Bundesamt
für Sicherheit in der
Informationstechnik

Leitfaden zur Entwicklung sicherer Webanwendungen

Empfehlungen und Anforderungen an die Auftragnehmer

SEC Consult Deutschland Unternehmensberatung GmbH
Bockenheimer Landstraße 17/19
60325 Frankfurt/Main
www.sec-consult.com

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
Tel.: +49 22899 9582-0
E-Mail: bsi@bsi.bund.de
Internet: https://www.bsi.bund.de

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Webanwendungen stellen in IT-Infrastrukturen meist die exponiertesten IT-Systeme eines Unternehmens oder einer Behörde dar. Dies wurde auch durch den UK Security Breach Investigations Report 2010[1] festgestellt. Demnach sind 86% der Angriffe auf Unternehmen im Vereinigten Königreich auf Schwachstellen in Webanwendungen zurückzuführen. Für Deutschland liegen entsprechende Daten nicht vor, es ist jedoch davon auszugehen, dass sich diese in einem ähnlichen Bereich befinden.

Immer mehr, teilweise hochsensible Daten und Dienste werden heutzutage über das Internet verfügbar gemacht. Nicht selten geschieht dies durch selbst entwickelte Individualsoftware, bei deren Entwicklung Sicherheit kaum eine Berücksichtigung gefunden hat. Auch deshalb lässt sich das Sicherheitsniveau bei solchen Individualanwendungen gewöhnlich um ein Vielfaches schwerer abschätzen, als dies etwa bei Standardanwendungen mit einem großen Nutzerkreis der Fall ist.

Historisch betrachtet hat der starke Anstieg des Bedrohungspotentials von Webapplikationen auch mit der rasanten Entwicklung der zugrundeliegenden Technologien zu tun. Den Grundstein dafür legten die ersten verlinkten Webseiten (sog. Web 1.0) im Jahre 1993, womit das Internet für ein Massenpublikum eröffnet wurde. In Anbetracht der rein statischen Inhalte dieser neuen Technologie war das damalige Gefährdungspotential für Unternehmen sehr gering. Dies änderte sich nach der Jahrtausendwende durch die Entwicklung von Webseiten, bei denen der Benutzer im Zentrum steht, dem sogenannten Web 2.0. Ab diesem Zeitpunkt ist der Benutzer dieser Infrastruktur kein reiner Konsument mehr, das Einbringen von neuen Inhalten in die Webseiten und damit in die Unternehmensinfrastruktur ist beabsichtigt. Als logische Konsequenz daraus setzen unsichere Webanwendungen das Unternehmen großen Gefahren aus, auf die es zu reagieren gilt.

Diesen Umstand gilt es durch Security Policies dahingehend zu behandeln, dass die eingesetzten Anwendungen einem Sicherheitsniveau entsprechen, um das stetige IT-Risiko auf ein akzeptables Maß zu reduzieren. Hier spielen auch Kostengesichtspunkte eine wichtige Rolle. So hat das National Institute of Standards and Technology (NIST) bereits im Jahre 2002 in einer Studie über die ökonomischen Auswirkungen von fehlerhafter Software[2] festgestellt, dass das Beheben eines Softwarefehlers nach der Produktveröffentlichung um den Faktor 30mal teurer ist als in der Designphase. Selbst ohne Berücksichtigung des Risikos einer Systemkompromittierung durch einen Angreifer hat die frühzeitige Beseitigung von Sicherheitslücken in einer Applikation damit klare ökonomische Vorteile. Daher werden durch diesen Leitfaden Sicherheitsanforderungen bereits frühzeitig an den Entwicklungsprozess gestellt.

1.2 Ziele und Anwendungsgebiete

Dieser Leitfaden richtet sich an Projekt-, Fach- und Sicherheitsverantwortlichen sowie an potenzielle Auftragnehmer, die eigene oder in Auftrag gegebene Webanwendungen entwickeln. Er ist als Orientierungshilfe gedacht und hat keinen bindenden Charakter.

Folgende Ziele stehen im Vordergrund:

- **Die Vermeidung sowie frühzeitige Beseitigung von Sicherheitslücken** in Webanwendungen ist eines der zentralen Ziele dieses Leitfadens. Dieses kann nicht zuletzt auch durch das Zusammenspiel zwischen Fachabteilung und Entwicklung mithilfe der Anwendung von etablierten Designprinzipien und der Durchführung einer Bedrohungsanalyse maßgeblich beeinflusst werden. Zusätzlich hat eine frühzeitige Entdeckung von Sicherheitslücken ökonomische Vorteile. Die

1 [7saf2010]
2 [Nist 2002]

Kosten für die Beseitigung von Sicherheitslücken in einer frühen Phase stellen einen Bruchteil von jenen dar, die bei der Korrektur einer Sicherheitslücke in der Produktion anfallen.

- **Die bessere Planbarkeit von Sicherheitsmaßnahmen** wird durch die Berücksichtigung von Sicherheitsmaßnahmen (inkl. Sicherheitstests) bereits innerhalb der Planungs- bzw. Ausschreibungsphase erreicht. Dies hilft dabei auch erforderliche Ressourcen frühzeitig projektieren zu können. Zahlreiche Softwareprojekte haben gezeigt, dass eine fehlende Sicherheitsplanung im Projektmanagement die Ausgaben für Sicherheit in der Wartungsphase stark erhöht. Dabei wird oft versucht das Produkt im Nachhinein abzusichern, was vor allem bei Fehlern im Anwendungsdesign jedoch nur sehr schwer möglich ist.
- **Die Erhöhung des Sicherheitsniveaus** sollte sowohl das Ziel des Auftraggebers als auch des Auftragnehmers darstellen. Dabei ist zu beachten, dass sich eine „perfekte" Sicherheit selten erreichen lässt und sich diese ebenso wenig nachweisen lässt. Vielmehr ist das Ziel, ein erforderliches Sicherheitsniveau abhängig vom Schutzbedarf der Anwendung zu bestimmen und dieses zu verfolgen. Dieser Leitfaden gibt dem Auftragnehmer die Möglichkeit, geeignete Sicherheitsanforderungen aus einem definierten Schutzbedarf abzuleiten und damit das Sicherheitsniveau in einer angemessenen Höhe zu gewährleisten.
- **Eine allgemeine Verbesserung der Softwarequalität** wird auch durch gezielte Einbeziehungen und Schulung der an der Erstellung der Anwendung beteiligten Personen erreicht.[3]
- **Die nachhaltige Vermeidung von Sicherheitslücken** wird durch regelmäßige Sicherheitsüberprüfungen der Anwendung (etwa mittels Penetrationstests), Vorgaben an die Infrastruktur für einen sicheren Betrieb und der regelmäßigen Wartung der Infrastruktur durch Patches und Updates gefördert.

1.3 Darstellung des gewählten Ansatzes

Dieser Leitfaden beschreibt notwendige Aktivitäten eines Secure Development Lifecycle (SDLs). Darunter sind Sicherheitsmaßnahmen zu verstehen, die im Rahmen des Softwareentwicklungsprozesses einer Anwendung umgesetzt werden können, um deren Sicherheit laufend zu prüfen und nachhaltig zu verbessern. Der Fokus wird in diesem Dokument auf die Entwicklung von Webanwendungen gelegt. Aufgrund der generischen, organisatorischen und technischen Maßnahmen kann es auch für die Entwicklung anderer Applikationsarten verwendet werden.

Der zentrale Unterschied dieses Leitfadens zu existierenden Best-Practices wie BSIMM oder Open SAMM, auf welchen dieser Leitfaden in Teilen basiert, ist durch das in Kapitel 4.1 dargestellte Reifegradmodell gegeben. Ähnlich wie beim IT-Grundschutz wird hierbei eine Reihe notwendiger Aktivitäten definiert, durch die sich ein bestimmter Basisschutz für den Entwicklungsprozess vorgeben lässt.

Jeder Software-Entwicklungsprozess folgt dem in diesem Dokument beschriebenen Modell bis zu einem bestimmten Grad. Das klassische Wasserfallmodel folgt strikt diesen definierten Phasen, die agilen Entwicklungsmodellen dagegen wiederholen einen Schritt so oft es nötig ist, bis das gewünschte Ergebnis erzielt ist. Für die Ziele dieses Dokuments spielt das gewählte Modell keine signifikante Rolle, da jedes Unternehmen auf unterschiedliche Art und Weise Software entwickelt. Es ist wichtig, um erfolg zu haben, dass die in diesem Dokument beschriebenen Aktivitäten pragmatisch und ergebnisorientiert an die eigenen Gegebenheiten angepasst werden.

3 Die WhiteHat Website Security Statistics Report 2013 ([White_2013]) kommt zu einem sehr anschaulichen Ergebnis: Organisationen, die in Schulung und Sensibilisierung zum Thema IT-Sicherheit investieren, reduzieren die Anzahl der Schwachstellen um 40%, beseitigen diese 59% schneller und müssen nur bei 14% der Fälle nachbessern.

2 Begriffe, Ziele und Konzepte

2.1 Webanwendungssicherheit

Als Teildisziplin der IT-Sicherheit befasst sich die Webanwendungssicherheit im Wesentlichen mit dem Schutz von Assets[4] einer Webanwendung. Unter Assets versteht man in diesem Fall all jene Bestandteile (Daten, Systeme und Funktionen) einer Webanwendung, für die sich ein Schutzbedarf hinsichtlich eines der drei primären Schutzziele

- Vertraulichkeit,
- Integrität sowie
- Verfügbarkeit

ableiten lässt.

Darüber hinaus leiten sich aus diesen primären Schutzzielen auch sekundäre Schutzziele ab, zu denen etwa Authentizität oder Nicht-Abstreitbarkeit zählen.

Die Disziplin der Webanwendungssicherheit stellt sich dabei als Querschnittthema dar, welches sich auf alle Phasen des Lebenszyklus einer Webanwendung bezieht und auch einen Bestandteil der Qualitätssicherung darstellt. Sie befasst sich sowohl mit der Abwehr von Bedrohungen und der Prävention von Schwachstellen als auch mit der Identifikation und der Behebung von Schwachstellen.

2.2 Software Assurance (SwA)

Wie in allen Bereichen der Sicherheit kann es auch für Software keine hundertprozentige Sicherheit geben. Ziel ist es stattdessen, einen bestimmten Grad an Vertrauen in die Sicherheit von Software zu ermitteln. Vertrauen ist dabei dahin gehend zu verstehen, dass die Software, soweit wie möglich und umsetzbar, frei von Sicherheitslücken ist und so funktioniert, wie diese spezifiziert wurde. Dies bezeichnet man als Software Assurance (SwA).

Das Entwerfen und Implementieren von spezifischen Sicherheitseigenschaften unter Wahrung der definierten Schutzziele ist für die Entwicklung sicherer Software dabei unabdingbar. Daher ist ein Verständnis für die durch eine Anwendung verarbeiteten Assets und deren Schutzbedarf essentiell. Denn genau aus diesem leitet sich die für eine Webanwendung erforderlichen Software Assurance ab.

2.3 Funktionale und nicht-funktionale Sicherheit

Häufig wird Softwaresicherheit ausschließlich auf funktionale Aspekte wie Authentisierung, Autorisierung und Kryptographie bezogen. In der Praxis lässt sich allerdings eine Vielzahl von realen Sicherheitsproblemen vielmehr auf mangelnde Softwarequalität als auf das Fehlen bestimmter Security Controls zurückführen. Zu diesen nicht-funktionalen Sicherheitsbestandteilen zählen etwa Fehler bei der Eingabe- und Ausgabevalidierung, die die Ursache für Angriffe wie Cross-Site-Scripting oder SQL Injection bilden.

4 Allgemein werden Assets nach ISO 27001 als „anything that has value to the organisation“ definiert. In ISO 27005 [ISO2] werden Assets weiter in primary Assets, die Geschäftsprozesse und Informationen umfassen, und supporting Assets, die Informations- und Kommunikationssysteme umfassen, unterteilt. Bei der Betrachtung von Webanwendungen beziehen sich Assets in erster Linie auf die von ihr verarbeiteten Daten. Diese Assets haben zentrale Bedeutung, um den Schutzbedarf zu definieren und davon, losgelöst vom Risikomanagement, entsprechende Controls abzuleiten.

2.4 Risiken unsicherer Software

Von Risiken durch unsichere Software können sämtliche Interessensvertreter (Stakeholder) in unterschiedlicher Form betroffen sein.

2.4.1 Risiken für Organisationen

Für Organisationen (Auftraggeber bzw. Auftragnehmer) ergeben sich zahlreiche Risiken, die einen beträchtlichen finanziellen Schaden zur Folge haben können.

Diese lassen sich größtenteils einer der folgenden Gruppen zuordnen:

1) Ausfall, Einschränkung oder Missbrauch kritischer Geschäftsfunktionen
2) Entwendung vertraulicher Unternehmens- oder Kundendaten
3) Wettbewerbsnachteile durch ungenügende Adressierung von Sicherheit in vertriebener Software
4) Negative Außendarstellung und negativer Einfluss auf die Marke durch publik gewordene Sicherheitslücken oder publik gewordene Daten
5) Verstoß gegen externe Vorgaben (etwa Gesetze, verbindliche Standards etc.)

Die letzten drei genannten Risiken treffen dabei auch auf einen Auftragnehmer (Softwarehersteller) zu.

2.4.2 Risiken für Benutzer

Aber auch für die Benutzer (etwa Kunden) einer Webanwendung selbst ergeben sich mitunter immense Risiken. Insbesondere deshalb, da Webanwendungen häufig eine Vielzahl sensibler Benutzerdaten verarbeiten und Transaktionen für diese durchführen.

Auch hier lässt sich der Großteil der Risiken in bestimmte Kategorien einordnen:

1. Entwendung vertraulicher Daten
2. Missbräuchliche Verwendung von Benutzeraccounts und Benutzerdaten
3. Risiken durch uneingeschränkte Verwendung kritischer Geschäftsfunktionen

2.5 Ursachen unsicherer Software

Die Ursachen für die Entwicklung unsicherer Software aus Sicht des Auftragnehmers können unterschiedlich sein. In diesem Kapitel werden einige zentrale Ursachen für unsichere Software dargestellt.

2.5.1 Komplexität

Komplexität ist eine Eigenschaft von Systemen, die sich im Allgemeinen sehr nachteilig auf deren Sicherheit auswirkt. Durch Komplexität können Schwachstellen in die Anwendung eingebracht werden und sie erhöht vor allem in erheblichem Maße den Aufwand, der für deren Verifikation erforderlich ist.

Komplexität wird häufig auch durch externe Komponenten vergrößert. So setzen sich moderne Webanwendungen häufig aus einer Vielzahl eingesetzter APIs, Frameworks und anderer Komponenten zusammen. In den seltensten Fällen schöpft eine Webanwendung den vollen Funktionsumfang der zugrunde liegenden Frameworks aus. Aus diesen ungenutzten Funktionalitäten ergibt sich jedoch eine vergrößerte Angriffsfläche und damit steigt auch das Risiko, dass Funktionalitäten missbraucht oder Schwachstellen in der Webanwendung ausgenutzt werden können.

Eines der zentralen Sicherheitsprinzipien lautet aus diesem Grund, auch nach dem Grundsatz „Keep it small and simple“, das Gesamtsystem so einfach wie möglich zu halten.

2.5.2 Fehlende Sichtbarkeit und Messbarkeit

Um die Qualität von Software zu steuern, muss die Qualität messbar sein. Da die Sicherheit von Software auch ein Teilaspekt der Softwarequalität ist, gilt dies auch für diesen Bereich. Fehlende Sichtbarkeit und Messbarkeit ergibt sich daraus, dass Sicherheit gewöhnlich nur dann wahrgenommen wird, wenn diese nicht oder nur unzureichend vorhanden ist, beispielsweise wenn Sicherheitslücken identifiziert werden. Diese fehlende Sichtbarkeit und Messbarkeit der Softwaresicherheit stellt sich insbesondere während der Qualitätssicherung als Problem dar. Aus diesem Grund müssen Metriken definiert werden. Anhand dieser können Softwareprojekte miteinander verglichen werden und das Sicherheitsniveau der Anwendung kann abgeschätzt werden. Solche Metriken können beispielsweise die Anzahl identifizierter und behobener Schwachstellen sein oder die Höhe des Risikos, das von verbleibenden Schwachstellen ausgeht.

2.5.3 Fehlende Sicherheitsanforderungen

Konkrete Anforderungen an die Anwendungssicherheit stellen die Basis einer sicheren Anwendungsentwicklung dar. Werden keine oder unzureichende Sicherheitsanforderungen zu Beginn der Entwicklung definiert, können die davon abhängigen Aktivitäten nicht effektiv durchgeführt werden. Hinzu kommt die Schwierigkeit, diese Anforderungen präzise zu formulieren, um Missverständnissen vorzubeugen und die spätere Validierung der Einhaltung der Anforderungen zu gewährleisten. Besonders problematisch stellt sich das präzise Formulieren von Anforderungen in agilen Vorgehensweisen dar, wo auf Grund häufig geänderter Anwendungsfälle Anforderungen an die Anwendungssicherheit dementsprechend angepasst werden müssen. Fehlende oder unzureichende Anforderungen an die Sicherheit von Software sind oft eine wesentliche Ursache für spätere Sicherheitsprobleme.

2.5.4 Mangelndes Sicherheitsbewusstsein

Unzureichendes Sicherheitsbewusstsein im gesamten Projektteam kann zu Fehleinschätzungen des Schutzbedarfs im Allgemeinen sowie der Auswirkungen von implementierten Funktionen und zu Fehlern bei der konkreten Implementierung führen.

2.5.5 Schlechte Testbarkeit

Das Testen von Software auf Sicherheit ist aus verschiedenen Gründen kein triviales, sondern ein enorm schwieriges Thema. Auch deshalb scheitern selbst die angesehensten Hersteller daran, Tools herzustellen, die dieses hinreichend gut ermöglichen.

Daher müssen Aktivitäten der Software Assurance so früh wie möglich im Entwicklungsprozess adressiert und in den entsprechenden Phasen angemessen berücksichtigt werden. Denn je schlechter sich die Testbarkeit einer Anwendung gestaltet, desto leichter können Sicherheitsprobleme übersehen werden. Schlechte Testbarkeit erhöht nicht nur die Aufwände für Sicherheitsüberprüfungen signifikant, sondern erhöht gleichzeitig das Risiko, dass Schwachstellen verborgen bleiben.

2.6 Standards und Best Practices

Der Leitfaden orientiert sich an zahlreichen etablierten Standards und Studien zum Thema Informationssicherheit. Dieser Abschnitt gibt einen kurzen Überblick über die Einflüsse dieser Dokumente.

Beim Eingehen auf Vorgaben, Standards und Best Practices wird insbesondere Wert auf Publikationen rund um das Thema Webanwendungssicherheit gelegt.

2.6.1 BSI-Standards zur Internet-Sicherheit (ISi-Reihe), Modul „ISi Web Server“[5]

Die ISi-Reihe bietet allen Personen in Behörden und Unternehmen, die sich mit der IT-Sicherheit befassen, umfassende und aktuelle Informationen zu weiten Bereichen der IT-Sicherheit. Durch ihren Aufbau aus mehreren Teilen wendet sie sich zielgruppenspezifisch mit dem Leitfaden ISi-L an Entscheidungsträger sowie mit den Studien ISi-S an diejenigen, die mit der Umsetzung der IT-Sicherheit betraut sind. Die unterschiedlichen Module der ISi-Reihe befassen sich jeweils mit einem sicherheitsrelevanten Teilbereich von IT-Systemen.

Zusammen mit der Studie zur sicheren Nutzung von Web-Angeboten (ISi-Web-Client) können bestehende Systeme in Hinblick auf den Dienst Web abgesichert oder auch neue konzipiert, umgesetzt und betrieben werden. Hierzu sind die Checklisten, die ergänzend zu jeder Studie veröffentlicht werden, eine praktische Hilfe.

Die Vorgaben der ISi-Reihe werden in vielen Bereichen des Leitfadens berücksichtigt. Insbesondere das sichere Design und der Betrieb der Infrastruktur werden durch die ISi-Reihe unterstützt.

2.6.2 BSI IT-Grundschutz[6]

Der BSI IT-Grundschutz stellt eine Sammlung von Dokumenten, sogenannten Bausteinen, dar, mit denen sich für die meisten IT-Systeme mit normalem Schutzbedarf ein angemessenes Sicherheitsniveau umsetzen lässt. Trotz des großen Umfanges der IT-Grundschutzkataloge bilden diese den Themenbereich sichere Softwareentwicklung derzeit noch nicht mit einem entsprechenden Baustein ab, eine Vorabversion ist jedoch bereits verfügbar. Websicherheit wird in den Grundschutzkatalogen zum einen durch Konfigurationsempfehlungen, etwa für Webserver, behandelt und zum anderen durch einen Baustein zu Web-Anwendungen abgedeckt.

2.6.3 Sicherheit von Webanwendungen – Maßnahmenkatalog und Best Practices (BSI)[7]

Das Dokument „Sicherheit von Webanwendungen - Maßnahmenkatalog und Best Practices“ bietet eine umfangreiche Zusammenstellung von Schutzmaßnahmen und Best Practices zur Vorbeugung gegen typische Schwachstellen in Webanwendungen. Hierbei werden sowohl bereits bestehende als auch neu zu entwickelnde Webanwendungen betrachtet. Diese Zusammenstellung bietet sich insbesondere als Ergänzung zu Kapitel 5 „Vorgaben an die Implementierung“ an.

2.6.4 ÖNORM A 7700[8]

Die ÖNORM A 7700 legt Anforderungen an die Sicherheit von Webapplikationen fest. Sicherheit wird durch Wahrung der Vertraulichkeit, Integrität und Verfügbarkeit aller von der Anwendung verarbeiteten Informationen erfüllt. Die Klassifizierung dieser Anforderungen obliegt dem Anwender dieser ÖNORM und ist schriftlich zu dokumentieren.

