test in kürzerer Zeit durchzuführen.

Wenn möglich, ist bei der Durchführung einer Bewertung nach den Stufen 2 oder 3 der Zugang zu Entwicklern, Dokumentation, Code und Zugriff auf eine Testanwendung mit nicht produktiven Daten erforderlich. Penetrationstests, die auf diesen Ebenen durchgeführt werden, erfordern diese Zugriffsebene, die wir als *Hybride Überprüfungen* oder *Hybride Penetrationstests* nennen.

Andere Verwendungszwecke des ASVS

Abgesehen von der Verwendung zur Bewertung der Sicherheit einer Anwendung haben wir eine Reihe anderer möglicher Anwendungen für den ASVS identifiziert.

Detaillierter Leitfaden für eine Sicherheitsarchitektur

Eine der häufigeren Anwendungen des Application Security Verification Standards ist die Verwendung als Ressource für Sicherheitsarchitekten. In der *Sherwood Applied Business Security Architecture (SABSA)* fehlt eine Vielzahl an Informationen, die für eine gründliche Überprüfung einer sicheren Anwendungsarchitektur erforderlich sind. ASVS kann verwendet werden, um diese Lücken zu füllen, indem Sicherheitsarchitekten bessere Kontrollen für häufige Probleme, wie z.B. Best Practices für Datenschutz und Strategien für Eingabevalidierung, wählen können.

Als Ersatz für Standard Secure Coding Checklisten

Viele Organisationen können von der Übernahme des ASVS profitieren, indem sie sich für eine der drei Stufen entscheiden oder einen *Fork* des ASVS erstellen und die Anforderungen für jede Anwendungsrisikostufe domänenspezifisch ändern. Wir fördern diese Art der Abzweigung, solange die Rückverfolgbarkeit gewährleistet ist. Das heisst, wenn eine Anwendung die Anforderung 4.1 bestanden hat, soll dies für die Verzweigung dasselbe bedeuten wie für den sich weiter entwickelnde Standard.

Als Leitfaden für automatisierte Unit- und Integrationstests

Der ASVS ist so konzipiert, dass er in hohem Maße testbar ist - mit der einzigen Ausnahme der Anforderungen an die Architektur und Vermeidung bösartigen Codes. Durch die Erstellung von Unit- und Integrationstests, die auf spezifische und relevante Fuzzing- und Missbrauchsfälle testen, wird die Anwendung mit jedem einzelnen Build nahezu *selbstverifizierend*. Beispielsweise können zusätzliche Tests für die Testsuite eines Login-Controllers erstellt werden, die den Benutzernamen auf gebräuchliche Standardbenutzernamen, *Account Enumeration*, *Brute-Forcing*, LDAP- und SQL-Injektionen und XSS-Angriffe testen. In ähnlicher Weise sollte ein Test für das Kennwort übliche Standard-Kennwörter, Kennwortlänge, Null-Byte-Injektionen, Entfernen des Kennwortparameters, XSS und mehr umfassen.

Für Trainings zur sicheren Entwicklung

Der ASVS kann auch dazu verwendet werden, Merkmale sicherer Software zu definieren. Viele Trainingskurse zur "sicheren Kodierung" sind einfach ethische Hacking-Kurse mit nur einem kurzen Ausflug zu *Secure Coding*. Dies hilft den Entwicklern aber nicht unbedingt, sichereren Code zu schreiben. Stattdessen können sichere

Entwicklungskurse den ASVS verwenden, wobei der Schwerpunkt hier auf den *ProActive Controls* des ASVS liegt und nicht auf den OWASP Top 10 der Fehler, die man nicht machen sollte.

Als Treiber für die Sicherheit agil entwickelter Anwendungen

Der ASVS kann innerhalb eines agilen Entwicklungsprozesses als Rahmenwerk verwendet werden, um spezifische Aufgaben z.B. innerhalb eines Sprints zu definieren, die vom Team implementiert werden müssen, um ein sicheres Produkt zu erhalten. Ein Ansatz könnte sein: Beginnen Sie mit Stufe 1, prüfen Sie ihre Anwendung oder das System gemäß den ASVS-Anforderungen für die spezifizierte Stufe, finden Sie heraus, welche Kontrollen fehlen und erfassen Sie spezifische Tickets/Aufgaben im Backlog. Dies hilft bei der Priorisierung bestimmter Aufgaben (z.B. beim Grooming/Refinement) und macht die Sicherheit im agilen Prozess sichtbar. Dies kann auch zur Priorisierung von Revisionsprüfungen und Review-Tasks in der Organisation verwendet werden. Das kann bedeuten, dass eine bestimmte ASVS-Anforderung für ein bestimmtes Teammitglied ein Treiber für ein Codereview, ein Refactoring oder eine Auditierung sein kann und als *Technical Debt* im Backlog sichtbar wird, das zeitnah erledigt werden sollte.

Als Rahmen für die Steuerung von Dienstleistern zur Erstellung sicherer Software

ASVS ist ein großartiges Rahmenwerk, um bei der sicheren Softwarebeschaffung oder der Beschaffung von kundenspezifischen Entwicklungsdienstleistungen zu unterstützen. Der Auftraggeber kann einfach die Anforderung stellen, dass die Software, die er beschaffen bzw. beauftragen möchte, auf ASVS-Ebene *X* entwickelt werden muss, und vom Anbieter den Nachweis verlangen, dass die Software die ASVS-Ebene *X* erfüllt. Dies funktioniert optimal, wenn es mit dem Anhang zum OWASP-Vertrag für sichere Software (*OWASP Secure Software Contract Annex*) kombiniert wird.

V1: Anforderungen für Architektur, Design und die Bedrohungsmodellierung

Zielsetzung

Die Sicherheitsarchitektur ist in vielen Organisationen fast zu einer verschollenen Kunst verkommen. Die Tage des Enterprisearchitekten sind im Zeitalter von DevSecOps größtenteils vorbei. Der Bereich der Anwendungssicherheit muss hier verlorenes Terrain aufholen, agile Sicherheitsprinzipien übernehmen und gleichzeitig anerkannte sichere Architekturprinzipien wieder in die Software-Praxis einführen. Architektur ist nicht die Implementierung, sondern die Art und Weise, über ein Problem nachzudenken, das potenziell viele verschiedene Lösungsmöglichkeiten hat, und nicht nur eine einzige "richtige" Antwort besitzt. Nur allzu oft wird gerade die Sicherheit als unflexibel angesehen und verlangt von den Entwicklern, Code auf eine bestimmte Art und Weise zu reparieren, obwohl die Entwickler vielleicht eine viel bessere Möglichkeit kennen, das Problem zu lösen. Es gibt nicht die eine einfache Lösung für die Architektur, und so zu tun, als ob es nicht so wäre, ist ein Bärendienst für die Softwareentwicklung.

Eine spezifische Implementierung einer Webanwendung wird wahrscheinlich während ihrer gesamten Lebensdauer kontinuierlich überarbeitet, die Gesamtarchitektur wird sich dagegen nur selten ändern, sondern sich eher langsam weiter entwickeln. Eine Sicherheitsarchitektur unterscheidet sich hier nicht - wir brauchen heute eine Authentifizierung, wir werden morgen eine Authentifizierung benötigen, und wir werden sie in fünf Jahren brauchen. Wenn wir heute fundierte Entscheidungen treffen, können wir viel Aufwand, Zeit und Geld sparen, wenn wir architekturkonforme Lösungen auswählen und wiederverwenden. Vor einem Jahrzehnt beispielsweise wurde die Multi-Faktor-Authentifizierung nur selten implementiert.

Hätten Entwickler in ein einziges, sicheres Identitätsprovider-Modell investiert, wie z.B. eine föderierte SAML-Identität, könnte der Identitätsprovider einfach aktualisiert werden, um neue Anforderungen wie die NIST 800-63-Konformität einzubeziehen, ohne die Schnittstellen der ursprünglichen Anwendung zu ändern. Immer wenn viele Anwendungen die gleiche Sicherheitsarchitektur und damit die gleiche Komponente nutzen, profitieren sie alle gleichzeitig von diesem Upgrade. SAML wird jedoch nicht immer die beste oder geeignetste Authentifizierungslösung bleiben - es muss möglicherweise gegen andere Lösungen wie z.B. OpenID Connect ausgetauscht werden, wenn sich die Anforderungen ändern. Änderungen wie diese sind entweder kompliziert, so kostspielig, dass eine komplette Neuentwicklung erforderlich ist, oder ohne eine Sicherheitsarchitektur schlichtweg unmöglich.

In diesem Kapitel behandelt der ASVS die wichtigsten Aspekte jeder soliden Sicherheitsarchitektur:

- Verfügbarkeit
- Vertraulichkeit
- Verarbeitungsintegrität
- Nichtabstreitbarkeit
- Datenschutz

Alle dieser Sicherheitsprinzipien müssen als Bestandteil in allen Anwendungen integriert sein. Ein *Shift-Left* ist entscheidend, beginnend mit der Befähigung der Entwickler durch Checklisten für die sichere Kodierung, Mentoring und Schulungen, Kodierung und Tests, Aufbau, Bereitstellung, Konfiguration und Betrieb und endend mit unabhängigen Tests, um sicherzustellen, dass alle Sicherheitskontrollen vorhanden und funktionsfähig sind. Früher war der letzte Testschritt alles, was wir als Industrieunternehmen umgesetzt haben, aber das reicht nicht mehr aus, wenn Entwickler zehn- oder hundertmal am Tag Code in die Produktion schieben. Anwendungssicherheitsexperten müssen mit agilen Techniken Schritt halten. Das bedeutet, dass sie Entwicklerwerkzeuge einsetzen, den Code lernen und mit den Entwicklern zusammenarbeiten müssen, anstatt das Projekt monatelang mit Sicherheitsblockaden zu überhäufen, während alle anderen Projektbeteiligten längst weiter gearbeitet haben.

