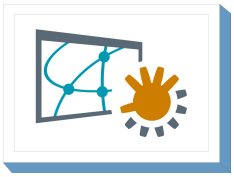
B 5.21 Webanwendungen



Beschreibung

Webanwendungen stellen Funktionen und dynamische Inhalte über das Internetprotokoll HTTP (*Hypertext Transfer Protocol*) bzw. HTTPS (HTTP über SSL bzw. TLS, d. h. geschützt durch eine verschlüsselte Verbindung) zur Verfügung. Dazu werden auf einem Server Dokumente und Benutzeroberflächen (z. B. Bedienelemente und Eingabemasken) erzeugt und an entsprechende Clientprogramme (Web-Browser) ausgeliefert.

Webanwendungen werden gewöhnlich auf der Grundlage von Frameworks entwickelt. Diese stellen ein Rahmenwerk für häufig wiederkehrende Aufgaben zur Verfügung (z. B. für Sicherheitskomponenten). Für eine Webanwendung werden häufig mehrere Frameworks für verschiedene Bereiche (z. B. Zugriff auf Datenbanken, Formatierung der Ausgaben) und Komponenten (z. B. Authentisierung, Session-Management) eingesetzt. Daher müssen bereits in der Planungsphase Sicherheitsaspekte bei der Auswahl der Frameworks sowie der Software-Architektur berücksichtigt werden.

Um eine Webanwendung zu betreiben, sind in der Regel mehrere IT-Systemkomponenten notwendig. Hierzu gehören üblicherweise ein Webserver zur Auslieferung der Daten, ein Applikationsserver für den Betrieb der Anwendung und zusätzliche Hintergrundsysteme, die als Datenquellen über unterschiedliche Schnittstellen angebunden sind (z. B. Datenbank oder Verzeichnisdienst).

Webanwendungen werden sowohl in öffentlichen IT-Netzen (z. B. dem Internet) als auch in Firmennetzen (Intranet) zur Bereitstellung von Daten und Anwendungen eingesetzt. Dabei müssen Webanwendungen Sicherheitsmechanismen umsetzen, die den Schutz der Daten gewährleisten und Missbrauch verhindern.

Typische Sicherheitskomponenten bzw. -mechanismen einer Webanwendung sind:

- Authentisierung
  Für den Zugriff auf geschützte Ressourcen der Webanwendung müssen sich die Benutzer gegenüber der Authentisierungskomponente ausweisen (z. B. durch Zugangsdaten).
- Autorisierung
  Vor dem Zugriff auf geschützte Ressourcen und Funktionen muss geprüft werden, ob die Benutzer über ausreichend Rechte verfügen.
- Ein- und Ausgabevalidierung
  Ein- und Ausgabedaten müssen geprüft und gefiltert werden, damit die Verarbeitung von schadhaften Daten (z. B. ausführbarer Schadcode) vermieden wird.
- Session-Management
  Da das Internetprotokoll HTTP keine Zuordnung zusammengehörender Anfragen zu einem Benutzer unterstützt, erfolgt diese Zuordnung durch das Session-Management der Webanwendung.
- Fehlerbehandlung
  Auftretende Fehler müssen so behandelt werden, dass die Daten der Webanwendung auch im Fehlerfall geschützt werden.
- Protokollierung
  Ereignisse müssen von der Webanwendung derart erfasst werden, dass durchgeführte Aktionen und sicherheitsrelevante Vorfälle auch zu einem späteren Zeitpunkt nachvollzogen werden können.

Abgrenzung des Bausteins

In diesem Baustein werden die für Webanwendungen spezifischen Gefährdungen und Maßnahmen betrachtet. Während Webserver die Webseiten ausliefern (siehe auch B 5.4 *Webserver*), stellen Webanwendungen Funktionen zur Verfügung und bereiten dynamische Inhalte zur Auslieferung durch den Webserver vor. Der Baustein B 5.4 *Webserver* beinhaltet auch die redaktionelle Planung des Webauftritts sowie das Notfallmanagement, das deshalb in diesem Baustein nicht nochmals behandelt wird.

Da Web-Services ähnlich wie Webanwendungen Geschäftslogik abbilden, lässt sich ein Großteil der Gefährdungen und Maßnahmen von Webanwendungen auch auf die Logik-Komponenten von Web-Services übertragen.

Bei herkömmlichen Webanwendungen werden Funktionalitäten innerhalb dieser Anwendung angeboten. Im Gegensatz dazu werden diese bei SOAP-basierten Web Services (*Simple Object Access Protocol*) als lose gekoppelte, unabhängige, austauschbare Dienste über standardisierte Schnittstellen von einem Service Provider angeboten. Anders als Webanwendungen bereiten Web-Services üblicherweise die Ausgabe von Ergebnissen nicht für einen Browser auf, sondern stellen sie in strukturierter, maschinenlesbarer Form (z. B. SOAP-Nachrichten) zur weiteren automatisierten Verarbeitung zur Verfügung. Dabei werden diese Daten durch unterschiedliche Komponenten des Web-Service (z. B. durch einen Parser oder durch Ver- und Entschlüsselungskomponenten) aufbereitet. Die sicherheitsrelevanten Aspekte bei der Realisierung einer Service-orientierten Architektur (SOA) werden im vorliegenden Baustein nicht betrachtet.

Gefährdungslage

Für den IT-Grundschutz werden pauschal die folgenden Gefährdungen als typisch im Zusammenhang mit einer Webanwendung angenommen:

Organisatorische Mängel

- G 2.1 *Fehlende oder unzureichende Regelungen*
- G 2.4 *Unzureichende Kontrolle der Sicherheitsmaßnahmen*
- G 2.7 *Unerlaubte Ausübung von Rechten*
- G 2.22 *Fehlende oder unzureichende Auswertung von Protokolldaten*
- G 2.27 *Fehlende oder unzureichende Dokumentation*
- G 2.67 *Ungeeignete Verwaltung von Zugangs- und Zugriffsrechten*
- G 2.87 *Verwendung unsicherer Protokolle in öffentlichen Netzen*
- G 2.103 *Unzureichende Schulung der Mitarbeiter*
- G 2.157 *Mangelhafte Auswahl oder Konzeption von Webanwendungen*
- G 2.158 *Mängel bei der Entwicklung und der Erweiterung von Webanwendungen und Web-Services*
- G 2.159 *Unzureichender Schutz personenbezogener Daten bei Webanwendungen und Web-Services*

Menschliche Fehlhandlungen

- G 3.16 *Fehlerhafte Administration von Zugangs- und Zugriffsrechten*
- G 3.38 *Konfigurations- und Bedienungsfehler*
- G 3.43 *Ungeeigneter Umgang mit Passwörtern*

Technisches Versagen

- G 4.22 *Software-Schwachstellen oder -Fehler*
- G 4.33 *Schlechte oder fehlende Authentikation*
- G 4.35 *Unsichere kryptographische Algorithmen*
- G 4.84 *Unzureichende Validierung von Ein- und Ausgabedaten bei Webanwendungen und Web-Services*
- G 4.85 *Fehlende oder mangelhafte Fehlerbehandlung durch Webanwendungen und Web-Services*
- G 4.86 *Unzureichende Nachvollziehbarkeit von sicherheitsrelevanten Ereignissen bei Webanwendungen*
- G 4.87 *Offenlegung vertraulicher Informationen bei Webanwendungen und Web-Services*

Vorsätzliche Handlungen

- G 5.18 *Systematisches Ausprobieren von Passwörtern*
- G 5.19 *Missbrauch von Benutzerrechten*
- G 5.20 *Missbrauch von Administratorrechten*
- G 5.28 *Verhinderung von Diensten*
- G 5.87 *Web-Spoofing*
- G 5.88 *Missbrauch aktiver Inhalte*

- G 5.131 *SQL-Injection*
- G 5.165 *Unberechtigter Zugriff auf oder Manipulation von Daten bei Webanwendungen und Web-Services*
- G 5.166 *Missbrauch einer Webanwendung durch automatisierte Nutzung*
- G 5.167 *Fehler in der Logik von Webanwendungen und Web-Services*
- G 5.168 *Umgehung clientseitig umgesetzter Sicherheitsfunktionen von Webanwendungen und Web-Services*
- G 5.169 *Unzureichendes Session-Management von Webanwendungen und Web-Services*
- G 5.170 *Cross-Site Scripting (XSS)*
- G 5.171 *Cross-Site Request Forgery (CSRF, XSRF, Session Riding)*
- G 5.172 *Umgehung der Autorisierung bei Webanwendungen und Web-Services*
- G 5.173 *Einbindung von fremden Daten und Schadcode bei Webanwendungen und Web-Services*
- G 5.174 *Injection-Angriffe*
- G 5.175 *Clickjacking*

Maßnahmenempfehlungen

Um Webanwendungen abzusichern, müssen zusätzlich zu diesem Baustein noch weitere Bausteine umgesetzt werden, gemäß den Ergebnissen der Modellierung nach IT-Grundschutz.

Der Betrieb einer Webanwendung setzt den Einsatz weiterer Komponenten voraus. Daher muss der Baustein B 3.101 *Allgemeiner Server* und abhängig von dem eingesetzten Betriebssystem beispielsweise Baustein B 3.102 *Server unter Unix* oder B 3.108 *Windows Server 2003* berücksichtigt werden. Darüber hinaus wird für den Betrieb einer Webanwendung ein Webserver (siehe B 5.4 *Webserver*) benötigt.

Funktionalität oder Daten werden bei Webanwendungen gewöhnlich in Hintergrundsystemen ausgelagert (z. B. Datenbank und Identitätsspeicher). Aus diesem Grund sind in Abhängigkeit der eingesetzten Hintergrundsysteme weitere Bausteine, wie beispielsweise B 5.7 *Datenbanken* und B 5.15 *Allgemeiner Verzeichnisdienst* (bzw. B 5.16 *Active Directory*), zu berücksichtigen.

Verarbeitet die Webanwendung personenbezogene Daten oder wertet sie Nutzerdaten aus (z. B. Abrufstatistiken, Benutzerprofile), muss zusätzlich Baustein B 1.5 *Datenschutz* beachtet werden.

Wird die Webanwendung von externen Dienstleistern betrieben oder entwickelt, ist zusätzlich der Baustein B 1.11 *Outsourcing* zu betrachten.

Für eine sichere Webanwendung sind eine Reihe von Maßnahmen umzusetzen. Die zu durchlaufenden Phasen, sowie die Maßnahmen, die in den jeweiligen Phasen zu beachtet sind, werden im Folgenden aufgeführt.

Planung und Konzeption

Bei der Planung einer Webanwendung muss üblicherweise entschieden werden, ob die Anforderungen an die Webanwendung durch Standardprodukte abgedeckt werden können oder eine Eigenentwicklung notwendig ist. Wird eine Webanwendung auf Basis von Standardsoftware umgesetzt, so sind gewöhnlich Anpassungen erforderlich, die über reine Konfigurationsänderungen hinausgehen und oft auch Entwicklungsarbeiten mit einschließen. Daher müssen auch Webanwendungen, die auf Standardsoftware basieren, häufig die Vorgaben an die Entwicklung und Erweiterung von Webanwendungen erfüllen (siehe M 2.487 *Entwicklung und Erweiterung von Anwendungen*).

Bereits in der Entwurfsphase einer Webanwendung müssen Sicherheitsaspekte beachtet werden, um die zu verarbeitenden Daten zu schützen (siehe M 5.169 *Systemarchitektur einer Webanwendung*). Hierbei müssen auch die Integration der Hintergrundsysteme (z. B. Datenbank) und deren sichere Anbindung miteinbezogen werden (siehe M 5.168 *Sichere Anbindung von Hintergrundsystemen an Webanwendungen und Web-Services*).

Werden personenbezogene Daten von Webanwendungen verarbeitet, aufgezeichnet oder ausgewertet (z. B. Nutzerverhalten), sind die rechtlichen Rahmenbedingungen bei der Planung von techni-

schen Lösungen zu berücksichtigen (siehe M 2.110 *Datenschutzaspekte bei der Protokollierung* und M 2.488 *Web-Tracking*).

Beschaffung

Soll eine Webanwendung mit verfügbarer Standardsoftware realisiert werden, muss ein passendes Produkt ausgewählt werden (siehe M 2.80 *Erstellung eines Anforderungskatalogs für Standardsoftware*).

Umsetzung

Vor der Übernahme einer Webanwendung in den Wirkbetrieb müssen die Sicherheitsfunktionen konfiguriert oder entwickelt werden. Die dafür umzusetzenden Komponenten müssen die Webanwendung vor bekannten Bedrohungen und Angriffstechniken schützen (siehe z. B. M 2.363 *Schutz gegen SQL-Injection*).

Darüber hinaus sind die kontextbezogene Validierung und Filterung der Daten (siehe M 4.392 *Authentisierung bei Webanwendungen*) und der Schutz von Benutzer-Sitzungen durch das Session-Management (siehe M 4.394 *Session-Management bei Webanwendungen und Web-Services*) wesentliche Sicherheitskomponenten einer Webanwendung.

Betrieb

Nachdem eine Webanwendung das Abnahme- und Freigabeverfahren erfolgreich durchlaufen hat und betriebsbereit konfiguriert wurde, kann der Regelbetrieb aufgenommen werden.

Insbesondere bei der Nutzung der Webanwendung über öffentliche Netze (z. B. Internet) besteht die Gefahr, dass bekannt gewordene Schwachstellen ausgenutzt werden. Daher müssen Prozesse definiert werden, um das angestrebte Sicherheitsniveau der Webanwendung dauerhaft aufrecht erhalten zu können (siehe M 2.35 *Informationsbeschaffung über Sicherheitslücken des Systems* und M 2.273 *Zeitnahes Einspielen sicherheitsrelevanter Patches und Updates*).

Es muss sichergestellt werden, dass von Webanwendungen übermittelte Daten keine sicherheitsrelevanten Informationen beinhalten, die einem Angreifer Hinweise zur Umgehung von Sicherheitsmechanismen geben (siehe M 4.400 *Restriktive Herausgabe sicherheitsrelevanter Informationen bei Webanwendungen und Web-Services*).

Für den hohen Schutzbedarf sind Penetrationstests auf die Webanwendung durchzuführen, um das Sicherheitsniveau der Webanwendung zu überprüfen und mögliche Schwachstellen schnell abzustellen (M 5.150 *Durchführung von Penetrationstests*).

Nachfolgend werden die Maßnahmen für Webanwendungen vorgestellt. Auf eine Wiederholung von Maßnahmen anderer Bausteine wird hier aus Gründen der Redundanz verzichtet.

Planung und Konzeption

- M 2.1 (A) *Festlegung von Verantwortlichkeiten und Regelungen*
- M 2.11 (A) *Regelung des Passwortgebrauchs*
- M 2.63 (A) *Einrichten der Zugriffsrechte*
- M 2.80 (A) *Erstellung eines Anforderungskatalogs für Standardsoftware*
- M 2.363 (B) *Schutz gegen SQL-Injection*
- M 2.486 (A) *Dokumentation der Architektur von Webanwendungen und Web-Services*
- M 2.487 (B) *Entwicklung und Erweiterung von Anwendungen*
- M 2.488 (W) *Web-Tracking*
- M 4.176 (B) *Auswahl einer Authentisierungsmethode für Webangebote*
- M 4.404 (A) *Sicherer Entwurf der Logik von Webanwendungen*
- M 5.168 (A) *Sichere Anbindung von Hintergrundsystemen an Webanwendungen und Web-Services*
- M 5.169 (A) *Systemarchitektur einer Webanwendung*
- M 5.177 (B) *Serverseitige Verwendung von SSL/TLS*

Beschaffung

- M 2.62 (B) *Software-Abnahme- und Freigabe-Verfahren*

Umsetzung

- M 4.392 (A) *Authentisierung bei Webanwendungen*
- M 4.393 (B) *Umfassende Ein- und Ausgabevalidierung bei Webanwendungen und Web-Services*
- M 4.394 (A) *Session-Management bei Webanwendungen und Web-Services*
- M 4.395 (B) *Fehlerbehandlung durch Webanwendungen und Web-Services*
- M 4.396 (B) *Schutz vor unerlaubter automatisierter Nutzung von Webanwendungen*
- M 4.398 (B) *Sichere Konfiguration von Webanwendungen*
- M 4.399 (A) *Kontrolliertes Einbinden von Daten und Inhalten bei Webanwendungen*
- M 4.401 (B) *Schutz vertraulicher Daten bei Webanwendungen*
- M 4.402 (A) *Zugriffskontrolle bei Webanwendungen*
- M 4.403 (C) *Verhinderung von Cross-Site Request Forgery (CSRF, XSRF, Session Riding)*
- M 4.405 (C) *Verhinderung der Blockade von Ressourcen (DoS) bei Webanwendungen und Web-Services*
- M 4.406 (Z) *Verhinderung von Clickjacking*

Betrieb

- M 2.8 (A) *Vergabe von Zugriffsrechten*
- M 2.31 (A) *Dokumentation der zugelassenen Benutzer und Rechteprofile*
- M 2.34 (A) *Dokumentation der Veränderungen an einem bestehenden System*
- M 2.35 (B) *Informationsbeschaffung über Sicherheitslücken des Systems*
- M 2.64 (A) *Kontrolle der Protokolldateien*
- M 2.110 (A) *Datenschutzaspekte bei der Protokollierung*
- M 2.273 (A) *Zeitnahes Einspielen sicherheitsrelevanter Patches und Updates*
- M 3.5 (A) *Schulung zu Sicherheitsmaßnahmen*
- M 4.78 (A) *Sorgfältige Durchführung von Konfigurationsänderungen*
- M 4.397 (C) *Protokollierung sicherheitsrelevanter Ereignisse von Web-Anwendungen und Web-Services*
- M 4.400 (B) *Restriktive Herausgabe sicherheitsrelevanter Informationen bei Webanwendungen und Web-Services*
- M 5.150 (Z) *Durchführung von Penetrationstests*

Goldene Regeln

Webanwendungen stellen Funktionalität zur Verfügung, die über einen Web-Client (Browser) abrufbar ist. Benutzereingaben werden verarbeitet und die Resultate in dynamisch erstellten Webseiten präsentiert. Häufig greift eine Webanwendung dabei auf unterschiedliche Hintergrundsysteme und -dienste (z. B. Datenbanken) zurück.

- Die Architektur der Webanwendung ist im Sicherheitskonzept zu dokumentieren. Anhand der Dokumentation muss nachvollziehbar sein, welche Komponenten existieren und welche Geschäfts- und Sicherheitsfunktionen implementiert sind.
- Bei Webanwendungen handelt es sich in der Regel um Individualsoftware. Als Basis für eine sichere Anwendung ist ein geregelter Entwicklungsprozess nach einem geeigneten Vorgehensmodell sicherzustellen. Dabei sind die Anforderungsanalyse, die Konzeption und das Design der Anwendung, die Entwicklung, das Testen sowie die Integration und die Softwareverteilung im Rahmen des Vorgehensmodells zu berücksichtigen. Die Sicherheit der Webanwendung ist in allen Phasen zu berücksichtigen.
- Alle Daten aus nicht-vertrauenswürdigen Quellen, die von der Webanwendung verarbeitet werden, müssen in eine einheitliche Form überführt werden (z. B. durch Kanonalisierung) bevor sie validiert und ggf. enkodiert oder maskiert werden. Hier ist das Whitelist-Verfahren zu bevorzugen. Um Schwachstellen wie beispielsweise Cross-Site-Scripting (XSS) und SQL-Injection zu vermeiden, müssen für alle Daten entsprechende Validierungs- und ggf. Enkodierungs- und Filterfunktionen verwendet werden.
- Für den Zugriff auf sensitive Funktionen oder Informationen muss die Webanwendung eine wirksame Authentisierung und Sitzungsverwaltung unterstützen. Die Eigenschaften der Sitzungs-ID müssen so gewählt werden, dass die Sitzung ausreichend geschützt ist.
- Eine Zugriffskontrolle muss die unbefugte Nutzung geschützter Funktionen sowie die Einsicht und Manipulation von geschützten Informationen verhindern können.

- Die Webanwendung sollte schützenswerte Daten bei der Übermittlung sicher übertragen. Dazu sind unter anderem Direktiven in den HTTP-Headern (z. B. Caching- und Cookie-Felder) und der Einsatz verschlüsselter Verbindungen (z. B. SSL/TLS) zu berücksichtigen. Schützenswerte Daten dürfen nicht in der URL übertragen werden.
- Die Preisgabe interner Informationen (z. B. in Kommentaren, Fehlermeldungen, HTTP-Headern) sollte möglichst restriktiv erfolgen. Nur Informationen, die für die Nutzung der Webanwendung notwendig sind, sollten dem Benutzer übermittelt werden.
- Fehler müssen so behandelt werden, dass sich die Webanwendung und die zugehörigen Komponenten zu jeder Zeit in einem sicheren Zustand befinden. Z. B. müssen Fehler bei der Autorisierung zu einer Verweigerung des Zugriffs führen.
- Die Logging-Funktionen der Webanwendung müssen alle sicherheitsrelevanten Ereignisse derart protokollieren, dass sicherheitskritische Vorfälle nachvollzogen werden können.
- Die Anwendungslogik sollte gegen automatisierte und logische Angriffe (z. B. DoS, Enumeration) geschützt werden. Dazu können Maßnahmen wie z. B. Grenzwerte für fehlgeschlagene Anmeldeversuche umgesetzt werden.
- Sicherheitsmechanismen (z. B. Authentisierung und Zugriffskontrolle) sind serverseitig zu implementieren und sollten durch clientseitige Mechanismen ergänzt werden, um Angriffe auf die Webanwendung über den Client abzuwehren.

Die Sicherheitsempfehlungen zum Thema Webanwendungen müssen zielgruppengerecht aufbereitet und institutionsweit veröffentlicht werden. Weitere Informationen zum Thema Webanwendungen finden sich im Baustein B 5.21 Webanwendungen und in den weiteren Bereichen der IT-Grundschutz-Kataloge.

G 2.1 Fehlende oder unzureichende Regelungen

Die Bedeutung übergreifender organisatorischer Regelungen und Vorgaben für das Ziel Informationssicherheit nimmt mit dem Umfang der Informationsverarbeitung, aber auch mit dem Schutzbedarf der zu verarbeitenden Informationen zu.

Von der Frage der Zuständigkeiten angefangen bis hin zur Verteilung von Kontrollaufgaben kann das Spektrum der Regelungen sehr umfangreich sein. Auswirkungen von fehlenden oder unzureichenden Regelungen werden beispielhaft in den anderen Gefährdungen des Gefährdungskatalogs G2 beschrieben.

Vielfach werden nach Veränderungen technischer, organisatorischer oder personeller Art, die wesentlichen Einfluss auf die Informationssicherheit haben, bestehende Regelungen nicht angepasst. Veraltete Regelungen können einem störungsfreien Betrieb entgegen stehen. Probleme können auch dadurch entstehen, dass Regelungen unverständlich oder zusammenhanglos formuliert sind und dadurch missverstanden werden.

Dass Regelungsdefizite zu Schäden führen können, machen folgende **Beispiele** deutlich:

- Durch eine mangelhafte Betriebsmittelverwaltung kann der termingerechte Arbeitsablauf in einem Rechenzentrum schon durch eine unterbliebene Druckerpapierbestellung stark beeinträchtigt werden.
- Neben einer Beschaffung von Handfeuerlöschern muss auch deren regelmäßige Wartung geregelt sein, um sicherzustellen, dass diese im Brandfall auch funktionstüchtig sind.
- Bei einem Wasserschaden wird festgestellt, dass dieser auch den darunter liegenden Serverraum in Mitleidenschaft zieht. Durch eine unzureichende Schlüsselverwaltung kann der Wasserschaden im Serverraum allerdings nicht unmittelbar behoben werden, weil keiner darüber informiert ist, wo sich der Schlüssel zum Serverraum gerade befindet. Dadurch steigt der Schaden erheblich.

G 2.4 Unzureichende Kontrolle der Sicherheitsmaßnahmen

Werden bereits eingeführte Sicherheitsmaßnahmen (z. B. Klassifizierung von Informationen, Datensicherung, Zutrittskontrolle, Vorgaben für Verhalten bei Notfällen) nicht konsequent umgesetzt und regelmäßig kontrolliert, kann es sein, dass sie nicht wirksam sind oder missachtet werden. Mängel, die bei einer Kontrolle festgestellt werden, lassen sich meist ohne Schaden abstellen. Wenn Verstöße erst anlässlich eines Schadensfalls auffallen, kann oft nicht mehr rechtzeitig und der jeweiligen Situation angemessen reagiert werden.

Darüber hinaus gibt es Sicherheitsmaßnahmen, die nur wirksam sind, wenn Verantwortliche sie kontrollieren. Hierzu zählen beispielsweise Protokollierungsfunktionen, deren Sicherheitseigenschaften erst zum Tragen kommen, wenn die Protokolldaten ausgewertet werden.

Beispiele:

- Zur Vorbereitung von Straftaten kommt es vor, dass Schließzylinder in Außentüren und Toren von nicht autorisierten Personen ausgetauscht werden. Gerade wenn es sich um Zugänge handelt, die selten genutzt werden oder lediglich als Notausgänge vorgesehen sind, werden diese bei Streifengängen nur in Panikrichtung geprüft. Die Funktionalität der Schließzylinder wird dabei oft vernachlässigt. Zur Vorbereitung von Straftaten kommt es vor, dass Schließzylinder in Außentüren und Toren von nicht autorisierten Personen ausgetauscht werden. Gerade wenn es sich um Zugänge handelt, die selten genutzt werden oder lediglich als Notausgänge vorgesehen sind, werden diese bei Streifengängen nur in Panikrichtung geprüft. Die Funktionalität der Schließzylinder wird dabei oft vernachlässigt.
- Die Sicherheitsleitlinie einer Institution schreibt vor, dass die eingesetzten Smartphones nicht "gerootet" werden dürfen bzw. dass kein "Jailbreak" durchgeführt werden darf, da so die Sicherheitseigenschaften des Betriebssystems umgangen werden können. Solche modifizierten Smartphones sind innerhalb einer Institution nicht mehr sicher einsetzbar. Wird diese Vorgabe nicht überprüft, ist es möglich, dass Mitarbeiter mit einem unsicheren Smartphone auf das Netz oder schützenswerte Informationen der Institution zugreifen.
- In einer Behörde werden einige Unix-Server zur externen Datenkommunikation eingesetzt. Aufgrund der zentralen Bedeutung dieser IT-Systeme sieht das Sicherheitskonzept vor, dass die Unix-Server wöchentlich einer Integritätsprüfung unterworfen werden. Da nicht regelmäßig kontrolliert wird, ob diese Überprüfungen tatsächlich stattfinden, fällt erst bei der Klärung eines Sicherheitsvorfalls auf, dass die IT-Abteilung auf solche Integritätsprüfungen verzichtet hat. Als Grund wurde die mangelhafte personelle Ausstattung der Abteilung genannt.
- In einem Unternehmen wurde die Rolle des z/OS-Security-Auditors nicht besetzt. Dies hatte zur Folge, dass die Einstellungen im RACF im Laufe der Zeit nicht mehr den Sicherheitsvorgaben des Unternehmens entsprachen. Erst nach einem Produktionsausfall wurde bemerkt, dass einige Anwender mehr Rechte hatten, als sie für ihre Tätigkeit benötigten. Eine für die Produktion wichtige Anwendung war von ihnen versehentlich gestoppt worden. In einem Unternehmen wurde die Rolle des z/OS-Security-Auditors nicht besetzt. Dies hatte zur Folge, dass die Einstellungen im RACF im Laufe der Zeit nicht mehr den Sicherheitsvorgaben des Unternehmens entsprachen. Erst nach einem Produktionsausfall wurde bemerkt, dass einige Anwender mehr Rechte hatten, als sie für ihre Tätigkeit benötigten.

Eine für die Produktion wichtige Anwendung war von ihnen versehentlich gestoppt worden.

G 2.7 Unerlaubte Ausübung von Rechten

Rechte wie Zutritts-, Zugangs- und Zugriffsberechtigungen werden als organisatorische Maßnahmen eingesetzt, um Informationen, Geschäftsprozesse und IT-Systeme vor unbefugtem Zugriff zu schützen. Werden solche Rechte an die falsche Person vergeben oder wird ein Recht unautorisiert ausgeübt, können sich eine Vielzahl von Gefahren für die Vertraulichkeit und Integrität von Daten oder die Verfügbarkeit von Rechnerleistung ergeben.

Beispiele:

- Der Arbeitsvorbereiter, der keine Zutrittsberechtigung zum Datenträgerarchiv besitzt, entnimmt in Abwesenheit des Archivverwalters Magnetbänder, um Sicherungskopien einspielen zu können. Durch die unkontrollierte Entnahme wird das Bestandsverzeichnis des Datenträgerarchivs nicht aktualisiert, die Bänder sind für diesen Zeitraum nicht auffindbar. Der Arbeitsvorbereiter, der keine Zutrittsberechtigung zum Datenträgerarchiv besitzt, entnimmt in Abwesenheit des Archivverwalters Magnetbänder, um Sicherungskopien einspielen zu können. Durch die unkontrollierte Entnahme wird das Bestandsverzeichnis des Datenträgerarchivs nicht aktualisiert, die Bänder sind für diesen Zeitraum nicht auffindbar.
- Ein Mitarbeiter ist erkrankt. Ein Zimmerkollege weiß aufgrund von Beobachtungen, wo dieser sein Passwort auf einem Merkzettel aufbewahrt und verschafft sich Zugang zum Rechner des anderen Mitarbeiters. Da er erst kürzlich durch ein Telefonat mitbekommen hat, dass der Kollege noch eine fachliche Stellungnahme abzugeben hatte, nimmt er hier unberechtigterweise diese Aufgabe im Namen seines Kollegen wahr, obwohl er zu der Thematik nicht auf dem aktuellen Sachstand ist. Eine daraus folgende Erstellung einer Ausschreibungsunterlage in der Verwaltungsabteilung fordert im Pflichtenheft daher eine längst veraltete Hardwarekomponente, weil die dortigen Mitarbeiter der fachlichen Stellungnahme des erfahrenen Kollegen uneingeschränkt vertraut haben.

G 2.22 Fehlende oder unzureichende Auswertung von Protokolldaten

In vielen IT-Systemen und Anwendungen sind Funktionalitäten integriert, um bestimmte Ereignisse in ihrem zeitlichen Ablauf protokollieren zu können. Dadurch werden in einem Informationsverbund oft große Mengen an Protokolldaten erzeugt, die sich nur schwer und mit einem hohen Zeitaufwand auswerten lassen. Allerdings ist eine sinnvolle Auswertung dieser Protokolldaten notwendig, um beispielsweise Fehleranalysen durchführen und erfolgte Angriffe identifizieren zu können.

Im Lebenszyklus eines IT-Systems kommen verschiedene Protokollierungskonzepte zum Einsatz. So werden während der Entwicklungsphase ausführliche Protokolle erstellt, um die Problemanalyse bei Fehlern zu erleichtern.

In der Einführungsphase werden Protokolle genutzt, um unter anderem die Performance des IT-Systems in der Produktivumgebung zu optimieren oder um die Wirksamkeit des Sicherheitskonzepts erstmals in der Praxis zu überprüfen.

In der Produktivphase dienen Protokolle hauptsächlich dazu, den ordnungsgemäßen Betrieb sicherzustellen. Über Protokolldaten werden dann unter anderem nachträglich Sicherheitsverletzungen im IT-System oder Angriffsversuche identifiziert. Die Protokollierung kann auch der Täterermittlung und in Folge der Abschreckung von potenziellen Tätern dienen. Über eine regelmäßige Auswertung der Protokolldaten lassen sich die Daten für Präventivmaßnahmen wie ein Frühwarnsystem heranziehen, wodurch unter Umständen vorsätzliche Angriffe auf ein IT-System frühzeitig erkannt oder vereitelt werden können.

Zentrale Protokollierung

Werden Protokolldaten an zentraler Stelle ausgewertet, ist es möglich, dass bei der großen Menge an Daten wichtige Informationen übersehen und zum Beispiel Angriffe nicht entdeckt werden. Aus diesem Grund gibt es Systeme, die den Administrator bei der Auswertung der Protokolldaten unterstützen oder die Daten sogar selbstständig auswerten. Je nach Produkt können die Informationen der verschiedenen Datenquellen miteinander vereint und zu einer Protokollmeldung verarbeitet werden. Es besteht jedoch die Gefahr, dass die Protokolldaten eventuell nicht mehr auf ihre ursprüngliche Datenquelle zurückgeführt werden können, sodass auch nicht mehr auf Anhieb nachvollzogen werden kann, wo das Ereignis ursprünglich aufgetreten ist.

Weitere Probleme bei der Auswertung können durch falsch eingestellte Filterfunktionen der Analysewerkzeuge entstehen. Das kann dazu führen, dass Protokolldaten, die für die Störungserkennung, Fehlersuche oder Frühwarnung erforderlich sind, nicht ausgewertet werden.

Beispiele:

- Ein Angreifer versucht, den Datenbank-Server durch eine DoS-Attacke zum Absturz zu bringen. Dieser Angriff wird auf dem betreffenden IT-System protokolliert. Der Angriff bleibt aber durch die fehlende Auswertung der Protokolldaten unentdeckt, und der Angreifer kann die DoS-Attacke bis zum Erfolg wiederholen.
- Bei einem Angriff auf einen Webserver wurde eine RPC-Sicherheitslücke dazu benutzt, um in das System einzudringen. Zwar hat der Webserver entsprechende Protokolldaten erzeugt, diese wurden jedoch aufgrund feh-

lerhafter Filtereinstellungen am zentralen Protokollierungsserver verworfen. Somit wurde kein automatischer Alarm ausgelöst, und der Angriff blieb unentdeckt.

G 2.27 Fehlende oder unzureichende Dokumentation

Verschiedene Formen der Dokumentation können betrachtet werden: die Produktbeschreibung, die Administrator- und Benutzerdokumentation zur Anwendung des Produktes und die Systemdokumentation.

Eine fehlende oder unzureichende Dokumentation der eingesetzten IT-Komponenten kann sowohl im Auswahl- und Entscheidungsprozess für ein Produkt, als auch bei einem Schadensfall im Wirkbetrieb erhebliche Auswirkungen haben.

Bei einer unzureichenden Dokumentation kann sich im Schadensfall, beispielsweise durch den Ausfall von Hardware bzw. Fehlfunktionen von Programmen, die Fehlerdiagnose und -behebung erheblich verzögern oder völlig undurchführbar sein.

Dies gilt auch für die Dokumentation von Leitungswegen und Verkabelungen innerhalb der Gebäude-Infrastruktur. Ist aufgrund unzureichender Dokumentation die genaue Lage von Leitungen nicht bekannt, so kann es bei Bauarbeiten außerhalb oder innerhalb eines Gebäudes zu Beschädigungen von Leitungen kommen. Dabei kann es zu längeren Ausfallzeiten (Eintritt eines Notfalls) oder unter Umständen sogar zu lebensbedrohenden Gefahren, zum Beispiel durch Stromschlag, kommen.

Beispiele:

- Wenn von einem Programm Arbeitsergebnisse in temporären Dateien zwischengespeichert werden, ohne dass dies ausreichend dokumentiert ist, kann dies dazu führen, dass die temporären Dateien nicht angemessen geschützt und vertrauliche Informationen offengelegt werden. Durch fehlenden Zugriffsschutz auf diese Dateien oder eine nicht korrekte physikalische Löschung der nur temporär genutzten Bereiche können Informationen Unbefugten zugänglich werden.
- Bei Installation eines neuen Softwareproduktes werden bestehende Konfigurationen abgeändert. Andere, bislang fehlerfrei laufende Programme, sind danach falsch parametrisiert und stürzen gegebenenfalls ab. Durch eine detaillierte Dokumentation der Veränderung bei der Installation von Software ließe sich der Fehler schnell lokalisieren und beheben.
- In einer größeren Behörde wurde die Verkabelung der IT durch eine externe Firma vorgenommen. Die Anfertigung einer Dokumentation war im Leistungsumfang nicht enthalten. Da nach Fertigstellung der Verkabelung mit der Firma kein Wartungsvertrag abgeschlossen wurde, verfügte die Behörde nicht über die notwendige Dokumentation. Erweiterungen des Netzes konnten nur mit erheblichen Verzögerungen vorgenommen werden (siehe auch G 2.12 *Unzureichende Dokumentation der Verkabelung*).
- In einer z/OS-Installation wurden jeden Abend automatisch Batch-Jobs zur Verarbeitung von Anwendungsdaten gestartet. Für die Verarbeitung war es wichtig, dass die Batch-Jobs in der richtigen Reihenfolge abliefen. Als eines Abends die Automation versagte, mussten die Jobs manuell gestartet werden. Aufgrund fehlender Dokumentation wurden die Batch-Jobs in der falschen Reihenfolge gestartet. Dies führte zu Abbrüchen in der Verarbeitung der Anwendungsdaten und zu Verzögerungen in der Produktion um mehrere Stunden.
- Fehlende Datenblätter von (flüchtigen) Halbleiterspeichern wie SRAM (Static Random Access Memory) und DRAM (Dynamic Random Access

Memory) können dazu führen, dass die Speicher nicht ordnungsgemäß gelöscht und damit vertrauliche Informationen bekannt werden können.

- In einem Unternehmen sollten USB-Sticks (nichtflüchtige Speicher) ausgemustert werden. Die Produktbeschreibungen waren nicht aufzufinden. Die USB-Sticks wurden daher mit dem vorhandenen Löschtool behandelt. Auf herstellerspezifische Besonderheiten konnte jedoch nicht eingegangen werden, so dass nicht alle Daten ordnungsgemäß und sicher gelöscht wurden.

G 2.67 Ungeeignete Verwaltung von Zugangs- und Zugriffsrechten

Wenn die Vergabe von Zugangs- und Zugriffsrechten schlecht geregelt ist, führt das schnell zu gravierenden Sicherheitslücken, z. B. durch Wildwuchs in der Rechtevergabe.

In vielen Institutionen ist die Verwaltung von Zugangs- und Zugriffsrechten eine extrem arbeitsintensive Aufgabe, weil sie schlecht geregelt ist oder die falschen Tools dafür eingesetzt werden. Dadurch kann z. B. viel "Handarbeit" erforderlich sein, die gleichzeitig wiederum sehr fehleranfällig ist. Außerdem sind in diesem Prozess dann auch häufig viele unterschiedliche Rollen und Personengruppen eingebunden, sodass hier auch leicht der Überblick über durchgeführte Aufgaben verloren geht.

Weiterhin gibt es auch Institutionen, in denen kein Überblick über alle auf den verschiedenen IT-Systemen eingerichteten Benutzer und deren Rechteprofil vorhanden ist. Typischerweise führt das dazu, dass sich Accounts finden von Benutzern, die die Behörde bzw. das Unternehmen längst verlassen haben oder die durch wechselnde Tätigkeiten zu viele Rechte aufgehäuft haben.

Wenn die Tools zur Verwaltung von Zugangs- und Zugriffsrechten schlecht ausgewählt wurden, sind diese oft nicht flexibel genug, um auf Änderungen in der Organisationsstruktur oder auf Wechsel der IT-Systeme angepasst zu werden.

Die Rollentrennung von Benutzern kann falsch vorgenommen worden sein, so dass Sicherheitslücken entstehen, beispielsweise durch falsche Zuordnung von Benutzern in Gruppen oder zu großzügige Rechtevergabe. Benutzer können Rollen zugeordnet werden, die nicht ihren Aufgaben entsprechen (zu viel oder zu wenig Rechte) oder die sie aufgrund ihrer Aufgaben nicht haben dürfen (Rollenkonflikte).

G 2.87 Verwendung unsicherer Protokolle in öffentlichen Netzen

Bei der Kommunikation über öffentliche Netze, insbesondere das Internet, existiert eine Reihe von Gefahren, die aus der Verwendung unsicherer Protokolle entstehen.

Eine wichtige Gefahr ist, dass vertrauliche Informationen in fremde Hände gelangen können. Als unsichere Protokolle müssen insbesondere solche Protokolle gelten, bei denen Informationen im Klartext übertragen werden. Da der Weg der Datenpakete im Internet nicht vorhersagbar ist, können in diesem Fall die übertragenen Informationen an verschiedensten Stellen mitgelesen werden. Besonders kritisch ist dies, wenn es sich um

- Authentisierungsdaten wie Benutzernamen und Passwörter,
- Autorisierungsdaten, beispielsweise Transaktionsnummern beim Electronic Banking oder Electronic Brokerage,
- andere vertrauliche Informationen, beispielsweise in Dokumenten, die per E-Mail verschickt werden, handelt.

Protokolle, bei denen sämtliche Informationen im Klartext übertragen werden, sind beispielsweise

- das Hypertext Transfer Protocol *HTTP*, das bei der Kommunikation zwischen Webbrowsern und Webservern verwendet wird,
- das *TELNET* Protokoll, das noch an einigen Stellen für Remote Logins verwendet wird,
- das File Transfer Protocol *FTP*, das noch häufig für den Zugriff auf Server benutzt wird, die Dateien zum Download bereitstellen,
- das Simple Mail Transfer Protocol *SMTP*, das zur Übertragung von E-Mail verwendet wird,
- die Protokolle *rsh* (Remote Shell), *rlogin* (Remote Login) und andere verwandte Protokolle.

Bei solchen Protokollen können sämtliche übertragenen Informationen auf jedem Rechner, über den eine entsprechende Verbindung läuft, mitgelesen und gegebenenfalls auch verändert werden. Kritisch ist beispielsweise die Übertragung von Kreditkartennummern oder Passwörtern über HTTP-Verbindungen im Internet.

Mittels Password-Sniffings können in einem ersten Schritt Passwörter bei der Übertragung zu einem System abgefangen werden. Dies erlaubt dem Angreifer anschließend auf dieses IT-System zu gelangen, um dann weitere Angriffe lokal auf dem Rechner durchzuführen.

Bei den erwähnten Protokollen (besonders bei *HTTP* oder *TELNET*) drohen auch sogenannte Man-in-the-middle-Angriffe oder Session Hijacking (siehe G 5.89 *Hijacking von Netz-Verbindungen*). Bei dieser Art von Angriffen ist ein Angreifer nicht nur dazu in der Lage, Informationen mitzulesen, sondern kann darüber hinaus aktiv Schaden anrichten, indem laufende Transaktionen verändert werden. Beispielsweise können Preise oder Bestellmengen bei Geschäften über das Internet so verändert werden, dass der Besteller nur die Artikel oder Lieferadresse sieht und bestätigt bekommt, die er eingibt, während der Angreifer eine wesentlich höhere Menge und eine andere Lieferadresse an den Verkäufer schickt.

Neben den erwähnten Protokollen, bei denen sämtliche Informationen im Klartext übertragen werden, existieren auch solche, bei denen zumindest die Über-

tragung der Authentisierungsdaten verschlüsselt erfolgt. Dabei droht jedoch immer noch das Mitlesen der übertragenen Nutzinformation.

G 2.103 Unzureichende Schulung der Mitarbeiter

IT-Benutzer aller Art werden häufig zu wenig in der Bedienung der von ihnen eingesetzten IT-Systeme geschult. Dies trifft leider sogar öfters auf Administratoren und Benutzerbetreuer zu. Vielfach werden teure Systeme und Anwendungen angeschafft, aber keine oder nur unzureichend Mittel für die Schulung der IT-Benutzer bereitgestellt.

Dies kann durch unabsichtliche Fehlbedienungen, falsche Konfiguration und ungeeignete Betriebsmittel zu gravierenden Sicherheitsproblemen führen. Häufig wenden Benutzer neu eingeführte Sicherheitsprogramme deswegen nicht an, weil sie nicht wissen, wie sie bedient werden und eine selbstständige Einarbeitung oft als zu zeitaufwendig im täglichen Arbeitsablauf gesehen wird. Daher reicht die Beschaffung und Installation einer Sicherheitssoftware noch lange nicht aus.

Beispiele:

- Während der Datenerfassung erschien eine dem Benutzer nicht bekannte Fehlermeldung. Da bei den meisten Fehlermeldungen das Anklicken von "ok" bisher keinen Schaden verursachte, wählte er an diesem Fall auch "ok". Nur diesmal bewirkte dies das Herunterfahren des Systems und folglich den Verlust der bis dahin eingegebenen Daten.
- Ein teures Firewall-System wurde beschafft. Der Administrator eines anderen IT-Systems wurde "durch Handauflegen" zum Administrator dieses Firewall-Systems bestimmt. Da er als unabkömmlich galt und alle verfügbaren Mittel für die System-Beschaffung verwendet worden waren, wurde er aber weder in der Bedienung der System-Plattform noch für den eingesetzten Firewall-Typ ausgebildet. Externe Seminare wurden aus Geldmangel verweigert, nicht einmal zusätzliche Handbücher angeschafft. Zwei Monate nach Inbetriebnahme des Firewall-Systems stellte sich heraus, dass durch eine Fehlkonfiguration der Firewall interne Systeme aus dem Internet frei zugänglich waren.
- In einem Unternehmen wurde die Migration auf ein neues Betriebssystem vorbereitet. Der dafür verantwortliche Mitarbeiter war zwar ein ausgezeichneter Kenner der bis dahin eingesetzten Plattform, kannte sich aber mit den diskutierten neuen Systemen nicht aus und erhielt auch keine dem entsprechende Schulung. Daher besuchte er einige kostenfreie Veranstaltungen eines Herstellers, dessen Produkte er auch danach favorisierte. Dies führte zu einer kostenintensiven Fehlentscheidung durch Einführung eines ungeeigneten Produktes.
- Für die Internet-Nutzung während der Dienstreisen wurden auf den Notebooks der Mitarbeiter Personal Firewalls installiert. Die Mitarbeiter wurden nicht dazu geschult, eine Abstimmung der Einstellungen der Firewall mit den Bedürfnissen der Mitarbeiter fand nicht statt. Viele Mitarbeiter haben daraufhin die Firewall abgeschaltet, um problemlos alle Internet-Seiten zu erreichen, die sie brauchten. Das Ergebnis war, dass schon nach einigen Wochen viele der Rechner mit Schadprogrammen verseucht waren. Neben dem Datenverlust war der Ansehensschaden erheblich, da sich ein Schadprogramm über Mails an Kunden weitergesendet hatte.

G 2.157 Mangelhafte Auswahl oder Konzeption von Webanwendungen

Eine Webanwendung nutzt in der Regel ein verteiltes, komplexes System bestehend aus unterschiedlichen Komponenten (z. B. Webserver, Applikationsserver, Hintergrundsysteme) und zugehörigen Schnittstellen. Oft sind diese in einer bestehenden Infrastruktur integriert, wobei der Schutz der Daten über alle Komponenten und Schnittstellen hinweg zu gewährleisten ist.

Individuell entwickelte Webanwendungen werden üblicherweise auf der Grundlage von Frameworks entworfen, die Basis-Funktionen zur Verfügung stellen und einsatzspezifisch konfiguriert bzw. abgesichert werden müssen. Im Rahmen der Konzeption sind Frameworks, Komponenten und Schnittstellen auszuwählen und deren Einbindung und Absicherung zu betrachten.

Im Gegensatz dazu ist bei der Konzeption von Webanwendungen auf Basis von Standardsoftware (z. B. Content Management Systeme) insbesondere auf die Auswahl der Software und die Konfiguration der Teilkomponenten zu achten. Dabei ist in diesem Zusammenhang unter Standardsoftware sowohl Free/Libre Open Source Software (FOSS/FLOSS) als auch kommerzielle Software zu verstehen.

Unabhängig davon, ob die Webanwendung als Individualentwicklung oder Standardsoftware umgesetzt wird, kann eine unzureichende Berücksichtigung deren Komplexität (z. B. von Frameworks, Komponenten und Schnittstellen) bei der Auswahl und Konzeption von Webanwendungen den Schutz der Daten gefährden.

Durch grundlegende Fehlentscheidungen in der Planungsphase können Schwachstellen entstehen, die möglicherweise nicht oder nur durch kostenintensive Nachbesserungen behoben werden können.

Beispiele:

Auswahl von Webanwendungen auf Basis von Standardsoftware

- Die Einsatzumgebung erfüllt nicht die Mindestanforderungen der Webanwendungen an die Hard- und Software. Somit ist die Integration in die bestehende Infrastruktur (z. B. Anbindung an eine Datenbank oder einen Identitätsspeicher) nicht möglich.
- Das ausgewählte Produkt verfügt nicht über eine ausreichende Sicherheitsfunktionalität, um schützenswerte Daten vor unbefugten Zugriffen zu schützen. Daher müssen die notwendigen Schutzmechanismen nachträglich hinzugefügt werden. Sollte das Produkt nicht um Schutzmechanismen erweiterbar sein, muss der Schutz perimetrisch (z. B. Web Application Firewall) realisiert werden. Hierdurch entstehen zusätzliche Aufwände.

Entwurf der Software-Architektur der Webanwendung

- Eine Sicherheitsfunktion (z. B. Authentisierung, Autorisierung) wird nicht nur an einer Stelle umgesetzt und verwendet, sondern ist mehrfach in der Webanwendung realisiert. Wird diese Sicherheitsfunktion an den verschiedenen Stellen unterschiedlich umgesetzt, führt dies zu einem uneinheitlichen Sicherheitsniveau. Darüber hinaus erhöht sich der Entwicklungs- und Wartungsaufwand bei redundant umgesetzten Funktionen.
- Die Sicherheitsfunktion wird ausschließlich clientseitig (z. B. im Web-Browser) umgesetzt. Wird die Konfiguration des Web-Browsers durch

einen Angreifer manipuliert, können die clientseitig umgesetzten Sicherheitsfunktionen umgangen werden. Somit kann der Angreifer unbefugt auf schützenswerte Daten und Funktionen zugreifen.

Integration und Betrieb der Webanwendung

- Die Hintergrundsysteme werden unzureichend abgesichert und dadurch ist die Datenbank der Webanwendung aus dem Internet erreichbar. Somit kann ein Angreifer direkt auf die Datenbank und die dort hinterlegten Daten zugreifen, ohne die Funktionen der Webanwendung zu nutzen. Die Sicherheitsmechanismen der Webanwendung werden demnach umgangen und können den unbefugten Zugriff auf die gespeicherten Daten in Hintergrundsystemen nicht verhindern.
- Es werden Anwendungskomponenten oder Frameworks zum Session-Management in einer unsicheren Konfiguration eingesetzt. Infolgedessen werden kurze SessionIDs mit einer langen Laufzeit verwendet. Somit kann ein Angreifer die SessionID eines authentisierten Benutzers erraten und die zugeordnete Sitzung übernehmen (siehe G 5.169 *Unzureichendes Session-Management von Webanwendungen und Web-Services*).

Erweiterung der Webanwendung

- Bestehende Sicherheitsfunktionen der Webanwendung werden durch falsch umgesetzte Funktionserweiterungen außer Kraft gesetzt. Wird die Webanwendung um ein Formular ergänzt, bei dem die Eingabedaten nicht validiert werden, kann dies von einem Angreifer für den unbefugten Zugriff auf schützenswerte Daten genutzt werden (z. B. bei einem SQL-Injection-Angriff; siehe G 5.131 *SQL-Injection*).

G 2.158 Mängel bei der Entwicklung und der Erweiterung von Webanwendungen und Web-Services

Wird eine Webanwendung oder ein Web-Service mit fehlenden oder unzureichenden Vorgaben und Standards entwickelt beziehungsweise erweitert, so kann dies zu Fehlern, Qualitätseinbußen oder einer unvollständig umgesetzten Funktionalität führen. Fehler, die in frühen Entwicklungsphasen der Anwendung gemacht werden, werden häufig erst in einem fortgeschrittenen Entwicklungsstadium entdeckt. Um diese Fehler nachträglich zu beheben, müssen oft aufwendige Änderungen vorgenommen werden. Dadurch können die Entwicklungskosten deutlich zunehmen. Im Fall von grundlegenden, architektonischen Fehlern ist die Webanwendung oder der Web-Service gegebenenfalls sogar komplett neu zu entwickeln.

Gibt es darüber hinaus keine Vorgaben für die Umsetzung von Sicherheitsmechanismen, wird der erforderliche Schutzbedarf (zum Beispiel hoher Schutzbedarf bezüglich Verfügbarkeit) der zu verarbeitenden Daten möglicherweise nicht erfüllt.

Nachfolgend werden Auswirkungen von fehlenden Vorgaben bei der Entwicklung und Erweiterung von Webanwendungen und Web-Services beispielhaft aufgeführt.