5 [ISiW]
6 [ITGS]
7 [SiWe 2006]
8 [ÖN77]

Diese ÖNORM bezieht sich ausschließlich auf die Webapplikation selbst. Komponenten, die zum Betrieb der Anwendung benötigt werden (z.B. Betriebssystem, Webserver, Netzwerkkomponenten) sowie Hintergrundsysteme (z.B. Datenbanken, Legacysysteme), werden nicht behandelt.

Ist die Sicherheit der Webapplikation untrennbar mit der Konfiguration einer zugrunde liegenden Komponente verbunden, muss die relevante Konfiguration in den Anwendungsbereich mit aufgenommen werden.

2.6.5 OWASP Application Security Verification Standard (ASVS)[9]

Ziel des OWASP Application Security Verification Standard (ASVS) Projekts ist es, einen kommerziell einsetzbaren, offenen Standard zur Verfügung zu stellen, der den Abdeckungsgrad einer Sicherheitsüberprüfung von Webapplikationen mehrstufig definiert und den vorzunehmenden Umfang vorschreibt. Der Standard stellt eine Vorgehensweise zur Verfügung, mit der nicht nur technische Sicherheitskontrollen der Applikation durchgeführt werden können, sondern auch die relevanten Umfelder miteinbezogen werden.

Um diese Ziele zu erreichen, definiert der ASVS insgesamt vier Prüflevel, die unterschiedlich hohe Vorgaben an die Verifikation der Sicherheit einer Webanwendung stellen. So fordert das ASVS Level 1 die Überprüfung der Webapplikation mittels automatisierter Tools, wohingegen zur Erreichung des nächsten Levels bereits ein manueller Test notwendig ist. Dieser modulare Aufbau liefert vor allem bei der Definition der Audit-Pläne in Abhängigkeit zum Schutzbedarf wertvolle Informationen.

2.6.6 OWASP Open SAMM[10]

Das Open Software Assurance Maturity Model (SAMM) ist ein offenes Framework und unterstützt Organisationen bei der Formulierung und Implementierung einer Strategie zur sicheren Softwareentwicklung. Der Einsatz führt zu:

- Evaluierung der vorhandenen sicherheitstechnischen Software Praktiken
- Erstellen eines ausgeglichenen Software Security Assurance Programms
- Darstellung konkreter Verbesserungen im Security Assurance Programm
- Definieren und Messen sicherheitsrelevanter Aktivitäten innerhalb einer Organisation

Das Framework definiert vier Kategorien (Business Functions). Jede Kategorie fasst praxisorientierte Aktivitäten (Security Practices) im Bereich Softwareentwicklung zusammen. Zu jeder Aktivität werden drei Reifegrade (Maturity Levels) definiert. Jeder Reifegrad spezifiziert die umzusetzenden kritischen Faktoren, um den Level erfolgreich abzuschließen. Jeder abgeschlossene Reifegrad steigert den Reifegrad des Entwicklungsprozesses und somit die Sicherheit der zu entwickelnden Software.

2.6.7 BSIMM

Das Building Security In Maturity Model (BSIMM)[11] ist dem Open SAMM sehr ähnlich. Anders als dieses stellt BSIMM jedoch kein Framework zur Einführung von Sicherheitsaktivitäten in die Softwareentwicklung dar, sondern ist vielmehr eine Studie der Firma Cigital, die diese in zahlreiche große Unternehmen (vornehmlich in den USA und Europa) analysiert und bewertet hat. BSIMM erlaubt damit wertvolle Rückschlüsse für eigene Initiativen in diesem Bereich, ohne dabei eine detaillierte Vorgehensweise hierfür zu liefern.

9 [ASVS]
10 [SAMM]
11 [BSIM 2011]

Die durch BSIMM beschriebenen Praktiken (die BSIMM-Bezeichnung für Sicherheitsaktivitäten) werden dabei in dem Software Security Framework (SSF) zusammengefasst. Das SSF besteht aus vier Domänen, zwölf Praktiken und 110 Aktivitäten.

3 Einführung von Sicherheit in der Organisation

Software sicher zu entwickeln ist eine komplexe Herausforderung. Umso wichtiger ist daher eine strukturierte Herangehensweise, welche die notwendigen Aktivitäten sowie die Bewertungskriterien definiert und damit eine nachhaltige Berücksichtigung von Softwaresicherheit im Rahmen der Softwareentwicklung gewährleistet. Folglich werden in diesem Kapitel allgemeine Anforderungen definiert, die eine Organisation sowohl bei der Konzeption und Implementierung als auch bei der Verifikation von Sicherheit unterstützen.

Die Anforderungen können als Rahmenbedingungen für Softwarehersteller verwendet werden, bevor diese an Ausschreibungsverfahren teilnehmen. Weitere wichtige Vorgaben werden in Kapitel 4 und Kapitel 5 ausführlich behandelt. Sämtliche Anforderungen an den Auftragnehmer werden schlussendlich in Kapitel 6 in Form von Checklisten zusammengefasst. Die Checklisten bilden den Kern dieses Dokumentes und definieren einerseits die notwendigen Aktivitäten, die vom Auftragnehmer verpflichtend umzusetzen sind, aber auch zusätzlich optionale Aktivitäten, um ein noch höheres Sicherheitsniveau zu erreichen.

3.1 Einführung eines Secure Development Lifecycle im gesamten Unternehmen

Die Erkenntnis Softwarefehler frühzeitig dort zu behandeln bzw. ihnen dort vorzubeugen wo, diese auch entstehen, ist sicherlich nicht gerade neu. So zeigte bereits 1997 der Experte für Softwarequalität Casper Jones[12], wie mit fortschreitender Entwicklung die Kosten für die Behebung von Fehlern gleichsam sprunghaft ansteigen und wie deren Erkennungsrate zurückgeht (Abbildung 5):

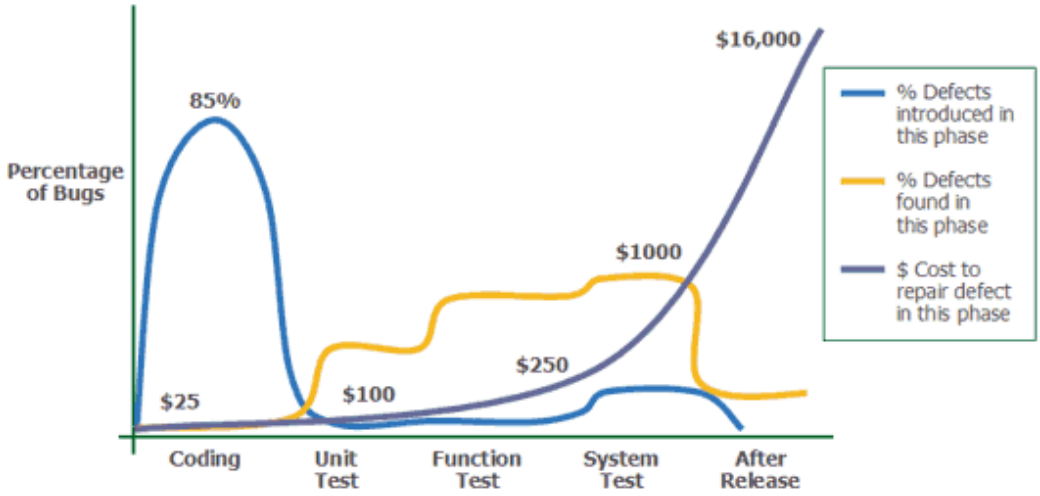

Abbildung 1: Kosten für Fehlerbehebung

Dabei können laut Gilb bereits 60 Prozent[13] der Fehler im Rahmen der Designphase identifiziert werden.

Auch das Sicherheitsniveau einer Anwendung wird grundlegend in frühen Phasen deren Entwicklung bestimmt. Spätere Änderungen sind vergleichbar mit dem Versuch, einen Fehler im Rohbau eines Hauses im Nachhinein zu korrigieren.

Da sicherheitsrelevante Entscheidungen im gesamten Lebenszyklus einer Webanwendung getroffen werden, leitet sich hieraus die Notwendigkeit ab, diese auch in jeder einzelnen Entwicklungsphase entsprechend zu berücksichtigen.

12 [Casp 1997]
13 [Gilb 1988]

Dies wird erreicht, indem über dem existierenden Software Development Lifecycle (SDLC) ein Secure Development Lifecycle (SDL) definiert wird, der entsprechende Aktivitäten zur Erstellung bzw. Analyse der Sicherheit jeder einzelnen Phase, also eine Art sicheren Entwicklungsprozess, definiert.

Microsoft hat mit dem MS Security Development Lifecycle bereits im Jahre 2002 im Rahmen seiner von Bill Gates persönlich initiierten Trustworthy Computing Initiative (TCI)[14] einen häufig zitierten Entwurf eines solchen SDLs vorgestellt. Abbildung 2 skizziert die unterschiedlichen Schritte des Entwicklungsprozesses von Microsoft. Über die vergangenen Jahre wurde dieser immer weiterentwickelt, um dabei neuen Entwicklungen wie agilen Vorgehensweisen Rechnung zu tragen. Andere Hersteller wie etwa Cisco oder HP haben hier inzwischen ähnliche Ansätze vorgestellt.

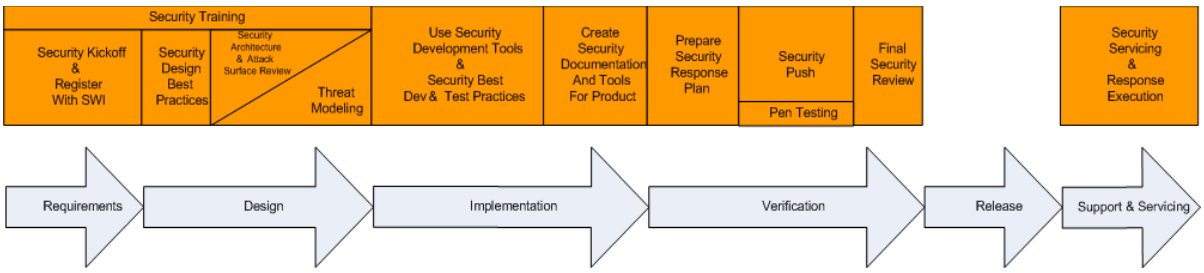

Abbildung 2: Entwicklungsprozess von Microsoft

3.2 Reifegrade

Organisationen entwickeln auf teilweise sehr unterschiedliche Art und Weise Software und auch in Bezug auf Sicherheit findet man große Unterschiede. Dies betrifft sowohl das aktuell erreichte Sicherheitsniveau als auch die angestrebten Sicherheitsziele. Diese Heterogenität zwischen den Organisationen muss berücksichtigt werden, wenn die Einführung von Sicherheitsaktivitäten (auch als „Sicherheitsinitiative“ bezeichnet) geplant und angegangen wird. Das Verständnis von Reifegraden ist dabei elementar. Solche Reifegrade werden von den bereits erwähnten Studien BSIMM und Open SAMM einbezogen.

Der Gedanke ist dabei einfach: Jede Aktivität einer Phase des Entwicklungsprozesses besitzt einen bestimmten Reifegrad. Sowohl hinsichtlich des aktuellen Implementierungsgrades (Ist-Zustandes) als auch hinsichtlich gewünschter oder theoretisch denkbarer Ausprägungen (Soll-Zustand). Der Reifegrad einer Aktivität resultiert aus der Gesamtheit der Teilaktivitäten, die in dieser Aktivität durchgeführt werden.

Gleichwohl bieten Reifegrade auch die Möglichkeit, einen Secure Development Lifecycle (SDL) zu bewerten und Unzulänglichkeiten aufzudecken. Häufig haben Unternehmen etwa einen sehr hohen Reifegrad beim Security Testing, berücksichtigen den Bereich der konzeptionellen Sicherheit dagegen kaum. Bezieht man den Reifegrad auf den SDL, wird dabei die Gesamtheit der umgesetzten Aktivitäten (und deren Reifegrade) betrachtet. Reifegrade spielen auch bei der strukturierten Planung und Einführung von Aktivitäten in der Organisation eine entscheidende Rolle. Im Rahmen dieses Leitfadens dienen Reifegrade dazu, einen erforderlichen Reifegrad an den gesamten SDL zu definieren. Für die visuelle Darstellung der Reifegrade und insbesondere für die Darstellung deren gewünschter Entwicklung können Scorecards (Kapitel 3.3) herangezogen werden.

3.3 Scorecards

Zur Unterstützung der Umsetzung der sicheren Softwareentwicklung (insbesondere zur Anhebung des Reifegrades) können Scorecards herangezogen werden. Jede Phase des Entwicklungsprozesses wird mit einem Reifegrad anhand der umgesetzten Aktivitäten bewertet und in der Scorecard eingetragen. Zusätzlich wird in der Scorecard zu jeder Phase des Entwicklungsprozesses der angestrebte Ziel-Reifegrad eingetragen.

14 [Micr 2005]

Scorecards dienen in erster Linie der anschaulichen Darstellung des derzeitigen Reifegrades sowie des angestrebten Reifegrades. Sie können aber auch zur Gap Analyse (Gegenüberstellung des SOLL- und IST-Zustandes) sowie zu laufenden Messungen herangezogen werden. Ein Beispiel für eine Scorecard zeigt Abbildung 3[15]. Der Unterschied zwischen IST und SOLL kann mit Unterstützung einer Roadmap (Kapitel 3.4) geplant und nach erfolgreicher Umsetzung verringert werden.

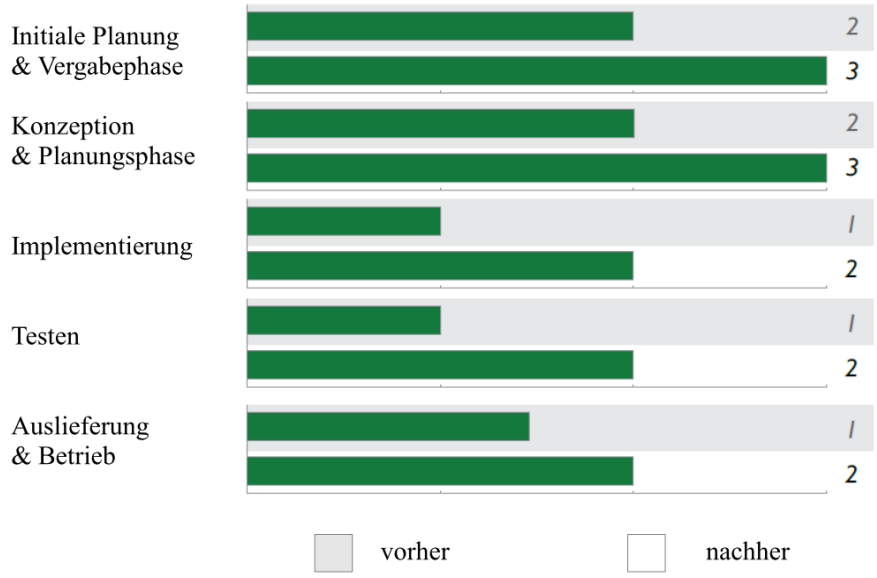

Abbildung 3: Beispiel für Scorecard

3.4 Roadmaps

Jede Vorgehensweise muss für eine erfolgreiche und nachvollziehbare Umsetzung geplant werden. Die Planung für die Einführung von Sicherheit in der Softwareentwicklung kann mit Hilfe einer Roadmap erfolgen. Roadmaps definieren, wie der Reifegrad einer Phase iterativ angehoben werden kann. Hierbei werden bestimmte Aktivitäten in bestimmten Iterationen definiert und umgesetzt. Je nach Organisation kann die Anzahl der Iterationen variieren. Hier ein Beispiel für Iterationen, die häufig verwendet werden:

- Iteration 1: Kurzfristig, z.B. innerhalb der nächsten drei Monate
- Iteration 2: Mittelfristig, z.B. innerhalb des aktuellen Jahres
- Iteration 3: Mittelfristig, z.B. innerhalb eines Jahres
- Iteration 4: Langfristig, z.B. innerhalb der nächsten zwei bis drei Jahre

15 [SAMM]

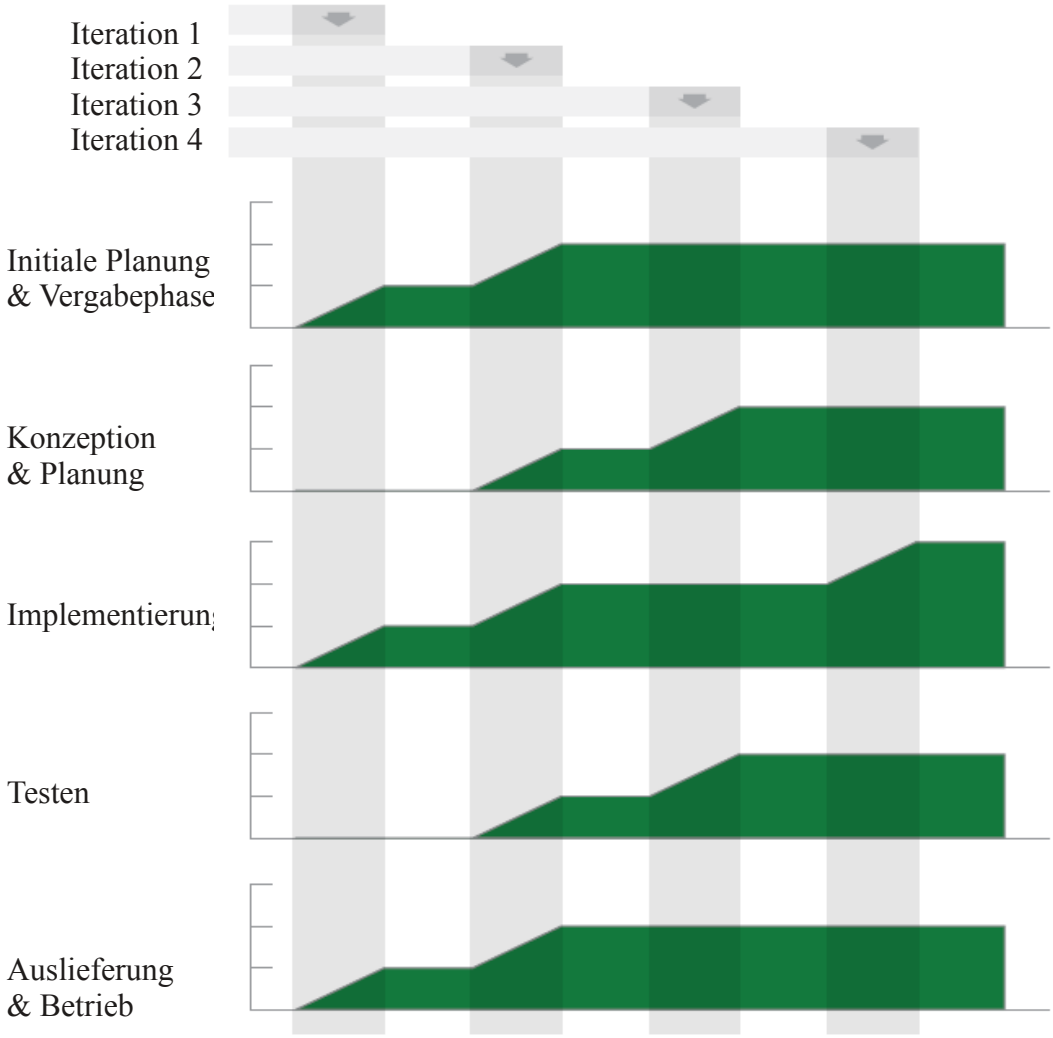

Abbildung 4: Beispiel für Roadmap

Den Organisationen steht es frei zu entscheiden, in welcher Iteration welche Aktivität umgesetzt werden soll. Ein Beispiel für eine grafische Darstellung einer Roadmap zeigt Abbildung 4[16].

An dem Beispiel wird gezeigt, dass die betreffende Organisation sich etwa Aktivitäten im Bereich Initiale Planung & Implementierung unmittelbar zuwendet, die Konzeption jedoch erst in Iteration 2 angeht. Am Ende von Iteration 4 sollen dann alle Bereiche in ausreichendem Maße abgebildet worden sein.

Wichtig: Das hier gezeigte Beispiel ist keine generelle Empfehlung, sondern soll lediglich die Einbeziehung einer Roadmap in die Planung verdeutlichen. In vielen Fällen wird es etwa durchaus Sinn machen, Aktivitäten im Bereich Testen sehr früh zu adressieren.

3.5 Sicherheitsspezifische Aktivitäten

Dieses Kapitel erläutert allgemeine sicherheitsspezifische Aktivitäten für den Entwicklungsprozess.

16 [SAMM]

3.5.1 Angemessene Sicherheit

Sicherheit betrifft den gesamten Lebenszyklus einer Webanwendung und folglich auch jede Phase deren Entwicklung. Da dies mitunter erhebliche Aufwände zur Folge haben kann, ist es wichtig, ein für eine Anwendung erforderliches Niveau an Sicherheit zu ermitteln.

Wie bei infrastrukturellen Komponenten existiert auch für Webanwendungen ein bestimmtes Mindestschutzniveau, das immer umzusetzen ist. Dieses entspricht der generellen Sorgfaltspflicht, die ein Auftraggeber von einem Auftragnehmer erwarten kann. Etwa, dass ein Auftragnehmer sicherstellt, alle nötigen Schritte zu ergreifen, um sicherzustellen, dass eine Webanwendung nach Möglichkeit keine kritische Sicherheitslücken oder andere grobe Qualitätsmängel besitzt.

Komplizierter wird es bei funktionalen Sicherheitsaspekten wie Authentisierung, Autorisierung oder auch Aktivitäten im Bereich Software Assurance (Reviews, Tests).

Abhängig sind diese dabei vor allem vom Schutzbedarf der Anwendung bzw. der durch diese verarbeiteten Daten. Eine Anwendung, die hochsensible Daten über das Internet abruft, wird somit im Allgemeinen einen recht hohen Schutzbedarf besitzen, aus dem sich folglich auch Anforderungen an genannte Sicherheitsaspekte ableiten, die über das Mindestschutzniveau hinausgehen.

