



Bundesamt
für Sicherheit in der
Informationstechnik

Leitfaden zur Entwicklung sicherer Webanwendungen

Empfehlungen und Anforderungen an Auftraggeber aus der öffentlichen
Verwaltung





ADVISOR FOR YOUR INFORMATION SECURITY

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>

© Bundesamt für Sicherheit in der Informationstechnik 2013

Inhaltsverzeichnis

1	Einleitung, Motivation und Ziele.....	5
1.1	Darstellung des gewählten Ansatzes.....	5
2	Grundlagen sicherer Webanwendungen.....	6
2.1	Risiken von Webanwendungen.....	6
2.2	Allgemeine Risiken des Einsatzes unterschiedlicher Webtechnologien.....	6
2.2.1	Mobile Web.....	6
2.2.2	Web-Services.....	7
2.2.3	Cloud Computing.....	7
2.3	Secure Development Lifecycle.....	7
3	Vergabephase.....	8
3.1	Ermittlung des Schutzbedarfs.....	8
3.2	Ausschreibung auf Basis des Schutzbedarfs.....	11
4	Überprüfung der Vorgaben an den Entwicklungsprozess.....	12
4.1	Aufbau.....	12
4.1.1	Reifegrade.....	12
4.2	Phase 1: Initiale Planung & Vergabephase.....	13
4.2.1	Leistungskriterien.....	13
4.2.2	Quality Gate: Prüfbare Ergebnisse der Phase.....	14
4.3	Phase 2: Konzeption & Planung.....	14
4.3.1	Datenflussanalyse.....	14
4.3.2	Rollen- und Berechtigungskonzept.....	14
4.3.3	Abuse Cases.....	15
4.3.4	Bedrohungsmodellierung.....	15
4.3.5	Sicherheitsarchitektur.....	18
4.3.6	Sicherheitstestplanung.....	18
4.3.7	Leistungskriterien.....	18
4.3.8	Quality Gate: Prüfbare Ergebnisse der Phase.....	18
4.4	Phase 3: Implementierung.....	20
4.4.1	Security APIs.....	20
4.4.2	Sicherer Umgang mit Sourcecode.....	20
4.4.3	Secure Coding Standards.....	20
4.4.4	Security Push.....	20
4.4.5	Leistungskriterien.....	20
4.4.6	Quality Gate: Prüfbare Ergebnisse der Phase.....	21
4.5	Phase 4: Testen.....	21
4.5.1	Security Tests.....	22
4.5.2	Final Security Review (FSR).....	27
4.5.3	Leistungskriterien.....	28
4.5.4	Quality Gate: Allgemeine prüfbare Ergebnisse der Phase.....	28
4.6	Phase 5: Auslieferung & Betrieb.....	28
4.6.1	Sicherheitsdokumentation.....	28
4.6.2	Security Change Management.....	29
4.6.3	Leistungskriterien.....	29
4.6.4	Quality Gate: Prüfbare Ergebnisse der Phase.....	29
4.7	Ende des Lebenszyklus.....	29

5	Checklisten für die Bewertung potenzieller Auftragnehmer.....	31
5.1	Reifegrad des Auftragnehmers	31
5.2	Detaillierter Reifegrad des Auftragnehmers.....	31
5.3	Checkliste für den Entwicklungsprozess.....	32
	Glossar.....	41
	Literaturverzeichnis.....	43

Abbildungsverzeichnis

Abbildung 1:	Entwicklungsprozess.....	12
Abbildung 2:	Aktivitäten der Phase 1 "Initiale Planung und Vergabephase".....	13
Abbildung 3:	Aktivitäten der Phase 2 "Konzeption & Planung".....	14
Abbildung 4:	Einfaches Modell einer Bedrohungsanalyse	17
Abbildung 5:	Detailliertes Modell einer Bedrohungsanalyse	17
Abbildung 6:	Aktivitäten der Phase 3 "Implementierung".....	20
Abbildung 7:	Aktivitäten der Phase 4 "Testen"	22
Abbildung 8:	Statische Code Scanner.....	27
Abbildung 9:	Aktivitäten der Phase 5 "Auslieferung und Betrieb"	30

Tabellenverzeichnis

Tabelle 1:	Schutzbedarfskategorien.....	10
Tabelle 2:	Schutzbedarfsfeststellung für Anwendung.....	11
Tabelle 3:	Access Control Matrix.....	15
Tabelle 4:	Zugangspunkte einer Applikation.....	17
Tabelle 5:	Gegenüberstellung von STRIDE-Modell und Schutzzielen.....	18
Tabelle 6:	Beispiel für einen Security Test Case.....	23
Tabelle 7:	Unit Test Dokumentation.....	25

1 Einleitung, Motivation und Ziele

Eine Vielzahl von Entwicklungsvorhaben im Bereich der Webanwendungen wird von Stellen der öffentlichen Verwaltung initiiert. Das ausgesprochene Ziel, die IT-Sicherheit in solchen Projekten signifikant zu verbessern, wird dadurch erreicht, dass mit diesem Dokument dem Auftraggeber eine Sammlung von Anforderungen mit sicherheitsrelevantem Bezug bereitgestellt wird. Anhand dieser können die Verantwortlichen die Einhaltung der Vorgaben überprüfen und diese als Teil der Vergabeunterlagen für den Auftragnehmer vertragsrelevant machen.

1.1 Darstellung des gewählten Ansatzes

Der Leitfaden beginnt mit den Grundlagen sicherer Webanwendungen. Das Kapitel dient der Einleitung in den Themenkomplex. Die eigentliche Behandlung der Thematik – insbesondere im Hinblick auf den Secure Development Lifecycle – wird im BSI-Dokument „Leitfaden zur Entwicklung sicherer Webanwendungen. Empfehlungen und Anforderungen an die Auftragnehmer“¹ abgehandelt. Das erwähnte BSI-Dokument ist die Grundlage für den vorliegenden Leitfaden und ist somit eine Voraussetzung für das Verständnis.

Der Kerninhalt dieses Dokuments beginnt mit der Vergabephase. Die Vergabephase orientiert sich an den Vorgaben des Bundesministeriums des Innern und dem von ihnen veröffentlichten Dokument „Unterlage für Ausschreibung und Bewertung von IT-Leistungen“² (kurz UfAB). Das Dokument definiert Eignungskriterien und Leistungskriterien. Eignungskriterien müssen unbedingt erfüllt werden, um als potenzieller Auftragnehmer für die Aufgabenerfüllung geeignet zu sein. Eignungskriterien werden in die Kategorien Fachkunde, Leistungsfähigkeit und Zuverlässigkeit bzw. Gesetzestreue unterteilt. Solche Eignungskriterien, wie beispielsweise eine Mitarbeiterqualifikation der Kategorie Fachkunde, ermöglichen es dem Auftraggeber eine Vorauswahl der Bewerber zu treffen. Nach dieser Vorauswahl werden die verbliebenen Bewerber, unabhängig von Eignungskriterien, anhand von Leistungskriterien bewertet. Leistungskriterien sollen es dem Auftraggeber u.a. ermöglichen, den Softwareentwicklungsprozess von Bewerbern bewerten zu können. In Kapitel 4 *Überprüfung der Vorgaben an den Entwicklungsprozess* werden am Ende jeder Phase mögliche Leistungskriterien vorgeschlagen. Anhand dieser Liste kann der Auftraggeber für jede Ausschreibung einen projektspezifischen Anforderungskatalog für Bewerber erstellen und in der Folge noch vor der Vergabe an einen Auftragnehmer prüfen. Während der Umsetzungsphase ermöglichen es die in Kapitel 4 definierten Quality Gates dem Auftraggeber einerseits die konkrete und vollständige Umsetzung der gestellten Anforderungen zu prüfen und stellen gleichzeitig mögliche Bewertungskriterien für die Abnahme der jeweiligen Phase dar. Für eine detailliertere Beschreibung der Aufgaben des Auftragnehmers kann das BSI-Dokument „Leitfaden zur Entwicklung sicherer Webanwendungen. Empfehlungen und Anforderungen an die Auftragnehmer“ verwendet werden.

Zum Schluss werden dem Auftraggeber in den Kapiteln 5.1 und 5.2 Checklisten zur Verfügung gestellt, um den Auftraggeber bei der Bewertung der Auftragnehmer zu unterstützen. Die Checklisten sind als Orientierungshilfe gedacht und dem Auftraggeber steht es frei, bei Bedarf die Checklisten zu modifizieren bzw. zu erweitern.

1 [BSI 2013]

2 [UfAB2010]

2 Grundlagen sicherer Webanwendungen

Dieses Kapitel dient als Einleitung in den Themenkomplex und verdeutlicht, warum die Sicherheit bei Webanwendungen nicht vernachlässigt werden darf. Zu Beginn werden die allgemeinen Risiken von Webanwendungen und die Risiken unterschiedlicher Webtechnologien beleuchtet. Abgeschlossen wird dieses Kapitel mit der Erläuterung der Notwendigkeit eines Secure Software Development Lifecycles (SDL).

2.1 Risiken von Webanwendungen

Webanwendungen sind hochgradig komplex, immer und von überall verfügbar und stellen über das Internet oft kritische Daten und Dienste bereit. Sie sind das Tor zum Unternehmen und seine IT-Infrastruktur, was sie zu einem beliebten Angriffsziel macht. SQL Injections und Cross-Site-Scripting (CSS), um nur einige Angriffsvektoren zu nennen, sind für einige der größten Vorfälle in den letzten Jahren verantwortlich, bei den sensiblen Daten gestohlen wurden. Die Gründe hierzu sind vielschichtig – von fehlerhafter Implementierung der Anwendung selbst oder in den Erweiterungen vom Dritthersteller³, über unkorrekte Konfiguration der Backend-Systeme bis hin zum mangelnden Sicherheitsbewusstsein.

2.2 Allgemeine Risiken des Einsatzes unterschiedlicher Webtechnologien

Seit dem Bestehen des World Wide Web versorgt es uns mit vielen Diensten. Beginnend bei einfach gestalteten, starren Webseiten für den reinen Informationsaustausch, über hochdynamische und grafisch aufwendige multimediale Angebote, bis hin zu hochkomplexen, über mehrere Kontinente verstreute Geschäftsanwendungen. Leider werden Sicherheitsthemen aber nach wie vor vernachlässigt und finden erst dann Beachtung, wenn es zu spät ist. Um konkret auf einige Probleme einzugehen, werden im Folgenden einige Bereiche näher betrachtet und auf mögliche Lösungen verwiesen.

2.2.1 Mobile Web

Unter Mobile Web versteht man den Zugang zum Internet durch die Benutzung eines mobilen Endgerätes. Die hierzu verwendeten Endgeräte sind in der Regel ein Smartphone oder ein Tablet-PC.

Durch die Verlagerung des Webbrowsers von dem klassischen PC auf die mobilen Endgeräte stehen die Entwickler neuen Herausforderungen gegenüber. Mobile Endgeräte haben in der Regel einen kleineren Bildschirm, weshalb die Darstellung der Webseite problematisch sein kann. In der Anfangsphase haben viele Hersteller deshalb eine explizite mobile Webseite zur Verfügung gestellt. Der neue Trend, dem viele Unternehmen folgen, ist die explizite Herstellung von Applikation (umgangssprachlich „Apps“ genannt) für mobile Endgeräte. Dabei stehen das optische Design und die Benutzerfreundlichkeit an erster Stelle, wobei die Sicherheit der Applikation vernachlässigt wird.

Auch wenn Applikationen für mobile Plattformen in diesem Dokument nicht explizit betrachtet werden, sollten diese die gleichen Sicherheitsanforderungen erfüllen, die auch von konventionellen Webanwendungen erwartet werden.

3 In [BSI_CMS] wird festgestellt, dass ein Großteil der Schwachstellen in Content Management Systeme in den Erweiterungsmodulen für diesen zu finden sind.