- Aufgrund eines fehlenden Vorgehensmodells bei der Softwareentwicklung (Software Development Lifecycle) werden nicht alle Entwicklungsphasen strukturiert durchlaufen, sodass Sicherheitsaspekte gar nicht oder erst in einer späten Entwicklungsphase berücksichtigt werden. In der Folge sinkt die Qualität der Sicherheitsfunktionen, wodurch das angestrebte Sicherheitsniveau nicht erreicht wird oder die Entwicklungskosten aufgrund notwendiger Nachbesserungen steigen.
- Fehlende Programmierrichtlinien (Coding Guidelines) führen zu einer uneinheitlichen Struktur und unterschiedlichen Ausprägungen von Programmierstilen und Sicherheitsmechanismen. Die Einarbeitung in den Programmcode bei der Erweiterung oder Wartung der Webanwendung oder des Web-Service wird dadurch erschwert. Demzufolge sind nachträgliche Änderungen und Erweiterungen nur mit hohem Aufwand umzusetzen und mit steigender Komplexität auch fehleranfälliger.
- Durch die falsche Spezifikation von (sicherheitsrelevanten) Testfällen und die unvollständige Auswahl von Testdaten werden nicht alle denkbaren Anwendungsfälle abgedeckt, sodass Fehler unerkannt bleiben. Wird zum Beispiel die Komponente einer Webanwendung oder eines Web-Service zur Filterung von Eingabedaten auf Basis unzureichender Testfälle und Testdaten geprüft, so werden unvollständig umgesetzte Filtermechanismen nicht erkannt.
- Falls funktionale und rechtliche Anforderungen an die Barrierefreiheit nicht erfüllt werden, ist die Webanwendung nur eingeschränkt von Menschen mit Behinderung nutzbar.

G 2.159 Unzureichender Schutz personenbezogener Daten bei Webanwendungen und Web-Services

Das Benutzerverhalten bei der Bedienung einer Anwendung kann durch das sogenannte *User Tracking* (üblicherweise ohne explizite Zustimmung des Benutzers) aufgezeichnet werden. Da häufig nicht der Betreiber der Webanwendung oder der von der Anwendung genutzten Web-Services die Datenauswertung durchführt, sondern diese als Dienstleistung integriert, werden die erhobenen Daten in der Regel auf Systemen von Drittanbietern gespeichert. Aus den aufgezeichneten Daten können mittels *User Profiling* Personenprofile erstellt werden, die nicht konform mit den Datenschutzbestimmungen sind. Damit besteht die Gefahr, gegen gesetzliche Vorschriften zu verstoßen.

Im Folgenden sind Beispiele für die unbefugte Sammlung personenbezogener Daten aufgeführt:

- Detaillierte Informationen zu Seitenaufrufen und Eingaben bei Webanwendungen und Web-Services werden Benutzern zugeordnet (zum Beispiel mittels Cookies) und über einen längeren Zeitraum protokolliert. Auf der Grundlage dieser Datensammlung können Personenprofile von den Benutzern der Webanwendung oder des Web-Service ohne ihr Wissen erstellt und zum Beispiel unbefugt für Werbezwecke verwendet werden.
- In den Webseiten der Webanwendung werden Bilder von fremden Servern eingebettet und somit von den Clients der Benutzer geladen. Anhand der angeforderten Bilder können die Betreiber der fremden Server Abrufstatistiken über die Webseiten der Webanwendung führen. Werden darüber hinaus IP-Adressen auf dem fremden Server protokolliert, können den Webseiten-Aufrufen IP-Adressen (und somit eventuell Benutzer) zugeordnet werden. Zusätzlich kann der fremde Server mittels Cookies (engl. *Third Party Cookies*) das Verhalten des Benutzers detailliert nachverfolgen.
- In den HTML-Seiten der Webanwendung ist JavaScript-Code eingebettet, der Anweisungen zur Sammlung von Daten über den Client (zum Beispiel installierte Plugins, grafische Auflösung) enthält. Bei dem Aufruf der Webseite werden diese Anweisungen unbemerkt vom Client ausgeführt. Die gesammelten Daten können demnach ohne Kenntnis und Zustimmung des Benutzers als Identifikationsmerkmale zur Erstellung von Benutzerprofilen verwendet werden.
- Es werden von der Webanwendung oder vom Web-Service personenbezogene Daten zwar rechtskräftig erhoben, jedoch nicht angemessen gespeichert, sodass sie von Dritten unbefugt ausgelesen werden können.
- Ein Web-Service überträgt personenbezogene Daten zur aufrufenden Anwendung oder zu anderen Web-Services mit ungesicherten Klartext-Protokollen über unsichere Netze.

G 3.16 Fehlerhafte Administration von Zugangs- und Zugriffsrechten

Zugangsrechte zu einem IT-System und Zugriffsrechte auf gespeicherte Daten und IT-Anwendungen dürfen nur in dem Umfang eingeräumt werden, wie sie für die Wahrnehmung der Aufgaben erforderlich sind. Werden diese Rechte fehlerhaft administriert, so kommt es zu Betriebsstörungen, falls erforderliche Rechte nicht zugewiesen wurden, bzw. zu Sicherheitslücken, falls über die notwendigen Rechte hinaus weitere vergeben werden.

Beispiel:

Durch eine fehlerhafte Administration der Zugriffsrechte hat ein Sachbearbeiter die Möglichkeit, auf die Protokolldaten zuzugreifen. Durch gezieltes Löschen einzelner Einträge ist es ihm daher möglich, seine Manipulationsversuche am Rechner zu verschleiern, da sie in der Protokolldatei nicht mehr erscheinen.

G 3.38 Konfigurations- und Bedienungsfehler

Konfigurationsfehler entstehen durch eine falsche oder nicht vollständige Einstellung von Parametern und Optionen, mit denen ein Programm gestartet wird. In diese Gruppe fallen unter anderem falsch gesetzte Zugriffsrechte für Dateien. Bei Bedienungsfehlern sind nicht einzelne Einstellungen falsch, sondern die IT-Systeme oder IT-Anwendungen werden falsch behandelt. Ein Beispiel hierfür ist das Starten von Programmen, die für den Einsatzzweck des Rechners nicht notwendig sind, aber von einem Angreifer missbraucht werden können.

Beispiele für aktuelle Konfigurations- bzw. Bedienungsfehler sind das Speichern von Passwörtern auf einem PC, auf dem ungeprüfte Software aus dem Internet ausgeführt wird, oder das Laden und Ausführen von schadhaften ActiveX-Controls. Diese Programme, die unter anderem, die Aufgabe haben, Webseiten durch dynamische Inhalte attraktiver zu machen, werden mit den gleichen Rechten ausgeführt, die auch der Benutzer hat. Sie können beliebig Daten löschen, verändern oder versenden.

Viele Programme, die für die ungehinderte Weitergabe von Informationen in einem offenen Umfeld gedacht waren, können bei falscher Konfiguration potenziellen Angreifern Daten zu Missbrauchszwecken liefern. So kann beispielsweise der *finger*-Dienst darüber informieren, wie lange ein Benutzer bereits am Rechner sitzt. Aber auch Browser übermitteln bei jeder Abfrage einer Datei eine Reihe von Informationen an den Webserver (z. B. die Version des Browsers und des verwendeten Betriebssystems, den Namen und die Internet-Adresse des PCs). In diesem Zusammenhang sind auch die Cookies zu nennen. Hierbei handelt es sich um Dateien, in denen Webserver-Betreiber Daten über den Benutzer auf dessen Rechner speichern. Diese Daten können beim nächsten Besuch des Servers abgerufen und vom Server-Betreiber für eine Analyse, der vom Benutzer vorher auf dem Server besuchten Webseiten, verwendet werden.

Der Einsatz eines Domain Name Systems stellt eine weitere Gefahrenquelle dar. Zum einen ermöglicht ein falsch konfigurierter DNS-Server die Abfrage von vielen Informationen über ein lokales Netz. Zum anderen hat ein Angreifer durch die Übernahme dieses Servers die Möglichkeit, gefälschte IP-Adressen zu verschicken, sodass jeglicher Verkehr von ihm kontrolliert werden kann.

Eine große Bedrohung geht auch von den automatisch ausführbaren Inhalten (*Executable Content*) in E-Mails oder HTML-Seiten aus. Dies ist unter dem Stichwort Content-Security-Problem bekannt. Dateien, die aus dem Internet geholt werden, können Code enthalten, der bereits beim "Betrachten" und ohne Rückfrage beim Benutzer ausgeführt wird. Dies ist z. B. bei Makros in Office-Dateien der Fall und wurde zum Erstellen von sogenannten Makro-Viren ausgenutzt. Auch Programmiersprachen und -schnittstellen wie ActiveX, Javascript oder Java, die für Anwendungen im Internet entwickelt worden sind, besitzen bei falscher Implementierung der Kontrollfunktionen ein Schadpotenzial.

Die Verfügbarkeit des Sicherheitssystems RACF ist bei z/OS-Betriebssystemen von zentraler Bedeutung für die Verfügbarkeit des gesamten Systems. Durch unsachgemäßen Einsatz von z/OS-Utilities bei der RACF-Datenbanksicherung oder fehlerhafte Bedienung der RACF-Kommandos kann diese eingeschränkt werden.

G 3.43 Ungeeigneter Umgang mit Passwörtern

Selbst die Nutzung von durchdachten Authentikationsverfahren hilft wenig, wenn die Benutzer nicht sorgfältig mit den benötigten Zugangsmitteln umgehen. Unabhängig davon, ob Passwörter, PINs oder Authentikationstoken eingesetzt werden, werden diese immer wieder weitergegeben oder unsicher aufbewahrt.

Benutzer geben oft aus Bequemlichkeit Passwörter an andere Personen weiter. Häufig werden Passwörter innerhalb von Arbeitsgruppen geteilt, um jedem Mitarbeiter den Zugriff auf gemeinsam zu bearbeitende Dateien zu erleichtern. Der Zwang zur Passwortbenutzung wird oft als lästig empfunden und dadurch unterlaufen, dass Passwörter selten bis nie gewechselt werden oder alle Mitarbeiter dasselbe Passwort benutzen.

Wird zur Benutzer-Authentisierung ein Token-basiertes Verfahren eingesetzt (zum Beispiel Chipkarte oder Einmal-Passwortgenerator), so ergibt sich bei Verlust die Gefahr, dass das Token unberechtigt verwendet wird. Ein unberechtigter Benutzer kann mit diesem Token unter Umständen erfolgreich eine Remote-Access-Verbindung aufbauen.

Aufgrund der Vielzahl verschiedener Passwörter und PINs können sich Benutzer diese oftmals nicht alle merken. Daher werden Passwörter immer wieder vergessen, was teilweise zu hohem Aufwand führt, um mit dem System weiterarbeiten zu können. Authentikationstoken können ebenso verloren werden. Bei sehr sicheren IT-Systemen kann der Verlust von Passwörtern oder Token sogar dazu führen, dass alle Benutzerdaten verloren sind.

Passwörter werden oft notiert, damit sie nicht vergessen werden. Dies ist solange kein Problem, wie sie sorgfältig, also vor unbefugtem Zugriff geschützt, aufbewahrt werden. Dies ist nicht immer der Fall. Ein klassisches Beispiel ist das Passwort unter der Tastatur oder auf einem Klebezettel am Bildschirm. Auch Authentikationstoken finden sich oft unter der Tastatur.

Ein anderer Trick, um Passwörter nicht zu vergessen, ist die "geeignete" Auswahl. Wenn Benutzer Passwörter selbst auswählen können und nicht ausreichend für die Probleme hierbei sensibilisiert sind, werden in vielen Fällen Trivial-Passwörter wie *4711* oder Namen von Freunden gewählt.

Beispiele:

- In einem Unternehmen wurde bei Stichproben festgestellt, dass viele Passwörter zu schlecht gewählt beziehungsweise zu selten gewechselt wurden. Es wurde technisch erzwungen, dass die Passwörter monatlich gewechselt wurden und außerdem Zahlen oder Sonderzeichen enthalten mussten. Es stellte sich heraus, dass ein Administrator seine Passwörter wie folgt auswählte: *Januar2009, Februar2009, Maerz2009, ...*. Diese Passwörter entsprachen zwar den Vorgaben, waren aber leicht zu erraten.
- In einer Behörde zeigte sich, dass einige der Benutzer, die ihre Büros zur Straßenseite hatten, dasselbe Passwort benutzten: den Namen des gegenüberliegenden Hotels, der in großen Leuchtbuchstaben die Aussicht dominierte. Mitarbeiter einer Institution geben untereinander Anmeldeinformationen (Nutzername und Passwort) weiter, die zur Nutzung eines Cloud Services berechtigen. Der betroffene Cloud-Dienst wurde in der Folge von mehreren Benutzern unter der gleichen Kennung verwendet. Dadurch konnten die Benutzer nicht mehr unterschieden werden und es war

nicht mehr möglich nachzuvollziehen, welcher Mitarbeiter tatsächlich aktiv angemeldet war und wer welche Informationen weitergegeben oder bearbeitet hat.

G 4.22 Software-Schwachstellen oder -Fehler

Für jede Art von Software gilt: je komplexer sie ist, desto häufiger treten Programmier- oder Designfehler auf. Unter Software-Schwachstellen sollen unbeabsichtigte Programmfehler verstanden werden, die dem Anwender nicht oder noch nicht bekannt sind und ein Sicherheitsrisiko für das IT-System darstellen. Es werden ständig neue Sicherheitslücken in vorhandener, auch in weitverbreiteter oder ganz neuer Software gefunden.

Zu Fehlern oder Schwachstellen in Software kann es durch eine Vielzahl von Gründen kommen. Dazu gehören beispielsweise Kommunikationsfehler zwischen Kunden und Entwicklern, unzureichende Ausbildung der Programmierer oder ungenügende Tests. Auch zu hohe Erwartungen der Anwender und zeitlich zu knapp bemessene Fertigstellungstermine können dazu führen, dass die Hersteller ihre Produkte teilweise unausgereift oder nicht fehlerfrei anbieten.

Werden Softwarefehler nicht erkannt, können die bei der Anwendung entstehenden Fehler zu weitreichenden Folgen führen. Bei weitverbreiteter Standardsoftware können Software-Schwachstellen schnell dazu führen, dass weltweit schwerwiegende Sicherheitsprobleme für alle Arten von Institutionen entstehen können.

Beispiele:

- Ein Software-Fehler in der Sicherheitssoftware RACF des z/OS-Betriebssystems kann bedeuten, dass nicht nur RACF den Dienst einstellt, sondern dadurch das ganze System nicht mehr funktionsfähig ist und neu gestartet werden muss.
- Die Stärke der in Standardsoftware implementierten Sicherheitsfunktionalitäten (wie Passwörter oder Verschlüsselungsalgorithmen) wird vom Anwender häufig zu hoch eingeschätzt. Häufig können diese Sicherheitsfunktionalitäten einem sachkundigen Angriff nicht dauerhaft standhalten. Dies gilt z. B. für die Verschlüsselungsfunktionen, die in vielen Textverarbeitungsprogrammen integriert sind. Für fast alle davon gibt es im Internet zahlreiche Tools, um diese Verschlüsselung zu überwinden.
- Nachweislich führte das Auftreten eines bestimmten Wortes in der Rechtschreibprüfung eines Textverarbeitungsprogrammes immer zu dessen Absturz.
- Vielfach enthält Standardsoftware nicht dokumentierte Funktionen, wie sog. "Ostereier" oder "Gagscreens", mit denen sich die Entwickler des Produktes verewigt haben. Zum einen werden hierdurch zusätzliche IT-Ressourcen verbraucht, zum anderen wird dadurch auch deutlich, dass im Softwaretest die gesamte Funktionalität des Produktes nicht bis ins Letzte geklärt werden kann.
- Die meisten Warnmeldungen der Computer Emergency Response Teams in den letzten Jahren bezogen sich auf sicherheitsrelevante Programmierfehler. Dies sind Fehler, die bei der Erstellung von Software entstehen und dazu führen, dass diese Software von Angreifern missbraucht werden kann. Der größte Teil dieser Fehler wurde durch Speicherüberläufe (Buffer Overflow) hervorgerufen. Hierbei handelt es um Fehler, bei denen eine Routine zum Einlesen von Zeichen nicht prüft, ob die Länge der eingegebenen Zeichenkette mit der Länge des dafür vorgesehenen Speicherbereiches übereinstimmt. Dadurch ist es Angreifern möglich, eine überlange Zeichenfolge zu übertragen, so dass hinter dem für die Eingabe reservierten Speicherbereich zusätzliche Befehle gespeichert werden können, die

zur Ausführung gebracht werden. Diese Befehle können zum Beispiel beliebige Programme sein.

- Eine weitere große Anzahl von Warnmeldungen wurde durch Verfügbarkeitsangriffe (Denial of Service, DoS) verursacht, bei denen durch Fehler in einzelnen Routinen, die für die Netzdatenverarbeitung eingesetzt werden, der gesamte Rechner zum Absturz gebracht werden kann.
- Die unzureichende Absicherung der Registrierung zur Nutzung eines Cloud Services führt dazu, dass ein Benutzer den Service in der Folge unter einem falschen Namen missbrauchen kann. Der Benutzer registriert sich dabei beim Cloud Service im Namen eines anderen Cloud-Service-Benutzers. Bei der Registrierung wird auf eine ausreichende Absicherung, beispielsweise mithilfe eines Aktivierungs-Links unter der angegebenen E-Mail-Adresse, verzichtet.

G 4.33 Schlechte oder fehlende Authentikation

Authentikationsmechanismen können zur Authentikation von Benutzern oder Komponenten oder zur Bestimmung des Datenursprungs eingesetzt werden. Wenn Authentikationsmechanismen fehlen oder zu schlecht sind, besteht die Gefahr, dass

- Unbefugte auf IT-Systeme oder Daten Zugriff nehmen können,
- die Verursacher von Problemen nicht identifiziert werden können oder
- die Herkunft von Daten nicht bestimmt werden kann.

Weiter besteht die Gefahr, dass Sicherheitslücken entstehen

- bei der Benutzerauthentikation, wenn z. B. Benutzer Passwörter wählen, die einfach zu erraten sind, oder wenn sie die Passwörter nie wechseln,
- bei der Komponentenauthentikation, wenn z. B. nach Inbetriebnahme eines IT-Systems Default-Passwörter nicht durch individuell gewählte ersetzt werden, wenn die Passwörter, die bei vielen IT-Systemen fest eingegeben werden, nie wieder geändert werden oder wenn die Passwörter nicht sicher hinterlegt werden und sich nach einem Systemabsturz herausstellt, dass das jetzt dringend benötigte Passwort vergessen wurde,
- bei der Wahl der Verfahren, wenn diese z. B. völlig untauglich sind oder Sicherheitslücken bekannt werden, auf die im laufenden Betrieb aber nicht reagiert wird.

G 4.35 Unsichere kryptographische Algorithmen

Der Sicherheitszugewinn durch Einsatz kryptographischer Verfahren ist grundsätzlich von zwei Parametern abhängig: es müssen sichere kryptographische Algorithmen eingesetzt werden und die geheimen Schlüssel müssen vertraulich gehandhabt werden (zur Kompromittierung kryptographischer Schlüssel siehe G 5.83 *Kompromittierung kryptographischer Schlüssel*).

Unsichere kryptographische Algorithmen sind dadurch gekennzeichnet, dass es einem potentiellen Angreifer mit vertretbaren Ressourcen gelingt, das eingesetzte kryptographische Verfahren zu brechen. Bei Verschlüsselungsalgorithmen bedeutet dies, dass es gelingt, aus dem verschlüsselten Text den ursprünglichen Klartext zu ermitteln, ohne dass zusätzliche Informationen bekannt sind. Dabei sind als relevante Ressourcen auf Angreiferseite z. B. die verfügbare Rechenleistung, Hilfsmittel wie Analysetools, vorhandene Kenntnisse, verfügbare Arbeitszeit, Kenntnisse über Schwachstellen etc. zu berücksichtigen. Werden also unsichere kryptographische Algorithmen eingesetzt, besteht für Angreifer die Möglichkeit, den kryptographischen Schutz zu unterlaufen.

Ob jedoch ein kryptographischer Algorithmus unsicher ist, muss jeweils im Einzelfall untersucht werden. Es gibt jedoch einige Kriterien, die auf Unsicherheiten schließen lassen:

- Werden bei symmetrischen Verschlüsselungsverfahren geheime Schlüssel benutzt, deren effektive Länge geringer als 100 Bit ist, so können sie heute mit moderatem Rechnereinsatz durch Ausprobieren aller potentiell möglichen Schlüssel gebrochen werden. Mit steigender Rechnerleistung ist anzunehmen, dass diese Grenze in Zukunft über 100 Bit steigen wird.
- Werden bei asymmetrischen Verschlüsselungs- und Signaturverfahren Algorithmen eingesetzt, deren Sicherheit auf dem Problem des Faktorisierens großer Zahlen basiert, so wird heute angenommen, dass Schlüssellängen von weniger als 2000 Bit als unsicher zu betrachten sind. Dies begründet sich in den Fortschritten bei der Entwicklung effizienter Faktorisierungsalgorithmen, die heute unter massivem Rechnereinsatz Faktorisierungen von Zahlen mit rund 500 Bit erlauben. Daneben ist die mögliche Entwicklung von opto-elektronischen Beschleunigern für einen wesentlichen Teil-Rechenschritt bei diesen Verfahren in Betracht zu ziehen, was diese deutlich beschleunigen würde.
- Hashfunktionen, die eine beliebig lange Zeichenkette auf einen Hashwert mit konstanter Bitlänge abbilden, können als unsicher betrachtet werden, wenn die konstante Länge des Hashwertes geringer ist als 128 Bit, da sonst zwei Zeichenketten ermittelt werden können, die den gleichen Hashwert ergeben.
- Kryptographische Algorithmen, die von unerfahrenen Entwicklern entworfen wurden und nicht in der wissenschaftlichen Szene untersucht wurden, sollten als potentiell unsicher betrachtet werden, da die Entwicklung sicherer kryptographischer Algorithmen langjährige Erfahrung voraussetzt.
- Nicht veröffentlichte kryptographische Algorithmen, die auffällig schnell in Software ablaufen, sollten ebenfalls als potentiell unsicher betrachtet werden. Die Erfahrung zeigt, dass sichere Algorithmen meist auf komplexen mathematischen Funktionen beruhen müssen.
- Bei der Anwendung kryptographischer Verfahren werden häufig Zufallszahlen benötigt. Schlechte Zufallszahlengeneratoren können dazu führen, dass die damit erzeugten Werte vorhersagbar sind. Dadurch können z. B.

kryptographische Checksummen, die die Nachrichtenintegrität sicherstellen sollen, wertlos werden.

Von diesen Kriterien betroffen ist beispielsweise der weltweit sehr häufig eingesetzte DES-Algorithmus zur symmetrischen Verschlüsselung. Dieser benutzt eine effektive Schlüssellänge von 56 Bit. Der so genannte Triple-DES-Algorithmus als dreifache Hintereinanderausführung mit zwei Schlüsseln hat eine effektive Schlüssellänge von 112 Bit und kann zurzeit noch als ausreichend sicher betrachtet werden. Auch betroffen ist der RSA-Algorithmus, der als asymmetrisches Verfahren auf dem Faktorisierungsproblem basiert. Wird RSAmit einer Schlüssellänge unter 1024 Bit betrieben, muss davon ausgegangen werden, dass dies keine ausreichende Sicherheit bietet. Für die nächsten Jahre kann eine Schlüssellänge von mindestens 2000 Bit noch als ausreichend sicher angesehen werden.

Der Hash-Algorithmus MD5 ist veraltet und weist bekannte Schwächen auf, die auch bereits anhand praktischer Beispiele demonstriert werden konnten. Auch der Hash-Algorithmus SHA-1 ist nicht mehr für alle Einsatzzwecke geeignet.

Ein häufiges Beispiel unsicherer, aber sehr schneller Algorithmen ist die so genannte XOR-Funktion, bei der konstante Werte mit dem ursprünglichen Klartext auf einfache Weise verknüpft werden. Dies ist ein hochperformanter Algorithmus, der jedoch sehr schnell gebrochen werden kann. Die XOR-Funktion kann andererseits aber der sicherste Verschlüsselungsalgorithmus überhaupt sein, wenn die zu verschlüsselnden Daten mit nicht vorhersagbaren Zufallswerten XOR-iert werden (One-Time-Pad).

Für den Laien ist es praktisch unmöglich, festzustellen, ob ein kryptographischer Algorithmus ausreichend sicher ist. Daher sollten nur solche Algorithmen eingesetzt werden, die bekanntermaßen von Experten entwickelt wurden oder die einer langjährigen Untersuchung durch die wissenschaftliche Szene unterzogen wurden.

G 4.84 Unzureichende Validierung von Ein- und Ausgabedaten bei Webanwendungen und Web-Services

Webanwendungen werden im Allgemeinen von generischen Clients (Web-Browsern) verwendet, sodass Benutzer beliebige Eingabedaten an den Server übermitteln können. Auf Web-Services wird dagegen durch andere Anwendungen oder Dienste zugegriffen (beispielsweise Smartphone-Apps). Eingabedaten können aber auch hier oft modifiziert werden, beispielsweise durch den Einsatz eines Proxys oder durch Manipulation der Clients. Werden schadhafte Eingaben eines Angreifers von der Webanwendung beziehungsweise dem Web-Service verarbeitet, können möglicherweise Schutzmechanismen umgangen werden.

Beispiele für Angriffe, die auf einer unzureichenden Validierung von Eingabedaten beruhen, sind SQL-Injection (siehe G 5.131 *SQL-Injection*), Path Traversal (siehe G 5.172 *Umgehung der Autorisierung bei Webanwendungen und Web-Services*) und Remote File Inclusion. Diese Angriffe können Unbefugten Zugriff auf das Betriebssystem oder auf Hintergrundsysteme ermöglichen. Bei einem erfolgreichen Angriff können schützenswerte Daten unautorisiert ausgelesen oder manipuliert werden.

Nachdem die Webanwendung beziehungsweise der Web-Service die Eingabedaten erfolgreich verarbeitet hat, werden üblicherweise wieder Daten ausgegeben. Die Ausgabedaten werden entweder direkt an den Browser des Benutzers (zum Beispiel Statusmeldungen oder ein neuer Eintrag im Gästebuch) oder die aufrufende Anwendung übermittelt oder an nachgelagerte Systeme weitergereicht. Werden die Daten vor der Ausgabe nicht ausreichend validiert, könnten die Ausgaben Schadcode enthalten, der auf den Zielsystemen interpretiert oder ausgeführt wird.

Die folgenden Beispiele beschreiben mögliche Auswirkungen einer unzureichenden Validierung von Ein- und Ausgaben:

- Eine Webanwendung beziehungsweise ein Web-Service verwendet Eingabedaten ungefiltert zur Erzeugung von Datenbankabfragen. Dies kann ein Angreifer ausnutzen und eine Anfrage formulieren, die neben den regulären Eingabedaten zusätzliche Befehle für die Datenbank enthält. Durch das ungefilterte Einbetten der Eingabedaten in die Datenbankabfrage werden die Befehle von der Datenbank ausgeführt. So kann der Angreifer direkten Zugriff auf die Datenbank erhalten.
- Eine Webanwendung bietet eine Funktion zum Datei-Upload an und schränkt diese auf gewisse Dateitypen ein. Zur Bestimmung des Dateityps überprüft die Webanwendung ausschließlich die Dateiendung und berücksichtigt dabei nicht den Inhalt der Datei. Wird eine erlaubte Dateiendung für den Upload verwendet, können so Dateien mit beliebigem Inhalt zum Server übermittelt werden.
- Werden Eingabedaten durch die Filterkomponente automatisiert geändert und angepasst (Sanitizing), können die Daten durch gezielte Eingaben eines Angreifers von der Filterkomponente in einen Angriffsvektor überführt werden.
- Ein- und Ausgabedaten können in verschiedenen Kodierungen (zum Beispiel UTF-8, ISO 8859-1) und Notationen (zum Beispiel bei UTF-8 ist "." = "2E" = "C0 AE") vorliegen. Abhängig vom angewandten Kodierungssche-

ma kann der gleiche Wert unterschiedlich interpretiert werden. Interpretiert die Filterkomponente die Daten anders als die verarbeitenden Komponenten der Webanwendung oder des Web-Service, so kann ein Angreifer schadhafte Daten (zum Beispiel SQL-Anweisungen) derart codieren, dass sie bei der Filterung nicht erkannt werden. Somit werden die vom Angreifer schadhaften Daten an die verarbeitenden Komponenten weitergereicht und aufgrund der unterschiedlichen Interpretation ausgeführt.

- Die Kommentar-Funktion einer Webanwendung erlaubt eine Formatierung der Texte durch HTML. Die Eingaben werden zum Beispiel nicht auf spezielle HTML-Tags eingeschränkt, sodass ein Angreifer über diese Funktion beliebigen HTML-Code auf der Webanwendung platzieren kann. Dies kann ein Angreifer dazu nutzen, um Elemente der Webseite zu manipulieren oder zu überlagern und Benutzereingaben abzufangen (siehe G 5.175 *Clickjacking*). Derselbe Angriff ist übertragbar auf Web-Services, welche HTML-Code als Eingabe erlauben und diesen ungefiltert in ihre Ausgabe übernehmen.

G 4.85 Fehlende oder mangelhafte Fehlerbehandlung durch Webanwendungen und Web-Services

Treten Fehler während des Betriebs einer Webanwendung oder eines Web-Service auf, kann dies unvorhersehbare Auswirkungen haben und die Verfügbarkeit der Webanwendung oder des Web-Service bis zur Unerreichbarkeit einschränken. So werden gegebenenfalls Aktionen unvollständig durchgeführt, zwischengespeicherte Zustände und Daten gehen verloren oder Sicherheitsmechanismen fallen aus. Werden Fehler nicht korrekt behandelt, kann sowohl der Betrieb als auch der Schutz der Funktionen und Daten einer Webanwendung oder eines Web-Service nicht mehr gewährleistet werden.

Beispiele:

- Im laufenden Betrieb belegen Webanwendungen und Web-Services üblicherweise Ressourcen, wie offene Netz- oder Datei-Streams, um auf Hintergrundsysteme, zwischengespeicherte Zustände oder sonstige Daten zugreifen zu können. Solange die Webanwendung/der Web-Service auf diese Ressourcen zugreift, sind diese häufig für den exklusiven Zugriff reserviert und können nicht durch andere Prozesse verwendet werden. Werden im Fehlerfall die belegten Ressourcen nicht ordnungsgemäß freigegeben, so bleiben diese gegebenenfalls in einem blockierten Zustand. Dadurch können Daten verloren gehen, da beispielsweise zwischengespeicherte Änderungen nicht ordnungsgemäß geschrieben werden können.
- Treten während der Ausführung der Sicherheitskomponenten (zum Beispiel Authentisierung, Autorisierung) Fehler auf und werden diese unzureichend behandelt, so werden angeforderte Aktionen möglicherweise unkontrolliert ausgeführt. Aktionen, die im normalen Zustand abgelehnt werden, könnten im Fehlerfall zugelassen werden.
- Fehlermeldungen können detaillierte Hinweise zur Fehlerursache beinhalten, die für den Benutzer nicht notwendig sind jedoch gezielte Angriffe ermöglichen. Zu diesen detaillierten Informationen gehören Stacktraces, Debugging-Ausgaben, Fehlermeldungen bei ungültigen SQL-Abfragen, Angaben zu verwendeten Webservern und anderen Anwendungskomponenten. Auch scheinbar unkritische Informationen, wie die Meldung bei einer fehlgeschlagenen Anmeldung mit Benutzernamen und Passwort, dass der Benutzernamen bekannt ist, aber ein ungültiges Passwort eingegeben wurde, können zum Beispiel im Rahmen von Brute-Force-Angriffen ausgenutzt werden. In diesem Fall weiß der Angreifer, dass der Benutzername existiert.
- Wird die Fehlerbehandlung ausschließlich auf dem Client (zum Beispiel Webbrowser oder Anwendung, welche einen Web-Service nutzt) durchgeführt, kann sie manipuliert oder außer Kraft gesetzt werden. Ein Angreifer kann somit die Behandlung der Fehler beeinflussen und steuern.

G 4.86 Unzureichende Nachvollziehbarkeit von sicherheitsrelevanten Ereignissen bei Webanwendungen

Werden sicherheitsrelevante Ereignisse von der Webanwendung unzureichend protokolliert, können diese zu einem späteren Zeitpunkt nicht nachvollzogen und die Ursache nicht mehr ermittelt werden. Kritische Fehler und Angriffe bleiben gegebenenfalls unbemerkt und die Behebung einer Schwachstelle ist dann nicht oder nur unter erschwerten Bedingungen möglich.

Werden darüber hinaus Ereignisse auf der System- und Netzebene nur eingeschränkt protokolliert, sind sicherheitsrelevante Vorfälle nur noch schwer zu erkennen und nachzuvollziehen.

Beispiele:

- Sicherheitsrelevante Ereignisse der Webanwendung werden nicht oder nur eingeschränkt protokolliert. So bleiben unbefugte Konfigurationsänderungen (z. B. durch einen Angreifer) unentdeckt.
- Es werden nicht alle notwendigen Eigenschaften eines Ereignisses protokolliert, sodass Vorgänge nicht vollständig nachvollzogen werden können (z. B. nur das Datum, aber keine Uhrzeit).
- Ist der Schutz der Protokolldaten nicht gewährleistet, können sie unbemerkt manipuliert werden. Somit kann ein Angreifer Hinweise auf durchgeführte Aktionen löschen und der Angriff bleibt unentdeckt oder ist nicht mehr nachvollziehbar.

G 4.87 Offenlegung vertraulicher Informationen bei Webanwendungen und Web-Services

Webseiten und Daten, die von einer Webanwendung oder einem Web-Service generiert und ausgeliefert werden, können vertrauliche Informationen enthalten, die nicht für die Nutzung der Webanwendung oder des Web-Service erforderlich sind (zum Beispiel Angaben zu Produkt und Versionsständen von Frameworks oder Informationen über Schnittstellen und Funktionen in WSDL-Dokumenten oder im UDDI-Register). Diese Informationen können einem Angreifer Hinweise zur Durchführung gezielter Angriffe geben. Wenn demzufolge Informationen unnötig offengelegt werden, kann dies einen erfolgreichen Angriff erleichtern. Hierbei können diese Informationen auch über weniger offensichtliche Übertragungswege übermittelt werden (zum Beispiel im HTTP-Header).

Beispiele:

- Es werden detaillierte Informationen über Sicherheitsmechanismen oder -attribute ausgegeben, die für einen Benutzer der Webanwendung oder des Web-Service nicht notwendig sind, aber Hinweise für potenzielle Angriffe geben (zum Beispiel "Geben Sie bitte die 6-stellige, numerische PIN ein" anstelle von "Geben Sie bitte die PIN ein"). Aufgrund dieser Information könnte ein Angreifer den möglichen Zeichenraum bei einem Brute-Force-Angriff einschränken und somit schneller eine gültige PIN ermitteln.
- Kommentare (zum Beispiel im HTML-Quelltext) können Informationen zu bekannten Fehlern, Funktionsweisen, eingesetzten Techniken und der angebundenen Infrastruktur beinhalten. Ein Angreifer kann hierdurch gezielt nach Schwachstellen in der Webanwendung, dem Web-Service und der Infrastruktur suchen und diese ausnutzen. Werden zum Beispiel die in der Entwicklungsphase verwendeten Zugangsdaten für eine Datenbank in Kommentaren erwähnt, können diese gegebenenfalls auch noch im produktiven Betrieb der Webanwendung für den unautorisierten Zugriff verwendet werden.
- Dateien mit unbekannter Dateiendung (zum Beispiel temporäre Dateien mit *.tmp* oder Backup-Dateien mit *.bak* von Skripten der Webanwendung) werden von der Webanwendung im Quelltext ausgeliefert. Auf diese Weise können vertrauliche Informationen wie fest kodierte Zugangsdaten ausgelesen werden. Darüber hinaus können von einem Angreifer Programmabläufe aus dem offengelegten Code auf Schwachstellen untersucht werden.
- In WSDL-Dateien werden die Schnittstellen von Web-Services definiert. Werden diese Dateien einem Angreifer offengelegt, kann er dadurch Informationen über die Schnittstellen und Aufrufparameter der angebotenen Web-Services erhalten, die Angriffe auf diese Schnittstellen deutlich erleichtern oder beschleunigen.

G 5.18 Systematisches Ausprobieren von Passwörtern

Zu einfache Passwörter lassen sich durch systematisches Ausprobieren herausfinden. Dabei ist zwischen dem simplen Ausprobieren aller möglichen Zeichenkombinationen bis zu einer bestimmten Länge (sogenannter Brute-Force-Angriff) und dem Ausprobieren anhand einer Liste mit Zeichenkombinationen (sogenannter *Wörterbuch-Angriff*) zu unterscheiden. Beide Ansätze lassen sich auch kombinieren.

Die meisten Betriebssysteme verfügen über eine Datei oder Datenbank (z. B. *passwd-* bzw. *shadow-Datei* bei Unix oder RACF-Datenbank bei z/OS) mit den Kennungen und Passwörtern der Benutzer. Allerdings werden zumindest die Passwörter bei vielen Betriebssystemen nicht im Klartext gespeichert, sondern es kommen kryptographische Mechanismen zum Einsatz. Ist die Datei nur unzureichend gegen unbefugten Zugriff geschützt, kann ein Angreifer diese Datei möglicherweise kopieren und mit Hilfe leistungsfähigerer Rechner und ohne Einschränkungen hinsichtlich der Zugriffszeit einem Brute-Force-Angriff aussetzen.

Die Zeit, die bei einem Brute-Force-Angriff zum Herausfinden eines Passworts benötigt wird, hängt ab von

- der Dauer einer einzelnen Passwortprüfung,
- der Länge des Passworts und
- der Zeichenzusammensetzung des Passworts (z. B. Buchstaben/Zahlen).

Die Dauer einer einzelnen Passwortprüfung hängt stark vom jeweiligen System und dessen Verarbeitungs- bzw. Übertragungsgeschwindigkeit ab. Im Falle eines Angriffs spielen auch die Methode und die Technik des Angreifers eine Rolle.

Länge und Zeichenzusammensetzung des Passworts lassen sich dagegen durch organisatorische Vorgaben oder sogar durch technische Maßnahmen beeinflussen.

Beispiel:

- Mit einem gut ausgestatteten PC lassen sich derzeit bei der Standard-Passwort-Verschlüsselung von Unix bzw. Linux etwa 400.000 Passwortprüfungen pro Sekunde durchführen. Bei der Standard-Passwort-Verschlüsselung von Windows NT/2000/XP sind es sogar über 6.000.000 Prüfungen pro Sekunde (Quelle: Der Hamburgische Datenschutzbeauftragte, 2003). Bei einem Zeichenvorrat von 26 Zeichen dauert es somit etwa 6 Tage, um ein 8 Zeichen langes Passwort unter Unix bzw. Linux zu ermitteln (Standard-Passwort-Verschlüsselung). Die gleiche Aufgabe dauert unter Windows sogar nur etwa 9 Stunden.

G 5.19 Missbrauch von Benutzerrechten

Eine missbräuchliche Nutzung liegt vor, wenn man vorsätzlich recht- oder unrechtmäßig erworbene Möglichkeiten ausnutzt, um dem System oder dessen Benutzern zu schaden.

In nicht wenigen Fällen verfügen Anwender aus systemtechnischen Gründen über höhere oder umfangreichere Zugriffsrechte, als sie für ihre Tätigkeit benötigen. Diese Rechte können zum Ausspähen von Daten verwendet werden, auch wenn Arbeitsanweisungen den Zugriff verbieten.

Beispiele:

- Auf vielen Unix-Systemen ist die Datei */etc/passwd* für jeden Benutzer lesbar, so dass er sich Informationen über dort eingetragene persönliche Daten verschaffen kann. Außerdem kann er mit Wörterbuchattacken (siehe G 5.18 *Systematisches Ausprobieren von Passwörtern*) versuchen, die verschlüsselten Passwörter zu erraten. Bei zu großzügiger Vergabe von Gruppenrechten, insbesondere bei den Systemgruppen wie z. B. *root, bin, adm, news* oder *daemon*, ist ein Missbrauch wie z. B. das Verändern oder Löschen fremder Dateien leicht möglich.
- Ein für die Verwaltung der Festplatten in z/OS-Systemen zuständiger Storage-Administrator konnte dank des Attributes *Operations*, das er für die Ausführung seiner Tätigkeit von der RACF-Administration erhalten hatte, Kundendateien einsehen. Er nutzte dieses Zugriffsrecht aus, um unerlaubt Kopien zu erstellen.

G 5.20 Missbrauch von Administratorrechten

Eine missbräuchliche Administration liegt vor, wenn man vorsätzlich recht- oder unrechtmäßig erworbene Super-User- (*root*-) Privilegien ausnutzt, um dem System oder dessen Benutzern zu schaden.

Beispiele:

- Da *root* auf Unix-Anlagen keinerlei Beschränkungen unterliegt, kann der Administrator unabhängig von Zugriffsrechten jede Datei lesen, verändern oder löschen. Außerdem kann er die Identität jedes Benutzers seines Systems annehmen, ohne dass dies von einem anderen Benutzer bemerkt wird, es ist ihm also z. B. möglich, unter fremden Namen Mails zu verschicken oder fremde Mails zu lesen und zu löschen.
- Es gibt verschiedene Möglichkeiten, missbräuchlich Super-User-Privilegien auszunutzen. Dazu gehören der Missbrauch von falsch administrierten Super-User-Dateien (Dateien mit Eigentümer *root* und gesetztem s-Bit) und des Befehls *su*.
- Die Gefährdung kann auch durch automatisches Mounten von austauschbaren Datenträgern entstehen: Sobald das Medium in das Laufwerk gelegt wird, wird es gemountet. Dann hat jeder Zugriff auf die dortigen Dateien. Mit sich auf dem gemounteten Laufwerk befindenden s-Bit-Programmen kann jeder Benutzer Super-User-Rechte erlangen.
- In Abhängigkeit von der Unix-Variante und der zugrunde liegenden Hardware kann bei Zugangsmöglichkeit zur Konsole der Monitor-Modus aktiviert oder in den Single-User-Modus gebootet werden. Das ermöglicht die Manipulation der Konfiguration.
- Durch Softwarefehler kann es möglich sein, dass eine Anwendung nur eine begrenzt große Menge an Daten verarbeiten kann. Werden dieser Anwendung übergroße Datenmengen oder Parameter übergeben, können Bereiche im Hauptspeicher mit fremdem Code überschrieben werden. Dadurch können Befehle mit den Rechten der Anwendung ausgeführt werden. Dies war unter anderem mit dem Befehl *eject* unter SunOS 5.5 möglich, der mit SetUID-Rechten ausgestattet ist, also bei der Ausführung Super-User-Rechte besitzt.

G 5.28 Verhinderung von Diensten

Ein solcher Angriff, auch "Denial of Service" genannt, zielt darauf ab, die Benutzer daran zu hindern, Funktionen oder Geräte zu verwenden, die ihnen normalerweise zur Verfügung stehen. Dieser Angriff steht häufig im Zusammenhang mit verteilten Ressourcen, indem ein Angreifer diese Ressourcen so stark in Anspruch nimmt, dass andere Benutzer an der Arbeit gehindert werden. Es können zum Beispiel die folgenden Ressourcen künstlich verknappt werden: Prozesse, CPU-Zeit, Plattenplatz, Inodes, Verzeichnisse.

Dies kann zum Beispiel geschehen durch:

- das Starten von beliebig vielen Programmen gleichzeitig,
- das mehrfache Starten von Programmen, die viel CPU-Zeit verbrauchen,
- das Belegen aller freien Inodes in einem Unix-System, sodass keine neuen Dateien mehr angelegt werden können,
- unkoordiniertes Belegen von Bandstationen in z/OS-Systemen, sodass Anwendungen auf freie Bandstationen warten müssen und die Online-Verarbeitung eingeschränkt ist,
- bewusste Falscheingabe von Passwörtern (auch Skript-gesteuert) mit dem Ziel der Sperrung aller Kennungen eines z/OS-Systems,
- das Versenden bestimmter konstruierter Datenpakete, die beim Empfänger aufgrund von Software-Schwachstellen zu Fehlfunktionen oder zu einer Überlastung führen können (zum Beispiel indem exzessiv kryptographische Operationen aufgerufen werden),
- die gezielte Überlastung des Netzes,
- das Kappen von Netzverbindungen
- das gezielte Generieren von XML-Nachrichten mit großen Datenmengen, rekursiven Inhalten, einer großen Anzahl an Verschachtelungen und fehlerhaften DTDs, sodass ein XML-Parser intensiv Speicherressourcen seines Systems belegt.

G 5.87 Web-Spoofing

Der Begriff Spoofing beschreibt das Vortäuschen einer falschen Identität durch einen Angreifer. Verschiedene Arten von Spoofing sind zum Beispiel ARP-, DHCP-, DNS-, IP-, MAC-, Mail- und URL-Spoofing.

Beim Web- oder URL-Spoofing fälscht ein Angreifer eine existierende Webseite, das heißt er gestaltet eine von ihm angebotene Webseite so, dass diese wie die Webseite einer bekannten Institution aussieht. Die bereits vorhandene Webseite, die nachgebildet wurde, wird dabei nicht verändert, sondern ist weiterhin in der ursprünglichen Form erreichbar. Mit Hilfe verschiedener Tricks versucht der Angreifer dann Benutzer auf die von ihm ins Netz gestellte Webseite zu locken.

Dazu könnte er beispielsweise seine Web-Adresse so wählen, dass viele Benutzer alleine durch die Adresswahl davon ausgehen, mit einer bestimmten Institution verbunden zu sein. So kann er beispielsweise eine Seite registrieren, bei der der Hostname mit dem der Original-Webseite identisch ist, aber die Top-Level-Domain ausgetauscht wurde (zum Beispiel http://www.example.net statt http://www.example.de). Er kann aber auch versuchen, eine Adresse zu verwenden, die häufige Tipp- oder Schreibfehler ("Typosquatting") enthält und so Benutzer auf die gefälschte Seite locken (zum Beispiel http://www.exampel.com).

Eine weitere Möglichkeit besteht darin, manipulierte Links zu verbreiten. Es können unterschiedliche Zeichensätze und gleich aussehende Buchstaben verwendet werden, um täuschend echt aussehende Links zu erzeugen. Beispielsweise könnten Zahlen, die auf den ersten Blick wie Buchstaben aussehen oder Buchstaben, die sich ähneln, verwendet werden. Neben dem kaum zu erkennenden Unterschied zwischen "l" (großes "i") und "l" (kleines "L") können auch ähnlich aussehende Buchstaben verwendet werden. Ein Beispiel hierfür ist die lateinische und die kyrillische Schreibweise des Buchstabens "a", der jeweils gleich aussieht, aber unterschiedlich kodiert wird.

Benutzern können auch Adressen angezeigt werden, die aber nicht mit denen identisch sind, zu denen der Link führt. Beispielsweise kann durch die Nutzung eines HTML-Links die URL der vertrauenswürdigen Seite angezeigt werden, obwohl der Link zu einer gefälschten Seite führt. Als andere Möglichkeit können Benutzername und Passwort in der URL dem Seitennamen vorangestellt werden (http://www.example.com@attacker.com). Benutzer, die diese Schreibweise nicht kennen, nehmen an, dass sie zu der Webseite, die als Benutzername/Passwort angegeben ist, geleitet werden, obwohl sich der tatsächlich verwendete Hostname wesentlich weiter hinten in der URL befindet.

Spoofing bei Web-Services

Bevor ein Web-Service-Client einen Web-Service nutzt, benötigt der Client Informationen, wie und mit welchen Parametern der gewünschte Web-Service aufzurufen ist. Diese Informationen sind in den Metadaten-Dokumenten, also zum Beispiel einer WSDL-Datei oder einer WS-Security-Policy-Datei, definiert. Gelingt es einem Angreifer, diese Informationen absichtlich zu verändern, kann er damit das Verhalten des Clients oder die eingesetzten Sicherheitsmechanismen beeinflussen. Diese Art der Veränderung an Metadaten-Dateien ist auch als *Metadata Spoofing* bekannt.

Weitere Spoofing-Angriffe auf Web-Services basieren auf den im SOAP-Protokoll vorgesehenen optionalen Routing-Informationen. So kann beim Aufruf

eines Web-Service im <ReplyTo>-Feld der SOAP-Anfrage ein vom anfragenden Client abweichendes System angegeben und dadurch eine IP-Verbindung vom Web-Server zu einem anderen System initiiert werden, um zum Beispiel einen Denial-of-Service-Angriff oder komplexere andere Angriffsszenarien durchzuführen.

- Die XY Bank verwendet die URL www.xy-bank.de für ihren Internetauftritt. Ein Angreifer richtet unter den URLs www.xybank.de oder www.xy-bank.com eine Webseite ein, die auf den ersten Blick derjenigen der XY Bank ähneln. Zusätzlich sorgt er dafür, dass diese Adressen von XY-Kunden über Suchmaschinen gefunden werden.
  Benutzer, die diese Seiten aufrufen, werden annehmen, dass sie mit dem Webserver ihrer Bank kommunizieren. Daher sind sie bereit, ihre Kontonummer und PIN oder andere Zugangscodes einzugeben.
- Die Seite whitehouse.com durchlebte eine wechselhafte Geschichte. Hier befand sich allerdings nie, wie von vielen Benutzern zunächst vermutet, der Webauftritt des amerikanischen Weißen Hauses, sondern wechselnde kommerzielle oder pornographische Inhalte.
- Die beiden URLs www.BSI.bund.de und www.BSl.bund.de sehen zumindest auf den ersten Blick identisch aus. Erst beim genauen hinsehen ist zu erkennen, dass nur der erste zum Webauftritt des Bundesamtes für Sicherheit in der Informationstechnik führt. Beim zweiten Link wurde das große "I" durch die Ziffer 1 ersetzt.
- Gelingt es einem Angreifer, einem Web-Service-Client eine modifizierte WS-Security-Policy eines Web-Service unterzuschieben, so kann er damit zum Beispiel eine Sicherheitsanforderung an verschlüsselte Kommunikation außer Kraft setzen und in der Folge die Daten des vom Client initiierten Diensteaufrufs mitlesen.

G 5.88 Missbrauch aktiver Inhalte

Aktive Inhalte sind Programmteile oder Skripte, die im Browser ausgeführt werden. Weit verbreitete Arten von aktiven Inhalten sind JavaScript, Java-Applets, ActiveX-Elemente, Flash etc. Sie dienen häufig dazu, Webseiten interaktiver zu gestalten, besondere grafische Effekte zu erzielen oder Multimedia-Inhalte einzubetten.

Andererseits können aktive Inhalte jedoch auch gezielt zu dem Zweck erstellt worden sein, vertrauliche Daten auszuspionieren, Manipulationen vorzunehmen oder den Computer mit Schadprogrammen zu infizieren. Auch Angriffe auf die Verfügbarkeit des jeweiligen Clients sind möglich. Die gängigen Browser enthalten Sicherheitsmechanismen, die die Zugriffsmöglichkeiten von aktiven Inhalten einschränken. Es werden jedoch immer wieder Schwachstellen und Möglichkeiten bekannt, diese Sicherheitsmechanismen zu unterlaufen.

Die folgenden Aspekte tragen dazu bei, dass das Ausführen aktiver Inhalte zu Sicherheitsproblemen führen kann:

- Aktive Inhalte können aus dem Netz geladen und ausgeführt werden, ohne dass die Benutzer aktiv daran beteiligt sind. Häufig ist die Ausführung für die Benutzer auch nicht erkennbar.
- Aktive Inhalte können über Standardnetzprotokolle, beispielsweise SMTP, mit Computern im Internet kommunizieren. Auf diese Weise können zum Beispiel vertrauliche Informationen an Unbefugte weitergeleitet werden.
- Für die verschiedenen Arten von aktiven Inhalten bestehen unterschiedliche Möglichkeiten, auf die Ressourcen des Betriebssystems und der Hardware zuzugreifen.

Anders als bei Java und JavaScript ist die Funktionsvielfalt von ActiveX-Controls kaum eingeschränkt. Die Controls können direkt auf dem Rechner ablaufen und Zugriff auf die Hardware und das Betriebssystem haben. Aufgrund dieser vielfältigen Zugriffsmöglichkeiten ist mit der Ausführung von ActiveX-Komponenten ein hohes Risiko verbunden.

Bei entsprechender Voreinstellung seines Browsers kann ein Benutzer dafür sorgen, dass nur digital signierte ActiveX-Controls ausgeführt werden. Eine solche gültige Signatur beweist allerdings nur, dass der Hersteller des ActiveX-Controls bei einer Zertifizierungsstelle bekannt ist und dass das von diesem Hersteller bereitgestellte Control unverändert geladen wurde. Hierdurch wird nichts über die Funktionsweise oder Unbedenklichkeit eines solchen Controls ausgesagt und auch keine Gewähr dafür übernommen.

G 5.131 SQL-Injection

Greift eine Anwendung auf die Daten einer SQL-Datenbank zu, so werden Befehle in Form von SQL-Anweisungen an die Datenbank übermittelt. Ist die Anwendung anfällig für SQL-Injection, kann ein Angreifer durch Manipulation der Eingabedaten geänderte oder zusätzliche SQL-Anweisungen injizieren, die von der Anwendung an die Datenbank weitergeleitet und dort bearbeitet werden. Auf diese Weise können wie bei einem direkten Datenbankzugriff beliebige SQL-Anweisungen ausgeführt werden und so Sicherheitsmechanismen der Anwendung beim Datenzugriff umgangen werden.