V1.1 Anforderungen an einen sicheren Softwareentwicklungslebenszyklus

#	Beschreibung	L1	L2	L3	CWE
1.1.1	Überprüfen Sie die Verwendung eines sicheren Softwareentwicklungslebenszyklus, der die Sicherheit in allen Entwicklungsphasen berücksichtigt. (C1)		✓	✓	
1.1.2	Überprüfen Sie bei jeder Designänderung oder Sprintplanung den Einsatz von Bedrohungsmodellen (*threat models*), um Bedrohungen zu identifizieren, Gegenmaßnahmen zu planen, angemessene Reaktionen auf Risiken zu erleichtern und Sicherheitstests anzuleiten.		✓	✓	1053
1.1.3	Vergewissern Sie sich, dass alle Userstories und Features funktionale Sicherheitseinschränkungen enthalten, wie z.B. *Als Benutzer sollte ich mein Profil anzeigen und bearbeiten können. Ich sollte nicht in der Lage sein, das Profil anderer Personen anzuzeigen oder zu bearbeiten*.				
	✓	✓	1110		
1.1.4	Überprüfen Sie die Dokumentation und Begründung aller Vertrauensgrenzen, Komponenten und wichtigen Datenflüsse der Anwendung.		✓	✓	1059
1.1.5	Überprüfen Sie die Definition und Sicherheitsanalyse der High-Level-Architektur der Anwendung und aller angeschlossenen Remotedienste. (C1)		✓	✓	1059
1.1.6	Überprüfen Sie die Implementierung zentralisierter, einfacher (Wirtschaftlichkeit des Designs), geprüfter, sicherer und wiederverwendbarer Sicherheitskontrollen, um doppelte, fehlende, unwirksame oder unsichere Kontrollen zu vermeiden. (C10)		✓	✓	637
1.1.7	Überprüfen Sie die Verfügbarkeit einer Checkliste für sichere Kodierung, Sicherheitsanforderungen, von Leitfäden oder Richtlinien für alle Entwickler und Tester.		✓	✓	637

V1.2 Anforderungen an die Authentifizierungsarchitektur

Beim Entwurf der Authentifizierung spielt es keine Rolle, ob Sie über eine starke, hardwarebasierte Multi-Faktor-Authentifizierung verfügen, wenn ein Angreifer gleichzeitig ein Konto zurücksetzen kann, indem er

lediglich ein Callcenter anruft und allgemein bekannte Fragen beantwortet. Beim Identitätsnachweis müssen **alle** Authentifizierungswege die gleiche Stärke aufweisen.

#	Beschreibung	L1	L2	L3	CWE
1.2.1	Überprüfen Sie, ob die Kommunikation zwischen Anwendungskomponenten, einschließlich APIs, Middleware und Datenschichten, authentifiziert ist und individuelle Benutzerkonten verwendet. (C3)		✓	✓	306
1.2.2	Vergewissern Sie sich, dass die Anwendung einen einzigen geprüften Authentifizierungsmechanismus verwendet, der bekanntermaßen sicher ist, auf eine stärkere Authentifizierung ausgedehnt werden kann und über eine ausreichende Protokollierung und Überwachung verfügt, um Kontomissbrauch oder -verletzungen zu erkennen.				
	✓	✓	306		
1.2.3	Vergewissern Sie sich, dass alle Authentifizierungspfade und Identitätsmanagement-APIs eine einheitliche Stärke der Authentifizierungssicherheitskontrolle implementieren, so dass es keine schwächeren Alternativen für das Risiko der Anwendung gibt.				
		✓	✓	306	

V1.3 Architektonische Anforderungen an das Sitzungsmanagement

Dies dient als ein Platzhalter für zukünftige architektonische Anforderungen.

V1.4 Anforderungen an die Architektur der Zugriffskontrolle

#	Beschreibung	L1	L2	L3	CWE
1.4.1	Vergewissern Sie sich, dass vertrauenswürdige Durchsetzungspunkte, z.B. an Gateways für Zugangskontrolle, Servern und *serverlosen* Funktionen, die Zugangskontrollen durchsetzen. Setzen Sie niemals Zugriffskontrollen nur auf dem Client durch.		✓	✓	602
1.4.2	Überprüfen Sie, ob die gewählte Lösung zur Zugangskontrolle flexibel genug ist, um die Anforderungen der Anwendung zu erfüllen.		✓	✓	284
1.4.3	Überprüfen Sie die Durchsetzung des Prinzips der wenigsten erforderlichen Privilegien (*least privilege*) bei Funktionen, Dateien, URLs, Controllern, Diensten und anderen Ressourcen. Dies impliziert den Schutz vor Spoofing und die Erweiterung von Privilegien.		✓	✓	272
1.4.4	Vergewissern Sie sich, dass die Kommunikation zwischen Anwendungskomponenten, einschließlich APIs, Middleware und Datenschichten, mit den am wenigsten erforderlichen Berechtigungen (*least privilege*) durchgeführt wird.				
	(C3)	✓	✓	272	
1.4.5	Überprüfen Sie, dass die Anwendung einen einzigen und gut erprobten Zugriffskontrollmechanismus für den Zugriff auf geschützte Daten und Ressourcen verwendet. Alle Anfragen müssen diesen einzigen Mechanismus durchlaufen, um Kopieren und Einfügen oder unsichere Alternativpfade zu vermeiden.				
	(C7)	✓	✓	284	

1.4.6	Stellen Sie sicher, dass eine attribut- oder merkmalsbasierte Zugriffskontrolle verwendet wird, wobei der Code die Berechtigung des Benutzers für ein Merkmal/Datenelement und nicht nur seine Rolle prüft. Die Berechtigungen sollten weiterhin über Rollen zugewiesen werden. (C7)		✓	✓	275

V1.5 Anforderungen an die Eingabe- und Ausgabe-Architektur

In Version 4.0 haben wir uns von dem Begriff *server-seitig* als etablierte Vertrauensgrenze verabschiedet. Die Vertrauensgrenze ist immer noch ein Thema - Entscheidungen über nicht vertrauenswürdige Browser oder Client-Geräte zu treffen, können weiterhin umgangen werden. In den heutigen Standardarchitekturen hat sich jedoch der Punkt der Vertrauensdurchsetzung (*Trust Enforcement*) dramatisch verändert. Wenn in dem ASVS der Begriff *vertrauenswürdige Serviceschicht* verwendet wird, meinen wir daher jeden vertrauenswürdigen Durchsetzungspunkt, unabhängig vom Standort, wie z.B. einen Microservice, eine *serverless* API, eine serverseitige, eine vertrauenswürdige API auf einem Clientgerät, usw.

#	Beschreibung	L1	L2	L3	CWE
1.5.1	Vergewissern Sie sich, dass die Eingabe- und Ausgabeanforderungen klar definieren, wie die Daten auf der Grundlage des Typs, des Inhalts und der anwendbaren Gesetze, Vorschriften und anderen Richtlinien zu behandeln und zu verarbeiten sind.				
	✓	✓	1029		
1.5.2	Stellen Sie sicher, dass bei der Kommunikation mit nicht vertrauenswürdigen Clients keine Serialisierung verwendet wird. Wenn dies nicht möglich ist, stellen Sie sicher, dass angemessene Integritätskontrollen (und möglicherweise Verschlüsselung, wenn sensible Daten gesendet werden) durchgesetzt werden, um Deserialisierungsangriffe einschließlich der Injektion von Objekten zu verhindern.		✓	✓	502
1.5.3	Vergewissern Sie sich, dass die Eingabevalidierung auf einer vertrauenswürdigen Serviceschicht durchgesetzt wird. (C5)		✓	✓	602
1.5.4	Vergewissern Sie sich, dass die Ausgabekodierung in der Nähe des oder durch den Interpreter erfolgt, für den sie bestimmt ist. (C4)		✓	✓	116

V1.6 Anforderungen an die kryptographische Architektur

Anwendungen müssen mit einer Architektur für eine starke Kryptographie entworfen werden, um die Datenbestände gemäß ihrer Klassifizierung zu schützen. Alles zu verschlüsseln ist verschwenderisch, nichts zu verschlüsseln ist rechtlich fahrlässig. Es muss ein Gleichgewicht gefunden werden, in der Regel während des Architektur- oder High-Level-Designs, bei Design-Sprints oder Architektur-Spikes. Wenn Sie die Kryptographie nach und nach entwerfen oder nachrüsten, wird die sichere Implementierung unweigerlich viel mehr kosten, als wenn Sie sie gleich von Anfang an einbauen würden.

Architektonische Anforderungen sind der gesamten Codebasis inhärent und daher schwierig zu vereinheitlichen oder zu integrieren. Die architektonischen Anforderungen müssen bei den Kodierungsstandards während der gesamten Umsetzungsphase berücksichtigt werden und sollten während der Sicherheitsarchitektur, bei Peer- oder Code-Reviews oder bei Retrospektiven überprüft werden.

#	Beschreibung	L1	L2	L3	CWE
1.6.1	Vergewissern Sie sich, dass es eine explizite Richtlinie für die Verwaltung von kryptografischen Schlüsseln gibt und dass der Lebenszyklus eines kryptografischen Schlüssels einem Schlüsselverwaltungsstandard wie NIST SP 800-57 folgt.		✓	✓	320

1.6.2	Stellen Sie sicher, dass Verwender von kryptografischen Diensten Schlüsselmaterial und andere Geheimnisse schützen, indem Sie Key Vaults oder API-basierte Alternativen verwenden.		✓	✓	320
1.6.3	Vergewissern Sie sich, dass alle Schlüssel und Passwörter austauschbar sind und Teil eines genau definierten Prozesses zur Neuverschlüsselung sensibler Daten sind.		✓	✓	320
1.6.4	Vergewissern Sie sich, dass symmetrische Schlüssel, Kennwörter oder API-Geheimnisse, die von Kunden erzeugt oder mit ihnen geteilt werden, nur zum Schutz von Geheimnissen mit geringem Risiko, wie z.B. die Verschlüsselung des lokalen Speichers, oder für vorübergehende, kurzzeitige Verwendungen wie die Verschleierung von Parametern, verwendet werden. Das Teilen von Geheimnissen mit Kunden ist gleichzusetzen mit dem Teilen von Klartext und sollte auch architektonisch als solches behandelt werden.				
	✓	✓	320		

V1.7 Architektonische Anforderungen an Fehlerbehandlung, Protokollierung und Auditierung

#	Beschreibung	L1	L2	L3	CWE
1.7.1	Vergewissern Sie sich, dass im gesamten System ein einheitliches Protokollformat und ein einheitlicher Protokollierungsansatz verwendet wird. (C9)		✓	✓	1009
1.7.2	Vergewissern Sie sich, dass die Protokolle sicher an ein, vorzugsweise entferntes, System zur Analyse, Erkennung, Alarmierung und Eskalation übertragen werden.				
(C9)		✓	✓		

V1.8 Architektonische Anforderungen für Datenschutz und Privatsphäre

#	Beschreibung	L1	L2	L3	CWE
1.8.1	Vergewissern Sie sich, dass alle sensiblen Daten identifiziert und in Schutzstufen eingeteilt werden.		✓	✓	
1.8.2	Vergewissern Sie sich, dass alle Schutzebenen mit einer Reihe von Schutzanforderungen verbunden sind, z.B. Verschlüsselungsanforderungen, Integritätsanforderungen, Aufbewahrung, Datenschutz und andere Vertraulichkeitsanforderungen, und dass diese in der Architektur angewendet werden.		✓	✓	

