Leitfaden zur Entwicklung sicherer Webanwendungen

Empfehlungen und Anforderungen an die Auftragnehmer
Inhaltsverzeichnis

1 Einleitung .................................................................................................................. 6
  1.1 Motivation ............................................................................................................. 6
  1.2 Ziele und Anwendungsgebiete ............................................................................ 6
  1.3 Darstellung des gewählten Ansatzes ................................................................... 7

2 Begriffe, Ziele und Konzepte .................................................................................. 8
  2.1 Webanwendungssicherheit .................................................................................. 8
  2.2 Software Assurance (SwA) ................................................................................. 8
  2.3 Funktionale und nicht-funktionale Sicherheit ....................................................... 8
  2.4 Risiken unsicherer Software ................................................................................ 9
    2.4.1 Risiken für Organisationen ............................................................................. 9
    2.4.2 Risiken für Benutzer ................................................................................... 9
  2.5 Ursachen unsicherer Software ............................................................................. 9
    2.5.1 Komplexität ................................................................................................ 9
    2.5.2 Fehlende Sichtbarkeit und Messbarkeit ......................................................... 10
    2.5.3 Fehlende Sicherheitsanforderungen .............................................................. 10
    2.5.4 Mangelndes Sicherheitsbewusstsein ............................................................. 10
    2.5.5 Schlechte Testbarkeit .................................................................................. 10
  2.6 Standards und Best Practices ............................................................................ 10
    2.6.1 BSI-Standards zur Internet-Sicherheit (ISi-Reihe), Modul „ISi Web Server“ .... 11
    2.6.2 BSI IT-Grundschutz .................................................................................. 11
    2.6.3 Sicherheit von Webanwendungen – Maßnahmenkatalog und Best Practices (BSI) 11
    2.6.4 ÖNORM A 7700 ......................................................................................... 11
    2.6.5 OWASP Application Security Verification Standard (ASVS) ......................... 12
    2.6.6 OWASP Open SAMM ............................................................................... 12
    2.6.7 BSIMM ...................................................................................................... 12

3 Einführung von Sicherheit in der Organisation ....................................................... 14
  3.1 Einführung eines Secure Development Lifecycle im gesamten Unternehmen ........ 14
  3.2 Reifegrad .......................................................................................................... 15
  3.3 Scorecards ......................................................................................................... 15
  3.4 Roadmaps ......................................................................................................... 16
  3.5 Sicherheitsspezifische Aktivitäten ..................................................................... 17
    3.5.1 Angemessene Sicherheit ............................................................................. 18
    3.5.2 Low Hanging Fruits .................................................................................. 18
    3.5.3 Erstellung von Richtlinien ......................................................................... 18
    3.5.4 Awareness Trainings & Schulungen ............................................................. 20
    3.5.5 Etablierung von Prozessen zur Qualitätssicherung ..................................... 20

4 Vorgaben an den Entwicklungsprozess ................................................................. 21
  4.1 Aufbau .............................................................................................................. 21
    4.2 Phase 1: Initiale Planung & Vergabephase ......................................................... 22
      4.2.1 IPV 1: Analyse der Anforderungen der Auftraggeber ................................. 22
      4.2.2 IPV 2: Identifikation und Bewertung existierender Sicherheitsanforderungen 22
      4.2.3 IPV 3: Erstellung einer Business Impact Analyse ...................................... 23
      4.2.4 IPV 4: Erstellung des Sicherheitsprojektplans ......................................... 24
      4.2.5 IPV 5: Definition einer Software Security Group (SSG) ............................ 24
  4.3 Phase 2: Konzeption & Planung ...................................................................... 24
<table>
<thead>
<tr>
<th>Kapitel</th>
<th>Inhalt</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.1</td>
<td>KOP 1: Beschreibung der Anwendung</td>
</tr>
<tr>
<td>4.3.2</td>
<td>KOP 2: Datenbehandlungsstrategie</td>
</tr>
<tr>
<td>4.3.3</td>
<td>KOP 3: Rollen- und Berechtigungskonzept</td>
</tr>
<tr>
<td>4.3.4</td>
<td>KOP 4: Sicherheitsanforderungen überprüfen und ggf. ergänzen</td>
</tr>
<tr>
<td>4.3.5</td>
<td>KOP 5: Erstellung von Abuse Cases</td>
</tr>
<tr>
<td>4.3.6</td>
<td>KOP 6: Bedrohungsmodellierung</td>
</tr>
<tr>
<td>4.3.7</td>
<td>KOP 7: Erstellung der Sicherheitsarchitektur</td>
</tr>
<tr>
<td>4.3.8</td>
<td>KOP 8: Sicherheitstestplanung</td>
</tr>
<tr>
<td>4.3.9</td>
<td>KOP 9: Software Security Metriken</td>
</tr>
<tr>
<td>4.4</td>
<td>Phase 3: Implementierung</td>
</tr>
<tr>
<td>4.4.1</td>
<td>IMP 1: Security APIs</td>
</tr>
<tr>
<td>4.4.2</td>
<td>IMP 2: Sicherer Umgang mit Sourcecode</td>
</tr>
<tr>
<td>4.4.3</td>
<td>IMP 3: Secure Coding Standards</td>
</tr>
<tr>
<td>4.4.4</td>
<td>IMP 4: Security Pushes</td>
</tr>
<tr>
<td>4.5</td>
<td>Phase 4: Testen</td>
</tr>
<tr>
<td>4.5.1</td>
<td>TES 1: Security Tests</td>
</tr>
<tr>
<td>4.5.2</td>
<td>TES 2: Penetration Tests</td>
</tr>
<tr>
<td>4.5.3</td>
<td>TES 3: Final Security Review (FSR)</td>
</tr>
<tr>
<td>4.6</td>
<td>Phase 5: Auslieferung &amp; Betrieb</td>
</tr>
<tr>
<td>4.6.1</td>
<td>AUB 1: Sichere Auslieferung der Software</td>
</tr>
<tr>
<td>4.6.2</td>
<td>AUB 2: Sicherheitsdokumentation</td>
</tr>
<tr>
<td>4.6.3</td>
<td>AUB 3: Plattformhär tung</td>
</tr>
<tr>
<td>4.6.4</td>
<td>AUB 4: Security Change Management</td>
</tr>
<tr>
<td>4.6.5</td>
<td>AUB 5: Security Response</td>
</tr>
<tr>
<td>4.6.6</td>
<td>AUB 6: 3rd Party Software Vulnerability Monitoring</td>
</tr>
<tr>
<td>4.6.7</td>
<td>AUB 7: Sichere Standardkonfiguration</td>
</tr>
<tr>
<td>4.6.8</td>
<td>AUB 8: Web Application Firewall</td>
</tr>
<tr>
<td>4.6.9</td>
<td>AUB 9: Wartung</td>
</tr>
<tr>
<td>5</td>
<td>Vorgaben an die Implementierung</td>
</tr>
<tr>
<td>5.1</td>
<td>Designprinzipien für sichere Systeme</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Economy of Mechanism („Minimalprinzip“)</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Fail-safe Defaults &amp; Default Deny (Sichere Standardeinstellungen)</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Complete Mediation (etwa „Vollständige Zugriffskontrolle“)</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Open Design (Offenes Design)</td>
</tr>
<tr>
<td>5.1.5</td>
<td>Segregation of Duties (Funktionstrennung)</td>
</tr>
<tr>
<td>5.1.6</td>
<td>Least Privilege („Minimale Berechtigungen“)</td>
</tr>
<tr>
<td>5.1.7</td>
<td>Least Common Mechanism (Minimale gemeinsame Ressourcen)</td>
</tr>
<tr>
<td>5.1.8</td>
<td>Psychological Acceptability (Psychologische Akzeptanz)</td>
</tr>
<tr>
<td>5.1.9</td>
<td>Compromise Recording (Protokollierung von Vorfällen)</td>
</tr>
<tr>
<td>5.2</td>
<td>Datenvalidierung</td>
</tr>
<tr>
<td>5.3</td>
<td>Authentisierung &amp; Sitzungen</td>
</tr>
<tr>
<td>5.4</td>
<td>Autorisierung</td>
</tr>
<tr>
<td>5.5</td>
<td>Kryptographie</td>
</tr>
<tr>
<td>5.6</td>
<td>Datenhaltung &amp; Datentransport</td>
</tr>
<tr>
<td>5.7</td>
<td>Konfiguration</td>
</tr>
<tr>
<td>5.8</td>
<td>Datenschutz</td>
</tr>
<tr>
<td>5.9</td>
<td>Fehlerbehandlung und Protokollierung</td>
</tr>
<tr>
<td>6</td>
<td>Checklisten</td>
</tr>
<tr>
<td>6.1</td>
<td>Checkliste für den Entwicklungsprozess</td>
</tr>
<tr>
<td>6.2</td>
<td>Checkliste für technische Vorgaben an die Implementierung</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Datenvalidierung</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Authentisierung &amp; Sitzungen</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Autorisierung</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Kryptographie</td>
</tr>
<tr>
<td>6.2.5</td>
<td>Datenhaltung &amp; Datentransport</td>
</tr>
<tr>
<td>6.2.6</td>
<td>Konfiguration</td>
</tr>
<tr>
<td>6.2.7</td>
<td>Datenschutz</td>
</tr>
<tr>
<td>6.2.8</td>
<td>Fehlerbehandlung und Protokollierung</td>
</tr>
<tr>
<td>6.2.9</td>
<td>HTTP Protokoll &amp; Web-Seiten</td>
</tr>
</tbody>
</table>