Eine SQL-Injection kann daher z. B. die folgenden Auswirkungen haben:

- unberechtigter Zugriff auf Daten,
- Erzeugen, Auslesen, Verändern oder Löschen von Daten,
- Ausführen von Betriebssystembefehlen,
- Kontrolle über die Datenbank,
- Zugriff auf weitere Server (z. B. HTTP-Get-Request oder DNS-Abfrage).

Das Einschleusen der SQL-Anweisung wird dabei durch eine unzureichende Validierung von Eingabedaten innerhalb der Anwendung ermöglicht, die in dieser Form direkt in eine dynamische Datenbankabfrage eingebaut werden (siehe auch G 4.84 *Unzureichende Validierung von Ein- und Ausgabedaten bei Webanwendungen und Web-Services*).

Die SQL-Injection ist ein spezieller Injection-Angriff (siehe G 5.174 *Injection-Angriffe*), der sich ausschließlich gegen SQL-Datenbanken richtet. So ist das grundsätzliche Vorgehen zum Einschleusen von Befehlen auch bei anderen Interpretern möglich (z. B. LDAP-Injection, XML-Injection).

G 5.165 Unberechtigter Zugriff auf oder Manipulation von Daten bei Webanwendungen und Web-Services

Wenn ein Benutzer eine Webanwendung bedient oder wenn ein Programm auf einen Web-Service zugreift, werden Daten übertragen und üblicherweise sowohl client- als auch serverseitig gespeichert (zum Beispiel in Protokolldateien, Browser- und Proxy-Cache). Wenn diese Daten bei der Übertragung und Speicherung nicht angemessen geschützt sind, können sie unbefugt durch Dritte eingesehen oder manipuliert werden.

Aufgrund der unterschiedlichen Übertragungswege und Speicherorte der Daten ergeben sich besondere Gefährdungen, die anhand nachfolgender Beispiele erläutert werden:

- Zugangs- und Formulardaten, die ein Benutzer im Web-Browser eingibt, werden im Browser-Cache zwischengespeichert. Kann ein Angreifer auf den Rechner zugreifen, dann kann er den Browser-Cache und somit die schützenswerten Daten auslesen, da der Browser-Cache üblicherweise nicht gesondert geschützt ist (zum Beispiel durch Verschlüsselung).
- Werden GET-Parameter in der URL übertragen, können diese auf dem Weg von der Webanwendung zum Client von den dazwischenliegenden IT-Systemen (zum Beispiel Proxy-Server) in deren Protokolldateien gespeichert werden. Proxy-Server protokollieren üblicherweise die aufrufende URL inklusive übertragener GET-Parameter. Personen mit Zugriff auf die Protokolle können daher die Daten in den GET-Parametern lesen. Werden von der Webanwendung schützenswerte Daten in GET-Parametern übermittelt, kann demzufolge der Schutz der Daten nicht gewährleistet werden. Darüber hinaus können vertrauliche Daten in GET-Parametern beim Versenden eines Links oder durch Einsicht der Browser-Historie offengelegt werden.
- Müssen Sitzungsdaten einer Webanwendung auf dem Client hinterlegt werden, geschieht dies häufig über eine Speicherung in Cookies. Hierbei kann es sich um schützenswerte Daten wie die Session-ID handeln. Erlangt ein Angreifer Zugriff auf den Client (zum Beispiel durch das clientseitige Ausführen von Schadcode), so ist es möglich, den Inhalt von Cookies unbefugt auszulesen oder zu verwenden und unbemerkt an den Angreifer zu versenden (siehe auch G 5.170 *Cross-Site Scripting (XSS)*).
- Wird die Verbindung zwischen dem Web-Service-Client und dem Web-Service nicht ausreichend durch Verschlüsselung oder elektronische Signaturen abgesichert, besteht die Möglichkeit, dass Angreifer vertrauliche Daten während der Übertragung einsehen oder manipulieren können.

G 5.166 Missbrauch einer Webanwendung durch automatisierte Nutzung

Bei der automatisierten Bedienung von Webanwendungen werden Funktionen der Anwendung computergesteuert genutzt, z. B. durch Skripte, die Eingaben durch Tastatur und Maus emulieren. Dadurch können Vorgänge in kurzer Zeit durchgeführt werden und Angreifer können somit auf Wiederholung basierende Angriffe gegen die Webanwendung effizient durchführen. Mithilfe eines wiederholt durchgeführten Login-Prozesses können z. B. gültige Kombinationen aus Benutzernamen und Passwort systematisch ermittelt (Brute-Force) oder Listen mit gültigen Benutzernamen erzeugt werden (Enumeration).

Darüber hinaus kann das wiederholte Aufrufen von ressourcenintensiven Funktionen (z. B. komplexe Datenbankabfragen) für Denial-of-Service-Angriffe auf Anwendungsebene missbraucht werden. Während Denial-of-Service-Angriffe auf Netzwerkebene häufig viele Verbindungsversuche erfordern, können Angriffe auf Webanwendungsebene oft effizienter durchgeführt werden.

Beispiele:

- Kann bei einer Online-Umfrage das Formular automatisiert ausgefüllt und abgeschickt werden, so kann ein Angreifer das Umfrageergebnis durch eine automatisierte Stimmabgabe per Skript leicht verfälschen.
- Die Informationen über registrierte Benutzer (z. B. Profilnamen und E-Mail-Adressen) können über eine URL der Webanwendung (z. B. http://host.tld/app/userDetails.php?UserID=###) abgerufen werden. Wird diese Funktion automatisiert aufgerufen (z. B. durch einfache Inkrementierung der numerischen UserID), kann so mit wenig Aufwand eine große Anzahl von Benutzerinformationen gesammelt werden (Enumeration). Die gesammelten Informationen können beispielsweise für den Versand von SPAM verwendet werden.
- Werden Benutzerkonten nach fünf fehlgeschlagenen Anmeldeversuchen für 10 Minuten gesperrt, um Brute-Force-Angriffe zu erschweren, und sind einem Angreifer die Benutzernamen der Webanwendung bekannt, so können automatisiert fehlgeschlagene Anmeldeversuche auf diese Benutzerkonten provoziert werden. In der Folge sind diese Benutzerkonten dauerhaft gesperrt und die Webanwendung kann von den Benutzern nicht mehr genutzt werden.

G 5.167 Fehler in der Logik von Webanwendungen und Web-Services

Damit Geschäftsprozesse von einer Webanwendung abgebildet werden können, werden in der Regel einzelne Funktionen zu einer komplexen Anwendungslogik zusammengefasst. Dabei ist es für einen Prozess entscheidend, in welcher Reihenfolge die einzelnen Funktionen oder Prozessschritte aufgerufen werden. In einer Service-orientierten Architektur beschreibt der Begriff "Orchestrierung" die Zusammenstellung einzelner Web-Services. In der Orchestrierung werden die logische Abfolge der Web-Services sowie die Bedingungen zum Aufruf und sämtliche Abhängigkeiten der einzelnen Services untereinander definiert.

Werden solche logischen Abläufe bei sicherheitsrelevanten Funktionen verwendet, wie zum Beispiel bei der Authentisierung von Benutzern, kann dieser Ablauf unvorhergesehen manipuliert (zum Beispiel durch Übergehen von Einzelschritten) und somit gesteuert werden. Einem Angreifer ist es so unter Umständen möglich, den Sicherheitsmechanismus zu umgehen.

Darüber hinaus können schadhafte Aktionen auch ausgelöst werden, wenn Funktionen der Webanwendung oder des Web-Service für nicht vorgesehene Zwecke verwendet werden können. Beispielsweise kann ein Kontaktformular einer Webanwendung zum Versand von SPAM missbraucht werden, wenn die vorgegebene Kontaktadresse des Formulars geändert werden kann.

Weitere Beispiele:

- Eine Webanwendung hat ein Eingabefeld, das auf eine Länge von 20 Zeichen begrenzt werden soll. Die Eingabedaten dieses Feldes werden von der Webanwendung zusätzlich gefiltert. Dabei ist die Filterung der Eingabedaten rechenintensiver als die Prüfung der Länge der Zeichenkette. Findet die aufwendigere Filterung vor der Längenprüfung statt, kann ein Angreifer das Feld mit einer sehr langen Zeichenkette füllen, die von der ressourcenintensiven Filterkomponente verarbeitet wird. Damit kann aufgrund der Prüfreihenfolge ein hoher Ressourcenverbrauch provoziert werden, der für Denial-of-Service-Angriffe ausgenutzt werden kann.
- In einem Online-Shop wird ein Preisnachlass gewährt, wenn ein bestimmtes Produkt (Produkt X) bestellt wird. Ein Käufer möchte allerdings nicht Produkt X kaufen, sondern Produkt Y. Indem der Käufer sowohl Produkt X als auch Produkt Y zu seinem Warenkorb hinzufügt, wird der Preisnachlass gewährt. Der Zahlungsvorgang wird allerdings durch den Benutzer abgebrochen und Produkt X aus dem Warenkorb entfernt. Somit besteht kein Anspruch mehr auf einen Preisnachlass. Trotzdem wird dieser auf das Produkt Y nach einem erneuten Wechsel in den Zahlungsprozess gewährt. Aufgrund einer fehlenden Abschlussprüfung der Kriterien für den Preisnachlass kann demzufolge ein Betrüger den Kaufpreis für Produkt Y unbefugt ändern.
- Ein Angreifer verändert bei einem Web-Service Routing-Regeln und Funktionen zum Informationsaustausch, so dass eine Durchführung der abgebildeten Geschäftsprozesse verhindert wird oder vertrauliche Nachrichten an nicht-vertrauenswürdige Systeme weitergeleitet werden. Die durch die Orchestrierung abgebildete Logik in den Geschäftsprozessen kann nicht mehr sichergestellt werden und erweist sich als fehlerhaft. Informationen können falsch oder unvollständig beim Web-Service-Client ankommen.

G 5.168 Umgehung clientseitig umgesetzter Sicherheitsfunktionen von Webanwendungen und Web-Services

Auf Webanwendungen wird gewöhnlich mit generischen Clients (zum Beispiel Web-Browsern) zugegriffen. Diese können üblicherweise durch den Benutzer konfiguriert und angepasst werden. Sie unterliegen damit nicht der Kontrolle der Webanwendung, sondern sind von einem Angreifer, der sich Zugriff verschafft hat, beliebig manipulierbar. So können clientseitige Sicherheitsfunktionen außer Kraft gesetzt werden. Sind keine zusätzlichen, serverseitigen Schutzmaßnahmen vorgesehen, kann ein Angreifer somit unbefugt auf Ressourcen der Webanwendung zugreifen.

Auch Web-Services werden zum Teil durch Anwendungen genutzt, die sich nicht in einem vom Betreiber kontrollierbaren Sicherheitskontext befinden, zum Beispiel als Anwendungen auf mobilen Endgeräten ("Apps"). Werden Web-Services für solche Nutzungsszenarien realisiert, darf auch hier nicht von der Umsetzung von Sicherheitsfunktionen durch den aufrufenden Client ausgegangen werden, da für den Web-Service nicht erkennbar ist, ob der aufrufende Client manipuliert oder gegen einen anderen Client ohne entsprechende Sicherheitsfunktionen ausgetauscht wurde.

In der Praxis tritt diese Gefährdung besonders häufig in Verbindung mit Berechtigungsprüfungen auf, die clientseitig durchgeführt, aber nach dem Aufruf des Web-Service nicht vom Server verifiziert werden. So schützt beispielsweise das Ausblenden einer Schaltfläche im Client nicht davor, die für diese Schaltfläche hinterlegte Funktion auf Serverseite aufzurufen, indem zum Beispiel der Client manipuliert wird, URLs direkt aufgerufen werden oder Replay- oder Man-in-the-Middle-Attacken bei der Kommunikation durchgeführt werden.

Beispiele:

- Die Eingabevalidierung ist ausschließlich clientseitig in der Programmiersprache JavaScript umgesetzt. Ist die JavaScript-Unterstützung auf dem Client deaktiviert, wird daher die Validierungsfunktion nicht ausgeführt und somit umgangen. Somit können beliebige Eingaben (wie Schadcode) an die Webanwendung gesendet und ungeprüft verarbeitet werden. Ein Angreifer kann dies ausnutzen, um beispielsweise unbefugt Befehle an Hintergrundsysteme der Webanwendung zu übermitteln (zum Beispiel in Form von Datenbankabfragen um eine SQL-Injection auszuführen).
- Die Webanwendung prüft ausschließlich einen clientseitig gesetzten Parameter zur Authentisierung (zum Beispiel admin=true). Ist einem Angreifer dieser Parameter bekannt, so kann er den Parameter manuell setzen und verwenden, um sich ohne Kenntnis der Zugangsdaten an der Webanwendung anzumelden.
- Eine Anwendung zeigt den Menüpunkt "Benutzerverwaltung" nur an, wenn der eingeloggte Anwender Administrationsrechte hat. Durch einen direkten Aufruf des entsprechenden Web-Services ist aber auch eine Bearbeitung der Benutzerverwaltung ohne Administrationsrechte möglich, weil der Programmierer des Web-Service sich darauf verlassen hat, dass eine Berechtigungsprüfung im Client bereits durchgeführt wurde.

G 5.169 Unzureichendes Session-Management von Webanwendungen und Web-Services

Da das von Webanwendungen und Web-Services verwendete Protokoll HTTP zustandslos ist, wird ein zusätzlicher Mechanismus benötigt, um den Benutzer über die Dauer einer Sitzung zu identifizieren. Webanwendungen verwenden hierbei typischerweise Session-IDs in Form eines Cookies. Bei Web-Services kann alternativ der Standard WS-SecureConversation verwendet werden. Hier werden Sessions als sogenannter *Security Context* repräsentiert, welcher wiederum über eine Session-ID innerhalb eines *Security Context Token* referenziert werden kann. Dieser Standard umfasst zusätzlich Mechanismen zur Transportsicherung, welche bei Webanwendungen sonst typischerweise über SSL/TSL realisiert wird.

Kann eine dritte Person aufgrund eines unzureichenden Session-Managements die Session-ID ermitteln, so kann sie die Webanwendung oder den Web-Service im Kontext dieser Sitzung verwenden. Dies hat zum Beispiel zur Folge, dass ein Angreifer mit der Webanwendung als legitimer authentisierter Benutzer interagieren kann, ohne die eigentlichen Zugangsdaten (Benutzername, Passwort) zu kennen.

Die Funktionalität der Webanwendung, beziehungsweise des Web-Service kann somit von Dritten mit den Rechten des legitimen Benutzers genutzt werden, um unbefugt auf schützenswerte Daten zuzugreifen oder Befehle auszuführen.

Die folgenden Beispiele beschreiben Szenarien, die zu einer kompromittierten Sitzung führen können.

- Bei einem Session-Fixation-Angriff lässt sich ein Angreifer zunächst eine Session-ID von der Webanwendung zuweisen und übermittelt diese dem Opfer (zum Beispiel über einen Link in einer E-Mail). Folgt das Opfer diesem Link und authentisiert sich anschließend gegenüber der Webanwendung mit der vom Angreifer übermittelten Session-ID, so kann der Angreifer die Anwendung anschließend mit der ihm bekannten Session-ID verwenden. Auf diese Weise ist es ihm möglich, im Sicherheitskontext des angegriffenen Benutzers auf die Webanwendung zuzugreifen und so Funktionen zu nutzen, die einem unauthentisierten Benutzer nicht zur Verfügung stehen.
- Im Falle eines Session-Hijacking-Angriffs (Sitzungsübernahme) ist das Opfer bereits an der Webanwendung beziehungsweise dem Web-Service mit einer gültigen Session-ID angemeldet. Wird die Session-ID nicht zufällig gewählt (zum Beispiel einfaches Inkrementieren eines Zählers bei der Vergabe von Session-IDs) kann ein Angreifer gültige Session-IDs durch gezieltes Ausprobieren erraten und die entsprechenden Sitzungen der angemeldeten Benutzer übernehmen.
- Werden Sitzungen von inaktiven Benutzern einer Webanwendung oder eines Web-Service nicht automatisch nach einem bestimmten Zeitintervall ungültig (Session Timeout), bleiben die Sitzungen von nicht ordnungsgemäß von der Anwendung abgemeldeten Benutzern (zum Beispiel durch Browser-Schließung) weiterhin gültig. Erlangt ein Angreifer Kenntnis von einer solchen gültigen, aber nicht mehr genutzten Session-ID oder Zugriff-

stoken, so kann er die Webanwendung im Sicherheitskontext des nicht abgemeldeten Benutzers weiter verwenden.

G 5.170 Cross-Site Scripting (XSS)

Cross-Site Scripting-Angriffe (XSS-Angriffe) richten sich gegen die Benutzer einer Webanwendung und deren Clients. Hierbei versucht ein Angreifer indirekt Schadcode (in der Regel Browser-seitig ausführbare Skripte, wie z. B. JavaScript) an den Client des Benutzers der Webanwendung zu senden.

Werden die Ein- und Ausgaben von einer Webanwendung nicht ausreichend validiert, so kann ein Angreifer schadhaften Code in die Webanwendung einschleusen (z. B. innerhalb eines Kommentars zu einem Artikel) und so verteilen. Wird eine infizierte Webseite von einem Benutzer aufgerufen, führt der Client (z. B. Browser) den eingefügten Schadcode aus. Aus Sicht des Benutzers stammt der schadhafte Code von der Webanwendung und wird somit als vertrauenswürdig eingestuft. Daher wird der Schadcode im Sicherheitskontext der Webanwendung interpretiert und es ist dem Angreifer möglich, Befehle im Kontext einer möglicherweise bestehenden Sitzung des betroffenen Benutzers auszuführen.

Es werden drei Klassen von XSS-Angriffen unterschieden:

- persistent (beständig)
- reflektiert (nicht-persistent)
- DOM-basiert (lokal)

Die folgenden Beispiele verdeutlichen die Unterschiede der Angriffsklassen:

- Einem Angreifer gelingt es einen Eintrag in einem Gästebuch zu hinterlassen, der JavaScript-Code enthält. Ruft ein Benutzer den entsprechenden Gästebucheintrag auf, wird das Skript übermittelt und vom Browser ausgeführt. Das Skript wird im Sicherheitskontext der Webanwendung ausgeführt und hat somit Zugriff auf die clientseitig im Cookie gespeicherte SessionID des Benutzers, wenn dieses Session-Cookie (fehlerhaft) ohne HttpOnly-Flag gesetzt wurde. Diese Information wird von dem Skript an den Angreifer weitergeleitet, der die SessionID nutzen und damit die Sitzung eines authentisierten Benutzers übernehmen kann. Da der JavaScript-Code vom Browser lediglich interpretiert und nicht angezeigt wird, kann dieser Vorgang von dem Benutzer nur schwer erkannt werden. Hierbei handelt es sich um einen persistenten XSS-Angriff, da der Schadcode in dem Gästebuch-Eintrag und somit in der Webanwendung dauerhaft gespeichert wird.
- Ein Angreifer präpariert den GET-Parameter einer URL so, dass dieser JavaScript-Code enthält. Da die Webanwendung den verwendeten Parameter ungeprüft für die Aufbereitung der Webseite verwendet, wird der eingeschleuste JavaScript-Code an den Client übermittelt und vom Browser im Sicherheitskontext der Webanwendung ausgeführt. Gelingt es dem Angreifer, einen derart präparierten Link so zu verteilen (z. B. per E-Mail) und klickt ein angemeldeter Benutzer diesen Link an, so wird das schadhafte Skript im Browser des Benutzers ausgeführt. Ein solcher XSS-Angriff wird als reflektiert oder nicht-persistent bezeichnet, da der Schadcode nicht dauerhaft gespeichert wird, sondern nach der Eingabe direkt von der Webanwendung zurückgesendet wird.
- JavaScript-Code in einer Webseite verarbeitet Parameter aus der URL (z. B. http://host.tld/param="Inhalt") und bindet sie zur Anzeige in die Webseite ein. Über die Manipulation der Parameter können somit beliebige Inhalte in die Webseite eingefügt werden. Wird die Seite mit schadhaftem JavaScript-Code im Parameter aufgerufen, so wird dieser Code in die Webseite eingebunden und vom Browser ausgeführt. Im Gegensatz zu vorherigen Angriffstypen wird der Schadcode nicht von der Webanwendung

in die Webseite eingefügt, sondern erst lokal vom Browser durch die clientseitige JavaScript-Verarbeitung der URL-Parameter. Hierbei kann der Schadcode die Document Object Model (DOM)-Umgebung manipulieren und darüber die Webseitenstruktur und Inhalte verändern.

G 5.171 Cross-Site Request Forgery (CSRF, XSRF, Session Riding)

Können schreibende Aktionen einer Webanwendung ohne weitere Überprüfung der Authentizität des HTTP-Requests (z. B. durch Tokens in versteckten Formularfeldern) genutzt werden, kann ein Angreifer dem Benutzer einen präparierten Link zur Ausführung eines Befehls übermitteln.

Der Link kann beispielsweise mithilfe von Social Engineering-Methoden (z. B. als Link in einer E-Mail) einem Benutzer mit der Aufforderung zur Ausführung übermittelt werden. Ist der Benutzer an einer Webanwendung mit einer bestehenden Sitzung angemeldet und folgt diesem präparierten Link, wird der übertragene Befehl von der Webanwendung ausgeführt. Die Webanwendung interpretiert hierbei den HTTP-Request als eine vom Benutzer bewusst durchgeführte Aktion. Dabei können sich hinter einem solchen Link privilegierte Befehle wie das Ändern der Zugangsdaten oder das Anlegen eines neuen Benutzers verbergen. Dem Benutzer bleibt unter Umständen der Vorgang verborgen und es wird lediglich eine Mitteilung zur erfolgreich durchgeführten Aktion angezeigt.

Im Gegensatz zu XSS (siehe G 5.170 *Cross-Site Scripting (XSS)*) ist das Angriffsziel nicht das Ausführen von Skriptcode, sondern von unbefugten, schreibenden Aktionen im Kontext des angemeldeten Benutzers.

Mit einer Kombination von CSRF und XSS ist es möglich, den Client über die Ausführung von Skripten unbemerkt zu steuern, sodass eine Interaktion durch den Anwender nicht mehr notwendig ist. Anweisungen im Skript können z. B. eine Weiterleitung auf einen präparierten Link automatisieren.

Beispiel:

- Während ein Anwender an der Administrationsoberfläche eines Routers angemeldet ist, surft er mit demselben Browser gleichzeitig im Internet. Durch einen präparierten Link auf einer Webseite wird eine Anfrage zur Änderung des Zugangspasswortes an den Router gesendet. Hierbei sendet der Browser automatisch das Session Cookie mit, worüber die Webanwendung die Authentizität der Anfrage verifiziert und die Änderung durchführt. Da der Benutzer mit einer gültigen Sitzung an der Administrationsoberfläche angemeldet ist, wird der Befehl ausgeführt und das Zugangspasswort auf ein ihm unbekanntes Passwort abgeändert.

G 5.172 Umgehung der Autorisierung bei Webanwendungen und Web-Services

Wenn ein Benutzer oder ein Web-Service-Client sich ordnungsgemäß an einer Webanwendung oder einem Web-Service angemeldet hat, so hat er (in Abhängigkeit von der ihm zugewiesenen Rolle) nicht zwangsläufig Zugriff auf alle Funktionen, welche die Webanwendung oder der Web-Service bereitstellen. Daher muss die Webanwendung oder der Web-Service nach erfolgreicher Authentisierung des Benutzers oder des Clients für einzelne Funktionen verifizieren, ob dieser für die Ausführung berechtigt ist (Autorisierung).

Bei Angriffen gegen die Autorisierungskomponente wird versucht, auf Funktionen oder Daten zuzugreifen, die eigentlich nur einer eingeschränkten Benutzergruppe zur Verfügung stehen. Ist die Autorisierung der Zugriffe fehlerhaft umgesetzt, kann ein Angreifer seine Berechtigungen erweitern und Zugriff auf geschützte Bereiche und Daten erhalten. Dies geschieht üblicherweise durch gezielte manipulierte Eingaben eines Angreifers.

Denkbare Angriffsziele sind zum Beispiel Konfigurationsdateien mit fest codierten Zugangsdaten für Hintergrundsysteme, geschützte Bereiche oder Funktionen der Webanwendung.

Im Folgenden werden mögliche Schwachstellen bei der Autorisierung von Zugriffen auf Web-Ressourcen aufgeführt.

Beispiele:

- Bei der Eingabe von Pfadangaben können über einen relativen Bezug (durch sogenanntes Path Traversal) nicht für den Zugriff über die Webanwendung vorgesehene Ressourcen abgerufen werden (zum Beispiel *../../../config.xml*). Hierdurch können unbefugt schützenswerte Dateien wie Konfigurationsdateien aus dem Dateisystem heruntergeladen oder auch überschrieben werden. Über relative Pfadangaben lassen sich nicht nur Dateien der Webanwendung erreichen, sondern es können unter Umständen ebenso Ressourcen des darunter liegenden IT-Systems abgerufen werden.
- Webanwendungen verwenden häufig Objekt-Referenzen zur Adressierung einer Ressource in Hintergrundsystemen (zum Beispiel http://host.tld/get.php?id=2). So können Ressourcen wie Inhalte zur Darstellung einer Webseite einem Datenbankeintrag zugeordnet werden. Werden Objekt-Referenzen von der Autorisierungskomponente nicht berücksichtigt, kann über eine Manipulation der Referenz *id* in der URL gegebenenfalls auf vertrauenswürdige Ressourcen zugegriffen werden.
- Eine manchmal genutzte Möglichkeit Informationen einer Webanwendung zu schützen besteht darin, die URL, die diese Informationen verlinkt, nur autorisierten Benutzern anzuzeigen. Unautorisierten Benutzern ist die URL nicht bekannt. Ein Angreifer kann durch systematisches Ausprobieren versuchen, die URL zu erraten und so Zugriff auf geschützte Informationen beziehungsweise Funktionen der Webanwendung zu erhalten. Dieser Angriff wird "Forced Browsing" genannt.
- Ist die Autorisierungskomponente eines Web-Service fehlerhaft konfiguriert, sodass sie zum Beispiel auch anonyme Zugriffe oder Zugriffe auf falsche Dienste erlaubt, so kann ein Unberechtigter diese Dienste aufrufen und sich so Zugang zu Daten und Funktionen verschaffen.

G 5.173 Einbindung von fremden Daten und Schadcode bei Webanwendungen und Web-Services

Werden die Ein- und Ausgabedaten einer Webanwendung oder eines Web-Service nicht ausreichend validiert, so kann ein Angreifer Inhalte, wie zum Beispiel Schadcode zur Manipulation von Server, Clients oder nachgelagerten Systemen, einbinden. Die eingebundenen Daten werden dem Benutzer im Sicherheitskontext der Webanwendung oder des Web-Service zurückgegeben. Demzufolge ist es dem Benutzer der Webanwendung beziehungsweise dem Consumer des Web-Service nicht oder nur eingeschränkt möglich, die manipulierten Anteile der Ausgabe zu erkennen. Der Angreifer kann so die Vertrauensstellung des authentisierten Benutzers gegenüber der Webanwendung oder dem Web-Service ausnutzen.

Bei Web-Services kann Schadcode auf den Web-Service selbst oder aber auf einen Consumer des Web-Service (Endanwendung oder nachgelagerter Web-Service) abzielen. Bei Webanwendungen können sowohl die Clients als auch die Server der Webanwendung durch eingebundenen Schadcode angegriffen werden. So können von einem Angreifer eingebettete Daten beispielsweise Schadcode zur Ausführung auf den Clients (zum Beispiel zum Auslesen von vertraulichen Daten) oder gefälschte Anmelde-Formulare zum Diebstahl von Zugangsdaten beinhalten. Wird der eingebundene Programmcode von der Webanwendung oder dem Web-Service ausgeführt, so kann darüber hinaus das Betriebssystem des Servers kompromittiert werden.

Beispiele:

- Über Parameter in der URL können in dynamischen Webseiten fremde Inhalte eingebunden werden, die sich nicht von den Inhalten der Webanwendung unterscheiden lassen (zum Beispiel http://host.tld/index.php?frame=http://angreifer.tld&title=modifizierter Titel). Hierbei wird der übermittelte Parameter *title* in der zurückgelieferten Webseite der Webanwendung als Titel im HTML-Dokument eingebettet. Ebenso wird der Parameter *frame* als Quelle für einen Frame auf der Webseite verwendet. Hiermit lassen sich über die Parameterwerte beliebige Inhalte und Programmcode (zum Beispiel JavaScript) in die Webseite einfügen. Derselbe Angriff ist auf Web-Services übertragbar, welche über eine REST-Schnittstelle angesprochen werden, ihre Parameter also als Teil der URL übergeben bekommen.
- Eine Weiterleitungsfunktion akzeptiert beliebige Werte als Zieladresse. In der Folge kann über einen manipulierten Parameter eine Weiterleitung auf nicht vertrauenswürdige Webseiten durch einen Angreifer veranlasst werden (zum Beispiel http://host.tld/redirect.php?target=http://angreifer.tld). Der Benutzer erwartet aufgrund der Ursprungs-Domäne der Webanwendung die Weiterleitung auf eine vertrauenswürdige Adresse. Dies kann von einem Angreifer ausgenutzt werden, um über die Weiterleitung auf eine gefälschte Anmeldeseite zur Eingabe der Zugangsdaten einen Phishing-Angriff zu realisieren.
- In Webanwendungen können fremde Inhalte von Partnern (zum Beispiel Werbeanzeigen in einem iFrame) eingebunden werden. Die Kontrolle über diese Inhalte liegt üblicherweise beim Partner und nicht beim Betreiber der Webanwendung. Werden Schadsoftware oder unerwünschte Inhalte über den Partner eingebunden, so kann dies den Ruf des Webanwendungs-Be-

treibers schädigen, da die Inhalte dem Benutzer im Kontext der Webanwendung dargestellt werden. Darüber hinaus können die Clients der Besucher von der Schadsoftware infiziert und somit kompromittiert werden.

- Über eine Upload-Funktion der Webanwendung lassen sich beliebige Dateien in der Verzeichnisstruktur auf dem Server speichern. Dadurch können gegebenenfalls schadhafte Skripte zur Ausführung auf der Webanwendung gespeichert oder bestehende Dateien (zum Beispiel Konfigurationsdateien) überschrieben werden. Der Upload von großen Mengen an Daten kann auch zu einer Verhinderung des Dienstes führen.
- In die XML-formatierten Parameter für einen Web-Service werden externe Referenzen eingebettet, zum Beispiel <!DOCTYPE sample PUBLIC "foo" "">. Falls die serverseitige Anwendung bei der Interpretation der Ergebnisse der externen Referenz folgt, kann der Angreifer damit ausgehende IP-Verbindungen vom Server aus initiieren und so entweder den Betrieb stören (Denial-of-Service) oder Informationen über interne Netzstrukturen gewinnen.

G 5.174 Injection-Angriffe

Bei einem Injection-Angriff versucht ein Angreifer, Befehle in eine Webanwendung oder einen Web-Service zu injizieren und auszuführen. Der Angriff richtet sich dabei in der Regel gegen serverseitig verwendete Interpreter oder einen Parser.

Werden beispielsweise eingehende Daten unzureichend validiert, so können Eingaben (zum Beispiel Formulardaten, Cookies, SOAP-Nachrichten oder HTTP-Header) so gewählt werden, dass sie von der Webanwendung und den verwendeten Interpretern beziehungsweise Parsern (zum Beispiel SQL-Datenbank, XML-Prozessoren, LDAP-Verzeichnisdienst) als Befehl interpretiert werden. Auf diese Weise können unbefugt Befehle zum Auslesen oder Manipulieren von Daten übermittelt werden.

Können mittels Injection beliebige System-Kommandos ausgeführt werden, so kann ein Angreifer die Webanwendung oder den Web-Service als Ersatz für eine System-Shell nutzen. Die abgesetzten System-Kommandos werden dabei üblicherweise im Sicherheitskontext und somit mit den Privilegien der Webanwendung/des Web-Service oder des verwendeten Interpreters beziehungsweise Parsers ausgeführt.

Injection-Angriffe werden anhand der angegriffenen Interpreter/Parser in Angriffstypen klassifiziert. Die folgenden Beispiele verdeutlichen diese Klassifizierung:

- SQL-Injection (siehe auch G 5.131 *SQL-Injection*)
- LDAP-Injection
- Mail-Command-Injection
- OS-Command-Injection
- SSI-Injection
- XPath-Injection
- Code-Injection

G 5.175 Clickjacking

Bei einem Clickjacking-Angriff werden Teile einer Webseite bei der Darstellung überdeckt, sodass für den Benutzer nicht sichtbare, transparente Ebenen die angezeigten Inhalte der Webseite überlagern.

In diesen transparenten Ebenen können beliebige Inhalte oder Bedienelemente eingebunden werden, ohne dass sie für den Benutzer sichtbar sind. Klickt der Benutzer auf die vermeintlichen Inhalte der Webseite, so wird der Klick nicht an die sichtbare Ebene, sondern an die überlagerten Ebenen gesendet und somit entführt (engl. *Hijacking*). Die Angriffsbezeichnung Clickjacking ergibt sich aus der Wortkombination *Click* für Mausklick und *Jacking* von Hijacking.

Neben Mausklicks können darüber hinaus auch Tastatureingaben mittels transparent eingeblendeter Textfelder auf fremde Server umgeleitet werden (z. B. Platzierung über Passwortfelder).

Beispiele:

- Ein Angreifer nimmt an einem Programm für Werbeanzeigen teil, bei dem die Höhe der Provision anhand der Klicks durch die Besucher ermittelt wird (Klickvergütung oder Pay-per-Click). Dabei überlagert er einen Teil einer Webanwendung mit einem unsichtbaren Link zur Werbeanzeige, sodass der Benutzer unbemerkt auf die Werbeanzeige klickt. Dadurch erhöht sich die Anzahl der Klicks und damit die auszuzahlende Provision.
- Ein Angreifer platziert auf einer Webseite einen unsichtbaren "Gefällt mir"-Button für seine eigene Facebook-Seite, der immer dem Mauszeiger folgt. Für den Benutzer ist das nicht erkennbar. Klickt er irgendwo übermittelt auf der Seite, wird die "Gefällt- mir"-Funktion von Facebook ausgeführt er übermittelt seine Facebook-Daten an den Angreifer, der diese dann weiter ausnutzen kann.

M 2.1 Festlegung von Verantwortlichkeiten und Regelungen

Verantwortlich für Initiierung: Behörden-/Unternehmensleitung
Verantwortlich für Umsetzung: Leiter IT, Leiter Organisation

Für alle wesentlichen Aufgaben und Geschäftsprozesse in einer Institution sollten die Verantwortlichkeiten nachvollziehbar geregelt sein. Die Aufgaben sollten dabei so zugeschnitten sein, dass es keine Überschneidungen zwischen ähnlichen Aufgaben gibt, aber auch keine Zuständigkeitslücken. Dies sollte für alle Bereiche eine Selbstverständlichkeit sein, für alle sicherheitsrelevanten Aufgaben ist es aber unabdingbar.

Die sicherheitsrelevanten Aufgaben aller internen und externen Mitarbeiter und Dienstleister müssen nachvollziehbar festgelegt sein. Sie müssen mit den Sicherheitszielen der Institution abgestimmt sein. Zu den Bereichen, die geregelt werden sollten, gehören beispielsweise:

- explizite Zuweisung der Verantwortlichkeiten und Befugnisse an Rollen bzw. Organisationseinheiten bei allen sicherheitsrelevanten Aufgaben (Dabei ist sicherzustellen, dass alle Rollen konkreten Personen zugeordnet sind),
- geeigneter Umgang mit geschäftskritischen Informationen, so dass deren Vertraulichkeit, Integrität und Verfügbarkeit angemessen geschützt sind,
- Vertraulichkeitsvereinbarungen,
- Einbeziehung des Sicherheitsbeauftragten bei Aufträgen und Projekten, die geschäftskritische Informationen betreffen,
- Unterrichtungen über den geeigneten Umgang mit geschäftskritischen Informationen, beispielsweise im Kontakt mit Kunden oder auf Reisen,
- Festlegung von Verhaltensregeln und Informationspflichten bei sicherheitsrelevanten Aktionen und bei Sicherheitsvorfällen,
- Klassifikation von Informationen entsprechend ihres Schutzbedarfs.

Die Regelungen für Informationssicherheit sollten mit denen für Datenschutz und Geheimschutz in geeigneter Weise zusammengeführt werden, damit sie von den Mitarbeitern leichter adaptiert und besser wahrgenommen werden können. Wichtig ist auch, dass alle Regelungen zusammengenommen widerspruchsfrei sind.

Übergreifende Regelungen zur Informationssicherheit müssen als ein Aspekt der Informationsverarbeitung verbindlich festgelegt werden.

Es empfiehlt sich, Regelungen unter anderem über die Themen

- Datensicherung,
- Datenarchivierung,
- Datenträgertransport,
- Datenübertragung,
- Datenträgervernichtung,
- Dokumentation von IT-Verfahren, Software, IT-Konfiguration,
- Zutritts-, Zugangs- und Zugriffsberechtigungen,
- Wartungs- und Reparaturarbeiten,
- Datenschutz,
- Schutz gegen Schadsoftware,
- Revision,
- Notfallvorsorge und

- Vorgehensweise bei der Verletzung von Sicherheitsrichtlinien

zu treffen. Hinweise dazu finden sich in den Maßnahmenbeschreibungen der jeweils relevanten IT-Grundschutz-Bausteine.

Die in Kraft gesetzten Regelungen sind den betroffenen Mitarbeitern in geeigneter Weise bekannt zu geben (siehe M 3.2 *Verpflichtung der Mitarbeiter auf Einhaltung einschlägiger Gesetze, Vorschriften und Regelungen*). Es empfiehlt sich, die Bekanntgabe zu dokumentieren. Darüber hinaus sind sämtliche Regelungen in der aktuellen Form an einer Stelle vorzuhalten und bei berechtigtem Interesse zugänglich zu machen.

Die getroffenen Regelungen sind regelmäßig zu aktualisieren, um Missverständnisse, ungeklärte Zuständigkeiten und Widersprüche zu vermeiden und gegebenenfalls aufzulösen. Alle Regelungen sollten deshalb auch ein Erstellungsdatum oder eine Versionsnummer enthalten.

Prüffragen:

- Sind die Verantwortlichkeiten und Befugnisse bei allen sicherheitsrelevanten Aufgaben klar geregelt?
- Werden die Regelungen regelmäßig überarbeitet und auf einem aktuellen Stand gehalten?
- Wurden die Regelungen allen Mitarbeitern bekannt gegeben?

M 2.8 Vergabe von Zugriffsrechten

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator, Fachverantwortliche

Über Zugriffsrechte wird geregelt, welche Person im Rahmen ihrer Funktion bevollmächtigt wird, IT-Anwendungen oder Daten zu nutzen. Die Zugriffsrechte (z. B. Lesen, Schreiben, Ausführen) auf IT-Anwendungen, Teilanwendungen oder Daten sind von der Funktion abhängig, die die Person wahrnimmt, z. B. Anwenderbetreuung, Arbeitsvorbereitung, Systemprogrammierung, Anwendungsentwicklung, Systemadministration, Revision, Datenerfassung, Sachbearbeitung. Dabei sollten immer nur so viele Zugriffsrechte vergeben werden, wie es für die Aufgabenwahrnehmung notwendig ist ("Need-to-know-Prinzip"). Umgesetzt werden müssen die Zugriffsrechte durch die Rechteverwaltung des IT-Systems.

Eine Vielzahl von IT-Systemen lassen es zu, dass verschiedene Rechte als Gruppenrechte bzw. als Rechteprofil definiert werden (z. B. Gruppe Datenerfassung). Diese Definition entspricht der technischen Umsetzung der Rechte, die einer Funktion zugeordnet werden. Für die Administration der Rechte eines IT-Systems ist es vorteilhaft, solche Gruppen oder Profile zu erstellen, da damit die Rechtezuteilung und deren Aktualisierung erheblich vereinfacht werden kann.

Die Festlegung und Veränderung von Zugriffsrechten ist vom jeweils Verantwortlichen zu veranlassen und zu dokumentieren. Aus der Dokumentation muss hervorgehen:

- welche Funktion unter Beachtung der Funktionstrennung (siehe M 2.5 *Aufgabenverteilung und Funktionstrennung*) mit welchen Zugriffsrechten ausgestattet wird,
- welche Gruppen bzw. Profile eingerichtet werden,
- welche Person welche Funktion wahrnimmt,
- welche Zugriffsrechte eine Person im Rahmen welcher Rolle erhält (hierbei sollten auch die Zugriffsrechte von Vertretern erfasst werden) und
- welche Konflikte bei der Vergabe von Zugriffsrechten aufgetreten sind. Diese Konflikte können z. B. daraus resultieren, dass eine Person unvereinbare Funktionen wahrnimmt oder daraus, dass abhängig vom IT-System die Trennung bestimmter Zugriffsrechte nicht vorgenommen werden kann.
- welche Personen in einem Notfall welche Zugriffsrechte erhalten, z. B. da sie zum Krisenstab gehören.

Die Vorgehensweise bei der Funktionstrennung und der Rechtevergabe wird am nachfolgenden Beispiel erläutert.

Die betrachtete Anwendung ist ein Reisekosten-Abrechnungssystem. Die relevanten Räume sind in nachfolgender Graphik erläutert. Das IT-System besteht aus einem LAN, an dem neben einem Server und der Bedienkonsole drei PCs als Arbeitsplatzrechner angeschlossen sind.

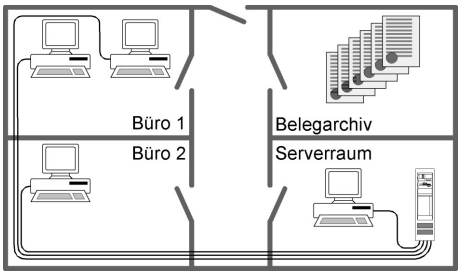


Abbildung: Aufgabenverteilung und Funktionstrennung

Schritt 1: Aufgabenverteilung und Funktionstrennung

Folgende Funktionen sind für das betrachtete Reisekosten-Abrechnungssystem notwendig:

1. LAN-Administration
2. Revision
3. Datenerfassung
4. Sachbearbeitung mit Feststellung der rechnerischen Richtigkeit
5. Sachbearbeitung mit Feststellung der sachlichen Richtigkeit
6. Sachbearbeitung mit Anordnungsbefugnis

Folgende Funktionen sind aufgrund der Sachzwänge nicht miteinander vereinbar:

- Funktion 1 und Funktion 2 (die Administration darf sich nicht selbst kontrollieren)
- Funktion 2 und Funktion 6 (der Anordnungsbefugte darf sich nicht selbst kontrollieren)
- die Kombination der Funktionen 4 oder 5 mit 6 (das Vier-Augen-Prinzip wäre verletzt für Zahlungsanweisungen)

Diese Funktionen werden durch folgende Personen wahrgenommen:

		Hr. Mayer	**Fr. Schmidt**	**Hr. Müller**	**Fr. Fleiß**
1.	LAN-Administration	X			
2.	Revision		X		
3.	Datenerfassung			X	
4.	Sachbearbeitung rechn.			X	
5.	Sachbearbeitung sachl.			X	
6.	Anordnungsbefugnis				X

Schritt 2: Vergabe von Zutrittsrechten

Nachfolgend wird der Schutzbedarf der einzelnen Räume begründet und in der Tabelle die Vergabe der Zutrittsrechte dokumentiert:

- Serverraum:
  Der unbefugte Zutritt zum Server muss verhindert werden, weil die Verfügbarkeit, Integrität und Vertraulichkeit der gesamten Anwendung von dieser zentralen Komponente abhängig ist.
- Belegarchiv:
  Für die Rechnungslegung müssen die Reisekostenabrechnungen längerfristig aufbewahrt werden. Es ist sicherzustellen, dass die Belege vollständig und unverändert aufbewahrt werden.
- Büro 1:
  In diesem Büro werden die notwendigen Daten erfasst sowie die rechnerische und sachliche Richtigkeit festgestellt. Für die Gewährleistung der Korrektheit dieser Vorgänge muss verhindert werden, dass Unbefugte Zutritt zu den Arbeitsplatzrechnern erhalten.
- Büro 2:
  Hier wird die Auszahlung der Reisekosten am APC angeordnet. Dieser Vorgang darf nur von einer befugten Person vorgenommen werden. Unbefugten ist der Zutritt zu verwehren.

		Serverraum	**Belegarchiv**	**Büro 1**	**Büro 2**
1.	LAN-Administration	X			
2.	Revision	X	X	X	X
3.	Datenerfassung			X	
4.	Sachbearbeitung rechn.		X	X	
5.	Sachbearbeitung sachl.		X	X	
6.	Anordnungsbefugnis		X	X	X

Schritt 3: Vergabe von Zugangsberechtigungen

Aufgrund der Funktionen ergeben sich folgende Zugangsberechtigungen:

		Betriebssystem Server	**Anwendung Protokollauswertung**	**Anwendung Datenerfassung**	**Anwendung Belegbearbeitung**
1.	LAN-Administration	X			
2.	Revision	X	X		X

		Betriebssystem Server	Anwendung Protokollauswertung	Anwendung Datenerfassung	Anwendung Belegbearbeitung
3.	Datenerfassung			X	
4.	Sachbearbeitung rechn.				X
5.	Sachbearbeitung sachl.				X
6.	Anordnungsbefugnis				X

Schritt 4: Vergabe von Zugriffsrechten

Im folgenden werden die Zugriffsrechte, die eine Funktion zur Ausübung benötigt, dargestellt. Es bezeichnen:

A = Recht zur Ausführung der Anwendung/Software

L = Leserecht auf Daten

S = Schreibrecht, d.h. Erzeugen von Daten

M = Recht zum Modifizieren von Daten

Ö = Recht zum Löschen von Daten

U = Recht zum Unterschreiben von Zahlungsanweisungen

		Betriebssystem Server	Protokollauswertung	Anwendung Datenerfassung	Anwendung Belegbearbeitung
1.	LAN-Administration	A,L,S,M,Ö			
2.	Revision	A,L	A,L,Ö		A,L
3.	Datenerfassung			A,S	
4.	Sachbearbeitung rechn.				A,L,M
5.	Sachbearbeitung sachl.				A,L,M
6.	Anordnungsbefugnis				A,L,U

Eine solche Dokumentation erleichtert die Rechteverteilung. Angenommen, dass Frau Schmidt den Arbeitgeber wechseln würde und ihre Stelle neu besetzt werden müsste, so lässt sich anhand der obigen Tabellen einfach feststellen, welche der ehemaligen Rechte Frau Schmidts zu löschen und für die neue Kraft einzurichten sind. Wenn die neue Kraft zusätzlich vertretungsweise die Funktion Sachbearbeitung mit Anordnungsbefugnis übernehmen soll, so wird anhand der durchzuführenden Rechteverteilung der Konflikt offenbar, dass die neue Kraft im Vertretungsfall Manipulationen unbemerkt durchführen könnte.

Prüffragen:

- Liegt eine aktuelle Dokumentation der vergebenen Zugriffsrechte vor?
- Werden nur die Zugriffsrechte vergeben, die für die jeweiligen Aufgaben erforderlich sind?
- Werden beantragte Zugriffsrechte oder Änderungen erteilter Zugriffsrechte von den Verantwortlichen bestätigt und geprüft?
- Existiert ein geregeltes Verfahren für den Entzug von Zugriffsrechten?

M 2.11 Regelung des Passwortgebrauchs

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Benutzer, IT-Sicherheitsbeauftragter

Werden in einem IT-System oder einer Anwendung Passwörter zur Authentisierung verwendet, so ist die Sicherheit der Zugangs- und Zugriffsrechteverwaltung des Systems entscheidend davon abhängig, dass die Passwörter korrekt gebraucht werden. Dafür ist es empfehlenswert, eine Regelung zum Passwortgebrauch einzuführen und die IT-Benutzer diesbezüglich zu unterweisen.

Vorgaben für die Passwortgestaltung müssen immer einen praktikablen Kompromiss zwischen folgenden Sicherheitszielen darstellen:

- Die Zeichenzusammensetzung des Passwortes muss so komplex sein, dass es nicht leicht zu erraten ist.
- Die Anzahl der möglichen Passwörter im vorgegebenen Schema muss so groß sein, dass es nicht in kurzer Zeit durch einfaches Ausprobieren ermittelt werden kann.
- Das Passwort darf nicht zu kompliziert sein, damit der Besitzer mit vertretbarem Aufwand in der Lage ist, es auswendig zu lernen.

Folgende Regeln zum Passwortgebrauch sollten deshalb beachtet werden:

- Das Passwort darf nicht leicht zu erraten sein. Namen, Kfz-Kennzeichen, Geburtsdatum usw. dürfen deshalb nicht als Passwörter gewählt werden.
- Ein Passwort sollte aus Großbuchstaben, Kleinbuchstaben, Sonderzeichen und Zahlen bestehen. Es sollten mindestens zwei dieser Anforderungen umgesetzt sein.
- Wenn für das Passwort alphanumerische Zeichen gewählt werden können, sollte es mindestens 8 Zeichen lang sein.
- Wenn für das Passwort nur Ziffern zur Verfügung stehen, sollte es mindestens 6 Zeichen lang sein und das Authentisierungssystem sollte den Zugang nach wenigen Fehlversuchen sperren (für eine bestimmte Zeitspanne oder dauerhaft).
- Es muss getestet werden, wie viele Stellen des Passwortes vom Rechner wirklich überprüft werden.
- Voreingestellte Passwörter (bzw. des Herstellers bei Auslieferung von Systemen) müssen durch individuelle Passwörter ersetzt werden.
- Passwörter dürfen nicht auf programmierbaren Funktionstasten gespeichert werden.
- Passwörter müssen geheim gehalten werden und sollten nur dem Benutzer persönlich bekannt sein.
- Das Passwort sollte allenfalls für die Hinterlegung schriftlich fixiert werden, wobei es in diesem Fall in einem verschlossenen Umschlag sicher aufbewahrt werden muss. Wird es darüber hinaus aufgeschrieben, ist das Passwort zumindest so sicher wie eine Scheckkarte oder ein Geldschein aufzubewahren (siehe M 2.22 *Hinterlegen des Passwortes*).
- Das Passwort muss regelmäßig gewechselt werden, z. B. alle 90 Tage.
- Ein Passwortwechsel ist durchzuführen, wenn das Passwort unautorisierten Personen bekannt geworden ist oder der Verdacht besteht.
- Alte Passwörter sollten nach einem Passwortwechsel nicht mehr gebraucht werden.
- Die Eingabe des Passwortes sollte unbeobachtet stattfinden.

Falls IT-technisch möglich, sollten folgende Randbedingungen eingehalten werden:

- Die Wahl von Trivialpasswörtern (z. B. "BBBBBBBB", "123456", Namen, Geburtsdaten) sollte verhindert werden.
- Jeder Benutzer muss sein eigenes Passwort jederzeit ändern können.
- Für die Erstanmeldung neuer Benutzer sollten Einmalpasswörter vergeben werden, also Passwörter, die nach einmaligem Gebrauch gewechselt werden müssen. In Netzen, in denen Passwörter unverschlüsselt übertragen werden, empfiehlt sich die dauerhafte Verwendung von Einmalpasswörtern (siehe M 5.34 *Einsatz von Einmalpasswörtern*).
- Erfolglose Anmeldeversuche sollten mit einer kurzen Fehlermeldung ohne Angabe von näheren Einzelheiten abgelehnt werden. Insbesondere darf bei erfolglosen Anmeldeversuchen nicht erkennbar sein, ob der eingegebene Benutzername oder das eingegebene Passwort (oder beides) falsch ist. Nach fünf aufeinander folgenden fehlerhaften Passworteingaben für dieselbe Kennung sollte das Authentisierungssystem den Zugang hierfür sperren (für eine bestimmte Zeitspanne oder dauerhaft). Die Sperrung einer Kennung darf bei nachfolgenden erfolglosen Anmeldeversuchen ebenfalls nicht erkennbar sein, sondern sollte dem jeweiligen Benutzer auf separatem Weg mitgeteilt werden.
- Bei der Authentisierung in vernetzten Systemen sollten Passwörter selbst im Intranet nicht unverschlüsselt übertragen werden. Erfolgt die Authentisierung über ein ungesichertes Netz hinweg, so dürfen Passwörter keinesfalls unverschlüsselt übertragen werden.
- Bei der Eingabe sollte das Passwort nicht auf dem Bildschirm angezeigt werden.
- Die Passwörter müssen im System zugriffssicher gespeichert werden, z. B. mittels Einweg-Verschlüsselung (Hashfunktionen).
- Der Passwortwechsel sollte vom System regelmäßig initiiert werden.
- Die Wiederholung alter Passwörter beim Passwortwechsel sollte vom IT-System verhindert werden (Passworthistorie).