V1.9 Architektonische Anforderungen an die Kommunikation

#	Beschreibung	L1	L2	L3	CWE
1.9.1	Überprüfen Sie, ob die Anwendung die Kommunikation zwischen Komponenten verschlüsselt, insbesondere wenn sich diese Komponenten in verschiedenen Containern, Systemen, Standorten oder Cloud-Anbietern befinden. (C3)		✓	✓	319
1.9.2	Überprüfen Sie, dass die Anwendungskomponenten die Authentizität jeder Seite in einer Kommunikationsverbindung überprüfen, um "Person-in-the-Middle"-Angriffe zu verhindern. Beispielsweise sollten Anwendungskomponenten TLS-Zertifikate und -Ketten validieren.		✓	✓	295

V1.10 Architekturanforderungen hinsichtlich bösartiger Software

#	Beschreibung	L1	L2	L3	CWE
1.10.1	Vergewissern Sie sich, dass ein Quellcode-Kontrollsystem verwendet wird, mit Verfahren, die sicherstellen, dass beim Einchecken zugehörige Tickets referenziert werden. Das Quellcode-Kontrollsystem sollte über eine Zugriffskontrolle und identifizierbare Benutzer verfügen, um die Rückverfolgbarkeit von Änderungen zu ermöglichen.		✓	✓	284

V1.11 Architekturanforderungen an die Geschäftslogik

#	Beschreibung	L1	L2	L3	CWE
1.11.1	Überprüfen Sie die Definition und Dokumentation aller Anwendungskomponenten in Bezug auf die von ihnen bereitgestellten Geschäfts- oder Sicherheitsfunktionen.		✓	✓	1059
1.11.2	Vergewissern Sie sich, dass alle kritischen Abläufe der Geschäftslogik, einschließlich Authentifizierung, Sitzungsverwaltung und Zugriffskontrolle, keinen unsynchronisierten Zustand aufweisen.		✓	✓	362
1.11.3	Vergewissern Sie sich, dass alle kritischen Abläufe der Geschäftslogik, einschließlich Authentifizierung, Sitzungsverwaltung und Zugriffskontrolle, thread-sicher und resistent gegen Prüfzeit- und Nutzungszeit-Nebenläufigkeiten sind.			✓	367

V1.12 Architekturanforderungen an sicheres Hochladen von Dateien

#	Beschreibung	L1	L2	L3	CWE
1.12.1	Überprüfen Sie, ob die vom Benutzer hochgeladenen Dateien außerhalb des Web-Stammverzeichnisses gespeichert sind.		✓	✓	552
1.12.2	Vergewissern Sie sich, dass die vom Benutzer hochgeladenen Dateien - falls sie angezeigt oder von der Anwendung heruntergeladen werden müssen - entweder durch Octet-Stream-Downloads oder von einer nicht verwandten Domäne, wie z.B. einem Cloud File Storage Bucket, bereitgestellt werden. Implementieren Sie eine geeignete Content Security Policy (CSP), um das Risiko von XSS-Vektoren oder anderen Angriffen auf die hochgeladene Datei zu reduzieren.		✓	✓	646

V1.13 Architekturanforderungen für APIs

Dies dient als Platzhalter für zukünftige Architekturanforderungen.

V1.14 Architekturanforderungen für die Konfiguration

#	Beschreibung	L1	L2	L3	CWE
1.14.1	Überprüfen Sie die Verwendung eindeutiger oder spezieller Betriebssystemkonten mit niedriger Privilegierung für alle Anwendungskomponenten, Dienste und Server.				
(C3)		✓	✓	250	
1.14.2	Überprüfen Sie die Trennung von Komponenten unterschiedlicher Vertrauensebenen durch klar definierte Sicherheitskontrollen, Firewall-Regeln, API-Gateways, Reverse-Proxies, Cloud-basierte Sicherheitsgruppen				

	oder vergleichbare Mechanismen.				
	✓	✓	923		
1.14.3	Stellen Sie sicher, dass bei der Bereitstellung von Binärdateien auf nicht vertrauenswürdigen Geräten binäre Signaturen, vertrauenswürdige Verbindungen und verifizierte Endpunkte verwendet werden.		✓	✓	494
1.14.4	Vergewissern Sie sich, dass die Build-Pipeline vor veralteten oder unsicheren Komponenten warnt und entsprechende Maßnahmen ergreift.				
	✓	✓	1104		
1.14.5	Vergewissern Sie sich, dass die Build-Pipeline einen Build-Schritt enthält, um die sichere Bereitstellung der Anwendung automatisch zu erstellen und zu verifizieren, insbesondere wenn es sich bei der Anwendungsinfrastruktur um Software handelt, wie z.B. Build-Skripts für Cloud-Umgebungen.		✓	✓	
1.14.6	Vergewissern Sie sich, dass Anwendungsimplementierungen auf der Netzwerkebene angemessen sandboxed, containerisiert und/oder isoliert sind, um Angriffe zu verzögern und Angreifer davon abzuhalten, andere Anwendungen anzugreifen, insbesondere wenn sie sensible oder gefährliche Aktionen wie Deserialisierung durchführen. (C5)		✓	✓	265
1.14.7	Stellen Sie sicher, dass die Anwendung keine nicht mehr unterstützte, unsicheren oder veralteten clientseitigen Technologien wie NSAPI-Plugins, Flash, Shockwave, ActiveX, Silverlight, NACL oder clientseitige Java-Applets verwendet.		✓	✓	477

Verweise

Für weitere Informationen siehe auch:

- OWASP Threat Modeling Cheat Sheet
- OWASP Attack Surface Analysis Cheat Sheet
- OWASP Threat modeling
- OWASP Secure SDLC Cheat Sheet
- Microsoft SDL
- NIST SP 800-57

V2: Anforderungen für die Prüfung der Authentifizierung

Zielsetzung

Authentifizierung ist der Akt der Feststellung oder Bestätigung, dass jemand (oder etwas) glaubwürdig ist und dass die von einer Person oder über ein Gerät gemachten Behauptungen korrekt sind, widerstandsfähig gegen Nachahmung sind und die Wiederherstellung oder das Abfangen von Passwörtern verhindern.

Als der ASVS zum ersten Mal veröffentlicht wurde, war außerhalb von Hochsicherheitssystemen die Kombination aus Benutzername + Passwort die gebräuchlichste Form der Authentifizierung. Die Multi-Faktor-Authentifizierung (MFA) war zwar in Sicherheitskreisen allgemein akzeptiert, aber wurde anderswo nur selten verlangt. Seit die Zahl der Passwortverletzungen zunahm, machte der Gedanke, dass Benutzernamen irgendwie vertraulich und Passwörter unbekannt sind, viele Sicherheitskontrollen untragbar. So betrachtet beispielsweise NIST 800-63 die Benutzernamen und wissensbasierte Authentifizierung (KBA) als öffentliche Information, SMS- und E-Mail-Benachrichtigungen als *eingeschränkte* Authentifizierungstypen und Passwörter als potentiell verletzt. Diese Realität macht wissensbasierte Authentifikatoren, SMS- und E-Mail-

Wiederherstellung, Passwortverläufe, Komplexität und Rotationskontrollen nutzlos. Diese Kontrollen waren schon immer wenig hilfreich und zwangen die Benutzer dazu, sich alle paar Monate neue schwache Passwörter auszudenken, aber mit der Veröffentlichung von über 5 Milliarden Verletzungen von Benutzernamen und Passwörtern ist es an der Zeit, weiter voranzuschreiten.

Von allen Abschnitten innerhalb des ASVS haben sich die Kapitel Authentifizierung und Sitzungsverwaltung am häufigsten verändert. Die Einführung einer führenden effektiven, wissenschaftlich fundierten Praxis wird für viele eine Herausforderung sein, und das ist völlig in Ordnung. Wir müssen jetzt mit dem Übergang zu einer *Post-Passwort* Zukunft starten.

NIST 800-63 - Ein moderner, wissenschaftlich fundierter Authentifizierungsstandard

NIST 800-63b ist ein moderner, wissenschaftlich fundierter Standard und stellt unabhängig von dessen Anwendbarkeit die beste aktuell verfügbare Empfehlung dar. Der Standard ist für alle Organisationen auf der ganzen Welt hilfreich, aber besonders relevant für US-Behörden und diejenigen, die mit US-Behörden zu tun haben.

Die Terminologie nach NIST 800-63 kann anfangs etwas verwirrend sein, besonders wenn man nur die Authentifizierung per Benutzername und Passwort gewohnt ist. Fortschritte in der modernen Authentifizierung sind notwendig, daher müssen wir eine Terminologie einführen, die in Zukunft allgemein üblich sein wird. Aber wir können auch die Verständnisschwierigkeiten nachvollziehen bis sich die ganze Branche auf diese neuen Begrifflichkeiten eingestellt hat. Wir haben zur Unterstützung am Ende dieses Kapitels ein Glossar bereitgestellt. Wir haben viele Anforderungen umformuliert, um dem Zweck der Anforderung zu entsprechen und nich nur der Absichtserklärung zu dienen. Zum Beispiel verwendet der ASVS den Begriff *Passwort*, obwohl das NIST in diesem Standard *memorized secret* als Begriff verwendet.

ASVS V2-Authentifizierung, V3-Sitzungsmanagement und in geringerem Maße auch V4-Zugangskontrollen wurden so angepasst, dass sie eine übereinstimmende Teilmenge ausgewählter NIST 800-63b-Kontrollen darstellen, die sich auf allgemeine Bedrohungen und häufig ausgenutzte Authentifizierungsschwächen konzentrieren. Wenn eine vollständige Konformität zu NIST 800-63 erforderlich ist, dann ziehen Sie bitte das Dokument zu NIST 800-63 zu Rate.

Auswahl einer geeigneten NIST AAL-Stufe

Der Application Security Verification Standard hat versucht, die ASVS Stufe 1 den NIST AAL1-Anforderungen, die Stufe 2 den AAL2-Anforderungen und die Stufe 3 den AAL3-Anforderungen zuzuordnen. Der Ansatz der ASVS-Stufe 1 als "wesentliche" Kontrollen ist jedoch nicht unbedingt die richtige AAL-Stufe zur Prüfung einer Anwendung oder API. Wenn es sich bei der Anwendung beispielsweise um eine Anwendung der Stufe 3 handelt oder wenn die Anwendung die gesetzlichen Anforderungen an AAL3 erfüllt, sollte in den Abschnitten V2 und V3 Session Management die Stufe 3 gewählt werden. Die Wahl der NIST-konformen Authentifizierungs-Assertion-Level (AAL) sollte gemäß den Richtlinien von NIST 800-63b erfolgen, wie in *Selecting AAL* in NIST 800-63b Abschnitt 6.2 festgelegt.