**Glossar** | 64 |

**Literaturverzeichnis** | 67 |

**Stichwort- und Abkürzungsverzeichnis** | 69 |

**Abbildungsverzeichnis**

Abbildung 1: Kosten für Fehlerbehebung | 14
Abbildung 2: Entwicklungsprozess von Microsoft | 15
Abbildung 3: Beispiel für Scorecard | 16
Abbildung 4: Beispiel für Roadmap | 17
Abbildung 5: Hierarchischer Aufbau von Richtlinien [ITGS] | 19
Abbildung 6: Entwicklungsprozess | 21
Abbildung 7: Aktivitäten der Phase 1 "Initiale Planung & Vergabephase" | 22
Abbildung 8: Aktivitäten der Phase 2 „Konzeption & Planung“ | 25
Abbildung 9: Aktivitäten der Phase 3 „Implementierung“ | 30
Abbildung 10: Aktivitäten der Phase 4 "Testen" | 32
Abbildung 11: Aktivitäten der Phase 5 "Auslieferung und Betrieb" | 35
Einleitung

1 Einleitung

1.1 Motivation


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.


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:


1 [7saf2010]
2 [Nist 2002]
Einleitung

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.


- **Eine allgemeine Verbesserung der Softwarequalität** wird auch durch gezielte Einbeziehungen und Schulung der an der Erstellung der Anwendung beteiligten Personen erreicht.  

- **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


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 Wasserfallmodell 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 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 Schutzzzielen auch sekundäre Schutzziele ab, zu denen etwa Authentizität oder Nicht-Abstreitbarkeit zählen.


2.2 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


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 Geschäftsfunctionen
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 Geschäftsfunctionen

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.

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


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 Schutzbereichs 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 Schwächen 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“

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

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)


2.6.4 ÖNORM A 7700

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)\(^9\)


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 der 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


### 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](image)

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.


3.2 Reifegrade


3.3 Scorecards

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. Der Unterschied zwischen IST und SOLL kann mit Unterstützung einer Roadmap (Kapitel 3.4) geplant und nach erfolgreicher Umsetzung verringert werden.

3.4 Roadmaps


- 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]

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

3.5.2 Low Hanging Fruits


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


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.


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


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


Abbildung 6: Entwicklungsprozess


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.

Vorgaben an den Entwicklungsprozess


4.2 Phase 1: Initiale Planung & Vergabephase


Phase 1: Initiale Planung & Vergabephase

Abb. 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 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


Die hier erstellte Liste von funktionellen 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 Auftragnehmer 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.

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.


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 Querschnittsthema 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
Vorgaben an den Entwicklungsprozess

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” verwiesen. Die Studie ist jedoch rein informativ und kann in dieser Phase unterstützend sein. Der Inhalt ist aber nicht verpflichtend.

**Phase 2: Konzeption & Planung**

![Abbildung 8: Aktivitäten der Phase 2 „Konzeption & Planung“](image)

### 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. 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 Berechtigungen (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), sowohl statisch als auch dynamisch.

### 4.3.4 KOP 4: Sicherheitsanforderungen überprüfen und ggf. ergänzen


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.


Die identifizierten Sicherheitsanforderungen werden in der Folge für die Bedrohungsmodellierung verwendet. Um einen möglichst hohen Abdeckungsgrad zu erreichen, müssen die Schritte KOP 4: Sicherheitsanforderungen überprüfen 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)
4.3.5 **KOP 5: Erstellung von Abuse Cases**


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 festgehalten. Ein Abuse Case 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 existieren 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]
Vorgaben an den Entwicklungsprozess

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


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\textsuperscript{29}
- identifizierte Assets
  - zentrale Systemkomponenten
  - wie werden vertrauliche Daten übertragen (Datenflüsse)
  - wo werden vertrauliche Daten gespeichert und wie
  - wie erfolgen die zentralen Datenflüsse

\textsuperscript{25} Weitere Informationen über die Open web Application Security Project Top 10: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2013_Project
\textsuperscript{26} Weitere Informationen zu CVSS: http://www.first.org/cvss
\textsuperscript{27} Weitere Informationen zum Web Application Security Consortium Project: http://projects.webappsec.org/w/page/13246927/FrontPage
\textsuperscript{28} Weitere Informationen zu CAPEC: http://capec.mitre.org/
\textsuperscript{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)


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 Toolvaluierung erforderlich.
• Werden bestimmte Security Pushes (siehe Kapitel 4.4.4) innerhalb der Entwicklung vorgesehen.