2.2.2 Web-Services

Ein Web-Service ist ein Softwaresystem, das entwickelt wurde, um eine interoperable Maschine-zu-Maschine Kommunikation zu ermöglichen⁴. Web-Services werden als eine logische Weiterentwicklung des World Wide Web betrachtet und unterstützen die Zusammenarbeit verschiedener Anwendungen. Da auch Web-Services vertrauliche Daten verarbeiten bzw. diese übertragen, müssen entsprechende Sicherheitsaspekte berücksichtigt werden.

WS-Security ist ein Standard aus der Linie der WS-* -Spezifikationen und wurde dazu entwickelt, um Sicherheitsaspekte hinsichtlich Integrität und Vertraulichkeit im Rahmen der Kommunikation sicherzustellen.

2.2.3 Cloud Computing

Cloud Computing beschreibt den Ansatz, verschiedene IT-Dienste oder -Infrastrukturen (z.B. Rechenleistung, Datenspeicher, Software etc.) bei Bedarf aus dem Internet zu beziehen. Für den Benutzer steht der bezogene IT-Dienst im Vordergrund, ohne die dahinterliegende Infrastruktur zu kennen bzw. kennen zu müssen.

Vor allem auf Klein- und Mittelunternehmen, aber auch Privatpersonen zielen die Anbieter von Cloud Computing Diensten ab, da sich diese meist die Kosten für die dahinterliegenden Systeme, die für den Betrieb des IT-Dienstes notwendig sind, nicht leisten können. Jedoch zögern viele Unternehmen noch, wenn es darum geht, unternehmensrelevante Informationen außerhalb der Organisation zu speichern. Dieses Zögern bestätigt auch eine Studie von Deloitte⁵ aus dem Jahr 2011, wonach 74% aller befragten Unternehmen angeben, dass Cloud Computing derzeit nicht Teil ihrer Strategie ist bzw. der Status unbekannt ist. Oft ist es unklar, wie die Informationen innerhalb der Cloud gespeichert, verarbeitet und übermittelt werden. Dementsprechend müssen folgende Sicherheitsaspekte berücksichtigt und gelöst werden, um das Vertrauen in Cloud Computing zu steigern:

- Zugriffsregelung auf die gespeicherten Daten
- Sichere Speicherung der Daten
- Sicherer Transport der Daten
- Sichere Löschung der Daten

2.3 Secure Development Lifecycle

Der Secure Software Development Lifecycle wird im BSI-Dokument „Leitfaden zur Entwicklung sicherer Webanwendungen. Empfehlungen und Anforderungen an die Auftragnehmer“⁶ ausführlich beschrieben. An dieser Stelle sei erwähnt, dass er die Grundlage sicherer Webanwendungen bildet. Je früher ein Fehler gefunden wird, desto geringer sind die Behebungskosten. Fehler in der Designphase können relativ einfach behoben werden. Wird ein Fehler jedoch erst in der Betriebsphase entdeckt, so kann dies mit sehr hohen Kosten verbunden sein und auch potenziell einen völligen Neustart der Entwicklung bedeuten.

Um den Reifegrad der Entwicklung bewerten zu können, wird im BSI-Dokument „Leitfaden zur Entwicklung sicherer Webanwendungen. Empfehlungen und Anforderungen an die Auftragnehmer“⁷ ein Reifegradmodell vorgestellt und erläutert. Das Reifegradmodell ermöglicht es dem Auftraggeber, den Auftragnehmer nach einer definierten Vorgehensweise objektiv zu bewerten.

4 [W3C 2004]

5 [Delo 2011]

6 [BSI 2013]

7 [BSI 2013]

3 Vergabephase

Die Vergabephase legt neben den im Dokument UfAB⁸ definierten Anforderungen die notwendigen Schritte fest, die vor bzw. während der Ausschreibung zu berücksichtigen sind.

3.1 Ermittlung des Schutzbedarfs

Noch vor der Vergabephase sind vom Auftraggeber die durch die Anwendung zu verarbeitenden Daten zu definieren und deren Schutzbedarf festzulegen⁹. Der Auftragnehmer erhält so die Möglichkeit, die erforderlichen Aufwände für die Software Assurance¹⁰ entsprechend frühzeitig zu planen.

Das Ziel der Schutzbedarfserhebung ist es zu ermitteln, in welchem Ausmaß die Webapplikation in Bezug auf Vertraulichkeit, Integrität und Verfügbarkeit geschützt werden muss, um in späteren Phasen daraus die Sicherheitsanforderungen, Maßnahmen sowie Audit-Pläne ableiten zu können. Die Bewertung wird in diesem Leitfaden bewusst kompakt aufgebaut, um die Schutzbedarfsbestimmung auch auf verhältnismäßig kleinen Applikationen anwendbar zu machen.

Die Grundwerte stellen jene Faktoren dar, die von Angriffen bedroht werden. In der Informationssicherheit sind drei fundamentale Grundwerte vorhanden: Vertraulichkeit, Integrität und Verfügbarkeit.

- **Vertraulichkeit** ist der Schutz vor unbefugter Preisgabe von Informationen. Das Belauschen, eine Weitergabe oder Veröffentlichung von vertraulichen Informationen ist daher nicht erwünscht.
- Die **Integrität** von Daten gewährleistet deren Vollständigkeit und Unversehrtheit über einen bestimmten Zeitraum und ermöglicht die Aufdeckung einer Manipulation.
- **Verfügbarkeit** kann sich auf IT-Systeme oder Daten beziehen und fordert, dass sie zu definierten Zeiten, im Einklang mit der Vertraulichkeit und der Integrität, von Anwendern stets wie vorgesehen genutzt werden können.

Die Grundwerte beziehen sich nicht ausschließlich auf Daten, sondern sollten ebenfalls auf Datenflüsse, Kommunikationskanäle oder ganze IT-Systeme umgelegt werden.¹¹

Die im Einsatz befindlichen Webanwendungen besitzen unterschiedliche Schutzbedürfnisse. Da diese in der Praxis meist nicht quantifizierbar sind, werden sie in drei Kategorien eingeteilt: „Normal“, „Hoch“ und „Sehr hoch“. Diese Definition entspricht den Begriffen des IT-Grundschutzes des BSI. Unter einem normalen Schutzbedarf sind Schadensauswirkungen begrenzt und überschaubar, während unter einem hohen bzw. sehr hohen Schutzbedarf die Schadensauswirkungen beträchtlich bzw. existenziell bedrohlich sein können.

Diese Kategorien dienen als Entscheidungsgrundlage, welche Sicherheitsanforderungen für die jeweilige Webanwendung getroffen werden sollten.

Um die Zuordnung zu einer der Kategorien zu erleichtern, können verschiedene Schadensszenarien betrachtet werden (z.B. ein Verstoß gegen Gesetze oder Vorschriften). Die Schadensszenarien fassen Schäden zusammen, die beim Eintreten einen Verlust eines Schutzzieles bedeuten. Die Webanwendung muss mindestens hinsichtlich der folgenden Schadensszenarien bewertet werden. Darüber hinaus – wenn notwendig – können weitere Schadensszenarien definiert oder es müssen die individuellen Gegebenheiten berücksichtigt werden.

8 [UfAB2010]

9 An dieser Stelle sei auch auf den BSI-Standard 100-2 [BSI 2008] verwiesen, der in Kapitel 4.3 eine Vorgehensweise zur Schutzbedarfsfeststellung für IT-Systeme vorschlägt, auf dem der hier dargestellte Ansatz basiert.

10 Software Assurance wird wie folgt definiert: „Level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle and that the software functions in the intended manner.“ [CNSS2010]

11 Nach [ISO27001] sollte alles, „that has value to the organization“, vor Gefahren geschützt werden. Dies bezeichnet der Standard als Asset. Assets sind Informationen, Fachanwendungen, Software, Hardware, Kommunikationsverbindungen und Infrastrukturkomponenten.

Schadensszenario	Schutzbedarf
Verstoß gegen Gesetze/ Vorschriften/Verträge	
Verstöße gegen Vorschriften und Gesetze mit geringfügigen Konsequenzen. Geringfügige Vertragsverletzungen mit maximal geringen Konventionalstrafen.	Normal
Verstöße gegen Vorschriften und Gesetze mit erheblichen Konsequenzen. Vertragsverletzungen mit hohen Konventionalstrafen.	Hoch
Fundamentaler Verstoß gegen Vorschriften und Gesetze. Vertragsverletzungen, deren Haftungsschäden ruinös sind.	Sehr hoch
Beeinträchtigung des informationellen Selbstbestimmungsrechts	
Es handelt sich um personenbezogene Daten, durch deren Verarbeitung der Betroffene in seiner gesellschaftlichen Stellung oder in seinen wirtschaftlichen Verhältnissen beeinträchtigt werden kann.	Normal
Es handelt sich um personenbezogene Daten, bei deren Verarbeitung der Betroffene in seiner gesellschaftlichen Stellung oder in seinen wirtschaftlichen Verhältnissen erheblich beeinträchtigt werden kann.	Hoch
Es handelt sich um personenbezogene Daten, bei deren Verarbeitung eine Gefahr für Leib und Leben oder die persönliche Freiheit des Betroffenen gegeben ist.	Sehr hoch

Schadenskategorie	Schutzbedarf
Beeinträchtigung der persönlichen Unversehrtheit	
Eine Beeinträchtigung erscheint nicht möglich.	Normal
Eine Beeinträchtigung der persönlichen Unversehrtheit kann nicht absolut ausgeschlossen werden.	Hoch
Gravierende Beeinträchtigungen der persönlichen Unversehrtheit sind möglich. Gefahr für Leib und Leben.	Sehr hoch
Beeinträchtigung der Aufgabenerfüllung	
Die Beeinträchtigung würde von den Betroffenen als tolerabel eingeschätzt werden. Die maximal tolerierbare Ausfallzeit ist größer als 24 Stunden.	Normal
Die Beeinträchtigung würde von einzelnen Betroffenen als nicht tolerabel eingeschätzt. Die maximal tolerierbare Ausfallzeit liegt zwischen einer und 24 Stunden.	Hoch
Die Beeinträchtigung würde von allen Betroffenen als nicht tolerabel eingeschätzt werden. Die maximal tolerierbare Ausfallzeit ist kleiner als eine Stunde.	Sehr hoch
Negative Innen- oder Außenwirkung	
Eine geringe bzw. nur interne Ansehens- oder Vertrauensbeeinträchtigung ist zu erwarten.	Normal
Eine breite Ansehens- oder Vertrauensbeeinträchtigung ist zu erwarten.	Hoch
Eine landesweite Ansehens- oder Vertrauensbeeinträchtigung, eventuell sogar existenzgefährdender Art, ist denkbar.	Sehr hoch
Finanzielle Auswirkungen	
Der finanzielle Schaden bleibt für die Institution tolerabel.	Normal
Der Schaden bewirkt beachtliche finanzielle Verluste, ist jedoch nicht existenzbedrohend.	Hoch
Der finanzielle Schaden ist für die Institution existenzbedrohend.	Sehr hoch
Schutzbedarf:	

Tabelle 1: Schutzbedarfskategorien¹²

Nach dem die Schutzbedarfskategorien definiert wurden, muss die Webanwendung hinsichtlich der Schutzziele bewertet werden. Das Vorgehen ist in [BSI 2008] ausführlich beschrieben.

12 [BSI 2008]

Anwendung		Schutzbedarfsfeststellung		
Bezeichnung	Pers. Daten	Grundwert	Schutzbedarf	Begründung
		Vertraulichkeit		
		Integrität		
		Verfügbarkeit		

Tabelle 2: Schutzbedarfsfeststellung für Anwendung¹³

Der Schutzbedarf einer Webanwendung ergibt sich durch den am höchsten auftretenden Schutzbedarf eines Schadensszenarios. Der erhobene Schutzbedarf hat umfassende Auswirkungen auf die Sicherheitsanforderungen in den weiteren Phasen.