Die Erklärung dazu

Applications can always exceed the current level's requirements, especially if modern authentication is on an application's roadmap. Previously, the ASVS has required mandatory MFA. NIST does not require mandatory MFA. Therefore, we have used an optional designation in this chapter to indicate where the ASVS encourages but does not require a control. The following keys are used throughout this standard:

Anwendungen können immer die Anforderungen der aktuellen Stufe überschreiten, insbesondere wenn eine moderne Authentifizierung auf der Roadmap der Anwendung steht. Früher hat der ASVS obligatorisch eine MFA verlangt. Das NIST verlangt keine obligatorische MFA. Daher haben wir in diesem Kapitel eine optionale Bezeichnung verwendet, um anzugeben, wo der ASVS eine Kontrolle empfiehlt, aber nicht verlangt. Die folgenden Schlüssel werden in diesem Standard verwendet:

Kennzeichnung	Beschreibung
	Nicht erforderlich

- o Empfohlen, aber nicht erforderlich
- ✓ Erforderlich

V2.1 Anforderungen an die Passwortsicherheit

Passwörter, die von NIST 800-63 als *Memorized Secrets* bezeichnet werden, umfassen Passwörter, PINs, Entsperrmuster, die Auswahl des richtigen Kätzchens oder eines anderen Bildelements sowie Passphrasen. Sie werden im Allgemeinen als "etwas, das Sie kennen" betrachtet und oft als Ein-Faktor-Authentifikatoren verwendet. Die weitere Verwendung der Ein-Faktor-Authentifizierung ist mit erheblichen Herausforderungen verbunden. Dazu gehören Milliarden von gültigen Kombinationen aus Benutzernamen und Passwörtern, die im Internet offengelegt werden, Standardpasswörter oder schwache Passwörter, *Rainbow Tables* und Wörterbücher der gängigsten Passwörter.

Die Anwendungen sollten die Benutzer nachdrücklich dazu auffordern, sich für die Multi-Faktor-Authentifizierung anzumelden, und sollten es den Benutzern ermöglichen, Token wiederzuverwenden, die sie bereits besitzen, wie z.B. FIDO- oder U2F-Token, oder sich mit einem Identityprovider zu verbinden, der Multi-Faktor-Authentifizierung anbietet.

Identityprovider (__C__redential __s__ervice __p__roviders, CSPs) bieten eine föderierte Identität für Benutzer an. Benutzer verfügen oft über mehr als eine Identität mit mehreren CSPs, z.B. eine Unternehmensidentität mit *Azure AD*, *Okta*, *Ping Identity* sowie *Google* oder eine Social-Media-Identität mit *Facebook*, *Twitter*, *Google* oder *WeChat*, um nur einige gängige Alternativen zu nennen. Diese Liste ist keine Empfehlung dieser Unternehmen oder Dienste, sondern lediglich eine Ermutigung für Entwickler, die Realität zu berücksichtigen, dass viele Benutzer mehrere etablierte Identitäten haben. Organisationen sollten entsprechend dem Risikoprofil bezüglich der Stärke der Identitätsprüfung durch den CSP die Integration mit bestehenden Benutzeridentitäten in Betracht ziehen. So ist es beispielsweise unwahrscheinlich, dass eine Regierungsorganisation eine Social-Media-Identität als Login für sensible Systeme akzeptiert, da es einfach möglich ist, gefälschte oder *Wegwerf-Identitäten* zu erstellen, während ein Unternehmen für Handyspiele möglicherweise eine Integration mit großen Social-Media-Plattformen vornehmen muss, um seine aktive Spielerbasis zu vergrößern.

#	Beschreibung	L1	L2	L3	CWE	NIST §
2.1.1	Vergewissern Sie sich, dass die Passwörter der Benutzer mindestens 12 Zeichen lang sind. (C6)	✓	✓	✓	521	5.1.1.2
2.1.2	Stellen Sie sicher, dass Passwörter mit mindestens 64 Zeichen erlaubt sind. (C6)	✓	✓	✓	521	5.1.1.2
2.1.3	Stellen Sie sicher, dass Passwörter Leerzeichen enthalten können und dass kein Abschneiden von Zeichen durchgeführt wird. Mehrere aufeinander folgende Leerzeichen KÖNNEN optional zusammengeführt werden. (C6)	✓	✓	✓	521	5.1.1.2
2.1.4	Überprüfen Sie, ob Unicode-Zeichen in Kennwörtern zulässig sind. Ein einzelner Unicode-Codepoint wird als Zeichen betrachtet, daher sollten 12 Emoji- oder 64 Kanji-Zeichen gültig und erlaubt sein.	✓	✓	✓	521	5.1.1.2
2.1.5	Stellen Sie sicher, dass Benutzer ihr Passwort ändern können.	✓	✓	✓	620	5.1.1.2
2.1.6	Vergewissern Sie sich, dass die Funktionalität zur Kennwortänderung das aktuelle und das neue Kennwort des Benutzers erfordert.	✓	✓	✓	620	5.1.1.2
2.1.7	Vergewissern Sie sich, dass die bei der Konto-Registrierung, Anmeldung und Passwortänderung übermittelten Passwörter entweder lokal (z.B. die 1.000 oder 10.000 häufigsten Passwörter, die mit der Passwortrichtlinie des Systems übereinstimmen) oder mit Hilfe einer externen API gegen einen Satz von verletzten Passwörtern	✓	✓	✓	521	5.1.1.2

	geprüft werden. Bei Verwendung einer API sollte ein Nachweis über die Unkenntnis oder ein anderer Mechanismus verwendet werden, um sicherzustellen, dass das Klartext-Passwort nicht gesendet oder zur Überprüfung des Verletzungsstatus des Passworts verwendet wird. Wenn das Passwort verletzt wird, muss die Anwendung den Benutzer auffordern, ein neues, nicht verletztes Passwort zu setzen. (C6)					
2.1.8	Vergewissern Sie sich, dass eine Anzeige zur Passwort-Stärke zur Verfügung gestellt wird, um Benutzern die Einstellung eines stärkeren Passworts zu erleichtern.	✓	✓	✓	521	5.1.1.2
2.1.9	Stellen Sie sicher, dass es keine Regeln für Zusammensetzung des Passworts gibt, die die Art der zulässigen Zeichen einschränken. Es sollte keine Groß- oder Kleinschreibung, Zahlen oder Sonderzeichen vorgeschrieben werden. (C6)	✓	✓	✓	521	5.1.1.2
2.1.10	Vergewissern Sie sich, dass es keine Anforderungen an die periodischen Rotation der Anmeldedaten oder die Kennworthistorie gibt.	✓	✓	✓	263	5.1.1.2
2.1.11	Stellen Sie sicher, dass die *Paste*-Funktionalität, Browser-Passworthilfen und externe Passwortmanager zugelassen sind.	✓	✓	✓	521	5.1.1.2
2.1.12	Prüfen Sie, dass der Benutzer auf Plattformen, die dies nicht als native Funktionalität haben, wählen kann, entweder vorübergehend das gesamte maskierte Kennwort anzuzeigen oder vorübergehend das letzte eingetippte Zeichen des Kennworts anzuzeigen.	✓	✓	✓	521	5.1.1.2

Note: The goal of allowing the user to view their password or see the last character temporarily is to improve the usability of credential entry, particularly around the use of longer passwords, passphrases, and password managers. Another reason for including the requirement is to deter or prevent test reports unnecessarily requiring organizations to override native platform password field behavior to remove this modern user-friendly security experience.

Hinweis: Das Ziel, dem Benutzer die Möglichkeit zu geben, sein Passwort einzusehen oder vorübergehend das letzte Zeichen zu sehen, ist es, die Benutzerfreundlichkeit bei der Eingabe von Anmeldeinformationen zu verbessern, insbesondere bei der Verwendung längerer Passwörter, Passphrasen und Passwort-Manager. Ein weiterer Grund für die Aufnahme der Anforderung ist das Verhindern von Testberichten, welche Organisationen unnötigerweise dazu zwingen, das native Verhalten der Plattform für Passwortfelder zu überschreiben, um diese moderne benutzerfreundliche Sicherheitserfahrung zu entfernen.

V2.2 Allgemeine Anforderungen an den Authentifizierer

Die Agilität des Authentisierers ist für zukunftssichere Anwendungen unerlässlich. Refactorings der Authentisierers sind notwendig, um zusätzliche Authentifikatoren gemäß den Benutzerpräferenzen zuzulassen und um veraltete oder unsichere Authentifikatoren in geordneter Weise aus dem Verkehr zu ziehen.

Das NIST betrachtet E-Mail und SMS als *eingeschränkte* Authentifizierertypen, und es ist wahrscheinlich, dass sie irgendwann in der Zukunft aus dem NIST 800-63 und damit dem ASVS entfernt werden. Anwendungen sollten in ihrer Roadmap eine Authentifizierung einplanen, die keine Verwendung von E-Mail oder SMS erfordert.