Wichtig ist dabei, dass sich der Grad der erforderlichen Software Assurance einzig auf den Schutzbereich 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 Los31 (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


**Phase 3: Implementierung**

![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 zu verwenden.

31 [Rafa 2010]
einzubinden. Durch die OWASP ESAPI existiert hier zudem eine für mehrere Sprachen verfügbare API, die zahlreiche Sicherheitsfunktionen abbildet.


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:

- 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-Grundschatz sicherzustellen.

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

### 4.5 Phase 4: Testen


#### 4.5.1 TES 1: Security Tests


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]

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 ausgewiesen werden):

- **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 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**
• **Code Reviews**  

• **Statische Code Scanner**  

• **Web Security Scanner**  

• **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:
Vorgaben an den Entwicklungsprozess

- Informationsbasis: Hier wird festgelegt, von welchem Wissensstand der Tester ausgeht.
- 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 „Durchführungskonzept für Penetrationstests“ entnommen werden.

4.5.3 TES 3: Final Security Review (FSR)


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]
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
Vorgaben an den Entwicklungsprozess

– 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.


• **Wesentliche Änderung**: Wesentliche Änderungen lassen sind in folgende Unterkategorien unterteilen:
Vorgaben an den Entwicklungsprozess


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


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


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.


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“)


5.1.4 Open Design (Offenes Design)


5.1.5 Segregation of Duties (Funktionstrennung)


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)


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


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 ausgeliefert 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 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


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 Bundesdatenschutzgesetzes 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


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. 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.

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


6.1 Checkliste für den Entwicklungsprozess

<table>
<thead>
<tr>
<th>ID</th>
<th>Anforderung</th>
<th>Schutzbedarf</th>
<th>Reifegrad</th>
<th>Erfüllt</th>
<th>Dokum.</th>
<th>Anmerkung</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>H</td>
<td>SH</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Awareness Trainings &amp; Schulungen</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Abhalten einer Awareness-Schulung für jeden Mitarbeiter</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>□</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Abhalten regelmäßiger technischen Awareness-Training für Entwickler</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>□</td>
<td></td>
</tr>
<tr>
<td>IPV1</td>
<td>Analyse der Anforderungen der Auftraggeber</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPV1.1</td>
<td>Erste Analyse der Ziele und Anforderungen des Auftraggebers</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>□</td>
<td></td>
</tr>
<tr>
<td>IPV2</td>
<td>Identifikation und Bewertung existierender Sicherheitsanforderungen</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPV2.1</td>
<td>Identifikation von funktionalen Sicherheitsanforderungen</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>□</td>
<td></td>
</tr>
<tr>
<td>IPV2.2</td>
<td>Identifikation von nicht-funktionalen Sicherheitsanforderungen</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>□</td>
<td></td>
</tr>
<tr>
<td>IPV3</td>
<td>Erstellung einer Business Impact Analyse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