3.2 Ausschreibung auf Basis des Schutzbedarfs

Durch ein Ausschreibungsverfahren werden potenzielle Auftragnehmer aufgefordert, ein Angebot abzugeben. Der Auftraggeber sollte innerhalb der Ausschreibung dem Bewerber u.a. folgende Informationen zur Verfügung zu stellen:

- Schutzbedarf der Anwendung
- funktionale Anforderungen
- funktionale Sicherheitsanforderungen
- nicht funktionale Anforderungen
- nicht funktionale Sicherheitsanforderungen
- Der Auftraggeber muss definieren, zumindest auf hohem Abstraktionsniveau, welche Qualifikationen er sich von den Projektbeteiligten erwartet.

Von einem potenziellen Auftragnehmer können folgende Angaben gefordert werden:

- Angaben zu den Personen, die an dem Projekt beteiligt sein werden. Dazu zählen aber nicht nur die Softwareentwickler, sondern auch beispielsweise die Personen, die für die Projektleitung oder für die Qualitätssicherung zuständig sind. Zu jeder Person ist jeweils ein Qualifikationsprofil anzugeben. Das Qualifikationsprofil soll darstellen, warum die Person für die eingesetzte Aufgabe geeignet ist. Das Qualifikationsprofil soll beispielsweise die Ausbildung (Schulbildung, Weiterbildung, Zertifikate etc.), die bislang erlangte Berufserfahrung, Publikationen, Vorträge und Referenzprojekte beinhalten.
- Angaben zur Erfüllung der Anforderungen aus dem Dokument „Leitfaden zur Entwicklung sicherer Webanwendungen. Empfehlungen und Anforderungen an die Auftragnehmer“¹⁴ hinsichtlich des Entwicklungsprozesses, insbesondere Angaben über den vom Auftragnehmer erreichten Reifegrad (siehe Kapitel 4.1.1 *Reifegrade*). In der Regel sollte mindestens der Reifegrad „Vollständig“ erfüllt werden. Ist.

13 [BSI 2008]

14 [BSI 2013]

4 Überprüfung der Vorgaben an den Entwicklungsprozess

4.1 Aufbau

Der Softwareentwicklungsprozess wird in fünf Phasen „Initiale Planung & Vergabe“, „Konzeptions- & Planungsphase“, „Implementierung“, „Testen“ und „Auslieferung & Betrieb“, wie in Abbildung 1 dargestellt, unterteilt.



Abbildung 1: Entwicklungsprozess

In den folgenden Kapiteln wird grob beschrieben, welche Aufgaben vom Auftragnehmer in der jeweiligen Phase durchzuführen sind und warum diese sicherheitsrelevant sind. Eine detaillierte Beschreibung der Aufgaben des Auftragnehmers kann dem Dokument „Leitfaden zur Entwicklung sicherer Webanwendungen. Empfehlungen und Anforderungen an die Auftragnehmer“¹⁵ entnommen werden. Am Ende jeder Phase wird ein „Quality Gate“ für den Auftraggeber definiert, in dem beschrieben wird, welche Ergebnisse vom Auftraggeber zum Ende der Phase geprüft werden können. Die im „Quality Gate“ beschriebenen Anforderungen sind aber stark abhängig vom Schutzbedarf der zu entwickelnden Anwendung und nicht immer vollständig umzusetzen. Eine detaillierte Auflistung der notwendigen Qualitätskriterien jeder Phase kann Kapitel 5.3 *Checkliste für den Entwicklungsprozess* entnommen werden.

4.1.1 Reifegrade

Durch viele verschiedene Möglichkeiten in der Softwareentwicklung, unterschiedliche Sicherheitsanforderungen und unterschiedliche Sicherheitsniveaus, ist es schwierig, die Sicherheit einer Webanwendung zu bewerten. Um eine aussagekräftige und vor allem vergleichbare Bewertung der Sicherheit einer Anwendung zu ermöglichen, werden von bekannten Studien wie BSIMM¹⁶ und OpenSAMM¹⁷ Reifegrade verwendet. Die Idee hinter der Einführung eines solchen Systems ist es, dass für jede Phase des Entwicklungsprozesses, Reifegrade definiert werden, die mit unterschiedlichen Aktionen und Sicherheitsanforderungen verknüpft sind. Je nach festgelegtem Schutzbedarf einer Webanwendung können somit unterschiedliche Reifegrade für den Entwicklungsprozess vorgeschrieben werden.

Während den folgenden Phasen des Entwicklungsprozesses ermöglicht es ein Reifegradmodell dem Auftraggeber sicherzustellen, dass vor der Entwicklung definierte Sicherheitsanforderungen vom Auftragnehmer umgesetzt werden. Die vom Auftragnehmer durchzuführenden Aktionen werden mit Checklisten in Kapitel 5.3 *Checkliste für den Entwicklungsprozess* festgelegt. Eine Bewertung erfolgt anhand der definierten Reifegrade.

0. Nicht vorhanden

Aktivitäten wurden nicht umgesetzt.

(0% der Punktzahl für verpflichtende Aktivitäten wurden erreicht)

15 [BSI 2013]

16 [BSIMM 2012]

17 [OSAMM]

1. Ad-Hoc (Informell)

Die Aktivitäten wurden teilweise umgesetzt.

(mehr als 0% und weniger als 50% der Punkteanzahl für verpflichtende Aktivitäten wurden erreicht)

2. Partiiell

Die Aktivitäten wurden größtenteils 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.

Während des Entwicklungsprozesses muss der Auftragnehmer in der Regel mindestens den Reifegrad „Vollständig“ erfüllen, damit ein Mindestsicherheitsniveau der Webanwendung sichergestellt ist.

Phase 1: Initiale Planung & Vergabephase

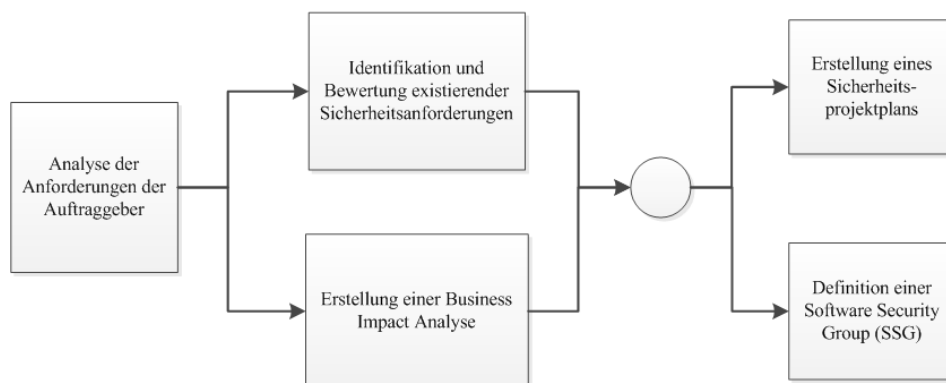


Abbildung 2: Aktivitäten der Phase 1 "Initiale Planung und Vergabephase"

4.2 Phase 1: Initiale Planung & Vergabephase

Basierend auf dem Schutzbedarf der Applikation müssen die bereits vom Auftraggeber spezifizierten Sicherheitsanforderungen gemeinsam mit Auftragnehmer auf ihre Vollständigkeit hin überprüft werden. Die Anforderungen werden anschließend in einem Sicherheitsprojektplan gesammelt.

4.2.1 Leistungskriterien

- Stellen Sie dar, wie die Anforderungen der zu entwickelnden Software erhoben werden.
- Beschreiben Sie, wie die möglichen Auswirkungen der Verlust von den Grundwerten für den Auftraggeber ermittelt werden.
- Gibt es einen Ansprechpartner für das Thema Sicherheit?
- Findet vor dem Start der Implementierung ein Security Kickoff statt?
- Wie gehen Sie mit Secure Coding Guidelines um?
 - Sind Guidelines vorhanden?
 - Auf welchen Standards, Guidelines etc. basieren diese?
 - Werden die Guidelines regelmäßig überarbeitet und aktualisiert?

4.2.2 Quality Gate: Prüfbare Ergebnisse der Phase

Am Ende dieser Phase muss vom Auftragnehmer ein Sicherheitsprojektplan erstellt worden sein, der einen groben Überblick über die geplanten sicherheitsrelevanten Schritte gibt. Der Sicherheitsprojektplan wird auf Basis des Schutzbedarfs und den Anforderungen des Auftraggebers erstellt. Der Auftraggeber kann im Zuge dieses Quality Gates die folgenden Punkte prüfen:

- Spezifikationsdokument: Werden alle Anforderungen abgedeckt?
 - Softwareeigenschaften
 - Funktionale und nicht funktionale Anforderungen
 - Funktionale und nicht funktionale Sicherheitsanforderungen
 - Dokumentationsanforderungen
- Werden alle gesetzlichen Vorgaben eingehalten?
- Werden entsprechende Coding Guidelines verwendet?
- Wird vom Auftragnehmer ein Security-Kickoff durchgeführt?
- Wurde ein Ansprechpartner für das Thema Sicherheit benannt?

4.3 Phase 2: Konzeption & Planung

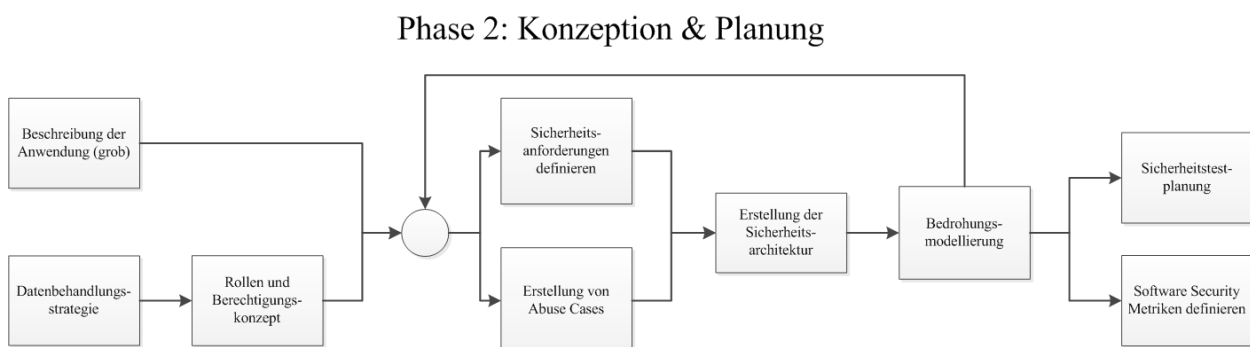


Abbildung 3: Aktivitäten der Phase 2 "Konzeption & Planung"

4.3.1 Datenflussanalyse

Mit einem Datenflussdiagramm wird der Datenfluss innerhalb einer Software dargestellt. Dabei werden sowohl Datenflüsse zwischen Benutzern und Systemen als auch passive Ressourcen (wo werden Daten gespeichert) aufgezeigt. Es wird festgelegt, wie Daten in der Anwendung verarbeitet werden (gespeichert, übertragen, angezeigt) und wie die Daten klassifiziert werden. In der Folge können im Zuge der Bedrohungsanalyse im Datenflussdiagramm Trust Boundaries (Vertrauensgrenzen) eingezeichnet werden und für jede Schnittstelle und jeden Übergang Angriffe identifiziert werden. Eine Datenflussanalyse hilft also bei der strukturierten Erhebung von Bedrohungen. Gleichzeitig ist aber dabei zu beachten, dass dabei nicht alle Bedrohungen (z.B. Denial of Service) abgedeckt werden.

4.3.2 Rollen- und Berechtigungskonzept

Durch ein Rollen- und Berechtigungskonzept soll dargestellt werden, welche Benutzer bzw. welche Benutzergruppe mit welchen Rechten auf Assets und Funktionen zugreifen dürfen. Die Darstellung erfolgt üblicherweise in einer Access Control Matrix. Eine einfache Darstellung zeigt Tabelle 3.

	Assets				
		Asset 1	Asset 2	Asset3	...
Benutzerrolle	Projektmanager	RW	-	RW	
	Entwickler	R	RWX	-	
	Tester	-	R	R	
	...				