3.5.2 Low Hanging Fruits

Sicherheitsmaßnahmen, die mit wenig Ressourcenaufwand umgesetzt werden können, sollten in der Regel immer getroffen werden. Solche Low Hanging Fruits finden sich im gesamten Lebenszyklus der Entwicklung. Bei der Definition von Sicherheitsanforderungen von Webanwendungen kann beispielsweise auf bestehende Standards wie den IT-Grundschutz oder die ÖNORM A 7700 referenziert werden. Bei der Implementierung können für gängige Programmiersprachen verfügbare Secure Coding Guidelines verwendet werden und für Security Tests können automatisierte Tools eingesetzt werden, um einfach zu identifizierende Schwachstellen aufzudecken und ihrer Behebung zuzuführen.

Dies sind nur einige Beispiele, wie mit wenig Aufwand das Sicherheitsniveau signifikant erhöht werden kann.

3.5.3 Erstellung von Richtlinien

Mit einer Sicherheitsrichtlinie werden die Schutzziele und Sicherheitsmaßnahmen unter Berücksichtigung der Vorgaben von Unternehmen und Behörden formuliert. Doch die besten Überlegungen und Maßnahmen zur Erhöhung der Informations-Sicherheit sind wertlos, wenn diese nicht entsprechend dokumentiert und kommuniziert werden. Erst durch die schriftliche Dokumentation werden die Sicherheit bzw. die dafür getroffenen Maßnahmen analysierbar und damit auch nachweisbar.

3.5.3.1 Management der Sicherheitsrichtlinie

Während des gesamten Sicherheitsprozesses eines Unternehmens entstehen viele Dokumente wie z.B. Sicherheitsrichtlinien, Konzepte, Verfahrensbeschreibungen oder Arbeitsanweisungen. Um den Überblick über diese Dokumente zu behalten, ist es wichtig, eine tragfähige Struktur aufzubauen. Einen guten Ansatz stellt dabei, wie in Abbildung 5 gezeigt, eine Pyramidenform dar. Durch den hierarchischen Aufbau lassen sich Dokumente schnell in das System einordnen und die Auswirkung von möglichen Änderungen leichter abschätzen.

Abbildung 5: Hierarchischer Aufbau von Richtlinien [ITGS]

Für jede Richtlinie muss es innerhalb des Unternehmens einen Verantwortlichen geben, der jedes neue Dokument und jede Änderung an bestehenden Dokumenten nach einer entsprechenden Prüfung genehmigt und damit das Dokument zur Verteilung freigibt. Änderungen an Richtlinien sollten nicht nur nach Zwischenfällen, sondern auch im Zuge von regelmäßigen Überprüfungen erfolgen. Die Intervalle für ein Dokumentenreview müssen im Dokument selbst festgehalten werden. Um Änderungen nachvollziehbar zu machen, ist es wichtig, dass Änderungen und der aktuelle Überarbeitungsstatus der Dokumente gekennzeichnet werden. Um die Verwendung veralteter Dokumente zu verhindern, ist nach genehmigten Änderungen sicherzustellen, dass alle von der Richtlinie betroffenen Mitarbeiter die aktuelle Fassung zur Verfügung haben. Durch Klassifizierung der Dokumente und ein entsprechendes Berechtigungskonzept muss sichergestellt werden, dass die Dokumente nur den Personen zugänglich sind, die sie benötigen.

3.5.3.2 Inhalte der Sicherheitsrichtlinie

Mit einer Sicherheitsrichtlinie werden folgende Punkte festgelegt:

- **Anwendungsbereich:** In diesem Abschnitt wird festgelegt, für welchen Bereich (Standort, Organisationseinheiten, Geschäftsprozesse) die Richtlinie gilt, welche Personen davon betroffen sind bzw. Zugang zum Dokument erhalten sollen und welche Informationen die Richtlinie betrifft.
- **Bedeutung von Informations-Sicherheit:** Um die Bedeutung der Sicherheit entsprechend kommunizieren zu können, ist es wichtig dazustellen, welchen Wert der entsprechende Anwendungsbereich bzw. seine Sicherheit für das Unternehmen hat. Eine Bewertung könnte die Abhängigkeit des Unternehmens vom funktionsfähigen Geschäftsprozess oder die Auswirkung eines Sicherheitsvorfalls sein.
- **Gefährdungsanalyse:** In diesem Bereich wird die Gefährdungslage für den Anwendungsbereich dargelegt.
- **Vorgaben:** Datenschutzgesetz, Signaturgesetz oder das Telekommunikationsgesetz sind nur einige Beispiele von gesetzlichen Bestimmungen, an die sich ein Unternehmen halten muss. Solche Vorgaben müssen frühzeitig dokumentiert werden, um diese auch im Entwicklungsprozess entsprechend berücksichtigen zu können.
- **Änderungsübersicht:** Durchgeführte Änderungen müssen dokumentiert und das Dokument muss entsprechend versioniert werden, um Änderungen nachvollziehbar zu machen, Verantwortlichkeiten zu dokumentieren und sicherzustellen, dass alle betroffenen Personen mit der aktuellen Version arbeiten. Auch ein Zeitplan für die regelmäßige Prüfung der Richtlinie muss im Dokument enthalten sein.

- **Verpflichtungserklärung:** Jeder Mitarbeiter ist dazu verpflichtet, die für ihn relevanten Sicherheitsrichtlinien zu kennen und einzuhalten. In diesem Zuge muss jeder Mitarbeiter mit einer Verpflichtungserklärung bestätigen, die Richtlinie gelesen und verstanden zu haben sowie sich zu ihrer Einhaltung zu verpflichten. Eine solche Verpflichtungserklärung ist allgemeiner Teil der Personalakte, der entsprechende Text kann aber der Richtlinie beigefügt werden.

3.5.4 Awareness Trainings & Schulungen

Unter dem Begriff Awareness versteht man die Bewusstseinsbildung hinsichtlich Sicherheitsthemen. Alle an der Entwicklung einer Anwendung beteiligten Mitarbeiter müssen mit dem Thema Sicherheit konfrontiert werden. Das betrifft sowohl Entwickler als auch Projektmanager, Qualitätstester, Administratoren usw.

Es muss sichergestellt werden, dass von jedem Mitarbeiter die Folgen von Sicherheitsmängeln sowie geltenden Anforderungen verstanden werden. Auch das Thema Datenschutz sollte in diesem Rahmen in ausreichendem Maße adressiert werden. Diese Bewusstseinsbildung erfolgt zumeist durch Schulungen.

Diese Schulungen können sowohl von externen als auch internen Personen durchgeführt werden (etwa durch die Software Security Group). Eine solche Veranstaltung kann abhängig von den spezifischen Erfordernissen entweder als mehrtägiger Workshop oder als computergestützter modularer Kurs aufgebaut werden. Der Inhalt sollte sowohl konzeptuelle als auch technische Informationen beinhalten.

Die Ausgestaltung kann und sollte dabei unterschiedliche Zielgruppen angemessen adressieren. Einem Projektmanager muss in der Regel nicht das gleiche technische Verständnis vermittelt werden, wie dies etwa bei Entwicklern der Fall ist, insbesondere wenn diese mit der konkreten Implementierung von sicherheitsrelevanten Funktionen betraut worden sind.

Ein grundlegendes Awareness-Training sollte für alle an der Entwicklung einer Webanwendung beteiligten Personen verpflichtend sein und periodisch durchgeführt werden. Neue Mitarbeiter sollen ein entsprechendes Training unmittelbar nach Firmeneintritt erhalten.

3.5.5 Etablierung von Prozessen zur Qualitätssicherung

Um die Sicherheit und damit auch die Qualität der Software zu erhöhen, ist es notwendig, während des gesamten Entwicklungszeitraums sicherheitsspezifische Tests durchzuführen. Je früher dabei Schwachstellen identifiziert werden, desto geringer fallen die zur Behebung notwendigen Kosten aus. Eine häufig verwendete Methodik ist die Verwendung von Security Gates. Security Gates stellen eine Form der innerhalb der Qualitätssicherung häufig verwendeten Quality Gates dar. Dabei handelt es sich um Meilensteine im Projekt, die nur unter Einhaltung definierter Sicherheits- bzw. Qualitätskriterien passiert werden dürfen. Security Gates werden häufig mit den Ergebnissen obligatorischer Sicherheitstests verknüpft. So ließe sich bestimmen, dass eine Anwendung vor dem Produktivgang einen Penetrationstest durchlaufen muss und durch diesen keine gravierenden Mängel identifiziert werden dürfen.

Security Gates helfen damit, ein Mindestmaß an Sicherheit in einem Projekt umzusetzen.

4 Vorgaben an den Entwicklungsprozess

In diesem Kapitel werden verpflichtende und optionale Sicherheitsaktivitäten für den Entwicklungsprozess (also ein Secure Development Lifecycle) beschrieben. Wie bereits erwähnt, entwickelt jede Organisation Software auf unterschiedliche Weise. Zur Gewährleistung einer sicheren Anwendungsentwicklung sollten die beschriebenen Aktivitäten pragmatisch und ergebnisorientiert an die individuellen Gegebenheiten angepasst werden.

4.1 Aufbau

Die Umsetzung der sicheren Softwareentwicklung erfolgt mit Hilfe eines Entwicklungsprozesses, der sich in die fünf Phasen „Initiale Planung & Vergabe“, „Konzeptions- & Planungsphase“, „Implementierung“, „Testen“ und „Auslieferung & Betrieb“, wie in Abbildung 6 dargestellt, unterteilt.

Abbildung 6: Entwicklungsprozess

Jede Phase stellt setzt sich aus unterschiedlichen Aktivitäten zusammen. Diese sind abhängig vom Schutzbedarf der Anwendung in unterschiedlichen Ausprägungen durchzuführen. Auf Basis der Checkliste für den Entwicklungsprozess in Kapitel 6.1 lässt sich feststellen, ob die Aktivitäten für den geforderten Schutzbedarf durchgeführt werden.

Darauf basierend lassen sich folgende Reifegrade über den gesamten SDL definieren:

0. **Nicht vorhanden**
   Mit der Umsetzung der Aktivitäten wurde nicht begonnen.
   (0% der Punkteanzahl für verpflichtende Aktivitäten wurden erreicht)
1. **Ad-Hoc** (Informell)
   Die Aktivitäten werden zwar teilweise umgesetzt, sind jedoch nicht standardisiert oder kontrolliert.
   (mehr als 0% und weniger als 50% der Punkteanzahl für verpflichtende Aktivitäten wurden erreicht)
2. **Partiell**
   Mit der Umsetzung der Aktivitäten wurde begonnen. Es wurden jedoch nicht alle Aktivitäten umgesetzt.
   (mindestens 50% und weniger als 100% der Punkteanzahl für verpflichtende Aktivitäten wurden erreicht)
3. **Vollständig** (erforderliches Niveau)
   Es wurden alle verpflichtenden Aktivitäten umgesetzt.
   (100% der Punkteanzahl für verpflichtende Aktivitäten wurden erreicht)
4. **Best in Class**
   Es wurden alle verpflichtenden und darüber hinaus zusätzlichen optionalen Aktivitäten umgesetzt. Dadurch haben Auftragnehmer die Möglichkeit, weitere Punkte zu erreichen und sich so vom Mitbewerber abzuheben.

Diese Vorgehensweise ermöglicht eine objektive Bewertung des SDL eines Auftragnehmers. Ein Auftragnehmer muss mindestens den Reifegrad „Vollständig“ erfüllen, um als Auftragnehmer in Betracht gezogen zu werden. Zudem werden Auftragnehmer motiviert, den Reifegrad „Best in Class“ zu erreichen.

Jede zusätzliche Aktivität kann einen Vorteil gegenüber einem anderen Mitbewerber bringen. Der Auftraggeber entscheidet nach einem transparenten Puktesystem.

Nachfolgend werden die Phasen mit ihren Aktivitäten näher erläutert.

4.2 Phase 1: Initiale Planung & Vergabephase

Die erste Phase ist die initiale Planungs- und Vergabephase. Ziel ist die Planung notwendiger Sicherheitsressourcen innerhalb der initialen Projektplanung bzw. Vergabe und Angebotserstellung. Dazu werden sämtliche - soweit sie aus den Vergabeunterlagen abgeleitet werden können - sicherheitsrelevanten Informationen zu der zu entwickelnden Software identifiziert und ein Überblick über die für die Softwareentwicklung erforderlichen Ressourcen geschaffen, sodass etwa notwendige Aufwände abgeschätzt werden können. Diese Phase setzt sich aus fünf Aktivitäten zusammen, wie in Abbildung 7 dargestellt.

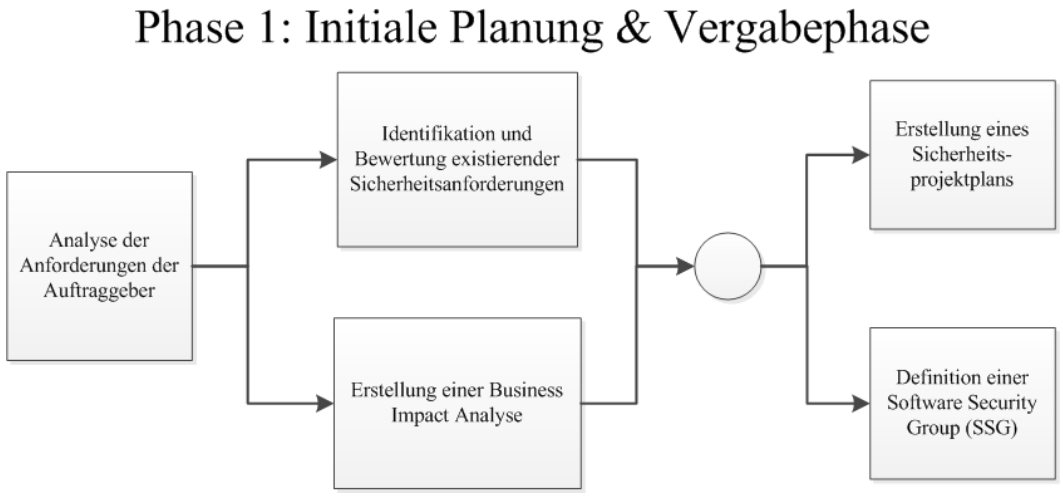

Abbildung 7: Aktivitäten der Phase 1 "Initiale Planung & Vergabephase"

4.2.1 IPV 1: Analyse der Anforderungen der Auftraggeber

Ausgehend vom Zweck der zu entwickelnden Webanwendung müssen in diese Teilphase die Anforderungen des Auftraggebers identifiziert und hinsichtlich ihrer Sicherheit bewertet werden. Fehlen sie oder werden sie falsch formuliert, kann daraus eine Kosten- bzw. Terminüberschreitung resultieren. Anforderungen können sich aber auch während des Projekts ändern oder stehen zum Beginn des Projekts noch nicht fest. Diese Möglichkeiten sind in der Analyse zu berücksichtigen.

4.2.2 IPV 2: Identifikation und Bewertung existierender Sicherheitsanforderungen

Bei der Identifikation und Bewertung existierender Sicherheitsanforderungen werden sämtliche bis zu diesem Zeitpunkt bekannten Sicherheitsanforderungen gesichtet und bewertet. Generell ist es wichtig, dass diese klar und deutlich formuliert sind und keinen Spielraum für Interpretationen offen lassen. Darüber hinaus muss jede Anforderung entsprechend begründet und messbar sein. Häufig existieren bereits gewisse Richtlinien in Unternehmen (Policies, Coding Guidelines, Standards etc.), die identifiziert und auf ihre Relevanz bewertet werden müssen. Rechtliche Vorgaben wie beispielsweise das BDSG sind einzubeziehen. Weitere Quellen für die Identifikation von Sicherheitsanforderungen sind die Ziele und Interessen der Stakeholder sowie die möglichen Auswirkungen von Sicherheitsmechanismen auf die funktionalen Spezifikationen.

Bei der Identifikation von Sicherheitsanforderungen sind zwei grundsätzliche Kategorien zu unterscheiden.

Funktionale Sicherheitsanforderungen decken eine konkrete Funktion der Anwendung ab. Beispiele sind:

- Benutzer Management (inkl. Authentisierung)
- Passwort-Management
- Datenbehandlung (inkl. Kryptographie)
- Rollen und Berechtigungen
- Schnittstellen

Funktionale Anforderungen lassen sich häufig auch direkt aus Anwendungsfällen ableiten und können im Rahmen sogenannter Security Tests, etwa durch das Qualitätssicherungsteam, während der späteren Abnahme der Anwendung geprüft werden. Funktionale Anforderungen, wie beispielsweise die Notwendigkeit der Implementierung einer 2-Faktor-Authentisierung, der Aufbau einer PKI, die Nutzung von SAML, WS-Security etc., sind stark vom jeweiligen Schutzbedarf der Anwendung abhängig.

Dagegen werden die **nicht-funktionalen Sicherheitsanforderungen** häufig vernachlässigt. Sie stellen keine konkrete Funktion der Anwendung dar, sondern beziehen sich vielmehr auf die Softwarequalität bzw. den Grad der erforderlichen Software Assurance. Nicht-funktionale Sicherheitsanforderungen sind schwer zu testen, können aber anhand geeigneter Kennzahlen gemessen werden. Häufiges Beispiel einer nicht-funktionalen Anforderung ist, eine Anwendung robust gegenüber bestimmten Angriffen zu machen. Nicht-funktionale Sicherheitsanforderungen decken in den meisten Fällen daher Bereiche ab, die in Anwendungen jeglicher Schutzbedarfsklasse umgesetzt werden sollten.

Die hier erstellte Liste von funktionalen und nicht-funktionalen (Sicherheits-)Anforderungen wird im Rahmen weiterer Aktivitäten in der nächsten Phase aktualisiert und an die konkreten Eigenschaften der Anwendung angepasst. Dazu gehören:

- 4.3.5 KOP 5: Erstellung von Abuse Cases
- 4.3.6 KOP 6: Bedrohungsmodellierung
- 4.3.7 KOP 7: Erstellung der Sicherheitsarchitektur

4.2.3 IPV 3: Erstellung einer Business Impact Analyse

Bereits im Rahmen der initialen Planung und Vergabephase sollten die möglichen Auswirkungen einer Kompromittierung oder des Verlustes der Schutzziele Integrität, Vertraulichkeit oder Verfügbarkeit auf das gesamte Unternehmen vom Auftraggeber ermittelt werden. Dies kann durch die Erstellung einer Business Impact Analyse erfolgen und sollte folgende Fragen beantworten:

- Wie hoch kann der maximale Schaden sein, wenn die durch die Anwendung verarbeiteten Daten durch Unbefugte ausgelesen oder manipuliert werden können?
- Wie hoch ist der maximale Schaden, wenn es einem Angreifer gelingt, Teile der Anwendung selbst zu manipulieren (temporär oder dauerhaft)?
- Wie hoch ist der maximale Schaden, wenn die Anwendung (oder bestimmte Funktionen) nicht oder nur eingeschränkt zur Verfügung stehen? Die Auswirkung kann hierbei auch in mehreren Abstufungen (z.B. eine Minute, eine halbe Stunde oder mehrere Stunden) unterschieden werden.

Die Business Impact Analyse soll den Stakeholdern des Projekts dabei helfen, auf einer höheren Ebene die Risiken und Kritikalitäten zu identifizieren sowie den Nutzen des Projektes darzulegen.[17]

4.2.4 IPV 4: Erstellung des Sicherheitsprojektplans

In dieser Aktivität werden unter anderem auf Basis von Schutzbedarf, gesetzlichen Vorgaben und Kundenanforderungen die für das Projekt erforderlichen Sicherheitsaktivitäten und -maßnahmen (z.B. Testing, Quality Gates etc.) definiert und ihre Aufwände ermittelt. Als Unterstützung können Checklisten oder Mappings von Anwendungseigenschaften dienen. Hier wird auch ein geeignetes Modell für den SDL definiert.

Wie in jedem Projekt ist die Aufwandsschätzung auch bei der Entwicklung von Webanwendungen von essentieller Bedeutung. Hier soll ein Überblick über die notwendigen Ressourcen geschaffen werden, die im Projektplan zu berücksichtigen sind. Die eingeplanten Ressourcen müssen während des Projekts zur Verfügung gestellt werden, um einen reibungslosen Ablauf zu ermöglichen.

Ressourcen können im Allgemeinen in folgende Kategorien unterteilt werden:

- Personelle Ressourcen (Arbeitsaufwand der beteiligten Personen, ggf. externe Ressourcen)
- Finanzielle Ressourcen (Investitionen, Schulungskosten etc.)
- Sachliche Ressourcen (Lizenzen, Einkauf von Appliances (WAFs etc.), Beschaffung von Tools, Arbeitsräume, Sachmittel etc.)

4.2.5 IPV 5: Definition einer Software Security Group (SSG)

Sicherheit ist ein Querschnittthema der Softwareorganisation. Das heißt, es betrifft alle Bereiche und sollte nicht nur von einzelnen Personen verfolgt werden. Im Rahmen der BSIMM- Studie wurde daher die Software Security Group (SSG) als wichtige organisatorische Einheit identifiziert, die maßgeblich zum Erfolg einer Sicherheitsinitiative beiträgt.

Sowohl innerhalb eines bestimmten Projektes, wie auch übergreifend, werden daher Personen aus mehreren Bereichen (Entwicklung, Betrieb, Projektmanagement, Qualitätssicherung und natürlich der Sicherheit selbst) in einem Team zusammengefasst, das über bestimmte Sicherheitsaktivitäten entscheidet. Je nach Organisationsgröße kann es sich hierbei um Mitarbeiter handeln, die sich vollständig diesem Bereich widmen und etwa auch Code Analysen durchführen und/oder Sicherheitskonzepte erstellen. Die BSIMM-Studie schätzt hierbei die Größe der SSG auf erfahrungsgemäß 1% der Größe der Entwicklungsabteilung.

Zumindest teilweise, und insbesondere bei kleineren Organisationen kann die SSG aber auch durch Mitarbeiter gebildet werden, die sich nur anteilig diesem Bereich widmen. Diese Personen dienen dann als Multiplikatoren, um Sicherheitsbelange und -entscheidungen in den einzelnen Teams (etwa der Entwicklung) zu adressieren und das Feedback von dort wiederum in der SSG zu diskutieren.