Prüffragen:

- Gibt es eine verbindliche Regelung für den Passwortgebrauch?
- Sind die Benutzer angewiesen dem Schutzbedarf angemessene Passwörter mit ausreichender Komplexität zu verwenden?
- Sind die Benutzer angewiesen, Ihr Passwort geheim zu halten?
- Wird getestet, wieviele Stellen des Passwortes tatsächlich vom IT-System überprüft werden?
- Werden Passwörter in regelmäßigen Abständen gewechselt?
- Werden Passwörter sofort gewechselt, sobald sie unautorisierten Personen bekannt geworden sind oder der Verdacht darauf besteht?
- Bei erfolglosen Anmeldeversuchen: Wird nicht bekannt gegeben, ob Benutzername und/oder Passwort falsch waren?

M 2.31 Dokumentation der zugelassenen Benutzer und Rechteprofile

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator

Es muss eine Dokumentation der am IT-System zugelassenen Benutzer, angelegten Benutzergruppen und Rechteprofile erfolgen. Dabei gibt es verschiedene Dokumentationsmöglichkeiten wie beispielsweise über

- vorgegebene Administrationsdateien des Systems,
- individuelle Dateien, die vom zuständigen Administrator verwaltet werden,
- in Papierform.

Es sollte eine geeignete Form ausgewählt werden, möglichst einheitlich für die gesamte Institution.

Dokumentiert werden sollten insbesondere folgende Angaben zur Rechtevergabe an Benutzer und Benutzergruppen:

Zugelassene Benutzer:

- zugeordnetes Rechteprofil (gegebenenfalls Abweichungen vom verwendeten Standard-Rechteprofil)
- Begründung für die Wahl des Rechteprofils (und gegebenenfalls der Abweichungen)
- Zuordung des Benutzers zu einer Organisationeinheit, Raum- und Telefonnummer
- Zeitpunkt und Grund der Einrichtung
- Befristung der Einrichtung

Zugelassene Gruppen:

- zugehörige Benutzer
- Zeitpunkt und Grund der Einrichtung
- Befristung der Einrichtung

Die Dokumentation der zugelassenen Benutzer und Rechteprofile sollte regelmäßig (mindestens alle 6 Monate) daraufhin überprüft werden, ob sie den tatsächlichen Stand der Rechtevergabe widerspiegelt und ob die Rechtevergabe noch den Sicherheitsanforderungen und den aktuellen Aufgaben der Benutzer entspricht. Die vollständige Dokumentation ist Voraussetzung für Kontrollen der vergebenen Benutzerrechte.

Die Dokumentation muss so gespeichert beziehungsweise aufbewahrt werden, dass sie vor unbefugtem Zugriff geschützt ist und so, dass auch bei einem größeren Sicherheitsvorfall oder IT-Ausfall darauf zugegriffen werden kann. Falls die Dokumentation in elektronischer Form erfolgt, muss sie in das Datensicherungsverfahren einbezogen werden.

Prüffragen:

- Sind die zugelassenen Benutzer, angelegten Benutzergruppen und Rechteprofile dokumentiert?
- Wird die Dokumentation der zugelassenen Benutzer, angelegten Benutzergruppen und Rechteprofile regelmäßig auf Aktualität überprüft?
- Ist die Dokumentation der zugelassenen Benutzer, Benutzergruppen und Rechteprofile vor unbefugtem Zugriff geschützt?

- Wird die Dokumentation der zugelassenen Benutzer, Benutzergruppen und Rechteprofile - sofern sie elektronisch erfolgt - in das Datensicherungsverfahren einbezogen?

M 2.34 Dokumentation der Veränderungen an einem bestehenden System

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator

Um einen reibungslosen Betriebsablauf zu gewährleisten, muss der Administrator einen Überblick über das System haben bzw. sich verschaffen können. Dieses muss auch für seinen Vertreter möglich sein, falls der Administrator unvorhergesehen ausfällt. Der Überblick ist auch Voraussetzung, um Prüfungen des Systems (z. B. auf problematische Einstellungen, Konsistenz bei Änderungen) durchführen zu können.

Daher sollten die Veränderungen, die Administratoren am System vornehmen, dokumentiert werden, nach Möglichkeit automatisiert. Dieses gilt insbesondere für Änderungen an Systemverzeichnissen und -dateien.

Bei Installation neuer Betriebssysteme oder bei Updates sind die vorgenommenen Änderungen besonders sorgfältig zu dokumentieren. Möglicherweise kann durch die Aktivierung neuer oder durch die Änderung bestehender Systemparameter das Verhalten des IT-Systems (insbesondere auch Sicherheitsfunktionen) maßgeblich verändert werden.

Unter Unix müssen ausführbare Dateien, auf die auch andere Benutzer als der Eigentümer Zugriff haben oder deren Eigentümer *root* ist, vom Systemadministrator freigegeben und dokumentiert werden (siehe auch M 2.9 *Nutzungsverbot nicht freigegebener Hard- und Software*). Insbesondere müssen Listen mit den freigegebenen Versionen dieser Dateien geführt werden, die außerdem mindestens das Erstellungsdatum, die Größe jeder Datei und Angaben über evtl. gesetzte s-Bits enthalten. Sie sind Voraussetzung für den regelmäßigen Sicherheitscheck und für Überprüfungen nach einem Verlust der Integrität.

Prüffragen:

- Werden Systemänderungen ausreichend und für eine fachkundige Person nachvollziehbar dokumentiert?

M 2.35 Informationsbeschaffung über Sicherheitslücken des Systems

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator, IT-Sicherheitsbeauftragter

Gegen bekannt gewordene und durch Veröffentlichungen zugänglich gemachte Sicherheitslücken müssen die erforderlichen organisatorischen und administrativen Maßnahmen ergriffen werden. Sicherheitsrelevante Updates oder Patches für die eingesetzte Hard- und Software müssen gegebenenfalls installiert werden (siehe auch M 2.273 *Zeitnahes Einspielen sicherheitsrelevanter Patches und Updates*). Sind keine entsprechenden Updates oder Patches verfügbar, so muss eventuell zusätzliche Sicherheitshardware bzw. Sicherheitssoftware eingesetzt werden.

Es ist daher sehr wichtig, dass sich die Systemadministratoren regelmäßig über neu bekannt gewordene Schwachstellen informieren. Informationsquellen zu diesem Thema sind beispielsweise:

- Das Bundesamt für Sicherheit in der Informationstechnik (BSI) (siehe *http://www.bsi.bund.de/*)
- Hersteller bzw. Distributoren von Programmen und Betriebssystemen. Diese informieren oft registrierte Kunden über bekannt gewordene Sicherheitslücken ihrer Systeme und stellen korrigierte Varianten des Systems oder Patches zur Behebung der Sicherheitslücken zur Verfügung.
- Computer Emergency Response Teams (CERTs).
  Dies sind Computer-Notfallteams, die als zentrale Anlaufstelle für präventive und reaktive Maßnahmen in bezug auf sicherheitsrelevante Vorfälle in Computersystemen dienen. CERTs informieren in sogenannten *Advisories* über aktuelle Schwachstellen in Hard- und Softwareprodukten und geben Empfehlungen zu deren Behebung. Verschiedene Organisationen oder Verbände unterhalten eigene CERTs.
  Das ursprüngliche CERT der Carnegie Mellon Universität diente als Vorbild für viele weitere derartige Teams und ist heute eine Art "Dach-CERT": Computer Emergency Response Team / Coordination Center (CERT/CC), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890,
  Telefon: +1-412-268-7090 (24-Stunden-Hotline), E-Mail: *cert@cert.org*, WWW: *http://www.cert.org*
  Die CERT-Mitteilungen werden in Newsgruppen (*comp.security.announce* und *info.nsfnet.cert*) und über Mailinglisten (Aufnahme durch E-Mail an: *cert-advisory-request@cert.org*) veröffentlicht.
  In Deutschland existieren unter anderem folgende CERTs:
- CERT-Bund, Bundesamt für Sicherheit in der Informationstechnik, Postfach 20 03 63, D-53133 Bonn, Telefon: 0228 99-9582-222, Fax: 022899-9582-5427, E-Mail: *certbund@bsi.bund.de*, WWW: *https://www.bsi.bund.de/certbund/*
- DFN-CERT, Zentrum für sichere Netzdienste GmbH, Heidenkampsweg 41, D-20097 Hamburg, Telefon: 040-808077-555, Fax: -556, E-Mail: *info@dfn-cert.de*, WWW: *http://www.dfn-cert.de*. Das DFN-CERT bietet verschiedene Mailinglisten an, siehe *http://www.dfn-cert.de/info-serv/dml.html*.
- An verschiedenen Hochschulen existieren CERTs, die auch Informationen öffentlich zur Verfügung stellen. Ein Beispiel ist das RUS-CERT der Universität Stuttgart (siehe *http://cert.uni-stuttgart.de*).

- Hersteller- und systemspezifische sowie sicherheitsspezifische Newsgruppen oder Mailinglisten. In solchen Foren werden Hinweise auf existierende oder vermutete Sicherheitslücken oder Fehler in diversen Betriebssystemen und sonstigen Softwareprodukten diskutiert. Besonders aktuell sind meist die englischsprachigen Mailinglisten wie *Bugtraq*, von denen es an vielen Stellen öffentlich zugängliche Archive gibt, beispielsweise unter *http://www.securityfocus.com*.
- Manche IT-Fachzeitschriften veröffentlichen ebenfalls regelmäßig Beiträge mit einer Übersicht über neue Sicherheitslücken in verschiedenen Produkten.

Idealerweise sollten sich die Administratoren und der IT-Sicherheitsbeauftragte bei mindestens zwei verschiedenen Stellen über Sicherheitslücken informieren. Dabei ist es empfehlenswert, neben den Informationen des Herstellers auch eine "unabhängige" Informationsquelle zu benutzen.

Die Administratoren sollten jedoch in jedem Fall auch produktspezifische Informationsquellen des Herstellers nutzen, um beispielsweise darüber Bescheid zu wissen, ob für ein bestimmtes Produkt beim Bekanntwerden von Sicherheitslücken überhaupt Patches oder Updates bereitgestellt werden. Bei Produkten, für die der Hersteller keine Sicherheitspatches mehr zur Verfügung stellt, muss rechtzeitig geprüft werden, ob ein Einsatz unter diesen Umständen noch zu verantworten ist und durch welche zusätzlichen Maßnahmen ein Schutz der betroffenen Systeme trotzdem gewährleistet werden kann.

Prüffragen:

- Informieren sich die Administratoren regelmäßig bei verschiedenen Quellen über neu bekannt gewordene Schwachstellen?
- Werden sicherheitsrelevante Updates zeitnah eingespielt?
- Bei fehlenden Updates für bekannte Schwachstellen: Werden andere technische oder organisatorische Maßnahmen ergriffen?

M 2.62 Software-Abnahme- und Freigabe-Verfahren

Verantwortlich für Initiierung: Leiter IT
Verantwortlich für Umsetzung: Leiter IT

Der Einsatz von IT zur Aufgabenbewältigung setzt voraus, dass die maschinelle Datenverarbeitung soweit wie möglich fehlerfrei arbeitet, da die Kontrolle der Einzelergebnisse in den meisten Fällen nicht mehr zu leisten ist. Im Zuge eines Software-Abnahme-Verfahrens wird deshalb überprüft, ob die betrachtete Software fehlerfrei arbeitet, das heißt, ob die Software die erforderliche Funktionalität zuverlässig bereitstellt und ob sie darüber hinaus keine unerwünschten Nebeneffekte hat. Mit der anschließenden Freigabe der Software durch die fachlich zuständige Stelle wird die Erlaubnis erteilt, die Software zu nutzen. Gleichzeitig übernimmt diese Stelle damit auch die Verantwortung für das IT-Verfahren, dass durch die Software realisiert wird.

Bei der Software-Abnahme unterscheidet man sinnvollerweise zwischen Software, die selbst oder im Auftrag entwickelt wurde, und Standardsoftware, die nur für den speziellen Einsatzzweck angepasst wird.

Abnahme von selbst- oder im Auftrag entwickelter Software

Bevor der Auftrag zur Software-Entwicklung intern oder extern vergeben wird, muss die Anforderungsdefinition für die Software erstellt sein, aus der dann das Grob- und Feinkonzept für die Realisierung entwickelt wird. Anhand dieser Dokumente erstellt die fachlich zuständige Stelle, nicht die für die Software-Entwicklung zuständige Stelle, im allgemeinen einen Abnahmeplan.

Üblicherweise werden hierzu Testfälle und die erwarteten Ergebnisse für die Software erarbeitet. Anhand dieser Testfälle wird die Software getestet und der Abgleich zwischen berechnetem und erwartetem Ergebnis wird als Indiz für die Korrektheit der Software benutzt.

Zur Entwicklung der Testfälle und zur Durchführung der Tests ist folgendes zu beachten:

- die Testfälle werden von der fachlich zuständigen Stelle entwickelt,
- für Testfälle werden keine Daten des Wirkbetriebs benutzt,
- Testdaten, insbesondere wenn sie durch Kopieren der Wirkdaten erstellt werden, dürfen keine vertraulichen Informationen beinhalten; personenbezogene Daten sind zu anonymisieren oder zu simulieren,
- die Durchführung der Tests darf keine Auswirkungen auf den Wirkbetrieb haben; nach Möglichkeit sollte ein logisch oder physikalisch isolierter Testrechner benutzt werden.

Eine Abnahme ist zu verweigern, wenn:

- schwerwiegende Fehler in der Software festgestellt werden,
- Testfälle auftreten, in denen die erwarteten Ergebnisse nicht mit den berechneten übereinstimmen und
- Benutzerhandbücher oder Bedienungsanleitungen nicht vorhanden oder von nicht ausreichender Qualität sind und
- die Software, unter anderem der Quellcode und die Abläufe, nicht oder nicht ausreichend dokumentiert ist.

Die Ergebnisse der Abnahme sind schriftlich festzuhalten. Die Dokumentation des Abnahmeergebnisses sollte umfassen:

- Bezeichnung und Versionsnummer der Software und gegebenenfalls des IT-Verfahrens,
- Beschreibung der Testumgebung,
- Testfälle und Testergebnisse und
- Abnahmeerklärung.

Abnahme von Standardsoftware

Wird Standardsoftware beschafft, so sollte auch diese einer Abnahme und einer Freigabe unterzogen werden. In der Abnahme sollte überprüft werden, ob

- die Software frei von Computer-Viren ist,
- die Software kompatibel zu den anderen eingesetzten Produkten ist,
- die Software in der angestrebten Betriebsumgebung lauffähig ist und welche Parameter zu setzen sind,
- die Software komplett einschließlich der erforderlichen Handbücher ausgeliefert wurde und
- die geforderte Funktionalität erfüllt wird.

Freigabe-Verfahren

Ist die Abnahme der Software erfolgt, muss die Software für die Nutzung freigegeben werden. Dazu ist zunächst festzulegen, wer berechtigt ist, Software freizugeben. Die Freigabe der Software ist schriftlich festzulegen und geeignet zu hinterlegen.

Die Freigabeerklärung sollte umfassen:

- Bezeichnung und Versionsnummer der Software und gegebenenfalls des IT-Verfahrens,
- Bestätigung, dass die Abnahme ordnungsgemäß vorgenommen wurde,
- Einschränkungen für die Nutzung (Parametereinstellung, Benutzerkreis,...),
- Freigabedatum, ab wann die Software eingesetzt werden darf und
- die eigentliche Freigabeerklärung.

Falls IT-technisch möglich, muss verhindert werden, dass Software nach der Freigabe unbemerkt verändert oder manipuliert werden kann, beispielsweise durch geeignete Verfahren zum Integritätsschutz. Andernfalls müssen geeignete organisatorische Regelungen festgelegt werden, um Änderungen an der Software zu verhindern bzw. zeitnah festzustellen.

Auch nach intensiven Abnahmetests kann es vorkommen, dass im laufenden Einsatz Fehler in der Software festgestellt werden. Für diesen Fall ist festzulegen, wie in einem solchen Fehlerfall verfahren werden soll (Ansprechpartner, Fehlerbeseitigungsablauf, Beteiligung der fachlich zuständigen Stelle, Wiederholung der Abnahme und Freigabe, Versionskontrolle).

Für weiterführende Erklärungen siehe Baustein B 1.10 *Standardsoftware*.

Prüffragen:

- Gibt es für sämtliche eingesetzte Software eine Abnahmebestätigung und eine Freigabeerklärung?
- Existiert ein Verfahren, welches die Fehlerbehebung während des laufenden Einsatzes definiert?

M 2.63 Einrichten der Zugriffsrechte

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator, Verantwortliche der einzelnen Anwendungen

Arbeiten mit einem IT-System mehrere Benutzer, so muss durch eine ordnungsgemäße Administration der Zugriffsrechte sichergestellt werden, dass die Benutzer das IT-System nur gemäß ihren Aufgaben nutzen können.

Vorausgesetzt sei, dass von den Fachverantwortlichen die Zugangs- und Zugriffsberechtigungen für die einzelnen Funktionen festgelegt wurden (siehe M 2.7 *Vergabe von Zugangsberechtigungen* und M 2.8 *Vergabe von Zugriffsrechten*). Anschließend werden die Benutzer des IT-Systems den einzelnen Funktionen zugeordnet. Die Ergebnisse sind schriftlich zu dokumentieren.

Der Administrator muss dann das IT-System so konfigurieren, dass diese Benutzer Zugang zum IT-System erhalten und mit den ihnen zugewiesenen Zugriffsrechten nur ihre Aufgaben wahrnehmen können. Bietet das IT-System keine Möglichkeit, Zugriffsrechte zuzuweisen (z. B. beim DOS-PC mit mehreren Benutzern), so ist ein Zusatzprodukt zu diesem Zweck einzusetzen (siehe z. B. M 4.41 *Einsatz angemessener Sicherheitsprodukte für IT-Systeme*).

Lässt das IT-System es zu, so sind die sinnvoll einsetzbaren Protokollfunktionen zur Beweissicherung durch den Administrator zu aktivieren. Dazu gehören erfolgreiche und erfolglose An- und Abmeldevorgänge, Fehlermeldungen des Systems, unerlaubte Zugriffsversuche.

Für den Vertretungsfall muss der Administrator vorab kontrollieren, ob der Vertreter vom Fachverantwortlichen autorisiert ist. Erst dann darf er die erforderlichen Zugriffsrechte im akuten Vertretungsfall einrichten.

Prüffragen:

- Stellt die Konfiguration des IT-Systems sicher, dass Benutzer nur die Ihnen zugewiesenen Aufgaben erledigen können?
- Falls das IT-System keine Möglichkeit bietet, Zugriffsrechte zuzuweisen: Kommt ein zusätzliches Sicherheitsprodukt zum Einsatz, das sicherstellt, dass Zugriffe nur gemäß der vorher definierten Vorgaben erfolgen können?
- Wurden Protokollfunktionen, z. B. für erfolglose Anmeldevorgänge, unerlaubte Zugriffsversuche sowie Systemfehler aktiviert?
- Bei Vertretungslösungen: Wird die Autorisierung des Vertreters vor Erteilung von Zugriffsrechten durch den Administrator geprüft?

M 2.64 Kontrolle der Protokolldateien

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Revisor, Verantwortliche der einzelnen Anwendungen

Die Protokollierung sicherheitsrelevanter Ereignisse ist als Sicherheitsmaßnahme nur wirksam, wenn die protokollierten Daten in regelmäßigen Abständen durch einen Revisor ausgewertet werden. Ist es personell oder technisch nicht möglich, die Rolle eines unabhängigen Revisors für Protokolldateien zu implementieren, kann ihre Auswertung auch durch den Administrator erfolgen. Für diesen Fall bleibt zu beachten, dass damit eine Kontrolle der Tätigkeiten des Administrators nur schwer möglich ist. Das Ergebnis der Auswertung sollte daher dem IT-Sicherheitsbeauftragten, dem IT-Verantwortlichen oder einem anderen besonders zu bestimmenden Mitarbeiter vorgelegt werden.

Die regelmäßige Kontrolle dient darüber hinaus auch dem Zweck, durch die anschließende Löschung der Protokolldaten ein übermäßiges Anwachsen der Protokolldateien zu verhindern. Je nach Art der Protokolldaten kann es sinnvoll sein, diese auf externen Datenträgern zu archivieren.

Da Protokolldateien in den meisten Fällen personenbezogene Daten beinhalten, ist sicherzustellen, dass diese Daten nur zum Zweck der Datenschutzkontrolle, der Datensicherung oder zur Sicherstellung eines ordnungsgemäßen Betriebes verwendet werden (siehe § 14 Abs. 4 BDSG und M 2.110 *Datenschutzaspekte bei der Protokollierung*). Der Umfang der Protokollierung und die Kriterien für deren Auswertung sollte dokumentiert und innerhalb der Organisation abgestimmt werden.

Aus verschiedenen gesetzlichen Regelungen können sich einerseits Mindestaufbewahrungsfristen, aber andererseits auch Höchstaufbewahrungsfristen an Protokolldaten ergeben. So kann durch datenschutzrechtliche Regelungen eine Löschung erforderlich sein (siehe dazu auch M 2.110 *Datenschutzaspekte bei der Protokollierung*).

Für bestimmte Protokolldaten gelten aber unter Umständen gesetzliche Mindestaufbewahrungsfristen, z. B. wenn sie Aufschluss über betriebswirtschaftliche Vorgänge geben. Diese Fristen müssen auf jeden Fall eingehalten werden. Vor der Löschung von Protokolldaten ist daher sorgfältig zu prüfen, ob entsprechende Rechtsvorschriften zu beachten sind und ggf. welche Aufbewahrungsfristen sich daraus ergeben. Hierbei sollte die Rechtsabteilung beteiligt werden.

Die nachfolgenden Auswertungskriterien dienen als Beispiele, die Hinweise auf eventuelle Sicherheitslücken, Manipulationsversuche und Unregelmäßigkeiten erkennen lassen:

- Liegen die Zeiten des An- und Abmeldens außerhalb der Arbeitszeit (Hinweis auf Manipulationsversuche)?
- Häufen sich fehlerhafte Anmeldeversuche (Hinweis auf den Versuch, Passwörter zu erraten)?
- Häufen sich unzulässige Zugriffsversuche (Hinweis auf Versuche zur Manipulation)?
- Gibt es auffällig große Zeitintervalle, in denen keine Protokolldaten aufgezeichnet wurden (Hinweis auf eventuell gelöschte Protokollsätze)?
- Ist der Umfang der protokollierten Daten zu groß (eine umfangreiche Protokolldatei erschwert das Auffinden von Unregelmäßigkeiten)?

- Gibt es auffällig große Zeitintervalle, in denen anscheinend kein Benutzerwechsel stattgefunden hat (Hinweis darauf, dass das konsequente Abmelden nach Arbeitsende nicht vollzogen wird)?
- Gibt es auffallend lange Verbindungszeiten in öffentliche Netze hinein (siehe G 4.25 *Nicht getrennte Verbindungen*)?
- Wurde in einzelnen Netzsegmenten oder im gesamten Netz eine auffällig hohe Netzlast oder eine Unterbrechung des Netzbetriebes festgestellt (Hinweis auf Versuche, die Dienste des Netzes zu verhindern bzw. zu beeinträchtigen oder auf eine ungeeignete Konzeption bzw. Konfiguration des Netzes)?

Bei der Auswertung der Protokolldateien sollte besonderes Augenmerk auf alle Zugriffe gelegt werden, die unter Administratorkennungen durchgeführt wurden.

Wenn regelmäßig umfangreiche Protokolldateien ausgewertet werden müssen, ist es sinnvoll, ein Werkzeug zur Auswertung zu benutzen. Dieses Werkzeug sollte wählbare Auswertungskriterien zulassen und besonders kritische Einträge (z. B. mehrfacher fehlerhafter Anmeldeversuch) hervorheben.

Das oben Gesagte gilt analog auch für die Erhebung von Auditdaten, da es sich dabei im Prinzip nur um die Protokollierung sicherheitskritischer Ereignisse handelt.

Prüffragen:

- Gibt es einen Verantwortlichen für die Auswertung von Protokolldaten?
- Werden die Ergebnisse der Auswertung dem IT-Sicherheitsbeauftragten oder einem anderen hierfür bestimmten Mitarbeiter vorgelegt?
- Existiert ein Konzept, das den Umfang und die Auswertung der Protokollierung festlegt?
- Werden die gesetzlichen Vorgaben in Bezug auf die Protokolldaten eingehalten?

M 2.80 Erstellung eines Anforderungskatalogs für Standardsoftware

Verantwortlich für Initiierung: Leiter Fachabteilung
Verantwortlich für Umsetzung: Fachabteilung, Leiter IT

Zur Lösung einer Aufgabe, die mit IT bearbeitet wird, bietet der Markt meist eine Vielzahl gleichartiger Standardsoftwareprodukte an. In ihrer Grundfunktionalität vergleichbar, unterscheiden sie sich jedoch in Kriterien wie Anschaffungs- und Betriebskosten, Zusatzfunktionalitäten, Kompatibilität, Administration, Ergonomie und Informationssicherheit .

Anforderungskatalog

Für die Auswahl eines geeigneten Produktes muss daher zunächst ein Anforderungskatalog erstellt werden. Der Anforderungskatalog sollte unter anderem zu den folgenden Punkten Aussagen enthalten:

- **Funktionale Anforderungen**, die das Produkt zur Unterstützung der Aufgabenerfüllung der Fachabteilung erfüllen muss. Die für die Fachaufgabe relevanten Einzelfunktionalitäten sollten hervorgehoben werden.
  Verkürzte Beispiele:
  - Textverarbeitung mit den Zusatzfunktionen Einbinden von Grafiken, Makro-Programmierung, Rechtschreibprüfung und Silbentrennung. Makro-Programmierung muss abschaltbar sein, Rechtschreibprüfung muss in Englisch, Französisch und Deutsch verfügbar sein. Die spezifizierten Textformate müssen im- und exportiert werden können.
  - Datenbank (Front-End und Back-End) für Multi-User-Betrieb mit Unterstützung der Standardabfragesprache SQL und grafischer Bedienoberfläche
  - Terminplaner zur Koordinierung und Kontrolle von Terminen der Abteilungsangehörigen mit integrierter Terminabstimmung, automatischem Versand von Einladungen und Aufgaben- und Prioritäten-Listen, Schnittstelle zum hausinternen Mailprogramm
- **IT-Einsatzumgebung**, diese wird einerseits beschrieben durch die Rahmenbedingungen, die durch die vorhandene oder geplante IT-Einsatzumgebung vorgegeben werden, und andererseits durch die Leistungsanforderungen, die durch das Produkt an die Einsatzumgebung vorgegeben werden.
  Verkürztes Beispiel:
  - Erforderliche IT-Einsatzumgebung und Leistungsanforderungen: Betriebssystem, Prozessor, Hauptspeicher, Festplattenkapazität, Schnittstellen für externe Datenträger und für Vernetzung
- **Kompatibilitätsanforderungen** zu anderen Programmen oder IT-Systemen, also Migrationsunterstützung und Aufwärts- und Abwärtskompatibilität.
  Verkürzte Beispiele:
  - Datenbestände aus der vorhandenen Datenbank XYZ müssen übernommen werden können.
  - Die Funktionen A, B, C müssen bei Versionswechseln erhalten bleiben.
  - Der Datenaustausch mit dem Unix-System XYZ muss möglich sein.

- **Performanceanforderungen** beschreiben die erforderlichen Leistungen hinsichtlich Durchsatz und Laufzeitverhalten. Für die geforderten Funktionen sollten möglichst genaue Angaben über die maximal zulässige Bearbeitungszeit getroffen werden.
  Verkürzte Beispiele:
  - Die maximale Antwortzeit bei Ausführung von Funktion X darf 2 Sekunden nicht überschreiten.
  - Andere gleichzeitig verarbeitete Prozesse dürfen durch das Produkt maximal um 30% verlangsamt werden.
- **Interoperabilitätsanforderungen**, d. h. die Zusammenarbeit mit anderen Produkten über Plattformgrenzen hinweg muss möglich sein.
  Verkürzte Beispiele:
  - Versionen des Textverarbeitungsprogramms sollen für Windows-, Unix- und MacOS-Plattformen verfügbar sein (in den zu benennenden Versionen). Dokumente sollen auf einem Betriebssystem erstellt und auf einem anderen weiterverarbeitet werden können.
  - Das Textverarbeitungsprogramm muss mit dem eingesetzten Mailprogramm zusammenarbeiten können.
- **Zuverlässigkeitsanforderungen** betreffen die Stabilität des Produktes, also Fehlererkennung und Toleranz sowie Ausfall- und Betriebssicherheit.
  Verkürzte Beispiele:
  - Fehleingaben des Benutzers müssen erkannt werden und dürfen nicht zum Programmabbruch oder Systemabsturz führen.
  - Die Datenbank muss über Mechanismen verfügen, die es erlauben, bei einem Systemabbruch mit Zerstörung der Datenbank alle Transaktionen zu rekonstruieren (Roll-Forward).
- **Konformität zu Standards**, dies können internationale Normen, De-facto-Standards oder auch Hausstandards sein.
  Verkürztes Beispiel:
  - Das Produkt muss der EU-Bildschirmrichtlinie 90/270/EWG entsprechen.
- **Einhaltung von internen Regelungen und gesetzlichen Vorschriften** (z. B. ausreichender Datenschutz bei der Verarbeitung personenbezogener Daten)
  Verkürzte Beispiele:
  - Das Produkt muss den Grundsätzen ordnungsmäßiger DV-gestützter Buchführungssysteme genügen.
  - Da personenbezogene Daten verarbeitet werden, müssen die Bestimmungen des Bundesdatenschutzgesetzes mit den implementierten Funktionen erfüllt werden können.
- **Anforderungen an die Benutzerfreundlichkeit**, die durch die leichte Bedienbarkeit, Verständlichkeit und Erlernbarkeit gekennzeichnet ist, also insbesondere durch die Güte der Benutzeroberfläche sowie die Qualität der Benutzerdokumentation und der Hilfefunktionen.
  Verkürzte Beispiele:
  - Eine Online-Hilfefunktion muss implementiert sein.
  - Die Benutzeroberfläche muss so gestaltet sein, dass ungelernte Kräfte innerhalb von zwei Stunden in die Benutzung eingewiesen werden können.
  - Die Benutzerdokumentation und die Benutzeroberfläche sollten in der Landessprache vorliegen.

- **Anforderungen an die Wartbarkeit** ergeben sich für den Anwender hauptsächlich aus der Fehlerbehandlung des Produktes.
  Verkürzte Beispiele:

  - Der Administrationsaufwand darf nicht zu hoch sein.
  - Der Anbieter muss eine Hotline für Fragen anbieten.
  - Das Produkt muss einfach zu installieren und zu konfigurieren sein.
  - Das Produkt muss einfach zu deinstallieren sein.
- die **Obergrenze der Kosten**, die durch die Beschaffung dieses Produktes verursacht würden, werden vorgegeben. Dabei müssen nicht nur die unmittelbaren Beschaffungskosten für das Produkt selber einbezogen werden, sondern auch Folgekosten, wie z. B. eine Aufrüstung der Hardware, Personalkosten oder notwendige Schulungen.
  Verkürzte Beispiele:

  - Das Produkt darf maximal 15.000,- Euro kosten.
  - Die Schulungskosten dürfen 2.000,- Euro nicht überschreiten
- Aus den **Anforderungen an die Dokumentation** muss hervorgehen, welche Dokumente in welcher Güte (Vollständigkeit, Verständlichkeit) erforderlich sind.
  Verkürzte Beispiele:

  - Die Benutzerdokumentation muss leicht nachvollziehbar und zum Selbststudium geeignet sein. Die gesamte Funktionalität des Produktes ist zu beschreiben.
  - Die Systemverwalterdokumentation muss Handlungsanweisungen für mögliche Fehler enthalten.
- Bezüglich der **Softwarequalität** können Anforderungen gestellt werden, die von Herstellererklärungen zum eingesetzten Qualitätssicherungsverfahren, über ISO 9000 ff. Zertifikate bis hin zu unabhängigen Softwareprüfungen nach ISO/IEC 25051 reichen.
  Verkürzte Beispiele:

  - Der Software-Herstellungsprozess des Herstellers muss nach ISO 9000 zertifiziert sein.
  - Die Funktionalität des Produktes muss unabhängig gemäß ISO/IEC 25051 überprüft worden sein.
- Sollen durch das Produkt Sicherheitsfunktionen erfüllt werden, sind sie in Form von **Sicherheitsanforderungen** zu formulieren (siehe M 4.42 *Implementierung von Sicherheitsfunktionalitäten in der IT-Anwendung*). Dies wird nachfolgend noch ausführlich erläutert.

Sicherheitsanforderungen

Abhängig davon, ob das Produkt Sicherheitseigenschaften bereitstellen muss, können im Anforderungskatalog Sicherheitsfunktionen aufgeführt werden. Typische Sicherheitsfunktionen, die hier in Frage kommen, seien kurz erläutert. Weitere Ausführungen findet man in den Common Criteria (den "Gemeinsamen Kriterien für die Prüfung und Bewertung der Sicherheit von Informationstechnik").

- **Identifizierung und Authentisierung**
  In vielen Produkten wird es Anforderungen geben, diejenigen Benutzer zu bestimmen und zu überwachen, die Zugriff auf Betriebsmittel haben, die vom Produkt kontrolliert werden. Dazu muss nicht nur die behauptete Identität des Benutzers festgestellt, sondern auch die Tatsache nachgeprüft werden, dass der Benutzer tatsächlich die Person ist, die er zu sein vorgibt. Dies geschieht, indem der Benutzer dem Produkt Informationen liefert, die fest mit dem betreffenden Benutzer verknüpft sind.

- **Zugriffskontrolle**
  Bei vielen Produkten wird es erforderlich sein sicherzustellen, dass Benutzer und Prozesse, die für diese Benutzer tätig sind, daran gehindert werden, Zugriff auf Informationen oder Betriebsmittel zu erhalten, für die sie kein Zugriffsrecht haben oder für die keine Notwendigkeit zu einem Zugriff besteht. Desgleichen wird es Anforderungen bezüglich der unbefugten Erzeugung oder Änderung (einschließlich Löschung) von Informationen geben.
- **Beweissicherung**
  Bei vielen Produkten wird es erforderlich sein sicherzustellen, dass über Handlungen, die von Benutzern bzw. von Prozessen im Namen solcher Benutzer ausgeführt werden, Informationen aufgezeichnet werden, damit die Folgen solcher Handlungen später dem betreffenden Benutzer zugeordnet werden können und der Benutzer für seine Handlungen verantwortlich gemacht werden kann.
- **Protokollauswertung**
  Bei vielen Produkten wird sicherzustellen sein, dass sowohl über gewöhnliche Vorgänge als auch über außergewöhnliche Vorfälle ausreichend Informationen aufgezeichnet werden, damit durch Nachprüfungen später festgestellt werden kann, ob tatsächlich Sicherheitsverletzungen vorgelegen haben und welche Informationen oder sonstigen Betriebsmittel davon betroffen waren.
- **Unverfälschbarkeit**
  Bei vielen Produkten wird es erforderlich sein sicherzustellen, dass bestimmte Beziehungen zwischen unterschiedlichen Daten korrekt bleiben und dass Daten zwischen einzelnen Prozessen ohne Änderungen übertragen werden.
  Daneben müssen auch Funktionen bereitgestellt werden, die es bei der Übertragung von Daten zwischen einzelnen Prozessen, Benutzern und Objekten ermöglichen, Verluste, Ergänzungen oder Veränderungen zu entdecken bzw. zu verhindern, und die es unmöglich machen, die angebliche oder tatsächliche Herkunft bzw. Bestimmung der Datenübertragung zu ändern.
- **Zuverlässigkeit**
  Bei vielen Produkten wird es erforderlich sein sicherzustellen, dass zeitkritische Aufgaben genau zu dem Zeitpunkt durchgeführt werden, zu dem es erforderlich ist, also nicht früher oder später, und es wird sicherzustellen sein, dass zeitunkritische Aufgaben nicht in zeitkritische umgewandelt werden können. Desgleichen wird es bei vielen Produkten erforderlich sein sicherzustellen, dass ein Zugriff in dem erforderlichen Moment möglich ist und Betriebsmittel nicht unnötig angefordert oder zurückgehalten werden.
- **Übertragungssicherung**
  Dieser Begriff umfasst alle Funktionen, die für den Schutz der Daten während der Übertragung über Kommunikationskanäle vorgesehen sind:

  - Authentisierung
  - Zugriffskontrolle
  - Datenvertraulichkeit
  - Datenintegrität
  - Sende- und Empfangsnachweis

  Einige dieser Funktionen werden mittels kryptographischer Verfahren realisiert.

Darüber hinaus können weitere Sicherheitsanforderungen an Standardsoftware konkretisiert werden.

- **Datensicherung**

An die Verfügbarkeit der mit dem Produkt verarbeiteten Daten werden hohe Anforderungen gestellt. Unter diesen Punkt fallen im Produkt integrierte Funktionen, die Datenverlusten vorbeugen sollen wie die automatische Speicherung von Zwischenergebnissen oder die automatische Erstellung von Sicherungskopien vor der Durchführung größerer Änderungen.

- **Verschlüsselung**
  Verschlüsselung dient der Wahrung der Vertraulichkeit von Daten. Bei vielen Produkten wird es erforderlich sein, Nutzdaten vor einer Übertragung oder nach der Bearbeitung zu verschlüsseln und sie nach Empfang oder vor der Weiterverarbeitung zu entschlüsseln. Hierzu ist ein anerkanntes Verschlüsselungsverfahren zu verwenden. Es ist sicherzustellen, dass die zur Entschlüsselung benötigten Parameter (z. B. Schlüssel) in der Weise geschützt sind, dass kein Unbefugter Zugang zu diesen Daten besitzt.
- **Funktionen zur Wahrung der Datenintegrität**
  Für Daten, deren Integritätsverlust zu Schäden führen kann, können Funktionen eingesetzt werden, die Fehler erkennen lassen oder sogar mittels Redundanz korrigieren können. Meist werden Verfahren zur Integritätsprüfung eingesetzt, die absichtliche Manipulationen am Produkt bzw. den damit erstellten Daten sowie ein unbefugtes Wiedereinspielen von Daten zuverlässig aufdecken können. Sie basieren auf kryptographischen Verfahren (siehe M 4.34 *Einsatz von Verschlüsselung, Checksummen oder Digitalen Signaturen*).
- **Datenschutzrechtliche Anforderungen**
  Wenn mit dem Produkt personenbezogene Daten verarbeitet werden sollen, sind über die genannten Sicherheitsfunktionen hinaus zusätzliche spezielle technische Anforderungen zu stellen, um den Datenschutzbestimmungen genügen zu können.

Stärke der Mechanismen / Angriffsresistenz

Sicherheitsfunktionen werden durch Mechanismen umgesetzt. Je nach Einsatzzweck müssen diese Mechanismen eine unterschiedliche Stärke besitzen, mit der sie Angriffe abwehren können. Die erforderliche Stärke der Mechanismen ist im Anforderungskatalog anzugeben. Bei Anwendung der Common Criteria (CC) wird die Angriffsresistenz eines IT-Produktes, das in einer bestimmten Einsatzumgebung betrieben wird, an den in den Sicherheitsvorgaben oder gegebenenfalls in einem Schutzprofil definierten Bedrohungen der zu schützenden Datenobjekte und der für die Evaluierung angesetzten Prüftiefe bewertet. Die geforderte Prüftiefe beinhaltet die Festlegung der Angriffsresistenz und richtet sich nach dem Schutzbedarf und dem Einsatzzweck des Produktes. Die Prüftiefe wird anhand eines Kataloges (siehe CC, Teil 3) meist mittels vordefinierter Evaluierungsstufen (EAL 1 bis 7) festgelegt.

Für die Bewertung der Angriffsresistenz werden die für das Einsatzszenario relevanten Angriffe nach dem Stand der Technik bis zu einer bestimmten Stärke unter Berücksichtigung der erforderlichen Angriffszeit, technischen Expertise des Angreifers, Kenntnissen über das Produkt, Gelegenheit zum Angriff und benötigten Hilfsmittel analysiert. Die Bestätigung der Angriffsresistenz im Rahmen der Zertifizierung erfolgt dabei dann in den Abstufungen niedrig (basic), erweitert (enhanced basic), mittel (moderate) und hoch (high).

Basic bedeutet Schutz gegen öffentlich bekannte Angriffe und gegen Angreifer mit sehr begrenzten Fähigkeiten und Möglichkeiten. Hoch bedeutet, dass ein erfolgreicher Angriff sehr gute Fachkenntnisse, Produktkenntnisse, Gelegenheiten und Betriebsmittel erfordert, und damit insgesamt als extrem aufwändig gilt.

Beispiele für Anforderungen zu Sicherheitseigenschaften

Nachfolgend werden für einige wichtige Sicherheitsfunktionen Beispiele genannt, aus denen typische Anforderungen an Sicherheitseigenschaften deutlich werden.

Soll das Produkt über einen **Identifizierungs- und Authentisierungsmechanismus** verfügen, können beispielsweise folgende Anforderungen gestellt werden:

- Der Zugang darf ausschließlich über eine definierte Schnittstelle erfolgen. Dabei kann z. B. ein Anmeldemechanismus zum Einsatz kommen, der eine eindeutige Benutzer-Kennung und ein Passwort verlangt. Wird beim Zugang zum IT-System bereits die Identität des Benutzers sichergestellt, ist eine anonyme Passworteingabe ausreichend. Andere Möglichkeiten sind Verfahren, die auf dem Besitz bestimmter "Token" beruhen, wie z. B. einer Chipkarte.
- Das Zugangsverfahren selbst muss die sicherheitskritischen Parameter, wie Passwort, Benutzer-Kennung, usw., sicher verwalten. So dürfen aktuelle Passwörter nie unverschlüsselt auf den entsprechenden IT-Systemen gespeichert werden.
- Das Zugangsverfahren muss definiert auf Fehleingaben reagieren. Erfolgt zum Beispiel dreimal hintereinander eine fehlerhafte Authentisierung, ist der Zugang zum Produkt zu verwehren oder alternativ sind die zeitlichen Abstände, nach denen ein weiterer Zugangsversuch erlaubt wird, sukzessiv zu vergrößern.
- Das Zugangsverfahren muss das Setzen bestimmter Minimalvorgaben für die sicherheitskritischen Parameter zulassen. So sollte die Mindestlänge eines Passwortes acht Zeichen, die Mindestlänge einer PIN vier Ziffern betragen. Auch die Komplexität für Passwörter sollte vorgegeben werden.

Soll das Produkt über eine **Zugriffskontrolle** verfügen, können beispielsweise folgende Anforderungen gestellt werden:

- Das Produkt muss verschiedene Benutzer unterscheiden können.
- Das Produkt muss je nach Vorgabe Ressourcen einzelnen autorisierten Benutzer zuteilen können und Unberechtigten den Zugriff gänzlich verwehren
- Mittels einer differenzierten Rechtestruktur (lesen, schreiben, ausführen, ändern, ...) sollte der Zugriff geregelt werden können. Die für die Rechteverwaltung relevanten Daten sind manipulationssicher vom Produkt zu verwalten.

Soll das Produkt über eine **Protokollierung** verfügen, können folgende Anforderungen sinnvoll sein:

- Der Mindestumfang, den das Produkt protokollieren können muss, sollte parametrisierbar sein. Beispielsweise sollten folgende Aktionen protokollierbar sein:
  - bei Authentisierung: Benutzer-Kennung, Datum und Uhrzeit, Erfolg, ...,
  - bei der Zugriffskontrolle: Benutzer-Kennung, Datum und Uhrzeit, Erfolg, Art des Zugriffs, was wurde wie geändert, gelesen, geschrieben, ...
  - Durchführung von Administratortätigkeiten,
  - Auftreten von funktionalen Fehlern.
- Die Protokollierung darf von Unberechtigten nicht deaktivierbar sein. Die Protokolle selbst dürfen für Unberechtigte weder lesbar noch modifizierbar sein.

- Die Protokollierung muss übersichtlich, vollständig und korrekt sein.

Soll das Produkt über eine **Protokollauswertung** verfügen, können folgende Anforderungen sinnvoll sein:

- Eine Auswertefunktion muss nach den bei der Protokollierung geforderten Datenarten unterscheiden können (z. B. "Filtern aller unberechtigten Zugriffe auf alle Ressourcen in einem vorgegebenen Zeitraum").
- Die Auswertefunktion muss auswertbare ("lesbare") Berichte erzeugen, so dass keine sicherheitskritischen Aktivitäten übersehen werden.

Soll das Produkt über Funktionen zur **Unverfälschbarkeit** verfügen, könnte beispielsweise folgende Anforderung gestellt werden:

- Ein Datenbank-Managementsystem muss über Möglichkeiten zur Beschreibung von Regeln bestimmter Beziehungen zwischen den gespeicherten Daten verfügen (z. B. referentielle Integrität). Außerdem müssen geeignete Mechanismen existieren, die verhindern, dass es durch Änderungen der Daten zu Verstößen gegen diese Regeln kommt.

Soll das Produkt über Funktionen zur **Datensicherung** verfügen, können beispielsweise folgende Anforderungen gestellt werden:

- Es muss konfigurierbar sein, welche Daten wann gesichert werden.
- Es muss eine Option zum Einspielen beliebiger Datensicherungen existieren.
- Die Funktion muss das Sichern von mehreren Generationen ermöglichen.
- Datensicherungen von Zwischenergebnissen aus der laufenden Anwendung sollen möglich sein.

Soll das Produkt über eine **Verschlüsselungskomponente** verfügen, sind folgende Anforderungen sinnvoll:

- Der implementierte Verschlüsselungsalgorithmus sollte dem Schutzbedarf entsprechen und eine ausreichende Mechanismenstärke besitzen (siehe auch M 2.164 *Auswahl eines geeigneten kryptographischen Verfahrens*).
- Das Schlüsselmanagement muss mit der Funktionalität des Produktes harmonieren. Dabei sind insbesondere grundsätzliche Unterschiede der Algorithmen zu berücksichtigen:
  - symmetrische Verfahren benutzen einen geheim zu haltenden Schlüssel für die Ver- und Entschlüsselung,
  - asymmetrische Verfahren benutzen einen öffentlichen Schlüssel für die Verschlüsselung und einen privaten (geheim zu haltenden) für die Entschlüsselung.
- Das Produkt muss die sicherheitskritischen Parameter wie Schlüssel sicher verwalten. So dürfen Schlüssel (auch mittlerweile nicht mehr benutzte) nie ungeschützt, das heißt auslesbar, auf den entsprechenden IT-Systemen abgelegt werden.

Soll das Produkt über Mechanismen zur **Integritätsprüfung** verfügen, sind folgende Anforderungen sinnvoll:

- Das Produkt führt bei jedem Programmaufruf einen Integritätscheck durch.
- Bei der Datenübertragung müssen Mechanismen eingesetzt werden, mit denen absichtliche Manipulationen an den Adressfeldern und den Nutzdaten erkannt werden können. Daneben darf die bloße Kenntnis der eingesetzten Algorithmen ohne spezielle Zusatzkenntnisse nicht ausreichen, unerkannte Manipulationen an den obengenannten Daten vorzunehmen.

Werden personenbezogene Daten mit dem Produkt verarbeitet, können beispielsweise folgende **datenschutzrechtlichen Anforderungen** gestellt werden:

- Das Produkt darf keine freie Abfrage für Datenauswertungen zulassen. Die Auswertungen von Datensätzen müssen auf bestimmte Kriterien einschränkbar sein.
- Es muss parametrisierbar sein, dass für bestimmte Dateien Änderungen, Löschungen oder Ausdrucke von personenbezogenen Daten nur nach dem Vier-Augen-Prinzip möglich sind.
- Die Protokollierung muss parametrisierbar sein, so dass aufgezeichnet werden kann, wer wann an welchen personenbezogenen Daten welche Änderungen vorgenommen hat.
- Die Übermittlung personenbezogener Daten muss durch geeignete Stichprobenverfahren festgestellt und überprüft werden können (BDSG, § 10). Die Art der Stichprobe muss sich individuell einstellen lassen.
- Das Produkt muss das Löschen von personenbezogenen Daten ermöglichen. Ersatzweise muss das Sperren personenbezogener Daten möglich sein, um ihre weitere Verarbeitung oder Nutzung einzuschränken bzw. zu verhindern.

Bewertungsskala

Um einen Vergleich verschiedener Produkte im Sinne einer Nutzwertanalyse durchführen zu können, müssen Kriterien vorhanden sein, wie die Erfüllung der einzelnen Anforderungen gewertet wird. Dazu ist es erforderlich, vorab die Bedeutung der einzelnen Anforderungen für die angestrebte IT-gestützte Aufgabenerfüllung quantitativ oder qualitativ zu bewerten.

Diese Bewertung kann beispielsweise in drei Stufen vorgenommen werden. In der ersten Stufe wird festgelegt, welche im Anforderungskatalog geforderten Eigenschaften **notwendig** und welche **wünschenswert** sind. Wenn eine notwendige Eigenschaft nicht erfüllt ist, wird das Produkt abgelehnt (so genanntes K.O.-Kriterium). Das Fehlen einer wünschenswerten Eigenschaft wird zwar negativ gewertet, dennoch wird aber das Produkt aufgrund dessen nicht zwingend abgelehnt.

Als zweite Stufe wird die **Bedeutung** der geforderten wünschenswerten Eigenschaft für die Aufgabenerfüllung angegeben. Dies kann z. B. quantitativ mit Werten zwischen 1 für niedrig und 5 für hoch erfolgen. Notwendige Eigenschaften müssen nicht quantitativ bewertet werden. Ist dies aber aus rechnerischen Gründen erforderlich, müssen sie auf jeden Fall höher bewertet werden als jede wünschenswerte Eigenschaft (um die Bedeutung einer notwendigen Eigenschaft hervorzuheben, kann sie z. B. mit 10 bewertet werden).

In der dritten Stufe wird ein **Vertrauensanspruch** für die Korrektheit der geforderten Eigenschaften für die Aufgabenerfüllung angegeben (z. B. mit Werten zwischen 1 für niedrig und 5 für hoch). Anhand des Vertrauensanspruchs wird später entschieden, wie eingehend die Eigenschaft getestet wird. Der Vertrauensanspruch der Sicherheitsmechanismen muss entsprechend ihrer Mechanismenstärke bewertet werden, beispielsweise kombiniert man

- Mechanismenstärke niedrig mit Vertrauensanspruch 1
- Mechanismenstärke mittel mit Vertrauensanspruch 3
- Mechanismenstärke hoch mit Vertrauensanspruch 5

Diese Orientierungswerte müssen im Einzelfall verifiziert werden.

Beispiele:

Auszugsweise sollen für einige typische Standardsoftwareprodukte Sicherheitsanforderungen erläutert werden:

Textverarbeitungsprogramm:

Notwendige Sicherheitseigenschaften:

- Automatische Datensicherung von Zwischenergebnissen im laufenden Betrieb

Wünschenswerte Sicherheitseigenschaften:

- Passwortschutz einzelner Dateien
- Verschlüsselung einzelner Dateien
- Makro-Programmierung muss abschaltbar sein

Dateikompressionsprogramm:

Notwendige Sicherheitseigenschaften:

- Im Sinne der Datensicherung dürfen nach Kompression zu löschende Dateien erst dann vom Kompressionsprogramm gelöscht werden, wenn die Kompression fehlerfrei abgeschlossen wurde.
- Vor der Dekomprimierung einer Datei muss deren Integrität überprüft werden, damit z. B. Bitfehler in der komprimierten Datei erkannt werden.

Wünschenswerte Sicherheitseigenschaften:

- Passwortschutz komprimierter Dateien

Terminplaner:

Notwendige Sicherheitseigenschaften:

- Eine sichere Identifikation und Authentisierung der einzelnen Benutzer muss erzwungen werden, z. B. über Passwörter.
- Eine Zugriffskontrolle für die Terminpläne der einzelnen Mitarbeiter ist erforderlich.
- Zugriffsrechte müssen für Einzelne, Gruppen und Vorgesetzte getrennt vergeben werden können.
- Eine Unterscheidung zwischen Lese- und Schreibrecht muss möglich sein.

Wünschenswerte Sicherheitseigenschaften:

- Eine automatisierte Datensicherung in verschlüsselter Form ist vorzusehen.

Reisekostenabrechnungssystem:

Notwendige Sicherheitseigenschaften:

- Eine sichere Identifikation und Authentisierung der einzelnen Benutzer muss erzwungen werden, z. B. über Passwörter.
- Eine Zugriffskontrolle muss vorhanden und auch für einzelne Datensätze einsetzbar sein.
- Zugriffsrechte müssen für Benutzer, Administrator, Revisor und Datenschutzbeauftragten getrennt vergeben werden können. Eine Rollentrennung zwischen Administrator und Revisor muss durchführbar sein.
- Datensicherungen müssen so durchgeführt werden können, dass sie verschlüsselt abgelegt werden und nur von Berechtigten wiedereingespielt werden können.
- Detaillierte Protokollierungsfunktionen müssen verfügbar sein.

Wünschenswerte Sicherheitseigenschaften:

- Ein optionaler Integritätscheck für zahlungsrelevante Daten sollte angeboten werden.