Tabelle 3: Access Control Matrix

R = Lesen

W = Schreiben

X = Ausführen

4.3.3 Abuse Cases

Ziel von Abuse Cases ist es, das Verhalten des Systems bei Missbrauch bzw. Angriffen zu beschreiben. Die definierten Abuse Cases können in der Folge auch dazu verwendet werden, um nicht funktionale Sicherheitsanforderungen abzuleiten. Das Beispiel „Jeder Benutzer eines Systems kann auf alle Einstellungen zugreifen“ könnte zur Anforderung „Einführung eines Berechtigungssystems“ führen. Wichtig bei der Erstellung von Abuse Cases ist es, vorher zu ermitteln, welcher Bereich das größte Angriffspotential bietet, da niemals alle Abuse Cases abgedeckt werden können.

4.3.4 Bedrohungsmodellierung

Die Bedrohungsmodellierung (engl. Threat Modeling) ist eines der zentralen Konzepte der sicheren Softwareentwicklung und unterstützt bei der Ermittlung des Schutzbedarfs und möglicher Bedrohungen der Software. Des Weiteren stellt eine Bedrohungsmodellierung die einzige Aktivität dar, mit der sicherheitsrelevante Designfehler erkannt und adressiert werden. Auf Basis dieser Bedrohungsmodellierung können Gegenmaßnahmen und Testanforderungen abgeleitet werden, die wieder in die Sicherheitsarchitektur miteinfließen. Mit der systematischen Ermittlung und Analyse der möglichen Bedrohungen sollte so früh wie möglich im Softwareentwicklungsprozess begonnen werden, da die gewonnenen Erkenntnisse Auswirkungen auf alle Folgeaktivitäten haben können. Ein wichtiger Punkt bei der Bedrohungsmodellierung ist, dass sowohl Auftragnehmer als auch Auftraggeber gemeinsam daran arbeiten. Nur durch die Zusammenarbeit beider Parteien kann sichergestellt werden, dass alle sicherheitsrelevanten Anforderungen im Konzept berücksichtigt werden. Die Bedrohungsmodellierung baut dabei auf die in den vorangehenden Kapiteln 4.3.1 – 4.3.3 auf und verwendet die in diesen Schritten ermittelten Informationen.

Generell besteht eine Bedrohungsmodellierung aus folgenden Phasen:

- **Assets und Benutzer identifizieren:**
Im ersten Schritt werden alle schützenswerten Objekte identifiziert. Als schützenswertes Objekt gilt alles, das einen Wert für den Auftraggeber darstellt (z.B. Kundendaten, Image etc.). Anschließend werden die involvierten Akteure identifiziert und festgelegt, welche Operationen durch diese erfolgen dürfen. Für diesen Schritt wird nur die Sichtweise des Herstellers bzw. der Kunden benötigt.
- **Erstellen einer Architektur:**
Nachdem alle Assets und Akteure identifiziert wurden, wird eine High-Level Architektur der Software erstellt. Als zusätzlicher Input dienen Datenfluss- und

Datenhaltungs-Diagramme. Sobald die genannten Komponenten miteinander verknüpft wurden, wird das Modell um Vertrauensgrenzen (engl. Trust Boundaries) erweitert. Eine Vertrauensgrenze ist dabei ein Übergang zwischen zwei Komponenten, die sich nicht blind vertrauen und an denen sichergestellt werden muss, dass sich beide Seiten vertrauen können. Je nach Größe und Komplexität der Architektur kann das System anschließend entlang der Vertrauensgrenzen unterteilt und detaillierter analysiert werden. Zusätzlich sollten alle möglichen Zugangspunkte für Benutzer und Angreifer definiert werden. Tabelle 4 zeigt ein Beispiel für die ermittelten Zugangspunkte zu Portalfunktionen. Die ID würde in diesem Beispiel dazu verwendet, um eine Baumstruktur der möglichen Zugangspunkte zu der Applikation zu bilden. Die ID kann in der Folge dazu verwendet werden, um Bedrohungen und Sicherheitsmaßnahmen den entsprechenden Zugangspunkten im Threatmodel zuordnen zu können.

Zugangspunkt			
ID	Name	Beschreibung	Benutzer mit Zugang
1	HTTPS Port	Alle Seiten des Portals können nur über SSL/TLS erreicht werden.	Anonyme Benutzer Benutzer mit gültigen Zugangsdaten Benutzer mit ungültigen Zugangsdaten
1.1	Start-Seite	Start-Seite wird jedem Besucher als Erstes präsentiert.	Anonyme Benutzer Benutzer mit gültigen Zugangsdaten Benutzer mit ungültigen Zugangsdaten
1.2	Login-Seite	Um auf Informationen zugreifen zu können, ist ein Login erforderlich.	Anonyme Benutzer Benutzer mit gültigen Zugangsdaten Benutzer mit ungültigen Zugangsdaten
1.2.1	Login-Funktion	Die Login-Funktion vergleicht eingegebene Benutzerdaten mit in der Datenbank gespeicherten Daten.	Benutzer mit gültigen Zugangsdaten Benutzer mit ungültigen Zugangsdaten
1.3	Such-Seite	Seite ermöglicht den Zugriff auf die Suchfunktion.	Benutzer mit gültigen Zugangsdaten
1.3.1	Such-Funktion	Funktion zum Senden von Suchanfragen.	Benutzer mit gültigen Zugangsdaten
1.4	Administrator-Seite	Seite zum Administrieren des Portals.	Benutzer mit gültigen Zugangsdaten und Administratorberechtigungen

Tabelle 4: Zugangspunkte einer Applikation¹⁸

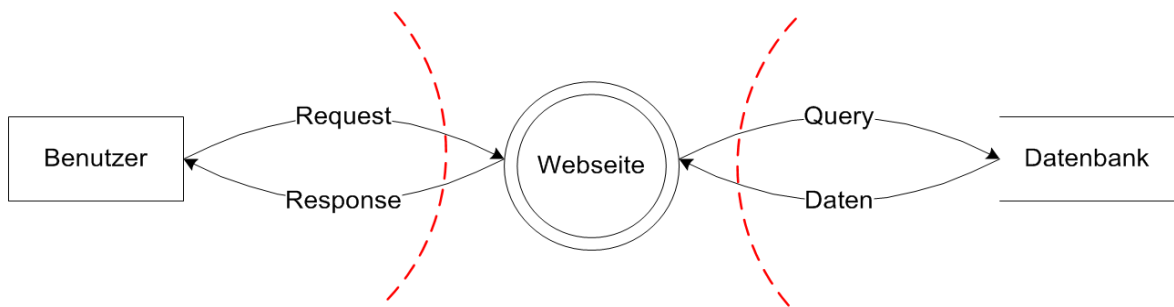


Abbildung 4: Einfaches Modell einer Bedrohungsanalyse

18 [OTHM]

Abbildung 4 stellt ein einfaches Modell für eine Datenabfrage dar. Die beiden Vertrauensgrenzen zeigen dabei, dass aus Sicht der Webseite den Benutzern und der Datenbank nicht blind vertraut werden kann und diese ohne weitere Sicherheitsmaßnahmen als unsicher eingeschätzt werden müssen. Für die Analyse der Bedrohungen sollte nun der Prozess „Webseite“ entlang der Vertrauensgrenzen weiter zerlegt werden. Abbildung 5 zeigt das mögliche Ergebnis, das anschließend als Ausgangspunkt für die Ermittlung der Bedrohungen dient.

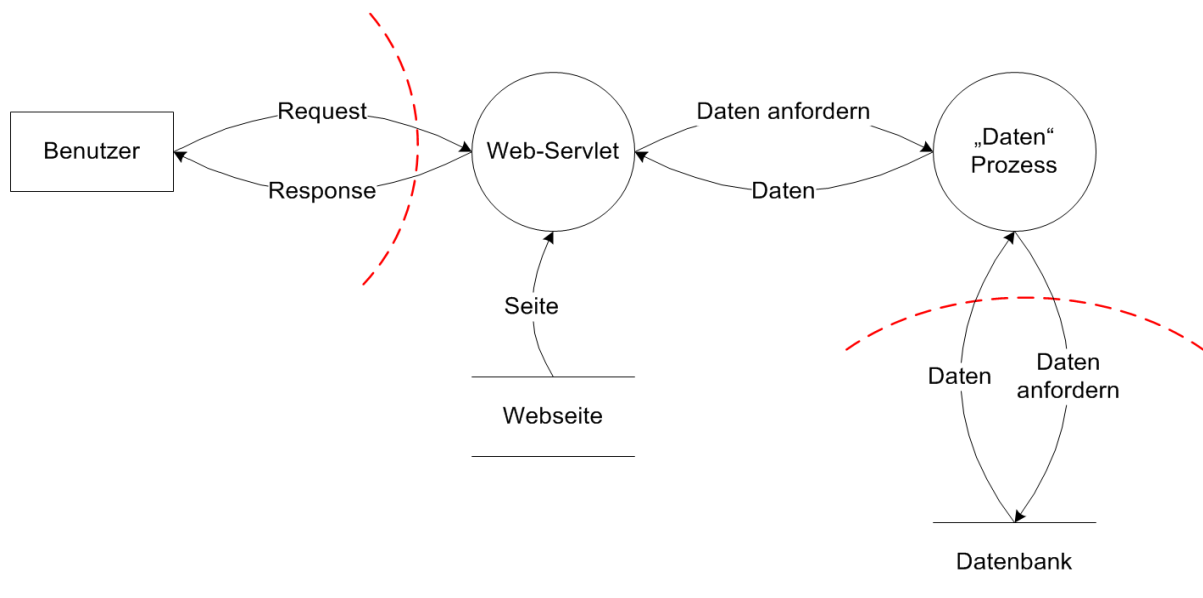


Abbildung 5: Detailliertes Modell einer Bedrohungsanalyse

- Bedrohungen identifizieren:**
 In der wichtigsten Phase der Bedrohungsmodellierung gilt es, sich in die Lage eines potenziellen Angreifers zu versetzen und mögliche Angriffsvektoren zu identifizieren. Da alle möglichen Angriffe niemals vollständig behandelt werden können, empfiehlt es sich, die Analyse nach Angriffsmustern durchzuführen. Eine häufig verwendete Klassifizierung stellt das von Microsoft entwickelte STRIDE-Modell dar. Mit diesem Modell werden Bedrohungen nach den entsprechenden Schutzzielen klassifiziert. Tabelle 5 zeigt die Gegenüberstellung von Bedrohung und Schutzziel anhand des STRIDE-Modells.

Bedrohung	Schutzziel
Spoofing	Authentifizierung
Tampering	Integrität
Repudiation	Nicht-Abstreitbarkeit
Information Disclosure	Vertraulichkeit
Denial of Service	Verfügbarkeit
Elevation of Privilege	Autorisierung

Tabelle 5: Gegenüberstellung von STRIDE-Modell und Schutzzielen

- Bedrohungen dokumentieren:**
 Nach der Identifikation der Bedrohungen ist es wichtig, diese zu dokumentieren, um in späteren Entwicklungsphasen nachvollziehen zu können, ob Tests gegen erkannte Bedrohungen durchgeführt wurden und ob getroffene Gegenmaßnahmen ausreichend sind.

- **Bedrohungen bewerten:**
In den seltensten Fällen ist es möglich, alle erkannten Bedrohungen vollständig mit Maßnahmen abzudecken. Aus diesem Grund ist es notwendig, gefundene Bedrohungen zu bewerten, um vorhandene Ressourcen dafür zu verwenden, um Maßnahmen für die kritischsten Bedrohungen zu definieren.
- **Gegenmaßnahmen definieren:**
In der letzten Phase der Bedrohungsmodellierung werden entsprechend der Priorisierung Gegenmaßnahmen für erkannte Bedrohungen ermittelt. Neue Gegenmaßnahmen und Sicherheitsanforderungen werden anschließend wieder in die Sicherheitsarchitektur integriert. Hilfestellung bei der Identifikation von Gegenmaßnahmen können u.a. die „ASF Threat & Countermeasures List“¹⁹ oder die „STRIDE Threat & Mitigation Techniques List“²⁰ der OWASP geben.