4.3 Phase 2: Konzeption & Planung

Die zweite Phase ist die Konzeptions- und Planungsphase. Das Ziel dieser Phase besteht in der Erstellung einer technischen Planung zur konkreten Umsetzung der Sicherheitsanforderungen (statisch und

17 Die Erstellung einer Business Impact Analyse gehört klassisch zu den Pflichten des Auftraggebers. Die Aufgabe wurde explizit in diesem Dokument aufgenommen, da sie für die Festlegung des Schutzbedarfs eine wichtige Rolle spielt. Sollte dem Auftraggeber an Erfahrung und Know-How fehlen, um die Analyse selbst durchzuführen, kann der Auftragnehmer eine unterstützende Rolle übernehmen.

dynamisch) im Projekt. Diese Phase setzt sich aus neuen Aktivitäten zusammen, wie in Abbildung 8 dargestellt.

An dieser Stelle wird auch zusätzlich auf die BSI Studie „ISi-S Sicheres Bereitstellen von Web-Angeboten“[18] verwiesen. Die Studie ist jedoch rein informativ und kann in dieser Phase unterstützend sein. Der Inhalt ist aber nicht verpflichtend.

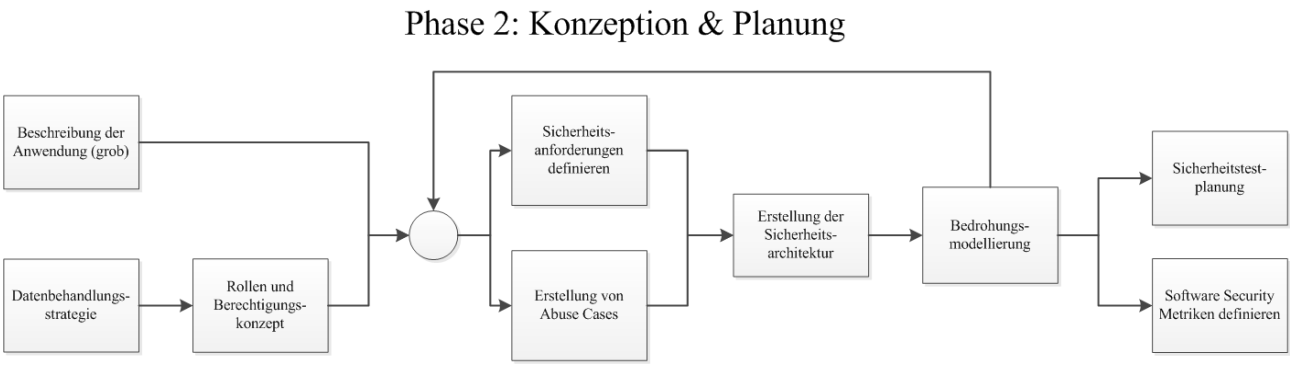

Abbildung 8: Aktivitäten der Phase 2 „Konzeption & Planung“

4.3.1 KOP 1: Beschreibung der Anwendung

Am Anfang der Konzept- und Planungsphase muss die Anwendung mit ihren Funktionen (Functional Decomposition), verarbeiteten Daten und Komponenten sowie allen gewünschten und nicht gewünschten Abläufen beschrieben werden. Der benötigte Detailgrad hängt stark von der Komplexität der zu erstellenden Software ab.

Diese erste, grobe Beschreibung der Anwendung sollte dabei helfen:

- Die Ziele und Wünsche der Stakeholder erkennen.
- Zu beschreiben, welcher Zweck die Anwendung erfüllen soll.
- Welche Funktionen führen zur Erfüllung der Ziele und der Kundenanforderungen?
- Welche Sicherheitsbedürfnisse sollten berücksichtigt werden. Diese Erkenntnisse helfen, die Risikobereiche der Anwendung zu identifizieren.

An dieser Stelle sei erwähnt, dass diese Aktivität das Ziel verfolgt, einen ersten Überblick über das *Was* und nicht über das *Wie* zu verschaffen. Die Ergebnisse werden mit den weiteren Aktivitäten dieser Phase ergänzt und fließen in die Sicherheitsarchitektur mit ein.

4.3.2 KOP 2: Datenbehandlungsstrategie

Die Datenbehandlungsstrategie definiert sehr konkrete Sicherheitsanforderungen für einzelne Datentypen. Sie legt fest, wie bestimmte Daten innerhalb der Anwendung behandelt (gespeichert, übertragen, dargestellt) werden dürfen und auf welche Weise dies geschehen darf (im Klartext, als Hash, Salted Hash, verschlüsselt, maskiert etc.).

Um eine Datenbehandlungsstrategie zu definieren, muss eine Datenklassifizierung durchgeführt werden. Hierbei werden Daten in Gruppen mit bestimmten Attributen aufgegliedert. Die Gruppen werden meist zu drei bis fünf Klassen (z.B. öffentlich, eingeschränkt, vertraulich) definiert. Basierend auf der Klassifikation

18 [ISiS2008]

wird entschieden, welcher Schutzbedarf angemessen ist. Entscheidend im Rahmen der Schutzbedarfsanalyse ist vor allem die Vertraulichkeit der Daten. Darüber hinaus unterliegt der Umgang mit bestimmten Daten (z.B. personenbezogene Daten) auch rechtlichen Regulierungen.

Idealerweise wird die Datenbehandlungsstrategie innerhalb des Projektes laufend gepflegt (z.B. als Tabelle in einem Wiki) und hinterfragt. So haben Entwickler stets die Gewissheit, wie sie bestimmte Funktionen in die Anwendung einbauen sollen.

4.3.3 KOP 3: Rollen- und Berechtigungskonzept

Für die Anwendung ist ein Rollen- und Berechtigungskonzept zu erstellen.[19] Dieses sollte die folgenden Informationen enthalten:

- **Erläuterung der vorgesehenen Rollen** (bei komplexeren Konzepten ist hier eine Gruppierung, etwa in verschiedene Privilegierungsstufen hilfreich)
- **Erläuterung zu den vorgesehenen Berechtigunge**n (z.B. „lesen“, „schreiben“, „Artikel erstellen“, „Benutzer anlegen“ etc.)
- Eine **Access Control Matrix**, über welche anschaulich die Berechtigungen der existierenden Rollen auf bestimmte Assets dargestellt werden.
- Annahmen bzgl. **Rollen- und Funktionstrennung** (Segregation of Duties, SOD)[20], sowohl statisch als auch dynamisch.

4.3.4 KOP 4: Sicherheitsanforderungen überpüfen und ggf. ergänzen

Die Sicherheitsanforderungen sind das Ergebnis des Security Requirements Engineering Process. Das Konzept des Requirements Engineering zielt darauf, – siehe Kapitel 4.2.2 sowie Kapitel 4.3.1 - funktionale und nicht-funktionale Anforderungen aus den Wünschen und den Zielen des Auftraggebers abzuleiten.

In diesem Abschnitt sollten die in der Planungsphase identifizierten Sicherheitsanforderungen (IPV1 und IPV2) nochmals überprüft und ggf. ergänzt werden. Dieser Schritt ist deswegen von großer Bedeutung, da fehlende, unvollständige oder missverständliche Sicherheitsanforderungen zu Schwachstellen in der Anwendung führen können.

Es exisiteren mehrere Vorgehensweisen, um Sicherheitsanforderungen zu erarbeiten. Eine davon ist die Security Quality Requirements Engineering (SQUARE) Methodologie[21], weitere sind z. B. in [DACS 2008] aufgelistet. Es spielt eine untergeordnete Rolle, welche Methodologie angewendet wird, es kommt viel mehr darauf an, dass ein systematisches Vorgehen gewährleistet ist, sodass für alle – vor allem kritische – relevante Komponenten die richtigen[22] Sicherheitsanforderungen abgeleitet werden können.

Die identifizierten Sicherheitsanforderungen werden in der Folge für die Bedrohungsmodellierung verwendet. Um einen möglichst hohen Abdeckungsgrad zu erreichen, müssen die Schritte KOP 4: Sicherheitsanforderungen überpüfen und ggf. ergänzen und KOP 6: Bedrohungsmodellierung mehrmals ausgeführt werden.

19 Das Rollen und Berechtigungskonzept wird nicht nur für die Benutzer, sondern auch für alle Systeme, die mit der Anwendung interagieren, erstellt.

20 Siehe Abschnitt 5.1.5 Segregation of Duties (Funktionstrennung)

21 Die SQUARE Methodologie wurde am Carnegie Mellon Software Engineering Institute entwickelt. Das Ziel dieser Methode ist die systematische Entwicklung von Sicherheitsanforderungen während der Softwareentwicklung. Die SQUARE Methodologie ist in [SQUARE 2005] beschrieben.

22 Mit „Richtige“ Sicherheitsanforderungen ist gemeint, dass eine Organisation – sei sie eine Behörde oder ein Unternehmen – aufgrund ihrer begrenzten Ressourcen evtl. nicht alle Sicherheitsanforderungen implementieren kann. Bei der Entscheidung sollte die Risikostrategie des Auftraggebers ebenfalls berücksichtigt werden.

4.3.5 KOP 5: Erstellung von Abuse Cases

Eine beliebte Methode zur Identifizierung und Darstellung von Anforderungen bei der Entwicklung von Software stellen die Use Cases dar. Use Cases sind abstrakte Beschreibungen der Interaktion zwischen einem System (z.B. einer Anwendung) und seiner Umgebung (z.B. einem Benutzer, einer anderen Anwendung). Demnach charakterisiert ein Use Case eine Verwendungsart des Systems bzw. einen Dialog zwischen dem System und seiner Umgebung.

Ein Use Case beschreibt also eine Funktion, die das System leisten soll. Der Inhalt des Use Cases wird in einer Use Case Description festgehalten. Use Cases bieten einen guten Gesamtüberblick über das System. Dieser kann vor Entwicklungsbeginn zur Strukturierung und Ermittlung der Anforderungen genutzt werden. Basierend darauf können die Anforderungen auch während des Projektes weiterentwickelt werden. Bezieht sich ein solcher Use Case auf Sicherheitseigenschaften, so können wir von einem Security Use Case sprechen.

Für jedes funktionales Use Case sollte überlegt werden, wie dessen Funktionalität absichtlich missbraucht werden kann. Das Ergebnis wird in Form eines Abuse Case[23] festgehalten. Ein Abuse Case[24] ist also eine Funktion, die das System nicht erlauben darf. Der Inhalt des Abuse Cases wird in einer Abuse Case Description festgehalten, was beispielsweise in tabellarischer Form erfolgen kann. In jeder Iteration können Sicherheitsanforderungen aus einem Security Use Case/Abuse Case auf Aktualität und Angemessenheit überprüft werden.

Security Use Cases / Abuse Cases tragen somit zu einem besseren Verständnis des Risikos der zu entwickelnden Webanwendung bei.

4.3.6 KOP 6: Bedrohungsmodellierung

Ein Modell bildet Teile der Realwelt ab, um die Komplexität für einen subjektiven Zweck zu reduzieren. Ist das Ziel, Bedrohungen frühzeitig zu erkennen und durch geeignete Maßnahmen zu vermeiden bzw. die Auswirkungen auf ein tragbares Niveau zu reduzieren, so spricht man von Bedrohungsmodellierung.

Mit der Bedrohungsmodellierung sollte so früh wie möglich – idealerweise schon während der Konzeptionsphase – begonnen werden. Dadurch wird sichergestellt, dass Schwachstellen frühzeitig und somit kosteneffektiv identifiziert und behoben werden können.

Ausgangspunkt für die Bedrohungsmodellierung sind die in diesem Kapitel bereits angesprochenen Aktivitäten bzw. deren Ergebnisse:

- Die Beschreibung der Anwendung sowie
- die identifizierten Assets,
- die Ergebnisse der Datenflussanalyse,
- identifizierte Sicherheitsanforderungen,
- Schnittstellen zu anderen Komponenten sowie
- die Komponenten selbst,
- Zugangspunkte und
- Abhängigkeiten sowie
- die auf dieser Basis erstellten Use und Abuse Cases.

23 Neben Abuse Case exisitieren weitere Begriffe wie z.B. Misuse Case, die oft als Synonym verwendet werden. Für eine bessere Lesbarkeit wird im weiteren Verlauf dieses Dokuments nur der eine Begriff verwendet.

24 [Mcgr 2004]

In der wichtigsten, aber auch schwierigsten, Phase der Bedrohungsmodellierung gilt es, für die wichtigen Assets die Bedrohungen möglichst vollständig zu identifizieren. Diese Aufgabe erfordert viel Zeit und fundiertes Fachwissen, sodass ein Heranziehen von Experten – darunter auch Software-Architekten, Entwickler und Tester - empfohlen wird. Hilfreich an dieser Stelle sind z. B.:

- OWASP Top 10[25], die sowohl die häufigsten Risiken für Webanwendungen als auch Lösungen beschreibt,
- das Common Vulnerability Scoring System[26], das eine standardisierte Methode zur Bewertung von Schwachstellen anbietet,
- die Web-Hacking-Incident-Database[27] und Threat Classification Projekte des Web Application Security Consortium,
- die Common Attack Pattern Enumeration and Classification (CAPEC)[28] Datenbank, die zahlreiche Angriffsmuster (Attack Patterns) enthält.

Alle identifizierten Bedrohungen sollten in einem weiteren Schritt dokumentiert und priorisiert werden. Dies ist deswegen von Bedeutung, weil der großen Anzahl ermittelter Bedrohungen die beschränkten Ressourcen eines Unternehmens gegenüberstehen. Anhand dieser Ergebnisse können geeignete Maßnahmen und Mechanismen in der Sicherheitsarchitektur eingeplant werden.

Die beschriebenen Schritte der Bedrohungsanalyse werden iterativ durchgeführt, bis die verbliebenen Bedrohungen akzeptabel sind. Da es sich bei der Bedrohungsanalyse um einen Prozess handelt, sollten die Bedrohungen regelmäßig oder bei Bedarf, z. B. beim Bekanntwerden neuer Schwachstellen oder der Einführung neuer Funktionalitäten, überprüft und neu bewertet werden.

4.3.7 KOP 7: Erstellung der Sicherheitsarchitektur

In dieser Aktivität wird die Sicherheitsarchitektur der Anwendung definiert. Die Sicherheitsarchitektur beschreibt die grundlegenden Komponenten und deren Zusammenspiel innerhalb einer Software.

Innerhalb der Sicherheitsarchitektur werden die gewonnenen Erkenntnisse der Aktivitäten dieser Phase zu einem zusammenhängenden Gesamtkonzept gebündelt:

- Akteure (Benutzer, Administratoren etc.) und wie diese auf die Anwendung zugreifen
- Trust Boundaries[29]
- identifizierte Assets
  - zentrale Systemkomponenten
  - wie werden vertrauliche Daten übertragen (Datenflüsse)
  - wo werden vertrauliche Daten gespeichert und wie
  - wie erfolgen die zentralen Datenflüsse

25 Weitere Informationen über die Open web Application Security Project Top 10: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2013_Project

26 Weitere Informationen zu CVSS: http://www.first.org/cvss

27 Weitere Informationen zum Web Application Security Consortium Project: http://projects.webappsec.org/w/page/13246927/FrontPage

28 Weitere Informationen zu CAPEC: http://capec.mitre.org/

29 Trust Boundaries grenzen Prozesse, Systemkomponenten und andere Elemente mit unterschiedlichen Trust Levels ab. Es kann als eine Linie im Programm gesehen werden. Auf der einen Seite des Trust Boundary werden die Daten als vertrauenswürdig, auf der anderen Seite als nicht vertrauenswürdig eingestuft. Konkret lässt sich über eine Trust Boundary etwa ermitteln, an welcher Stelle der Anwendung Daten validiert oder verschlüsselt werden müssen.

- Anbindung externer Systeme
- Verwendung von architekturellen Sicherheitsmechanismen wie Authentisierung (von Benutzern und Systemkomponenten untereinander), Autorisierung (Policy Decision und Enforcement Points), Kryptographie (etwa PKI-Komponenten), Validierung etc.
- Sicherheitskomponenten, insbesondere wenn diese auf den Anwendungslayer wirken (z.B. WAFs)

Mögliche Probleme oder bekannte Schwächen lassen sich in der Sicherheitsarchitektur ebenfalls darstellen.

Die Sicherheitsarchitektur ist ein kritischer Punkt im Entwicklungsprozess. Einmal eingerichtet ist sie später nur mit hohem Aufwand änderbar. Aus ihr werden auch zahlreiche konkrete (zumeist funktionale) Sicherheitsanforderungen abgeleitet.

Das Vorhandensein einer Sicherheitsarchitektur kann zudem die Aufwände für eine Bedrohungsmodellierung deutlich verringern. Sie erfüllt dabei große Teile der Phase der Application Decomposition.

4.3.8 KOP 8: Sicherheitstestplanung

Durch diese Aktivität werden die später im Projektverlauf durchzuführenden Tests (wie in Phase 4 näher beschrieben) geplant. Dazu ist im Vorfeld wichtig zu klären:

- Welche Anwendungsteile werden zu welchem Zeitpunkt in welcher Tiefe analysiert?
- Bei einer agilen Entwicklung (etwa auf Basis von Scrum) sollte die Sicherheitsplanung für einen Sprint erfolgen, da sich Anforderungen bei agilen Vorgehensweisen häufig ändern können.
- Auf welchem System (idealerweise sollte zu diesem Zeitpunkt ein bestimmtes Testsystem exklusiv für Sicherheitstests zur Verfügung stehen) werden die Tests durchgeführt?
- Durch welche Person bzw. welches Unternehmen werden die Tests ausgeführt? Hier sollte nach Möglichkeit bereits frühzeitig die Verfügbarkeit geklärt werden.
- Sind hierfür bestimmte Tools oder Lizenzen zu beschaffen. Ggf. ist auch eine frühzeitige Toolevaluierung erforderlich.
- Werden bestimmte Security Pushes (siehe Kapitel 4.4.4) innerhalb der Entwicklung vorgesehen.

Die angesprochene Untersuchungstiefe abzuschätzen ist dabei elementar. Damit ist konkret der bereits oben angesprochene Grad an Software Assurance gemeint, der sich häufig auch für bestimmte Teilkomponenten unterscheidet. Hierbei ist zu klären, ob es ausreichend ist einzelne Anwendungsteile einem Penetrationstest zu unterziehen, oder ob zusätzlich auch ein Codereview oder toolgestützte Tests erfolgen sollen. Ähnlich geschieht dies im Rahmen des Application Security Verification Standards[30] (ASVS) von OWASP. Dieser definiert hierzu verschiedene Prüfstufen (Verification Level).

Wichtig ist dabei, dass sich der Grad der erforderlichen Software Assurance einzig auf den Schutzbedarf der Anwendung (bzw. deren Assets/Daten) bezieht, nicht jedoch auf das eingeplante Budget.

Hilfreich ist es in diesem Zusammenhang auch bestimmte sicherheitsrelevante Teilfunktionen (wie etwa die Benutzerverwaltung, Authentisierung oder kryptographische Aspekte) möglichst frühzeitig zu implementieren (etwa im Rahmen einer der ersten Sprints), so dass diese bereits einer Sicherheitsuntersuchung unterzogen werden können. Dabei sollte beachtet werden, dass eine einmal getestete Komponente in der Regel einem Regressionstest zu unterziehen ist, wenn an dieser eine nachträgliche Änderung erfolgt.

30 [ASVS]

4.3.9 KOP 9: Software Security Metriken

Metriken dienen im Qualitätsmanagement als wichtige quantitative Kennzahlen. Auch im Bereich der Softwaresicherheit lassen sich Metriken als Key Performance Indikatoren (KPIs) definieren und dadurch deren Umsetzungsgrad ermitteln.

Auch ermöglichen es Security Metriken das bereits angesprochene Problem der fehlenden Sichtbarkeit von Anwendungssicherheit zu verringern und so auch Entwicklungsteams darin zu motivieren, dieses Thema zu priorisieren.

Rafal Los[31] (Security Evangelist von HP) nennt hierzu einige Beispiele:

- **Defect Remediation Window (DRW)**: Die Dauer zwischen Identifikation einer Schwachstelle und deren Verifikation bzw. Korrektur.
- **Rate of Defect Recurrence (RDR)**: Wie häufig werden geschlossene Schwachstellen wieder eingeführt.
- **Specific Coverage Metric (SCM)**: Die Abdeckung, die durch das Security Testing erzielt wird.
- **Security to Quality Defects (SQD)**: Das Verhältnis von sicherheitsrelevanten Defects zur gesamten Anzahl identifizierter Defects.

Etwas problematisch ist dagegen der Vergleich der Anzahl identifizierter Schwachstellen, da diese mitunter von der Anzahl existierender Schwachstellen deutlich abweichen kann. Für die Ermittlung eines allgemeinen Trends, unter Berücksichtigung des jeweils angewendeten Testverfahrens, kann dies möglicherweise jedoch sinnvoll sein.

4.4 Phase 3: Implementierung

Die dritte Phase ist die Implementierungsphase. Sie besteht aus vier Aktivitäten, wie in Abbildung 9 dargestellt. Durch diese Phase soll sichergestellt werden, dass innerhalb der konkreten Implementierung Sicherheit im erforderlichen Rahmen berücksichtigt wird. Grundlegende Vorgaben an die Implementierung werden noch zusätzlich ausführlich im Kapitel 5 beschrieben.