43 Ist die Maßnahme dokumentiert?
<p>| 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 | • | • | □ | □ |</p>
<table>
<thead>
<tr>
<th>KOP4</th>
<th>Sicherheitsanforderungen überprüfen und ggf. ergänzen</th>
<th>●</th>
<th>●</th>
<th>●</th>
</tr>
</thead>
<tbody>
<tr>
<td>KOP4.1</td>
<td>Sicherheitsanforderungen überprüfen und ggf. ergänzen</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>KOP5</td>
<td>Erstellung von Abuse- und Security Use Cases</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KOP5.1</td>
<td>Definition von Sicherheitsanforderungen anhand Use Cases</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>KOP5.2</td>
<td>Definition von Sicherheitsanforderungen anhand Abuse Cases</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>KOP6</td>
<td>Bedrohungsmodellierung</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KOP6.1</td>
<td>Identifikation von Bedrohungen</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>KOP6.2</td>
<td>Analyse der identifizierten Bedrohungen</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>KOP6.3</td>
<td>Dokumentation der identifizierten Bedrohungen</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>KOP7</td>
<td>Erstellung der Sicherheitsarchitektur</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KOP7.1</td>
<td>Definition von architekturellen Sicherheitsmechanismen</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>KOP7.2</td>
<td>Definition von Sicherheitskomponenten</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>KOP8</td>
<td>Sicherheitstestplanung</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KOP8.1</td>
<td>Planung der Testaktivitäten in Zeit und Umfang</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>KOP8.2</td>
<td>Definition der Testumgebung</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>KOP8.3</td>
<td>Definition der Zuständigkeiten und Planung der Ressourcen</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>KOP8.4</td>
<td>Beschaffung bzw. Evaluierung von ggf. benötigten Softwarekomponenten</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>KOP9</td>
<td>Software Security Metriken</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KOP9.1</td>
<td>Definition der KPI „Defect Remediation Window“ (DRW)</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>KOP9.2</td>
<td>Definition der KPI „Rate of Defect Recurrence“ (RDR)</td>
<td>●</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>KOP9.3</td>
<td>Definition der KPI „Specific Coverage Metric“ (SCM)</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>KOP9.4</td>
<td>Definition der KPI „Security to Quality Defects“ (SQD)</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>IMP1</td>
<td>Security APIs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMP1.1</td>
<td>Einsatz von vertrauenswürdigen APIs für Sicherheitsfunktionen</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>IMP1.2</td>
<td>Monitoring der APIs auf Bekanntwerden von Sicherheitslücken</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>IMP2</td>
<td>Sicherer Umgang mit Sourcecode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMP2.1</td>
<td>Verwendung eines Revisionssystems zur Sourcecode Verwaltung</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>IMP2.2</td>
<td>Zugriffsberechtigungen für Lese- und Schreibzugriffe sind erforderlich</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>IMP2.3</td>
<td>Einsatz eines Change Management Prozesses zur definierten Programmmänderung</td>
<td>●</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>IMP2.4</td>
<td>Sicherheitsüberprüfung von Systemen, die den Programmcode speichern oder verarbeiten</td>
<td>●</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>IMP2.5</td>
<td>Trennung von Entwicklungs-, Test- und Produktivumgebungen</td>
<td>●</td>
<td>●</td>
<td>□</td>
</tr>
<tr>
<td>IMP2.6</td>
<td>Einsatz von Code-Signaturen</td>
<td>●</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>IMP3</td>
<td>Secure Coding Standards</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMP3.1</td>
<td>Verwendung von technologieunabhängigen Secure Coding Standards</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMP3.2</td>
<td>Ableitung von technologiespezifischen Guidelines</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMP3.3</td>
<td>Durchführung einer periodischen Entwicklerschulung für Secure Coding Standards</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMP4</td>
<td>Security Pushs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMP4.1</td>
<td>Durchführung von Security Pushs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TES1</th>
<th>Security Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>TES1.1</td>
<td>Definierte Sicherheitsanforderungen mittels Security Test Cases verifizieren</td>
</tr>
<tr>
<td>TES1.2</td>
<td>Durchführung von Security Unit Tests</td>
</tr>
<tr>
<td>TES1.3</td>
<td>Durchführung von Code Reviews</td>
</tr>
<tr>
<td>TES1.4</td>
<td>Verwendung eines statischen Code Scanners inkl. Erstellen eines Prüfberichts mit Bewertung der offenen Einträge und Übergabe der Konfigurationsdateien der Prüftools (optional)</td>
</tr>
<tr>
<td>TES1.5</td>
<td>Verwendung eines Web Security Scanners (optional)</td>
</tr>
<tr>
<td>TES1.6</td>
<td>Durchführung von Fuzz-Testing (optional)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TES2</th>
<th>Penetration Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>TES2.1</td>
<td>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</td>
</tr>
<tr>
<td>TES2.2</td>
<td>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</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TES2.3</td>
<td>Durchführung von Social Engineering Angriffen</td>
</tr>
<tr>
<td>TES2.4</td>
<td>Überprüfung der physischen Sicherheit</td>
</tr>
<tr>
<td>TES3</td>
<td>Final Security Review</td>
</tr>
<tr>
<td>TES3.1</td>
<td>Durchführung eines Final Security Reviews (FSR)</td>
</tr>
<tr>
<td>AUB1</td>
<td>Sichere Auslieferung der Software</td>
</tr>
<tr>
<td>AUB1.1</td>
<td>Definition einer Vorgehensweise zur sicheren Auslieferung der Software</td>
</tr>
<tr>
<td>AUB1.2</td>
<td>Verwendung von Prüfsummen zur Wahrung der Integrität</td>
</tr>
<tr>
<td>AUB1.3</td>
<td>Verwendung von digitalen Signaturen der Softwarekomponenten zur Wahrung der Authentizität</td>
</tr>
<tr>
<td>AUB1.5</td>
<td>Entfernen von Testdaten vor der Auslieferung</td>
</tr>
<tr>
<td>AUB1.6</td>
<td>Übergabe der Webanwendung mit allen Softwarekomponenten inkl. sämtlicher Abhängigkeiten und aller Informationen zum Übersetzen</td>
</tr>
<tr>
<td>AUB2</td>
<td>Sicherheitsdokumentation</td>
</tr>
<tr>
<td>AUB2.1</td>
<td>Erstellen einer Sicherheitsdokumentation für den Anwender (bei Clientanwendung)</td>
</tr>
<tr>
<td>AUB2.2</td>
<td>Erstellen einer Sicherheitsdokumentation für den Betrieb</td>
</tr>
<tr>
<td>AUB3</td>
<td>Plattformhärtung</td>
</tr>
<tr>
<td>AUB3.1</td>
<td>Serverkonfiguration nach dem Minimalprinzip</td>
</tr>
<tr>
<td>AUB3.2</td>
<td>Berechtigungskonfiguration nach dem „Least Privilege“-Prinzip</td>
</tr>
<tr>
<td>AUB3.3</td>
<td>Härtung von Standardaccounts</td>
</tr>
<tr>
<td>AUB4</td>
<td>Security Change Management</td>
</tr>
<tr>
<td>AUB4.1</td>
<td>Klassifizierung von Systemänderungen</td>
</tr>
<tr>
<td>AUB4.2</td>
<td>Definition und Anwendung eines Änderungsprozesses basierend auf der Änderungsklassifizierung</td>
</tr>
<tr>
<td>AUB4.3</td>
<td>Validierung der Änderung</td>
</tr>
<tr>
<td>AUB5</td>
<td>Security Response</td>
</tr>
<tr>
<td>AUB5.1</td>
<td>Reaktion auf Incidents mittels eines Security Incident Prozess</td>
</tr>
<tr>
<td>AUB5.2</td>
<td>Gewährleistung eines Zeitraumes für Security Response von kritischen Schwachstellen</td>
</tr>
<tr>
<td>AUB5.3</td>
<td>Gewährleistung eines Zeitraumes für Security Response von jeglichen Schwachstellen</td>
</tr>
<tr>
<td>AUB5.4</td>
<td>Gewährleistung eines Zeitraumes für rasche Workarounds von jeglichen Schwachstellen</td>
</tr>
<tr>
<td>AUB6</td>
<td>3rd Party Software Vulnerability Monitoring</td>
</tr>
<tr>
<td>AUB6.1</td>
<td>Erstellung und Wartung einer Liste mit Abhängigkeiten von Drittsoftware</td>
</tr>
<tr>
<td>AUB6.2</td>
<td>Monitoring der Liste von Drittsoftware auf neue Sicherheitslücken</td>
</tr>
<tr>
<td>AUB7</td>
<td>Sichere Standardkonfiguration</td>
</tr>
<tr>
<td>AUB7.1</td>
<td>Auslieferung der Anwendung mit einer sicheren Standardkonfiguration</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>AUB8</td>
<td>Web Application Firewall</td>
</tr>
<tr>
<td>AUB8.1</td>
<td>Zusätzlicher Schutz mittels Web Application Firewalls (WAF)</td>
</tr>
<tr>
<td>AUB8.2</td>
<td>Konfiguration der WAF durch fachmännisches Personal</td>
</tr>
<tr>
<td>AUB9</td>
<td>Wartung</td>
</tr>
<tr>
<td>AUB9.1</td>
<td>Sichere Zurverfügungstellung von Fernwartungszugängen</td>
</tr>
<tr>
<td>AUB9.2</td>
<td>Sichere Durchführung der Wartung Vorort</td>
</tr>
</tbody>
</table>
### 6.2 Checkliste für technische Vorgaben an die Implementierung