Die beschriebenen Schritte der Bedrohungsanalyse werden iterativ durchgeführt, bis die verbliebenen Bedrohungen akzeptabel sind. Das Ergebnis der Bedrohungsmodellierung ist eine Liste identifizierter und bewerteter Bedrohungen mit entsprechend begründeten Gegenmaßnahmen.

4.3.5 Sicherheitsarchitektur

In einer Sicherheitsarchitektur werden vom Auftragnehmer alle für die Anwendung notwendigen Komponenten und Sicherheitsmaßnahmen zusammengefasst und dokumentiert. Dies umfasst alle Akteure, Assets und Komponenten, Datenflüsse und Trust Boundaries.

4.3.6 Sicherheitstestplanung

Schon vor Beginn der Implementierungsphase muss der Auftragnehmer ein Testkonzept erstellen. Darin wird festgelegt, welche Bestandteile der Software zu welchem Entwicklungszeitpunkt in welchem Detailgrad getestet werden. Insbesondere bei agilen Entwicklungsmethoden ist es wichtig, im Vorhinein zu definieren, was am Ende von jedem Sprint getestet werden muss. Zusätzlich muss im Testkonzept enthalten sein, wer für die Durchführung verantwortlich ist. Die Tests können sowohl vom Auftragnehmer als auch von externen Sicherheitsexperten durchgeführt werden.

Ein weiteres Qualitätsmerkmal für eine sichere Softwareentwicklung sind Security Pushes. Während einer solchen Phase wird keine zusätzliche Funktionalität entwickelt, sondern ausschließlich die Sicherheit der Software verbessert.

4.3.7 Leistungskriterien

- Welche Dokumente werden im Zuge der Design-Phase erstellt?
- Stellen Sie ihren Ansatz für eine Bedrohungsmodellierung dar.
- Welche Design-Prinzipien werden beim Softwareentwurf berücksichtigt?
- Welche sicherheitsrelevanten Schritte werden bereits vor der Implementierung geplant?

4.3.8 Quality Gate: Prüfbare Ergebnisse der Phase

Am Ende der Konzeptions- und Planungsphase müssen vom Auftragnehmer folgende Dokumente erstellt worden sein, die vom Auftraggeber auf ihre Vollständigkeit geprüft werden können:

- Liste der verwendeten Guidelines und Good Practices für das Softwaredesign
- Objektmodell mit Datentypen, Datenklassifizierungen und Datenflüssen
- Abuse Cases

19 [OTHM]

20 [OTHM]

- Bedrohungsmodellierung
- Access Control Matrix bzw. Rollen und Berechtigungskonzepte
 - Liste mit Zugangspunkten zur Applikation
 - Bedrohungsmodell mit Vertrauensgrenzen (Trust Boundaries)
 - Liste identifizierter und bewerteter Bedrohungen
 - Liste mit begründeten Gegenmaßnahmen
- Liste aller Sicherheitsanforderungen
- Detaillierte Testplanung
- Planung von Quality Gates
- Planung von Security Pushes
- Software Security Metriken²¹
- Sicherheitsarchitektur
 - Werden Design-Prinzipien angewendet?
 - z.B. Separation of Concerns: Ist jede Aufgabe einer und nur einer logischen Komponente zugeordnet?
- Wurden alle notwendigen Angriffsvektoren ermittelt und behandelt?
- Ist die Architektur vollständig in Bezug auf funktionale und nicht funktionale Sicherheitsanforderungen?
- Ist die Architekturbeschreibung auf dem aktuellen Stand und entspricht der vorliegenden Code-Basis?
- Wurden die wesentlichen Designentscheidungen durch Dokumentation explizit gemacht?
- Sind alle wesentlichen Entscheidungen nachvollziehbar?
- Qualität und Ausführlichkeit der Dokumentation
- Sind fachliche und technologische Aspekte getrennt adressiert?
- Ist die Anwendung über eine logische Komponentenbildung auf verschiedensten Ebenen strukturiert?
- Erfolgen der Datenfluss, der Steuerungsfluss und die Fehlerbehandlung durchgehend konsistent nach demselben Schema (zumindest innerhalb von Komponenten derselben Architektur-Ebene)?
- Wie gut erlaubt die Architektur die Erweiterbarkeit der Anwendung während der Wartung?
- Sind vorhandene Schnittstellen definiert und hinreichend dokumentiert?

21 [BSI 2013]

4.4 Phase 3: Implementierung

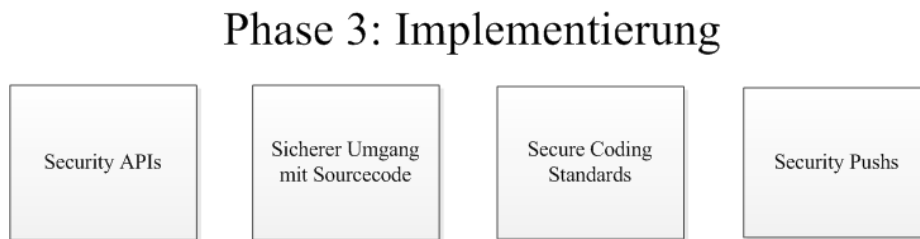


Abbildung 6: Aktivitäten der Phase 3 "Implementierung"

4.4.1 Security APIs

Bei sicherheitskritischen Funktionen wie z.B. Kryptographie sollte unbedingt auf Eigenentwicklungen verzichtet werden. Stattdessen sollten nur ausgiebig getestete APIs verwendet werden. Im Idealfall werden vom Auftragnehmer eigene Repositories bzw. Listen mit erlaubten Frameworks geführt. Diese Listen müssen regelmäßig geprüft und aktualisiert werden und es muss einen Prozess geben, mit dem auf das Bekanntwerden einer Schwachstelle in einem erlaubten Framework reagiert wird.

4.4.2 Sicherer Umgang mit Sourcecode

Sicherheit betrifft nicht nur die Daten und Funktionen einer Software, sondern auch den Quellcode selbst. Während der Entwicklungsphase ist es deshalb wichtig, dass die Schutzziele Vertraulichkeit, Integrität und Verfügbarkeit auch für den Sourcecode bewertet werden. Die genauen Anforderungen an den sicheren Umgang mit Sourcecode können dem Dokument „Leitfaden zur Entwicklung sicherer Webanwendungen. Empfehlungen und Anforderungen an die Auftragnehmer“²² entnommen werden.

4.4.3 Secure Coding Standards

Secure Coding Standards legen Vorgaben für die Entwickler fest, um ein Mindestmaß an Sicherheit im Entwicklungsprozess zu schaffen. Die Ausgangsbasis für Secure Coding Standards sollte eher allgemein und technologie-unabhängig gehalten werden. Von dieser Basis aus können in der Folge spezifische Secure Coding Standards abgeleitet werden, die die Charakteristiken der verwendeten Programmiersprachen und Frameworks berücksichtigen.

4.4.4 Security Push

Ein Security Push ist ein Vorgehen, bei dem für einen Sprint die Entwicklung neuer Funktionalität eingestellt wird. Stattdessen dient die Phase ausschließlich der Verbesserung der Sicherheit der zu entwickelnden Software. Ein Security Push kann in jeder Projektphase zur Verbesserung der Sicherheit beitragen und sollte schon frühzeitig im Rahmen der Gesamtplanung berücksichtigt werden. Für den Auftraggeber sind die Ergebnisse eines Security Pushes interessant, da sich daraus wieder neue Sicherheitsanforderungen ergeben können, die Auswirkungen auf die Umsetzung des Projekts haben.

4.4.5 Leistungskriterien

- Erläutern Sie Ihren Umgang mit APIs und Frameworks.

²² [BSI 2013]

- Erläutern Sie Ihre Richtlinien für den Umgang mit Source Code.
- Welche Maßnahmen ergreifen Sie zur Erhöhung und Verbesserung der Sicherheit in der Implementierungsphase?

4.4.6 Quality Gate: Prüfbare Ergebnisse der Phase

Während der Implementierungsphase können folgende Punkte vom Auftraggeber geprüft werden:

- An welchen Stellen wurden Standard-APIs verwendet?
- An welchen Stellen wurden Standard-APIs nicht verwendet und warum?
- Wurde der eigene Code sauber von Fremdmodulen getrennt?
- Wurden Daten bzw. Ressourcen von dem Code getrennt?
- Existiert eine Liste/ein Repository mit zu verwendenden Frameworks?
 - Ja: Wie wird die Liste gewartet und wie wird auf Schwachstellen reagiert?
 - Nein: Welche Frameworks kommen zum Einsatz und warum?
- Kommen Eigenentwicklungen für sicherheitskritische Komponenten zum Einsatz?
 - Wenn ja, dann nur mit Begründung, warum man sich für eine Eigenentwicklung entschieden hat.
- Werden Maßnahmen zum Schutz des Source Codes getroffen?
- Werden die im Sicherheitsprojektplan vorgesehenen Quality Gates und Security Pushes eingehalten?
- Werden die festgelegten Sicherheitsanforderungen erfüllt?
- Ist der Quellcode, insbesondere sicherheitskritische Abschnitte, ausreichend dokumentiert (auf Qualität und Ausführlichkeit der Dokumentation ist zu achten)?
- Werden allgemeine Secure Coding Standards verwendet?
- Werden programmiersprachen-spezifische Secure Coding Standards verwendet?
- Sind Qualität und Umfang der Secure Coding Standards hinreichend und beinhalten diese z.B.
 - einen einheitlichen Programmierstil,
 - eine Trennung von Eingabedaten und Befehlen und
 - eine Fehlerbehandlung.

4.5 Phase 4: Testen

Phase 4: Testen

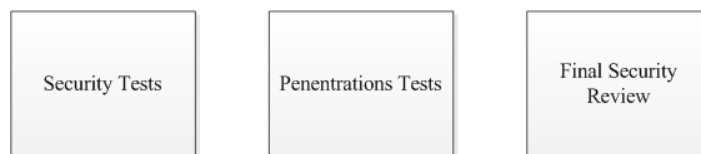


Abbildung 7: Aktivitäten der Phase 4 "Testen"

In Abhängigkeit zum ermittelten Schutzbedarf sind unterschiedliche Softwaretests durchzuführen. Diese Tests können entweder direkt durch den Auftragnehmer oder durch externe Sicherheitsexperten durchgeführt werden. Als Allgemeinkriterium gilt, dass das Testen nicht von dem Entwicklerteam erfolgen darf, sondern hierfür unabhängige Tester herangezogen werden. Unabhängige Tester sind unvoreingenommen und können die Ergebnisse objektiv bewerten. Die für den jeweiligen Schutzbedarf durchzuführenden Tests können der Checkliste in Kapitel 5.3 *Checkliste für den Entwicklungsprozess* entnommen werden. Der Auftragnehmer ist dafür verantwortlich, dass alle notwendigen Testverfahren entsprechend den Vorgaben im Dokument „Leitfaden zur Entwicklung sicherer Webanwendungen. Empfehlungen und Anforderungen an die Auftragnehmer“ umgesetzt werden. Das Ergebnis jedes durchgeführten Tests muss in einem für den Auftraggeber zugänglichen Bericht festgehalten werden. Die

folgenden Abschnitte beschreiben die Inhalte, die jeder einzelne Bericht mindestens enthalten muss. Im Unterschied zu den anderen Phasen werden die Quality Gates aufgrund der Vielzahl zusätzlich pro Testverfahren definiert. Am Ende werden allgemeine Anforderungen an die Phase angeführt.

4.5.1 Security Tests

Dieses Kapitel beschreibt, worauf bei den einzelnen Testaktivitäten zu achten ist. Im Gegensatz zu den vorangehenden Kapiteln werden pro Security Test Quality Gates definiert, was eine aussagekräftigere Bewertung der einzelnen Testschritte ermöglichen soll.

4.5.1.1 Security Test Cases

Security Test Cases werden herangezogen, um die spezifizierten funktionalen und nicht funktionalen Sicherheitsanforderungen einer Anwendung zu überprüfen. Ein Beispiel für einen Security Test Case zeigt Tabelle 6.