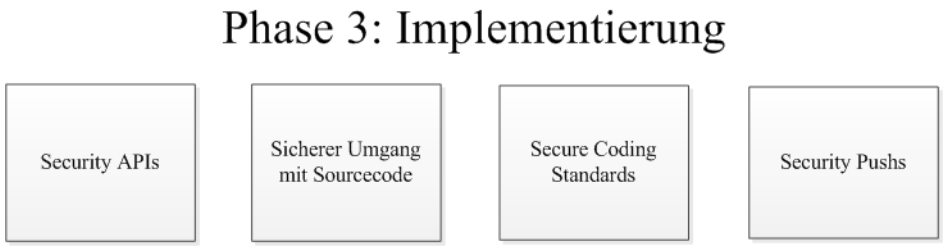

Abbildung 9: Aktivitäten der Phase 3 „Implementierung“

4.4.1 IMP 1: Security APIs

Sicherheitsfunktionen, wie etwa solche zur Datenvalidierung oder Kryptographie, stellen die Basis einer sicheren Anwendungsimplementierung dar. Wie in jeder Softwareentwicklung können natürlich auch hier Fehler gemacht werden, die sich in der Folge fatal auf die Sicherheit der gesamten Anwendung auswirken können.

Daher ist es elementar wichtig, an dieser Stelle von der eigenen Implementierung von Sicherheitsfunktionen nach Möglichkeit abzusehen und hier stattdessen ausgiebig geprüfte APIs

31 [Rafa 2010]

einzubinden. Durch die OWASP ESAPI existiert hier zudem eine für mehrere Sprachen verfügbare API, die zahlreiche Sicherheitsfunktionen abbildet.

Security APIs sollen über entsprechende Prozesse und zentrale Repositories abgebildet und ausgewählt werden. Die hierbei verwendeten Bibliotheken müssen einer ständigen Überwachung unterzogen werden. Es muss ein Prozess definiert werden, wie im Falle einer bekannt gewordenen Schwachstelle in einer Library reagiert wird und welche Schritte zu unternehmen sind, um die Schwachstelle zu eliminieren.

4.4.2 IMP 2: Sicherer Umgang mit Sourcecode

Die Einhaltung der Schutzziele Vertraulichkeit, Integrität und Verfügbarkeit muss nicht nur bei der Erstellung von Code, sondern auch bei dessen Umgang sichergestellt werden. Dazu zählen unter anderem:

- Zugriffsregelung: nur berechtige Personen können Source Code modifizieren. Die Definition von Zugriffsregeln erfolgt dabei nach dem Least-Privilege-Prinzip. Wichtig ist an dieser Stelle, den Zugriff auf besonders schutzwürdigen Code (etwa Security APIs) sicherzustellen. Dies lässt sich auch über getrennte Code Repositories erreichen.
- Strenges Change Management (Änderungen am Programmcode nur nach definiertem Änderungsprozess)
- Systeme, die den Programmcode speichern oder verarbeiten, müssen hinsichtlich ihrer Sicherheit überprüft werden. Hier ist mindestens ein IT-Grundschutz sicherzustellen.
- Trennung von Entwicklungs-, Test- und Produktivumgebungen. Systeme dürfen nicht auf mehrere dieser Umgebungen zur gleichen Zeit Zugriff haben.
- Zusätzlich können Code-Signaturen eingesetzt werden, um die Authentizität des Codes zu gewährleisten, insbesondere wiederum von besonders schutzbedürftigem Code. Dies ist insbesondere dann vorteilhaft, wenn Code das Unternehmen verlässt und in anderen Umgebungen zum Einsatz kommt.

4.4.3 IMP 3: Secure Coding Standards

Das vielleicht wichtigste Gebot in Verbindung mit dem Thema Secure Coding lautet „Keep it small and simple“. Je einfacher und schlanker der Code ist, umso einfacher gestaltet sich das Testen und das Beseitigen von Fehlern, was wiederum zur Verbesserung der Sicherheit von Software führt.[32]

Standards zur sicheren Codeerstellung bieten verpflichtende Vorgaben für sichere Entwicklung. Vorhandene Standards müssen dabei gesichtet und in die Entwicklung einbezogen werden. Die Secure Coding Standards sollten Vorgaben hinsichtlich der folgenden Bereiche umfassen:

- Authentifizierung und Session Management
- Zugriffskontrolle
- Datenvalidierung (Ein- und Ausgabe)
- Protokollierung
- Fehlerbehandlung etc.
- Datensicherheit und Kryptographie

Es empfiehlt sich dabei, diese technologieunabhängig zu erstellen und daraus technologie-spezifische Guidelines abzuleiten. Insbesondere für letztere empfiehlt sich die aktive Einbeziehung der Entwicklung.

32 [DACS 2008]

In Kapitel 5 (Vorgaben an die Implementierung) werden entsprechende Vorgaben definiert, die als Grundlage für einen solchen Secure Coding Standard dienen können.

4.4.4 IMP 4: Security Pushes

Ein Security Push ist ein Vorgehen, welches von Microsoft eingeführt wurde[33]. Ein Security Push dient dazu, das Thema Sicherheit in bestimmten Phasen der Entwicklung zu priorisieren. Dabei konzentriert sich das gesamte Team, etwa innerhalb eines gesamten Sprints, auf die Verbesserung der Sicherheit der Webanwendung.

Ein Security Push sollte daher idealerweise im Rahmen der Testplanung (siehe Kapitel Fehler: Referenz nicht gefunden) entsprechend berücksichtigt werden.

4.5 Phase 4: Testen

Die vierte Phase ist die Testphase. Diese Phase besteht aus drei Aktivitäten, wie in Abbildung 10 dargestellt. Das Ziel dieser Phase besteht in der Überprüfung der definierten Anforderungen in den vorigen Phasen sowie die Überprüfung der Anwendung auf Fehlverhalten. Der Auftragnehmer muss dem Auftraggeber (nach Absprache) den Zugriff auf Testberichte bzw. auf den Quellcode ermöglichen.

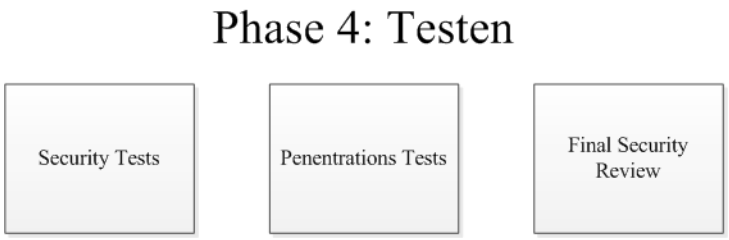

Abbildung 10: Aktivitäten der Phase 4 "Testen"

4.5.1 TES 1: Security Tests

Bereits im Rahmen der Entwicklung und in jedem Fall vor der Auslieferung ist die Webanwendung hinsichtlich Sicherheitsmängel zu testen. Dies erfolgt im Zuge eines oder mehrerer Security Assessments. Ein Security Assessment soll dabei sicherstellen, dass die definierten Sicherheitsanforderungen in der Implementierungsphase korrekt umgesetzt worden sind. Zusätzlich soll in diesem Rahmen auch die Effektivität der gesetzten Sicherheitsanforderungen überprüft werden und so mögliche Unzulänglichkeiten in der Spezifikation selbst aufgedeckt werden. Das heißt, es soll sichergestellt werden, dass die spezifizierten Sicherheitsanforderungen auch angemessen und wirksam sind.

Üblicherweise durchlaufen Security Assessments die folgenden Phasen:

- **Planung**: Ein Security Assessment benötigt eine initiale Planung. Hier werden sämtliche Informationen, die für die Durchführung des Assessments notwendig sind, zusammengetragen.
- **Durchführung**: Die Hauptaufgabe dieser Phase ist das Suchen nach Schwachstellen und deren Bewertung.
- **Auswertung**: Hierbei werden die gefundenen Schwachstellen analysiert. Es werden die Ursachen und die Gegenmaßnahmen bestimmt und in einem Abschlussbericht zusammengefasst. Der Abschlussbericht wird an die Entwickler geleitet, die die identifizierten Schwachstellen beheben. Nach der Behebung werden die Schwachstellen erneut untersucht.

33 [Micr 2005]

Die Überprüfung kann unter Verwendung mehrerer verschiedener Testverfahren erfolgen. Dabei ist zu berücksichtigen, dass unterschiedliche Testverfahren unterschiedliche Aufgaben erfüllen und unterschiedliche Ergebnisse liefern. Beispielsweise überprüfen Unit Tests die Korrektheit und die vollständige Abdeckung einer Funktionalität, während ein Code Review die korrekte Implementierung überprüft. Je nachdem was genau überprüft werden soll, können auch unterschiedliche Testrollen herangezogen werden. Ein Auditor kann beispielsweise die Einhaltung von Vorgaben überprüfen, während ein Penetrationstester die korrekte Implementierung überprüft.

Folgende Testverfahren können bereits während der Implementierung durchgeführt werden (Penetrationstests finden in der Regel nach der Fertigstellung der Webanwendung statt, weshalb diese in einer eigenen Aktivität ausgegliedert wurden):

- **Security Test Cases**
  Ein Test Case beschreibt einen Softwaretest, in dem die laut den Sicherheitsanforderungen bestimmte spezifizierte Funktionalität (funktional oder nicht funktional) der Webanwendung auf ihre Korrektheit überprüft wird. Test Cases einer Webanwendung werden hierbei in einfacher Sprache beschrieben.

  Ein Test Case besteht immer zumindest aus einer eindeutigen ID, einer Beschreibung und einem erwarteten Ergebnis.

  Beispiel:
  ID: XY
  Beschreibung: Zur Authentifizierung das Passwort an Webserver übertragen.
  Erwartetes Ergebnis: Passwort muss verschlüsselt an den Webserver übertragen werden und ist auf der Übertragungsstrecke nicht lesbar.

  Im Zuge des Tests Cases muss die Einhaltung des erwarteten Ergebnisses überprüft werden.

  Test Cases lassen sich in folgende Arten unterteilen:

  – Positivtest: Das Verhalten der Webanwendung wird mit gültigen Rahmenbedingungen und Eingaben überprüft.

  – Negativtest: Das Verhalten der Webanwendung wird mit ungültigen Rahmenbedingungen und Eingaben überprüft.

  Test Cases sollten priorisiert werden. Jedem Test Case wird ein je nach Kritikalität der zugehörigen Funktion ein Prioritätslevel (z.B. Normal, Hoch, Kritisch) zugewiesen und beim Testen berücksichtigt.

  Test Cases müssen im Rahmen der Entwicklung durch den Entwickler erstellt werden.

- **Security Unit Tests**
  Unit Tests überprüfen Softwareeinheiten (Units) einer Webanwendung auf korrekte Funktionalität. In der Regel sind das Funktionen, Methoden oder Klassen. Units werden mit verschiedenen Parametern aufgerufen und es wird überprüft, ob die Ausgabe mit den Erwartungen übereinstimmt.

  Unit Tests müssen im Rahmen der Entwicklung durch die Entwickler erstellt und gepflegt werden.

- **Design Review**
  Bei Design Reviews geht es darum, die Architektur der Anwendung auf hoher Ebene zu untersuchen. Dabei sollen konzeptionelle Fehler und Verwundbarkeiten aufgedeckt werden und die Umsetzung der Sicherheitsanforderungen und -mechanismen auf ihre Vollständigkeit hin untersucht werden. Wenn erhebliche Mängel im Design erkannt werden, muss die Phase 2 „Konzeption & Planung“ überarbeitet werden. Die im Design Review erkannten Mängel dienen hierbei als Ansatzpunkt.

- **Code Reviews**
  Beim Code Review wird der Programmcode manuell überprüft. Hierbei ist zu beachten, dass die Durchsicht durch eine andere Person erfolgt (etwa innerhalb eines Peer Reviews) als derjenigen, die den Programmcode verfasst hat, da ansonsten Fehler leicht übersehen werden können. Typische Fehler, die entdeckt werden können, sind Abweichungen von Standards, Abweichungen gegenüber Anforderungen, Fehler im Design, Buffer Overflows etc. In erster Linie ist das bei sensitiven Funktionen wie beispielsweise Transaktionsmanagement, Authentisierung und Kryptographie wichtig. Die Ergebnisse des Code Reviews fließen zurück in den Implementierungsprozess (Phase 3: Implementierung).
- **Statische Code Scanner**
  Mittels Statischen Code Scannern (SCA) können eine Vielzahl von Schwachstellen bereits innerhalb der Entwicklung identifiziert und so zeitnah behandelt werden. In der Praxis ist die Auswahl entsprechender Tools zur Durchführung von Sicherheitsscans sehr beschränkt. Weiterhin müssen diese Tools die eingesetzten Technologien unterstützen. Analysieren sie den Sourcecode, müssen diese zusätzlich die eingesetzten APIs kennen, da diese gewöhnlich nicht im Sourcecode vorliegen. Schließlich erzeugen SCA-Tools häufig eine sehr hohe Anzahl an False Positives, weshalb eine Verifikation der Scanergebnisse durch fachmännisches Personal erforderlich ist.
- **Web Security Scanner**
  Mit einem Web Security Scanner wird eine Webanwendung automatisiert nach Schwachstellen untersucht. Web Security Scanner können einfache Schwachstellen (Low Hanging Fruits) erkennen. Scanner sind jedoch nicht perfekt und können nicht sämtliche Schwachstellen abdecken. Wie SCA-Tools können auch diese nur solche Schwachstellen identifizieren, welche in ihrer Datenbank vorher spezifiziert wurden. Genauso wenig können sie logische Fehler in der Webanwendung identifizieren. Daher können viele Fehler durch solche Tools nicht erkannt werden. Allerdings bietet sich dies dort an, wo kontinuierliche Tests nach bestimmten, bekannten Schwachstellen durchgeführt werden sollen. Auch innerhalb der Entwicklung ermöglichen es diese (häufig kostenlosen) Tools einfache Schwachstellen frühzeitig zu erkennen.
- **Fuzz Testing**
  Fuzz Testing ist ein Testverfahren, welches automatisiert oder teilweise automatisiert durchgeführt wird. Dabei werden ungültige, unerwartete oder zufällige Werte generiert und als Eingabe an die Anwendung weitergegeben. Das Verhalten der Anwendung wird während des Tests auf Fehlverhalten überwacht. Das Prinzip ist simpel und führt dazu, dass Fehler entdeckt werden, die oft übersehen worden wären. Fuzz Testing kann nur einfache Programmfehler entdecken, die auf falsche Eingabewerte zurückzuführen sind, jedoch keine Designfehler etc.

4.5.2 TES 2: Penetration Tests

Penetration Tests stellen Sicherheitstests aus der Sicht eines Angreifers dar. Hierbei wird versucht, Schwachstellen innerhalb der Webanwendung zu identifizieren und zu nutzen, um eine Verletzung eines Schutzzieles herbeizuführen.

Üblicherweise erfolgen Penetration Tests folgendermaßen:

- Informationssammlung über das Zielsystem
- Scan nach angebotenen Diensten
- Identifikation des Systems und der Anwendungen
- Recherche nach Schwachstellen
- Ausnutzen der Schwachstellen

Penetration Tests können basierend auf unterschiedlichen Kriterien klassifiziert werden:

- Informationsbasis: Hier wird festgelegt, von welchem Wissensstand der Tester ausgeht.
- Aggressivität: Hier wird festgelegt, wie aggressiv der Angreifer vorgeht. z.B. passiv (gefundene Schwachstellen werden nicht ausgenutzt) oder aggressiv (gefundene Schwachstellen werden unabhängig von den Konsequenzen ausgenutzt).
- Umfang: Hier wird festgelegt, welche Systeme getestet werden.
- Vorgehensweise: Der Angreifer kann verdeckt (Penetration wird nicht direkt als Angriff erkannt) oder offensichtlich (Penetration erfolgt offensichtlich) vorgehen.
- Technik: Es muss definiert werden, welche Techniken eingesetzt werden. Erfolgt der Angriff z.B. nur über das Netzwerk oder auch über weitere Kommunikationsnetze (Telefon, physischer Zugriff etc.)?
- Ausgangspunkt: Hier wird festgelegt, ob der Penetrationstest von innen oder von außen erfolgen soll.

Penetration Tests können nicht nur auf technische Systeme, sondern auch auf die organisatorische und personelle Infrastruktur erfolgen. Angriffe auf personelle Infrastrukturen können beispielsweise mittels Social Engineering erfolgen. Angriffe auf organisatorische Infrastruktur können beispielsweise die physische Sicherheit betreffen.

Unabhängig von der Vorgehensweise haben Penetration Tests folgende Ziele:

- Erhöhung der Sicherheit der IT-Systeme
- Identifikation von Schwachstellen
- Bestätigung der Sicherheit der IT-Systeme von unabhängigen Dritten
- Erhöhung der Sicherheit der organisatorischen und personellen Infrastruktur

In jedem Fall liefern Penetrationstests Security Gaps zwischen Planung und Implementierung. Die getroffenen Maßnahmen, um die Lücken zu schließen, müssen in einem weiteren Schritt nicht nur auf ihre korrekte Umsetzung überprüft werden, sondern auf ihre Angemessenheit, wie effektiv sie das Risiko tatsächlich senken beziehungsweise beseitigen oder ob durch sie ein falsches Sicherheitsgefühl vermittelt wird. Üblicherweise werden Penetrationstests erst dann durchgeführt, wenn die Webanwendung mit allen Komponenten fertiggestellt wird.

Weitere Details können der BSI- Studie[34] „Durchführungskonzept für Penetrationstests“ entnommen werden.

4.5.3 TES 3: Final Security Review (FSR)

Der Final Security Review wird auf die fertiggestellte Anwendung durchgeführt. Hierbei müssen die eventuell noch nicht erfüllten Anforderungen evaluiert, dokumentiert und bewertet werden. Bei positivem Abschluss (keine offenen Anforderungen) wird eine Freigabeerlaubnis der Anwendung erteilt.

4.6 Phase 5: Auslieferung & Betrieb

Die fünfte Phase ist die Auslieferungs- und Betriebsphase. Diese Phase setzt sich aus neun Aktivitäten zusammen, wie in Abbildung 11 dargestellt. Das Ziel ist die Gewährleistung, dass die Anwendung mit allen notwendigen Informationen an den Betrieb übergeben und dort basierend auf den definierten Rahmenbedingungen in Betrieb genommen werden kann.

34[PenT2003]

Abbildung 11: Aktivitäten der Phase 5 "Auslieferung und Betrieb"

4.6.1 AUB 1: Sichere Auslieferung der Software

Es müssen entsprechende Rahmenbedingungen festgelegt werden, wie die Auslieferung der fertigen Webanwendung zu erfolgen hat bzw. welche Sicherheitsmechanismen dabei zu befolgen sind. Wird eine Webanwendung beispielsweise per Download zur Verfügung gestellt, müssen entsprechende Sicherheitsmechanismen realisiert werden. Dazu gehören:

- Prüfsummen zum Abgleich der Webanwendung zur Wahrung der Integrität
- digitale Signatur der Pakete zur Wahrung der Authentizität
- geeignete Verschlüsselungsmechanismen zur Wahrung der Vertraulichkeit

Sämtliche Testdaten sind vor der Auslieferung zu entfernen. Darüber hinaus ist sicherzustellen, dass die fertige Webanwendung mit allen Softwarekomponenten inkl. sämtlicher Abhängigkeiten und aller Informationen zum Übersetzen übergeben wird.

4.6.2 AUB 2: Sicherheitsdokumentation

Die Sicherheitsdokumentation beinhaltet sämtliche sicherheitsrelevante Informationen hinsichtlich der Webanwendung. Die Sicherheitsdokumentation sollte darüber hinaus den Umgang mit Sicherheitsproblemen klar darstellen. Die Sicherheitsprobleme helfen beim Verständnis, wie die Anwendung optimal konfiguriert und betrieben werden soll. Die Sicherheitsdokumentation soll mindestens folgende Punkte abdecken, wobei jeweils eine Dokumentation für Anwender und für den Betrieb erstellt werden soll:

- Die Dokumentation für Anwender beinhaltet
  - sämtliche sicherheitsrelevante Einstellungen mit den jeweiligen Defaultwerten (sofern relevant, etwa bei Clientanwendungen)
  - alle Systemmeldungen mit zugehörigen Ursachen und Behebungsmaßnahmen (sofern relevant, etwa bei Clientanwendungen)
  - eine Erläuterung der sicherheitsspezifischen Anwender-Funktionen (z.B. Passwort-Reset)
  - die Grundregeln für die sichere Verwendung
- Die Dokumentation für den Betrieb beinhaltet

- das Sicherheitskonzept (welches etwa das Bedrohungsmodell sowie die Sicherheitsarchitektur enthält)
- alle Anforderungen an die Umgebung (Netzwerk, Infrastruktur, Betriebssystem, Datenbanken, Sicherheitsmechanismen etc.)
- sämtliche sicherheitsrelevanten Einstellungen mit den jeweiligen Standardwerten
- alle Anwendungsfehler mit zugehörigen Ursachen und Behebungsmaßnahmen
- sämtliche sicherheitsspezifischen Funktionen
- die Grundregeln für den sicheren Betrieb
- Hinweise auf die Konsequenzen von sicherheitskritischer Konfigurationen
- sicherheitsrelevante Aspekte, die im Zuge der Wartungsarbeiten zu beachten sind.

4.6.3 AUB 3: Plattformhärtung

Unter Plattformhärtung versteht man die Entfernung aller Softwarebestandteile und Funktionen der Serversysteme, die für die Aufgabenerfüllung nicht erforderlich sind. Das Serversystem wird restriktiv auf das absolut Notwendige (Minimalprinzip) konfiguriert, um mögliche Angriffspfade zu reduzieren.

Folgende Punkte sind hierbei unter anderem zu berücksichtigen:

- Beschränkung der Serverdienste,
- Beschränkung der Berechtigungen auf das Dateisystem (Least Privilege),
- Serverdienste laufen nur mit minimal erforderlichen Rechten,
- Standardmäßig konfigurierte Accounts müssen deaktiviert, gelöscht oder das Passwort geändert werden.

4.6.4 AUB 4: Security Change Management

Das Ziel des Security Change Managements ist es, Änderungen an der Webanwendung kontrolliert vorzunehmen (insbesondere sollen Änderungen kein Sicherheitsproblem für die Webanwendung darstellen). Die Änderungen müssen nach dem Ausmaß bewertet und müssen eventuell einen Audit-Prozess durchlaufen, um freigegeben zu werden. Für eine Änderung muss eine Person verantwortlich sein, die die Auswirkungen der Änderung abschätzen kann.