#### 6.2.1 Datenvalidierung

<table>
<thead>
<tr>
<th>ID</th>
<th>Anforderung</th>
<th>Schutzbedarf</th>
<th>Anmerkung</th>
</tr>
</thead>
<tbody>
<tr>
<td>DV1</td>
<td>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.</td>
<td>☐ ☐ ☐</td>
<td></td>
</tr>
<tr>
<td>DV2</td>
<td>Es ist sicherzustellen, dass alle Fehler während der Eingabevalidierung zu einer Zurückweisung bzw. Bereinigung der entsprechenden Eingaben führen.</td>
<td>☐ ☐ ☐</td>
<td></td>
</tr>
<tr>
<td>DV3</td>
<td>Es ist sicherzustellen, dass ein bestimmter Zeichensatz (z.B. UTF-8) für alle Eingabequellen definiert ist.</td>
<td>☐ ☐ ☐</td>
<td></td>
</tr>
<tr>
<td>DV4</td>
<td>Es ist sicherzustellen, dass sämtliche Eingabevalidierung mindestens serverseitig durchgeführt wird.</td>
<td>☐ ☐ ☐</td>
<td></td>
</tr>
<tr>
<td>DV5</td>
<td>Dateiuploads</td>
<td>☐ ☐ ☐</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Der Dateityp der am Server zu speichernden Datei muss dem erwarteten Dateityp entsprechen.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Die Dateiendung der generierten Datei muss überprüft und eingeschränkt werden.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Die maximale Größe einer zu generierenden Datei muss definiert werden.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bilder und Dokumente sollten konvertiert werden, um Schadcode zu entfernen und Fehlformatierungen zu korrigieren.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vom Benutzer hochgeladene Dateien dürfen auf dem Server nicht ausgeführt werden können.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DV6</td>
<td>Einbindung externer sowie interner Ressourcen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Es darf keine Ressource eingebunden werden, deren Identifikator durch ungefilterte Benutzereingaben definiert wird.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alle eingebundenen Ressourcen müssen definiert und in einer eigenen Liste bzw. einem eigenen Verzeichnis abgelegt werden.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Der Zugriff ist auf Ressourcen zu beschränken, die für die Funktion der Webapplikation notwendig sind.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Externe Ressourcen dürfen nur mit den minimal notwendigen Rechten verwendet werden.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Die einzubindenden Ressourcen müssen so abgelegt werden, dass sie nicht unbefugt gelesen oder verändert werden können.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DV7</th>
<th>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:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Es ist sicherzustellen, dass die Verwendung von nicht vertrauenswürdigen Daten bei SQL-Interpretern nur mittels parametrisierter Interfaces, Prepared Statements oder zumindest korrekter Encodierung erfolgt.</td>
</tr>
<tr>
<td></td>
<td>Es ist sicherzustellen, dass die Verwendung von vertrauenswürdigen Daten bei XML-Interpreter nur mittels parametrisierter Interfaces oder korrekter Encodierung erfolgt.</td>
</tr>
<tr>
<td></td>
<td>Es ist sicherzustellen, dass die Verwendung von vertrauenswürdigen Daten beim LDAP-Interpreter nur mittels korrekter Encodierung erfolgt.</td>
</tr>
<tr>
<td></td>
<td>Es ist sicherzustellen, dass die Verwendung von vertrauenswürdigen Daten bei Betriebssystem Kommando Interpretern nur mittels korrekter Encodierung erfolgt.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DV8</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Es ist sicherzustellen, dass nicht vertrauenswürdige Daten, welche in HTML ausgegeben werden (inklusive HTML-Elemente, Attribute, JavaScript-Datenobjekte, CSS-Blöcke und URI-Attribute), werden.</td>
</tr>
</tbody>
</table>
6.2.2 Authentisierung & Sitzungen

<table>
<thead>
<tr>
<th>ID</th>
<th>Anforderung</th>
<th>Schutzbedarf</th>
</tr>
</thead>
<tbody>
<tr>
<td>AH1</td>
<td>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 Sitzungidentifikatoren, beispielsweise ein eindeutiger und zufällig generierter Sitzungsschlüssel, welche über einen vertraulichen Kanal transferiert werden, identifiziert werden kann.</td>
<td>□</td>
</tr>
<tr>
<td>AH2</td>
<td>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).</td>
<td>□</td>
</tr>
<tr>
<td>AH3</td>
<td>Es ist sicherzustellen, dass alle Authentisierungskontrollen serverseitig umgesetzt sind.</td>
<td>□</td>
</tr>
<tr>
<td>AH4</td>
<td>Es ist sicherzustellen, dass alle Authentisierungskontrollen bei einem Fehlerfall in einen sicheren Zustand fallen.</td>
<td>□</td>
</tr>
<tr>
<td>AH6</td>
<td>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.</td>
<td>□</td>
</tr>
<tr>
<td>AH7</td>
<td>Es ist sicherzustellen, dass Passwörter von Benutzern nur als gesalzener Hash gespeichert werden.</td>
<td>□</td>
</tr>
<tr>
<td>AH8</td>
<td>Es ist sicherzustellen, dass Sitzungidentifikatoren ungültig werden, wenn sich der Benutzer abmeldet.</td>
<td>□</td>
</tr>
<tr>
<td>AH9</td>
<td>Es ist sicherzustellen, dass Sitzungen bzw. Sitzungidentifikatoren nach einer bestimmten Zeit der Inaktivität des Gegenübers ungültig werden (Zeitüberschreitung durch Inaktivität).</td>
<td>☐</td>
</tr>
<tr>
<td>AH10</td>
<td>Es ist sicherzustellen, dass Sitzungen nach einer administrativ konfigurierbaren Zeit der Aktivität ungültig werden (absolute Zeitüberschreitung).</td>
<td>☐</td>
</tr>
<tr>
<td>AH11</td>
<td>Es ist sicherzustellen, dass die Sitzungidentifikatoren nach erfolgtem/erneutem Anmelden erneuert werden.</td>
<td>☐</td>
</tr>
<tr>
<td>AH12</td>
<td>Es ist sicherzustellen, dass alle Seiten, deren Zugriff eine Authentisierung erfordert, über einen Abmeldemechanismus verfügen.</td>
<td>☐</td>
</tr>
<tr>
<td>AH13</td>
<td>Es ist sicherzustellen, dass Sitzungidentifikatoren nirgendwo anders als in Cookies von der Applikation preisgegeben werden; insbesondere nicht in URLs, Fehlermeldungen oder Protokolldateien.</td>
<td>☐</td>
</tr>
<tr>
<td>AH14</td>
<td>Es ist sicherzustellen, dass nur authentische Sitzungidentifikatoren, die von der Applikation selbst oder einer verwendeten Drittkomponente erstellt wurden, von der Anwendung akzeptiert werden.</td>
<td>☐</td>
</tr>
<tr>
<td>AH15</td>
<td>Es ist sicherzustellen, dass verwendete Sitzungidentifikatoren über eine ausreichende Länge und Zufälligkeit verfügen, um typischen Angriffen für die jeweilige Umgebung zu widerstehen.</td>
<td>☐</td>
</tr>
<tr>
<td>AH16</td>
<td>Wird ein Benutzer mit Hilfe der Sitzung authentifiziert, darf der Sitzungidentifikator nur über verschlüsselte Kanäle übertragen werden.</td>
<td>☐</td>
</tr>
<tr>
<td>AH17</td>
<td>Es ist sicherzustellen, dass im angemeldeten Bereich verwendete Sitzungidentifikatoren ein entsprechend restriktiv gesetztes Domain- und Pfad-Attribut verwenden.</td>
<td>☐</td>
</tr>
<tr>
<td>AH18</td>
<td>Es ist sicherzustellen, dass alle Authentisierungskontrollen (inklusive Bibliotheken, welche externe Authentisierungsdienste aufrufen) an einer zentralen Stelle implementiert werden.</td>
<td>☐</td>
</tr>
<tr>
<td>AH19</td>
<td>Es ist sicherzustellen, dass eine erneute Authentisierung (Re-Authentisierung) erfolgt, bevor der Aufruf einer sensiblen Operation zugelassen wird.</td>
<td>☐</td>
</tr>
<tr>
<td>AH20</td>
<td>Es ist sicherzustellen, dass Zugangsdaten nach einer administrativ konfigurierbaren Zeit ungültig werden und erneuert bzw. geändert werden müssen.</td>
<td>☐</td>
</tr>
<tr>
<td>AH21</td>
<td>Es ist sicherzustellen, dass alle Entscheidungen in Bezug auf die Authentisierung protokolliert werden.</td>
<td>☐</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>ID</th>
<th>Anforderung</th>
<th>Schutzbedarf</th>
<th>Anmerkung</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO1</td>
<td>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.</td>
<td>☐  ☐  ☐</td>
<td></td>
</tr>
<tr>
<td>AO2</td>
<td>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.</td>
<td>☐  ☐  ☐</td>
<td></td>
</tr>
<tr>
<td>AO3</td>
<td>Applikationen dürfen Benutzern nur die minimalen Rechte, welche für die Ausführung der erforderlichen Funktionen unbedingt benötigt werden, zuweisen.</td>
<td>☐  ☐  ☐</td>
<td></td>
</tr>
<tr>
<td>AO4</td>
<td>Applikationen dürfen nur die minimalen Rechte besitzen, welche für die Ausführung der erforderlichen Funktionalität unbedingt benötigt werden.</td>
<td>☐  ☐  ☐</td>
<td></td>
</tr>
<tr>
<td>AO5</td>
<td>Es ist sicherzustellen, dass Zugriffskontrollen im Fehlerfall in einen sicheren Zustand fallen.</td>
<td>☐  ☐  ☐</td>
<td></td>
</tr>
<tr>
<td>AO6</td>
<td>Es ist sicherzustellen, dass Zugriffskontrollen serverseitig umgesetzt sind.</td>
<td>☐  ☐  ☐</td>
<td></td>
</tr>
<tr>
<td>AO7</td>
<td>Es ist sicherzustellen, dass für alle Arten von Zugriffen auf geschützte Ressourcen ein zentraler Mechanismus existiert (inklusive Bibliotheken, welche externe Autorisierungsdienste aufrufen).</td>
<td>☐  ☐  ☐</td>
<td></td>
</tr>
<tr>
<td>AO8</td>
<td>Es ist sicherzustellen, dass das Protokollieren sämtlicher Entscheidungen von Zugriffskontrollen</td>
<td>☐  ☐  ☐</td>
<td></td>
</tr>
</tbody>
</table>
möglicherweise und alle fehlerhaften Entscheidungen stets protokolliert werden.