Beispiel für eine Bewertungsskala:

Eine Fachabteilung will für Datensicherungszwecke ein Komprimierungsprogramm beschaffen. Nach der Erstellung eines Anforderungskataloges könnten die dort spezifizierten Eigenschaften wie folgt bewertet werden:

Eigenschaft	**notwendig**	**wünschenswert**	**Bedeutung**	**Vertrauensanspruch**
korrekte Kompression und Dekompression	X		10	5
Erkennen von Bitfehlern in einer komprimierten Datei	X		10	2
Löschung von Dateien nur nach erfolgreicher Kompression	X		10	3
Windows-PC, x86, 256 MB	X		10	5
Linux-tauglich		X	2	1
Durchsatz bei 1 GHz über 10 MB/s		X	4	3
Kompressionsrate über 40% bei Textdateien des Programms XYZ		X	4	3
Online-Hilfefunktion		X	3	1
Maximale Kosten 50.- Euro pro Lizenz	X		10	5
Passwortschutz für komprimierte Dateien (Mechanismenstärke hoch)		X	2	5

Prüffragen:

- Wird für die Auswahl von einzusetzenden Software-Produkten ein Anforderungskatalog erstellt, der Sicherheitsanforderungen umfasst?

- Wurde eine Bewertungsskala zu den einzelnen Anforderungen an Software-Produkte erstellt, um die Produkte vergleichen zu können?

M 2.110 Datenschutzaspekte bei der Protokollierung

Verantwortlich für Initiierung: Leiter IT
Verantwortlich für Umsetzung: Administrator

Unter Protokollierung beim Betrieb von IT-Systemen ist im datenschutzrechtlichen Sinn die Erstellung von manuellen oder automatisierten Aufzeichnungen zu verstehen, aus denen sich die Fragen beantworten lassen: "Wer hat wann mit welchen Mitteln was veranlasst bzw. worauf zugegriffen?" Außerdem müssen sich Systemzustände ableiten lassen: "Wer hatte von wann bis wann welche Zugriffsrechte?"

Art und Umfang von Protokollierungen hängen vom allgemeinen Datenschutzrecht und auch von bereichsspezifischen Regelungen ab.

Die Protokollierung der Administrationsaktivitäten entspricht einer Systemüberwachung, während die Protokollierung der Benutzeraktivitäten im wesentlichen der Verfahrensüberwachung dient. Dementsprechend finden sich die Anforderungen an die Art und den Umfang der systemorientierten Protokollierung überwiegend im allgemeinen Datenschutzrecht, während die verfahrensorientierte Protokollierung oft durch bereichsspezifische Regelungen definiert wird. Beispiele für verfahrensorientierte Protokollierung sind u. a. Meldegesetze, Polizeigesetze, Verfassungsschutzgesetze.

Mindestanforderungen an die Protokollierung

Bei der Administration von IT-Systemen sind die folgenden Aktivitäten vollständig zu protokollieren:

- **Systemgenerierung und Modifikation von Systemparametern**
  Da auf dieser Ebene in der Regel keine systemgesteuerten Protokolle erzeugt werden, bedarf es entsprechender detaillierter manueller Aufzeichnungen, die mit der Systemdokumentation korrespondieren sollten.
- **Einrichten von Benutzern**
  Wem von wann bis wann durch wen das Recht eingeräumt worden ist, das betreffende IT-System zu benutzen, ist vollständig zu protokollieren. Für diese Protokolle sollten längerfristige Aufbewahrungszeiträume vorgesehen werden, da sie Grundlage praktisch jeder Revisionsmaßnahme sind.
- **Erstellung von Rechteprofilen**
  Im Rahmen der Protokollierung der Benutzerverwaltung kommt es insbesondere auch darauf an aufzuzeichnen, wer die Anweisung zur Einrichtung bestimmter Benutzerrechte erteilt hat (siehe auch M 2.31 *Dokumentation der zugelassenen Benutzer und Rechteprofile*).
- **Einspielen und Änderung von Anwendungssoftware**
  Die Protokolle repräsentieren das Ergebnis der Programm- und Verfahrensfreigaben.
- **Änderungen an der Dateiorganisation**
  Im Hinblick auf die vielfältigen Manipulationsmöglichkeiten, die sich bereits bei Benutzung der "Standard-Dateiverwaltungssysteme" ergeben, kommt einer vollständigen Protokollierung eine besondere Bedeutung zu (siehe z. B. Datenbankmanagement).
- **Durchführung von Datensicherungsmaßnahmen**
  Da derartige Maßnahmen (Backup, Restore) mit der Anfertigung von Kopien bzw. dem Überschreiben von Datenbeständen verbunden sind und häufig in "Ausnahmesituationen" durchgeführt werden, besteht eine erhöhte Notwendigkeit zur Protokollierung.

- **Sonstiger Aufruf von Administrations-Tools**
  Die Benutzung aller Administrations-Tools ist zu protokollieren, um feststellen zu können, ob Unbefugte sich Systemadministrator-Rechte erschlichen haben.
- **Versuche unbefugten Einloggens und Überschreitung von Befugnissen**
  Geht man von einer wirksamen Authentisierungsprozedur und sachgerechten Befugniszuweisungen aus, kommt der vollständigen Protokollierung aller "auffälligen Abnormalitäten" beim Einloggen und der Benutzung von Hard- und Software-Komponenten eine zentrale Bedeutung zu. Benutzer in diesem Sinne ist auch der Systemadministrator.

Bei der Verarbeitung von personenbezogenen Daten sind folgende Benutzeraktivitäten in Abhängigkeit von der Sensibilität der Verfahren bzw. Daten vollständig bzw. selektiv zu protokollieren:

- **Eingabe von Daten**
  Die so genannte Eingabekontrolle erfolgt grundsätzlich verfahrensorientiert (z. B. Protokollierung in Akten, soweit vorhanden, Protokollierung direkt im Datenbestand, sofern keine Akten geführt werden). Auch wenn man davon ausgeht, dass Befugnisüberschreitungen anderweitig protokolliert werden, sollte eine vollständige Protokollierung von Dateneingaben als Regelfall angesehen werden.
- **Datenübermittlungen**
  Nur soweit nicht gesetzlich eine vollständige Protokollierung vorgeschrieben ist, kann eine selektive Protokollierung als ausreichend angesehen werden.
- **Benutzung von automatisierten Abrufverfahren**
  In der Regel dürfte eine vollständige Protokollierung der Abrufe und der Gründe der Abrufe (Vorgang, Aktenzeichen etc.) erforderlich sein, um unbefugte Kenntnisnahme im Rahmen der grundsätzlich eingeräumten Zugriffsrechte aufdecken zu können.
- **Löschung von Daten**
  Die Durchführung der Löschung ist zu protokollieren.
- **Aufruf von Programmen**
  Dies kann erforderlich sein bei besonders "sensiblen" Programmen, die z. B. nur zu bestimmten Zeiten oder Anlässen benutzt werden dürfen. Deshalb ist in diesen Fällen eine vollständige Protokollierung angezeigt. Die Protokollierung dient auch der Entlastung der befugten Benutzer (Nachweis des ausschließlich befugten Aufrufs der Programme).

Zweckbindung bei der Nutzung von Protokolldaten

Protokolldaten unterliegen aufgrund der nahezu übereinstimmenden Regelungen im Datenschutzrecht des Bundes und der Länder einer besonderen engen Zweckbindung. Sie dürfen nur zu den Zwecken genutzt werden, die Anlass für ihre Speicherung waren. Dies sind in der Regel die in einem Sicherheitskonzept festgelegten allgemeinen Kontrollen, die in den meisten Datenschutzgesetzen geforderte Überwachung der ordnungsgemäßen Anwendung der Datenverarbeitungsprogramme, mit denen personenbezogene Daten verarbeitet werden und die Kontrollen durch interne oder externe Datenschutzbeauftragte. Nur in Ausnahmefällen lassen die bereichsspezifischen Regelungen die Nutzung dieser Daten für andere Zwecke, z. B. zur Strafverfolgung, zu.

Aufbewahrungsdauer

Soweit nicht bereichsspezifische Regelungen etwas anderes vorsehen, richtet sich die Aufbewahrungsdauer der Protokolle nach den allgemeinen Löschungsregeln der Datenschutzgesetze. Protokolldaten sind unverzüglich zu

löschen, wenn sie zur Erreichung des Zwecks nicht mehr erforderlich sind. Gibt es keinen zwingenden Grund für das weitere Vorhalten von Protokolldateien, besteht eine Löschungspflicht.

Als Anhaltspunkte können dienen:

- die Wahrscheinlichkeit, dass Unregelmäßigkeiten (noch) offenbar werden können und
- die Möglichkeit, die Gründe von Unregelmäßigkeiten anhand der Protokolle und anderer Unterlagen aufdecken zu können.

Erfahrungsgemäß sollte eine Frist von einem Jahr nicht überschritten werden.

Soweit Protokolle zum Zwecke gezielter Kontrollen angefertigt werden, kommen kürzere Speicherungsfristen in Betracht. In der Regel reicht eine Aufbewahrung bis zur tatsächlichen Kontrolle aus. Auch hier sind die bereichsspezifischen Vorschriften zu beachten.

Technische und organisatorische Rahmenbedingungen

Die Effektivität der Protokollierung und ihre Auswertung im Rahmen von Kontrollen hängt im entscheidenden Maße von den technischen und organisatorischen Rahmenbedingungen ab. In diesem Zusammenhang sollten folgende Aspekte Berücksichtigung finden:

- Es sollte ein Konzept erstellt werden, das den Zweck der Protokolle und deren Kontrollen sowie Schutzmechanismen für die Rechte der Mitarbeiter und der sonstigen betroffenen Personen klar definiert (siehe auch B 5.22 *Protokollierung*).
- Die Zwangsläufigkeit und damit die Vollständigkeit der Protokolle muss ebenso gewährleistet werden wie die Manipulationssicherheit der Einträge in Protokolldateien.
- Entsprechend der Zweckbindung der Datenbestände müssen wirksame Zugriffsbeschränkungen realisiert werden.
- Die Protokolle müssen so gestaltet sein, dass eine effektive Überprüfung möglich ist. Dazu gehört auch eine IT-Unterstützung der Auswertung.
- Die Auswertungsmöglichkeiten sollten vorab abgestimmt und festgelegt sein.
- Kontrollen sollten so zeitnah durchgeführt werden, dass bei aufgedeckten Verstößen noch Schäden abgewendet sowie Konsequenzen gezogen werden können. Kontrollen müssen rechtzeitig vor dem Ablauf von Löschungsfristen von Protokolldateien stattfinden.
- Kontrollen sollten nach dem Vier-Augen-Prinzip erfolgen.
- Die Mitarbeiter sollten darüber informiert sein, dass Kontrollen durchgeführt werden, ggf. auch unangekündigt.
- Für Routinekontrollen sollten automatisierte Verfahren (z. B. watch dogs) verwendet werden.
- Personal- bzw. Betriebsräte sollten bei der Erarbeitung des Protokollierungskonzeptes und bei der Festlegung der Auswertungsmöglichkeiten der Protokolle beteiligt werden.

Prüffragen:

- Wurde ein Konzept erstellt, das den Zweck der Protokollierung, deren Kontrollen sowie Schutzmechanismen für die Rechte der betroffenen Personen beschreibt?
- Wird die Zweckbindung der Protokolldaten beachtet, insbesondere bei den Zugriffsregelungen?
- Lässt die Form der Protokollierung effektive Auswertungsmöglichkeiten zu?

- Wurden die Auswertungsmöglichkeiten mit dem Datenschutzbeauftragten und der Personalvertretung abgestimmt?

M 2.273 Zeitnahes Einspielen sicherheitsrelevanter Patches und Updates

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator, IT-Sicherheitsbeauftragter

Häufig werden Fehler in Produkten bekannt, die dazu führen können, dass die Informationssicherheit des Informationsverbundes, wo diese betrieben werden, beeinträchtigt wird. Entsprechende Fehler können Hardware, Firmware, Betriebssysteme und Anwendungen betreffen. Diese Schwachstellen müssen so schnell wie möglich behoben werden, damit sie nicht durch interne oder externe Angreifer ausgenutzt werden können. Dies ist ganz besonders wichtig, wenn die betreffenden Systeme mit dem Internet verbunden sind. Die Hersteller von Betriebssystem- oder Software-Komponenten veröffentlichen in der Regel Patches oder Updates, die auf dem jeweiligen IT-System installiert werden müssen, um den oder die Fehler zu beheben.

Die Systemadministratoren sollten sich daher regelmäßig über bekannt gewordene Schwachstellen informieren (siehe auch M 2.35 *Informationsbeschaffung über Sicherheitslücken des Systems*).

Wichtig ist, dass Patches und Updates, wie jede andere Software, nur aus vertrauenswürdigen Quellen bezogen werden dürfen. Für jedes eingesetzte System oder Softwareprodukt muss bekannt sein, wo Sicherheitsupdates und Patches erhältlich sind. Außerdem ist es wichtig, dass Integrität und Authentizität der bereits installierten Produkte oder der einzuspielenden Sicherheitsupdates und Patches überprüft werden (siehe M 4.177 *Sicherstellung der Integrität und Authentizität von Softwarepaketen*), bevor ein Update oder Patch installiert wird. Vor der Installation sollten sie außerdem mit Hilfe eines Computer-Virenschutzprogramms geprüft werden. Dies sollte auch bei solchen Paketen gemacht werden, deren Integrität und Authentizität verifiziert wurde.

Sicherheitsupdates oder Patches dürfen jedoch nicht voreilig eingespielt werden, sondern müssen vor dem Einspielen getestet werden. Falls sich ein Konflikt mit anderen kritischen Komponenten oder Programmen herausstellt, kann ein solches Update sonst zu einem Ausfall des Systems führen. Nötigenfalls muss ein betroffenes System so lange durch andere Maßnahmen geschützt werden, bis die Tests abgeschlossen sind.

Vor der Installation eines Updates oder Patches sollte stets eine Datensicherung des Systems erstellt werden, die es ermöglicht, den Originalzustand wieder herzustellen, falls Probleme auftreten. Dies gilt insbesondere dann, wenn ausführliche Tests aus Zeitgründen oder mangels eines geeigneten Testsystems nicht durchgeführt werden können.

In jedem Fall muss dokumentiert werden, wann, von wem und aus welchem Anlass Patches und Updates eingespielt wurden (siehe auch M 2.34 *Dokumentation der Veränderungen an einem bestehenden System*). Aus der Dokumentation muss sich der aktuelle Patchlevel des Systems jederzeit schnell ermitteln lassen, um beim Bekanntwerden von Schwachstellen schnell Klarheit darüber zu erhalten, ob das System dadurch gefährdet ist.

Falls festgestellt wird, dass ein Sicherheitsupdate oder Patch mit einer anderen wichtigen Komponente oder einem Programm inkompatibel ist oder Probleme verursacht, so muss sorgfältig überlegt werden, wie weiter vorgegan-

gen wird. Wird entschieden, dass auf Grund der aufgetretenen Probleme ein Patch nicht installiert wird, so ist diese Entscheidung auf jeden Fall zu dokumentieren. Außerdem muss in diesem Fall klar beschrieben sein, welche Maßnahmen ersatzweise ergriffen wurden, um ein Ausnutzen der Schwachstelle zu verhindern. Eine solche Entscheidung darf nicht von den Administratoren alleine getroffen werden, sondern sie muss mit den Vorgesetzten und dem IT-Sicherheitsbeauftragten abgestimmt sein.

Prüffragen:

- Sind Regelungen für das Patchmanagement definiert?
- Werden Software-Updates und Patches ausschließlich aus vertrauenswürdigen Quellen bezogen?
- Werden Software-Updates und Patches vor dem Roll-Out getestet?
- Ist sichergestellt, dass bei einem fehlgeschlagenem Update der ursprüngliche Systemzustand wieder hergestellt werden kann?
- Wird die Entscheidung, einen Patch aufgrund aufgetretener Probleme nicht zu installieren, dokumentiert?

M 2.363 Schutz gegen SQL-Injection

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator, Anwendungsentwickler

Um die Ausnutzung von SQL-Injections (siehe G 5.131 *SQL-Injection*) zu verhindern oder zumindest zu erschweren, sind eine Reihe von Maßnahmen zu ergreifen. Diese erstrecken sich über alle Komponenten einer Anwendung, von der Applikation selbst über den Server bis hin zum Datenbank-Managementsystem (DBMS).

Maßnahmen bei der Programmierung von Applikationen

Eine der wichtigsten Maßnahmen zur Vermeidung von SQL-Injection ist die sorgfältige Überprüfung und Filterung von Eingaben und Parametern durch die Applikation. Überprüft werden sollte, ob die übergebenen Daten dem erwarteten Datentyp entsprechen. Wird z. B. ein numerischer Parameter erwartet, kann man diesen in PHP ("PHP: Hypertext Preprocessor") mit der Funktion *is_numeric()* prüfen. Die Filterung hingegen muss dafür sorgen, dass Sonderzeichen wie das Quote-Zeichen ('), das Semikolon (;) und doppelte Bindestriche (--) ignoriert werden.

Sicherer ist der Einsatz von *Stored Procedures* beziehungsweise *Prepared SQL-Statements*. Diese werden von vielen Datenbank-Managementsystemen (DBMS) angeboten und sind ursprünglich dazu gedacht, häufiger auftretende Abfragen zu optimieren. Der Vorteil dieser parametrisierten Statements ist, dass Parameter nicht mehr direkt in ein SQL-Statement eingebunden werden. Vielmehr werden diese getrennt vom SQL-Statement separat an die Datenbank übergeben. Das Zusammenführen von Statement und Parametern erfolgt durch das DBMS selbst, wobei die oben genannten Sonderzeichen **automatisch** maskiert werden.

Um potentiellen Angreifern keine Anhaltspunkte für Angriffe zu liefern, sollte besonderes Augenmerk darauf gelegt werden, dass Applikationen möglichst keine Fehlermeldungen nach außen ausgeben, die Rückschlüsse auf das verwendete System oder auf die Struktur der dahinterliegenden Datenbank zulassen.

Serverseitige Maßnahmen

Die wichtigste Sicherheitsmaßname auf dem Server ist das Härten des Betriebssystems. Um so wenig Angriffspunkte wie möglich zu bieten, werden dabei Maßnahmen ergriffen wie:

- das Deaktivieren nicht benötigter Dienste,
- das Löschen nicht benötigter Benutzerkonten,
- das Einspielen relevanter Patches und
- das Löschen aller für die Funktion des Servers unnötigen Bestandteile.

Darüber hinaus sollte der Einsatz eines Application-Level-Gateways (ALG) (siehe M 5.117 *Integration eines Datenbank-Servers in ein Sicherheitsgateway*) erwogen werden. ALGs können auf Applikationsebene die Daten überwachen, die zwischen Webbrowser und Anwendung ausgetauscht werden, und verhindern, dass schädliche Daten den Server erreichen.

Eine weitere zusätzliche Sicherheitsmaßnahme stellt der Einsatz von Intrusion-Detection-Systemen (IDS) und Intrusion-Prevention-Systemen (IPS) dar. IDS analysieren den über ein Netz übertragenen Datenverkehr und erkennen potentiell gefährliche Daten. Die dazu eingesetzten Analysetechniken unter-

teilen sich in *Misuse* und *Anomaly Detection*. Die Misuse Detection versucht, bereits bekannte Angriffsmuster zu erkennen. Die Anomaly Detection verfolgt den Ansatz, die zulässigen Verhaltensmuster zu lernen und Abweichungen davon als Angriff zu identifizieren. Während ein IDS in der Lage ist, Angriffe zu erkennen und Warnungen auszugeben, ist ein IPS in der Lage, entsprechende Reaktionen auszuführen. Die Reaktion kann beispielsweise darin bestehen, die Verbindung zu blockieren, Daten zu verwerfen oder zu ändern.

Bei erhöhten Sicherheitsanforderungen sollte geprüft werden, ob der Einsatz von IDS beziehungsweise IPS zweckmäßig ist.

Datenbankseitige Maßnahmen

Ebenso wie beim Betriebssystem sollte auch eine Härtung der Datenbank erfolgen. Im Falle der Datenbank bedeutet dies z. B.:

- das Entfernen nicht benötigter Stored Procedures,
- das Deaktivieren nicht benötigter Dienste,
- das Löschen nicht benötigter Benutzerkonten und Default Accounts und
- das Einspielen relevanter Patches.

In diesem Zusammenhang sollte auch ein speziell für den Datenbankzugriff vorgesehener Account angelegt werden, der mit möglichst eingeschränkten Zugriffsrechten auskommen sollte.

Darüber hinaus sollten sensitive Daten, wie z. B. Passwörter, in der Datenbank soweit möglich nur verschlüsselt gespeichert werden.

Von vielen Herstellern werden mittlerweile sogenannte Schwachstellen-Scanner angeboten, die sowohl Applikationen als auch Datenbanken auf Sicherheitslücken, wie beispielsweise mögliche SQL-Injections, überprüfen können.

Beispiel für prinzipielles Vorgehen zur Erstellung von sicherem Code bei Verwendung von PHP und MySQL:

In PHP verhindert die Funktion mysql_real_escape_sting() die Übergabe von Sonderzeichen an eine MySQL-Datenbank. Die Funktion maskiert die in dem übergebenen String enthaltenen Sonderzeichen wie z. B. Quotes und verhindert so SQL-Injections.

Anstatt der folgenden Syntax:

```
$query = "SELECT * FROM users
WHERE username=
'" . $_POST['username'] . "'
AND password=
'" . $_POST['password'] . """;
```

sollte also diese Syntax verwendet werden:

```
$query = "SELECT * FROM users
WHERE username=
'" . mysql_real_escape_string($_POST['username']) . "'
AND password=
'" . mysql_real_escape_string($_POST['password']) . """;
```

Beispiel für sicheren Code bei Verwendung von ASP mit ADO und SQL-Server:

Die Verwendung eines prepared Statements für das obige Beispiel sieht in diesem Fall folgendermaßen aus:

```
$query = "SELECT * FROM users WHERE username=?
AND password=?"
Set cmd = Server.CreateObject("ADODB.Command")
cmd.CommandText = query
cmd.CommandType = adCmdText
Set param = cmd.CreateParameter("",adVarChar, adParamInput,
nMaxUsernameLength, strUsername)
cmd.Parameters.Append
Set param = cmd.CreateParameter("",adVarChar, adParamInput,
nMaxUsernameLength, strPassword)
cmd.Parameters.Append
Set rs = cmd.Execute()
```

Hierbei ist zu beachten, dass die oben aufgeführten Code-Beispiele nur den grundsätzlichen Ansatz zur Vermeidung von SQL-Injection veranschaulichen sollen.

Prüffragen:

- Werden Eingaben und Parameter durch die Applikation vor Weiterleitung an das Datenbanksystem sorgfältig überprüft und gefiltert?
- Werden Stored Procedures beziehungsweise Prepared SQL Statements eingesetzt?
- Ist gewährleistet, dass keine Fehlermeldungen nach außen ausgeben werden, welche Rückschlüsse auf das verwendete System oder auf die Struktur der dahinterliegenden Datenbank zulassen?

M 2.486 Dokumentation der Architektur von Webanwendungen und Web-Services

Verantwortlich für Initiierung: Leiter Entwicklung, Verantwortliche der einzelnen Anwendungen
Verantwortlich für Umsetzung: Administrator, Entwickler

Das Verständnis der Software-Architektur einer Webanwendung beziehungsweise eines Web-Service ist notwendig, um diese effizient und fehlerfrei zu warten, zu entwickeln und zu erweitern. Neben der systemspezifischen Dokumentation (siehe zum Beispiel M 2.25 *Dokumentation der Systemkonfiguration*, M 2.31 *Dokumentation der zugelassenen Benutzer und Rechteprofile* und M 2.34 *Dokumentation der Veränderungen an einem bestehenden System*) sind bei der Dokumentation von Webanwendungen und Web-Services einige Besonderheiten zu berücksichtigen.

Die Dokumentation muss alle Bestandteile berücksichtigen. Dabei sollten mindestens folgende Punkte durch die spezifische Dokumentation abgedeckt werden:

- Alle Abhängigkeiten (zum Beispiel zu Frameworks, Bibliotheken, Betriebssystemen, Hardware) und Schnittstellen (zum Beispiel zu Hintergrundsystemen) sollten dokumentiert werden. Bei Web-Services muss auch die Interaktion mit anderen Web-Services dokumentiert werden.
- Für den Betrieb notwendige Komponenten, die nicht Bestandteil der Webanwendung oder des Web-Service sind, müssen als solche gekennzeichnet werden (zum Beispiel Hintergrundsysteme wie eine Datenbank).
- Aus der Dokumentation muss hervorgehen, welche Komponenten Sicherheitsmechanismen umsetzen. Im Folgenden sind die Sicherheitsfunktionen von Webanwendungen und Web-Services aufgeführt, die mindestens berücksichtigt werden sollten:
  - Benutzermanagement,
  - Rollen- und Berechtigungskonzept,
  - Authentisierung,
  - Autorisierung,
  - Session-Management,
  - Protokollierung und
  - Transportsicherheit.
- Die Integration in eine gegebenenfalls bestehende Netzinfrastruktur muss in der Dokumentation behandelt werden. Hierbei ist die Maßnahme M 5.169 *Systemarchitektur einer Webanwendung* zu beachten.
- Die eingesetzten kryptographischen Funktionen und Verfahren müssen dokumentiert sein, siehe Baustein B 1.7 *Kryptokonzept*.

Die Dokumentation sollte während des Projektverlaufs aktualisiert und angepasst werden, sodass sie schon während der Entwicklungstätigkeit genutzt werden kann und Entscheidungsfindungen dokumentiert sind.

Prüffragen:

- Ist die Software-Architektur der Webanwendung beziehungsweise des Web-Service mit allen Bestandteilen und Abhängigkeiten dokumentiert?
- Werden für den Betrieb notwendige Komponenten, die nicht Bestandteil der Webanwendung beziehungsweise des Web-Service sind, als solche gekennzeichnet?

- Ist eine Zuordnung umgesetzter Sicherheitsmechanismen zu den Komponenten der Webanwendung beziehungsweise des Web-Service dokumentiert?
- Berücksichtigt die Dokumentation eine Integration der Webanwendung beziehungsweise des Web-Service in bestehende Netzinfrastruktur?
- Sind die eingesetzten kryptographischen Funktionen und Verfahren dokumentiert?
- Erfolgt die Dokumentation der Architektur einer Webanwendung beziehungsweise eines Web-Service bereits während der Entwicklungstätigkeit?

M 2.487 Entwicklung und Erweiterung von Anwendungen

Verantwortlich für Initiierung: Fachverantwortliche, Verantwortliche der einzelnen Anwendungen
Verantwortlich für Umsetzung: Entwickler, Beschaffer, Leiter Entwicklung, Tester

Für die effiziente Entwicklung von Anwendungen (auch Webanwendungen) sollten Regeln festgelegt und eingehalten werden. Das Ziel dabei ist, sowohl Konzeptions- als auch Programmierfehler bereits in der frühen Phase des Entwicklungs- und Erweiterungsprozesses zu vermeiden oder zumindest frühzeitig zu erkennen.

Die folgenden Punkte sollten daher bei der Entwicklung und Erweiterung von Anwendungen beachtet werden.

Entwicklung nach einem Vorgehensmodell

Es sollte nach einem geeigneten Vorgehensmodell (bzw. V-Modell XT, Wasserfallmodell, Spiralmodell) entwickelt werden. Dabei hat eine Anwendung vor der Inbetriebnahme alle Entwicklungsphasen des Vorgehensmodells zu durchlaufen.

Das verwendete Vorgehensmodell sollte mindestens die folgenden oder vergleichbare Phasen abdecken.

Anforderungsanalyse

Unternehmenssicherheitsrichtlinien und unternehmensspezifische Vorgaben sollten bei der Erhebung der Anforderungen an die Anwendung berücksichtigt und den Entwicklungsteams zur Verfügung gestellt werden (z. B. Erfüllung von Industrie-Standards wie PCI DSS oder Vorgaben zur Barrierefreiheit). Hierzu zählen z. B. auch Vorgaben und Anforderungen an die Verwendung kryptographischer Algorithmen und sicherer Programmierrichtlinien (siehe auch Abschnitt *Umsetzung von Programmierrichtlinien*).

In dieser Phase sollten alle von der Anwendung zu verarbeitenden Daten identifiziert und nach dem Schutzbedarf klassifiziert werden. Es müssen adäquate Schutzmechanismen der Anwendung festgelegt werden, welche die Daten gemäß ihrem Schutzbedarf schützen.

Konzeption und Design

Bei der Konzeption sollten die Architektur und der Aufbau der Anwendung festgelegt und dokumentiert werden. Hierbei sollte die Auswahl von Entwicklungstechniken (z. B. Programmiersprachen, Frameworks) miteinbezogen werden. Auch das Wissen und der Erfahrungsschatz der Entwickler sollten aus Kosten- und Sicherheitsgründen berücksichtigt werden.

Die Architektur sollte vorsehen, dass Komponenten (z. B. zur Autorisierung, Authentisierung) vorzugsweise in Modulen umgesetzt werden, die wiederverwendet werden können. Durch die zentrale Verfügbarkeit und Nutzung von Modulen können Redundanzen vermieden und die Pflege erleichtert werden.

Bei Client-/Server-Architekturen (z. B. Webanwendung) sollten die zentralen Sicherheitsmechanismen nach Möglichkeit mindestens serverseitig umgesetzt werden.

Es sollte darauf geachtet werden, dass Sicherheitsanforderungen vollständig durch Sicherheitsmechanismen erfüllt und zur Erstellung von Testfällen festgehalten werden.

Getroffene Entscheidungen sollten dokumentiert werden, sodass später eine effiziente Weiterentwicklung der Anwendung durch ausreichende Dokumentation gewährleistet ist.

Entwicklung

Bei der Umsetzung der Anwendung sollten Programmierrichtlinien (siehe auch Abschnitt *Umsetzung von Programmierrichtlinien*) für die sichere Entwicklung der Komponenten eingehalten werden.

Es sollte darauf geachtet werden, dass die Dokumentation während der Entwicklungstätigkeit fortgeführt wird (z. B. durch Kommentare im Quelltext und Werkzeuge zur Generierung der Dokumentation). Somit ist der Quelltext zu einem späteren Zeitpunkt auch für Dritte nachvollziehbar.

Zum Schutz vor dem Verlust bereits entwickelter und verworfener Lösungen sowie zu Dokumentationszwecken sollte die Historie der Änderungen festgehalten werden (z. B. durch ein Revisionssystem).

Tests

Testfälle sollten nicht nur die Geschäftsfunktionen, sondern ebenfalls die Sicherheitsfunktionalität berücksichtigen. Dazu zählen z. B. Sicherheitskomponenten wie Autorisierungs-, Authentisierungs- und Filterkomponenten. Nach Möglichkeit sollten Penetrationstests und (für hohen Schutzbedarf) auch Source-Code-Analysen durchgeführt werden, um die umgesetzten Sicherheitsmechanismen zu kontrollieren (M 5.150 *Durchführung von Penetrationstests*).

Vor der Inbetriebnahme einer Anwendung sollte nicht nur die Funktionstüchtigkeit, sondern auch ein möglicher Missbrauch der angebotenen Funktionalität geprüft werden. Dies kann durch Penetrationstests erreicht werden. Damit ein Vier-Augen-Prinzip beim Testen umgesetzt wird, sollten die Tests nicht von den Personen durchgeführt werden, die zuvor an der Konzeption oder der Entwicklung der Anwendung beteiligt waren.

Bei den Tests ist darauf zu achten, dass diese nur mit Testdaten und nicht mit Live-Daten bzw. Kundendaten durchgeführt werden.

Bei Webanwendungen sollten die Webseiten auf Konformität zu dem verwendeten Standard (z. B. HTML-Standard) getestet werden. Dadurch können unvorgesehene Seiteneffekte aufgrund einer Fehlinterpretation seitens der Browser vermieden werden. Eine Überprüfung mit verschiedenen Browsern kann hier sehr hilfreich sein.

Bei der Planung und Durchführung der Tests sollte die Maßnahme M 2.62 *Software-Abnahme- und Freigabe-Verfahren* berücksichtigt werden.

Integration und Softwareverteilung (Deployment)

Vor der produktiven Inbetriebnahme sind die Anwendungen und gegebenenfalls notwendige Hintergrundsysteme sicher zu konfigurieren. Hierbei sollte

die Anbindung möglicher Hintergrundsysteme (z. B. Identitätsspeicher, Datenbanken) an die Anwendung berücksichtigt werden. Vor der Inbetriebnahme der Anwendung ist ebenfalls sicherzustellen, dass der Transportkanal geschützt ist.

Sensible Daten der Anwendung sind häufig in Hintergrundsystemen hinterlegt. Daher sollte das Sicherheitsniveau der Anwendung und möglicher Hintergrundsysteme einheitlich sein. Der Zugriff auf die Hintergrundsysteme sollte Benutzern lediglich über die definierten Schnittstellen der Anwendung möglich sein.

Darüber hinaus sollte sichergestellt werden, dass die Daten bei der Verteilung der Anwendung nicht durch Dritte manipuliert werden können.

Wartung

Es muss ein Prozess zur Pflege der Anwendung definiert werden, der auch die regelmäßige Prüfung der Sicherheit der Anwendung auf Schwachstellen bzw. verfügbare Patches berücksichtigt.

Wird die Anwendung angepasst oder erweitert, muss darauf geachtet werden, dass die Wirksamkeit der Sicherheitsmechanismen nicht beeinträchtigt wird. Zusätzlich sollte durch Tests in einer gesonderten Testumgebung die Wirksamkeit der Sicherheitsmechanismen erneut überprüft werden.

Umsetzung von Programmierrichtlinien

Eine Programmierrichtlinie hilft, einen einheitlichen Programmierstil zu definieren und ein einheitliches Sicherheitsniveau zu etablieren (z. B. durch die Verwendung von Sicherheitsbibliotheken). Die Qualität und Verständlichkeit des Quelltexts kann hierdurch verbessert und nachvollziehbarer werden. In der Folge können Fehler und Schwachstellen einfacher gefunden und eine spätere Erweiterung der Anwendung kosteneffektiv umgesetzt werden.

Programmierrichtlinien sollten nicht nur bei der Entwicklung im eigenen Haus, sondern auch beim Outsourcing der Entwicklungstätigkeit umgesetzt werden.

Zukunftssichere Entwicklung von Sicherheitsmechanismen

Wenn Sicherheitsmechanismen entworfen und entwickelt werden, sollten hierbei auch zukünftige Entwicklungen von Angriffstechniken als auch Standards (z. B. neuer HTML-Standard) berücksichtigt werden. So sollte beispielsweise eine Filterkomponente, die als schadhaft klassifizierte <script>-Tags filtert, ebenso unbekannte Tags filtern. Unbekannte Tags können gegebenenfalls zukünftig verwendet werden (z. B. mit der Einführung eines neuen HTML-Standards), um Sicherheitsmechanismen der Webanwendung zu umgehen.

Produktspezifische Sicherheitsfunktionalität

Falls eine Webanwendung ausschließlich mit einem spezifischen Browser (u. U. nur eines Herstellers) genutzt wird, so sollte der Einsatz von produktspezifischen Sicherheitsfunktionen des Browsers berücksichtigt werden.

Outsourcing der Anwendungsentwicklung

Im Fall von Outsourcing muss sichergestellt werden, dass das beauftragte Dritt-Unternehmen die nötigen Sicherheitsanforderungen bei der Umsetzung der Anwendung erfüllt. Dies kann beispielsweise durch die Vorgabe eines Vorgehensmodells oder durch Programmierrichtlinien erreicht werden.

Wird für die Entwicklung einer Anwendung mit hohem Schutzbedarf ein Dienstleister beauftragt, sollte der Quelltext (z. B. das Projektarchiv) unter der administrativen Kontrolle des Auftraggebers stehen. Dabei sollte der Auftraggeber jederzeit auf den Quelltext der Anwendung zugreifen und Änderungen am Quelltext nachvollziehen können.

Festlegung der Entwicklungsumgebung

Die Produktiv-, Test- und Entwicklungsumgebungen sind auf getrennten Systemen zu betreiben. In den Umgebungen sollten unterschiedliche Zugangsdaten gewählt werden. Testkonten sollten hierbei, soweit möglich, keine privilegierten Rechte erhalten. Grundsätzlich dürfen erfolgreiche Angriffe auf die Entwicklungs- oder Testumgebung keine Auswirkungen auf die Produktivumgebung haben.

Prüffragen:

- Wird die Anwendung nach einem geeigneten Vorgehensmodell entwickelt?
- Werden alle Phasen bei der Entwicklung von Anwendungen durch das Vorgehensmodell abgedeckt und vor der Inbetriebnahme vollständig durchlaufen?
- Werden Programmierrichtlinien für die Entwicklung von Anwendungen vorgegeben?
- Werden bei dem Entwurf und der Entwicklung von Sicherheitsmechanismen bei Anwendungen auch zukünftige Standards und Angriffstechniken berücksichtigt?
- Werden bei der Anwendungsentwicklung die Entwicklungs-, Test- und Produktivsysteme voneinander getrennt?
- Werden für die Anwendung Penetrationstests durchgeführt, bei denen die Anwendungslogik geprüft wird?
- Werden die Penetrationstests bei Anwendungen nach einem Vier-Augen-Prinzip durchgeführt?

M 2.488 Web-Tracking

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator

Web-Tracking bezeichnet die Auswertung von Nutzerdaten z. B. zur Verfolgung der Aktivitäten von Benutzern eines Webauftritts. Auf Grundlage dieser Auswertungen kann beispielsweise auf den Benutzer zugeschnittene Werbung eingeblendet oder die Popularität von Beiträgen anhand von Statistiken gemessen und daraufhin der Webauftritt optimiert werden. Hierfür können personenbezogene Informationen wie der Standort des Benutzers, der Status einer Transaktion (z. B. Geschäftsabschluss bei einer Einkaufsplattform) und eine Abrufstatistik über Webseiten protokolliert und herangezogen werden.

Werden externe Dienstleister beauftragt diese Nutzerdaten auszuwerten, ist zu beachten, dass diese Dienstleister möglicherweise die Nutzerdaten mit den Daten anderer Kunden und Webanwendungen korrelieren können. Auf dieser Basis können anwendungsübergreifend detaillierte Benutzerprofile erstellt werden.

Mögliche Techniken zur Sammlung von Nutzerdaten sind z. B.

- (persistente) Cookies,
- Web-Bugs (Ein-Pixel große Bildelemente z. B. in einer E-Mail zur Führung einer Abrufstatistik),
- Browser-Fingerabdrücke (z. B. durch Attribute wie installierte Zusatzprogramme, Bildschirm-Auflösung, Zeitzone, User-Agent, HTTP-Header),
- Protokollierung der IP-Adresse.

Die Techniken können darüber hinaus kombiniert werden, um Benutzer zuverlässiger zu identifizieren.

Falls eine Auswertung von Nutzerdaten, insbesondere personenbeziehbare Daten, vorgesehen ist oder ein Anbieter diesen Dienst übernehmen soll, müssen die rechtlichen Grundlagen geprüft werden.

M 3.5 Schulung zu Sicherheitsmaßnahmen

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Vorgesetzte
Verantwortlich für Umsetzung: IT-Sicherheitsbeauftragter, Vorgesetzte

Wie sich an vielen konkreten Beispielen wie den Schadensstatistiken von Elektronik-Versicherern belegen lässt, resultieren Schäden oft schlicht aus der Unkenntnis elementarer Sicherheitsmaßnahmen. Um dies zu verhindern, ist jeder einzelne Mitarbeiter zum sorgfältigen Umgang mit geschäftsrelevanten Informationen und der IT zu schulen und zu motivieren. Nur durch die Vermittlung der notwendigen Kenntnisse kann ein Verständnis für die erforderlichen Maßnahmen zur Informationssicherheit geweckt werden.

Im Folgenden werden die Kernthemen, die bei einer Schulung zu Sicherheitsmaßnahmen vermittelt werden sollten, vorgestellt. Eine ausführliche und zielgruppengerichtete Beschreibung von Schulungsinhalten findet sich in M 3.45 *Planung von Schulungsinhalten zur Informationssicherheit*.

- **Sensibilisierung für Informationssicherheit**
  Jeder Mitarbeiter ist auf die Bedeutung der Sicherheitsbelange hinzuweisen. Ein geeigneter Einstieg in die Sensibilisierung ist es beispielsweise, die Abhängigkeit der Behörde bzw. des Unternehmens und damit der Arbeitsplätze vom reibungslosen Funktionieren der Geschäftsprozesse aufzuzeigen. Darüber hinaus ist der Wert von Informationen unter den Gesichtspunkten Vertraulichkeit, Integrität und Verfügbarkeit herauszuarbeiten. Diese Sensibilisierungsmaßnahmen sind in regelmäßigen Zeitabständen zu wiederholen.
- **Mitarbeiterbezogene Informationssicherheitsmaßnahmen**
  Zu diesem Thema sollen die Sicherheitsmaßnahmen vermittelt werden, die in einem Informationssicherheitskonzept erarbeitet wurden und von den einzelnen Mitarbeitern umzusetzen sind. Je nach Geschäftsprozess oder Fachaufgabe kann es andere Werte geben, die zu schützen sind, oder einen anderen Schutzbedarf haben. Den Mitarbeitern sollte vermittelt werden, welche Bedeutung Informationen oder andere Objekte für die Institution haben und was sie beim Umgang mit diesen beachten sollten. Dieser Teil der Schulungsmaßnahmen hat eine große Bedeutung, da viele Sicherheitsmaßnahmen erst nach einer entsprechenden Schulung und Motivation effektiv umgesetzt werden können.
- **Produktbezogene Sicherheitsmaßnahmen**
  Zu diesem Thema sollen die Sicherheitsmaßnahmen vermittelt werden, die inhärent mit einem Produkt wie beispielsweise einem IT-System verbunden sind und häufig bereits im Lieferumfang enthalten sind. Dies können neben Passwörtern zur Anmeldung auch Möglichkeiten zur Verschlüsselung von Dokumenten oder Datenfeldern sein. So können beispielsweise Hinweise und Empfehlungen über die Strukturierung und Organisation von Dateien den Aufwand zur Datensicherung deutlich reduzieren.
- **Verhalten bei Auftreten von Schadsoftware**
  Hier soll den Mitarbeitern vermittelt werden, wie mit Computer-Viren oder anderer Schadsoftware umzugehen ist. Mögliche Inhalte dieser Schulung sind (siehe M 6.23 *Verhaltensregeln bei Auftreten von Schadprogrammen*):

  - Erkennen einer Schadsoftware-Infektion
  - Wirkungsweise und Arten von Schadsoftware
  - Sofortmaßnahmen im Verdachtsfall

  - Maßnahmen zur Eliminierung von Schadsoftware
  - Vorbeugende Maßnahmen
- **Authentikation**
  Mitarbeiter sollten mit den vorhandenen Authentikationsmechanismen und den hierfür genutzten Authentikationsmitteln (z. B. Passwörtern oder Token) korrekt umgehen gehen können. Beispielsweise sollen die Bedeutung von Passwörtern für die Informationssicherheit sowie die Randbedingungen erläutert werden, die einen wirksamen Einsatz eines Passwortes erst ermöglichen (siehe auch M 2.11 *Regelung des Passwortgebrauchs*).
- **Bedeutung der Datensicherung und deren Durchführung**
  Die regelmäßige Datensicherung ist eine der wichtigsten Sicherheitsmaßnahmen in jedem Informationsverbund. Vermittelt werden soll das Datensicherungskonzept (siehe Baustein B 1.4 *Datensicherungskonzept*) der Behörde bzw. des Unternehmens und die von jedem einzelnen durchzuführenden Datensicherungsaufgaben. Besonders wichtig ist dies für solche Bereiche, in denen Benutzer selbst die Datensicherungen durchführen müssen.
- **Umgang mit personenbezogenen Daten**
  An den Umgang mit personenbezogenen Daten sind besondere Anforderungen zu stellen. Mitarbeiter, die mit personenbezogenen Daten arbeiten, sind für die gesetzlich erforderlichen Sicherheitsmaßnahmen zu schulen. Dies betrifft beispielsweise den Umgang mit Auskunftsersuchen, Änderungs- und Verbesserungswünschen der Betroffenen, gesetzlich vorgeschriebene Fristen zur Datenlöschung, Schutz der Vertraulichkeit und die Übermittlung der Daten.
- **Einweisung in Notfallmaßnahmen**
  Sämtliche Mitarbeiter sind in bestehende Notfallmaßnahmen einzuweisen. Dazu gehört die Erläuterung der Fluchtwege, die Verhaltensweisen bei Feuer oder anderen Notfällen, der Umgang mit Feuerlöschern und das Notfall-Meldesystem (wer als erstes wie zu benachrichtigen ist).
- **Vorbeugung gegen Social Engineering**
  Die Mitarbeiter sollen auf die Gefahren des Social Engineering hingewiesen werden. Die typischen Muster solcher Versuche, über gezieltes Aushorchen an vertrauliche Informationen zu gelangen, ebenso wie die Methoden, sich dagegen zu schützen, sollten erläutert werden. Da Social Engineering oft mit der Vorspiegelung einer falschen Identität einhergeht, sollten Mitarbeiter regelmäßig darauf hingewiesen werden, die Identität von Gesprächspartnern zu überprüfen und insbesondere am Telefon keine vertraulichen Informationen weiterzugeben.

Bei der Durchführung von Schulungen sollte immer beachtet werden, dass es nicht reicht, einen Mitarbeiter einmal während seines gesamten Arbeitsverhältnisses zu schulen. Für nahezu alle Formen von Schulungen - insbesondere Front-Desk-Schulungen - gilt, dass sehr viele neue Informationen auf die Teilnehmer einstürzen. Diese gelangen nur zu einem kleinen Teil ins Langzeitgedächtnis, 80% des vermittelten Wissens sind meist schon bei Schulungsende wieder vergessen.

Daher sollten Mitarbeiter immer wieder zu Themen rund um die Informationssicherheit geschult bzw. sensibilisiert werden. Dies kann beispielsweise

- in kürzeren Veranstaltungen zu aktuellen Sicherheitsthemen,
- im Rahmen regelmäßiger Veranstaltungen wie Abteilungsbesprechungen, oder
- durch interaktive Schulungsprogramme, die allen Mitarbeitern zur Verfügung stehen, erfolgen.

Prüffragen:

- Werden die Mitarbeiter zu Themen rund um die Informationssicherheitsmaßnahmen geschult?
- Wird den Mitarbeitern vermittelt, welche Bedeutung Informationen oder andere Objekte haben und was sie beim Umgang mit diesen beachten sollten?
- Werden Mitarbeiter, die mit personenbezogenen Daten arbeiten, für die gesetzlich erforderlichen Sicherheitsmaßnahmen geschult?
- Werden die Mitarbeiter regelmäßig zu Themen der Informationssicherheit geschult bzw. sensibilisiert?

M 4.78 Sorgfältige Durchführung von Konfigurationsänderungen

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator

Die Durchführung von Änderungen an einem IT-System im Echtbetrieb ist immer als kritisch einzustufen und entsprechend sorgfältig muss hierbei vorgegangen werden.

Bevor mit Änderungen am System begonnen wird, muss als erstes die alte Konfiguration gesichert werden, sodass sie schnell verfügbar ist, wenn Probleme mit der neuen Konfiguration auftreten.

Bei vernetzten IT-Systemen müssen die Benutzer rechtzeitig über die Durchführung von Wartungsarbeiten in geeigneter Weise, wie z. B. durch einen Eintrag im Intranet oder per E-Mail, informiert werden, damit sie zum einen ihre Planung auf eine zeitweise Systemabschaltung einrichten können, und zum anderen nach Änderungen auftretende Probleme richtig zuordnen können.

Die Konfigurationsänderungen sollten immer nur schrittweise durchgeführt werden. Zwischendurch sollte immer wieder überprüft werden, ob die Änderungen korrekt durchgeführt wurden und das IT-System sowie die betroffenen Applikationen noch lauffähig sind.

Bei Änderungen an Systemdateien ist anschließend ein Neustart durchzuführen, um zu überprüfen, ob sich das IT-System korrekt starten lässt. Für Problemfälle sind alle für einen Notstart benötigten Datenträger vorrätig zu halten, z. B. Boot-Medien, Start-CD-ROM.

Vor Konfigurationsänderungen sollten von allen eventuell betroffenen Dateien und Verzeichnissen Datensicherungen angefertigt werden. Komplexere Konfigurationsänderungen sollten möglichst nicht in den Originaldateien vorgenommen werden, sondern in Kopien. Alle durchgeführten Änderungen sollten nach dem Vier-Augen-Prinzip überprüft werden, bevor sie in den Echtbetrieb übernommen werden.

Bei IT-Systemen mit hohen Verfügbarkeitsanforderungen ist auf Ersatzsysteme zurückzugreifen bzw. zumindest ein eingeschränkter IT-Betrieb zu gewährleisten. Das Vorgehen kann sich dabei idealerweise nach dem Notfall-Handbuch richten.

Die durchgeführten Konfigurationsänderungen sollten Schritt für Schritt notiert werden, so dass bei auftretenden Problemen das IT-System durch sukzessive Rücknahme der Änderungen wieder in einen lauffähigen Zustand gebracht werden kann (siehe auch M 2.34 *Dokumentation der Veränderungen an einem bestehenden System*).

Prüffragen:

- Werden die Benutzer über Wartungsarbeiten in geeigneter Weise informiert?
- Werden von allen Dateien, an denen Änderungen vorgenommen werden müssen, Datensicherungen angelegt?

M 4.176 Auswahl einer Authentisierungsmethode für Webangebote

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator

Für E-Commerce- und E-Government-Anwendungen, personalisierte Webangebote oder nur allgemein zur Realisierung von Zugriffsbeschränkungen auf bestimmte Bereiche eines Webangebots werden Mechanismen zur Identifikation und Authentisierung verschiedener Benutzer benötigt.

In Abhängigkeit von den konkreten Anforderungen an den Schutz der Informationen vor unbefugtem Zugriff und die Qualität der Authentisierung muss eine geeignete Methode ausgewählt werden. Die Wahl der Authentisierungsmethode und die Gründe, die zu der Wahl geführt haben, sollten dokumentiert werden.

Authentisierungsmethoden bei HTTP

Das Protokoll HTTP/1.1 sieht zwei verschiedene Methoden zur Benutzerauthentisierung vor.

Die erste Methode ist die so genannte *Basic-Access-Authentisierung*. Dabei sendet der Client den Benutzernamen und das Passwort *Base64*-kodiert im so genannten *Authorization Header* des HTTP-Requests an den Server. *Base64* ist eine Methode zur Kodierung von Binärdaten in 7-Bit ASCII, die hier zur Übertragung von Sonderzeichen über die HTTP-Schnittstelle genutzt wird. Das Passwort ist somit zwar nicht auf den ersten Blick ablesbar, kann aber von einem potentiellen "Lauscher" problemlos ermittelt werden, da es unverschlüsselt ist. Daher ist dieser Authentisierungstyp allenfalls für sehr geringe Vertraulichkeitsanforderungen zu gebrauchen.

Die zweite Methode zur HTTP-Authentisierung ist die *Digest-Authentisierung*. Bei dieser Art der Authentisierung muss auf dem Server das Passwort des Benutzers im Klartext vorliegen. Der Client erhält vom Server einen Zufallsstring, die so genannte Challenge. Aus dieser Challenge und dem Passwort des Benutzers errechnet der Client nach einem standardisierten Verfahren einen so genannten *Digest*, der dann zur Authentisierung an den Server gesandt wird. Da der Server sowohl über den von ihm generierten Zufallsstring, als auch über das Passwort des Benutzers verfügt, kann er den Digest ebenfalls berechnen und so die Authentisierung durchführen. Da bei der Digest-Authentisierung das Passwort nicht über das Netz verschickt wird, eignet sich diese Methode für einen etwas höheren Schutzbedarf.

Ein Problem bei der Verwendung der oben genannten Authentisierungsmethoden ist die Sicherheit der Passwortdaten auf dem Server: Bei Verwendung der Digest-Authentisierung müssen die Authentisierungsdaten der Benutzer auf dem Webserver im Klartext vorhanden sein. Bei Verwendung der Basic-Authentisierung wird meist ein Hash-Wert des Passwortes gespeichert. Eine Sicherung der Passwortdateien auf dem Server vor unbefugtem Zugriff ist daher besonders wichtig.

Neben der HTTP-Authentisierung existiert ein weiterer Weg, Zugriffskontrolle über das HTTP-Protokoll zu realisieren: die Authentisierung kann nicht über den Webserver selbst, sondern über eine serverseitige Anwendung durchge-

führt werden. Dabei werden Benutzername und Passwort über normale HTML Formulare eingegeben und von der Anwendung überprüft. Dieses Verfahren ist häufig bei Internet-Angeboten realisiert. Es sollte jedoch stets beachtet werden, dass Passwörter oder PINs, die im Klartext über das Internet übertragen werden, leicht mitgelesen werden können. Zudem werden natürlich auch sämtliche Daten, selbst wenn sie auf authentisierte Anfragen hin ausgeliefert werden, unverschlüsselt übermittelt.