Änderungen können in folgenden Kategorien unterteilt werden:

- **Triviale Änderung**: Die Änderung hat keine Auswirkung auf die Funktionalität der Webanwendung. Beispiel: Korrektur von Rechtschreibfehlern, Änderung des optischen Designs.
- **Reguläre Änderung**: Alle Änderungen, die weder als trivial noch als wesentlich klassifiziert werden können, fallen unter diese Kategorie. Beispiel: Beseitigung von Programmfehlern, Hinzufügen von geringen, nicht sicherheitsrelevanten Funktionen.
- **Wesentliche Änderung**: Wesentliche Änderungen lassen sind in folgende Unterkategorien unterteilen:
  - Änderung der Architektur oder Struktur der Webanwendung. Beispiel: Die Webanwendung wird auf ein neues Framework umgestellt. Weite Teile der Webanwendung sind dadurch betroffen.
  - Eine grundlegend neue Funktionalität wird hinzugefügt. Beispiel: Ein Dateiupload wird dem Benutzer zur Verfügung gestellt.

  - Änderung einer sicherheitsrelevanten Funktionalität. Beispiel: Die Mechanismen zur Authentifizierung werden geändert.

Änderungen an der Anwendung müssen in einem definierten Prozess erfolgen. Darüber hinaus muss ein Audit definiert werden, wie die Überprüfung nach einer Änderung zu erfolgen hat.

Der Änderungsprozess durchläuft somit zumindest folgende Punkte:

- Klassifizierung der Änderung
- Umsetzung der Änderung
- Überprüfung der Änderung

4.6.5 AUB 5: Security Response

Alle Incidents müssen über ihren gesamten Lebenszyklus verwaltet werden. Unter einem Incident versteht man einen nicht geplanten Ausfall bzw. die Störung eines Services. Die Verwaltung von Incidents erfolgt über einen definierten Prozess und hat das Ziel, den Incident schnellstmöglich zu beheben und zum Ursprungszustand zurückzukehren.

Die Behebung eines Incidents durchläuft üblicherweise folgende Prozessphasen:

- Incident aufnehmen und dokumentieren: Ein Incident wird beispielsweise durch technische Hilfsmittel erkannt oder durch Benutzer gemeldet. Der Incident muss zwecks Nachvollziehbarkeit erfasst und dokumentiert werden.
- Klassifizierung und Priorisierung: Der Incident muss nach entsprechenden Vorgaben klassifiziert und priorisiert werden. Höher priorisierte Incidents werden zuerst behandelt.
- Untersuchung und Diagnose: Hier wird der Incident behandelt. Mitarbeiter sammeln alle Informationen zu einem Incident und analysieren alle dazugehörigen Informationen.
- Übergabe des Prozesses an das Change Management: Im Zuge des Security Change Managements wird eine Lösung oder ein Workaround zur Verfügung gestellt und umgesetzt.
- Incident abschließen: Rückmeldung an den Initiator des Incidents (z.B. an Benutzer), dass der Incident behoben ist. Der Incident kann anschließend geschlossen werden.

Der Incident Prozess wird im Software Security Incident Response Plan (SSIRP) dokumentiert. Hierbei kann auch beispielsweise zusätzlich definiert werden, in welchem Zeitraum bestimmte Incidents korrigiert werden müssen.

4.6.6 AUB 6: 3rd Party Software Vulnerability Monitoring

Die kontinuierliche Beobachtung der verwendeten Third Party Komponenten ermöglicht die Behebung von Schwachstellen, bevor ein Angreifer diese ausnutzen kann. Diesbezüglich muss eine Liste aller Third Party Komponenten vorhanden sein. Insbesondere muss die Verantwortlichkeit delegiert werden, damit dieser Rolle eine oder mehrere Personen zugeordnet werden.

4.6.7 AUB 7: Sichere Standardkonfiguration

Die fertige Anwendung soll bereits mit sicherer Standardkonfiguration basierend auf den Sicherheitsanforderungen ausgeliefert werden. Darunter versteht man, dass die Webanwendung nach der Installation (ohne weiteres Zutun) sich in einer sicheren Standardkonfiguration befindet. Erforderliche Maßnahmen (z.B. Least Privilege, Minimalprinzip, keine Default Passwörter etc.) werden dokumentiert und wenn möglich auch technisch erzwungen.

4.6.8 AUB 8: Web Application Firewall

Web Application Firewalls (WAF) bieten, wenn entsprechend konfiguriert, einen zusätzlichen Schutz vor Angriffen auf die Anwendung. Eine WAF operiert auf der Anwendungsebene und kann eine Webanwendung vor Angriffen über das HTTP-Protokoll schützen.

Hierzu untersucht eine WAF alle eingehenden Anfragen und die ausgehenden Antworten des Webservers. Zu den üblichen Angriffen, die eine WAF verhindern kann, zählen Cross-Site-Scripting, SQL-Injection, Ausnutzen von bekannten Schwachstellen, Konfigurationsfehler etc. Der Schutz ist hier stets nur als zusätzlicher Schutzmechanismus zu verstehen und ersetzt in keinem Fall die Notwendigkeit, eine Webanwendung mit ausreichender Sicherheit zu entwickeln und auch zu testen. Im Rahmen von Tests sollte eine WAF dabei stets deaktiviert sein, damit diese nicht die Testergebnisse verfälscht.

Um den größtmöglichen Nutzen der WAF zu erhalten, muss deren Konfiguration durch fachmännisches Personal auf die Webanwendung angepasst werden.

Die durch Produkte in diesem Bereich angeboten Funktionalität ist teilweise sehr unterschiedlich. Auch für ein beabsichtigtes Deployment-Szenario (etwa innerhalb einer Cloud) muss ein Produkt geeignet sein. Das Best-Practice-Guide für den Einsatz von WAFs der OWASP[35] bietet hier einen guten Überblick über mögliche Auswahlkriterien.

4.6.9 AUB 9: Wartung

Nach der Inbetriebnahme der Webanwendung muss die Möglichkeit einer Wartung ermöglicht werden. Sämtliche Wartungstätigkeiten, insbesondere Änderungen an der Webanwendung und der Umgebung, sind zu dokumentieren. Änderungen müssen stets in der Art und Weise durchgeführt werden, dass keine Sicherheitsprobleme entstehen.

Es wird zwischen der Fern- und der Wartung vor Ort unterschieden:

- Fernwartung:

  Der Zugriff des Herstellers mittels Fernwartung darf nur über abgesicherte Systeme und robuster Authentisierung erfolgen.

  Diesbezüglich sollten mindestens folgende Punkte berücksichtigt werden:

  – Der Zugriff darf erst nach Freigabe des Betreibers erfolgen.
  – Der Zugriff muss personalisiert werden.
  – Der Zugriff muss stets über eine Firewall geführt werden.
  – Der Zugriff bzw. der Zugriffsversuch muss protokolliert werden.

- Wartung vor Ort:

  Die Wartung vor Ort darf nur nach Absprache und Zustimmung des Auftraggebers erfolgen.

  Diesbezüglich sollen mindestens folgende Punkte berücksichtigt werden:

  – Erfassung der Wartungsarbeiten mit Datum und Uhrzeit.
  – Erfassung der Personen, die die Wartungsarbeiten durchgeführt haben.

35[WAFs]

5 Vorgaben an die Implementierung

Dieses Kapitel beschreibt und erklärt die Vorgaben an die Implementierung. Hier werden die grundlegenden Prinzipien für den Entwurf sicherer Systeme beschrieben und die einzelnen technischen Teilbereiche der sicheren Entwicklung kurz erklärt. Die aus den Prinzipien in diesem Kapitel abgeleiteten konkreten Vorgaben der unterschiedlichen technischen Bereiche, wie beispielsweise Datenvalidierung oder Autorisierung, befinden sich im Kapitel 6.2 „Checkliste für technische Vorgaben an die Implementierung".

Diese technische Checkliste enthält Vorgaben an die Entwickler, welche bei der Implementierung zu berücksichtigen sind. Diese Vorgaben wurden hauptsächlich aus der OWASP ASVS [36] und zu Teilen aus der ÖNORM A7700 [37] abgeleitet. Sie sind bewusst technologieunabhängig gehalten, um sie technologieübergreifend verwenden zu können.

Die aufgelisteten Anforderungen stehen jeweils in Bezug zu der Schutzbedarfsklasse einer Webanwendung, welche in der Spalte „Schutzbedarf" wie folgt angegeben wird:

- „N": Ab normalem Schutzbedarf
- „H": Ab hohem Schutzbedarf
- „SH": Für sehr hohen Schutzbedarf

5.1 Designprinzipien für sichere Systeme

Im Folgenden werden grundlegende Designprinzipien erläutert, von denen sich viele Implementierungsvorgaben (siehe Kapitel 6.2) ableiten. Obwohl die konkreten technischen Anforderungen an den ermittelten Schutzbedarf gekoppelt sind, sind diese allgemeinen Designprinzipien bereits bei einem normalen Schutzbedarf zu beachten. Die Beachtung dieser Prinzipien erhöht das Sicherheitsniveau bereits in der Designphase drastisch. Sie sind unter anderem an die allgemeinen Empfehlungen von Saltzer und Schröder[38] angelehnt.

Um auch aktuelle Themen der Softwareentwicklung einzubeziehen, wurden diese Designprinzipien angepasst bzw. ergänzt. Für ein besseres Verständnis wurden die englischen Originaltitel beibehalten, gegebenenfalls ergänzt und frei ins Deutsche übersetzt.

5.1.1 Economy of Mechanism („Minimalprinzip")

Die Anwendung soll stets so einfach wie möglich gehalten werden (vgl. hierzu auch Kapitel 2.5.1). Dies ist insbesondere für sicherheitsrelevante Funktionen wichtig.

Die Funktionalität der Applikation soll sich dabei nach Möglichkeit auf die Aufgabenbewältigung beschränken. In diesem Zusammenhang wird auch häufig vom „Minimalprinzip" gesprochen, aus dem sich einige, im Kapitel 6.2 definierten, Prinzipien ableiten.

5.1.2 Fail-safe Defaults & Default Deny (Sichere Standardeinstellungen)

Standardeinstellungen, mit denen Systeme ausgeliefert werden, sind grundsätzlich restriktiv und sicher statt offen bzw. unsicher zu wählen.

Alle Sicherheitsentscheidungen sollen nur ein positives Ergebnis liefern, wenn dies explizit als solches entschieden wird. Dies zieht automatisch ein implizites Verbot mit sich, welches garantiert, dass die

36 [Bobe 2009]
37 [ÖN77 2008]
38 [Salt 1975]

Anwendung sichere Standardentscheidungen treffen kann. Aus diesem Prinzip sind Whitelist-Verfahren abgeleitet, die den Blacklist-Verfahren vorzuziehen sind.

5.1.3 Complete Mediation (etwa „Vollständige Zugriffskontrolle“)

Jeder Objektzugriff muss hinsichtlich seiner Berechtigung geprüft werden. Der Hintergedanke ist hierbei, die Autorisierung nicht nur an einer Stelle durchzuführen und Auswirkungen von Fehlern in der Implementierung so einzuschränken. Beispielsweise sollte nach diesem Prinzip nicht nur der Zugriff auf bestimmte Objekte in der Präsentationsschicht, sondern (sofern möglich) auch auf Business und Datenbank-Ebene geprüft werden.

5.1.4 Open Design (Offenes Design)

Die Sicherheit des Systems sollte nicht von der Geheimhaltung der Funktionsweise abhängen. Stattdessen sollten Algorithmen und der Aufbau der Applikation offengelegt werden können, ohne dadurch deren Sicherheit zu gefährden.

5.1.5 Segregation of Duties (Funktionstrennung)

Für besonders sicherheitsrelevante Aktionen kann es erforderlich sein, dass diese nicht von einer einzigen Person (bzw. Rolle) durchgeführt werden können, sondern hier das Zusammenwirken von mehreren Individuen bzw. Berechtigungsnachweisen erforderlich ist. Beispielsweise können Daten so verschlüsselt werden, dass diese nur durch das Zusammenwirken von zwei oder mehreren Personen entschlüsselt werden können.

Dieses Verfahren stellt sicher, dass die Kompromittierung einer Sicherheitsmaßnahme bzw. eines Berechtigungsnachweises nicht sofort zu einer kompletten Kompromittierung des Systems führt.

5.1.6 Least Privilege („Minimale Berechtigungen“)

Jede Komponente und jeder Benutzer eines Systems darf nur die notwendigen Berechtigungen besitzen, welche zur Erfüllung derer Aufgaben absolut erforderlich sind. Durch dieses Prinzip wird das Schadensausmaß im Fehlerfall oder bei Kompromittierung begrenzt.

5.1.7 Least Common Mechanism (Minimale gemeinsame Ressourcen)

Der Zugriff auf gemeinsam genutzte Ressourcen, wie beispielsweise globale Variablen, durch mehrere Benutzer oder Prozesse ist weitestgehend zu minimieren. Jede gemeinsam genutzte Ressource enthält mögliche unerwünschte Informationsflüsse. Auch kann dies dazu führen, dass ein Benutzer (oder Prozess) die Ressourcen eines anderen beeinflusst. Wenn möglich sollen gemeinsam genutzte Ressourcen daher sinnvoll separiert werden. Dies kann zum Beispiel durch den Einsatz lokaler Variablen bzw. Virtualisierung geschehen.

5.1.8 Psychological Acceptability (Psychologische Akzeptanz)

Sicherheitsmechanismen sollten so gestaltet werden, dass der Einfluss auf die Bedienbarkeit möglichst gering ist bzw. diese durch den Benutzer verstanden werden. Die Verwendung einer Passwortstärke-Funktion, bei der einem Benutzer die Auswirkungen von Änderungen an seinem Passwort anhand dessen Stärke visuell veranschaulicht werden, ist ein Beispiel positiver psychologischer Akzeptanz.

5.1.9 Compromise Recording (Protokollierung von Vorfällen)

Damit Angriffe auf die Anwendung (überhaupt) nachvollzogen werden können, ist es erforderlich, dass die in der Anwendung aufgetretenen Fehler in sicherheitsrelevanten Funktionen (etwa bei der Authentisierung) protokolliert werden. Natürlich ist es hierbei ebenfalls erforderlich, dass die hier erstellten Protokolle regelmäßig nach entsprechenden Vorkommnissen überprüft werden.

5.2 Datenvalidierung

Bei der Datenvalidierung geht es darum sicherzustellen, dass durch Ein- und Ausgaben keine ungewollten Aktionen ausgelöst bzw. Manipulationen durchgeführt werden können.

Die Validierung von Eingabedaten sollte dabei vorzugsweise gegen ein positives Sicherheitsmodell erfolgen (etwa eine Whitelist), in dem Eingabedaten explizit erlaubt bzw. akzeptiert werden. Alle anderen Daten, die nicht diesem Modell entsprechen, sind abzuweisen.

Wichtig hierbei ist, dass Ausgaben jeweils in Bezug auf ihren jeweiligen Ausgabe-Kontext hin validiert und bereinigt werden. So müssen beispielsweise Ausgaben in HTML-Seiten anders validiert werden als solche, die an einen SQL-Interpreter weitergereicht werden. Es ist empfehlenswert, zur Datenvalidierung entsprechende APIs zu verwenden.

5.3 Authentisierung & Sitzungen

Authentisierung und das Session Management stellen sicher, dass der Kommunikationspartner derjenige ist, der er vorgibt zu sein.

Hierbei muss darauf geachtet werden, dass die Stärke der Authentifizierungsmethoden dem Schutzbedarf der Anwendung entspricht. Für einen hohen bis sehr hohen Schutzbedarf sollte etwa die Verwendung von Zweifaktorauthentisierung erwägt werden.

Bei der Verwendung von Passwörtern ist sicherzustellen, dass diese niemals im Klartext übertragen oder gespeichert werden. Session IDs sollten vor dem Auslesen durch Dritte geschützt werden.

Eine Übertragung der Session ID in der URL (mittels URL Rewriting) darf nicht erfolgen. Diese können unter anderem ungewollt in Zugriffs-Protokollen des Webservers geschrieben werden oder über den HTTP-Referer an verlinkte Seiten gesendet werden.

5.4 Autorisierung

Autorisierung bezeichnet die Prüfung, ob ein Subjekt (Benutzer oder Prozess) berechtigt ist, eine bestimmte Aktion durchzuführen. Wichtig hierbei ist, dass diese Berechtigungsprüfung nicht nur einmalig, etwa durch einen vorgeschalteten Security Filter, erfolgt, sondern bei jedem Zugriff (Kapitel 5.1.3), egal, über welchen Pfad dieser Zugriff durchgeführt wird.

Ein Beispiel für fehlerhafte Autorisierung ist eine Webanwendung, in der dem Benutzer zwar nur für ihn erlaubte Inhalte angezeigt werden, jedoch bei einem direkten Aufruf einer Objekt-ID geschützte Inhalte ohne Überprüfung ausliefert werden.

5.5 Kryptographie

Bei dem Einsatz von Kryptographie ist zu beachten, dass nur vom BSI anerkannte Algorithmen und Verfahren zu verwenden sind, die dem Stand der Technik entsprechen. Alle anerkannten Verfahren sind im aktuellen Algorithmenkatalog der Bundesnetzagentur[39] spezifiziert.

Zudem ist darauf zu achten, dass die verwendeten kryptographischen Schlüssel eine ausreichende Entropie besitzen, vertrauliche Schlüsselteile vor unbefugtem Zugriff geschützt sind und dass eine verifizierte Implementierung (siehe Kapitel 4.4.1) verwendet wird.

5.6 Datenhaltung & Datentransport

Zugriffe auf Backendsysteme (wie etwa Datenbanken) sind stets zu authentisieren. Gleiches gilt für alle Systeme, bei denen der Zugriff über eine Trust Boundary erfolgt. Die Authentisierung muss hier entsprechend robust erfolgen. Der Zugriff muss nach dem Least-Privilege-Prinzip erfolgen.

Werden Daten über unsichere Netze übertragen, so darf dies nur verschlüsselt erfolgen (etwa über SSL). Die Speicherung von sensiblen Daten auf nicht-vertrauenswürdigen Systemen (etwa benutzerseitig) sollte nach Möglichkeit vermieden werden. Ist dies nicht möglich, sollte diese nur in verschlüsselter Form erfolgen.

5.7 Konfiguration

Die Konfiguration sollte auf Produktivsystemen stets ausreichend gehärtet werden. Dies bedeutet, dass sämtliche Einstellungen hinsichtlich des Minimalprinzips gesetzt werden dürfen, Fehlermeldungen restriktiv ausgegeben werden und Zugriffe auf die Systeme nach dem Default-Deny-Prinzip erfolgen (siehe hierzu auch 5.1.1, 5.1.2).

5.8 Datenschutz

Beim Umgang mit Benutzer-Daten müssen die Deutschen Datenschutzgesetze (BDSG) Anwendung finden, für Behörden der Länder und Gemeinden sowie für Unternehmen gelten die jeweiligen Landesdatenschutzgesetze. Laut §1 des Bundesdatenschutzgesetz[40] ist „Zweck dieses Gesetzes (...), den einzelnen davor zu schützen, dass er durch den Umgang mit seinen personenbezogenen Daten in seinem Persönlichkeitsrecht beeinträchtigt wird.“ Die Nutzung personenbezogener Daten durch folgende Stellen wird geregelt:

- öffentliche Stellen des Bundes
- öffentliche Stellen der Länder
- nicht-öffentliche Stellen (Unternehmen)

Personenbezogene Daten dürfen nur unter Einwilligung des Betroffenen erhoben werden (§4).

Ein weiteres Grundprinzip des BDSG stellt die Datenvermeidung und Datensparsamkeit (§3a) dar. Die Erhebung, Verarbeitung und Nutzung personenbezogener Daten und die Auswahl und Gestaltung von Datenverarbeitungssystemen sind an dem Ziel auszurichten, so wenig personenbezogene Daten wie möglich zu erheben, zu verarbeiten oder zu nutzen. Insbesondere sind personenbezogene Daten zu anonymisieren oder zu pseudonymisieren, soweit dies nach dem Verwendungszweck möglich ist und keinen im Verhältnis zu dem angestrebten Schutzzweck unverhältnismäßigen Aufwand erfordert.

39 [Algo 2012]
40 [BDSG 2009]

5.9 Fehlerbehandlung und Protokollierung

Bei der Fehlerbehandlung ist darauf zu achten, dass Anwendern niemals sensible Daten angezeigt werden dürfen. Hier gilt das Minimalprinzip. Anders verhält es sich bei der Protokollierung. Hierbei sollte jedoch sichergestellt werden, dass sensible Daten (z.B. personenbezogene Daten) niemals in Protokolldateien geschrieben werden. Hierbei kann eine Maskierungsfunktion helfen. Auch die Verwendung der oben dargestellten Datenbehandlungsstrategie ist hier hilfreich.

Zum einen soll ein übermäßig schnelles und ressourcenintensives Wachstum der Protokolldateien verhindert werden, zum anderen sollen keine wichtigen Informationen verloren gehen. Ebenfalls ist der Zugriff auf Protokolldateien möglichst restriktiv zu handhaben.

Der Webserverprozess darf keine Möglichkeit haben, Protokolldateien im Nachhinein zu ändern. Dies kann über entsprechende Zugriffsregeln oder einen entsprechenden Loggingdienst (z.B. auf einem separaten Server) sichergestellt werden. Auch kann die Verwendung einer separaten Protokolldatei für Sicherheitsereignisse vorteilhaft sein.