6.2.4 Kryptographie

<table>
<thead>
<tr>
<th>ID</th>
<th>Anforderung</th>
<th>Schutzbedarf</th>
<th>Anmerkung</th>
</tr>
</thead>
<tbody>
<tr>
<td>KY1</td>
<td>Es ist sicherzustellen, dass kryptographische Funktionen, die von der Anwendung für den Schutz sensible Informationen eingesetzt werden, serverseitig implementiert sind.</td>
<td>□ □ □</td>
<td></td>
</tr>
<tr>
<td>KY2</td>
<td>Es ist sicherzustellen, dass alle Arten von Zugangsdaten, welche im Klartext lokal gespeichert sind und für den Zugriff auf sicherheitsrelevante Konfigurationsdaten verwendet werden (z.B. Geheime Schlüssel, Passwörter), gegen nicht-autorisierten Zugriff geschützt sind.</td>
<td>□ □ □</td>
<td></td>
</tr>
<tr>
<td>KY3</td>
<td>Es ist sicherzustellen, dass Passwort-Hashes gesalzen sind.</td>
<td>□ □ □</td>
<td></td>
</tr>
<tr>
<td>KY4</td>
<td>Es ist sicherzustellen, dass alle kryptographischen Module im Fehlerfall in einen sicheren Zustand fallen.</td>
<td>□ □ □</td>
<td></td>
</tr>
<tr>
<td>KY5</td>
<td>Es ist sicherzustellen, dass Fehler von kryptographischen Modulen protokolliert werden.</td>
<td>□ □ □</td>
<td></td>
</tr>
<tr>
<td>KY6</td>
<td>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.</td>
<td>□ □ □</td>
<td></td>
</tr>
<tr>
<td>KY7</td>
<td>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.</td>
<td>□ □ □</td>
<td></td>
</tr>
<tr>
<td>KY8</td>
<td>Es ist sicherzustellen, dass ein Pfad zwischen einer vertrauenswürdigen CA und jedem TLS-Server-Zertifikat hergestellt werden kann und jedes Server-Zertifikat valide ist.</td>
<td>□ □ □</td>
<td></td>
</tr>
</tbody>
</table>

44 [BSI TR 102]
### 6.2.5 Datenhaltung & Datentransport

<table>
<thead>
<tr>
<th>ID</th>
<th>Anforderung</th>
<th>Schutzbedarf</th>
<th>Anmerkung</th>
</tr>
</thead>
<tbody>
<tr>
<td>DH1</td>
<td>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.</td>
<td>N</td>
<td>H</td>
</tr>
<tr>
<td>DH2</td>
<td>Es ist sicherzustellen, dass alle Verbindungen zu externen Systemen authentisiert sind, wenn diese sensible Informationen oder Funktionen betreffen.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH3</td>
<td>Es ist sicherzustellen, dass sensible Daten ausschließlich im HTTP-Body an den Server gesendet werden können.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH4</td>
<td>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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH5</td>
<td>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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DH6</td>
<td>Es ist sicherzustellen, dass nicht benötigte Dateien im Wurzelverzeichnis des Servers entfernt werden.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 6.2.6 Konfiguration

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

### 6.2.7 Datenschutz

<table>
<thead>
<tr>
<th>ID</th>
<th>Anforderung</th>
<th>Schutzbedarf</th>
<th>Anmerkung</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS1</td>
<td>Es ist sicherzustellen, dass die Anwendung keine nicht unbedingt notwendigen sensiblen Daten der Anwendung protokolliert. Dies beinhaltet beispielsweise Sitzungswidersprüche, personenbezogene oder andere sensible Informationen, z. B. Bewertungsnachweise.</td>
<td>□</td>
<td></td>
</tr>
<tr>
<td>DS2</td>
<td>Es ist sicherzustellen, dass ein Verfahren für die Löschung von sensiblen Daten existiert, wenn deren Aufbewahrungsfrist abgelaufen ist.</td>
<td>□</td>
<td></td>
</tr>
</tbody>
</table>
**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

<table>
<thead>
<tr>
<th>ID</th>
<th>Anforderung</th>
<th>Schutzbedarf</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE1</td>
<td>Es ist sicherzustellen, dass die Anwendung keine Fehlermeldungen (z.B. Stack Traces) ausgibt, welche sensible Daten enthalten (e.g. Sitzungidentifikatoren und personenbezogene Informationen).</td>
<td></td>
</tr>
<tr>
<td>FE2</td>
<td>Es ist sicherzustellen, dass alle serverseitigen Fehler auch serverseitig behandelt werden.</td>
<td></td>
</tr>
<tr>
<td>FE3</td>
<td>Es ist sicherzustellen, dass alle Protokollierungsmechanismen serverseitig implementiert sind.</td>
<td></td>
</tr>
<tr>
<td>FE4</td>
<td>Es ist sicherzustellen, dass Sicherheitsmechanismen im Fehlerfall den Zugriff verweigern.</td>
<td></td>
</tr>
<tr>
<td>FE5</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>FE6</td>
<td>Es ist sicherzustellen, dass sicherheitsrelevante Protokolldateien vor unbefugtem Zugriff und Modifikation geschützt sind.</td>
<td></td>
</tr>
<tr>
<td>FE7</td>
<td>Es ist sicherzustellen, dass die zur Protokollierung eingesetzten Sicherheitsmechanismen die Möglichkeit bieten, sowohl fehlerhafte als auch erfolgreiche sicherheitsrelevante Ereignisse protokolliert werden können.</td>
<td></td>
</tr>
</tbody>
</table>
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.
6. Ob das Ereignis erfolgreich oder fehlerhaft ist.