Manche Webangebote identifizieren die Benutzer über spezielle Cookies, die im Browser gespeichert werden. Da Cookies bei der Verwendung von HTTP ebenfalls im Klartext übertragen werden, ist diese Methode für die Authentisierung beim Zugriff auf schutzbedürftige Informationen ebenfalls nicht geeignet. Da im Zusammenhang mit Cookies noch weitere Sicherheitsprobleme existieren, sollte diese Methode generell nicht verwendet werden.

Verwendung von SSL

Wenn im Rahmen von E-Government- oder E-Commerce-Angeboten höhere Anforderungen an die Sicherheit der Authentisierung und die Vertraulichkeit der übertragenen Daten bestehen, dann sollte die Übertragung durch die Verwendung von SSL abgesichert werden (siehe auch M 5.66 *Verwendung von SSL*).

Bei der Verwendung von SSL gibt es zwei verschiedene Betriebsarten: bei der ersten Variante besitzt nur der Server ein Zertifikat. Dies dient dem Benutzer dazu, zu erkennen, dass er wirklich mit dem "richtigen" Server verbunden ist, und ermöglicht nach dem Aufbau einer verschlüsselten Verbindung die sichere Übertragung von Authentisierungsinformationen und Anwendungsdaten.

Ein Server-Zertifikat enthält neben dem Namen der Zertifizierungsstelle auch den Namen des Servers, für den es gültig ist. Es kann von einer Wurzelzertifizierungsstelle (Root-CA) ausgestellt sein oder auch selbst erzeugt werden, beispielsweise mit den im OpenSSL Paket enthaltenen Tools.

Zertifikate, die nicht von einer Wurzelzertifizierungsstelle ausgestellt wurden, die dem Browser bekannt ist, werden vom Browser meist nicht ohne weiteres akzeptiert, sondern der Benutzer muss explizit bestätigen, dass das betreffende Zertifikat akzeptiert werden soll.

Bei der zweiten Variante, verfügt auch der Benutzer über ein Zertifikat, das auf dem Client-Rechner vorhanden sein muss, und das der Browser zur Authentisierung an den Server schickt. Voraussetzung dafür ist jedoch, dass die Zertifizierungsstellen, deren Zertifikate verwendet werden, vertrauenswürdig sind. Dass diese Art der Authentisierung in der Praxis nicht häufiger verwendet wird, liegt an dem Aufwand, der zur Umsetzung einer solchen Lösung erforderlich ist. Die serverseitige Konfiguration ist relativ einfach: Neben der Konfiguration des Webservers für SSL muss ein SSL-Server-Zertifikat beschafft und implementiert werden. Der Aufwand, der für jeden einzelnen Benutzer zu betreiben ist, ist jedoch relativ hoch: Jeder Benutzer muss über ein SSL-Client-Zertifikat verfügen, das jeweils im Browser des Benutzers installiert ist. Dies führt zu einer gewissen Einschränkung der Bequemlichkeit, da einer der großen Vorteile der normalen Webserver-Nutzung gerade darin besteht, dass der Zugriff von praktisch jedem beliebigen Rechner aus erfolgen kann. Werden Client-Zertifikate zur Authentisierung benutzt, so ist diese Flexibilität deutlich eingeschränkt, weil das Client-Zertifikat meist nicht überall vorhanden ist. Andererseits kann in bestimmten Situationen, etwa beim Einsatz eines Intranet-Webservers, genau dies erwünscht sein.

Eine häufig verwendete Methode der Benutzerauthentisierung für Webangebote ist die Kombination von formularbasierter Authentisierung und SSL-verschlüsselter Datenübertragung. Diese Methode bietet, wenn die gewählte SSL-Verschlüsselung ausreichend stark gewählt wird, bei vertretbarem Aufwand (Benutzerverwaltung in der Webanwendung und Implementierung eines SSL-geschützten Zugriffs auf den Webserver) ein Sicherheitsniveau, das auch für höhere Sicherheitsanforderungen angemessen ist.

In M 5.160 *Authentisierung gegenüber Webservern* werden die verschiedenen Möglichkeiten der Benutzerauthentisierung bei Webservern vorgestellt und in der folgenden Tabelle zusammengefasst:

Methode	**Sicherheits-niveau**	**Aufwand für Implementierung**	**Serveranforderungen**	**Kommentare**
Standard-Authentisierung	Niedrig	Niedrig	Benutzerverwaltung	Authentisierungsinformationen und Daten werden unverschlüsselt übertragen!
Formularbasierte Authentisierung ohne gesicherte Übertragung	Niedrig	Niedrig bis mittel	Implementierung in der jeweiligen Anwendung	Authentisierungsinformationen und Daten werden unverschlüsselt übertragen!
Digest-Authenti-sierung	Mittel	Niedrig	Benutzerverwaltung	Daten werden unverschlüsselt übertragen.
Formularbasierte Authentisierung über SSL	Hoch	Mittel bis hoch	SSL-Unterstützung im Server, Implementierung in der jeweiligen Anwendung	Authentisierungsinformationen und Daten werden verschlüsselt übertragen!
Zertifikatbasierte Authentisierung über SSL	Hoch bis sehr hoch	Hoch bis sehr hoch	Installation von Server-Zertifikaten. Zertifikatsverwaltung, Public-Key Infrastruktur.	Wird hauptsächlich für sichere Transaktionen über das Internet verwendet.

Tabelle: Benutzerauthentisierung bei Webservern

Der Microsoft Internet Information Server bietet darüber hinaus noch eine weitere Methode, bei der die Windows-Benutzeranmeldung benutzt wird. Diese Methode funktioniert allerdings nur mit dem Microsoft Internet Explorer als Client.

Beim Aufbau einer SSL-Verbindung wird der zu verwendende Verschlüsselungsmodus zwischen Client und Server ausgehandelt. Unter den zur Verfügung stehenden Algorithmen befinden sich auch solche, die nicht mehr als sicher angesehen werden können. Insbesondere gibt es auch den so genannten Null-Encryption-Modus, bei dem keine Verschlüsselung stattfindet. Bei der Konfiguration des Webservers für die Verwendung von SSL muss darauf geachtet werden, dass der Server keinen der schwachen Algorithmen und insbesondere nicht den Null-Encryption-Modus akzeptiert. Andernfalls könnte es dazu kommen, dass scheinbar eine sichere Verbindung aufgebaut wird (es wird https verwendet), die jedoch in Wirklichkeit zu schwach oder gar nicht verschlüsselt ist. Eine solche Situation könnte von einem Angreifer bewusst herbeigeführt werden, um Authentisierungsinformationen und andere Daten abzuhören. Daher sollte in der SSL-Konfiguration des Webservers die Verwendung des Null-Encryption-Modus und der schwachen Algorithmen abgeschaltet werden.

Prüffragen:

- Wurden/Sind die Authentisierungsmethode für Webangebote und die Gründe für deren Auswahl dokumentiert?
- Digest-Authentisierung: Werden Passwortdateien auf dem Webserver vor dem Zugriff Unbefugter geschützt?
- Bei hohen Anforderungen an die Vertraulichkeit: Wird die Authentisierung und die Übertragung von Daten bei Webangeboten durch SSL abgesichert?
- Bei SSL-Verwendung: Werden schwache kryptographische Algorithmen vom Webserver nicht akzeptiert?

M 4.392 Authentisierung bei Webanwendungen

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Entwickler, Administrator

Soll eine Webanwendung oder Teile davon ausschließlich von einem eingeschränkten Benutzerkreis genutzt werden können, so muss sich der Benutzer gegenüber der Anwendung authentisieren. Für die Authentisierung können unterschiedliche Methoden verwendet werden, die in den Maßnahmen M 4.176 *Auswahl einer Authentisierungsmethode für Webangebote* und M 5.160 *Authentisierung gegenüber Webservern* beschrieben sind.

Bei der Umsetzung von Authentisierungsmechanismen für Webanwendungen sind folgende Punkte zu berücksichtigen.

Anforderungen an die Authentisierungskomponente

Die Authentisierungslogik sollte nur an einer Stelle und nicht mehrfach im Programmcode realisiert werden. Treten während der Authentisierung Fehler auf, sollte die angeforderte Aktion abgebrochen und die Anfrage zurückgewiesen werden.

Die Authentisierungskomponente sollte das Erzwingen sicherer Passwörter gemäß einer Passwort-Richtlinie unterstützen. Anforderungen an sichere Passwörter können der Maßnahme M 2.11 *Regelung des Passwortgebrauchs* entnommen werden.

Darüber hinaus wird empfohlen, die geschätzte Stärke des eingegebenen Passworts einzublenden (z. B. schwach, mittel, sicher). Dadurch wird der Benutzer dabei unterstützt, sichere Passwörter zu wählen.

Um Fehler bei der Entwicklung der Authentisierungskomponente zu vermeiden, wird empfohlen die Authentisierungskomponente auf Basis etablierter Standardkomponenten (Bibliotheken oder Frameworks) umzusetzen (z. B. Enterprise Security API der OWASP).

Besteht ein erhöhter Schutzbedarf der Webanwendung, sollte eine Zwei-Faktor-Authentisierung eingesetzt werden.

Damit ein Benutzer den Missbrauch seines Benutzerkontos erkennen kann, kann die Webanwendung das Datum und die Uhrzeit der letzten erfolglosen und erfolgreichen Anmeldeversuche nach der Anmeldung eines Benutzers als Warnhinweis anzeigen.

Remember-Me-Funktion

Zur Steigerung der Benutzerfreundlichkeit werden die Authentisierungsdaten von Webanwendungen teilweise dauerhaft auf dem Client der Benutzer gespeichert (z. B. innerhalb eines Cookies im Web-Browser). Diese Möglichkeit wird häufig als Remember-Me-Funktion bezeichnet. Wurden Authentisierungsdaten im Rahmen der Remember-Me-Funktion auf dem Client gespeichert, werden diese bei einer späteren Nutzung der Webanwendung automatisch übertragen. Der Benutzer muss somit keine Zugangsdaten mehr eingeben.

Erhält ein Angreifer Zugriff auf den Web-Browser oder wird Schadcode auf dem Client ausgeführt, können diese gespeicherten Authentisierungsdaten

ausgelesen werden. Aus diesem Grund sollte die Verwendung dieser Funktion vermieden werden. Kann auf die Remember-Me-Funktion nicht verzichtet werden, so sollte der Benutzer explizit einer Aktivierung zustimmen müssen (Opt-In). Darüber hinaus sollte der Benutzer auf die Risiken der Funktion hingewiesen werden.

Neben Authentisierungsdaten in Cookies können aktuelle Browser häufig Formularfelder (z. B. Benutzername/Passwort oder Adressdaten) für eine spätere Wiederverwendung speichern. Wird ein Web-Formular, für das die eingegebenen Daten zuvor gespeichert wurden, erneut aufgerufen, so werden die Daten automatisch vom Browser in die Felder eingetragen. Daher sollte die Option "autocomplete=off" für alle Formularfelder mit vertraulichen Daten gesetzt werden. Dadurch werden die Browser angewiesen, die Daten der entsprechenden Formularfelder nicht zu speichern.

Zusätzliche Authentisierung bei kritischen Funktionen

Hat sich ein Benutzer erfolgreich authentisiert, so wird ihm von der Webanwendung üblicherweise eine eindeutige Sitzung (mittels SessionID) zugewiesen. Die Webanwendung kann mithilfe dieser SessionID die eintreffenden Requests den angemeldeten Benutzern zuordnen. Somit kann die SessionID als eine Art temporäres Anmeldedatum betrachtet werden, mit dessen Hilfe auf die Sitzungen angemeldeter Benutzer zugegriffen werden kann (siehe auch M 4.394 *Session-Management bei Webanwendungen und Web-Services*).

Da viele Angriffe gegen die SessionID bekannt sind (siehe G 5.169 *Unzureichendes Session-Management von Webanwendungen und Web-Services*), kann eine Übernahme von gültigen Sitzungen nicht vollständig ausgeschlossen werden. Daher sollte bei sicherheitskritischen Aktionen (z. B. Änderung des Passworts oder Löschung des kompletten Datenbestandes) eine erneute Authentisierung des Benutzers (z. B. durch Eingabe des alten Passworts bei einem Passwortwechsel) erfolgen.

Grenzwerte für gescheiterte Anmeldeversuche

Versucht ein Benutzer sich mehrfach in kurzen Zeitabständen an der Webanwendung anzumelden, so sollten diese Authentisierungsversuche als Angriff gewertet werden. Wenn die Zahl der fehlgeschlagenen Versuche einen festgelegten Wert überschreitet (z. B. fünf Fehlversuche), sollte das Benutzerkonto für ein definiertes Zeitintervall (z. B. 10 Sekunden) gesperrt werden. Darüber hinaus können die Zeitintervalle zur Sperrung des Benutzerkontos mit der Anzahl der Fehlversuche progressiv ansteigen. Hierdurch soll verhindert werden, dass Benutzer unbefugt Kennwörter anderer Benutzer erraten.

Bei der Wahl des Grenzwerts und der Zeitintervalle sollte beachtet werden, dass dieser Mechanismus für Denial-of-Service-Angriffe missbraucht werden kann. Ein Angreifer kann bewusst das Sperren vieler Benutzerkonten provozieren und somit diese Benutzer von der Nutzung der Webanwendung ausschließen.

Automatisiertes Zurücksetzen von Authentisierungsdaten

Da Webanwendungen oftmals von einem großen Benutzerkreis genutzt werden, bieten sie häufig Funktionen zum automatisierten Zurücksetzen der Authentisierungsinformationen (Passwort-Reset) an. So soll der administrative Aufwand möglichst gering gehalten werden, wenn z. B. ein Benutzer sein Passwort vergessen hat. Können die Authentisierungsdaten unbefugt zurückgesetzt werden, kann auf diese Weise der Authentisierungsmechanismus umgangen werden. Deshalb ist darauf zu achten, dass alle Funktionen einer We-

banwendung zur Änderung der Authentisierungsdaten mindestens genauso abgesichert sind wie die primäre Authentisierung der Webanwendung.

Wird beispielsweise im Prozess zum Zurücksetzen des Passworts die Authentisierung des Benutzers über eine geheime Frage mit entsprechender Antwort sichergestellt, so sollten die Merkmale vom Benutzer formuliert werden können. Er sollte darauf hingewiesen werden, dass sie keine Daten beinhalten sollten, die öffentlich verfügbar oder leicht zu erraten sind. Zur Erhöhung des Schutzniveaus können mehrere Fragen und Antworten bei der Registrierung aufgenommen werden (z. B. fünf, von denen mindestens drei Fragen für eine erfolgreiche Authentisierung richtig beantwortet werden müssen).

Zusätzlich kann noch ein weiteres Sicherheitsmerkmal verwendet werden, indem nach der korrekten Beantwortung der Fragen ein Link an eine zuvor vom Benutzer spezifizierte E-Mail-Adresse versendet wird oder ein weiteres Sicherheitstoken (z. B. eine PIN) per SMS an eine hinterlegte Rufnummer gesendet wird. Erst nach Klicken auf den Link bzw. Eingabe der PIN kann sich der Benutzer dann anmelden.

Da das Authentisierungsverfahren beim Zurücksetzen von Anmeldeinformationen in der Regel nur schwer auf das gleiche Sicherheitsniveau der primären Authentisierung zu bringen ist, sollte nach Möglichkeit auf ein automatisiertes Zurücksetzen durch die Webanwendung verzichtet werden. Bei eingeschränkten Nutzerkreisen der Webanwendung (z. B. bei einer Webanwendung im Intranet) kann stattdessen das Passwort manuell beispielsweise über eine Hotline mit sicheren Authentisierungsmerkmalen und entsprechendem Freigabeverfahren zurückgesetzt werden. Insbesondere bei einem hohen Schutzbedarf sollte eine manuelle Zurücksetzungsfunktion umgesetzt werden.

Prüffragen:

- Verwendet die Webanwendung eine zentrale Authentisierungskomponente?
- Erzwingt die Webanwendung die Verwendung sicherer Passwörter?
- Setzen die Webanwendungen eine explizite Zustimmung des Benutzers bei der Verwendung der Remember-Me-Funktion voraus (Opt-In)?
- Werden kritische Funktionen der Webanwendung durch eine zusätzliche Authentisierung geschützt?
- Definiert die Webanwendung Grenzwerte für fehlgeschlagene Anmeldeversuche, die Brute-Force-Angriffe erschweren?
- Erfüllen alle angebotenen Authentisierungsverfahren der Webanwendung (beispielsweise auch Funktionen zum automatisierten Zurücksetzen des Passworts) das gleiche Sicherheitsniveau?
- Wird der Benutzer einer Webanwendung umgehend über die Nutzung der angebotenen Passwort-Zurücksetzungsfunktion informiert?

M 4.393 Umfassende Ein- und Ausgabevalidierung bei Webanwendungen und Web-Services

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator, Entwickler

Alle an eine Webanwendung oder einen Web-Service übergebenen Daten, unabhängig von Kodierung oder Form der Übermittlung, müssen als potenziell gefährlich behandelt und entsprechend gefiltert werden. Durch eine zuverlässige und gründliche Filterung der Ein- und Ausgabedaten mittels einer Validierungskomponente kann ein wirksamer Schutz vor gängigen Angriffen erreicht werden. Hierbei sollten sowohl die Eingabedaten von Benutzern an die Webanwendung als auch die Ausgabedaten der Webanwendung an die Clients oder an nachgelagerte Systeme wie zum Beispiel Datenbanken gefiltert und transformiert (output encoding) werden. Entsprechendes gilt für die Aufrufparameter und Rückgabewerte von Web-Services. Dadurch wird sichergestellt, dass nur erwartete und keine schadhaften Daten verarbeitet oder ausgegeben werden.

Ist es für einzelne Funktionen notwendig, den Datenfilter weniger restriktiv zu setzen, muss dies explizit beim Zugriff auf die Daten definiert und dokumentiert werden. Zusätzlich ist es möglich, kontextsensitive Filter in der Geschäftslogik der Anwendung oder in den Hintergrundsystemen zu nutzen.

Für eine sichere Verarbeitung der Daten sollten folgende Punkte bei der Umsetzung und der Konfiguration der Validierungskomponente berücksichtigt werden:

Identifizierung der Daten

Damit die Ein- und Ausgabedaten umfassend validiert werden können, müssen zunächst alle zu verarbeitenden Datenstrukturen (zum Beispiel E-Mail-Adresse) und die darin zulässigen Werte identifiziert werden. Für jede Datenstruktur sollte eine entsprechende Validierungsroutine umgesetzt werden. Neben der Datenstruktur sollte auch die Art und Weise der Datenverarbeitung erfasst werden (zum Beispiel Weiterleitung an einen Interpreter, Rückgabe an den Client, Speicherung in einer Datenbank).

Berücksichtigung aller Daten und Datenformate

Die Validierungskomponente sollte alle zu verarbeitenden Datenformate und verwendeten Interpreter berücksichtigen. Typische Datenformate bei Webanwendungen sind zum Beispiel personenbezogene Daten (Name, Telefonnummer, Postleitzahl), Bilder, PDF-Dateien und formatierte Texte. Typische Interpreter für Daten, die von Webanwendungen und Web-Services verarbeitet oder ausgegeben werden, sind zum Beispiel HTML-Renderer, SQL-, XML-, JSON-, LDAP-Interpreter und das Betriebssystem.

Daten können durch unterschiedliche Techniken auf ihre Gültigkeit geprüft werden. So kann die Validierungskomponente den Wertebereich der Eingaben überprüfen oder es können beispielsweise reguläre Ausdrücke verwendet werden, um erlaubte Zeichen und die Länge der erwarteten Daten zu validieren. Die Gültigkeit von XML-Daten kann unter anderem mithilfe des entsprechenden XML-Schemas überprüft werden. Darüber hinaus stellen Fra-

meworks und Bibliotheken für gängige Datenformate entsprechende Validierungsfunktionen bereit.

Die folgenden Zeichen werden gewöhnlich von in Webanwendungen oder Web-Services eingesetzten Programmen interpretiert und können daher für das Einschleusen von schadhaftem Code genutzt werden. Aus diesem Grund sollten sie bei der Filterung berücksichtigt werden:

Nullwert, Zeilenvorschub, Wagenrücklauf, Hochkommata, Kommas, Schrägstriche, Leerzeichen, Tabulator-Zeichen, größer als und kleiner als, XML- und HTML-Tags.

Diese Aufzählung erhebt keinen Anspruch auf Vollständigkeit. Zudem können die Interpreter-Zeichensätze (zum Beispiel SQL-Syntax) bei unterschiedlichen Produkten variieren. Beispiele für kritische Zeichen werden im Abschnitt *Potenziell gefährliche Zeichen für Interpreter* in *Hilfsmittel zum Baustein Webanwendungen* aufgeführt.

Neben den eigentlichen Nutzdaten (zum Beispiel Formular-Parameter in GET- oder POST-Variablen) sind auch Daten anderer Herkunft (Sekundärdaten) zu validieren. Dazu zählen beispielsweise:

- Namen von Form-Variablen (Sie können ebenso wie der Wert der Form-Variablen beliebig manipuliert werden),
- HTTP-Header-Felder (Header-Felder in HTTP-Requests und -Responses sollten ausschließlich ASCII-Zeichen enthalten und zum Beispiel keine Zeilenvorschubzeichen wie \r\n),
- Session-IDs (zum Beispiel aus Cookies).

Automatisierte Aufrufe durch den Client zum Beispiel durch Ajax- beziehungsweise Flash-Skripte oder JSON-Requests sind ebenfalls zu prüfen.

Bei den Hintergrundsystemen ist eine (gegebenenfalls erneute) Validierung der Daten vorzunehmen. Dies gilt auch dann, wenn Daten beispielsweise nach einem erfolgten Schreibvorgang in die Datenbank wieder ausgelesen werden, da die Daten auch in der Datenbank zwischenzeitlich geändert worden sein können.

Schadhafter Code kann aber auch über einen Weg übermittelt werden, der nicht von der Webanwendung oder dem Web-Service kontrolliert werden kann (zum Beispiel FTP, NFS). Kann ein Angreifer über diese Dienste Dateien ändern oder erzeugen, die von der Webanwendung oder dem Web-Service integriert werden, so kann Schadcode über diesen Umweg eingebettet werden. Bei dem sogenannten Cross-Channel-Scripting wird auf diese Weise JavaScript-Code eingefügt, der ähnlich wie bei persistentem Cross-Site-Scripting vom Browser ausgeführt wird. Daher sollten unabhängig von der Quelle immer alle Daten vor der Ausgabe an die Benutzer oder der Weiterverarbeitung durch die Anwendung validiert werden.

Serverseitige Validierung

Üblicherweise greifen die Benutzer mit generischen Clients (zum Beispiel Web-Browser) auf die Webanwendung zu. Diese Clients befinden sich nicht im Sicherheitskontext der Webanwendung, sondern stehen unter der Kontrolle der Benutzer. In gleicher Weise kann sich auch ein Web-Service nicht darauf verlassen, dass die aufrufende Anwendung die Daten überprüft hat. Die Datenvalidierung ist daher als serverseitiger Sicherheitsmechanismus auf einem vertrauenswürdigen IT-System umzusetzen.

Werden Daten zusätzlich durch Code von der Webanwendung clientseitig verarbeitet (zum Beispiel JavaScript-Code), so sollten diese Daten auch auf dem Client validiert werden. Die ausgelieferten Skripte der Webanwendung sollten hierbei die entsprechenden Validierungsroutinen mitliefern. Werden die Daten im nachgelagerten Verarbeitungsprozess an den Server gesendet, so ist zu beachten, dass die clientseitige Prüfung die serverseitige Validierung nicht ersetzen kann.

Validierungsansatz

Bei der Datenvalidierung wird zwischen dem White-List- und dem Black-List-Ansatz unterschieden.

Bei dem White-List-Ansatz werden ausschließlich solche Daten zugelassen, die in der Liste enthalten sind. Dabei werden, ausgehend von einer möglichst kleinen Zeichenmenge, Regeln erstellt, die Daten in einem festgelegten Zeichenraum zulassen und Daten zurückweisen, die abweichende Zeichen enthalten. Hierbei sollten komplexe Regeln durch die sequenzielle Verwendung einfacher Regeln abgebildet werden.

Dagegen werden bei einem Black-List-Ansatz solche Daten als unzulässig eingestuft und abgewiesen, die in der Liste enthalten sind. Alle Daten, die nicht explizit verboten sind, werden bei diesem Ansatz akzeptiert.

Bei dem Black-List-Ansatz besteht jedoch die Gefahr, dass nicht alle Variationen unzulässiger Daten berücksichtigt und somit erkannt werden. Daher sollte der White-List-Ansatz dem Black-List-Ansatz vorgezogen werden.

Kanonisierung vor der Validierung

Daten können in verschiedenen Kodierungen (zum Beispiel UTF-8, ISO 8859-1) und Notationen (zum Beispiel bei UTF-8 ist "." = "2E" = "C0 AE") vorliegen. Abhängig vom angewendeten Kodierungsschema kann der gleiche Wert demnach unterschiedlich interpretiert werden. Findet eine Validierung der Daten ohne Berücksichtigung der Kodierung und der Notation statt, so werden gegebenenfalls schadhafte Daten nicht erkannt und gefiltert. Daher sollten alle Daten vor der Validierung in eine einheitliche, normalisierte Form überführt werden. Dieser Vorgang wird als Kanonisierung der Daten bezeichnet. Die so dargestellten Daten werden dann weiterverarbeitet. Bei der Verwendung von AJAX sollte zudem für das Nachladen die Eigenschaft textContent anstatt von innerHTML genutzt werden, da textContent automatisch eine Enkodierung vornimmt.

Darüber hinaus sollte das Kodierungsschema bei der Auslieferung von Daten durch die Webanwendung explizit gesetzt werden (zum Beispiel über den Content-Type-Header: charset=ISO-8859-1). Auch bei Web-Services sollten die verwendeten Kodierungen den Client-Systemen mitgeteilt werden, beispielsweise in den entsprechenden XML-Tags.

Kontextsensitive Maskierung der Daten

Falls potenziell schadhafte Daten von einer Webanwendung oder einem Web-Service verarbeitet werden müssen (zum Beispiel Zeichen mit einer Bedeutung für verwendete Interpreter) und somit eine Filterung nicht durchgeführt werden kann, müssen diese Daten maskiert und so in eine andere Darstellungsform überführt werden. In dieser maskierten Form werden die Daten nicht mehr als ausführbarer Code interpretiert. Da die Maskierung Interpreter-spezifisch ist, müssen alle verwendeten Interpreter berücksichtigt werden (zum

Beispiel SQL, LDAP). Die Maskierung muss demnach kontextsensitiv für das erwartete Ein- und Ausgabeformat und die Interpretersprache durchgeführt werden. Aufgrund der Komplexität und der spezifischen Anforderungen unterschiedlicher Interpretersprachen wird empfohlen, für die Maskierung spezialisierte Bibliotheken einzusetzen.

Es sollte eine Maskierung aller Zeichen vorgenommen werden, die als unsicher für den beabsichtigten Interpreter eingestuft werden. Dazu zählen zum Beispiel:

- unerwartetes JavaScript und HTML zur Auslieferung an den Client (zum Beispiel den Web-Browser),
- unerlaubt eingefügte SQL-Statements an die Datenbank (zum Beispiel aus Eingaben in Formularfeldern),
- Befehle an das Betriebssystem (zum Beispiel in manipulierten HTTP-Variablen).

Eine Maskierung kann durch eine Überführung der betroffenen Daten beziehungsweise Metazeichen der jeweiligen Interpretersprache in sogenannte Zeichenreferenzen erfolgen. Das folgende Beispiel zeigt ausgewählte HTML-Zeichen mit den entsprechenden Zeichenreferenzen (engl. *HTML-Entities*):

- & => &
- < => <
- > => >
- " => "
- ' => '

Hier ist darauf zu achten, dass &-Zeichen im ersten Durchlauf ersetzt werden und dass keine Mehrfach-Maskierung erfolgt, da dieses Zeichen in anderen Zeichenreferenzen als Metazeichen wiederverwendet wird.

Verwendung eines eigenen Markups zur Filterung von HTML-Tags

Falls die Webanwendung HTML-Formatierungstags in Benutzereingaben erfordert (zum Beispiel zur Formatierung von Benutzer-Beiträgen), sollten erlaubte HTML-Tags von problematischen Tags unterschieden und gefiltert werden (siehe auch Abschnitt *Kontextsensitive Maskierung der Daten*).

Bei diesem Ansatz besteht das hohe Risiko, problematische Tags (beispielsweise <script>) zu übersehen. Auch scheinbar harmlose Tags lassen sich teilweise über Attribute wie "onMouseOver" zur Ausführung von Code missbrauchen. Daher sollte der alternative Ansatz, für das Markup des Benutzers eigene Markup-Tags zu definieren (zum Beispiel BBCode), vorgezogen werden. Diese Markup-Tags werden dann von der Anwendung in die zugehörigen HTML-Tags übersetzt. Herkömmliche Tags beziehungsweise problematische Zeichen werden nach wie vor gefiltert.

Ein mögliches Verfahren, wenn ein einfaches Markup zugelassen werden soll, ist die Verwendung von { und } statt < und >. Fett würde dann als {F}**Dies ist fett**{/F} geschrieben und ein Bild könnte auf diese Weise platziert: {img src=/images/img.gif width=1 height=1 img}.

Hierbei darf die Umwandlung in HTML nicht einfach geschweifte Klammern durch spitze Klammern ersetzen, sondern muss jedes Tag als Ganzes ansehen:

- {img nach <img,
- img} nach >,
- src=Datei nach src="Datei" (wobei Datei zusätzlich zu filtern ist).

Wenn HTML-Tags zugelassen sind, ist grundsätzlich darauf zu achten, dass mindestens die folgenden Tags nicht erlaubt sind:

- applet
- base
- iframe
- link
- object
- script
- style

Mithilfe dieser Tags können beliebige Inhalte in die Webseite eingefügt werden. Diese dürfen daher nicht genutzt werden können.

Behandlung von Fehleingaben (Sanitizing)

Anstatt Daten aufgrund eines unerwarteten Datenformats oder Zeichens abzulehnen, können Fehleingaben korrigiert und automatisch transformiert werden (engl. *sanitize*). Dadurch soll eine benutzerfreundliche Eingabe der Daten in unterschiedlichen Schreibweisen ermöglicht werden. Für eine Weiterverarbeitung lassen sich die Daten von unerwarteten Zeichen säubern (zum Beispiel die Telefonnummer (0049)-201-12345678 kann in das nur aus Zahlen bestehende Format 004920112345678 überführt werden).

Eine Säuberung kann darin bestehen, Zeichen zu löschen, zu ersetzen oder zu maskieren (siehe auch Abschnitt *Kontextsensitive Maskierung der Daten*).

Beim Sanitizing besteht die Gefahr, dass Änderungen an den Daten zu einer neuen Komplexität, neuen Angriffsvektoren oder einer Missinterpretation führen. Daher sollte Sanitizing nach Möglichkeit vermieden und nur in Fällen angewendet werden, in denen ein Missbrauch des Sanitizing ausgeschlossen werden kann.

Falls die Webanwendung oder der Web-Service fehlerhafte Daten erkannt hat, sollten Fehler, die auf eine bewusste Manipulation hindeuten (zum Beispiel eine veränderte Session-ID) nicht automatisch korrigiert, sondern abgelehnt werden. Darüber hinaus sollten Eingabedaten, die mit bestimmungsgemäßer Browser- beziehungsweise Client-Bedienung nicht eintreten können, grundsätzlich abgelehnt werden. Dazu zählen zum Beispiel:

- Zusätzliche oder fehlende Formular-Parameter,
- Session-Cookies mit unerwarteten Zeichen oder ungültiger Länge,
- Unerwartete Werte bei der Übertragung von Formular-Parametern aus vordefinierten HIDDEN-, SELECT- oder CHECKBOX-Feldern,
- Abweichender oder unerwünschter Übertragungsweg der Parameter (zum Beispiel GET, POST, Cookie).

Bei einer Säuberung der Daten sollte die geschachtelte Eingabe von Angriffsvektoren berücksichtigt werden. Problematisch ist zum Beispiel der auf den ersten Blick vernünftig erscheinende Filter s/<script>//g; (hier in Perl RegEx-Syntax geschrieben), um <script>-Tags im Eingabestrom zu löschen. Dieser kann jedoch mit einer geschachtelten Eingabe (zum Beispiel <sc<script>ript>) umgangen werden. Es ist daher rekursiv zu filtern. Im Zweifelsfall sind die Eingabedaten abzulehnen.

Grundsätzlich sollte bei einer Ablehnung der Daten die angeforderte Aktion ebenfalls abgebrochen und eine neutrale Fehlermeldung ausgegeben werden (siehe auch M 4.400 *Restriktive Herausgabe sicherheitsrelevanter Informationen bei Webanwendungen und Web-Services*). Bei Webanwendungen oder

Web-Services mit hohem Schutzbedarf sollte zusätzlich die Sitzung invalidiert (abgebrochen) werden.

Prüffragen:

- Werden alle Daten (Ein- und Ausgabedaten) und Datenströme der Webanwendung oder des Web-Service (zum Beispiel zwischen Benutzer, Webanwendung, Clientsystemen und Hintergrundsystemen) bei der Validierung berücksichtigt?
- Werden auch Sekundärdaten (wie beispielsweise Session-IDs) bei der Validierung berücksichtigt?
- Führt die Webanwendung oder der Web-Service eine serverseitige Validierung der Daten auf einem vertrauenswürdigen IT-System durch?
- Führt die Webanwendung oder der Web-Service vor der Validierung eine Kanonisierung der Daten durch?
- Findet in der Webanwendung oder dem Web-Service eine kontextsensitive Validierung der Daten unter Berücksichtigung des erwarteten Interpreters der Daten statt?
- Bei Webanwendungen/Web-Services mit automatischer Behandlung von Fehleingaben (engl. *Sanitizing*): Wird die Behandlung von Fehleingaben sicher umgesetzt?

M 4.394 Session-Management bei Webanwendungen und Web-Services

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator, Entwickler

Webanwendungen und Web-Services verwenden in der Regel das zustandslose Protokoll HTTP zur Übertragung der Daten. Es unterstützt keine Zuordnung zusammengehörender Anfragen zu einem Benutzer wie zum Beispiel einzelne Seitenaufrufe zur Füllung eines virtuellen Warenkorbs. Um zusammengehörende Anfragen eines Benutzers zu erkennen und einer Sitzung zuzuordnen, wird eine Session-ID (zum Beispiel nach erfolgreicher Anmeldung) vergeben, die anschließend bei jedem Seitenaufruf, oder bei jeder Interaktion mit dem Web-Service, übertragen wird. Die Session-ID wird typischerweise von der Webanwendung oder dem Web-Service selbst erzeugt. Verwendet ein Web-Service den Standard WS-SecureConversation, kann der Security Context, und damit auch der Identifikator für die Sitzung, auch von einem dedizierten Security Token Service erzeugt werden.

Wenn sich der Benutzer bei der Webanwendung oder dem Web-Service angemeldet hat, ist die Session-ID vergleichbar mit seinen Zugangsdaten. Die Webanwendung identifiziert mit ihr bei jedem Seiten- oder Dienstaufruf den Benutzer und ordnet ihn einer (gegebenenfalls privilegierten) Sitzung zu. Nutzen Unbefugte die Session-ID, werden sie als legitime Benutzer identifiziert und können die Anwendung oder den Dienst im Namen des Opfers verwenden.

Das Session-Management einer Webanwendung oder eines Web-Service hat zur Aufgabe, die Sitzungen zu verwalten und neue Session-IDs zu vergeben. Dabei sollten die folgenden Anforderungen und Aspekte berücksichtigt werden.

Anforderungen an die Session-ID

Es ist zu beachten, dass die Gültigkeitsdauer einer Session-ID (siehe auch Abschnitt *Beschränkte Sitzungsdauer*) deutlich kleiner sein sollte als die Zeit, die ein Angreifer zum Erraten einer Session-ID benötigt. Dies kann mit einer Formel für eine Webanwendung oder einen Web-Service individuell bewertet werden (siehe *Formel zur Berechnung der Bewertungsgrundlage für Session-IDs in Hilfsmittel zum Baustein Webanwendung*).

Die Session-ID sollte mindestens folgende Anforderungen erfüllen:

- Die Session-ID muss mithilfe kryptografischer Zufallszahlengeneratoren zufällig erzeugt werden und sollte eine Entropie von mindestens 64 Bit haben, damit sie von einem potentiellen Angreifer nicht erraten werden kann. Um die Entropie der Session-ID zu erhöhen, kann beispielsweise die Länge erhöht (zum Beispiel 128 Bit) und der Zeichenraum der Session-ID (zum Beispiel alphanumerische Zeichen und Sonderzeichen) vergrößert werden. Als Richtwert sollte hierbei die Länge der Session-ID mindestens die doppelte Anzahl an Bits haben wie die Anzahl an Entropie-Bits der Session-ID. Demzufolge sollte die Session-ID mindestens 128 Bit lang sein. Unter der Annahme, dass ein Zeichen durch 8 Bit dargestellt wird, bestünde eine solche Session-ID aus mindestens 16 Zeichen (128 Bit / 8 = 16 Byte).

- Es sollten keine extern bekannten oder erratbaren Daten (zum Beispiel RFC-Adresse, Uhrzeit) in die Berechnung der Session-ID einfließen, sofern dies die Entropie nicht tolerierbar verringert.
- Unterstützt das der Webanwendung zugrunde liegende Framework die Generierung von Session-IDs, sollte vorzugsweise die Funktion des Frameworks verwendet werden. Die Funktionalität von führenden Frameworks ist in der Regel getestet und unterstützt die sichere Erzeugung von Session-IDs. Eine fehleranfällige Neuentwicklung sollte daher vermieden werden.
- Wird ein Framework zur Verwaltung und Erzeugung der Session-IDs verwendet, so ist auf eine sichere Konfiguration des Frameworks zu achten, sodass die zuvor genanten Anforderungen an die Session-ID erfüllt sind.

Schutz vor unbefugtem Zugriff auf die Session-ID

Die Session-ID kann sowohl in der URL eines Requests (GET-Methode), im Rumpf des Requests (POST-Methode) oder als Cookie im Header des Requests übertragen werden. Wird bei Web-Services der Standard WS-SecureConversation eingesetzt, so ist die Session-ID Teil des XML-Elements wsc:SecurityContextToken, welches innerhalb der SOAP-Header übertragen wird.

Wenn Daten mithilfe der GET-Methode übermittelt werden, können sie von beteiligten IT-Systemen gespeichert und dadurch von Dritten eingesehen werden (zum Beispiel im Browser-Verlauf, auf Bildschirmfotos, Seitenkopien oder Ausdrucken). Daher sollte die Session-ID nicht über die GET-Methode (also in der URL) übertragen werden. Für Webanwendungen oder Web-Services mit hohem Schutzbedarf ist dies nicht erlaubt. Stattdessen sollte die Session-ID vorzugsweise in Cookies übertragen werden.

Erfordert die Anwendung die GET-Methode (zum Beispiel aus Gründen der Kompatibilität mit Clients, die keine Cookies verarbeiten können), sind folgende Punkte zu beachten:

- Benutzer sollten auf die genannten Gefahren hingewiesen werden und beim Verlassen des Rechners die Sitzung beenden oder den Rechner sperren.
- Die Benutzer sollten angewiesen werden, keine gespeicherten Seiten oder Bildschirmfotos von Seiten der Webanwendung zu versenden, bei der die Session-ID in der URL sichtbar ist.
- Bei Nutzung der Webanwendung über einen öffentlichen Rechner sollte eine Meldung darauf hinweisen, dass der Browser-Verlauf nach Beenden der Sitzung gelöscht werden sollte.
- Durch sehr lange Session-IDs kann das Abschreiben und das zufällige Mitlesen erschwert werden.
- Bei der Verlinkung auf externe Seiten darf die Session-ID nicht übertragen werden. Dies gilt sowohl für die Übertragung in der URL als auch für das Referrer-Feld. Daher sollte bei Verlinkungen auf externe Seiten eine erzwungene Weiterleitung erfolgen, welche das Referrer-Feld bereinigt.

Zum Schutz vor unbefugtem Mitlesen der Session-ID sollte nach einer erfolgreichen Anmeldung die Kommunikation über eine sichere Verbindung stattfinden. Dies kann über eine Transportsicherung, beispielsweise mittels SSL/TLS(siehe M 5.66 *Clientseitige Verwendung von SSL/TLS*) oder mittels WS-SecureConversation realisiert werden. Die Session-ID kann über eine ungesicherte Verbindung übertragen werden, wenn mit der bestehenden Sitzung keine zugriffsgeschützten Bereiche der Webanwendung verwendet werden können. Gewöhnlich ist der Benutzer in diesem Fall noch nicht authentisiert.

Der Zugriff auf die Session-ID als Authentisierungsmerkmal sollte streng reglementiert werden. Wird die Session-ID in einem Cookie übertragen, sollte der clientseitige Zugriff auf diesen Cookie nach Möglichkeit durch das Setzen folgender Flags eingeschränkt werden (für eine detaillierte Beschreibung der Cookie-Flags siehe M 4.401 *Schutz vertraulicher Daten bei Webanwendungen*):

- Path (zum Beispiel /webapp/),
- Secure und
- HttpOnly.

Beschränkte Sitzungsdauer

Eine Webanwendung oder ein Web-Service muss Benutzern die Möglichkeit geben, eine bestehende Sitzung nach ihrer Nutzung explizit zu beenden. Daher muss auf allen Webseiten, für deren Abruf eine Authentisierung des Benutzers notwendig ist, eine deutlich sichtbare Abmeldemöglichkeit bestehen. Bei der Verwendung von WS-SecureConversation sollte der Security Context einer Sitzung nach der Nutzung des Dienstes explizit durch das Senden einer Nachricht "WS-Trust Cancel" invalidiert werden. Nach erfolgter Abmeldung sollte die Sitzung vollständig beendet werden und die Session-ID ihre Gültigkeit verlieren. Darüber hinaus sollte der Benutzer bei der Verwendung von Webanwendungen und Web-Services für folgende Verhaltensweisen sensibilisiert werden:

- Ist der Benutzer angemeldet, sollte er sich nach Abschluss der Tätigkeiten von der Webanwendung ordnungsgemäß abmelden.
- Falls beim letzten Besuch keine Abmeldung erfolgt ist, sollte der Benutzer bei dem nächsten Anmeldevorgang an der Webanwendung darauf hingewiesen werden, sich zukünftig abzumelden.

Ungenutzte, bestehende Sitzungen bieten eine Angriffsfläche für Brute-Force-Angriffe auf die Session-ID. Daher sollten Sitzungen nach einem Zeitintervall der Inaktivität ihre Gültigkeit verlieren (Idletime). Darüber hinaus sollte eine maximale Gültigkeits-Lebensdauer vergeben werden (Timeout), sodass auch die Session-IDs von aktiven Sitzungen eine begrenzte Gültigkeit haben. Diese sollte für die Sitzungen so gering wie möglich gewählt werden, sodass Brute-Force-Angriffe erschwert werden, wobei die Benutzbarkeit der Webanwendung hierbei nicht unnötig eingeschränkt werden sollte. Die Formel aus dem Abschnitt *Anforderungen an die Session-ID* kann für die Ermittlung einer angemessenen Gültigkeitsdauer herangezogen werden.

Treten bei der Nutzung der Webanwendung oder des Web-Service schwerwiegende Fehler auf, sollten angeforderte Aktionen abgebrochen und zusätzlich die Sitzung beendet werden. Schwerwiegende Fehler sind zum Beispiel auftretende Ausnahmefehler (Exceptions) und erkannte Angriffsversuche. Bei einem hohen Schutzbedarf sollten noch engere Kriterien in Erwägung gezogen werden, die zur Invalidierung der Sitzung führen (zum Beispiel ungültige Eingaben, Aufruf fehlender Seiten).

Bei der Invalidierung sollten die Sitzungsdaten server- und clientseitig vollständig gelöscht werden, sodass die Sitzung serverseitig nicht weiter akzeptiert wird und clientseitig keine Informationen über zuvor aufgebaute Sitzungen verbleiben. Dies kann zum Beispiel durch Löschen des Cookies mit der Session-ID erfolgen.

Darüber hinaus können mehrere parallele Sitzungen unter dem gleichen Benutzerkonto verhindert werden. Eine bestehende Sitzung kann bei erneuter Anmeldung invalidiert werden, sodass nur die neue Sitzung gültig bleibt. Al-

ternativ ist es beispielsweise möglich, die erste Sitzung über einen begrenzten Zeitraum (zum Beispiel 15 Minuten) aufrechtzuerhalten, bevor sie invalidiert wird. Dabei sollte dem Benutzer bei der Anmeldung über eine parallele, zweite Sitzung eine Meldung über die ablaufende, erste Sitzung eingeblendet werden. Auf diese Weise können noch bestehende, aber nicht mehr verwendete Sitzungen nach erneuter Anmeldung nicht oder nur eingeschränkt unbefugt von Dritten genutzt werden.

Zum Schutz vor Session-Fixation-Angriffen sollte nach erfolgter Anmeldung eine bereits bestehende Session-ID durch eine neue ersetzt werden.

Ebenso sollte nach einem Wechsel von einem ungesicherten Kommunikationskanal (HTTP) auf einen gesicherten Kommunikationskanal (HTTPS) eine neue Session-ID vergeben werden, da die Session-ID bei der Übertragung über einen ungesicherten Kanal mitgelesen worden sein könnte.

Schutz der Sitzungsdaten

Zum Schutz der Vertraulichkeit sollten die anfallenden Sitzungsdaten (zum Beispiel Warenkorb) ausschließlich serverseitig auf einem vertrauenswürdigen IT-System gespeichert werden. Darüber hinaus sollten die Daten vor unbefugtem Zugriff von anderen Benutzern durch eine Zugriffskontrolle geschützt werden. Falls die Webanwendung oder der Web-Service eine clientseitige Speicherung der Sitzungsdaten erfordert, sollte ebenfalls M 4.401 *Schutz vertraulicher Daten bei Webanwendungen* für die Speicherung von Daten auf dem Client beachtet werden.

Zuordnung einer Sitzung anhand zusätzlicher Attribute

Neben der Session-ID können weitere Merkmale zur Zuordnung zwischen Benutzer und Sitzung verwendet werden (zum Beispiel die IP-Adresse). Hierdurch kann die unbefugte Nutzung bestehender Sitzungen erschwert werden, da ein Angreifer für eine erfolgreiche Übernahme der Sitzung neben einer gültigen Session-ID die zusätzlichen Merkmale kennen muss. Die Verwendung von zusätzlichen Attributen zur Zuordnung einer Sitzung ist zumindest bei Webanwendungen mit hohem Schutzbedarf zu berücksichtigen.

Wird die IP-Adresse als zusätzliches Merkmal für die Sitzungszuordnung verwendet, so ist diese serverseitig zu speichern und zu prüfen. Wechselt die IP-Adresse im Laufe einer Sitzung, so sollte dies bei Anwendungen mit hohem Schutzbedarf als Angriffsversuch gewertet und demzufolge die Sitzung invalidiert werden. Dabei ist jedoch zu berücksichtigen, dass die IP-Adresse nicht immer einem Benutzer eindeutig zugeordnet werden kann. Erfolgt die Verbindung einiger Benutzer der Webanwendung über einen Proxy mit gleicher (zum Beispiel Reverse-Proxy) oder wechselnder IP-Adresse (zum Beispiel wechselnde, ausgehende Proxys), besteht die Gefahr, dass die IP-Adressen dieser Benutzer nicht eindeutig einer Sitzung zugeordnet werden können. Es sollte somit bedacht werden, dass einige Benutzer die Webanwendung möglicherweise nur eingeschränkt oder gar nicht nutzen können.

Wenn der Referrer als Identitätsmerkmal verwendet wird, kann auf einen festen Teil des Referrer-Pfades geprüft werden, der für alle Zugriffe identisch bleibt (zum Beispiel die Domäne der Webanwendung). Die Benutzer müssen demnach eine Webseite der Webanwendung im Referrer vorweisen. Hierbei ist zu berücksichtigen, dass einige Browser eine Deaktivierung oder benutzerseitige Manipulation der Referrer-Übermittlung erlauben und Content-Filter dieses Feld gegebenenfalls filtern.

Die Identitätsmerkmale können zum Schutz vor unbefugter Nutzung der Sitzung auf mehrere Eigenschaften des HTTP-Headers verteilt werden. Denkbar sind zum Beispiel HTTP-Header-Informationen wie

- die Browsertypenbezeichnung (User-Agent-Header),
- unterstützte Formate und Sprachen des Clients (Accept- und Accept-Language-Header) und
- der Referrer (Referrer-Header).

Aufgrund der teilweise geringen Variationsbreite der genannten Merkmale des HTTP-Headers ist der zusätzlich erreichte Schutz begrenzt. Dagegen erhöht sich der Umsetzungsaufwand und unter Umständen die Komplexität bei der Fehlersuche. Aus diesem Grund sollte im Einzelfall abgewogen werden, ob der zusätzlich erreichte Schutz den Umsetzungsaufwand rechtfertigt.

Eigenimplementierungen vermeiden

Kann für die Sitzungsverwaltung einer Web-Anwendung oder eines Web-Service auf eine erprobte Implementierung in einem Framework oder einen verbreiteten Standard (wie WS-SecureConversation) zurückgegriffen werden, so ist dies gegenüber einer Eigenimplementierung in jedem Fall zu bevorzugen, da sich Eigenimplementierungen dieser sicherheitskritischen, komplexen Funktion sehr häufig als angreifbar erweisen.

Prüffragen:

- Hat die Session-ID der Webanwendung beziehungsweise des Web-Service eine ausreichende Entropie, um dem Erraten der Session-ID (zum Beispiel durch einen Brute-Force-Angriff) standzuhalten?
- Wird die Vertraulichkeit der Session-ID bei der Übertragung und clientseitigen Speicherung ausreichend geschützt?
- Hat die Sitzung eine begrenzte Gültigkeit (Timeout) und ist diese gemessen an den Anforderungen zur Nutzung der Webanwendung oder des Web-Service möglichst kurz gewählt worden?
- Erfolgt ein Wechsel der Session-ID nach erfolgreicher Authentisierung?
- Werden alle Sitzungsdaten (sowohl server- als auch clientseitig) nach der Invalidierung der Sitzung ungültig und gelöscht?
- Kommt für die Sitzungsverwaltung eine erprobte Implementierung oder ein verbreiteter Standard zum Einsatz?

M 4.395 Fehlerbehandlung durch Webanwendungen und Web-Services

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator, Entwickler

Tritt während des Betriebs einer Webanwendung oder eines Web-Services ein Fehler auf, sollte dieser so behandelt werden, dass ein konsistenter Zustand der Webanwendung gewährleistet ist und somit etwa der Schutz der Daten aufrechterhalten wird.

Eine Webanwendung oder ein Web-Service ist in einem inkonsistenten Zustand, wenn sie aufgrund eines Fehlers in einen unerwarteten Zustand überführt wird und dadurch Daten unkontrolliert verarbeitet werden (zum Beispiel keine Fehlermeldung bei erfolgloser Speicherung von Daten).

Der konsistente Zustand einer Webanwendung oder eines Web-Service kann unter anderem durch folgende Ereignisse gefährdet werden:

- Absturz der Anwendung
- unvollständig durchgeführte Transaktionen auf Anwendungsebene
- Durchführung einer Aktion trotz Fehler (zum Beispiel bei unvollständigen Prüfungen durch Sicherheitskomponenten)
- Verhinderung von Diensten (Denial-of-Service)
- Rechteausweitung (privilege escalation)
- Ausführen von Schadcode (code execution)

Folgende Punkte sollten bei der Fehlerbehandlung berücksichtigt werden:

Vermeidung vertraulicher Informationen in Fehlermeldungen

Die Webanwendung muss dem Benutzer im Falle eines Fehlers neutrale, angepasste Fehlerseiten ausgeben, die keine vertraulichen Informationen beinhalten. Auch die Rückmeldungen von Web-Services sollten im Fehlerfall keine vertraulichen Informationen, wie etwa interne Pfade oder die Versionsnummern von verwendeten Softwarekomponenten, enthalten. Siehe hierzu auch M 4.400 *Restriktive Herausgabe sicherheitsrelevanter Informationen bei Webanwendungen und Web-Services*.

Protokollierung der Fehler

Für eine vollständige Nachvollziehbarkeit aufgetretener Fehler müssen diese als Ereignis gemäß M 4.397 *Protokollierung sicherheitsrelevanter Ereignisse von Web-Anwendungen und Web-Services* protokolliert werden.

Abbruch des Vorgangs nach Auftreten eines Fehlers

Treten Fehler im Zusammenhang mit Sicherheitskomponenten der Webanwendung oder des Web-Service auf (zum Beispiel während der Autorisierung oder Authentisierung), muss die veranlasste Aktion abgebrochen und der Zugriff auf die angeforderte Ressource oder Funktion abgewiesen werden. Es muss gewährleistet sein, dass durch provozierte Fehler keine Sicherheitsmechanismen umgangen werden können.