Bei der Protokollierung sollten die gültigen Gesetze, darunter BDSG, Telekommunikationsgesetz (TKG) und Telemediengesetz (TMG), beachtet werden[41]. Diese regeln den Umgang mit Daten, die in einer Protokolldatei festgehalten werden dürfen, die Auswertung der Daten und die Aufbewahrungsfristen. Vor allem die Sonderfälle – IP-Adressen und Cookies – sollten genau geprüft werden und die erarbeiteten Lösungen gesetzeskonform sein.[42]

41 [BDSG 2009], [TKG], [TMG]
42 [bfdi 2009], [DFN 2008]

6 Checklisten

Dieses Kapitel fasst die beschriebenen Sicherheitsanforderungen in kompakter Form als Checklisten zusammen. Abhängig vom definierten Schutzbedarf durch den Auftraggeber, hat der Softwareentwicklungsprozess den in der Checkliste definierten Anforderungen zu entsprechen. Abhängig von der Anzahl der erfüllten Anforderungen und deren Erfüllungsgrad, wird der Reifegrad des Softwareentwicklungsprozesses abgeleitet. Die Checklisten sind in Prozessanforderungen und technische Implementierungsanforderungen unterteilt.

6.1 Checkliste für den Entwicklungsprozess

		Schutzbedarf			Reifegrad		
ID	Anforderung	N	H	SH	Erfüllt	Dokum.[43]	Anmerkung
	Awareness Trainings & Schulungen						
	Abhalten einer Awareness-Schulung für jeden Mitarbeiter	●	●	●	☐	☐	
	Abhalten regelmäßiger technischen Awareness-Training für Entwickler	●	●	●	☐	☐	
IPV1	Analyse der Anforderungen der Auftraggeber						
IPV1.1	Erste Analyse der Ziele und Anforderungen des Auftraggebers	●	●	●	☐	☐	
IPV2	Identifikation und Bewertung existierender Sicherheitsanforderungen						
IPV2.1	Identifikation von funktionalen Sicherheitsanforderungen	●	●	●	☐	☐	
IPV2.2	Identifikation von nicht-funktionalen Sicherheitsanforderungen	●	●	●	☐	☐	
IPV3	Erstellung einer Business Impact Analyse						

43 Ist die Maßnahme dokumentiert?

IPV3.1	Durchführung einer Business Impact Analyse (BIA)		●	●	☐	☐	
IPV4	Erstellung des Sicherheitsplans						
IPV4.1	Sicherheitsaktivitäten basierend auf Compliance- und Kundenanforderungen definieren	●	●	●	☐	☐	
IPV4.2	Erforderliche Ressourcen für Sicherheitsaktivitäten abschätzen und planen	●	●	●	☐	☐	
IPV5	Definition einer Software Security Group						
IPV5.1	Definition und Einberufung einer Software Security Group (SSG)		●	●	☐	☐	
KOP1	Beschreibung der Anwendung						
KOP1.1	Beschreibung der Anwendung (grob)	●	●	●	☐	☐	
KOP2	Datenbehandlungsstrategie						
KOP2.1	Datenklassifizierung anhand der Vertraulichkeit		●	●	☐	☐	
KOP2.2	Definition von Maßnahmen zum Umgang von klassifizierten Daten		●	●	☐	☐	
KOP3	Rollen- und Berechtigungskonzept						
KOP3.1	Erläuterung der vorgesehenen Rollen	●	●	●	☐	☐	
KOP3.2	Erläuterung der vorgesehenen Berechtigungen	●	●	●	☐	☐	
KOP3.3	Definition einer Access Control Matrix		●	●	☐	☐	
KOP3.4	Definition von Rollen- und Funktionstrennung		●	●	☐	☐	

KOP4	Sicherheitsanforderungen überprüfen und ggf. ergänzen	●	●	●			
KOP4.1	Sicherheitsanforderungen überprüfen und ggf. ergänzen	●	●	●	☐	☐	
KOP5	Erstellung von Abuse- und Security Use Cases						
KOP5.1	Definition von Sicherheitsanforderungen anhand Use Cases	●	●	●	☐	☐	
KOP5.2	Definition von Sicherheitsanforderungen anhand Abuse Cases		●	●	☐	☐	
KOP6	Bedrohungsmodellierung						
KOP6.1	Identifikation von Bedrohungen		●	●	☐	☐	
KOP6.2	Analyse der identifizierten Bedrohungen		●	●	☐	☐	
KOP6.3	Dokumentation der identifizierten Bedrohungen		●	●	☐	☐	
KOP7	Erstellung der Sicherheitsarchitektur						
KOP7.1	Definition von architekturellen Sicherheitsmechanismen		●	●	☐	☐	
KOP7.2	Definition von Sicherheitskomponenten		●	●	☐	☐	
KOP8	Sicherheitstestplanung						
KOP8.1	Planung der Testaktivitäten in Zeit und Umfang	●	●	●	☐	☐	
KOP8.2	Definition der Testumgebung		●	●	☐	☐	
KOP8.3	Definition der Zuständigkeiten und Planung der Ressourcen	●	●	●	☐	☐	

KOP8.4	Beschaffung bzw. Evaluierung von ggf. benötigten Softwarekomponenten		●	●	☐	☐	
KOP9	Software Security Metriken						
KOP9.1	Definition der KPI „Defect Remediration Window“ (DRW)	●	●	●	☐	☐	
KOP9.2	Definition der KPI „Rate of Defect Recurrence“ (RDR)			●	☐	☐	
KOP9.3	Definition der KPI „Specific Coverage Metric“ (SCM)		●	●	☐	☐	
KOP9.4	Definition der KPI „Security to Quality Defects“ (SQD)		●	●	☐	☐	
IMP1	Security APIs						
IMP1.1	Einsatz von vertrauenswürdigen APIs für Sicherheitsfunktionen	●	●	●	☐	☐	
IMP1.2	Monitoring der APIs auf Bekanntwerden von Sicherheitslücken		●	●	☐	☐	
IMP2	Sicherer Umgang mit Sourcecode						
IMP2.1	Verwendung eines Revisionssystems zur Sourcecode Verwaltung	●	●	●	☐	☐	
IMP2.2	Zugriffsberechtigungen für Lese- und Schreibzugriffe sind erforderlich	●	●	●	☐	☐	
IMP2.3	Einsatz eines Change Management Prozesses zur definierten Programmänderung		●	●	☐	☐	
IMP2.4	Sicherheitsüberprüfung von Systemen, die den Programmcode speichern oder verarbeiten			●	☐	☐	
IMP2.5	Trennung von Entwicklungs-, Test- und Produktivumgebungen	●	●	●	☐	☐	
IMP2.6	Einsatz von Code-Signaturen			●	☐	☐	

IMP3	Secure Coding Standards						
IMP3.1	Verwendung von technologieunabhängigen Secure Coding Standards	●	●	●	☐	☐	
IMP3.2	Ableitung von technologiespezifischen Guidelines	●	●	●	☐	☐	
IMP3.3	Durchführung einer periodischen Entwicklerschulung für Secure Coding Standards		●	●	☐	☐	
IMP4	Security Pushs						
IMP4.1	Durchführung von Security Pushs		●	●	☐	☐	
TES1	Security Tests						
TES1.1	Definierte Sicherheitsanforderungen mittels Security Test Cases verifizieren	●	●	●	☐	☐	
TES1.2	Durchführung von Security Unit Tests		●	●	☐	☐	
TES1.3	Durchführung von Code Reviews			●	☐	☐	
TES1.4	Verwendung eines statischen Code Scanners inkl. Erstellen eines Prüfberichts mit Bewertung der offenen Einträge und Übergabe der Konfigurationsdateien der Prüftools (optional)		●	●	☐	☐	
TES1.5	Verwendung eines Web Security Scanners (optional)	●	●	●	☐	☐	
TES1.6	Durchführung von Fuzz-Testing (optional)	●	●	●	☐	☐	
TES2	Penetration Tests						
TES2.1	Durchführung eines Blackbox Penetration Tests inkl. Erstellen eines Prüfberichts mit Bewertung der offenen Einträge, Übergabe der Konfigurationsdateien der Prüftools und, sofern Open Source Software, Quellcode bereitstellen für Auftraggeber	●	●	●	☐	☐	

TES2.2	Durchführung eines Glassbox Penetration Test inkl. Erstellen eines Prüfberichts mit Bewertung der offenen Einträge, Übergabe der Konfigurationsdateien der Prüftools und, sofern Open Source Software, Quellcode bereitstellen für Auftraggeber		●	●	☐	☐	
TES2.3	Durchführung von Social Engineering Angriffen			●	☐	☐	
TES2.4	Überprüfung der physischen Sicherheit			●	☐	☐	
TES3	Final Security Review						
TES3.1	Durchführung eines Final Security Reviews (FSR)	●	●	●	☐	☐	
AUB1	Sichere Auslieferung der Software						
AUB1.1	Definition einer Vorgehensweise zur sicheren Auslieferung der Software	●	●	●	☐	☐	
AUB1.2	Verwendung von Prüfsummen zur Wahrung der Integrität	●	●	●	☐	☐	
AUB1.3	Verwendung von digitalen Signaturen der Softwarekomponenten zur Wahrung der Authentizität		●	●	☐	☐	
AUB1.5	Entfernen von Testdaten vor der Auslieferung	●	●	●	☐	☐	
AUB1.6	Übergabe der Webanwendung mit allen Softwarekomponenten inkl. sämtlicher Abhängigkeiten und aller Informationen zum Übersetzen	●	●	●	☐	☐	
AUB2	Sicherheitsdokumentation						
AUB2.1	Erstellen einer Sicherheitsdokumentation für den Anwender (bei Clientanwendung)	●	●	●	☐	☐	
AUB2.2	Erstellen einer Sicherheitsdokumentation für den Betrieb	●	●	●	☐	☐	
AUB3	Plattformhärtung						

AUB3.1	Serverkonfiguration nach dem Minimalprinzip		●	●	☐	☐	
AUB3.2	Berechtigungskonfiguration nach dem „Least Privilege“-Prinzip		●	●	☐	☐	
AUB3.3	Härtung von Standardaccounts		●	●	☐	☐	
AUB4	Security Change Management						
AUB4.1	Klassifizierung von Systemänderungen	●	●	●	☐	☐	
AUB4.2	Definition und Anwendung eines Änderungsprozesses basierend auf der Änderungsklassifizierung	●	●	●	☐	☐	
AUB4.3	Validierung der Änderung	●	●	●	☐	☐	
AUB5	Security Response						
AUB5.1	Reaktion auf Incidents mittels eines Security Incident Prozess	●	●	●	☐	☐	
AUB5.2	Gewährleistung eines Zeitraumes für Security Response von kritischen Schwachstellen	●	●	●	☐	☐	
AUB5.3	Gewährleistung eines Zeitraumes für Security Response von jeglichen Schwachstellen		●	●	☐	☐	
AUB5.4	Gewährleistung eines Zeitraumes für rasche Workarounds von jeglichen Schwachstellen			●	☐	☐	
AUB6	3rd Party Software Vulnerability Monitoring						
AUB6.1	Erstellung und Wartung einer Liste mit Abhängigkeiten von Drittsoftware		●	●	☐	☐	
AUB6.2	Monitoring der Liste von Drittsoftware auf neue Sicherheitslücken		●	●	☐	☐	
AUB7	Sichere Standardkonfiguration						

AUB7.1	Auslieferung der Anwendung mit einer sicheren Standardkonfiguration	●	●	●	☐	☐	
AUB8	Web Application Firewall						
AUB8.1	Zusätzlicher Schutz mittels Web Application Firewalls (WAF)		●	●	☐	☐	
AUB8.2	Konfiguration der WAF durch fachmännisches Personal		●	●	☐	☐	
AUB9	Wartung						
AUB9.1	Sichere Zurverfügungstellung von Fernwartungszugängen	●	●	●	☐	☐	
AUB9.2	Sichere Durchführung der Wartung Vorort	●	●	●	☐	☐	

6.2 Checkliste für technische Vorgaben an die Implementierung

6.2.1 Datenvalidierung

		Schutzbedarf			
ID	Anforderung	N	H	SH	Anmerkung
DV1	Es ist sicherzustellen, dass alle Eingaben gegen ein positives Sicherheitsmodell geprüft werden. Dabei muss die Dateneingabe hinsichtlich Inhalt, Plausibilität, gültigen Wertebereich, erlaubten Zeichen, Länge und Typ überprüft werden.	☐	☐	☐	
DV2	Es ist sicherzustellen, dass alle Fehler während der Eingabevalidierung zu einer Zurückweisung bzw. Bereinigung der entsprechenden Eingaben führen.	☐	☐	☐	
DV3	Es ist sicherzustellen, dass ein bestimmter Zeichensatz (z.B. UTF-8) für alle Eingabequellen definiert ist.	☐	☐	☐	
DV4	Es ist sicherzustellen, dass sämtliche Eingabevalidierung mindestens serverseitig durchgeführt wird.	☐	☐	☐	
DV5	Dateiuploads	☐	☐	☐	
	Der Dateityp der am Server zu speichernden Datei muss dem erwarteten Dateityp entsprechen.				
	Die Dateiendung der generierten Datei muss überprüft und eingeschränkt werden.				
	Die maximale Größe einer zu generierenden Datei muss definiert werden.				
	In Bezug auf die Dateigenerierung muss das Minimalprinzip erfüllt werden. Dateien dürfen nur generiert werden, sofern es für die korrekte Funktionalität der Anwendung notwendig ist.				
	Bilder und Dokumente sollten konvertiert werden, um Schadcode zu entfernen und Fehlformatierungen zu korrigieren.				
	Vom Benutzer hochgeladene Dateien dürfen auf dem Server nicht ausgeführt werden können.				

DV6	Einbindung externer sowie interner Ressourcen	☐	☐	☐	
	Es darf keine Ressource eingebunden werden, deren Identifikator durch ungefilterte Benutzereingaben definiert wird.				
	Alle eingebundenen Ressourcen müssen definiert und in einer eigenen Liste bzw. einem eigenen Verzeichnis abgelegt werden.				
	Der Zugriff ist auf Ressourcen zu beschränken, die für die Funktion der Webapplikation notwendig sind.				
	Externe Ressourcen dürfen nur mit den minimal notwendigen Rechten verwendet werden.				
	Die einzubindenden Ressourcen müssen so abgelegt werden, dass sie nicht unbefugt gelesen oder verändert werden können.				
DV7	Es ist sicherzustellen, dass sämtliche Methoden, die Daten als Input für einen Interpreter bereitstellen, alle Zeichen encodieren, die für den jeweiligen Interpreter als unsicher bekannt sind. Dies bezieht sich unter anderem auf folgende Interpreter:	☐	☐	☐	
	Es ist sicherzustellen, dass die Verwendung von nicht vertrauenswürdigen Daten bei SQL-Interpretern nur mittels parametrisierter Interfaces, Prepared Statements oder zumindest korrekter Encodierung erfolgt.				
	Es ist sicherzustellen, dass die Verwendung von vertrauenswürdigen Daten bei XML-Interpreter nur mittels parametrisierter Interfaces oder korrekter Encodierung erfolgt.				
	Es ist sicherzustellen, dass die Verwendung von nicht vertrauenswürdigen Daten beim LDAP-Interpretern nur mittels korrekter Encodierung erfolgt.				
	Es ist sicherzustellen, dass die Verwendung von vertrauenswürdigen Daten bei Betriebssystem Kommando Interpretern nur mittels korrekter Encodierung erfolgt.				
DV8	Es ist sicherzustellen, dass nicht vertrauenswürdige Daten, welche von der Applikation ausgegeben werden, für den jeweiligen Kontext entsprechend validiert werden. Dies beinhaltet unter anderem folgende Datenausgaben.	☐	☐	☐	
	Es ist sicherzustellen, dass nicht vertrauenswürdige Daten, welche in HTML ausgegeben werden (inklusive HTML-Elemente, Attribute, JavaScript-Datenobjekte, CSS-Blöcke und URI-Attribute),				

	für den jeweiligen Kontext entsprechend validiert werden.				
DV9	Es ist sicherzustellen, dass die Anwendung für jeden akzeptierten Ein- und Ausgabe Datentyp einen zentralen Mechanismus verwendet, welcher die Daten gemäß ihres Verwendungszwecks validiert.		☐	☐	

6.2.2 Authentisierung & Sitzungen

		Schutzbedarf			
ID	Anforderung	N	H	SH	Anmerkung
AH1	Es ist sicherzustellen, dass alle nicht-öffentlichen Seiten und Ressourcen eine Authentisierung erforderlich machen. Die Webapplikation muss bei der Authentifizierung sicherstellen, dass jeder Benutzer mit Hilfe eindeutiger Sitzungsidentifikatoren, beispielsweise ein eindeutiger und zufällig generierter Sitzungsschlüssel, welche über einen vertraulichen Kanal transferiert werden, identifiziert werden kann.	☐	☐	☐	
AH2	Es ist sicherzustellen, dass die Anwendung geeignete Mechanismen implementiert hat, mit denen sie Bruteforce-Angriffe auf die Authentisierung-Mechanismen verhindert kann (z.B. eine Teergrube).	☐	☐	☐	
AH3	Es ist sicherzustellen, dass alle Authentisierungskontrollen serverseitig umgesetzt sind.	☐	☐	☐	
AH4	Es ist sicherzustellen, dass alle Authentisierungskontrollen bei einem Fehlerfall in einen sicheren Zustand fallen.	☐	☐	☐	
AH6	Es ist sicherzustellen, dass Anwender ihre Zugangsdaten nur über einen Mechanismus ändern können, welcher mindestens über den gleichen Schutzlevel verfügt, wie bei der primären Authentisierung.	☐	☐	☐	
AH7	Es ist sicherzustellen, dass Passwörter von Benutzern nur als gesalzener Hash gespeichert werden.	☐	☐	☐	
AH8	Es ist sicherzustellen, dass Sitzungsidentifikatoren ungültig werden, wenn sich der Benutzer abmeldet.	☐	☐	☐	

AH9	Es ist sicherzustellen, dass Sitzungen bzw. Sitzungsidentifikatoren nach einer bestimmten Zeit der Inaktivität des Gegenübers ungültig werden (Zeitüberschreitung durch Inaktivität).	☐	☐	☐	
AH10	Es ist sicherzustellen, dass Sitzungen nach einer administrativ konfigurierbaren Zeit der Aktivität ungültig werden (absolute Zeitüberschreitung).	☐	☐	☐	
AH11	Es ist sicherzustellen, dass die Sitzungsidentifikatoren nach erfolgtem/erneutem Anmelden erneuert werden.	☐	☐	☐	
AH12	Es ist sicherzustellen, dass alle Seiten, deren Zugriff eine Authentisierung erfordert, über einen Abmeldemechanismus verfügen.	☐	☐	☐	
AH13	Es ist sicherzustellen, dass Sitzungsidentifikatoren nirgendwo anders als in Cookies von der Applikation preisgegeben werden; insbesondere nicht in URLs, Fehlermeldungen oder Protokolldateien.	☐	☐	☐	
AH14	Es ist sicherzustellen, dass nur authentische Sitzungsidentifikatoren, die von der Applikation selbst oder einer verwendeten Drittkomponente erstellt wurden, von der Anwendung akzeptiert werden.	☐	☐	☐	
AH15	Es ist sicherzustellen, dass verwendete Sitzungsidentifikatoren über eine ausreichende Länge und Zufälligkeit verfügen, um typischen Angriffen für die jeweilige Umgebung zu widerstehen.	☐	☐	☐	
AH16	Wird ein Benutzer mit Hilfe der Sitzung authentifiziert, darf der Sitzungsidentifikator nur über verschlüsselte Kanäle übertragen werden.	☐	☐	☐	
AH17	Es ist sicherzustellen, dass im angemeldeten Bereich verwendete Sitzungsidentifikatoren ein entsprechend restriktiv gesetztes Domain- und Pfad-Attribut verwenden.		☐	☐	
AH18	Es ist sicherzustellen, dass alle Authentisierungskontrollen (inklusive Bibliotheken, welche externe Authentisierungsdienste aufrufen) an einer zentralen Stelle implementiert werden.		☐	☐	
AH19	Es ist sicherzustellen, dass eine erneute Authentisierung (Re-Authentisierung) erfolgt, bevor der Aufruf einer sensiblen Operation zugelassen wird.		☐	☐	
AH20	Es ist sicherzustellen, dass Zugangsdaten nach einer administrativ konfigurierbaren Zeit ungültig werden und erneuert bzw. geändert werden müssen.		☐	☐	
AH21	Es ist sicherzustellen, dass alle Entscheidungen in Bezug auf die Authentisierung protokolliert werden.		☐	☐	

AH22	Es ist sicherzustellen, dass alle Zugangsdaten für den Zugriff auf Anwendungs-externe Dienste nur verschlüsselt und an einem geschützten Ort gespeichert werden (z.B. nicht im Quellcode).		☐	☐	
AH23	Die Applikation muss in der Lage sein, die Umsetzung einer vorgegebenen Passwort Richtlinie zu erzwingen.	☐	☐	☐	

6.2.3 Autorisierung

		Schutzbedarf			
ID	Anforderung	N	H	SH	Anmerkung
AO1	Es ist sicherzustellen, dass Anwender nur dann auf geschützte Funktionen, Ressourcen oder Daten zugreifen können, wenn dieser Zugriff zuvor erfolgreich autorisiert wurde. Gemäß des Prinzips der Zugriffskontrolle, muss jeder Zugriff auf ein beliebiges Objekt auf seine Befugnis hin überprüft werden.	☐	☐	☐	
AO2	Es ist sicherzustellen, dass Anwender nur dann geschützte Funktionen, Ressourcen oder Daten ändern können, wenn dieser Zugriff zuvor erfolgreich autorisiert wurde. Gemäß des Prinzips der Zugriffskontrolle, muss jede Veränderung eines beliebigen Objektes auf seine Befugnis hin überprüft werden.	☐	☐	☐	
AO3	Applikationen dürfen Benutzern nur die minimalen Rechte, welche für die Ausführung der erforderlichen Funktionen unbedingt benötigt werden, zuweisen.	☐	☐	☐	
AO4	Applikationen dürfen nur die minimalen Rechte besitzen, welche für die Ausführung der erforderlichen Funktionalität unbedingt benötigt werden.	☐	☐	☐	
AO5	Es ist sicherzustellen, dass Zugriffskontrollen im Fehlerfall in einen sicheren Zustand fallen.	☐	☐	☐	
AO6	Es ist sicherzustellen, dass Zugriffskontrollen serverseitig umgesetzt sind.	☐	☐	☐	
AO7	Es ist sicherzustellen, dass für alle Arten von Zugriffen auf geschützte Ressourcen ein zentraler Mechanismus existiert (inklusive Bibliotheken, welche externe Autorisierungsdienste aufrufen).		☐	☐	
AO8	Es ist sicherzustellen, dass das Protokollieren sämtlicher Entscheidungen von Zugriffskontrollen		☐	☐	

	möglich ist und alle fehlerhaften Entscheidungen stets protokolliert werden.				