Es wird empfohlen, dass die Anwendung für das Protokollieren nur eine Implementierung bzw. ein Verfahren verwendet.

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

<table>
<thead>
<tr>
<th>ID</th>
<th>Anforderung</th>
<th>Schutzbedarf</th>
</tr>
</thead>
<tbody>
<tr>
<td>HT1</td>
<td>Es ist sicherzustellen, dass Weiterleitungen keine nicht validierten Daten enthalten.</td>
<td>□ □ □</td>
</tr>
<tr>
<td>HT2</td>
<td>Es ist sicherzustellen, dass die Anwendung nur eine definierte Menge von HTTP-Methoden erlaubt, wie etwa GET und POST.</td>
<td>□ □ □</td>
</tr>
<tr>
<td>HT3</td>
<td>Es ist sicherzustellen, dass das HTTP Only-Flag für alle Cookies verwendet wird, auf die nicht mittels JavaScript zugegriffen werden muss.</td>
<td>□ □ □</td>
</tr>
<tr>
<td>HT4</td>
<td>Es ist sicherzustellen, dass das Secure-Flag von allen Cookies verwendet wird, die vertrauliche Informationen enthalten.</td>
<td>□ □ □</td>
</tr>
<tr>
<td>HT5</td>
<td>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.</td>
<td>☐</td>
</tr>
<tr>
<td>HT6</td>
<td>Es ist sicherzustellen, dass für alle Formulare, welche sensible Daten enthalten, Client Caching deaktiviert ist (inklusive AutoComplete-Funktionen).</td>
<td>☐</td>
</tr>
<tr>
<td>HT7</td>
<td>Es ist sicherzustellen, dass jede HTTP-Antwort einen Content Type enthält, der einen sicheren Zeichensatz festlegt (z. B. UTF-8).</td>
<td>☐</td>
</tr>
<tr>
<td>HT8</td>
<td>Es ist sicherzustellen, dass HTTP-Header, sowohl in HTTP Anfragen als auch HTTP Antworten, nur druckbare ASCII-Zeichen enthalten.</td>
<td>☐</td>
</tr>
<tr>
<td>HT9</td>
<td>Es ist sicherzustellen, dass für alle Passwortfelder (sowie Formulare, welche diese enthalten) die Autocomplete-Funktion deaktiviert haben.</td>
<td>☐</td>
</tr>
<tr>
<td>HT10</td>
<td>Es ist sicherzustellen, dass potenziell kritische Debugging-Mechanismen vor der Inbetriebnahme abgeschaltet sind.</td>
<td>☐</td>
</tr>
<tr>
<td>HT11</td>
<td>Es ist sicherzustellen, dass von Benutzern erstellte Links auf Webseiten nicht verschleiert werden können. Links müssen stets komplett (inklusive sämtlicher Parameter) sichtbar sein.</td>
<td>☐</td>
</tr>
<tr>
<td>HT12</td>
<td>Es ist sicherzustellen, dass eine Online-Hilfefunktion zu sicherheitsspezifischen Funktionen und Einstellungen implementiert wurde. Eine Kontaktmöglichkeit zur Meldung von Sicherheitsvorfällen muss genannt werden.</td>
<td>☐</td>
</tr>
<tr>
<td><strong>Begriff</strong></td>
<td><strong>Beschreibung</strong></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Abuse Case</td>
<td>Abstrakte Beschreibung der Interaktion zwischen einem System und seiner Umgebung. Die Interaktion resultiert in einem Schaden für einen der Beteiligten.</td>
<td></td>
</tr>
<tr>
<td>Access Control Matrix</td>
<td>Eine Access Control Matrix, über welche anschaulich die Berechtigungen der existierenden Rollen auf bestimmte Assets dargestellt werden.</td>
<td></td>
</tr>
<tr>
<td>Asset</td>
<td>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 Schutzenbedarf hinsichtlich der Schutzziele ableiten lässt.</td>
<td></td>
</tr>
<tr>
<td>Authentisierung</td>
<td>Sicherstellung, dass der Kommunikationspartner derjenige ist, der er vorgibt zu sein.</td>
<td></td>
</tr>
<tr>
<td>Authentizität</td>
<td>Bezeichnet die Eigenschaft der Echtheit der Daten.</td>
<td></td>
</tr>
<tr>
<td>Autorisierung</td>
<td>Prüfung, ob Subjekt berechtigt ist, eine bestimmte Aktion durchzuführen.</td>
<td></td>
</tr>
<tr>
<td>Code Review</td>
<td>Sicherheitstest; Manuelle Überprüfung des Programmcodes.</td>
<td></td>
</tr>
<tr>
<td>Cross Site Scripting</td>
<td>Angriff, bei dem Schadcode in eine dynamische Webseite eingeschleust wird.</td>
<td></td>
</tr>
<tr>
<td>Final Security Review</td>
<td>Letzter Sicherheitstest; wird auf die fertiggestellte Anwendung durchgeführt.</td>
<td></td>
</tr>
<tr>
<td>Funktionale Sicherheit</td>
<td>Funktionalen Sicherheitsaspekte wie Authentisierung, Autorisierung, Kryptographie etc.</td>
<td></td>
</tr>
<tr>
<td>Fuzz Testing</td>
<td>Ungültige, unerwartete oder zufällige Werte werden generiert und als Eingabe an die Anwendung weitergegeben.</td>
<td></td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language; textbasierte Auszeichnungssprache zur Strukturierung von Inhalten in Dokumenten.</td>
<td></td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol; Protokoll zur Übertragung von Daten über ein Netzwerk.</td>
<td></td>
</tr>
<tr>
<td>Incident</td>
<td>Ein nicht geplanter Ausfall bzw. ein Qualitätsverlust eines Services.</td>
<td></td>
</tr>
<tr>
<td>Begriff</td>
<td>Beschreibung</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Integrität</td>
<td>Die Vollständigkeit und Unversehrtheit von Daten muss gewährleistet werden.</td>
<td></td>
</tr>
<tr>
<td>Least Privilege</td>
<td>Beschränkung der Berechtigungen eines Dateisystems auf das absolut Notwendige.</td>
<td></td>
</tr>
<tr>
<td>Low Hanging Fruits</td>
<td>Sicherheitsmaßnahmen, die mit wenig Ressourcenaufwand umgesetzt werden können.</td>
<td></td>
</tr>
<tr>
<td>Minimalprinzip</td>
<td>Die Funktionalität soll sich auf die Aufgabenbewältigung beschränken.</td>
<td></td>
</tr>
<tr>
<td>Nicht-Abstreitbarkeit</td>
<td>Eine durchgeführte Handlung ist eindeutig zurechenbar.</td>
<td></td>
</tr>
<tr>
<td>Nicht-funktionale Sicherheit</td>
<td>Nicht-funktionale Sicherheitsaspekte wie Fehler in Eingabe- und Ausgabevalidierung etc.</td>
<td></td>
</tr>
<tr>
<td>Penetration Test</td>
<td>Sicherheitstest; erfolgt aus der Sicht des Angreifers.</td>
<td></td>
</tr>
<tr>
<td>Plattformhärtung</td>
<td>Entfernung aller Softwarebestandteile der Serversysteme, die für die Aufgabenerfüllung nicht erforderlich sind.</td>
<td></td>
</tr>
<tr>
<td>Security API</td>
<td>Stellen Softwareentwicklern geprüfte Sicherheitsfunktionen zur Verfügung.</td>
<td></td>
</tr>
<tr>
<td>Security Push</td>
<td>Vorgehen in bestimmten Phasen der Entwicklung. Das gesamte Team konzentriert sich auf die Verbesserung der Sicherheit der Webanwendung.</td>
<td></td>
</tr>
<tr>
<td>Security Test Case</td>
<td>Sicherheitstest; bestimmte Funktionalität wird auf ihre Korrektheit überprüft.</td>
<td></td>
</tr>
<tr>
<td>Security Unit Test</td>
<td>Sicherheitstest; Softwareeinheiten (Funktionen, Methoden, Klassen) werden auf korrekte Funktionalität überprüft.</td>
<td></td>
</tr>
<tr>
<td>Software Assurance</td>
<td>Grad an Vertrauen in die Sicherheit von Software.</td>
<td></td>
</tr>
<tr>
<td>Software Security Group</td>
<td>Wichtige organisatorische Einheit, die über Sicherheitsaktivitäten entscheidet.</td>
<td></td>
</tr>
<tr>
<td>SQL-Injection</td>
<td>Einschleusen von SQL-Statements, aufgrund von mangelnder Eingabeüberprüfung.</td>
<td></td>
</tr>
<tr>
<td>Use Case</td>
<td>Abstrakte Beschreibung der Interaktion zwischen einem System und seiner Umgebung.</td>
<td></td>
</tr>
<tr>
<td>Begriff</td>
<td>Beschreibung</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Verfügbarkeit</td>
<td>Daten müssen zu definierten Zeiten im Einklang mit der Vertraulichkeit und Integrität zur Verfügung stehen.</td>
<td></td>
</tr>
<tr>
<td>Vertraulichkeit</td>
<td>Daten dürfen nur von Personen verarbeitet werden, die dafür bestimmt sind.</td>
<td></td>
</tr>
<tr>
<td>WAF</td>
<td>Web Application Firewall; Firewall auf Anwendungsebene.</td>
<td></td>
</tr>
</tbody>
</table>
Literaturverzeichnis