Für Webanwendungen oder Web-Services mit einem hohen Schutzbedarf sollte zusätzlich die Invalidierung einer gegebenenfalls bestehenden Sitzung

in Betracht gezogen werden (siehe auch M 4.394 *Session-Management bei Webanwendungen und Web-Services*).

Freigabe von reservierten Ressourcen

Im laufenden Betrieb belegen Webanwendungen und Web-Services Ressourcen wie zum Beispiel Netz- oder Datei-Streams, um auf Hintergrundsysteme, zwischengespeicherte Zustände oder sonstige Daten zuzugreifen. Solange die Webanwendung oder der Web-Service auf diese Ressourcen zugreift, sind diese in der Regel für deren exklusiven Zugriff reserviert und können von anderen Prozessen nicht verwendet werden.

Tritt ein Fehler auf, sollten zuvor reservierte Ressourcen (zum Beispiel ein Datei-Handle auf eine temporäre Datei) im Rahmen der Fehlerbehandlung freigegeben werden. Darüber hinaus sind zwischengespeicherte Daten bei der Fehlerbehandlung zu löschen.

Unmittelbare Behandlung von Fehlern

Interne Fehler sollten von der Webanwendung oder dem Web-Service selbst behandelt werden. Die Weiterleitung eines unbehandelten Fehlers an andere Komponenten (zum Beispiel Applikationsserver oder nachgelagerte Web-Services) kann zu einem Verlust von Informationen führen, die zur Behandlung des Fehlers notwendig sind (zum Beispiel zur Freigabe von gebundenen Ressourcen). Daher sollten unbehandelte Fehler nicht weitergeleitet werden.

Vermeidung einer zu hohen Fehlertoleranz

Sind Ursachen von Fehlerzuständen nicht vollständig geklärt, sollte der Fehler nicht zum Beispiel aufgrund der Bedienungsfreundlichkeit toleriert, sondern die Aktion im Zweifelsfall abgebrochen werden. Schwerwiegende Fehler sollten immer zum Abbruch der Aktion führen.

Das Ziel sind robuste und fehlertolerante Webanwendungen und Web-Services, die bestimmungsgemäße Bedienung durch den Anwender von offensichtlichen Missbrauchsversuchen und schwerwiegenden Fehlern unterscheiden und dann angemessen reagieren können.

Prüffragen:

- Werden von der Webanwendung oder dem Web-Service ausschließlich Fehlermeldungen ausgegeben, die keine vertraulichen Informationen beinhalten?
- Ist eine Protokollierung von Fehlern vorgesehen?
- Wird eine veranlasste Aktion im Fehlerfall abgebrochen und in der Folge der Zugriff auf die angeforderte Ressource oder Funktion abgewiesen?
- Sieht die Fehlerbehandlung eine Freigabe gebundener Ressourcen vor?
- Werden Fehler möglichst von der gleichen Komponente behandelt, in der der Fehler aufgetreten ist?

M 4.396 Schutz vor unerlaubter automatisierter Nutzung von Webanwendungen

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Entwickler, Administrator

Eine Webanwendung wird gewöhnlich von Menschen genutzt und erfordert somit keine automatisierte Nutzung (z. B. durch Skripte). Brute-Force-Angriffe (z. B. Erraten von Zugangsdaten) und Enumeration-Angriffe (z. B. automatisiertes Ermitteln von gültigen Login-Namen) beruhen hingegen auf der automatisierten Steuerung einer Webanwendung (Automation). Bei diesen Angriffen wird zumeist versucht, vertrauliche Daten durch wiederholende, leicht variierende Abfragen (z. B. geänderte Benutzernamen) zu sammeln.

Zur Verhinderung von Automation und der Abwehr damit einhergehender Angriffe muss die Webanwendung automatisierte von manuellen Zugriffen unterscheiden können. Automatisierte Angriffe zeichnen sich durch eine hohe Zahl an Zugriffsversuchen innerhalb einer kurzen Zeitspanne aus, die das übliche Maß deutlich übersteigt.

Daher kann eine Toleranzschwelle für wiederholt abgerufene Ressourcen derartige Angriffe erschweren (Teergrube). Werden Grenzwerte gegen automatisierte Anfragen festgelegt, ist darauf zu achten, dass legitime Benutzer in der Funktionalität und der Bedienung der Webanwendung möglichst wenig eingeschränkt werden. Falls Grenzwerte für elementare Funktionen der Webanwendung zu eng bemessen sind, können Angreifer dies auf Webanwendungs-Ebene für Denial-of-Service-Angriffe missbrauchen. Werden beispielsweise Benutzerkonten nach einer festgelegten Anzahl an erfolglosen Anmeldeversuchen für ein gewisses Zeitintervall gesperrt, können gezielte Falscheingaben zu einer längerfristigen Sperrung vieler Benutzerkonten führen. Demzufolge können sich legitime Benutzer in diesem Zeitraum nicht mehr an der Webanwendung anmelden.

Darüber hinaus ist die Effizienz automatisierter Angriffe in der Regel stark abhängig vom Detailgrad der Informationen in den Rückantworten der Webanwendung (siehe M 4.400 *Restriktive Herausgabe sicherheitsrelevanter Informationen bei Webanwendungen und Web-Services*).

Folgende Beispiele geben Hinweise auf mögliche Schutzmechanismen:

- Eine künstliche Verzögerung zwischen der Eingabe der Zugangsdaten bei der Benutzer-Authentisierung und der Meldung über einen fehlgeschlagenen Anmeldeversuch kann Brute-Force-Angriffe aufgrund des erhöhten Zeitbedarfs erschweren. Die Wirksamkeit dieser Methode kann durch ein progressives Ansteigen der Verzögerung nach jedem gescheiterten Versuch erhöht werden.
- Werden Eingaben zurückgewiesen, sollten Informationen über die Ursache generisch verfasst sein. Einem Angreifer darf es beispielsweise nicht möglich sein aufgrund von Meldungen, wie "Passwort ungültig" anstelle von "Zugangsdaten ungültig", auf ein gültiges Benutzerkonto zu schließen (siehe auch M 4.395 *Fehlerbehandlung durch Webanwendungen und Web-Services*).
- Angriffsversuche sind häufig gekennzeichnet durch vielfache Fehlversuche bei der Durchführung einer Aktion. Daher sollte eine vorhandene Sitzung beendet werden, wenn eine ungewöhnlich hohe Anzahl von Fehlver-

suchen identifiziert wird, und anschließend eine Neuanmeldung erforderlich sein.
- Automatisierte Angriffe können durch eine temporäre Sperrung der IP-Adresse bei Verdacht auf einen Angriff abgewehrt werden. Es sollte hierbei bedacht werden, dass durch diese Maßnahme gegebenenfalls Unbeteiligte ebenfalls von der Sperrung betroffen sind (z. B. wenn mehrere Benutzer denselben Proxy verwenden).
- Häufig werden sogenannte CAPTCHAs (Completely Automated Public Turing Test To Tell Computers and Humans Apart) zur Unterscheidung automatisierter und manueller Zugriffe eingesetzt. Hierbei müssen vom Benutzer der Webanwendung Aufgaben gelöst werden (z. B. die Zeichen in einem Bild müssen erkannt und abgetippt oder Rätselfragen beantwortet werden), was für ein Computerprogramm nicht ohne Weiteres möglich ist. Abhängig von der verwendeten Technik und Aufgabenstellung ist die Webanwendung dadurch gegebenenfalls nur eingeschränkt für Menschen mit Behinderung nutzbar. So sollte z. B. alternativ zum Einblenden der Aufgabe, diese auch akustisch zur Verfügung gestellt werden, um Menschen mit Sehbehinderung die Nutzung der Webanwendung zu ermöglichen. Es ist zu beachten, dass der Einsatz von CAPTCHAs aus Gründen der Diskriminierung in vielen Ländern gesetzlich geregelt oder verboten ist. In Deutschland ist die Bundesverwaltung verpflichtet ihre öffentlich zugänglichen Internet- und Intranet-Angebote nach der Barrierefreien Informationstechnik-Verordnung (BITV) zu gestalten.

Prüffragen:

- Erkennt die Webanwendung automatisierte Zugriffe und werden geeignete Maßnahmen getroffen, die eine automatisierte Nutzung erschweren oder unterbinden?
- Werden bei der Festlegung von Grenzwerten bei Webanwendungen mögliche Auswirkungen berücksichtigt (z. B. Anfälligkeit für Denial-of-Service-Angriffe)?
- Wird eine restriktive Informationspolitik von der Webanwendung umgesetzt?
- Werden die rechtlichen Rahmenbedingungen vor dem Einsatz von Schutzmaßnahmen geprüft, die die Nutzung der Webanwendung auf bestimmte Anwenderkreise einschränken und somit diskriminieren könnten (z. B. CAPTCHA)?

M 4.397 Protokollierung sicherheitsrelevanter Ereignisse von Web-Anwendungen und Web-Services

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator, Entwickler

Sicherheitsrelevante Ereignisse (zum Beispiel Zugriffe auf Ressourcen, Authentisierungsversuche) müssen nachvollziehbar protokolliert werden, damit im Stör- oder Fehlerfall oder nach Angriffsversuchen die Protokolldaten zur Ursachenfindung herangezogen werden können.

Neben den Empfehlungen in den Maßnahmen M 5.9 *Protokollierung am Server* und M 2.110 *Datenschutzaspekte bei der Protokollierung* sollten zusätzlich die folgenden Punkte bei der Protokollierung sicherheitsrelevanter Ereignisse von Web-Anwendungen und Web-Services beachtet werden.

Zu protokollierende Ereignisse bei Web-Anwendungen und Web-Services

Zusätzlich zur Protokollierung auf den Server- und Hintergrundsystemen (zum Beispiel Betriebssystem, Web- und Applikationsserver, Datenbank) sollte auch die Anwendung sicherheitsrelevante Ereignisse protokollieren. Mindestens folgende Ereignisse sollten auf Anwendungsebene erfasst werden:

- erfolgreiche und erfolglose Anmeldeversuche an der Webanwendung oder dem Web-Service,
- fehlgeschlagene Autorisierungsversuche beim Zugriff auf Ressourcen (zum Beispiel Datenbankzugriffe) und Funktionen der Webanwendung oder des Web-Service,
- fehlgeschlagene Validierung von Ein- und Ausgabedaten,
- fehlgeschlagene XML-Schema-Validierungen,
- XML-Parser-Fehler,
- aufgetretene Fehler (zum Beispiel Exceptions),
- Änderungen von Berechtigungen für Benutzer oder Benutzergruppen der Webanwendung oder des Web-Service (zum Beispiel Zugriffsrechte, Änderung an der Web-Service-Policy),
- Änderungen an Benutzerkonten (zum Beispiel Passwortänderung),
- Löschvorgänge der Webanwendung (zum Beispiel Beiträge),
- erkannte Manipulationsversuche und unerwartete Änderungen (zum Beispiel Anmeldeversuche mit ungültigen oder abgelaufenen Session-IDs),
- administrative Funktionsaufrufe und Änderungen an der Konfiguration (zum Beispiel Abruf von Benutzerdaten, Aktivierung und Deaktivierung der Protokollierung),
- Starten und Stoppen von Diensten,
- Produktionsübernahme (Deployment) neuer oder bestehender Web-Services.

Zu protokollierende Merkmale von Ereignissen

Um sicherheitsrelevante Vorgänge anhand von Protokolldaten nachvollziehen zu können, müssen grundlegende Merkmale der Ereignisse verfügbar sein. Daher sollten mindestens die folgenden Merkmale protokolliert werden:

- Datum,
- Uhrzeit mit Zeitzone,

- assoziierter Benutzername,
- betroffenes Objekt (zum Beispiel Benutzerkonto, Datei, Datenquelle),
- Status der Aktion (zum Beispiel fehlgeschlagen, erfolgreich),
- Ort des Auftretens (zum Beispiel Komponente),
- Aktion (zum Beispiel Authentisierung, Autorisierung),
- Schweregrad (zum Beispiel Information, Warnung, Fehler).

Darüber hinaus kann es auch hilfreich sein, folgende Merkmale zu protokollieren:

- Source-IP-Adresse,
- Referenzen auf die SessionID (nicht die SessionID selbst),
- IT-System, an dem der Fehler aufgetreten ist,
- Softwarestand (Version) der Webanwendung.

Vertrauliche und sicherheitsrelevante Daten (zum Beispiel SessionID, Zugangsdaten) sollten nicht protokolliert werden.

Geeignete Datenformate und Mechanismen

Die protokollierten Daten sollten in einem einheitlichen Format gespeichert werden, damit eine effiziente Auswertung möglich ist. Die Protokollierungskomponente der Webanwendung oder des Web-Service sollte aus diesem Grund ein Datenformat verwenden, das in bestehende Lösungen integriert werden kann. Wird beispielsweise eine zentrale Komponente für die Auswertung der Protokolldaten verwendet, so sollten Datenformate gewählt werden, die diese Komponente unterstützt.

Serverseitige Protokollierung durch eine zentrale Komponente

Die Protokollierung der Webanwendung oder des Web-Service ist ausschließlich serverseitig durchzuführen, da nur auf diese Weise die Protokolldaten zentral ausgewertet werden können. Die Protokolldaten sollten von einer einzigen, zentralen Protokollierungskomponente der Webanwendung oder des Web-Service und nicht von unterschiedlichen Protokollierungskomponenten erhoben werden.

Eine fehleranfällige Neuentwicklung der Protokollierungskomponente sollte vermieden werden. Stattdessen sollte auf die Funktionalität etablierter Frameworks zurückgegriffen werden, die in der Regel einen zentralisierten Protokollierungsansatz und die Protokollierung in verbreiteten Protokolldatenformaten unterstützen (siehe Abschnitt *Geeignete Datenformate und Mechanismen*).

Schutz vor unbefugtem Zugriff und der Manipulation von Protokolldaten

Da die Protokolldaten vertrauliche Informationen (zum Beispiel über das Benutzerverhalten und den Aufbau beziehungsweise die Konfiguration der Webanwendung oder des Web-Service) enthalten können, muss der Zugriff auf die Protokolldaten reglementiert und nur befugten Benutzern ermöglicht werden. Der Zugriff auf Protokolldaten sollte nicht über öffentliche Schnittstellen möglich sein. Protokolldaten sollten daher in dedizierten Logverzeichnissen (zum Beispiel außerhalb des Web-Root-Verzeichnisses des Web-Servers) gespeichert werden.

Werden die Protokolldaten in einer Datenbank abgelegt, so sollten die Protokolldaten von den eigentlichen Nutzdaten getrennt werden. Diese Trennung kann mittels einer separaten Datenbanktabelle erreicht werden. Darüber hinaus kann ein eigener Datenbankbenutzer für die Protokollierung den Schutz

der Protokolldaten erhöhen. In diesem Fall darf der Datenbankbenutzer für die Nutzdaten keine Zugriffsrechte auf die Protokolldaten haben.

Alternativ können die Protokollierungsdaten mit hohem Schutzbedarf auch in einer separaten Datenbankinstanz gespeichert werden.

Sichere Protokollauswertung

Ein Angreifer kann bewusst Protokoll-Einträge provozieren (zum Beispiel wenn Eingabefelder protokolliert werden), die schadhaften Programmcode beinhalten. Daher sollte bei der Auswertung der Protokolldaten sichergestellt werden, dass Schadcode in Protokoll-Einträgen vom Auswertungsprogramm nicht interpretiert wird (zum Beispiel durch die Ansicht in einem Browser und der Interpretation von JavaScript-Code in den Protokolldaten).

Da bei der Protokollauswertung keine Änderungen an den Protokolldaten vorgenommen werden dürfen, sind die Protokolldaten ausschließlich in einem schreibgeschützten Modus zu analysieren.

Zeitsynchronisation

Die Protokolldaten verschiedener Komponenten einer Webanwendung oder eines Web-Service (zum Beispiel Applikationsserver, Webserver, Datenbankserver) müssen in der Regel korreliert werden, um komponentenübergreifende Vorgänge vollständig nachvollziehen zu können. Dazu sollte die Zeit auf den Systemen synchronisiert sein, um anhand der Uhrzeiten Vorgänge in den Protokollen konsistent nachverfolgen zu können. Hierzu sollte M 4.227 *Einsatz eines lokalen NTP-Servers zur Zeitsynchronisation* beachtet werden.

Prüffragen:

- Werden sicherheitsrelevante Ereignisse mit den erforderlichen Merkmalen von der Webanwendung oder dem Web-Service protokolliert?
- Werden keine vertraulichen Daten (zum Beispiel Zugangsdaten) protokolliert?
- Wird die Protokollierung ausschließlich serverseitig von einer zentralen Komponente der Webanwendung oder des Web-Service durchgeführt?
- Ist der Zugriff auf die Protokolldaten nur befugten Benutzern ermöglicht?
- Wird der Zugriff auf die Protokolldaten über die öffentliche Schnittstelle unterbunden?
- Verwendet die Webanwendung oder der Web-Service Datenformate und Mechanismen zur Protokollierung, die eine Integration in bestehende Lösungen ermöglicht?
- Wird eine Zeitsynchronisation für die Komponenten der Webanwendung oder des Web-Service umgesetzt?
- Wird das Ausführen von Schadcode bei der Protokollauswertung verhindert?

M 4.398 Sichere Konfiguration von Webanwendungen

Verantwortlich für Initiierung: Leiter IT
Verantwortlich für Umsetzung: Entwickler, Administrator

Ist eine Webanwendung unzureichend konfiguriert, so kann ein Angreifer möglicherweise bestehende Sicherheitsmechanismen überwinden. Daher muss sichergestellt werden, dass die Webanwendung so konfiguriert wird, dass Zugriffe ausschließlich über die vorgesehenen, abgesicherten Kommunikationspfade möglich sind. Der Zugriff auf nicht benötigte Ressourcen und Funktionen ist daher einzuschränken.

Folgende Punkte sollten bei der Konfiguration der Webanwendung berücksichtigt werden:

Deaktivierung nicht benötigter HTTP-Methoden

Auf eine Webanwendung kann gemäß HTTP-Standard mit unterschiedlichen HTTP-Methoden (z. B. GET, POST, PUT, DELETE oder TRACE) zugegriffen werden. Üblicherweise benötigt eine Webanwendung jedoch nur eine sehr eingeschränkte Menge dieser HTTP-Methoden (z. B. GET und POST).

Darüber hinaus kann eine Webanwendung in Abhängigkeit der verwendeten HTTP-Methode unterschiedlich auf einen Request reagieren. Wird beispielsweise die Eingabedatenfilterung nur bei einem GET- oder POST-Request durchgeführt, so kann diese Sicherheitsfunktion durch den Aufruf einer nicht vorgesehenen HTTP-Methode gegebenenfalls umgangen werden.

Einige HTTP-Methoden (z. B. PUT) bieten Zugriff auf sicherheitskritische Funktionalität (z. B. Hochladen beliebiger Dateien) und ermöglichen auf diese Weise Restriktionen der Webanwendung zu umgehen (z. B. die Dateitypenprüfung bei einer Upload-Funktion).

Aus diesen Gründen sollten nicht benötigte HTTP-Methoden deaktiviert und von der Webanwendung nicht bearbeitet werden. Dies gilt auch für fiktive HTTP-Methoden, die nicht im entsprechenden Standard RFC 2616 definiert werden. Auch wenn die HTTP-Methoden bereits in der Konfiguration des Webservers deaktiviert wurden, sollte auch die Webanwendung nicht benötigte HTTP-Requests nicht bearbeiten.

Erzwingen der HTTP-POST-Methode

Bei der Bedienung einer Webanwendung werden üblicherweise Daten (z. B. Formulardaten oder die SessionID) an die Webanwendung übermittelt. Diese Daten können als Parameter in der URL (GET-Methode) und im Rumpf des HTTP-Requests (POST-Methode) übertragen werden.

Bei der Verwendung der GET-Methode sind vertrauliche Daten wie Formulardaten in der URL sichtbar (z. B. im Browser-Verlauf) und können von zwischengelagerten Systemen protokolliert und gespeichert werden.

Daher sollten schützenswerte Daten ausschließlich über die POST-Methode übertragen werden. Hierbei ist zu berücksichtigen, dass Frameworks häufig die HTTP-Request-Methode abstrahieren. Eine falsche Konfiguration des Frameworks kann dazu führen, dass trotz erzwungener Eingrenzung auf die POST-Methode durch die Webanwendung weiterhin beide Methoden zuläs-

sig sind (z. B. über eine Weiterleitung eines HTTP-GET-Requests auf einen HTTP-POST-Request durch das Framework).

Sicherer Umgang mit SSL/TLS

Zum Schutz der übertragenen Daten zwischen Webanwendung und Client des Benutzers kann der Transportkanal durch kryptographische Verfahren (z. B. SSL/TLS) geschützt werden. Vertrauliche Daten sollten immer über einen verschlüsselten Transportkanal übertragen werden (siehe auch M 5.66 *Clientseitige Verwendung von SSL/TLS*).

Darüber hinaus ist darauf zu achten, dass bei Fehlern während des SSL/TLS-Verbindungsaufbaus oder bei der Übertragung von Daten über einen verschlüsselten Kanal nicht zu einer unverschlüsselten Verbindung gewechselt wird. Stattdessen sollte der Verbindungsaufbau erneut erfolgen oder abgelehnt werden. Es muss verhindert werden, dass vertrauliche Daten über eine ungesicherte Verbindung übertragen werden (z. B. durch setzen des Secure-Flags für Cookies; siehe M 4.401 *Schutz vertraulicher Daten bei Webanwendungen*).

Zeichenkodierungskonfiguration

Die übermittelten Daten zwischen dem Client des Benutzers und der Webanwendung können in verschiedenen Kodierungen vorliegen. Abhängig von der erwarteten Kodierung werden die Daten von Clients, von der Webanwendung oder von den Hintergrundsystemen unterschiedlich interpretiert. Damit Clients Daten an die Webanwendung in der gewünschten Kodierung senden, sollte die Webanwendung bei der Auslieferung von Webseiten in den Header-Feldern der HTTP-Response das Zeichenkodierungsschema (z. B. UTF-8) mit angeben.

Falls die Webanwendung international verwendet wird, sollte darauf geachtet werden, dass alle internationalen Zeichensätze auf allen logischen Ebenen der Webanwendung und von den angebundenen Hintergrundsystemen unterstützt werden.

Speicherung von Konfigurationsdateien außerhalb von Web-Root

Konfigurationsdateien der Webanwendung enthalten häufig schützenswerte Informationen wie z. B. Zugangsdaten. Daher dürfen Benutzer der Webanwendung keine Zugriffsmöglichkeiten auf die Konfigurationsdateien haben.

Aus diesem Grund sollten Konfigurationsdateien ausschließlich außerhalb des Webserver-Root-Verzeichnisses gespeichert werden. Außerhalb dieses Verzeichnisses werden in der Regel keine Daten von der Webanwendung ausgeliefert.

Grundsätzlich müssen Konfigurationsdaten außerhalb des Quelltextes in separaten Konfigurationsdateien gespeichert werden. Konfigurationseinstellungen, die vertrauliche Daten beinhalten, sollten zudem verschlüsselt werden.

Festlegung von Grenzwerten

Einige Schutzmechanismen sehen den Einsatz von Grenzwerten vor (siehe z. B. M 4.396 *Schutz vor unerlaubter automatisierter Nutzung von Webanwendungen*. Wird ein Grenzwert überschritten, erfolgt häufig die zeitweise Sperrung einer betroffenen Funktion oder Ressource. So können wiederholt fehlgeschlagene Anmeldeversuche die Sperrung des Benutzer-Kontos zur Folge haben (z. B. zur Abwehr von Brute-Force-Angriffen).

Auf diese Weise eingeleitete Maßnahmen können die Bedienung der Webanwendung beeinflussen und somit ebenfalls unbeteiligte Benutzer betreffen. Diese Benutzer können sich beispielsweise nicht mehr an der Webanwendung anmelden, falls ihr Benutzer-Konto gesperrt wurde.

Diese Auswirkungen sollten daher auch bei der Festlegung von Grenzwerten berücksichtigt werden.

Restriktive Dateisystemberechtigungen

Webanwendungen bieten Benutzern häufig direkt oder indirekt Zugriff auf das darunterliegende Dateisystem (z. B. über abrufbare Dateien oder eine Upload-Funktion). Damit ein Angreifer nicht unbefugt schützenswerte Dateien lesen oder manipulieren kann, sollten diese zusätzlich zu Zugriffsbeschränkungen auf Webanwendungsebene durch restriktive Dateisystemberechtigungen geschützt werden. Der Server, auf dem die Webanwendung läuft, muss mit eingeschränkten Rechten gestartet werden und nicht als Administrator (root).

Administration einer Webanwendung

Die Webanwendung sollte vorrangig über ein von der Anwendung entkoppeltes System administriert werden. Im Fall einer E-Commerce-Anwendung kann beispielsweise die Artikelpflege über ein getrenntes System mit Zugriff auf die Datenbank der Webanwendung erfolgen. Das System sollte idealerweise alleine für diesen Zweck bestimmt sein und keine direkte Verbindung zu der Webanwendung haben. Dementsprechend sollte die Webanwendung die Artikeldaten ausschließlich von der Datenbank abrufen.

Häufig bieten Webanwendungen zur Administration eine Web-Oberfläche auf demselben Server an. Diese Funktion sollte gemieden und stattdessen die Administration über ein separates System durchgeführt werden. Falls die Administration auf demselben Server erforderlich ist, sollte die Administrationsoberfläche ausschließlich aus dem Administrationsnetz heraus erreichbar und der Zugriff durch gewöhnliche Benutzer der Webanwendung nicht möglich sein. Möglichkeiten zur Administration der Webanwendung, die nicht genutzt werden (z. B. Konsole), sollten nicht nutzbar sein.

Prüffragen:

- Werden bei der Webanwendung ausschließlich die HTTP-Methoden zugelassen, die erforderlich sind?
- Wird zur Übertragung vertraulicher Daten (z. B. Formulardaten) vorzugsweise die HTTP-POST-Methode verwendet?
- Werden vertrauliche Daten ausschließlich über einen verschlüsselten Transportkanal übertragen?
- Wird verhindert, dass im Fall von Verbindungsfehlern bei einem verschlüsselten Kanal nicht auf eine unverschlüsselte Verbindung gewechselt wird?
- Werden Konfigurationsdateien der Webanwendung außerhalb des Web-Root-Verzeichnisses gespeichert?
- Wird für die Administration der Webanwendung ein separates System verwendet oder ist die Administrationsoberfläche der Webanwendung nur aus dem Administrationsnetzwerk heraus erreichbar?

M 4.399 Kontrolliertes Einbinden von Daten und Inhalten bei Webanwendungen

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Entwickler, Administrator

Eine Webanwendung erstellt zur Laufzeit Webseiten, deren Inhalte sich aus unterschiedlichen Quellen zusammensetzen können. Diese Inhalte werden z. B. in Form von Dateien dynamisch bei der Erstellung der Webseite eingebunden oder von der Webanwendung generiert. Da dem Benutzer die fertige Webseite ausgeliefert wird, ist für ihn häufig nicht ersichtlich, aus welcher Quelle die angezeigten Inhalte stammen. Daher muss die Webanwendung sicherstellen, dass ausschließlich vorgesehene Daten und Inhalte eingebunden und an den Benutzer ausgeliefert werden.

Die Inhalte können mittels unterschiedlicher Techniken eingebunden werden. Daher werden in den folgenden Abschnitten Hinweise zur sicheren Verwendung üblicher Techniken zusammengefasst.

Einbinden von Dateien (File Inclusion)

Häufig werden bei der Generierung von Webseiten durch die Webanwendung Teile der ausgelieferten Internet-Seite aus unterschiedlichen Dateien dynamisch eingebunden (z. B. eine Navigationsleiste). Hierdurch verringert sich der Wartungsaufwand bei Änderungen an der Webseite (z. B. ein neuer Navigations-Eintrag). Dabei sollten der Inhalt und der Pfad der einzubindenden Dateien ausschließlich vom Administrator oder von privilegierten Benutzern der Webanwendung geändert werden können. Gewöhnlichen Benutzern sollte es dagegen nicht möglich sein, die Dateien zur Einbindung frei zu wählen oder zu modifizieren (z. B. über veränderte Parameter). Aus diesem Grund sollte die Verarbeitung von Benutzer-Eingaben zur Einbindung von Dateien grundsätzlich vermieden werden.

Erfordert die Webanwendung Benutzer-Eingaben als Quelle zur Einbindung von Dateien, sollten die vorgesehenen Pfadangaben zu den Quell-Dateien nicht frei wählbar sein. Benutzer sollten nicht in der Lage sein, den gesamten Pfad vorzugeben, sondern stattdessen sollten Benutzer-Eingaben in vordefinierte Pfadangaben gekapselt werden.

Angriffe, wie Path Traversal, versuchen durch relative Bezüge den Pfad auf schützenswerte Dateien umzusetzen (z. B. */../../../etc/passwd*) und so aus den vorgegebenen Pfadangaben auszubrechen. Zur Verhinderung derartiger Angriffe sollten daher die Benutzereingaben auf unerwünschte Zeichen zur Manipulation des Pfades (z. B. "/.." und "\..") gefiltert werden (siehe auch M 4.393 *Umfassende Ein- und Ausgabevalidierung bei Webanwendungen und Web-Services*).

Bei der Auswahl der Quell-Dateien können Indizes anstelle von Dateinamen verwendet werden, denen serverseitig hinterlegte Dateinamen zugeordnet werden. Somit hat ein Angreifer keinen direkten Einfluss auf den Dateinamen und kann durch Manipulation des Index keine beliebigen Inhalte direkt einbinden.

Webanwendungen können, neben Dateien auf dem Serversystem, auch entfernt gespeicherte Ressourcen über die Netzwerkverbindung mittels URL ein-

binden (Remote File Inclusion). Nach Möglichkeit sollte das Einbinden entfernter Inhalte komplett unterbunden werden. Kann auf die Einbindung externer Inhalte nicht verzichtet werden, so muss die vertrauenswürdige Herkunft dieser Dateien unbedingt sichergestellt werden (z. B. auf Basis einer Whitelist mit der Limitierung auf einen Server oder eine Auflistung von absoluten URLs).

Verwendung von Datei-Uploads

Bei vielen Webanwendungen kann der Benutzer Inhalte mittels einer Upload-Funktion übermitteln. Ein typischer Anwendungsfall ist der Upload eines Profilfotos. Die hochgeladenen Daten sind auf die benötigten Dateiformate zu beschränken (z. B. sollten für das Profilfoto ausschließlich Bilddateien zugelassen werden). Hierbei sollte neben der Prüfung der Dateiendung ebenfalls der Inhalt der Datei, z. B. durch eine Auswertung des Dateiheaders, geprüft werden.

Hochgeladene Dateien sollten nach Möglichkeit in einem Verzeichnis gespeichert werden, welches nicht über die Web-Schnittstelle erreichbar ist (z. B. außerhalb des Wurzelverzeichnisses des Webservers). So wird verhindert, dass ein Benutzer auf seine hochgeladenen Dateien direkt zugreifen kann (z. B. auf schadhafte Skripte). Werden die hochgeladenen Dateien zunächst in einem temporären Verzeichnis gespeichert, so ist sicherzustellen, dass andere Benutzer nicht unerlaubt auf die Datei zugreifen dürfen.

Stellt eine Webanwendung dem Benutzer eine Upload-Funktion von Dateien zur Verfügung, so sind folgende Punkte zu beachten:

- Die Funktionalität sollte möglichst nur angemeldeten Benutzern zur Verfügung stehen.
- Hochgeladene Dateien dürfen nicht im Wurzelverzeichnis des Webserver-Dienstes gespeichert werden. Es sollten entweder feste Verzeichnisstrukturen vorgegeben werden, in denen Ordner und Dateien angelegt werden können oder die Speicherung in einem anderen Kontext (wie z. B. in einer Datenbank oder einem fest vorgegebenen Pfad) erfolgen. Ein Angreifer sollte nicht aus dem vorgegebenen Kontext ausbrechen können.
- Der vorgegebene Pfad zur Speicherung der hochgeladenen Dateien darf nicht von den Benutzern geändert werden können.
- Zum Schutz vor Denial-of-Service-Angriffen sollte die Dateigröße begrenzt werden.
- Die Berechtigungen hochgeladener Dateien sollten restriktiv gesetzt sein, um einen unberechtigten Zugriff zu verhindern. Auf diese Weise soll unterbunden werden, dass hochgeladene Dateien eines Angreifers ausgeführt werden.
- Ein Virenschutzprogramm sollte die hochgeladenen Dateien auf Schadsoftware untersuchen.
- Die Wahl des Dateinamens sollte wie folgt eingeschränkt werden:
- Der Dateiname mit der Dateiendung sollte auf eine feste Anzahl von Zeichen begrenzt werden (z. B. 200 Zeichen).
- Alle nicht sichtbaren Zeichen (z. B. Steuerzeichen) und alle kodierten Varianten dieser Zeichen sollten vom Dateinamen entfernt werden (z. B. Unicode).
- Alle Zeichen mit einer spezifischen Bedeutung für Interpreter sollten entfernt werden (z. B. ; : > < / \ . * % $).
- Falls möglich sollten ausschließlich alphanumerische Zeichen und der Punkt für die Dateiendung erlaubt sein.

Einbinden von Inhalten aus übergebenen Parametern

Webanwendungen nehmen häufig Eingaben in Form von Parametern (z. B. aus Formularen) entgegen, verarbeiten diese und stellen sie erneut in der Rückantwort dar (z. B. der Suchbegriff bei einer Websuche). Ein Angreifer kann dies ausnutzen, um über ausgewählte Eingaben die Darstellung der Webseite zu manipulieren. Daher müssen alle von der Webanwendung zur Darstellung von Webseiten verwendeten Parameter gemäß M 4.393 *Umfassende Ein- und Ausgabevalidierung bei Webanwendungen und Web-Services* validiert werden.

Sicheres Weiterleiten von Requests (Redirect)

Die Weiterleitungsfunktion einer Webanwendung sollte nicht beliebige Webseiten als Weiterleitungsziel zulassen, sodass Benutzer ausschließlich auf vertrauenswürdige, vorgesehene Webseiten weitergeleitet werden. So sollte vermieden werden, dass Benutzer beispielsweise über einen präparierten Link auf die Weiterleitungsfunktion der Webanwendung auf eine Phishing-Seite geführt werden.

Die folgenden Punkte geben Hinweise zu Einschränkungsmöglichkeiten von Weiterleitungszielen.

- Beschränkung auf lokale Seiten
  Wenn keine Weiterleitung auf externe Webseiten erfolgen muss, kann das Weiterleitungsziel auf externe Adressen geprüft und nur lokale Seiten zugelassen werden. Hierbei sollten ausschließlich relative Pfadangaben auf Ziele innerhalb der Webanwendung als Eingabe zugelassen und der notwendige Host-Teil nachträglich statisch hinzugefügt werden.
- Vordefinierte Weiterleitungsziele
  Ist eine Weiterleitung ausschließlich auf bekannte, statische Ziele vorgesehen, sollten diese serverseitig in einer vordefinierten Liste mit Indizes hinterlegt werden. Den Zielen werden somit statische Index-Werte zugeordnet. Anstelle der Zieladresse übergibt der Client einen Index-Wert (z. B. aus einer Auswahlliste eines Formulars), der serverseitig einer Zieladresse aus der Liste zugeordnet wird.
- Manuelle Bestätigung
  Der Benutzer muss vor der Weiterleitung die Zieladresse und somit die Vertrauenswürdigkeit des Weiterleitungsziels prüfen und bestätigen (z. B. über eine eingeblendete Weiterleitungsseite). Hiermit wird der Benutzer vor dem Verlassen der Webanwendung und damit des Sicherheitskontextes gewarnt.
- Referrer-Test
  Das Refferer-Feld des HTTP-Requests kann von der Weiterleitungsfunktion als zusätzliches Merkmal für die bestimmungsgemäße Nutzung geprüft werden. Eine Weiterleitung sollte nur dann erfolgen, wenn das Referrer-Feld die Adresse zu einer Webseite der Webanwendung mit einem Verweis auf das Weiterleitungsziel enthält.

Einbindung von Inhalten Dritter

Von Partnern eingebundene Daten und Inhalte (z. B. Werbeeinblendungen) sollten grundsätzlich als weniger vertrauenswürdig eingestuft werden. Es wird daher eine starke Kontrolle dieser Inhalte empfohlen, da die Gefahr besteht, dass Schadcode oder nicht vertrauenswürdige Inhalte eingebettet werden.

Prüffragen:

- Wird durch die Webanwendung eine Manipulation einzubindender Ressourcen verhindert (z. B. File Inclusion und Remote File Inclusion)?
- Wird das Hochladen von Dateien über eine mögliche Upload-Funktion der Webanwendung eingeschränkt (z. B. auf notwendige Dateitypen) und werden die Zugriffs- und Ausführrechte restriktiv gesetzt?
- Wird ein Ausbruch aus dem vorgegebenen Pfad zur Speicherung von Dateien unterbunden (z. B. durch Path Traversal)?
- Werden die Ziele der Weiterleitungsfunktionen einer Webanwendung ausreichend eingeschränkt (z. B. nur lokale Seiten) und der Benutzer beim Verlassen der Vertrauensdomäne informiert?

M 4.400 Restriktive Herausgabe sicherheitsrelevanter Informationen bei Webanwendungen und Web-Services

Verantwortlich für Initiierung: Fachverantwortliche, Verantwortliche der einzelnen Anwendungen
Verantwortlich für Umsetzung: Administrator, Entwickler

Webseiten und Rückantworten von Webanwendungen und Web-Services können sicherheitsrelevante Informationen beinhalten, mit deren Hilfe Angreifer Sicherheitsmechanismen umgehen und Schwachstellen ausnutzen können. Daher dürfen keine sicherheitsrelevanten Informationen angezeigt werden, die nicht zwingend für den Betrieb und die Nutzung der Webanwendung oder des Web-Service notwendig sind.

Die folgenden Beispiele verdeutlichen, welche Informationen sicherheitsrelevante Hinweise enthalten können und wie verhindert werden kann, dass diese offengelegt werden.

Keine sicherheitsrelevanten Informationen in Fehlermeldungen

Tritt bei der Bedienung der Webanwendung oder des Web-Service ein Fehler auf (zum Beispiel Zugriffsfehler), sollten dem Benutzer neutrale Fehlermeldungen übermittelt werden. Die Fehlermeldungen dürfen keine direkten Rückschlüsse auf eingesetzte Techniken, Sicherheitsmechanismen und Zustände der Webanwendung ermöglichen.

Die folgenden Beispiele zeigen Informationen, die nicht in Fehlermeldungen enthalten sein sollten:

- Stacktraces und Debugging-Informationen,
- Meldungen wie "Benutzername ungültig" oder "Passwort ungültig" (anstelle von allgemeinen Fehlermeldungen wie "Benutzername oder Passwort ungültig"),
- von Hintergrundsystemen weitergereichte Fehlermeldungen wie zum Beispiel SQL-Fehlermeldungen einer Datenbank statt einer Meldung "Fehler bei der Überprüfung der Zugangsdaten",
- Fehlercodes statt zum Beispiel der Meldung "Ein Fehler ist aufgetreten".

Im Fall einer fehlgeschlagenen Authentisierung sollte beispielsweise unabhängig von der Gültigkeit des Benutzernamens stets eine allgemeingültige Meldung wie "Falsche oder ungültige Zugangsdaten" ausgegeben werden, damit ein Angreifer nicht auf die Existenz von Benutzerkonten rückschließen kann (*user enumeration*).

Grundsätzlich kann unterschiedlicher HTML-Code zur gleichen Ausgabe im Webbrowser führen. Beispielsweise werden zwei HTML-Seiten mit einer unterschiedlichen Anzahl von Leerzeichen im Browser gleich dargestellt, obwohl sie sich im HTML-Code unterscheiden. Es ist daher darauf zu achten, dass die Fehlermeldungen nicht nur in der Darstellung im Browser, sondern auch im HTML-Code identisch sind. Hiermit soll verhindert werden, dass ein Angreifer aufgrund eines veränderten HTML-Codes auf die Gültigkeit von Teil-Eingaben (zum Beispiel gültiger Benutzername bei falschem Passwort) schließen kann.

Weitere Informationen zur Fehlerbehandlung finden sich in M 4.395 *Fehlerbehandlung durch Webanwendungen und Web-Services*.

Vermeidung von sicherheitsrelevanten Kommentaren in ausgelieferten Webseiten oder Web-Service-Antworten

Bei der Entwicklung von Webanwendungen werden möglicherweise Kommentare in den HTML-Code geschrieben. Diese Kommentare können sicherheitsrelevante Informationen (zum Beispiel Todo-Listen, Versionsnummern, Zugangsdaten oder uninterpretierter Quellcode) enthalten, die als HTML-Kommentare in der Webseite vom Benutzer leicht eingesehen werden können. Auch die Rückantworten von Web-Services können Kommentare mit sicherheitsrelevanten Informationen enthalten, beispielsweise Kommentare in XML-Antworten eines SOAP-Dienstes. Aus diesem Grund ist darauf zu achten, dass in den Kommentaren keine sicherheitsrelevanten Informationen enthalten sind. Idealerweise sollten in in den ausgelieferten Webseiten oder Rückantworten einer produktiven Webanwendung oder eines Web-Service keine Kommentare verwendet werden.

Eingeschränkter Zugriff auf Dokumentation

Informationen in der Dokumentation einer Webanwendung oder eines Web-Service (zum Beispiel Dokumente zur Administration der Webanwendung) können einem Angreifer auf potentielle Schwachstellen (zum Beispiel Standardbenutzer nach der Installation) hinweisen und missbraucht werden, um Angriffe vorzubereiten. Daher sollte verzichtbare Dokumentation zur Webanwendung oder zum Web-Service und den zugehörigen Komponenten (zum Beispiel Datenbank) gelöscht werden. Ist die Dokumentation online verfügbar, so sollte ausschließlich der entsprechende Adressatenkreis darauf zugreifen können. Beispielsweise sollte die Dokumentation zur Administration einer Webanwendung oder eines Web-Service nicht aus dem Internet heraus erreichbar sein.

Löschen nicht benötigter Dateien

Im laufenden Betrieb einer Webanwendung oder eines Web-Service fallen häufig Dateien an, die nicht für den produktiven Betrieb benötigt werden (zum Beispiel temporäre Dateien, oder Backup-Dateien). Diese Dateien können sicherheitskritische Informationen beinhalten (zum Beispiel Test-Ergebnisse) oder Funktionen anbieten (zum Beispiel Testwerkzeuge zur Ermittlung von Versionsnummern der eingesetzten Bibliotheken), die für Angriffe auf die Webanwendung genutzt werden können.

Darüber hinaus ist zu beachten, dass insbesondere bei temporären Dateien oder Backup-Dateien häufig andere Dateiendungen (zum Beispiel *.bak-Dateien als Sicherheitskopien eines Editors) verwendet werden. Werden diese Dateien vom Webserver abgerufen, wäre es möglich, dass die Dateien aufgrund der unbekannten Dateiendung nicht mehr interpretiert werden und stattdessen der Quelltext der Webanwendung ausgeliefert wird.

Versionsverwaltungssysteme legen zumeist Dateien oder Ordnerstrukturen für die von ihnen verwalteten Objekte an (zum Beispiel Ordner wie .svn oder .git). Diese Dateien oder Ordner enthalten häufig detaillierte Informationen zu den verwalteten Projekten und ermöglichen unter Umständen einen kompletten Zugriff auf den Quellcode. Aus diesem Grund sollten Anwendungen oder Anwendungskomponenten grundsätzlich nicht über die Versionsverwaltung auf Produktivsysteme aufgespielt werden. Zumindest sollte der Zu-

griff auf von der Versionsverwaltungssoftware angelegte Dateien und Ordner blockiert werden.

Aus den genannten Gründen sind alle Dateien zu löschen, die für den produktiven Betrieb nicht benötigt werden. Darüber hinaus sollte regelmäßig kontrolliert werden, ob neue Dateien angefallen sind und ob diese gelöscht werden können. Ist dies nicht möglich, kann der Zugriff auf diese Dateien gesperrt werden.

Sichere Erfassung durch externe Suchmaschinen

Suchmaschinen setzen sogenannte Agenten (auch Robots oder Crawler genannt) ein, um neue oder geänderte Inhalte im Netz zu indizieren. Diese Agenten können durch die Datei *robots.txt* im Wurzelverzeichnis der Webanwendung instruiert werden, ausgewiesene Ressourcen (zum Beispiel Pfade) der Webanwendung zu ignorieren. Auf diese Weise können schützenswerte Informationen von der Indizierung in der Suchmaschine ausgenommen werden. Die vertraulichen Ressourcen (zum Beispiel Verzeichnis-Pfade) sollten in der Datei *robots.txt* unter der Direktive "Disallow" aufgeführt werden. So werden die Agenten veranlasst, die gelisteten Ressourcen nicht zu indizieren.

Damit die Einträge in der Datei *robots.txt* einem Angreifer keine Hinweise auf sicherheitskritische Ressourcen der Webanwendung geben, sollten alle zu schützenden Verzeichnisse nach Möglichkeit in einem gesonderten Verzeichnis der Webanwendung zusammengefasst werden. Ausschließlich dieses Verzeichnis sollte in die Datei *robots.txt* eingetragen werden, sodass diese keine internen Verzeichnisstrukturen mit sicherheitsrelevanten Informationen enthält.

Vermeidung von Produkt- und Versionsangaben

Häufig enthalten Antworten und Ausgaben der einzelnen Komponenten der Webanwendung Angaben zu Produktnamen und Versionsnummern. Diese Informationen können zum Beispiel in HTTP-Headern oder in Kommentaren im HTML-Quelltext der ausgelieferten Webseiten, aber auch in XML- oder JSON-Antworten von Web-Services enthalten sein. Auf der Grundlage dieser Angaben kann ein Angreifer gezielt nach bekannten Schwachstellen des Produkts suchen und über diese die Webanwendung oder den Web-Service angreifen. Daher sollten Angaben zu verwendeten Produkten und Versionen vermieden werden (zum Beispiel Applikationsframework, Webserver).

Verzicht auf absolute Pfadangaben

Absolute Pfadangaben ermöglichen oft Rückschlüsse auf die interne Struktur und den Aufbau der Webanwendung. So kann beispielsweise der Speicherort schützenswerter Informationen ermittelt werden. Daher sollten nach Möglichkeit keine absoluten Pfadangaben der Webanwendung oder des Web-Service veröffentlicht werden.

Prüffragen:

- Werden ausschließlich Informationen veröffentlicht, die für den Betrieb oder die Nutzung der Webanwendung oder des Web-Service erforderlich sind?
- Werden von der Webanwendung oder dem Web-Service ausschließlich neutrale Fehlermeldungen ausgegeben und sind diese im Quelltext identisch?

- Werden sicherheitsrelevante Informationen in Webseiten (zum Beispiel in Kommentaren) oder Web-Service Antworten vor der Auslieferung an die Benutzer gelöscht?
- Ist nur dem entsprechenden Adressatenkreis der Zugriff auf sicherheitsrelevante Dokumentation der Webanwendung oder des Web-Service möglich?
- Werden vor der produktiven Inbetriebnahme alle Dateien gelöscht, die nicht für den Betrieb der Webanwendung oder des Web-Service notwendig sind, und wird eine entsprechende Prüfung auf nicht benötigte Dateien regelmäßig durchgeführt?
- Enthält die Datei robots.txt ausschließlich URLs, die keine sicherheitsrelevanten Informationen enthalten?

M 4.401 Schutz vertraulicher Daten bei Webanwendungen

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Verantwortliche der einzelnen Anwendungen
Verantwortlich für Umsetzung: Entwickler, Administrator

Bei Webanwendungen werden Daten sowohl auf dem Server (z. B. in einer Webanwendung) als auch auf den Clients (z. B. im Browser) gespeichert und dabei über Netze übertragen. Hierbei kann es sich um vertrauliche Bankdaten, wie Kreditkarteninformationen oder Überweisungen, handeln. Daher müssen Maßnahmen getroffen werden, damit diese Daten nicht unbefugt eingesehen oder manipuliert werden können.

Allgemeine Aspekte

Werden vertrauliche Daten durch die Webanwendung verarbeitet, übertragen oder gespeichert (server- wie auch clientseitig), sollten sie durch kryptographische Verfahren geschützt werden. Auch wenn die Webanwendung kompromittiert ist, sollten die eingesetzten kryptographischen Verfahren diese Daten weiterhin schützen.

Vertrauliche Daten einer Webanwendung sind z. B.:

- Zugangsdaten (z. B. Benutzer, Passwort),
- Authentisierungsdaten (z. B. SessionID),
- kritische Daten, die von der Webanwendung verarbeitet werden (z. B. Zahlungsinformationen oder Gesundheitsdaten).

Kryptographische Verfahren können bei der Verarbeitung, Übertragung und Speicherung dieser Daten durch die Webanwendung und den Clients verwendet werden. Sie können dabei z. B. wie folgt eingesetzt werden:

- Verschlüsselung von Daten,
- Sichere Speicherung von Zugangsdaten,
- Schutz des Transportkanals.

Es ist darauf zu achten, kryptographische Algorithmen für den jeweiligen Einsatzzweck auszuwählen, die dem Stand der Technik entsprechen und keine bekannten Schwachstellen aufweisen (siehe hierzu M 2.164 *Auswahl eines geeigneten kryptographischen Verfahrens*). Die kryptographischen Algorithmen sollten serverseitig umgesetzt sein.

Eine besondere Bedeutung bei der Kryptographie kommt den verwendeten Schlüsseln zu. Diese müssen je nach Einsatzgebiet über eine gewisse Mindestlänge verfügen und verschiedenen mathematischen Anforderungen (z. B. Komplexität) genügen. Zudem muss für einen entsprechend sicheren Transport beziehungsweise Austausch von Schlüsseln gesorgt werden. Gleiches gilt auch für deren Speicherung. Bei der Gestaltung einer Webanwendung sollten diese Punkte geregelt und in einem Kryptokonzept zusammengefasst werden (siehe B 1.7 *Kryptokonzept*).

Für Webanwendungen mit hohem Schutzbedarf kann zusätzlich eine Absicherung der Nutzdaten erforderlich sein. Werden beispielsweise Sozialdaten mit hohen Anforderungen an die Vertraulichkeit von der Webanwendung verarbeitet, können diese Daten von der Webanwendung vor der Speicherung verschlüsselt werden. So kann sichergestellt werden, dass auch bei einem di-

rekten Zugriff auf die Datenbank (z. B. durch Datenbankadministratoren) keine verwertbaren Daten ausgelesen werden können.

Werden vertrauliche Daten übertragen, können sie durch einen sicheren Transportkanal vor dem unbefugten Einsehen oder der Manipulation geschützt werden. Bevor vertrauliche Daten übertragen werden, sollte daher zu einer gesicherten Verbindung gewechselt werden. Auch nach der Anmeldung eines Benutzers sollten die übertragenen Daten weiterhin durch eine gesicherte Verbindung geschützt werden. Der Transportkanal wird hierzu üblicherweise durch den Einsatz von SSL/TLS abgesichert (siehe M 5.66 *Clientseitige Verwendung von SSL/TLS*L).

Schutz clientseitig gespeicherter Daten

Die zwischen dem Client und der Webanwendung ausgetauschten Daten können vom Client im lokalen Browsercache zwischengespeichert werden. Wenn der Browser die Daten über die Sitzungsdauer des Webanwendungsbenutzers hinaus im Cache speichert, können diese von Personen mit Zugriff auf den Rechner des Benutzers und von Skripten und Browser-Plugins ohne zusätzliche Zugriffskontrolle aus dem Cache abgerufen werden.

Das clientseitige Zwischenspeichern (Cachen) von vertraulichen Daten der Webanwendung kann durch folgende Direktiven in den HTTP-Headern der Webanwendung unterbunden werden:

- *Cache-Control: no-cache, no-store*
- *Pragma: no-cache*
- Expires: -1

Da der Web-Browser üblicherweise nicht unter der Kontrolle des Betreibers der Webanwendung steht, kann somit nicht vollständig ausgeschlossen werden, dass Daten trotzdem zwischengespeichert werden. Daher kann es für Webanwendungen mit hohem Schutzbedarf zusätzlich erforderlich sein, dass der Benutzer den Browsercache während der Bedienung der Webanwendung deaktiviert oder ihn löscht, sobald er seine Tätigkeiten an der Webanwendung beendet hat. In diesem Fall kann dem Benutzer beispielsweise nach erfolgter Abmeldung ein Hinweis für das Löschen des Browsercaches angezeigt werden. Dies betrifft insbesondere Webanwendungen, die von öffentlichen IT-Systemen aus genutzt werden. Alternativ kann der Benutzer auf die Verwendung des Private Modes des Browsers hingewiesen werden, bei dem keine Daten über die Sitzung zwischengespeichert werden.