Test Case ID	TC12			
Beschreibung	Damit ein Benutzer auf die Anwendung zugreifen kann, muss der Benutzer authentifiziert werden.			
Priorität	Hoch			
Testschritt	Beschreibung	Erwartetes Resultat	Erfolgreich Ja/Nein	Anmerkung
1	Benutzer startet Anwendung	Anwendung wird geladen		
2	Benutzer gibt Benutzernamen und Passwort in die dafür vorgesehenen Felder ein und bestätigt mittels Drücken auf den Knopf „Anmelden“.	Benutzerdaten werden an den Server übertragen. Die Übertragung erfolgt verschlüsselt und nicht im Klartext. Die Eingabe wird einer Validierung unterzogen. Nur folgende Eingaben sind erlaubt a-z, A-Z, 0-9. Werden andere Zeichen erkannt, wird die Anmeldung unterbrochen und die Fehlerseite XY wird angezeigt. Ist die Anmeldung erfolgreich, wird die Startseite angezeigt.		
3		
4		

Tabelle 6: Beispiel für einen Security Test Case

Sämtliche Schritte müssen für eine erfolgreiche Abschließung des Security Test Cases abgearbeitet werden. Für jeden Zwischenschritt sollen Positiv- (das Verhalten der Anwendung wird mit gültigen Rahmenbedingungen und Eingaben überprüft) und Negativtests (das Verhalten der Anwendung wird mit ungültigen Rahmenbedingungen und Eingaben überprüft) erstellt werden. Für den Zwischenschritt „2“ können diese beispielsweise folgendermaßen aussehen.

Positivtest:

Folgende Benutzerdaten werden für die Anmeldung verwendet.

Benutzername: mueller

Passwort: Password1.

Ergebnis: Startseite wird angezeigt.

Negativtest:

Folgende Benutzerdaten werden für die Anmeldung verwendet.

Benutzername: “

Passwort: leer

Ergebnis: Fehlerseite wird angezeigt.

Wenn die Ergebnisse den erwarteten Resultaten entsprechen, wurde der Security Test Case erfolgreich abgeschlossen.

Quality Gate:

- Security Test Cases werden im Rahmen der Entwicklung durch die Entwickler erstellt.
- Security Test Cases werden mit einer eindeutigen ID versehen.
- Security Test Cases werden priorisiert.
- Zu jedem Security Test Case liegt eine allgemein verständliche Beschreibung vor.
- Zu jedem Testschritt liegt eine Beschreibung vor.
- Zu jedem Testschritt wird ein erwartetes Resultat definiert.
- Das erwartete Resultat ist angebracht.
- Es werden sowohl Positiv- als auch Negativtests erstellt und durchgeführt.
- Die Ergebnisse der Security Test Cases werden dokumentiert.
- Zu jeder Abweichung zu dem erwarteten Resultat werden Gegenmaßnahmen definiert und dokumentiert.
- Jeder Gegenmaßnahme wird eine Priorität zugewiesen.
- Die Gegenmaßnahmen werden nach Priorität abgearbeitet.
- Nach Umsetzung der Gegenmaßnahmen wird der betroffene Security Test Case wiederholt.

4.5.1.2 Security Unit Tests

Security Unit Tests überprüfen Softwareeinheiten (sog. Units) auf korrekte Funktionalität. Ein Security Unit Test ist ein Stück Code, das ein anderes Stück Code aufruft. Das Ergebnis des Aufrufs wird anschließend auf Richtigkeit überprüft.

Beispielsweise soll die Methode *CheckWhetherValid* auf Richtigkeit getestet werden. Diese Methode führt eine Eingabvalidierung durch. Entsprechen die der Methode übergebenen Zeichen den Vorgaben, so gibt die Anwendung den Wert TRUE zurück, ansonsten wird der Wert FALSE zurückgegeben.

Der dazugehörige Security Unit Test soll nun so gestaltet werden, dass die Methode auf Richtigkeit überprüft werden kann. Dazu muss der Security Unit Test die Methode *CheckWhetherValid* aufrufen und einerseits zugelassene, aber auch unzulässige Zeichen übergeben. Nach dem Durchführen des Security Unit Tests muss das Ergebnis interpretiert werden.

Wenn z.B. das Zeichen “;“ nicht erlaubt ist, darf die Methode bei diesem Zeichen nicht den Wert TRUE zurückliefern.

Wenn die Ergebnisse den erwarteten Resultaten entsprechen, so ist der Security Unit Test bestanden und erfolgreich abgeschlossen. Die Dokumentation des Security Unit Tests kann beispielsweise wie in Abbildung 6 dargestellt erfolgen.

Unit Test ID	UT12
Priorität	Hoch
Zu testende Methode/Klasse	CheckWhetherValid
Beschreibung	Überprüft die eingegebenen Zeichen auf Zulässigkeit
Erwartetes Resultat	Wenn die eingegebenen Zeichen den Vorgaben entsprechen, soll die Methode den Wert TRUE zurückliefern. Wenn die eingegebenen Zeichen den Vorgaben nicht entsprechen, soll die Methode den Wert FALSE zurückliefern.
Erfolgreich Ja/Nein	
Anmerkung	

Tabelle 7: Unit Test Dokumentation

Quality Gate:

- Security Unit Tests werden im Rahmen der Entwicklung durch die Entwickler definiert und dokumentiert.
- Jedem Security Unit Test wird eine eindeutige ID zugeordnet.
- Jedem Security Unit Test wird eine Priorität zugeordnet.
- Security Unit Tests werden der Priorität nach ausgeführt.
- Die Abarbeitung der Security Unit Tests erfolgt automatisiert.
- Die Ergebnisse der Security Unit Tests werden dokumentiert.
- Zu jeder Abweichung von dem erwarteten Resultat werden Gegenmaßnahmen definiert und dokumentiert.
- Jeder Gegenmaßnahme wird eine Priorität zugewiesen.
- Die Gegenmaßnahmen werden nach Priorität abgearbeitet.
- Nach Umsetzung der Gegenmaßnahmen wird der betroffene Security Unit Case wiederholt.

4.5.1.3 Design Review

Im Kapitel 4.3.5 *Sicherheitsarchitektur* wurden bereits die definierten Anforderungen an die Sicherheitsarchitektur definiert. Hier gilt es nun, die Umsetzung der Anforderungen zu überprüfen.

Quality Gate:

- Die Sicherheitsarchitektur wurde erstellt und wird gewartet.
- Security Design Principles (z.B. Minimalprinzip, Segregation of Duties, Separation of Concerns etc.) wurden eingehalten.
- Wenn nicht, wurde dokumentiert, warum Design Principles nicht eingehalten wurden?
- Alle beteiligten Akteure wurden identifiziert und dokumentiert.
- Alle externen Schnittstellen wurden identifiziert und dokumentiert.
- Die komplette Angriffsfläche wurde identifiziert und dokumentiert.
- Alle Komponenten sind mit den minimal erforderlichen Rechten ausgestattet.
- Es werden Authentifizierungsmaßnahmen nach dem Stand der Technik eingesetzt.
- Alle Eingaben und Ausgaben werden validiert.
- Alle vertraulichen Informationen wurden identifiziert.
- Vertrauliche Informationen werden bei der Verarbeitung, Speicherung und Übermittlung geschützt.
- Kryptographie wird verwendet.
- Protokolle für Kryptographie entsprechen dem Stand der Technik.
- Audit- und Loggingaktivitäten wurden in die Architektur miteinbezogen.

- Wie werden Fehler behandelt? Wie werden Fehlermeldungen gestaltet? Sind diese Informationen für einen Angreifer nützlich?
- Wie werden Konfigurationen geschützt? Wie wird sichergestellt, dass ein Angreifer die Konfiguration nicht modifizieren kann?
- Wie wurde die Erweiterbarkeit der Sicherheitsarchitektur sichergestellt?
- Design-Entscheidungen wurden stets dokumentiert.
- Struktur wurde modular erstellt (Präsentations-, Anwendungs-, Datensicht).

4.5.1.4 Code Review

Im Zuge eines Code Reviews wird der Programmcode manuell überprüft. Neben Fehlern im Code können mit einem Code Review auch Fehler in der Umsetzung von Sicherheitsanforderungen, der Einhaltung von Entwicklungsrichtlinien und unzureichende Dokumentation des Source Codes festgestellt werden. Der Vorteil des manuellen Code Review ist, dass dieses auch ohne vollständigen Code, beispielsweise für einzelne Module, durchgeführt werden kann. Wichtig bei der Auswahl des Testteams ist, dass die beteiligten Personen den Code nicht selbst entwickelt haben. Nur so kann eine objektive Betrachtung des Codes sichergestellt werden. Da eine Überprüfung des gesamten Source Codes mit großem Zeitaufwand verbunden ist, wird die manuelle Überprüfung des Codes in erster Linie für kritische Funktionen verwendet.

Um die Ergebnisse des Code Reviews sinnvoll zu verwerten und interpretieren zu können, sollten Metriken definiert werden. Metriken helfen auf der einen Seite, die Sicherheit bzw. den Fortschritt der Sicherheitsüberprüfungen einer Applikation festzustellen und auf der anderen Seite auch den Projektplan anzupassen.

Folgende Metriken könnten beispielsweise eingesetzt werden:

- Analyse Geschwindigkeit: Bsp. 250 LoC (Lines of Code) pro Stunde
Die Analysegeschwindigkeit ist stark von der Komplexität des Source Codes abhängig.
- Gefundene Fehler / Fehlerklassen pro Einheit
Als Einheit sollte hierbei möglichst ein Zeitfenster gewählt werden, da eine Interpretation der Fehler pro LoC stark von der Strukturierung des Codes abhängig ist und keine aussagekräftigen Schlüsse ziehen lässt.
- Codeabdeckung in Prozent
- Zeit für die Beseitigung eines Fehlers einer bestimmten Klasse
Bei größeren Projekten könnte man daraus ableiten, wie viel Fehler und Fehlerklassen pro Zeiteinheit gefunden werden und wie lange die Behebung dauert.
- Fehler/Fehlerklassen bei erneuter Überprüfung
Die Anzahl der alten und neuen Fehler, die bei einem Recheck gefunden werden.

Quality Gate:

- Definition des Anwendungsbereichs (kritische zu überprüfende Komponenten).
- Die Ergebnisse werden dokumentiert.
- Entsprechende Gegenmaßnahmen werden definiert und dokumentiert.
- Vor der Umsetzung der Gegenmaßnahmen erfolgt eine Priorisierung.
- Die Behebung der gefundenen Fehler wird im Zuge des nächsten Reviews überprüft.

4.5.1.5 Statische Code Scanner

Abbildung 8 zeigt den manuellen und automatisierten Schritt der statischen Codeanalyse. Der Einsatz mehrerer Tools zur automatisierten Analyse kann neben einer erhöhten Erkennungsrate auch zu einer Reduzierung der False/Positive-Meldungen führen. Trotzdem ist bei jedem Einsatz von automatisierten Tools eine manuelle Überprüfung der Ergebnisse notwendig. Ein großer Vorteil der automatisierten Codeanalyse ist, dass die eigentlichen Tests auch von Personen ohne tiefgreifenden Sicherheitskenntnissen

durchgeführt werden können und viele typische Fehler vor der aufwendigen manuellen Code Analyse gefunden werden können.

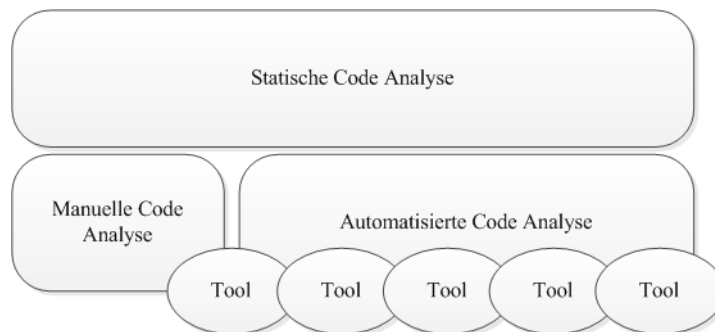


Abbildung 8: Statische Code Scanner

Kriterien für die Toolauswahl:

Bei der Auswahl der Tools ist darauf zu achten, dass die Tools mit dem eingesetzten Technologiestack kompatibel sind und möglichst viele der eingesetzten Technologien unterstützen.