#	Beschreibung	L1	L2	L3	CWE	NIST §
2.2.1	Prüfen Sie, dass Anti-Automatisierungskontrollen wirksam sind, wenn es darum geht, Angriffe auf verletze Anmeldedaten, Brute Forcing und Kontosperren zu minimieren. Zu diesen Kontrollen gehören u.a. das Blockieren der am häufigsten gebrochenen Passwörter, Soft-Lockouts, Rate Limiting, CAPTCHA, zunehmende	✓	✓	✓	307	5.2.2 / 5.1.1.2 / 5.1.4.2 / 5.1.5.2

	Verzögerungen zwischen den Anmeldeversuchen, IP-Adressbeschränkungen oder risikobasierte Einschränkungen bezüglich des Standorts, der erste Anmeldung auf einem Gerät, der letzten Versuche, das Konto zu entsperren. Stellen Sie sicher, dass nicht mehr als 100 Fehlversuche pro Stunde bei einem einzelnen Konto möglich sind.					
2.2.2	Prüfen Sie, dass der Einsatz schwacher Authentifikatoren (wie SMS und E-Mail) auf die sekundäre Verifizierung und Genehmigung der Transaktion beschränkt ist und nicht als Ersatz für sicherere Authentifizierungsmethoden dient. Vergewissern Sie sich, dass stärkere Methoden angeboten werden, bevor schwache Methoden eingesetzt werden, dass sich die Benutzer der Risiken bewusst sind, oder dass geeignete Maßnahmen zur Begrenzung der Risiken einer Kontoverletzung getroffen werden.	✓	✓	✓	304	5.2.10
2.2.3	Vergewissern Sie sich, dass nach Aktualisierungen der Authentifizierungsdetails, wie z.B. das Zurücksetzen von Anmeldedaten, E-Mail- oder Adressänderungen, Anmeldungen aus unbekannten oder risikoreichen Standorten, Sicherheitsbenachrichtigungen an Benutzer gesendet werden. Die Verwendung von Push-Benachrichtigungen - anstelle von SMS oder E-Mail - ist vorzuziehen, aber bei Fehlen von Push-Benachrichtigungen sind SMS oder E-Mail akzeptabel, solange in der Benachrichtigung keine sensiblen Informationen offengelegt werden.	✓	✓	✓	620	
2.2.4	Überprüfen Sie die Widerstandsfähigkeit gegen Phishing, z.B. durch die Verwendung von Multi-Faktor-Authentifizierung, kryptografische Geräte mit Bestätigung (z.B. Online-Schlüssel mit einer Push-Möglichkeit zur Authentifizierung) oder auf höheren AAL-Ebenen clientseitige Zertifikate.			✓	308	5.2.5
2.2.5	Stellen Sie sicher, dass dort, wo ein Identity Provider (CSP) und die Anwendung, die die Authentifizierung verifiziert, getrennt sind, Mutual TLS zwischen den beiden Endpunkten vorhanden ist.			✓	319	5.2.6
2.2.6	Überprüfen Sie die Widerstandsfähigkeit gegenüber Replayangriffen durch den vorgeschriebenen Einsatz von OTP-Geräten, kryptografischen Authentifikatoren oder Backupcodes.			✓	308	5.2.8
2.2.7	Verifizieren Sie die Absicht, sich zu authentifizieren, indem Sie die Eingabe eines OTP-Tokens oder eine vom Benutzer initiierte Aktion, wie z.B. einen Tastendruck auf einem FIDO-Hardwareschlüssel, verlangen.			✓	308	5.2.9

V2.3 Anforderungen an den Lebenszyklus des Authentisierers

Als Authentisierer dienen Passwörter, Soft-Token, Hardware-Token und biometrische Geräte. Der Lebenszyklus von Authentifikatoren ist für die Sicherheit einer Anwendung entscheidend - wenn jemand ein Konto ohne Identitätsnachweis selbst registrieren kann, kann es kaum Vertrauen in die Identitätsprüfung geben. Für Social-Media-Sites wie Reddit ist das völlig in Ordnung. Bei Banksystemen ist ein stärkerer Fokus auf die Registrierung und Ausgabe von Anmeldedaten und Geräten für die Sicherheit der Anwendung entscheidend.

Hinweis: Passwörter dürfen keine maximale Lebensdauer haben oder einer Passwort-Rotation unterliegen. Passwörter sollten stattdessen auf eine Verletzung überprüft und nicht regelmäßig ersetzt werden.

#	Beschreibung	L1	L2	L3	CWE	NIST §
2.3.1	Die vom System generierten Initialpasswörter oder Aktivierungscodes MÜSSEN sicher zufällig generiert werden, mindestens 6 Zeichen lang sein und KÖNNEN Buchstaben und Zahlen enthalten und sollten nach einer kurzen Zeitspanne ablaufen. Diese initialen Anmeldedaten dürfen nicht zum Langzeit-Passwort werden.	✓	✓	✓	330	5.1.1.2 / A.3
2.3.2	Vergewissern Sie sich, dass die Registrierung und die Verwendung der vom Teilnehmer bereitgestellten Authentifizierungsgeräte unterstützt werden, wie z.B. U2F- oder FIDO-Token.		✓	✓	308	6.1.3
2.3.3	Vergewissern Sie sich, dass die Anweisungen zur Erneuerung mit genügend Zeitvorlauf gesendet werden, um zeitgebundene Authentifizierer erneuern zu können.		✓	✓	287	6.1.4

V2.4 Anforderungen für die Speicherung von Anmeldedaten

This section cannot be penetration tested, so controls are not marked as L1. However, this section is of vital importance to the security of credentials if they are stolen, so if forking the ASVS for an architecture or coding guideline or source code review checklist, please place these controls back to L1 in your private version.

Architekten und Entwickler sollten sich bei der Erstellung oder dem Refactoring von Code an die Ausführungen dieses Kapitels halten. Dieser Teil kann nur durch Quellcodeüberprüfung oder durch sichere Unit- oder Integrationstests vollständig überprüft werden. Penetrationstests können keines dieser Probleme identifizieren.

Die Liste der genehmigten *One-way Key Derivation Functions* ist in NIST 800-63 B Abschnitt 5.1.1.2 und in BSI Kryptographische Verfahren: Empfehlungen und Schlussellängen (2018) ausführlich beschrieben. Anstelle dieser Auswahlmöglichkeiten können die neuesten nationalen oder regionalen Algorithmen und Schlüssellängenstandards gewählt werden.

Dieser Abschnitt kann nicht mittels Penetrationstests geprüft werden, daher sind die Kontrollen nicht mit der Stufe 1 gekennzeichnet. Dieser Abschnitt ist jedoch von entscheidender Bedeutung für die Sicherheit von Anmeldedaten, für den Fall, dass diese gestohlen werden. Wenn Sie einen Fork des ASVSs für eine Architektur- oder Kodierungsrichtlinie oder eine Checkliste zur Überprüfung des Quellcodes erstellt haben, platzieren Sie diese Kontrollen bitte in Ihrer eigenen Version zurück auf Stufe 1.

#	Beschreibung	L1	L2	L3	CWE	NIST §
2.4.1	Vergewissern Sie sich, dass die Passwörter in einer Form gespeichert sind, die gegen Offline-Angriffe resistent ist. Passwörter MÜSSEN gesalzen (*salted*) und unter Verwendung einer anerkannten Einweg-Schlüsselableitung (*One-way Key Derivation*) oder einer Passwort-Hashing-Funktion gehasht werden. Die Funktionen zur Einweg-Schlüsselableitung und zum Passwort-Hashing nehmen zur Generierung eines Passwort-Hashes ein Passwort, einen Salt und einen Kostenfaktor als Eingabe. (C6)		✓	✓	916	5.1.1.2
2.4.2	Vergewissern Sie sich, dass das Salt mindestens 32 Bit lang ist, und wählen Sie es willkürlich, um Salzwertkollisionen zwischen gespeicherten Hashes zu minimieren. Für jedes Credential MUSS ein eindeutiger Salzwert und der daraus resultierende Hash gespeichert werden. (C6)		✓	✓	916	5.1.1.2
2.4.3	Vergewissern Sie sich, dass bei Verwendung von *PBKDF2* die Iterationszahl so groß sein SOLLTE, wie es die Leistung des Verifikationsservers zulässt, normalerweise mindestens 100.000 Iterationen. (C6)		✓	✓	916	5.1.1.2

2.4.4	Vergewissern Sie sich, dass bei Verwendung von *bcrypt* der Arbeitsfaktor so groß sein SOLLTE, wie es die Leistung des Verifikationsservers erlaubt, normalerweise mindestens 13. (C6)	✓	✓	916	5.1.1.2
2.4.5	Vergewissern Sie sich, dass eine zusätzliche Iteration einer Schlüsselableitung durchgeführt wird, wobei ein Salzwert verwendet wird, der geheim und nur dem Verifizierer bekannt ist. Generieren Sie den Salzwert mit einem zugelassenen Zufallsbitgenerator [*SP 800-90Ar1*] und stellen Sie mindestens die in der letzten Revision von *SP 800-131A* angegebene Mindestsicherheitsstärke bereit. Der geheime Saltwert MUSS getrennt von den gehashten Passwörtern gespeichert werden (z.B. in einem speziellen Gerät wie einem Hardware-Sicherheitsmodul).	✓	✓	916	5.1.1.2

Dort wo US-Normen erwähnt werden, kann je nach Bedarf eine regionale oder lokale Norm anstelle der US-Norm oder zusätzlich zu dieser verwendet werden.

V2.5 Anforderungen für die Wiederherstellung von Anmeldedaten

#	Beschreibung	L1	L2	L3	CWE	NIST §
2.5.1	Stellen Sie sicher, dass ein vom System generiertes Geheimnis zur Erstaktivierung oder Wiederherstellung nicht im Klartext an den Benutzer gesendet wird. (C6)	✓	✓	✓	640	5.1.1.2
2.5.2	Stellen Sie sicher, dass keine Hinweise auf Passwörter oder wissensbasierte Authentifizierung (so genannte *Sicherheitsfragen*) vorhanden sind.	✓	✓	✓	640	5.1.1.2
2.5.3	Stellen Sie sicher, dass die Wiederherstellung von Kennwortdaten das aktuelle Kennwort in keiner Weise Preis gibt. (C6)	✓	✓	✓	640	5.1.1.2
2.5.4	Überprüfen Sie, dass gemeinsame Konten oder Standardkonten nicht vorhanden sind (z.B. *root*, *admin* oder *sa*).	✓	✓	✓	16	5.1.1.2 / A.3
2.5.5	Vergewissern Sie sich, dass der Benutzer darüber informiert wird, wenn ein Authentifizierungsfaktor geändert oder ersetzt wird.	✓	✓	✓	304	6.1.2.3
2.5.6	Stellen Sie sicher, dass Sie für vergessene Kennwörter und andere Wiederherstellungspfade einen sicheren Wiederherstellungsmechanismus verwenden, wie z.B. TOTP oder andere Soft-Token, Mobile Push bzw. einen anderen Offline-Wiederherstellungsmechanismus. (C6)	✓	✓	✓	640	5.1.1.2
2.5.7	Vergewissern Sie sich, dass bei Verlust von OTP- oder Multi-Faktor-Authentifizierungsfaktoren der Identitätsnachweis auf der gleichen Ebene wie bei der Registrierung durchgeführt wird.		✓	✓	308	6.1.2.3

V2.6 Anforderungen für Nachschlagelisten von Sicherheitscodes

Geheimnisse aus Nachschlagewerken sind vorgenerierte Listen von Geheimcodes, ähnlich wie Transaktionsberechtigungsnummern (TAN), Wiederherstellungscodes für soziale Medien oder ein Raster, das eine Reihe von Zufallswerten enthält. Diese werden sicher an die Benutzer verteilt. Diese Nachschlagecodes werden einmal verwendet, und wenn alle verwendet werden, wird die Nachschlageliste verworfen. Diese Art von Authentifizierer wird als "etwas, das Sie haben" betrachtet.