Nist 2002  National Institute of Standards & Technology: The Economic Impacts of Inadequate Infrastructure for Software Testing


ITGS  Bundesarbeitsamt: IT-Grundschatz - M.2.338 Erstellung von zielgruppengerechten Sicherheitsrichtlinien, https://www.bsi.bund.de/ContentBSI/grundschatz/kataloge/m/m02/m02338.html


BSIM 2011  Gary McGraw, Brian Chess, Sammy Migues: Building Security In Maturity Model


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


PenT2003  Bundesarbeitsamt für Sicherheit in der Informationstechnik: Durchführungskonzept für Penetrationstests,
<table>
<thead>
<tr>
<th>Autor</th>
<th>Titel</th>
<th>Quelle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salt 1975</td>
<td>Jerome Saltzer, Michael Schröder: The Protection of Information in Computer Systems</td>
<td></td>
</tr>
<tr>
<td>TMG</td>
<td>Telemediengesetz, 2007</td>
<td><a href="http://dejure.org/gesetze/TMG">http://dejure.org/gesetze/TMG</a></td>
</tr>
<tr>
<td>Stichwort- und Abkürzungsverzeichnis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abuse Case .......................................................... 27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Security Verification Standard ...................................................... 12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assets ........................................................................ 8, 26, 28f.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authonisierung .............................................................. 8, 18, 23, 28f., 34, 39, 43, 44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authentizität .................................................................. 8, 31, 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Autorisierung ................................................................. 8, 18, 28, 41f., 43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bedrohungsmodell ............................................................. 27, 29, 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSIMM ........................................................................ 7, 13, 15, 24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Review .................................................................. 33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cross-Site-Scripting .......................................................... 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Datenbehandlungsstrategie .......................................................... 25, 26, 45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Datenschutz .................................................................. 20, 44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entwicklungprozess ........................................................... 7, 10, 15, 29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fuzz Testing .................................................................. 34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incident .......................................................................... 38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integrität ....................................................................... 8, 12, 23, 31, 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISI Web Server .............................................................. 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IT-Grundschutz .............................................................. 12, 13, 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key Performance Indikatoren ......................................................... 34f.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kryptographie ................................................................. 8, 23, 28, 30f., 34, 44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Least Common Mechanism .................................................... 42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Least Privilege ................................................................. 37, 42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low Hanging Fruits ............................................................ 18, 34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimalprinzip ................................................................. 37, 41, 44f.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nicht-Abstrebbarkeit ........................................................... 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ÖNORM A 7700 ................................................................. 12, 18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open SAMM ................................................................ 12, 13, 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Penetration Test .............................................................. 34f.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plattformhärtung ............................................................... 37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reifegrad .................................................................... 12, 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roadmaps .................................................................... 16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schutzbedarf ................................................................. 7f., 10ff., 18, 21, 23ff., 29, 41, 43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scorecards .................................................................... 15, 16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secure Coding Guidelines ................................................ 18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secure Coding Standards .................................................. 31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secure Development Lifecycle ............................................... 7, 15, 21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security APIs ................................................................. 30, 31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Assessment ......................................................... 32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Gates ................................................................ 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Metriken ............................................................. 29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Push ................................................................. 29, 31f.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Response ............................................................ 38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Test Cases .......................................................... 33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Unit Tests ........................................................... 33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Segregation of Duties ........................................................ 42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sicherheit von Webanwendungen - Maßnahmenkatalog und Best Practices ...................... 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sicherheitssicherheit .......................................................... 28, 29, 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sicherheitsdokumentation ...................................................... 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sitzungen ...................................................................... 43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Assurance ........................................................ 8, 10, 12, 18, 23, 29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Development Lifecycle .................................................. 15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stichwort- und Abkürzungsverzeichnis

Software Security Group ........................................................................................................................................... 24
Statische Code Scanner ............................................................................................................................................. 34
Use Case .................................................................................................................................................................. 26f.
Verfügbarkeit ......................................................................................................................................................... 8, 12, 23, 29, 31
Vertraulichkeit ...................................................................................................................................................... 8, 12, 23, 25, 31, 36
Web Security Scanner ........................................................................................................................................... 34