Quality Gate:

- Die eingesetzten Tools werden dokumentiert.
- Die Ergebnisse werden nach Aussortierung von False/Positive-Meldungen dokumentiert.
- Die dokumentierten Fehler werden manuell verifiziert.
- Entsprechende Gegenmaßnahmen werden definiert und dokumentiert.
- Vor der Umsetzung der Gegenmaßnahmen erfolgt eine Priorisierung.
- Die Behebung der gefundenen Fehler wird im Zuge des nächsten Reviews überprüft.

4.5.1.6 Web Security Scanner

Web Security Scanner können dazu eingesetzt werden, um automatisch nach einfachen Schwachstellen (Low Hanging Fruits) zu suchen. Scanner arbeiten grundsätzlich nach dem Black-Box Ansatz und können nur Fehlerklassen identifizieren, die in ihrer jeweiligen Datenbank hinterlegt sind. Obwohl Web Security Scanner eine schlechte Erkennungsrate bei komplexeren Schwachstellen aufweisen, sind diese doch gut dazu geeignet, um eine Software vor dem eigentlichen Testprozess zu überprüfen. Web Security Scanner können sowohl vom Auftragnehmer als auch vom Auftraggeber eingesetzt werden, um die Sicherheit der Software zu überprüfen.

Bei der Auswahl der Scanner können folgende Kriterien berücksichtigt werden:

- **Unterstützte Protokolle**
z.B. http 1.1, SSL/TLS, Socks 5 Proxy
- **Unterstützte Authentisierungsmethoden**
z.B. Basic, HTML form-based, Single Sign In, SSL Certificates
- **Session Management**
z.B. Starte/Erneuere/Beende/Lösche Session, unterstützte Token-Arten
- **Crawler**
z.B. Einstellungsmöglichkeiten (IPs, Hostnames, ...), Funktionalität (Error Pages, Redirect, ...)
- **Parser**
z.B. Content Types (HRML, JavaScript, XML, ...), Encoding (UTF-8, ...)
- Berichterstattung

Quality Gate:

- Die eingesetzten Tools werden dokumentiert.

- Die Ergebnisse werden nach Aussortierung von False/Positive-Meldungen dokumentiert.
- Die dokumentierten Fehler werden manuell verifiziert.
- Entsprechende Gegenmaßnahmen werden definiert und dokumentiert.
- Vor der Umsetzung der Gegenmaßnahmen erfolgt eine Priorisierung.
- Die Behebung der gefundenen Fehler wird im Zuge des nächsten Reviews überprüft.

4.5.1.7 Fuzz Testing

Viele Fehler in Applikationen werden durch falsche oder unerwartete Benutzereingaben ausgelöst. Beim Fuzzing wird versucht, solche unerwarteten Eingabewerte zu finden, die beispielsweise einen Buffer Overflow auslösen. Theoretisch müssten bei einem vollständigen Fuzz Test alle Eingabemöglichkeiten getestet werden, was aber in der Praxis bei komplexeren Programmen nicht durchführbar ist. Aus diesem Grund ergibt es Sinn, nur Werte und Muster zu testen, die mit hoher Wahrscheinlichkeit einen Fehler auslösen. Ein Beispiel wären die Bereichsgrenzen von numerischen Variablen.

Quality Gate:

- Definition des Anwendungsbereichs (zu überprüfende Komponenten).
- Eingabewerte, die zu Fehlern führen, werden mit ihrem Ergebnis dokumentiert.
- Die dokumentierten Fehler werden manuell verifiziert.
- Entsprechende Gegenmaßnahmen werden definiert und dokumentiert.
- Vor der Umsetzung der Gegenmaßnahmen erfolgt eine Priorisierung.
- Die Behebung der gefundenen Fehler wird im Zuge des nächsten Reviews überprüft.

4.5.1.8 Penetrationstest

Penetrationstests sind Angriffe auf die Webanwendung aus der Sicht eines Angreifers. Dieses Kapitel definiert, worauf bei der Überprüfung zu achten ist. Dieses Themengebiet wird ausführlich im BSI-Dokument „Durchführungskonzept für Penetrationstest“²³ behandelt.

Quality Gate:

- Penetrationstests werden durchgeführt.
- Penetrationstests werden nach einem dokumentierten Prozess durchgeführt.
- Penetrationstests werden nach z.B. Informationsbasis, Aggressivität etc. klassifiziert.
- Der Anwendungsbereich erstreckt sich auf technische Systeme, personelle und organisatorische Infrastruktur.
- Ergebnisse der Penetrationstests werden dokumentiert.
- Gegenmaßnahmen werden definiert und dokumentiert.
- Gegenmaßnahmen werden nach einem Umsetzungsplan umgesetzt.

4.5.2 Final Security Review (FSR)

Das Final Security Review wird auf die fertiggestellte Anwendung durchgeführt.

Quality Gate:

- Ein Final Security Review wird durchgeführt.
- Nicht erfüllte Anforderungen werden evaluiert, bewertet und dokumentiert.
- Die Umsetzung der offenen Anforderungen erfolgt mittels eines Umsetzungsplans.
- Freigabeerlaubnis erfolgt nur nach positivem Abschluss (=keine offenen Anforderungen).

23 [BSI 2003]

4.5.3 Leistungskriterien

- Erläutern Sie Ihr Vorgehen beim Testen der Software und gehen Sie insbesondere auf die verwendeten Testverfahren ein.
- Machen Sie Angaben zu Testteam und Testumgebung.
- Wie werden Testergebnisse dokumentiert und kommuniziert?

4.5.4 Quality Gate: Allgemeine prüfbare Ergebnisse der Phase

Erste Softwaretests finden bereits während der Entwicklungsphase statt. Speziell beim Einsatz agiler Entwicklungsmethoden sollte es dem Auftraggeber zu jedem Ende eines Sprints möglich sein, die Ergebnisse der Testverfahren zu prüfen. Dabei muss sichergestellt werden, dass gefundene Schwachstellen und deren Behebung dokumentiert und nachvollziehbar sind. Der Auftraggeber muss im Zuge der Testphase die folgenden Fragestellungen beantworten:

- Sind Test-, Entwicklungs- und Produktivumgebung voneinander getrennt?
- Das Testteam ist nicht das Entwicklerteam.
- Werden die geplanten Tests durchgeführt?
- Sind Ergebnisberichte der durchgeführten Tests vorhanden?
- Wurden behobene Fehler dokumentiert?
- Findet vor der Auslieferung ein finaler Security Check statt?
- Die Qualität und die Ausführlichkeit der Dokumentation sind ausreichend.

4.6 Phase 5: Auslieferung & Betrieb



Abbildung 9: Aktivitäten der Phase 5 "Auslieferung und Betrieb"

4.6.1 Sicherheitsdokumentation

Neben der klassischen Projekt- und Codedokumentation muss vom Auftragnehmer auch eine eigene Dokumentation zur Sicherheit der Software erstellt werden. Die im Rahmen eines Projekts erstellte Sicherheitsdokumentation besteht aus folgenden Abschnitten und soll hinsichtlich Qualität und Ausführlichkeit ausreichend sein.

Sichere Installation:

Durch den Installationsprozess und die darauf folgende Konfiguration kann die Sicherheit einer Software wesentlich beeinflusst werden. Aus diesem Grund ist es wichtig, dass der Auftragnehmer alle sicherheitsrelevanten Einstellungen mit ihren Defaultwerten dokumentiert.

Sicherer Betrieb:

Um einen sicheren Betrieb und die Umsetzung verschiedener Schutzziele gewährleisten zu können, ist es notwendig, dass beispielsweise dokumentiert ist, wie das Berechtigungsmanagement funktioniert oder wie Informationen in der Applikation verarbeitet werden. Den Hauptbestandteil bilden Sicherheitsarchitektur und Bedrohungsmodell, die um Anforderungen an die Betriebsumgebung sowie Fehlermeldungen und Lösungen erweitert werden.

Sichere Updates:

Vor Inbetriebnahme der Software muss für den Auftraggeber ersichtlich sein, wie Updates bereitgestellt und installiert werden können und an welcher Stelle mögliche Sicherheitsprobleme mit Updates auftauchen können.

Incident Handling

Um im Falle einer Sicherheitsschwachstelle effektiv reagieren zu können, muss frühzeitig abgeklärt werden, wie das Risiko einer Schwachstelle bewertet werden kann, über welche Kanäle diese kommuniziert wird und welche Möglichkeiten der Hersteller zur Behebung bereitstellt.

4.6.2 Security Change Management

Im Rahmen des in Phase 1 erstellten Sicherheitsprojektplans muss auch der Prozess für ein sicheres Change Management festgelegt werden. Der Prozess muss zwischen Auftraggeber und Auftragnehmer abgestimmt werden und muss die Punkte Klassifizierung, Umsetzung und Überprüfung von Änderungen beinhalten.

4.6.3 Leistungskriterien

- Erläutern Sie den Change Management Prozess.
- Neben der klassischen Projekt- und Codedokumentation muss vom Auftragnehmer auch eine eigene Dokumentation zur Sicherheit der Software erstellt werden. Erklären Sie die groben Inhalte dieser Dokumentation.

4.6.4 Quality Gate: Prüfbare Ergebnisse der Phase

Folgende Quality Gates können überprüft werden:

- Sicherheitsdokumentation ist vorhanden und beinhaltet:
 - Sichere Installation
 - Sicheren Betrieb
 - Sichere Updates
 - Incident Handling
- Security Change Management Prozess ist erstellt und beinhaltet:
 - Klassifizierung von Änderungen
 - Umsetzung von Änderungen
 - Überprüfung von Änderungen

4.7 Ende des Lebenszyklus

Wird der Betrieb einer Webanwendung eingestellt, sind abschließende Anforderungen zu erfüllen, wie z. B.:

- Umsetzung von Regelungen zum Verbleib von Daten, Löschpflichten und -fristen²⁴;

²⁴ Verwaltungsvorschriften, Gesetze und Richtlinien sollten beachtet werden, z. B. Verwaltungsvorschrift des Innenministeriums zum Geheimschutz von Verschlusssachen beim Einsatz von Informationstechnik: <https://bmwi-sicherheitsforum.de/anlage/46/>

- Umsetzung von Regelungen zum Umgang mit der Hardware, z. B. Recycling oder Rückgabe an dem Hersteller;
- Widerruf / Sperrung von Zertifikaten;

5 Checklisten für die Bewertung potenzieller Auftragnehmer

Die in den folgenden Kapiteln 5.1 und 5.2 definierten Checklisten sind als Orientierungshilfe gedacht und dem Auftraggeber steht es frei, diese zu verändern oder durch eigene Checklisten zur ersetzen.

5.1 Reifegrad des Auftragnehmers

Auftragnehmer						Reifegrad					
Ansprechperson	Unternehmen	Adresse	Telefon	E-Mail	Webseite	Nicht vorhanden	Ad-Hoc	Partiell	Vollständig	Best in Class	Best in Class Aktivitäten
						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

5.2 Detaillierter Reifegrad des Auftragnehmers

Auftragnehmer	Initiale Planung und Vergabephase		Konzeptions- und Planungsphase		Implementierung		Testen		Auslieferung und Betrieb	
	Reifegrad	Best in Class Aktivitäten	Reifegrad	Best in Class Aktivitäten	Reifegrad	Best in Class Aktivitäten	Reifegrad	Best in Class Aktivitäten	Reifegrad	Best in Class Aktivitäten

5.3 Checkliste für den Entwicklungsprozess

ID	Anforderung	Schutzbedarf			Reifegrad		Anmerkung
		N	H	SH	Erfüllt	Dokum ²⁵	
0							
	Awareness Trainings & Schulungen		X	X			
1	Phase 1: Initiale Planung & Vergabephase						
	Bestimmung der funktionalen Sicherheitsanforderungen	X	X	X			
	Bestimmung der nicht funktionalen Sicherheitsanforderungen	X	X	X			
	Dokumentationsanforderungen	X	X	X			
	Compliance und gesetzliche Vorgaben	X	X	X			
	Secure Coding Guidelines	X	X	X			
	Durchführung eines Security Kickoffs	X	X	X			
	Ansprechpartner für Thema Sicherheit benannt	X	X	X			
2	Phase 2: Konzeption & Planung						
	Security Design Good Practice	X	X	X			
	Objektmodell: Datentypen, Datenklassifizierung, Datenfluss		X	X			
	Rollen und Berechtigungskonzept	X	X	X			

²⁵ Ist die Maßnahme dokumentiert?