#	Beschreibung	L1	L2	L3	CWE	NIST §

2.6.1	Stellen Sie sicher, dass Geheimnisse aus Nachschlagewerken nur einmal verwendet werden können.	✓	✓	308	5.1.2.2
2.6.2	Vergewissern Sie sich, dass die Geheimnisse aus Nachschlagewerken eine ausreichende Zufälligkeit aufweisen (112 Bit Entropie) oder, falls weniger als 112 Bit Entropie vorhanden sind, mit einem einzigartigen und zufälligen 32-Bit-Salz gesalzen und mit einem anerkannten Einweghash gehasht werden.	✓	✓	330	5.1.2.2
2.6.3	Vergewissern Sie sich, dass Geheimnisse aus Nachschlagewerken gegen Offline-Angriffe, wie z.B. vorhersehbare Werte, resistent sind.	✓	✓	310	5.1.2.2

V2.7 Anforderungen für Sicherheitsinformationen auf einem zweiten Kanal

In der Vergangenheit wäre eine übliche Prüfung auf einem zweiten Kanal eine E-Mail oder SMS mit einem Link zum Zurücksetzen des Passworts gewesen. Angreifer nutzen aber diesen schwachen Mechanismus, um Konten, die sie noch nicht kontrollieren, zurückzusetzen, z.B. indem sie das E-Mail-Konto einer Person übernehmen und entdeckte Reset-Links wieder verwenden. Es gibt bessere Möglichkeiten, die Verifizierung auf einem zweiten Kanal zu handhaben.

Sichere Sekundärkanal-Authentifikatoren sind physische Geräte, die mit dem Verifizierer über einen sicheren Sekundärkanal kommunizieren können. Beispiele hierfür sind Push-Benachrichtigungen an mobile Geräte. Diese Art von Authentifizierern wird als "etwas, das Sie haben" betrachtet. Wenn ein Benutzer sich authentifizieren möchte, sendet die verifizierende Anwendung eine Nachricht an den Sekundärkanal-Authentifikator über eine Verbindung direkt zum Authentifikator oder indirekt über einen Drittanbieterdienst. Die Nachricht enthält einen Authentifizierungscode (typischerweise eine zufällige sechsstellige Zahl oder einen modalen Genehmigungsdialog). Die verifizierende Anwendung wartet auf den Empfang des Authentifizierungscodes über den Primärkanal und vergleicht den Hash des empfangenen Wertes mit dem Hash des ursprünglichen Authentifizierungscodes. Wenn sie übereinstimmen, kann der Verifizierer des Sekundärkanals davon ausgehen, dass der Benutzer sich authentifiziert hat.

Der ASVS geht davon aus, dass nur wenige Entwickler neue Sekundärkanal-Authentifizierer wie Push-Benachrichtigungen entwickeln werden, und daher gelten die folgenden ASVS-Steuerelemente für Verifizierer, wie z.B. Authentifizierungs-API, Anwendungen und Single-Sign-On-Implementierungen. Wenn Sie einen neuen Sekundärkanal-Authentifikator entwickeln, lesen Sie bitte NIST 800-63B § 5.1.3.1.

Unsichere Sekundärkanal-Authentifizierer wie E-Mail und VOIP sind nicht zulässig. PSTN- und SMS-Authentifizierung sind derzeit durch NIST *eingeschränkt* und sollten zugunsten von Push-Benachrichtigungen oder ähnlichem eingestellt werden. Wenn Sie Telefon- oder SMS-Sekundärkanal-Authentifizierung verwenden müssen, lesen Sie bitte § 5.1.3.3.

#	Beschreibung	L1	L2	L3	CWE	NIST §
2.7.1	Vergewissern Sie sich, dass Klartext-Sekundärkanal-Authentifikatoren (NIST *restricted*), wie z.B. SMS oder PSTN, nicht standardmäßig angeboten werden, und dass zunächst stärkere Alternativen wie Push-Benachrichtigungen angeboten werden.					

| ✓ | ✓ | ✓ | 287 | 5.1.3.2 | | **2.7.2** | Vergewissern Sie sich, dass der Sekundärkanal-Überprüfer Authentifizierungsanforderungen, -Codes oder -Token nach 10 Minuten invalidiert. | ✓ | ✓ | ✓ | 287 | 5.1.3.2 | | **2.7.3** | Vergewissern Sie sich, dass Authentifizierungsanfragen, -Codes oder -Token des Sekundärkanal-

Überprüfers nur einmal und nur für die ursprüngliche Authentifizierungsanfrage verwendbar sind. | ✓ | ✓ | ✓ | 287 | 5.1.3.2 | | **2.7.4** | Überprüfen Sie, ob der Sekundärkanal-Authentifizierer und der Verifizierer über einen sicheren, unabhängigen Kanal kommunizieren. | ✓ | ✓ | ✓ | 523 | 5.1.3.2 | | **2.7.5** | Stellen Sie sicher, dass der Sekundärkanal-Überprüfer nur eine gehashte Version des Authentifizierungscodes aufbewahrt. | | ✓ | ✓ | 256 | 5.1.3.2 | | **2.7.6** | Vergewissern Sie sich, dass der anfängliche Authentifizierungscode von einem sicheren Zufallszahlengenerator erzeugt wird, der mindestens 20 Bit Entropie enthält (normalerweise ist eine sechsfache digitale Zufallszahl ausreichend).

| | ✓ | ✓ | 310 | 5.1.3.2 |

V2.8 Anforderungen an eine Ein- oder Mehrfaktor-Prüfung

Einmalpasswörter mit einem Faktor (OTP) sind physische oder Soft Token, die eine sich ständig ändernde pseudo-zufällige *one time challenge* darstellen. Diese Geräte machen Phishing (Nachahmung) schwierig, aber nicht unmöglich. Diese Art von Authentifikatoren wird als "etwas, das Sie haben" betrachtet. Multi-Faktor-Token sind ähnlich wie Ein-Faktor-OTPs, erfordern jedoch einen gültigen PIN-Code, biometrische Entsperrung, USB-Anschluss oder NFC-Paarung oder einen zusätzlichen Wert (z.B. einen Transaktionssignierungsrechner), der eingegeben werden muss, um den endgültigen OTP zu erstellen.

#	Description	L1	L2	L3	CWE	NIST §
2.8.1	Überprüfen Sie, ob zeitbasierte OTPs eine definierte Lebensdauer haben, bevor sie ablaufen.	✓	✓	✓	613	5.1.4.2 / 5.1.5.2
2.8.2	Vergewissern Sie sich, dass die symmetrischen Schlüssel, die zur Überprüfung der eingereichten OTPs verwendet werden, mit einem hohen Schutzlevel folgen, z. B. durch die Verwendung eines Hardware-Sicherheitsmoduls oder einer sicheren betriebssystembasierten Schlüsselspeicherung.		✓	✓	320	5.1.4.2 / 5.1.5.2
2.8.3	Verifizieren Sie, dass anerkannte kryptografische Algorithmen bei der Generierung, dem Seeding und der Prüfung verwendet werden.		✓	✓	326	5.1.4.2 / 5.1.5.2
2.8.4	Stellen Sie sicher, dass das zeitbasierte OTPs nur einmal innerhalb des Gültigkeitszeitraums verwendet werden können.		✓	✓	287	5.1.4.2 / 5.1.5.2
2.8.5	Vergewissern Sie sich, dass ein zeitbasiertes Mehrfaktor-OTP-Token, das während der Gültigkeitsdauer wiederverwendet wird, protokolliert und abgelehnt wird, und Sicherheitsbenachrichtigungen an den Inhaber des Geräts gesendet werden.		✓	✓	287	5.1.5.2
2.8.6	Vergewissern Sie sich, dass der physische Einfaktor-OTP-Generator im Falle eines Diebstahls oder anderweitigem Verlust widerrufen werden kann. Stellen Sie sicher, dass der Widerruf sofort für alle eingeloggten Sitzungen, unabhängig vom Standort, wirksam ist.		✓	✓	613	5.2.1
2.8.7	Vergewissern Sie sich, dass biometrische Authentifikatoren nur als sekundäre Faktoren in Verbindung mit *etwas, das Sie haben* und *etwas, das Sie kennen*, verwendet werden dürfen.					
	o	✓	308	5.2.3		

V2.9 Anforderungen an Überprüfer für kryptografische Software und Geräte

Kryptographische Sicherheitsschlüssel sind Smartcards oder FIDO-Schlüssel, bei denen der Benutzer das kryptographische Gerät an den Computer anschließen oder mit ihm koppeln muss, um die Authentifizierung abzuschließen. Überprüfer senden eine *Challenge Nonce* an die kryptographischen Geräte oder Software, und das Gerät oder die Software berechnet eine Antwort auf der Grundlage eines sicher gespeicherten kryptographischen Schlüssels.

Die Anforderungen für kryptographische Geräte und Software mit einem Faktor und für kryptographische Geräte und Software mit mehreren Faktoren sind dieselben, da die Prüfung des kryptographischen Authentifizierers den Besitz des Authentifizierungsfaktors nachweist.

#	Beschreibung	L1	L2	L3	CWE	NIST §
2.9.1	Vergewissern Sie sich, dass die bei der Prüfung verwendeten kryptografischen Schlüssel sicher gespeichert und gegen Offenlegung geschützt sind, z.B. durch Verwendung eines TPM, HSM oder eines Betriebssystemdienstes, der diesen sicheren Speicherplatz verwenden kann.		✓	✓	320	5.1.7.2
2.9.2	Vergewissern Sie sich, dass die *Challenge Nonce* mindestens 64 Bit lang ist und statistisch eindeutig oder über die Lebensdauer des kryptografischen Geräts eindeutig ist.					
	✓	✓	330	5.1.7.2		
2.9.3	Vergewissern Sie sich, dass anerkannte kryptografische Algorithmen bei der Generierung, dem Seeding und der Prüfung verwendet werden.		✓	✓	327	5.1.7.2

V2.10 Anforderungen an die Authentifizierungsdienst

Dieser Abschnitt ist nicht penetrationstestbar, hat also keine Anforderungen der Stufe 1. Wenn Sie jedoch in einer Architektur-, Codierungs- oder Sicherheitscodeprüfung verwendet werden, gehen Sie bitte davon aus, dass Software (beispielsweise der Java Key Store) die Mindestanforderung an die Stufe 1 ist. Die Speicherung von Geheimnissen im Klartext ist unter keinen Umständen akzeptabel.