6.2.4 Kryptographie

		Schutzbedarf			
ID	Anforderung	N	H	SH	Anmerkung
KY1	Es ist sicherzustellen, dass kryptographische Funktionen, die von der Anwendung für den Schutz sensible Informationen eingesetzt werden, serverseitig implementiert sind.	☐	☐	☐	
KY2	Es ist sicherzustellen, dass alle Arten von Zugangsdaten, welche im Klartext lokal gespeichert sind und für den Zugriff auf sicherheitsrelevante Konfigurationsdaten verwendet werden (e.g. Geheime Schlüssel, Passwörter), gegen nicht-autorisierten Zugriff geschützt sind.	☐	☐	☐	
KY3	Es ist sicherzustellen, dass Passwort-Hashes gesalzen sind.	☐	☐	☐	
KY4	Es ist sicherzustellen, dass alle kryptographischen Module im Fehlerfall in einen sicheren Zustand fallen.	☐	☐	☐	
KY5	Es ist sicherzustellen, dass Fehler von kryptographischen Modulen protokolliert werden.		☐	☐	
KY6	Es ist sicherzustellen, dass ausschließlich kryptographische Standards und Parameter verwendet werden, welche in der aktuellen technischen Richtlinie des BSI 02102 „Kryptographischen Verfahren: Empfehlungen und Schlüssellängen“ verzeichnet sind[44].	☐	☐	☐	
KY7	Es ist sicherzustellen, dass sowohl für alle externen als auch Backend-Verbindungen, die entweder authentisiert sind oder sensible Daten bzw. Funktionen betreffen, ein verschlüsselter Kanal verwendet wird. Ausgenommen hiervon sind interne Verbindungen wie lokal laufende Services, auf welche der Zugriff ausschließlich lokal erfolgt und die dementsprechend abgeschottet sind.	☐	☐	☐	
KY8	Es ist sicherzustellen, dass ein Pfad zwischen einer vertrauenswürdigen CA und jedem TLS-Server-Zertifikat hergestellt werden kann und jedes Server-Zertifikat valide ist.	☐	☐	☐	

44 [BSI TR 102]

KY9	Es ist sicherzustellen, dass fehlgeschlagene TLS-Verbindungen nicht auf eine unsichere Verbindung zurückgesetzt werden.		☐	☐	
KY10	Es ist sicherzustellen, dass Fehler von TLS-Verbindungen zum Backend geloggt werden.			☐	
KY11	Es ist sicherzustellen, dass Zertifikatspfade für alle Clientzertifikate mittels Vertrauensanker (Trust Anchor) und Sperrinformationen (Revocation Information) erstellt und verifiziert werden.			☐	

6.2.5 Datenhaltung & Datentransport

		Schutzbedarf			
ID	Anforderung	N	H	SH	Anmerkung
DH1	Es ist sicherzustellen, dass alle Verbindungen zu internen und externen Systemen, welche sensible Informationen oder Funktionen betreffen, ein Nutzerkonto verwenden, welches nur über minimale Privilegien verfügt.	☐	☐	☐	
DH2	Es ist sicherzustellen, dass alle Verbindungen zu externen Systemen authentisiert sind, wenn diese sensible Informationen oder Funktionen betreffen.	☐	☐	☐	
DH3	Es ist sicherzustellen, dass sensible Daten ausschließlich im HTTP-Body an den Server gesendet werden können.	☐	☐	☐	
DH4	Es ist sicherzustellen, dass alle zwischengespeicherten oder temporären Kopien von sensiblen Daten, welche zum Client gesendet werden, vor unbefugten Zugriff geschützt bzw. ungültig gemacht oder zerstört werden.	☐	☐	☐	
DH5	Es ist sicherzustellen, dass alle zwischengespeicherten oder temporären Kopien sensibler Daten vor unbefugtem Zugriff geschützt sind oder dass diese ungültig gemacht bzw. gelöscht werden, falls dies der Fall ist.			☐	
DH6	Es ist sicherzustellen, dass nicht benötigte Dateien im Wurzelverzeichnis des Servers entfernt werden.	☐	☐	☐	

6.2.6 Konfiguration

		Schutzbedarf			
ID	Anforderung	N	H	SH	Anmerkung
KO1	Standardeinstellungen müssen so definiert sein, dass keine Sicherheitsrisiken von ihnen ausgehen.	☐	☐	☐	
KO2	Konfigurationen müssen dem Minimalprinzip entsprechen.	☐	☐	☐	
KO3	Es ist sicherzustellen, dass alle sicherheitsrelevanten Konfigurationsinformationen nur an Orten gespeichert werden, die vor unbefugtem Zugriff geschützt sind.	☐	☐	☐	
KO4	Es ist sicherzustellen, dass alle Zugriffe auf die Anwendung abgelehnt werden, wenn diese keinen Zugriff zu ihrer Sicherheitskonfiguration herstellen kann.	☐	☐	☐	
KO5	Es ist sicherzustellen, dass Directory Browsing abgeschaltet ist, sofern dies nicht ausdrücklich gewünscht sein sollte.	☐	☐	☐	
KO6	Es ist sicherzustellen, dass ein Schutz gegen Massenabfragen eingerichtet wird.	☐	☐	☐	
KO7	Es ist sicherzustellen, dass serverseitiger Programmcode von dem Server interpretiert wird. Der Programmcode darf nicht im Quellcode des Benutzers sichtbar sein.	☐	☐	☐	

6.2.7 Datenschutz

		Schutzbedarf			
ID	Anforderung	N	H	SH	Anmerkung
DS1	Es ist sicherzustellen, dass die Anwendung keine nicht unbedingt notwendigen sensiblen Daten der Anwendung protokolliert. Dies beinhaltet beispielsweise Sitzungsidentifikatoren, personenbezogene oder andere sensible Informationen, z. B. Berechtigungsnachweise.	☐	☐	☐	
DS2	Es ist sicherzustellen, dass ein Verfahren für die Löschung von sensiblen Daten existiert, wenn deren Aufbewahrungsfrist abgelaufen ist.			☐	

DS3	Es ist sicherzustellen, dass eine Liste der sensiblen Daten existiert, die von der Anwendung verarbeitet werden. Eine explizite Richtlinie soll existieren, wie Zugriffe auf diese Daten kontrolliert werden und wann diese zu verschlüsseln sind (sowohl für Speicherung als auch Übertragung). Es ist sicherzustellen, dass diese Richtlinie korrekt umgesetzt ist.			☐	
DS4	Bei der Implementierung und Nutzung von Mechanismen zur automatisierten Nutzerverhaltensanalyse und Leistungskontrolle ist die aktuelle Rechtslage – BDSG, TKG, TMG - unbedingt zu beachten.	☐	☐	☐	

6.2.8 Fehlerbehandlung und Protokollierung

		Schutzbedarf			
ID	Anforderung	N	H	SH	Anmerkung
FE1	Es ist sicherzustellen, dass die Anwendung keine Fehlermeldungen (z.B. Stack Traces) ausgibt, welche sensible Daten enthalten (e.g. Sitzungsidentifikatoren und personenbezogene Informationen).	☐	☐	☐	
FE2	Es ist sicherzustellen, dass alle serverseitigen Fehler auch serverseitig behandelt werden.	☐	☐	☐	
FE3	Es ist sicherzustellen, dass alle Protokollierungsmechanismen serverseitig implementiert sind.	☐	☐	☐	
FE4	Es ist sicherzustellen, dass Sicherheitsmechanismen im Fehlerfall den Zugriff verweigern.	☐	☐	☐	
FE5	Es ist sicherzustellen, dass Ereignisse, die nicht vertrauenswürdige Informationen betreffen, nicht Code enthalten können, der im für die Protokollanzeige vorgesehenen Programm zur ungewollten Ausführung gebracht werden kann.	☐	☐	☐	
FE6	Es ist sicherzustellen, dass sicherheitsrelevante Protokolldateien vor unbefugten Zugriff und Modifikation geschützt sind.	☐	☐	☐	
FE7	Es ist sicherzustellen, dass die zur Protokollierung eingesetzten Sicherheitsmechanismen die Möglichkeit bieten, sowohl fehlerhafte als auch erfolgreiche sicherheitsrelevante Ereignisse protokolliert werden können.		☐	☐	

FE8	Es ist sicherzustellen, dass jedes protokollierte Ereignis folgende Informationen enthält (soweit bekannt): 1. Einen (sekundengenauen) Zeitstempel von einer vertrauenswürdigen Quelle. 2. Das Sicherheitslevel des Ereignisses. 3. Einen Hinweis darauf, ob dieses Ereignis sicherheitsrelevant ist. 4. Die Identität des Benutzers bzw. der Quelle, der/die das Ereignis ausgelöst hat. Die aktuelle Rechtslage - BDSG, TKG, TMG u.a. - ist unbedingt zu beachten. 5. Die Quell-IP, die mit dem Ereignis assoziiert ist. Die aktuelle Rechtslage - BDSG, TKG, TMG u.a. - ist unbedingt zu beachten. 6. Ob das Ereignis erfolgreich oder fehlerhaft ist. 7. Die Beschreibung des Ereignisses.		☐	☐	
FE9	Es wird empfohlen, dass die Anwendung für das Protokollieren nur eine Implementierung bzw. ein Verfahren verwendet.		☐	☐	
FE10	Es ist sicherzustellen, dass ein Programm zur Protokollanalyse vorhanden ist, welches es erlaubt, in den Protokolldateien nach Ereignissen auf Basis von Kombinationen von Suchkriterien aller Felder im verwendeten Format zu suchen.			☐	

6.2.9 HTTP Protokoll & Web-Seiten

		Schutzbedarf			
ID	Anforderung	N	H	SH	Anmerkung
HT1	Es ist sicherzustellen, dass Weiterleitungen keine nicht validierten Daten enthalten.	☐	☐	☐	
HT2	Es ist sicherzustellen, dass die Anwendung nur eine definierte Menge von HTTP-Methoden erlaubt, wie etwa GET und POST.	☐	☐	☐	
HT3	Es ist sicherzustellen, dass das HTTP Only-Flag für alle Cookies verwendet wird, auf die nicht mittels JavaScript zugegriffen werden muss.	☐	☐	☐	
HT4	Es ist sicherzustellen, dass das Secure-Flag von allen Cookies verwendet wird, die vertrauliche Informationen enthalten.	☐	☐	☐	

HT5	Es ist sicherzustellen, dass die Anwendung ein kryptographisch starkes Zufallstoken generiert, welches Teil sämtlicher Links und Formulare ist, die Transaktionen oder Zugriffe auf sensible Daten bereitstellen. Zudem hat die Anwendung zu prüfen, ob der korrekte Token des jeweiligen Benutzers vorhanden ist, bevor diese eine Anfrage ausführt.	☐	☐	☐	
HT6	Es ist sicherzustellen, dass für alle Formulare, welche sensible Daten enthalten, Client Caching deaktiviert ist (inklusive AutoComplete-Funktionen).	☐	☐	☐	
HT7	Es ist sicherzustellen, dass jede HTTP-Antwort einen Content Type enthält, der einen sicheren Zeichensatz festlegt (z. B. UTF-8).		☐	☐	
HT8	Es ist sicherzustellen, dass HTTP-Header, sowohl in HTTP Anfragen als auch HTTP Antworten, nur druckbare ASCII-Zeichen enthalten.		☐	☐	
HT9	Es ist sicherzustellen, dass für alle Passwortfelder (sowie Formulare, welche diese enthalten) die Autocomplete-Funktion deaktiviert haben.	☐	☐	☐	
HT10	Es ist sicherzustellen, dass potenziell kritische Debugging-Mechanismen vor der Inbetriebnahme abgeschaltet sind.	☐	☐	☐	
HT11	Es ist sicherzustellen, dass von Benutzern erstellte Links auf Webseiten nicht verschleiert werden können. Links müssen stets komplett (inklusive sämtlicher Paramter) sichtbar sein.	☐	☐	☐	
HT12	Es ist sicherzustellen, dass eine Online-Hilfefunktion zu sicherheitsspezifischen Funktionen und Einstellungen implementiert wurde. Eine Kontaktmöglichkeit zur Meldung von Sicherheitsvorfällen muss genannt werden.	☐	☐	☐	

Glossar

Begriff	Beschreibung
Abuse Case	Abstrakte Beschreibung der Interaktion zwischen einem System und seiner Umgebung. Die Interaktion resultiert in einem Schaden für einen der Beteiligten.
Access Control Matrix	Eine Access Control Matrix, über welche anschaulich die Berechtigungen der existierenden Rollen auf bestimmte Assets dargestellt werden.
Asset	Grundsätzlich sind Assets alles, was einen Vermögenswert für ein Unternehmen darstellt. In diesem Zusammenhang sind Assets alle Bestandteile (Daten, Systeme und Funktionen) einer Webanwendung, für die sich ein Schutzbedarf hinsichtlich der Schutzziele ableiten lässt.
Authentisierung	Sicherstellung, dass der Kommunikationspartner derjenige ist, der er vorgibt zu sein.
Authentizität	Bezeichnet die Eigenschaft der Echtheit der Daten.
Autorisierung	Prüfung, ob Subjekt berechtigt ist, eine bestimmte Aktion durchzuführen.
Code Review	Sicherheitstest; Manuelle Überprüfung des Programmcodes.
Cross Site Scripting	Angriff, bei dem Schadcode in eine dynamische Webseite eingeschleust wird.
Final Security Review	Letzter Sicherheitstest; wird auf die fertiggestellte Anwendung durchgeführt.
Funktionale Sicherheit	Funktionale Sicherheitsaspekte wie Authentisierung, Autorisierung, Kryptographie etc.
Fuzz Testing	Ungültige, unerwartete oder zufällige Werte werden generiert und als Eingabe an die Anwendung weitergegeben.
HTML	Hypertext Markup Language; textbasierte Auszeichnungssprache zur Strukturierung von Inhalten in Dokumenten.
HTTP	Hypertext Transfer Protocol; Protokoll zur Übertragung von Daten über ein Netzwerk.
Incident	Ein nicht geplanter Ausfall bzw. ein Qualitätsverlust eines Services.

Begriff	Beschreibung
Integrität	Die Vollständigkeit und Unversehrtheit von Daten muss gewährleistet werden.
Least Privilege	Beschränkung der Berechtigungen eines Dateisystems auf das absolut Notwendige.
Low Hanging Fruits	Sicherheitsmaßnahmen, die mit wenig Ressourcenaufwand umgesetzt werden können.
Minimalprinzip	Die Funktionalität soll sich auf die Aufgabenbewältigung beschränken.
Nicht-Abstreitbarkeit	Eine durchgeführte Handlung ist eindeutig zurechenbar.
Nicht-funktionale Sicherheit	Nicht-funktionale Sicherheitsaspekte wie Fehler in Eingabe- und Ausgabevalidierung etc.
Penetration Test	Sicherheitstest; erfolgt aus der Sicht des Angreifers.
Plattformhärtung	Entfernung aller Softwarebestandteile der Serversysteme, die für die Aufgabenerfüllung nicht erforderlich sind.
Security API	Stellen Softwareentwicklern geprüfte Sicherheitsfunktionen zur Verfügung.
Security Gates	Meilensteine im Projekt, die nur unter Einhaltung definierter Sicherheits- bzw. Qualitätskriterien passiert werden können.
Security Push	Vorgehen in bestimmten Phasen der Entwicklung. Das gesamte Team konzentriert sich auf die Verbesserung der Sicherheit der Webanwendung.
Security Test Case	Sicherheitstest; bestimmte Funktionalität wird auf ihre Korrektheit überprüft.
Security Unit Test	Sicherheitstest; Softwareeinheiten (Funktionen, Methoden, Klassen) werden auf korrekte Funktionalität überprüft.
Software Assurance	Grad an Vertrauen in die Sicherheit von Software.
Software Security Group	Wichtige organisatorische Einheit, die über Sicherheitsaktivitäten entscheidet.
SQL-Injection	Einschleusen von SQL-Statements, aufgrund von mangelnder Eingabeüberprüfung.
Use Case	Abstrakte Beschreibung der Interaktion zwischen einem System und seiner Umgebung.

Begriff	Beschreibung
Verfügbarkeit	Daten müssen zu definierten Zeiten im Einklang mit der Vertraulichkeit und Integrität zur Verfügung stehen.
Vertraulichkeit	Daten dürfen nur von Personen verarbeitet werden, die dafür bestimmt sind.
WAF	Web Application Firewall; Firewall auf Anwendungsebene.

Literaturverzeichnis

7saf2010	7safe: UK Security Breach Investigations Report, http://www.7safe.com/breach_report/Breach_report_2010.pdf
Nist 2002	National Institute of Standards & Technology: The Economic Impacts of Inadequate Infrastructure for Software Testing
White_2013	WhiteHat Security: Website Security Statistic Report 2013
ISO2	International Organization for Standardization: ISO/IEC 27005:2011 Information technology -- Security techniques -- Information security risk management
ISiW	Bundesamt für Sicherheit in der Informationstechnik: Sicheres Bereitstellen von Web-Angeboten (ISi-Web-Server), https://www.bsi.bund.de/DE/Themen/Cyber-Sicherheit/ISi-Reihe/ISi-Web-Server/web_server_node.html
ITGS	Bundesamt für Sicherheit in der Informationstechnik: IT-Grundschutz - M.2.338 Erstellung von zielgruppengerechten Sicherheitsrichtlinien, https://www.bsi.bund.de/ContentBSI/grundschutz/kataloge/m/m02/m02338.html
SiWe 2006	Bundesamtes für Sicherheit in der Informationstechnik: Sicherheit von Webanwendungen, Maßnahmen und Best Practices, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/WebSec/WebSec_pdf.pdf?__blob=publicationFile
ÖN77	Austrian Standards Institute: ÖNORM A 7700 - der Standard für die Sicherheit von Webanwendungen, http://www.a7700.org/
ASVS	OWASP Foundation: Application Security Verification Standard Project, https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project
SAMM	OWASP Foundation: OpenSAMM, http://www.opensamm.org/
BSIM 2011	Gary McGraw, Brian Chess, Sammy Migues: Building Security In Maturity Model
Casp 1997	Casper Jones: Applied Software Measurement: Assuring Productivity and Quality, 2d Edition, 1997
Gilb 1988	Tom Gilb: Principles of Software Engineering Management, 1988
Micr 2005	Microsoft Corporation: The Trustworthy Computing Security Development Lifecycle, http://msdn.microsoft.com/en-us/library/ms995349.aspx
ISiS2008	Bundesamt für Sicherheit in der Informationstechnik: ISi-S Sicheres Bereitstellen von Web-Angeboten, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Internetsicherheit/isi_web_server_studie_pdf.pdf;jsessionid=A681D388B61F2F12DFA2434EE1F2EDE4.2_cid294?__blob=publicationFile
SQUARE 2005	Mead, Nancy R., Eric D. Hough, Thodore R. Stehney, II.: Security Quality Requirements Engineering (SQUARE) Methodology; Technical Report CMU/SEI-2005-TR-009 | ESC-TR-2005-009, www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA443493
Mcgr 2004	Gary McCraw: Misuse and Abuse Cases: getting Past the Positive
Rafa 2010	Rafal Los: Magc Numbers, An In-Depth Guide to the 5 Key Performance Indicators for Web Application Security
DACS 2008	Data & Analysis Center for Software: Enhancing the Development Life Cycle to Produce Secure Software. A Reference Guidebook on Software Assurance, www.seas.upenn.edu/~lee/09cis480/papers/DACS-358844.pdf
PenT2003	Bundesamt für Sicherheit in der Informationstechnik: Durchführungskonzept für Penetrationstests,

	https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Penetrationstest/penetrationstest_pdf.pdf?__blob=publicationFile
WAFs	OWASP: Best Practices: Einsatz von Web Application Firewalls
ÖN77 2008	Austrian Standards Institute: ÖNORM A 7700 - der Standard für die Sicherheit von Webanwendungen, http://www.a7700.org/
Bobe 2009	Mike Boberksi, Jeff Williams, Dave Wichers: OWASP Application Security Verification Standard (ASVS) – Web Application Edition
Salt 1975	Jerome Saltzer, Michael Schröder: The Protection of Information in Computer Systems
Algo 2012	Bundesnetzagentur: Algorithmenkataloge, http://www.bundesnetzagentur.de/DE/Sachgebiete/QES/Veroeffentlichungen/Algorithmen/algorithmen_node.html
BDSG 2009	Bundesdatenschutzgesetz, 2009 http://www.bfdi.bund.de/SharedDocs/Publikationen/GesetzeVerordnungen/BDSG.pdf
bfdi 2009	Arbeitskreis „Technische und organisatorische Datenschutzfragen“ der Konferenz der Datenschutzbeauftragten des Bundes und der Länder: Orientierungshilfe "Protokollierung", http://www.bfdi.bund.de/DE/Themen/TechnologischerDatenschutz/TechnologischeOrientierungshilfen/Artikel/OHProtokollierung.html
TKG	Telekommunikationsgesetz, 2004 http://www.gesetze-im-internet.de/tkg_2004/
DFN 2008	Forschungsstelle Recht im DFN: Speicherrechte nach dem Telemediengesetz (TMG) und demTelekommunikationsgesetz (TKG), https://www.dfn.de/fileadmin/3Beratung/Recht/Speicherrechte_nach_TMG_und_TKG.pdf
TMG	Telemediengesetz, 2007 http://dejure.org/gesetze/TMG
BSI TR 102	Bundesamt für Sicherheit in der Informationstechnik : BSI TR-02102 Kryptographische Verfahren: Empfehlungen und Schlüssellängen, https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html

Stichwort- und Abkürzungsverzeichnis