	Access Control Matrix		X	X			
	Use Cases	X	X	X			
	Abuse Cases		X	X			
	Bedrohungsmodellierung		X	X			
	Design-Prinzipien	X	X	X			
	Schnittstellenbeschreibung	X	X	X			
	Sicherheitsanforderungen	X	X	X			
	Sicherheitsarchitektur	X	X	X			
	Quality Gates	X	X	X			
	Security Push			X			
	Planung der Testmethoden	X	X	X			
	Software Security Metriken		X	X			
3	Phase 3: Implementierung						
	Vorhandene Liste mit verwendeten Frameworks	X	X	X			
	Regelmäßige Wartung der Liste	X	X	X			
	Beschreibung der APIs		X	X			
	Begründung der eingesetzten Frameworks		X	X			
	Keine eigenentwickelten sicherheitskritischen Funktionen im Einsatz	X	X	X			

	Maßnahmen zum Schutz des Source Codes	X	X	X			
	Allgemeine Secure Coding Standards	X	X	X			
	Programmiersprachen spezifische Secure Coding Standards		X	X			
	Einhaltung von Quality Gates	X	X	X			
	Einhaltung von Security Pushes			X			
	Codedokumentation	X	X	X			
4	Phase 4: Testen						
	Security Tests Cases	X	X	X			
	Security Test Cases werden im Rahmen der Entwicklung durch die Entwickler erstellt.	X	X	X			
	Security Test Cases werden mit einer eindeutigen ID versehen.	X	X	X			
	Security Test Cases werden priorisiert.	X	X	X			
	Zu jedem Security Test Case liegt eine allgemein verständliche Beschreibung vor.	X	X	X			
	Zu jedem Testschritt liegt eine Beschreibung vor.	X	X	X			
	Das erwartete Resultat ist angebracht.	X	X	X			
	Positiv- und Negativtests	X	X	X			
	Die Ergebnisse der Security Test Cases werden dokumentiert.	X	X	X			
	Jeder Gegenmaßnahme wird eine Priorität zugewiesen.	X	X	X			

Die Gegenmaßnahmen werden nach Priorität abgearbeitet.	X	X	X			
Nach Umsetzung der Gegenmaßnahmen wird der betroffene Security Test Case wiederholt.	X	X	X			
Security Unit Tests			X			
Security Unit Tests werden im Rahmen der Entwicklung durch die Entwickler definiert und dokumentiert.			X			
Jedem Security Unit Test wird eine eindeutige ID zugeordnet.			X			
Jedem Security Unit Test wird eine Priorität zugeordnet.			X			
Security Unit Tests werden der Priorität nach ausgeführt.			X			
Die Abarbeitung der Security Unit Tests erfolgt automatisiert.			X			
Die Ergebnisse der Security Unit Tests werden dokumentiert.			X			
Zu jeder Abweichung zu dem erwarteten Resultat werden Gegenmaßnahmen definiert und dokumentiert.			X			
Jeder Gegenmaßnahme wird eine Priorität zugewiesen.			X			
Die Gegenmaßnahmen werden nach Priorität abgearbeitet.			X			
Nach Umsetzung der Gegenmaßnahmen wird der betroffene Security Unit Case wiederholt.			X			
Design Review	X	X	X			
Die Sicherheitsarchitektur wurde erstellt und wird gewartet.	X	X	X			

Security Design Principles (z.B. Minimalprinzip, Segregation of Duties, Separation of Concerns etc.) wurden eingehalten.	X	X	X			
Wenn nicht, wurde dokumentiert, warum Design Principles nicht eingehalten wurden?	X	X	X			
Alle beteiligten Akteure wurden identifiziert und dokumentiert.	X	X	X			
Alle externen Schnittstellen wurden identifiziert und dokumentiert.	X	X	X			
Die komplette Angriffsfläche wurde identifiziert und dokumentiert.	X	X	X			
Alle Komponenten sind mit den minimal erforderlichen Rechten ausgestattet.	X	X	X			
Es werden Authentifizierungsmaßnahmen nach dem Stand der Technik eingesetzt.	X	X	X			
Alle Eingaben und Ausgaben werden validiert.	X	X	X			
Alle vertraulichen Informationen wurden identifiziert.	X	X	X			
Vertrauliche Informationen werden bei der Verarbeitung, Speicherung und Übermittlung geschützt.	X	X	X			
Kryptographie wird verwendet.	X	X	X			
Protokolle für Kryptographie entsprechen dem Stand der Technik.	X	X	X			
Audit- und Loggingaktivitäten wurden in die Architektur miteinbezogen.	X	X	X			
Wie werden Fehler behandelt? Wie werden Fehlermeldungen gestaltet? Sind diese Informationen für einen Angreifer nützlich?	X	X	X			
Wie werden Konfigurationen geschützt? Wie wird sichergestellt, dass ein Angreifer die Konfiguration nicht modifizieren kann?	X	X	X			

Wie wurde die Erweiterbarkeit der Sicherheitsarchitektur sichergestellt?	X	X	X			
Design-Entscheidungen wurden stets dokumentiert.	X	X	X			
Struktur wurde modular erstellt (Präsentations-, Anwendungs-, Datensicht).	X	X	X			
Code Review		X	X			
Definition des Anwendungsbereichs (kritische zu überprüfende Komponenten).		X	X			
Die Ergebnisse werden dokumentiert.		X	X			
Entsprechende Gegenmaßnahmen werden definiert und dokumentiert.		X	X			
Vor der Umsetzung der Gegenmaßnahmen erfolgt eine Priorisierung.		X	X			
Die Behebung der gefundenen Fehler wird im Zuge des nächsten Reviews überprüft.		X	X			
Statische Code Scanner	X	X	X			
Die eingesetzten Tools werden dokumentiert.	X	X	X			
Die Ergebnisse werden nach Aussortierung von False/Positive-Meldungen dokumentiert.	X	X	X			
Die dokumentierten Fehler werden manuell verifiziert.	X	X	X			
Entsprechende Gegenmaßnahmen werden definiert und dokumentiert.	X	X	X			
Vor der Umsetzung der Gegenmaßnahmen erfolgt eine Priorisierung.	X	X	X			
Die Behebung der gefundenen Fehler wird im Zuge des nächsten Reviews überprüft.	X	X	X			
Web Security Scanner	X	X	X			

Die eingesetzten Tools werden dokumentiert.	X	X	X			
Die Ergebnisse werden nach Aussortierung von False/Positive-Meldungen dokumentiert.	X	X	X			
Die dokumentierten Fehler werden manuell verifiziert.	X	X	X			
Entsprechende Gegenmaßnahmen werden definiert und dokumentiert.	X	X	X			
Vor der Umsetzung der Gegenmaßnahmen erfolgt eine Priorisierung.	X	X	X			
Die Behebung der gefundenen Fehler wird im Zuge des nächsten Reviews überprüft.	X	X	X			
Fuzz Testing		X	X			
Definition des Anwendungsbereichs (zu überprüfende Komponenten).		X	X			
Eingabewerte, die zu Fehlern führen, werden mit ihrem Ergebnis dokumentiert.		X	X			
Die dokumentierten Fehler werden manuell verifiziert.		X	X			
Entsprechende Gegenmaßnahmen werden definiert und dokumentiert.		X	X			
Vor der Umsetzung der Gegenmaßnahmen erfolgt eine Priorisierung.		X	X			
Die Qualität und die Ausführlichkeit der Dokumentation sind ausreichend.		X	X			
Die Behebung der gefundenen Fehler wird im Zuge des nächsten Reviews überprüft.		X	X			
Penetrationstests		X	X			
Penetrationstests werden durchgeführt.		X	X			
Penetrationstests werden nach einem dokumentieren Prozess durchgeführt.		X	X			

Penetrationstests werden klassifiziert nach z.B. Informationsbasis, Aggressivität etc.		X	X			
Der Anwendungsbereich erstreckt sich auf technische Systeme, personelle und organisatorische Infrastruktur.		X	X			
Ergebnisse der Penetrationstests werden dokumentiert.		X	X			
Gegenmaßnahmen werden definiert und dokumentiert.		X	X			
Gegenmaßnahmen werden nach einem Umsetzungsplan umgesetzt.		X	X			
Final Security Review	X	X	X			
Ein Final Security Review wird durchgeführt.	X	X	X			
Nicht erfüllte Anforderungen werden evaluiert, bewertet und dokumentiert.	X	X	X			
Die Umsetzung der offenen Anforderungen erfolgt mittels eines Umsetzungsplans.	X	X	X			
Freigabeerlaubnis erfolgt nur nach positivem Abschluss (=keine offenen Anforderungen).	X	X	X			
Allgemeine Anforderung der Testphase						
Testteam ist nicht das Entwicklerteam.	X	X	X			
Trennung von Entwicklungs-, Test- und Produktionsumgebung.	X	X	X			
Geplante Tests werden durchgeführt.	X	X	X			
Ergebnisberichte der durchgeführten Tests.	X	X	X			
Dokumentation behobener Fehler.	X	X	X			

	Finaler Security Check.	X	X	X			
	Qualität und Ausführlichkeit der Dokumentation sind ausreichend.	X	X	X			
5	Phase 5: Auslieferung & Betrieb						
	Sicherheitsdokumentation	X	X	X			
	Sichere Installation	X	X	X			
	Sicherer Betrieb	X	X	X			
	Sichere Updates	X	X	X			
	Incident Handling	X	X	X			
	Security Change Management		X	X			
	Change Management Prozess ist vorhanden.		X	X			
	Change Management Prozess behandelt Klassifizierung, Umsetzung und Überprüfung von Änderungen.		X	X			

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.
Autorisierung	Prüfung, ob Subjekt berechtigt ist, eine bestimmte Aktion durchzuführen.
Code Review	Sicherheitstest; Manuelle Überprüfung des Programmcodes.
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.
Integrität	Die Vollständigkeit und Unversehrtheit von Daten muss gewährleistet werden.
Minimalprinzip	Die Funktionalität soll sich auf die Aufgabenbewältigung beschränken.

Begriff	Beschreibung
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.
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.
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.

Literaturverzeichnis

BSI 2013	Bundesamt für Sicherheit in der Informationstechnik: Leitfaden zur Entwicklung sicherer Webanwendungen. Empfehlungen und Anforderungen an die Auftragnehmer
UfAB2010	Bundesministerium des Inneren: Unterlage für Ausschreibung und Bewertung von IT-Leistungen
BSI_CMS	Bundesamt für Sicherheit in der Informationstechnik: Sicherheitsstudie - Content Management Systeme (CMS)
W3C 2004	W3C: Web Services Architecture
Delo 2011	Deloitte: Cloud Computing in Deutschland
BSI 2008	Bundesamt für Sicherheit in der Informationstechnik: BSI-Standard 100-2
CNSS2010	Committee on National Security Systems: National Information Assurance (IA) Glossary
ISO27001	Standard Policy and Strategy Committee: Information technology - Security techniques - Information security management systems - Overview and vocabulary
OSAMM	Open Web Application Security Project (OWASP): Software Assurance Maturity Model
BSIMM 2012	Gary McGraw, Sammy Miguez, Jacob West: Building Security In Maturity Model
OTHM	OWASP: OWASP Application Threat Modeling
BSI 2003	Bundesamt für Sicherheit in der Informationstechnik: Durchführungskonzept für Penetrationstests