#	Beschreibung	L1	L2	L3	CWE	NIST §
2.10.1	Stellen Sie sicher, dass die integrierte Sicherheitsdaten nicht auf unveränderlichen Passwörtern wie API-Schlüsseln oder gemeinsam genutzten privilegierten Konten beruhen.		Bertriebssystemunterstützt	HSM	287	5.1.1.1
2.10.2	Vergewissern Sie sich, dass die Zugangsdaten keinem Standardkonto entsprechen, wenn Passwörter erforderlich sind.		Bertriebssystemunterstützt	HSM	255	5.1.1.1
2.10.3	Stellen Sie sicher, dass die Passwörter mit ausreichendem Schutz gespeichert werden, um Offline-Wiederherstellungsangriffe, einschließlich des lokalen Systemzugriffs, zu verhindern.		Bertriebssystemunterstützt	HSM	522	5.1.1.1
2.10.4	Vergewissern Sie sich, dass Passwörter, Integrationen mit Datenbanken und Systemen von Drittanbietern, Seeds					

	und interne Geheimnisse sowie API-Schlüssel sicher verwaltet werden und nicht im Quellcode enthalten sind oder in Quellcode-Repositories gespeichert werden. Eine solche Speicherung SOLLTE Offline-Angriffen widerstehen. Die Verwendung eines sicheren Softwareschlüsselspeichers (Stufe 1), eines Hardware Trusted Platform Module (TPM) oder eines Hardware-Sicherheitsmoduls (Stufe 3) wird für die Speicherung von Passwörtern empfohlen.	
Bertriebssystemunterstützt	HSM	798

Zusätzliche Anforderungen der US-Behörden

US-Behörden haben verbindliche Anforderungen bezüglich NIST 800-63. Der Standard für die Verifizierung der Anwendungssicherheit entsprach schon immer etwa 80% der Kontrollen, die für fast 100% der Anwendungen gelten, und nicht den letzten 20% der fortgeschrittenen Kontrollen oder solche, die nur begrenzt anwendbar sind. Der ASVS ist somit eine genaue Untermenge von NIST 800-63, insbesondere für die IAL1/2- und AAL1/2-Klassifizierungen, aber er ist nicht umfassend genug, insbesondere was die IAL3/AAL3-Klassifizierungen betrifft.

Wir fordern die US-Regierungsbehörden nachdrücklich auf, NIST 800-63 in seiner Gesamtheit zu überprüfen und umzusetzen.

Glossar der Fachbegriffe

Fachbegriff	Bedeutung
CSP	*Credential Service Provider* auch als Identity Provider bekannt
Authentifikatoren	Code, der ein Kennwort, ein Token, MFA, eine föderierte Behauptung, usw. authentifiziert.
Verifier	Eine Einrichtung, die die Identität des Antragstellers durch die Überprüfung des Besitzes und der Kontrolle des Antragstellers über einen oder zwei Authentifizierer mittels eines Authentifizierungsprotokolls verifiziert. Zu diesem Zweck muss der Verifier möglicherweise auch die Anmeldedaten validieren, die den/die Authentifikator(en) mit der Kennung des Abonnenten verbinden und ihren Status überprüfen
OTP	One-time password
SFA	Single-Factor-Authentifikatoren, z.B. etwas, das Sie kennen (gespeicherte Geheimnisse, Passwörter, Passphrasen, PINs), etwas, das Sie sind (Biometrie, Fingerabdrücke, Gesichtsscans) oder etwas, das Sie haben (OTP-Token, ein kryptographisches Gerät wie eine Smartcard),
MFA	Multifaktor-Authentifikator, der zwei oder mehr Einzelfaktoren umfasst

Verweise

Für weitere Informationen siehe auch:

- NIST 800-63 - Digital Identity Guidelines
- NIST 800-63 A - Enrollment and Identity Proofing
- NIST 800-63 B - Authentication and Lifecycle Management

- NIST 800-63 C - Federation and Assertions
- NIST 800-63 FAQ
- OWASP Testing Guide 4.0: Testing for Authentication
- OWASP Cheat Sheet - Password storage
- OWASP Cheat Sheet - Forgot password
- OWASP Cheat Sheet - Choosing and using security questions

V3: Anforderungen an die Sitzungsmanagementprüfung

Zielsetzung

Eine der Kernkomponenten jeder webbasierten Anwendung oder zustandsabhängigen API ist ein Mechanismus, mit dessen Hilfe sie den Zustand für einen Benutzer oder ein Gerät, das mit ihr interagiert, steuert und aufrechterhält. Die Sitzungsverwaltung transformiert ein zustandsloses Protokoll in ein zustandsbehaftetes, was für die Unterscheidung verschiedener Benutzer oder Geräte entscheidend ist.

Es muss sichergestellt werden, dass eine geprüfte Anwendung die folgenden High-Level Anforderungen an das Sitzungsmanagement erfüllt:

- Sitzungen sind für jede Person einzigartig und können nicht erraten oder geteilt werden.
- Sitzungen werden als ungültig gekennzeichnet, wenn sie nicht mehr benötigt werden, und werden während inaktiver Phasen zeitweise deaktiviert.

Wie bereits erwähnt, wurden diese Anforderungen so angepasst, dass sie eine konforme Teilmenge ausgewählter NIST 800-63b-Kontrollen darstellen, die sich auf gemeinsame Bedrohungen und häufig ausgenutzte Authentifizierungsschwächen konzentrieren. Frühere Prüfungsanforderungen wurden zurückgezogen, abgeschafft oder in vielen Fällen so angepasst, dass sie sich stark an den Zielen der obligatorischen Anforderungen NIST 800-63b orientieren.

Anforderungen an die Sicherheitsüberprüfung

V3.1 Grundlegende Anforderungen an das Sitzungsmanagement

#	Beschreibung	L1	L2	L3	CWE	NIST §
3.1.1	Vergewissern Sie sich, dass die Anwendung niemals Session-Tokens in URL-Parametern oder Fehlermeldungen offenlegt.	✓	✓	✓	598	

V3.2 Anforderungen an die Sitzungsbindung

#	Beschreibung	L1	L2	L3	CWE	NIST §
3.2.1	Überprüfen Sie, ob die Anwendung bei jeder Benutzerauthentifizierung immer ein neues Sitzungs-Token erzeugt. (C6)	✓	✓	✓	384	7.1
3.2.2	Überprüfen Sie, ob Sitzungs-Token mindestens 64 Bit Entropie aufweisen. (C6)	✓	✓	✓	331	7.1
3.2.3	Vergewissern Sie sich, dass die Anwendung Sitzungs-Token im Browser nur mit sicheren Methoden wie z.B. entsprechend gesicherten Cookies (siehe Abschnitt 3.4) oder mit Hilfe von HTML 5 Session Storage speichert.	✓	✓	✓	539	7.1
3.2.4	Vergewissern Sie sich, dass die Sitzungs-Token mit anerkannten		✓	✓	331	7.1

kryptografischen Algorithmen generiert werden. (C6)

TLS oder ein anderer entsprechend sicherer Transportkanal ist für die Sitzungsverwaltung obligatorisch. Dies wird im Kapitel Kommunikationssicherheit behandelt.

V3.3 Anforderungen für Sitzungsabmeldung und -timeouts

Die Session-Timeouts wurden an NIST 800-63 angeglichen, der wesentlich längere Session-Timeouts erlaubt, als die Sicherheitsstandards traditionell erlauben. Unternehmen sollten die nachstehende Tabelle überprüfen, und wenn ein längeres Timeout aufgrund des Risikos der Anwendung wünschenswert ist, sollte der NIST-Wert die obere Grenze der Session-Timeouts im Leerlauf sein.

Stufe 1 entspricht in diesem Zusammenhang IAL1/AAL1, Stufe 2 entspricht IAL2/AAL3, Stufe 3 entspricht IAL3/AAL3. Für IAL2/AAL2 und IAL3/AAL3 ist das kürzere Timeout die untere Grenze der Leerlaufzeit für die Abmeldung oder erneute Authentifizierung zur Wiederaufnahme der Sitzung.

#	Beschreibung	L1	L2	L3	CWE	NIST §
3.3.1	Vergewissern Sie sich, dass die Abmeldung und der Ablauf der Sitzung das Sitzungs-Token ungültig machen, so dass der Zurück-Button oder eine sonstige nachgeschaltete vertrauenswürdige Partei eine authentifizierte Sitzung nicht wieder aufnimmt, auch nicht zwischen den Vertrauensparteien. (C6)	✓	✓	✓	613	7.1
3.3.2	Wenn Authentifizierer den Benutzern erlauben, angemeldet zu bleiben, überprüfen Sie, dass eine erneute Authentifizierung in regelmäßigen Abständen sowohl bei aktiver Nutzung als auch nach einer Leerlaufphase erfolgt. (C6)	30 Tage	12 Stunden oder 30 Minuten der Inaktivität, 2FA Optional	12 Stunden oder 15 Minuten der Inaktivität, mit 2FA	613	7.2
3.3.3	Vergewissern Sie sich, dass die Anwendung alle anderen aktiven Sitzungen nach einer erfolgreichen Kennwortänderung beendet, und dass dies in der gesamten Anwendung, der föderierten Anmeldung (falls vorhanden) und in allen sich darauf vertrauenden Parteien wirksam ist.		✓	✓	613	
3.3.4	Überprüfen Sie, ob die Benutzer in der Lage sind, vereinzelte oder alle derzeit aktiven Sitzungen und Geräte anzuzeigen und sich davon abzumelden.		✓	✓	613	7.1

V3.4 Cookie-basierte Sitzungsverwaltung

#	Beschreibung	L1	L2	L3	CWE	NIST §
3.4.1	Vergewissern Sie sich, dass Cookie-basierte Session-Tokens das Attribut *Secure* gesetzt haben. (C6)	✓	✓	✓	614	7.1.1
3.4.2	Überprüfen Sie, dass Cookie-basierte Session-Tokens das Attribut *HttpOnly* gesetzt haben. (C6)	✓	✓	✓	1004	7.1.1

3.4.3	Verifizieren Sie, dass Cookie-basierte Session-Tokens das *SameSite* Attribut verwenden, um die Anfälligkeit für Cross-Site Request Forgery (CSRF) Angriffe zu begrenzen. (C6)	✓	✓	✓	16	7.1.1
3.4.4	Stellen Sie sicher, dass Cookie-basierte Session-Tokens den Präfix *__Host-* verwenden (siehe Referenzen), um die Vertraulichkeit von Session-Cookies zu gewährleisten.	✓	✓	✓	16	7.1.1
3.4.5	Wenn die Anwendung unter einem gemeinsamen Domänennamen zusammen mit anderen Anwendungen veröffentlicht wird, die Sitzungscookies setzen oder verwenden, welche die Sitzungscookies außer Kraft setzen oder offenlegen könnten, stellen Sie das Pfadattribut in Cookie-basierten Sitzungs-Tokens unter Verwendung des konkretesten Pfades ein. (C6)	✓	✓	✓	16	7.1.1

V3.5 Token-basierte Sitzungsverwaltung

Die tokenbasierte Sitzungsverwaltung umfasst JWT-, OAuth-, SAML- und API-Schlüssel. Von diesen sind die API-Schlüssel bekanntermaßen schwach und sollten in neu implementiertem Code nicht verwendet werden.

#	Beschreibung	L1	L2	L3	CWE	NIST §
3.5.1	Vergewissern Sie sich, dass die Anwendung OAuth und Refresh Tokens nicht — selbst bearbeitet — als die Anwesenheit eines Teilnehmers auffasst und es so den Benutzern ermöglicht, Vertrauensbeziehungen mit verknüpften Anwendungen zu beenden.		✓	✓	290	7.1.2
3.5.2	Überprüfen Sie, dass die Anwendung Sitzungs-Tokens anstelle von statischen API-Geheimnissen und -Schlüsseln verwendet, außer bei Legacy-Implementierungen.		✓	✓	798	
3.5.3	Verifizieren Sie, dass zustandslose Session-Tokens digitale Signaturen, Verschlüsselung und andere Gegenmaßnahmen zum Schutz gegen Angriffe wie Manipulation, Verschleierung, Replay, Null-Chiffren und Schlüsselaustausch verwenden.		✓	✓	345	

V3.6 Erneute Authentifizierung einer Föderation oder anderweitigen Erklärung

Dieser Abschnitt bezieht sich auf diejenigen, die den Code für die vertrauenswürdigen Partei (*Relying Party*, RP) oder den Code für den *Credential Service Provider* (CSP) schreiben. Wenn Sie sich auf Code verlassen, der diese Funktionen implementiert, stellen Sie sicher, dass die nachfolgenden Probleme korrekt behandelt werden.

#	Description	L1	L2	L3	CWE	NIST §
3.6.1	Vergewissern Sie sich, dass die vertrauenswürdigen Parteien die maximale Authentifizierungszeit gegenüber den *Credential Service Providers* (CSPs) angeben und dass die CSPs den Teilnehmer erneut authentifizieren, wenn sie innerhalb dieses Zeitraums keine Sitzung genutzt haben.			✓	613	7.2.1
3.6.2	Vergewissern Sie sich, dass die *Credential Service Provider* (CSPs) die vertrauenswürdigen Parteien (*Relying Party*, RP) über das letzte Authentifizierungsereignis informieren, damit die RPs feststellen können, ob sie den Benutzer erneut authentifizieren müssen.			✓	613	7.2.1

V3.7 Schutz vor Ausnutzung von Schwächen des Sitzungsmanagements

Es gibt eine kleine Anzahl von Angriffen auf das Sitzungsmanagement, von denen einige mit der User Experience (UX) von Sitzungen zusammenhängen. Zuvor hat der ASVS auf der Grundlage der Anforderungen der ISO 27002 das Blockieren mehrerer gleichzeitiger Sitzungen gefordert. Das Blockieren gleichzeitiger Sitzungen ist nicht mehr angemessen, nicht nur, weil heutige Benutzer viele Geräte haben oder die Anwendung eine API ohne Browser-Sitzung ist. In den meisten dieser Implementierungen gewinnt der letzte Authentifizierer, der oft der Angreifer ist. Dieser Abschnitt bietet eine weiterführende Anleitung zum Abhalten, Verzögern und Erkennen von Angriffen auf das Sitzungsmanagement mittels Programmcode.

Half-Open Angriffe

Anfang 2018 wurden mehrere Finanzinstitutionen durch so genannte *Half-Open* Angriffe kompromittiert. Dieser Begriff ist in der Branche hängen geblieben. Die Angreifer schlugen bei mehreren Institutes mit unterschiedlichen proprietären Code-Basen zu, und in der Tat scheint es verschiedene Code-Basen innerhalb derselben Institute zu geben. Der *Half-Open* Angriff nutzt einen Designfehler aus, der in vielen bestehenden Authentifizierungs-, Sitzungsverwaltungs- und Zugangskontrollsystemen zu finden ist.

Die Angreifer beginnen einen *Half-Open* Angriff, indem sie versuchen, ein Anmeldedatum zu sperren, zurückzusetzen oder wiederherzustellen. Ein beliebtes Design-Pattern für die Sitzungsverwaltung setzt auf die Wiederverwendung von Sitzungsobjekten und -modellen des Benutzerprofils zwischen nicht authentifiziertem, halb-authentifiziertem (Kennwortrücksetzung, vergessener Benutzername) und vollständig authentifiziertem Code. Dieses Entwurfsmuster befüllt ein gültiges Sitzungsobjekt oder Token mit dem Profil des Opfers, einschließlich Passwort-Hashes und Rollen. Wenn die Zugriffskontrollprüfungen in Controllern oder Routern nicht korrekt überprüfen, ob ein Benutzer vollständig angemeldet ist, kann der Angreifer im Namen dieses Benutzers handeln.
Angriffe könnten u.a. die Änderung des Benutzerkennworts auf einen bekannten Wert, die Aktualisierung der E-Mail-Adresse zur Durchführung einer gültigen Kennwortrücksetzung, die Deaktivierung der Multi-Faktor-Authentifizierung oder die Registrierung eines neuen MFA-Geräts oder die Offenlegung bzw. Änderung von API-Schlüsseln umfassen.

#	Beschreibung	L1	L2	L3	CWE	NIST §
3.7.1	Vergewissern Sie sich, dass die Anwendung eine gültige Anmeldesitzung gewährleistet oder eine erneute Authentifizierung oder sekundäre Überprüfung erfordert, bevor sensible Transaktionen oder Kontoänderungen zugelassen werden.	✓	✓	✓	778	

Verweise

Für weitere Informationen siehe auch:

- OWASP Testing Guide 4.0: Session Management Testing
- OWASP Session Management Cheat Sheet
- Set-Cookie Host- prefix details

V4: Anforderungen für die Überprüfung der Zugriffskontrolle

Zielsetzung

Autorisierung ist das Konzept, den Zugang zu den Ressourcen nur denjenigen zu gestatten, die diese auch verwenden dürfen. Stellen Sie sicher, dass eine überprüfte Anwendung die folgenden High-Level Anforderungen erfüllt:

- Personen, die auf Ressourcen zugreifen, verfügen über gültige Anmeldedaten hierfür.
- Benutzer sind mit einem klar definierten Satz von Rollen und Privilegien verknüpft.
- Metadaten für Rollen und Berechtigungen sind vor Replay-Attacken oder Manipulation geschützt.

Anforderungen an die Sicherheitsüberprüfung

V4.1 Allgemeine Entwurfsrichtlinien für die Zugriffskontrolle

#	Beschreibung	L1	L2	L3	CWE
4.1.1	Vergewissern Sie sich, dass die Anwendung Zugriffskontrollregeln auf einer vertrauenswürdigen Dienstschicht durchsetzt, insbesondere wenn die Zugriffskontrolle auf der Client-Seite vorhanden ist und umgangen werden könnte.	✓	✓	✓	602
4.1.2	Vergewissern Sie sich, dass alle Benutzer- und Datenattribute und Richtlinieninformationen, die von den Zugriffskontrollen verwendet werden, von den Endbenutzern nicht manipuliert werden können, es sei denn, dies wird ausdrücklich genehmigt.	✓	✓	✓	639
4.1.3	Überprüfen Sie, ob das Prinzip der geringstmöglichen Privilegien (*least privilege*) angewandt wird - Benutzer sollten nur auf Funktionen, Dateien, URLs, Controller, Dienste und andere Ressourcen zugreifen können, für die sie eine bestimmte Berechtigung besitzen. Dies impliziert einen Schutz gegen Spoofing und gegen eine Erhöhung der Privilegien. (C7)	✓	✓	✓	285
4.1.4	Vergewissern Sie sich, dass das Prinzip der standardmäßigen Verweigerung (*deny by default*) besteht, wonach neue Benutzer/Rollen mit minimalen oder gar keinen Berechtigungen starten und Benutzer/Rollen keinen Zugang zu neuen Funktionen erhalten, bis der Zugang explizit zugewiesen wird. (C7)	✓	✓	✓	276
4.1.5	Vergewissern Sie sich, dass die Zugriffskontrollen Anfragen entsprechend sicher ablehnen, auch wenn eine Ausnahme auftritt. (C10)	✓	✓	✓	285

V4.2 Zugriffskontrolle auf Betriebsebene

#	Beschreibung	L1	L2	L3	CWE
4.2.1	Vergewissern Sie sich, dass sensible Daten und APIs vor direkten Objektangriffen geschützt sind, die auf das Erstellen, Lesen, Aktualisieren und Löschen von Datensätzen abzielen, z.B. das Erstellen oder Aktualisieren von Datensätzen einer anderen Person, das Anzeigen aller Datensätze oder das Löschen aller Datensätze.	✓	✓	✓	639
4.2.2	Vergewissern Sie sich, dass die Anwendung oder das verwendete Framework einen starken Anti-CSRF-Mechanismus zum Schutz authentifizierter Funktionalität durchsetzt und dass eine effektive Anti-Automatisierung oder Anti-CSRF nicht authentifizierte Funktionalität schützt.	✓	✓	✓	352

V4.3 Weitere Überlegungen zur Zugangskontrolle

#	Beschreibung	L1	L2	L3	CWE
4.3.1	Überprüfen Sie, ob die administrativen Schnittstellen eine geeignete Multi-Faktor-Authentifizierung verwenden, um eine unbefugte Nutzung zu verhindern.	✓	✓	✓	419
4.3.2	Vergewissern Sie sich, dass die Navigation in Verzeichnissen deaktiviert ist, es sei denn, dies ist absichtlich erwünscht. Außerdem sollten Anwendungen das Auffinden oder die Offenlegung von Datei- oder Verzeichnis-Metadaten, wie z.B. *Thumbs.db*, *.DS_Store*, *.git* oder *.svn* Ordner, nicht zulassen.	✓	✓	✓	548
4.3.3	Vergewissern Sie sich, dass die Anwendung über zusätzliche Berechtigungen (wie z.B. ansteigende oder adaptive Authentifizierung) für Systeme mit geringerem		✓	✓	732

Wert und/oder eine Aufgabentrennung für kritische Anwendungen mit hohem Wert verfügt, um Betrugsbekämpfungskontrollen entsprechend dem Risiko der Anwendung und früherer Betrugsfälle durchzusetzen.

Verweise

Für weitere Informationen siehe auch:

- OWASP Testing Guide 4.0: Authorization
- OWASP Cheat Sheet: Access Control
- OWASP CSRF Cheat Sheet
- OWASP REST Cheat Sheet