Häufig werden bei der Bedienung einer Webanwendung Daten in Cookies auf dem Client gespeichert. Bei jedem Zugriff auf die Webanwendung werden diese Cookies transparent für den Benutzer an die Webanwendung übermittelt. Dabei kann es sich auch um schützenswerte Daten wie die SessionID handeln. Der Zugriff auf Cookies mit vertraulichen Daten sollte daher so weit wie möglich eingegrenzt werden. Wenn Cookies durch die Webanwendung erstellt werden, sollten folgende Cookie-Flags gesetzt sein:

- *Domain*
  Das Cookie-Flag sollte nicht gesetzt werden, denn dann werden per Default nur Anfragen der Domain beantwortet, die das Cookie gesetzt hat. Sollte es notwendig sein, dies auch anderen (Sub-)Domains zu ermöglichen, dann sollte die Domäne so weit wie möglich eingeschränkt werden, ohne die Funktionalität der Webanwendung einzuschränken (z. B. webapp.domain.tld anstatt domain.tld).
- *Path*

Das Path-Attribut beschränkt die Gültigkeit des Cookies auf einen festgelegten Pfad der Webanwendung. Auch das Path-Attribut sollte so weit wie möglich eingeschränkt werden, ohne die Funktionalität der Webanwendung einzuschränken (z. B. /webapp/ anstelle von /).

- *Secure*
  Ist die Direktive Secure gesetzt, so wird das Cookie ausschließlich über verschlüsselte Kommunikationskanäle übertragen, wie z. B. über SSL/ TLS.
- *HttpOnly*
  Diese Direktive verhindert, dass clientseitige Skripte auf das Cookie zugreifen (z. B. JavaScript). Es ist zu beachten, dass dieses Attribut nicht von allen Browsern unterstützt wird.

Das folgende Beispiel zeigt die Anweisung zur Erstellung eines Cookies unter Verwendung genannter Direktiven:

Set-Cookie: SESSIONID=sl342kdfjslaal39skdj; path=/webapp; secure; HttpOnly

Bei der Authentisierung des Benutzers gegenüber einer Webanwendung wird gewöhnlich ein HTML-Formular verwendet, in das der Benutzername und das Passwort eingegeben werden. Wenn der Benutzer sein Passwort in das Passwortfeld eintippt, sollte es nicht im Klartext wiedergegeben, sondern durch sogenannte Wildcards ersetzt werden (z. B. Sterne oder Punkte). Hierfür muss in der Formulardefinition der Passwort-Feld-Typ ausgewählt werden (*type="password"*).

Darüber hinaus kann der Web-Browser angewiesen werden, vertrauliche Formulardaten (z. B. den Benutzernamen und das Passwort) nicht zwischenzuspeichern und beim nächsten Aufruf des Formulars als Auswahl vorschlagen. Die Option *autocomplete="Off"* sollte hierfür bei der Definition des Formulars im Formularkopf gesetzt werden.

Während der Sitzung eines Benutzers an einer Webanwendung müssen in der Regel benutzerspezifische Daten gespeichert werden (z. B. die Artikel im Warenkorb). Diese Daten können dabei nicht nur serverseitig, sondern auch clientseitig in einem Cookie oder im Web-Storage des Browsers gespeichert werden. Grundsätzlich sollte vermieden werden, vertrauliche Daten an den Client zu übertragen oder auf dem Client zu speichern, da die Webanwendung keinen Einfluss auf den Schutz von clientseitig hinterlegten Daten hat. So können vom Browser auf dem Client umgesetzte Sicherheitsmechanismen zum Schutz der Daten häufig umgangen werden (z. B. durch direkten Zugriff auf das Dateisystem durch lokale Benutzer oder durch Cross-Site Scripting). Stattdessen sollten vertrauliche Daten grundsätzlich serverseitig gespeichert werden und ausschließlich das Identifikationsmerkmal des Benutzers (z. B. die SessionID) clientseitig hinterlegt sein.

Falls nicht vermieden werden kann, dass Sitzungsdaten clientseitig gespeichert werden, sollten diese Daten verschlüsselt und vor der Verarbeitung durch die Webanwendung auf Integrität geprüft werden. Damit wird sichergestellt, dass die Daten während der Übertragung nicht unbefugt eingesehen oder unbemerkt manipuliert werden können.

Darüber hinaus sollten die Daten vorzugsweise nur über den Zeitraum der Sitzung und nicht persistent gespeichert werden. Beim Web-Storage-Mechanismus sollte daher das sessionStorage-Objekt vor dem localStorage-Objekt bevorzugt werden.

Schutz serverseitig hinterlegter Daten

Sollen sich Benutzer an der Webanwendung anmelden können, müssen Zugangsdaten auf der Webanwendung gespeichert werden. Damit auch nach einer möglichen Kompromittierung der Webanwendung durch einen Angreifer die Zugangsdaten geschützt sind, dürfen sie nicht im Klartext gespeichert werden. Stattdessen sollten sie mithilfe von zeitgemäßen kryptographischen Algorithmen als salted Hashes hinterlegt werden, bei denen eine zufällige Zeichenfolge an den Klartext angehängt wird. Hierbei sollte für jedes Passwort ein unterschiedlicher, zufälliger Salt verwendet werden.

Darüber hinaus sollten die Zugangsdaten serverseitig auf einem vertrauenswürdigen IT-System (z. B. auf dem der Webanwendung) und in einem geschützten Bereich (z. B. außerhalb des Web-Root-Verzeichnisses oder in separaten Datenbanktabellen) hinterlegt sein. Die Zugangsdaten sollen nicht im Quelltext der Webanwendung (Hardcoded Passwords) gespeichert werden.

Darüber hinaus sollte ausschließlich die Webanwendung mit Schreibrechten auf die Zugangsdaten zugreifen können. Die Zugangsdaten sollten nur durch den Benutzer und über die vorgesehenen Schnittstellen und Funktionen der Webanwendung geändert werden können, sodass es Benutzern nicht möglich ist, Zugangsdaten unbefugt über z. B. den direkten Zugriff auf das Dateisystem zu lesen, zu ändern oder zu löschen.

Ruft ein Benutzer eine Webseite von der Webanwendung auf, so wird die Seite in der Regel von der Anwendung zur Laufzeit erstellt. Die aufgerufene Datei enthält Code, der von der Webanwendung vor der Auslieferung interpretiert wird und eine Webseite zurückliefert. Diese Webseite wird an den Benutzer übermittelt.

Üblicherweise werden anhand der Datei-Endung diese Dateitypen einem Interpreter oder Parser zugeordnet. Ändert sich die Datei-Endung, werden diese möglicherweise an den Benutzer übertragen, ohne vorher interpretiert zu werden. Werden derartige Dateien abgerufen, erhält der Benutzer Einsicht in die Programmlogik und vertrauliche Informationen, die gegebenenfalls im Code gespeichert sind. Dies kann beliebige Dateien betreffen, deren Datei-Endung nicht einem Interpreter oder Parser zugeordnet ist. Beispiele für anfällige Dateien sind:

- temporäre Dateien (z. B. temp.tmp),
- Backup Daten (z. B. backup.bak),
- Konfigurationsdateien (z. B. config.conf),
- Einzubindende Dateien (z. B. include.inc).

Die Dateien können vertrauliche Informationen, wie Zugangsdaten, enthalten, die hierüber bei unzureichender Zugriffsbeschränkung abgerufen werden können.

Daher sollten Dateien, die nicht für die Interpretation und den direkten Abruf durch den Benutzer vorgesehen sind, von der Webanwendung nicht ausgeliefert werden. Zusätzlich sollten die Dateisystemberechtigungen auf diese Dateien restriktiv gesetzt sein. Nicht mehr benötigte Dateien sollten zeitnah gelöscht werden (siehe auch M 4.400 *Restriktive Herausgabe sicherheitsrelevanter Informationen bei Webanwendungen und Web-Services*, Absatz *Löschen nicht benötigter Dateien).*

Prüffragen:

- Verwendet die Webanwendung sichere, kryptographische Algorithmen zum Schutz der Daten und werden diese serverseitig auf einem vertrauenswürdigen IT-System umgesetzt?
- Übermittelt die Webanwendung vertrauliche Daten über einen geschützten Transportkanal (z. B. SSL/TLS)?
- Sind Direktiven in den HTTP-Headern der Webanwendung vorgesehen, die ein clientseitiges Zwischenspeichern vertraulicher Daten verhindern?
- Setzt die Webanwendung Cookie-Flags zum Schutz der Cookies vor unbefugter Einsicht?
- Sind Formularfelder der Webanwendung so konfiguriert, dass vertrauliche Formulardaten (z. B. das Passwort) nicht im Klartext angezeigt oder vom Browser gespeichert werden?
- Werden Zugangsdaten der Webanwendung serverseitig mithilfe von kryptographischen Algorithmen vor unbefugtem Zugriff geschützt (Salted Hash)?
- Wird der Abruf von Dateien unterbunden, die Quelltexte der Webanwendung enthalten?

M 4.402 Zugriffskontrolle bei Webanwendungen

Verantwortlich für Initiierung: Leiter Fachabteilung, Verantwortliche der einzelnen Anwendungen
Verantwortlich für Umsetzung: Entwickler, Administrator

Die Autorisierungskomponente einer Webanwendung muss sicherstellen, dass Benutzer nur solche Aktionen durchführen können, für die sie über ausreichende Berechtigungen verfügen. Die Zuweisung von Rechten kann dabei auf der Grundlage von Benutzer-Rollen erfolgen.

Die Autorisierungskomponente sollte alle verwalteten Ressourcen einer Webanwendung berücksichtigen. Dazu zählen z. B.

- URLs,
- Dateien,
- Objektreferenzen,
- Geschäftsfunktionen,
- Anwendungsdaten,
- Konfigurationsdaten und
- Protokolldaten.

Eine Zugriffskontrolle ist möglichst auf allen Ebenen einer Webanwendung (z. B. durch die Webanwendung, den Applikationsserver, den Webserver und das Betriebssystem) umzusetzen. Demzufolge sollten neben einem Zugriffsschutz auf Webanwendungs-Ebene die Maßnahmen M 4.94 *Schutz der Webserver-Dateien* als auch M 5.168 *Sichere Anbindung von Hintergrundsystemen an Webanwendungen und Web-Services* für den Zugriffsschutz von Daten auf dem Webserver und in Hintergrundsystemen berücksichtigt werden.

Folgende Punkte sollten für eine sichere Zugriffskontrolle auf Webanwendungs-Ebene berücksichtigt werden:

Generelle Aspekte

Berechtigungen sollten restriktiv und nach dem Minimalprinzip vergeben werden. Die Benutzer der Webanwendung sollten daher nur über solche Rechte verfügen, die sie zur Bewältigung ihrer Aufgaben unbedingt benötigen.

Jeder Zugriff auf geschützte Inhalte und Funktionen sollte kontrolliert werden, bevor er ausgeführt wird. Dies gilt auch dann, wenn beispielsweise der gleiche Benutzer auf eine geschützte Ressource wiederholt zugreift. Auch automatisierte Client-Requests durch Web-Technologien (z. B. Ajax) sollten als unabhängige Requests behandelt und entsprechend kontrolliert werden.

Anforderungen an die Autorisierungskomponente

Die Nutzung von Webanwendungen erfolgt gewöhnlich über einen generischen Client, der nicht unter der Kontrolle der Webanwendung steht. Damit kann ein Angreifer die Anfrage grundsätzlich beliebig manipulieren und clientseitig umgesetzte Sicherheitsmechanismen umgehen. Aus diesem Grund muss die Autorisierung serverseitig auf einem vertrauenswürdigen IT-System umgesetzt werden.

Die Routinen zur Autorisierung sollten zentral an einer Stelle und nicht verteilt im Programmcode der Webanwendung umgesetzt werden. Auf diese Weise wird der Code der Autorisierungskomponente von der Geschäftslogik der

Webanwendung getrennt und eine redundante und fehleranfällige Umsetzung vermieden. Bei der Entwicklung der Autorisierungskomponente sollte nach Möglichkeit auf Funktionen aus bereits existierenden Frameworks zurückgegriffen werden.

Kommt es zu Fehlern während der Zugriffskontrolle (z. B. weil unzureichende Informationen für die Autorisierung verwendet werden), müssen Zugriffe abgelehnt werden. Im Fehlerfall dürfen keine angeforderten Ressourcen übermittelt oder Funktionen unkontrolliert ausgeführt werden.

Kontrolle aller beteiligten Ressourcen an einer Aktion

Es darf einem Benutzer nicht möglich sein, eine Aktion auf eine Ressource durchzuführen, für die er keine ausreichenden Rechte hat. Falls z. B. ein authentisierter Benutzer einen URL-Parameter für die Zuordnung zu einem Bankkonto ändert, darf dieser dadurch keinen Zugriff auf ein fremdes Bankkonto erlangen. Werden die Rechte zur Durchführung einer Aktion geprüft, sollten daher ebenso alle an der Aktion beteiligten Ressourcen bei der Prüfung mit eingeschlossen werden.

Dies betrifft ebenfalls die Umsetzung und Konfiguration der Suchfunktion einer Webanwendung. Es sollte darauf geachtet werden, dass zugriffsgeschützte Ressourcen einem unbefugten Benutzer nicht als Suchergebnis präsentiert werden. Vor der Ausgabe der Suchergebnisse sollte daher beispielsweise geprüft werden, ob der Benutzer über ausreichende Rechte verfügt, um diese zu betrachten.

Zugriffskontrolle bei URL-Aufrufen und Objekt-Referenzen

Webseiten und andere Ressourcen der Webanwendung werden gewöhnlich über die URL identifiziert und abgerufen. Dabei ruft ein Benutzer Webseiten oder Funktionen der Webanwendung in der Regel über die angezeigten Verlinkungen auf einer bereits dargestellten Webseite der Anwendung auf.

Wenn Ressourcen der Webanwendung geschützt werden sollen, ist es nicht ausreichend, den Link auf die Ressource aus den angezeigten Webseiten zu entfernen (z. B. ein Link auf die Administrationsseite), sondern es muss auch der direkte Aufruf der Ressource über die URL geschützt werden.

Die Seiten von Webanwendungen werden häufig dynamisch anhand von Referenzen auf Objekte (z. B. die ID eines Datenbankeintrags) erstellt. Werden diese Referenzen durch die Benutzer der Webanwendung übergeben (z. B. als Parameter in der URL), kann der Parameter und somit die Referenz von einem Benutzer beliebig geändert werden.

Da es sich hierbei nicht um direkte Referenzen (z. B. auf Dateien), sondern um indirekte Referenzen (Verweise auf Objekte) handelt, sollte eine Zugriffskontrolle anhand der Referenzwerte (z. B. IDs) erfolgen. Des Weiteren sollte nach Möglichkeit eine zusätzliche Zugriffskontrolle für die angeforderten Objekte in den Hintergrundsystemen durchgeführt werden. Dies kann beispielsweise durch das Durchreichen der Benutzerauthentisierung an die Hintergrundsysteme realisiert werden (siehe auch M 5.168 *Sichere Anbindung von Hintergrundsystemen an Webanwendungen und Web-Services*).

Restriktive Dateisystemberechtigungen bei der Upload-Funktion

Grundsätzlich sollte der Zugriff auf Dateien durch die Benutzer der Webanwendung durch restriktive Dateisystemberechtigungen beschränkt werden (siehe M 4.398 *Sichere Konfiguration von Webanwendungen*).

Stellt die Webanwendung eine Funktionalität zum Hochladen von Dateien bereit, sollte nach dem erfolgten Hochladevorgang ausschließlich der Besitzer die Dateisystemberechtigung für den Zugriff auf die erstellten Dateien haben. In einem weiteren Schritt kann der Zugriff auf die Dateien explizit für andere Benutzer freigegeben werden. Generell ist darauf zu achten, dass hochgeladenen Dateien die Rechte für das Ausführen entzogen werden, sodass ein Angreifer hierüber keinen Schadcode zur Ausführung bringen kann (siehe auch M 4.399 *Kontrolliertes Einbinden von Daten und Inhalten bei Webanwendungen*).

Schutz temporärer Dateien

Bei dynamisch erstellten Webseiten werden häufig temporäre Daten (z. B. Auswertungsgrafiken, Berichte) erzeugt. Handelt es sich dabei um schützenswerte Daten, so sollten diese nach Möglichkeit nicht im Dateisystem zwischengespeichert werden. Stattdessen sollten die Daten direkt an den Benutzer ausgeliefert werden. Falls eine Speicherung zu schützender Daten in temporären Dateien notwendig ist, sollten folgende Punkte berücksichtigt werden:

- Die Zugriffsberechtigungen im Dateisystem sind restriktiv zu setzen, sodass der Zugriff nur befugten Benutzern und Diensten möglich ist.
- Dateinamen sollten sich aus Zufallswerten zusammensetzen (z. B. einem Globally Unique Identifier (GUID)), sodass sie nicht leicht erraten werden können.
- Sobald die Daten nicht mehr benötigt werden, sollten sie zeitnah gelöscht werden.
- Es wird empfohlen, die Dateien in einem Verzeichnis zu speichern, welches nicht über den Webserver erreichbar ist (z. B. außerhalb des Wurzelverzeichnisses des Webservers).
- Der Zugriff auf die Dateien sollte ausschließlich über solche Schnittstellen der Webanwendung möglich sein, die ausreichende Sicherheitsmechanismen bei der Zugriffskontrolle und Protokollierung umsetzen.

Prüffragen:

- Findet eine Zugriffskontrolle bei der Webanwendung auf der Grundlage von Benutzerrollen und -rechte statt?
- Werden alle von der Webanwendung verwalteten Ressourcen von der Autorisierungskomponente berücksichtigt?
- Erfolgt die Autorisierung bei der Webanwendung serverseitig und zentral auf einem vertrauenswürdigen IT-System?
- Führen Fehler während der Zugriffskontrolle bei der Webanwendung zur Ablehnung des Zugriffs?
- Werden Zugriffsberechtigungen für direkte URL-Aufrufe und Objekt-Referenzen von der Autorisierungskomponente der Webanwendung berücksichtigt?
- Werden Dateisystemberechtigungen bei der Webanwendung (insbesondere beim Hochladen von Dateien) restriktiv vergeben?
- Ist ein sicherer Umgang mit temporären Dateien von der Webanwendung vorgesehen (z. B. restriktive Vergabe von Berechtigungen und zeitnahes Löschen nicht mehr benötigter Dateien)?

M 4.403 Verhinderung von Cross-Site Request Forgery (CSRF, XSRF, Session Riding)

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Verantwortliche der einzelnen Anwendungen
Verantwortlich für Umsetzung: Entwickler, Administrator

Bei einem CSRF-Angriff (Cross-Site Request Forgery) wird einem Benutzer ein Befehl für eine Webanwendung (z. B. in Form eines Links in einem Gästebuch) von einem Angreifer übermittelt. Folgt der Benutzer diesem Link, wird der Befehl an die Webanwendung gesendet und im Kontext dieses Benutzers ausgeführt. Ist der Benutzer an der Webanwendung angemeldet, so wird die Vertrauensstellung des Benutzers gegenüber der Webanwendung ausgenutzt und der Befehl mit den Rechten des Benutzers ausgeführt.

Um derartige Angriffe zu erschweren, sollte die Webanwendung Sicherheitsmechanismen unterstützen, die es ermöglichen, beabsichtigte Seitenaufrufe des Benutzers von unbeabsichtigt weitergeleiteten Befehlen Dritter zu unterscheiden. Mit den folgenden Sicherheitsmaßnahmen soll verhindert werden, dass kritische Aktionen durch CSRF-Angriffe unbeabsichtigt ausgeführt werden.

Verwendung eines zusätzlichen Tokens

Bei einem CSRF-Angriff muss ein gültiger HTTP-Request nachgestellt und an das Opfer übermittelt werden. Ein solcher HTTP-Request kann z. B. durch eine URL auf die Webanwendung abgebildet werden (z. B. http://webapp.tld/addUser?name=benutzer). Hierfür muss der Angreifer die benötigten Request-Parameter, wie z. B. GET- und POST-Variablen, für den Aufruf kennen. Diese Parameter sind im Allgemeinen leicht zu ermitteln.

Als Schutz gegen einen CSRF-Angriff kann ein geheimes Token eingeführt werden, das nur schwer vom Angreifer erraten werden kann. Bei jedem Seitenaufruf der Webanwendung wird dieses Token als Parameter in URLs oder als verstecktes Element (Hidden-Field) in Formularen mit übertragen (Double Submit Cookies). Die Webanwendung prüft bei jedem Client-Request, ob das übertragene Token mit dem zur Sitzung hinterlegten Wert übereinstimmt. Im Fehlerfall wird der angeforderte Aufruf abgelehnt. Ohne Kenntnis dieses Tokens kann ein Angreifer keinen gültigen HTTP-Request nachstellen.

Obwohl die SessionID ein vertrauliches Datum ist und daher als Token zum Schutz gegen CSRF-Angriffe eingesetzt werden könnte, sollte vorzugsweise ein separates Token erzeugt werden. Für das Token sollten dabei ähnliche Anforderungen gelten, die auch an die SessionID gestellt werden.

Wird ein CSRF-Schutz implementiert, so wird empfohlen die Funktion aus bereits verwendeten Frameworks einzusetzen, falls diese eine entsprechende Implementierungen anbieten.

Bei Webanwendungen mit hohem Schutzbedarf sollte in Betracht gezogen werden, das Token für jeden Request derart zu erzeugen, dass bei jedem Aufruf der Webanwendung ein neues Token an den Client gesendet wird, das dann im darauffolgenden Request verwendet werden muss.

Bevor kritische Aktionen ausgeführt werden (z. B. zustandsändernde Anfragen wie eine Änderung des Passworts), sollte der Benutzer erneut von der Webanwendung authentisiert werden. Hierdurch können diese Funktionen nicht unbemerkt ausgeführt werden, sondern es ist eine Interaktion mit dem Benutzer erforderlich. Webanwendungen mit hohem Schutzbedarf sollten ein Authentisierungsverfahren mit mehreren Authentisierungsfaktoren (z. B. TAN, Chipkarte) verwenden.

Alternativ kann der Benutzer beim Aufruf von kritischen Aktionen auf eine Seite umgeleitet werden, die eine Benutzerinteraktion erfordert, bevor die Aktion ausgeführt wird (z. B. die Eingabe einer zufälligen Zeichenkette). Erst nachdem der Benutzer die Interaktion ausgeführt hat (z. B. richtige Zeichenkette eingegeben), wird er weitergeleitet und die ursprüngliche Anfrage bearbeitet. Anstelle der Zeichenkette können auch andere Mechanismen eingesetzt werden, die eine Benutzerinteraktion verlangen (z. B. CAPTCHAs oder Rätselfragen, siehe auch M 4.405 *Verhinderung der Blockade von Ressourcen (DoS) bei Webanwendungen und Web-Services*).

Das Referrer-Feld im HTTP-Request (die URL der Webseite, von der der Benutzer zur aktuellen Seite gekommen ist) kann als ein weiteres Sicherheitsmerkmal genutzt werden. Ein Request eines Benutzers der Webanwendung ist häufig nur dann gültig, wenn das Referrer-Feld eine URL der eigenen Webanwendung enthält. So kann davon ausgegangen werden, dass der Request durch den Klick auf einen Link der Webanwendung erzeugt wurde.

Dabei ist zu berücksichtigen, dass das Referrer-Feld deaktiviert oder gefiltert werden kann (z. B. aus Gründen des Datenschutzes) und die Maßnahme daher nicht für alle Webanwendungen umgesetzt werden kann.

Umgehung von Schutzmechanismen

Sicherheitsmechanismen zum Schutz vor CSRF-Angriffen, die auf das Referrer-Feld oder zusätzliche Tokens (siehe Abschnitt *Verwendung eines zusätzlichen Tokens*) basieren, können durch Cross-Site-Scripting-Angriffe umgangen werden. Die korrekte Filterung von Benutzerdaten (siehe M 4.393 *Umfassende Ein- und Ausgabevalidierung bei Webanwendungen und Web-Services*) ist daher entscheidend für die Wirksamkeit der Sicherheitsmaßnahmen zum Schutz vor CSRF-Angriffen.

Mindestens eine der ersten beiden Prüffragen sollte zum Schutz vor CSRF-Angriffen umgesetzt sein:

Prüffragen:

- Wird neben der SessionID ein geheimes Token für den Zugriff auf geschützte Ressourcen und Funktionen der Webanwendung benötigt?
- Wird bei Webanwendungen das Referrer-Feld im HTTP-Request als zusätzliches Merkmal zur Erkennung eines beabsichtigten Aufrufs durch einen Benutzer geprüft?
- Werden bei Webanwendungen kritische Aktionen nur nach einer erneuten Authentisierung oder einer manuellen Interaktion ausgeführt?

M 4.404 Sicherer Entwurf der Logik von Webanwendungen

Verantwortlich für Initiierung: Fachabteilung, IT-Sicherheitsbeauftragter, Verantwortliche der einzelnen Anwendungen
Verantwortlich für Umsetzung: Entwickler, Tester

Webanwendungen bilden komplexe Geschäftsprozesse ab, die über das bloße Anzeigen von einzelnen Webseiten hinausgehen. Beim technischen Entwurf dieser Prozesse muss darauf geachtet werden, dass die umgesetzte Anwendungslogik nicht missbräuchlich verwendet werden kann. So soll beispielsweise nicht aus einem vorgesehenen Prozess der Webanwendung ausgebrochen und somit der Ablauf des Prozesses von außen gesteuert werden können.

Die Anforderungen an die abgebildete Geschäftslogik müssen exakt erfasst und korrekt umgesetzt sein, sodass ausschließlich beabsichtigte und vorgesehene Aktionen durchgeführt werden können. Ein abweichendes Verhalten muss zurückgewiesen werden. Ist beispielsweise eine Empfehlungsfunktion der Webanwendung ausschließlich zum Versand von Artikelempfehlungen gedacht, sollte berücksichtigt werden, dass diese Funktion auch missbraucht werden kann, um SPAM-E-Mails zu versenden. Wird in diesem Beispiel der Empfehlungstext fest vorgegeben, so ist der Versand von SPAM über diese Funktion nicht möglich. Des weiteren ist auch zu prüfen, ob Fehler in der Geschäftslogik durch zwei gleichzeitige Sessions (concurrent sessions) auftreten können (race conditions).

Bei der Konzeption der Webanwendung sollten daher alle Funktionen anhand von Anwendungsfällen (Use Cases) dokumentiert werden. Dabei sollte erfasst werden, für welche Zwecke die Funktionen verwendet werden sollen und wie eine missbräuchliche Nutzung vermieden werden kann.

Wird die Webanwendung aus irgendeinem Grund abgebrochen, so muss die Logik sicherstellen, dass die Webanwendung wieder in einen konsistenten Zustand kommt (roll back).

Interaktive Funktionen in Web-Angeboten können auch durch aktive Inhalte umgesetzt werden, die auf dem Client-System ausgeführt werden (z. B. per JavaScript). Oft ist es auch möglich, diese Funktionalitäten mit dynamischen oder statischen Inhalten bereitzustellen. Da die Nutzung von aktiven Inhalten aus Sicherheitsgründen auf den Client-Systemen häufig deaktiviert ist, wird empfohlen, bei der Konzeption der Webanwendung auf die Verwendung aktiver Inhalte zu verzichten und die Anwendungslogik rein serverseitig zu realisieren.

Interaktive Funktionen in Webanwendungen können auf unterschiedliche Weise realisiert werden: Server-seitig oder Client-seitig. Da der Client nicht unter der Kontrolle der Webanwendung steht, kann nicht ausgeschlossen werden, dass dieser missbräuchlich benutzt wird. Gerade aktive Inhalte wie JavaScript oder ActiveX wurden und werden immer wieder ausgenutzt, Webanwendungen und die von ihnen verwalteten Informationen anzugreifen. Aus Sicherheitsüberlegungen empfiehlt es sich deshalb, aktive Inhalte Server-seitig umzusetzen oder, wo dies möglich ist, auf sie zu verzichten.

Ist die Nutzung aktiver Inhalte erforderlich, sind folgende Punkte zu beachten:

- Es sollte sichergestellt sein, dass die Webanwendung auch dann verwendet werden kann, wenn die Ausführung aktiver Inhalte im Browser deaktiviert ist.
- Für aktive Inhalte sollte nach Möglichkeit nachvollziehbar sein, aus welcher Quelle sie stammen, damit eine wirksame Prüfung im Client vorgenommen werden kann. Dies kann beispielsweise durch die Signatur von ActiveX-Komponenten erreicht werden.
- Ist die Serialisierung und die dynamische Erzeugung von XML-Daten bei der Verwendung von Ajax nicht zu vermeiden, sollte nach Möglichkeit auf Frameworks zurückgegriffen werden.
- Bei der Verwendung von JavaScript sollte auf den Funktionsaufruf eval() verzichtet werden.
- Benutzt die Webanwendung JSON für den Datenaustausch, so sind ausschließlich Objekte als Top-Level-Elemente zu verwenden.

Beispiele:

- Eine Webanwendung authentisiert die Benutzer in mehreren, aufeinanderfolgenden Schritten. Die Benutzer müssen im ersten Schritt Benutzername und Passwort und im zweiten Schritt ein Authentisierungstoken eingeben. Dabei sollte der erste Schritt nicht übersprungen werden können, damit sichergestellt ist, dass alle Authentisierungsmerkmale eingegeben werden. Im letzten Schritt für die endgültige Authentisierung muss dann eine erneute Prüfung aller Authentisierungsmerkmale durchgeführt werden.
- Bereits in der Konzeptionsphase einer Banking-Anwendung muss bedacht werden, dass ein Benutzer auch negative Beträge in ein Überweisungsformular eintragen kann. Die Webanwendung muss dabei sicherstellen, dass hierdurch nicht die Logik des Überweisungsformulars umgekehrt und keine unvorgesehene Gutschrift ausgelöst wird.

Prüffragen:

- Sind für die Funktionen der Webanwendung Anwendungsfälle (Use Cases) dokumentiert?
- Berücksichtigen die dokumentierten Anwendungsfälle der Webanwendung die missbräuchliche Nutzung der Funktionen?
- Wurde geprüft, ob auf die Verwendung von aktiven Inhalten bei Webanwendungen verzichtet werden kann?
- Wurde der Einsatz von aktiven Inhalten auf das Notwendige beschränkt?

M 4.405 Verhinderung der Blockade von Ressourcen (DoS) bei Webanwendungen und Web-Services

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator, Entwickler

Webanwendungen und Web-Services bieten häufig ressourcenintensive Funktionen an, die zum Beispiel komplexe Datenbankabfragen oder Datenübermittlungen auslösen. Werden diese rechenintensiven Operationen bewusst gehäuft aufgerufen oder die Webanwendungen und Web-Services mit Anfragen überflutet, können hierdurch Ressourcen im Übermaß belegt und der Betrieb bis zur Unerreichbarkeit eingeschränkt werden. Dieses Vorgehen wird als Denial-of-Service-Angriff (DoS) bezeichnet.

DoS-Angriffe beruhen in den meisten Fällen ebenso wie Brute-Force- und Enumeration-Angriffe auf Automation (siehe M 4.396 *Schutz vor unerlaubter automatisierter Nutzung von Webanwendungen*). Daher sollten zur Vorbeugung gegen DoS-Angriffe ähnliche Schutzmechanismen umgesetzt werden. Dazu zählen beispielsweise folgende Maßnahmen:

- Grenzwerte festlegen (zum Beispiel die vorübergehende Blockierung einer Ressource oder des Benutzerkontos nach wiederholten Fehlzugriffen),
- die Zeitspanne zwischen Anfrage und Verarbeitung durch die Webanwendung künstlich verzögern (zum Beispiel bei wiederholt erfolgloser Anmeldung),
- die aufrufende IP-Adresse bei Verdacht auf einen Angriff temporär sperren,
- CAPTCHAs verwenden,
- Eingaben bei Eingabefeldern verifizieren, bevor rechenintensive Operationen angestoßen werden,
- XML-Filtermechanismen und XML-Validitätsprüfungen einsetzen.

Zusätzlich geben die folgenden Beispiele Hinweise auf spezifische Schutzmaßnahmen, um Denial-of-Service-Angriffe bei Webanwendungen und Web-Services zu erschweren:

- Ressourcenintensive Operationen sind besonders anfällig für DoS-Angriffe. Daher kann die Ressourcennutzung pro Benutzer auf ein Maximum eingeschränkt werden. Darüber hinaus können bestimmte Operationen nur angemeldeten Benutzern zugänglich gemacht werden (zum Beispiel komplexe Datenbankaufrufe).
- Pro Benutzer sollte nur eine Anfrage zur gleichen Zeit bearbeitet werden. Mehrere Anfragen desselben Benutzers sollten sequenziell bearbeitet werden.
- Die Last durch DoS-Anfragen kann teilweise durch Zwischenspeichern (cachen) der Webseitenaufrufe deutlich verringert werden. Somit wird die angeforderte, rechenintensive Operation nicht bei jedem Aufruf ausgeführt, sondern lediglich das zwischengespeicherte Resultat zurückgegeben. Stark Ressourcen belastende Anfragen können auch in lastarmen Zeiten vorberechnet werden (Voraggregation).
- Die Architektur und Flusskontrolle der Webanwendung sollten darauf ausgelegt sein, rechenintensive Operationen zu vermeiden (zum Beispiel bei der Erstellung der Session-ID oder anderen kryptographischen Mechanismen sollten ressourcenintensive Operationen gemieden werden). Zur

Erkennung rechenintensiver Operationen können Lasttests durchgeführt werden.
- Ein Überlauf von Speicherplatz, zum Beispiel im Rahmen der Protokollierung, kann dazu führen, dass keine Schreibzugriffe mehr auf den Datenträger möglich sind. Werden Speichervorgänge von der Webanwendung oder dem Web-Service durchgeführt, kann dies den Betrieb gefährden. Daher sollte der Zugriff auf Datenspeicher begrenzt und die Kapazität der Datenspeicher regelmäßig geprüft werden. Ebenso sollte auch der Verbrauch von Arbeitsspeicher (RAM) pro Thread begrenzt werden.
- SOAP-Nachrichten müssen gemäß dem entsprechenden XML-Schema validiert werden. Ist die Validierung nicht erfolgreich, da sie zum Beispiel eine undefinierte Zahl an Elementen enthält, darf die SOAP-Nachricht nicht weiter verarbeitet werden, da diese ansonsten zu Problemen bei der Verarbeitung durch den XML-Parser führen kann.

Bei Web-Anwendungen und Web-Services, bei denen aufgrund ihrer Natur mit gezielten, zum Beispiel politisch motivierten DoS-Angriffen aus dem Internet zu rechnen ist, kann auch die Zusammenarbeit mit einem Dienstleister sinnvoll sein, der sich auf die Abwehr von DoS-Angriffen spezialisiert hat. Solche Dienstleister leiten den IP-Verkehr im Angriffsfall über eigene Systeme, die Zugriffe filtern und/oder die Zielsysteme durch andere Maßnahmen wie zum Beispiel Zwischenspeicher (Caching) entlasten. Dabei ist im Vorfeld abzuwägen, ob durch die Umleitung der Datenströme über die Systeme Dritter zusätzliche Gefährdungen oder Anforderungen entstehen. Eine beliebte Angriffsmethode für zwischengespeicherte Web-Seiten ist beispielsweise, dass der Angreifer nicht vorhandene Unterseiten aufruft. Wenn dies der Dienstleister nicht abfängt und die Anfrage nach der vermeintlich neuen Unterseite an die ursprüngliche Seite weiterleitet, kommt es zu einem unbeabsichtigten DoS-Angriff des Dienstleisters. Solchen neuen Angriffsvektoren muss bei der Auswahl des Anti-DoS-Dienstleisters begegnet werden.

Prüffragen:

- Werden der Einsatz und die Nutzung ressourcenintensiver Operationen bei Webanwendungen und Web-Services gemieden, und werden diese besonders geschützt?
- Wird ein möglicher Überlauf von Protokolldaten bei Webanwendungen und Web-Services überwacht und verhindert?
- Werden SOAP-Nachrichten anhand eines entsprechenden XML-Schemas validiert?
- Wurde bei kritischen Diensten und Anwendungen die Zusammenarbeit mit einem Anti-DoS-Dienstleister geprüft?

M 4.406 Verhinderung von Clickjacking

Verantwortlich für Initiierung: Leiter Entwicklung, Verantwortliche der einzelnen Anwendungen
Verantwortlich für Umsetzung: Entwickler, Administrator

Wird die Webanwendung Ziel eines Clickjacking-Angriffs, so werden Inhalte der Webanwendung in einem nicht sichtbaren Frame eingebunden. Besucht ein Benutzer eine Webseite, in der dieser Frame eingebunden ist, so werden Klicks auf sichtbare Inhalte unwissentlich vom unsichtbaren Frame abgefangen. Ist der Benutzer an der Webanwendung angemeldet, so können auf diese Weise zugriffsgeschützte Aktionen in der Webanwendung unbefugt ausgeführt werden. Um dies zu vermeiden, muss die Webanwendung sicherstellen, dass die Inhalte der eigenen Webanwendung nicht in Frames verwendet werden.

Daher sollten folgende Gegenmaßnahmen zur Verhinderung von Clickjacking umgesetzt werden:

- Eingebetteter Code (z. B. JavaScript) in den Webseiten sollte auf dem Client prüfen und sicherstellen, dass die Inhalte der Webanwendung auf der obersten Ebene des Browser-Fensters eingeblendet werden. Dies soll verhindern, dass keine anderen Ebenen über dem ursprünglichen Inhalt der Webseite gelagert werden können. Ist dies nicht möglich, so sollte die Anzeige der Webanwendung unterbunden werden (siehe *Skript zur Vermeidung von Clickjacking* in *Hilfsmittel zum Baustein Webanwendung*).
- Bei der Auslieferung der Webseiten durch die Webanwendung sollte zusätzlich in den HTTP-Response-Headern die Direktive X-FRAME-OPTIONS gesetzt sein. X-FRAME-OPTIONS DENY verhindert, dass Inhalte der Webseite in einem Frame angezeigt werden. Alternativ kann diese Einschränkung auf Seiten begrenzt werden, die nicht von derselben Domäne stammen (X-FRAME-OPTIONS SAMEORIGIN).

Prüffragen:

- Wird auf allen Webseiten der Webanwendung sichergestellt, dass die Inhalte ausschließlich auf der obersten Ebene des Browser-Fensters angezeigt werden?
- Ist in den HTTP-Response-Headern der Webanwendung die Direktive X-FRAME-OPTIONS gesetzt?

M 5.150 Durchführung von Penetrationstests

Verantwortlich für Initiierung: Behörden-/Unternehmensleitung, IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator, Leiter IT

Penetrationstests sind ein erprobtes und geeignetes Vorgehen, um die aktuelle Sicherheit eines IT-Netzes, eines einzelnen IT-Systems, einer (Web-) Anwendung oder eines Web-Service festzustellen. Sie dienen dazu, die Erfolgsaussichten eines vorsätzlichen Angriffs auf einen Informationsverbund oder ein einzelnes IT-System vorab einzuschätzen und daraus notwendige ergänzende Sicherheitsmaßnahmen abzuleiten beziehungsweise die Wirksamkeit von bereits umgesetzten Sicherheitsmaßnahmen zu überprüfen. Für sicherheitskritische Netze und Systeme sollten regelmäßig Penetrationstests erfolgen.

Im Detail werden dabei die installierten Anwendungen (Webanwendung, Mailserver, Web-Service etc.) beziehungsweise die zugrunde liegenden Trägersysteme (Betriebssystem, Datenbank etc.) überprüft.

Typische Ansatzpunkte für einen Penetrationstest sind:

- Netzkoppelelemente (Router, Switches, Gateways),
- Sicherheitsgateway (Paketfilter, Intrusion Detection System, Virenscanner, etc.),
- Server (Datenbankserver, Webserver, Fileserver, Speichersysteme etc.),
- Telekommunikationsanlagen,
- Webanwendungen (zum Beispiel Internetauftritt, Vorgangsbearbeitung, Webshop),
- Web-Services (zum Beispiel REST-Interface, SOAP-API, SOA),
- Clients,
- Drahtlose Netze (WLAN, Bluetooth) und
- Infrastruktureinrichtungen (Zutrittskontrollmechanismen).

Üblicherweise werden Penetrationstests in Blackbox-Tests und Whitebox-Tests unterteilt. Bei einem Blackbox-Test stehen dabei den Penetrationstestern lediglich die Adressinformationen des Zieles zur Verfügung, weitere Informationen werden ihnen nicht mitgeteilt. Mittels der Vorgehensweise "Blackbox-Test" soll damit der Angriff eines typischen Außentäters mit unvollständigen Kenntnissen über das Zielsystem simuliert werden. Dagegen verfügen die Penetrationstester bei einem Whitebox-Test über umfangreiche, für sie notwendige Informationen über die zu testenden Systeme. Dazu gehören beispielsweise Informationen über IP-Adressen, das interne Netz, die eingesetzte Soft- und Hardware etc. Diese Angaben werden ihnen zuvor vom Auftraggeber mitgeteilt.

Es ist jedoch fraglich, ob die Unterscheidung zwischen den Vorgehensweisen "Blackbox-Test" und "Whitebox-Test" heute noch sinnvoll ist. Beispielsweise besteht bei einem Blackbox-Test aufgrund nicht vorliegender Informationen ein höheres, durchaus vermeidbares Risiko, einen unbeabsichtigten Schaden zu verursachen. Weiterhin könnten beispielsweise Schwachstellen aufgrund nicht mitgeteilter Informationen übersehen werden.

Zudem besteht die Gefahr, dass im Rahmen eines Blackbox-Tests der Angriff eines informierten Innentäters nicht berücksichtigt wird.

Den Penetrationstestern sollten daher heutzutage alle für die Testdurchführung notwendigen Informationen über die zu testenden Systeme zur Verfügung gestellt werden, um eventuell mit dem Test verbundene Risiken minimieren zu können und eine möglichst vollständige Schwachstellensuche zu ermöglichen.

Die Klassifizierung von Penetrationstests in eine weitestgehend automatisierte Schwachstellensuche ("Vulnerability Scan") sowie eine in großen Teilen manuelle Sicherheitsrevision erscheint daher nach heutigem Kenntnisstand praxisnäher und erfolgsorientierter.

Personelle und fachliche Anforderungen an einen Dienstleister für Penetrationstests

Penetrationstests sind anspruchsvolle und diffizile Aufgaben, die auch Auswirkungen auf den IT-Betrieb haben können. Daher sollte hierfür nur hinreichend qualifiziertes und zuverlässiges Personal mit themenübergreifenden Kenntnissen auf folgenden Gebieten eingesetzt werden:

- Administration von Betriebssystemen und Anwendungen
- Netzwerkprotokolle und Auswertung von Netzwerkverkehr
- Sicherheitsprodukte (zum Beispiel Sicherheitsgateways, Intrusion Detection Systeme etc.)
- Programmiersprachen
- Schwachstellenscanner
- Audit- und Administrationssoftware

Werden externe Dienstleister mit der Durchführung von Penetrationstests beauftragt, so sollte darauf geachtet werden, dass ein qualifizierter und vertrauenswürdiger Dienstleister ausgewählt wird (siehe auch M 2.252 *Wahl eines geeigneten Outsourcing-Dienstleisters*), der entsprechend qualifizierte und zuverlässige Mitarbeiter bereitstellen kann.

Weiterhin sollten Anbieter von Penetrationstests dem Auftraggeber eine strukturierte Methodik zur Durchführung des Penetrationstests vorstellen können, auf deren Basis die jeweilige individuelle Vorgehensweise ausgearbeitet werden kann.

Strukturierung und Vorgehensweise für einen Penetrationstest

In einer Vorbereitungsphase müssen zunächst zwischen dem Auftraggeber und dem Auftragnehmer die Ziele sowie der Umfang des Penetrationstests so genau wie möglich festgelegt werden. Der Penetrationstester sollte hierbei dem Auftraggeber eine strukturierte Vorgehensweise, welche zwischen den Parteien abzustimmen ist, vorstellen.

Während des Abstimmungsprozesses sollte beachtet werden, dass unter Umständen Dritte über den geplanten Penetrationstest informiert beziehungsweise daran beteiligt werden müssen.

In der Regel müssen beispielsweise die Personalvertretung und der Datenschutzbeauftragte, häufig auch Externe, wie der Internet Service Provider oder der Webhoster, in das Vorhaben einbezogen werden.

Zwischen dem Auftraggeber und dem Dienstleister sollten bestimmte Voraussetzungen bereits im Vorfeld vereinbart werden. Hierzu zählen insbesondere:

- Vereinbarungen über die Verschwiegenheitspflichten
- Vereinbarungen über den Einsatz von Hard- und Software
- Vereinbarungen über die zu testenden IT-Systeme und IT-Anwendungen

- Festlegung von erlaubten und unerlaubten Aktivitäten der Penetrationstester, um Schäden möglichst zu vermeiden
- Vereinbarungen über den Umgang mit Datenträgern vor, während und nach Abschluss des Penetrationstests, da die Datenträger zum Beispiel sensible Informationen über die Testergebnisse enthalten können
- Festlegungen über den Ort der Durchführung sowie zur Auswertung und Berichterstellung für den Penetrationstest
- Festlegung eines Terminplans einschließlich Wartungsfenster für die Durchführung der Tests
- Detaillierte Vereinbarungen über den Zugang zum Internet beziehungsweise den Anschluss von Testsystemen an das Internet während der Durchführung und der Auswertung von Penetrationstests
- Vereinbarungen über Zuständigkeiten und die Erreichbarkeit von Ansprechpartnern sowie zur Notfallvorsorge

In der sich anschließenden Informationsphase sammeln die Penetrationstester möglichst viele Informationen über das zu testende Objekt. Zur Vorbereitung der Tests werden die gewonnenen Informationen anschließend hinsichtlich potenzieller Schwachstellen ausgewertet.

In der eigentlichen Testphase eines Penetrationstests sollten nach Möglichkeit die Testverfahren vermieden werden, welche ein destruktives Ergebnis für die untersuchten IT-Systeme oder IT-Anwendungen zur Folge haben könnten.

So zielen beispielsweise DoS-Angriffe (Denial of Service) darauf ab, den Zugriff auf einzelne Dienste, Systeme oder Netzsegmente zu unterbinden. Die Feststellung, ob derartige Attacken möglich sind, kann jedoch oftmals im Vorfeld durch eine Systemanalyse geklärt werden, sodass solche Angriffe während eines Penetrationstests überflüssig werden.

Sollen dennoch DoS-Angriffe oder ähnliche destruktive Angriffe im Rahmen eines Penetrationstests durchgeführt werden, sollte dies außerhalb der produktiven Nutzungszeiten des Systems erfolgen. Gegebenenfalls kann ein derartiger Angriff auch anhand eines Testsystems simuliert werden. Diese Vorgehensweisen sollten ausdrücklich vereinbart werden.

Erst danach werden aktive Eindringungsversuche unternommen. Dabei müssen die vereinbarten Wartungsfenster und der Terminplan strikt eingehalten werden. Wenn Änderungen am zeitlichen Ablauf erforderlich sind, muss dies auf jeden Fall mit dem Auftraggeber abgestimmt werden.

Anderenfalls besteht die erhöhte Gefahr, dass auf der Seite des Auftraggebers bestimmte Aktivitäten der Penetrationstester mit echten Angriffen verwechselt werden. Empfehlenswert ist die vollständige Aufzeichnung und Dokumentation des Penetrationstests.

Um möglichst aussagekräftige Ergebnisse zu erhalten, sollte darauf geachtet werden, dass die Penetrationstests unmittelbar an dem zu testenden IT-System sowie unter Umgehung der vorgeschalteten aktiven Netzkoppelelemente wie zum Beispiel Paketfilter durchgeführt werden. Liegen besondere Gründe vor, den Test mit aktiven vorgeschalteten Sicherheitskomponenten durchzuführen, so ist zu beachten, dass dabei eventuell Sicherheitsprobleme in der Anwendung selbst unentdeckt bleiben, weil die vorgelagerten Komponenten die Angriffsversuche im Penetrationstest abfangen. Solche unentdeckten Schwachstellen bilden jedoch ein relevantes Risiko, denn häufig können mit einem abgewandelten Angriff die Schutzsysteme ausgehebelt und die Schwachstellen ausgenutzt werden.

Typische Angriffstechniken

Netzwerk- und Portscanning: Netzwerk- und Portscanning werden genutzt, um die in einem Netz aktiven IT-Systeme aufzufinden und die dort angebotenen Dienste (Ports) zu identifizieren.

Seitens der IT-Administration werden solche Abfragen dazu genutzt, den aktuellen Status der eingesetzten IT-Systeme abzufragen. Allerdings kann ein Angreifer unter Umständen mit Hilfe dieser Informationen vorhandene Schwachstellen auf den einzelnen IT-Systemen identifizieren und basierend auf diesen Informationen einen Angriff durchführen.

Ausnutzung mangelhafter Eingabeüberprüfung: Als Eingabeüberprüfung wird das Verfahren bezeichnet, mit dem die Benutzereingaben (Daten), die einer Anwendung zur weiteren Bearbeitung übergeben werden, vorher gefiltert, bereinigt oder zurückgewiesen werden.

Diese Filterung soll verhindern, dass der Anwendung schädlicher Code übergeben werden kann, dessen Verarbeitung zu einem Fehlverhalten führt wie zum Beispiel der Offenlegung vertraulicher Informationen.

Angriffsmethoden, mit denen ein derartiges Fehlverhalten hervorgerufen werden kann, sind zum Beispiel "Cross-Site Scripting (XSS)", "Cross-Site Request Forgery (XSRF)", "SQL Injection", "LDAP Injection", "OS Injection", "Fuzzing" sowie im Bereich von Web-Services "XMLExternal Entity-Angriffe (XEE)" oder sogenannte "XML-Bomben".

Teilweise lassen sich auch Schwachstellen der verwendeten Protokolle und sonstigen Techniken ausnutzen, um Schaden zu bewirken, zum Beispiel mittels Angriffen auf veraltete SSL/TLS-Versionen oder etwa durch "XML Signature Wrapping (XSW)" bei Web-Services.

Denial-of-Service-Angriffe (DoS): Diese Angriffe zielen darauf ab, einen oder mehrere der zur Verfügung gestellten Dienste außer Betrieb zu setzen. Dies kann unter anderem mittels einer durch vermehrte Anfragen gesteigerten Last, durch ein massiv erhöhtes Datenaufkommen (zum Beispiel E-Mails), aber auch durch gezieltes Ausnutzen möglicher Softwarefehler durchgeführt werden. Ein bekanntes Beispiel für einen DoS-Angriff ist der "Ping of Death".

Information Gathering: Als "Information Gathering" wird die Sammlung aller Informationen bezeichnet, welche im weiteren für einen Angriff nützlich sein könnten. Beispiele für solche Informationen sind etwa das verwendete Nummerierungsschema für Verzeichnisse oder Server oder Erkenntnisse über Web-Service-Schnittstellen, die durch WSDL-Scanning gewonnen werden.

Social Engineering: Als "Social Engineering" werden beispielsweise fingierte Anrufe oder sonstige Kontaktaufnahmen mit Personen bezeichnet, die das betrachtete IT-System bedienen. Das Ziel ist meist, dadurch vertrauliche Informationen wie zum Beispiel Passwörter zu erhalten (siehe auch G 5.42 *Social Engineering*).

War Dialing: Hierunter wird der automatisierte und systematische Versuch verstanden, Telefonnummern, die in Verbindung mit einem Modem stehen, auszuforschen. Dabei werden die Telefonnummern des Zielsystems angerufen und auf ein antwortendes Modem hin abgeprüft.

Passwort-Attacken: Hierbei wird die Sicherheit beziehungsweise Stärke von Passwörtern mittels sogenannter Wörterbuchangriffe, Brute-Force-Attacken oder durch Entschlüsselungsversuche getestet.

Ausnutzen von Software-Schwachstellen: Bei diesen Angriffen wird beispielsweise getestet, ob die installierte Software anfällig für bestimmte Exploits ist, fehlerhaft konfiguriert ist, Schwachstellen aufweist oder veraltet ist. Häufig wird auch untersucht, ob etwa bekannte Schwachstellen der Standardinstallation des jeweiligen Produkts im vorliegenden Fall ausgenutzt werden können.

Kryptographische Angriffe: Hierbei werden beispielsweise die Stärke und die Implementierung der eingesetzten Verschlüsselungsmechanismen und -protokolle sowie der Schlüsselverwaltung untersucht.

Infrastruktur-Untersuchungen: Im Rahmen von Infrastruktur-Untersuchungen werden unter anderem bauliche Sicherungsmaßnahmen, Zutritts- und Schließeinrichtungen, aber auch die Entsorgung von Material durchleuchtet. Eine Variante hiervon ist das sogenannte "Dumpster Diving", also das Suchen nützlicher Unterlagen oder Datenträger im Abfall (zum Beispiel Papierkörbe, Abfallcontainer).

In der Auswertungs- und Berichtsphase werden die Ergebnisse gesammelt, ausgewertet und in Form eines Berichts zusammengestellt. Alle während des Penetrationstests gewonnenen Informationen sind hierbei entsprechend gesichert aufzubewahren. Der Auftraggeber sollte den Auftragnehmer im Vorfeld dazu verpflichten, alle Aufzeichnungen über den Penetrationstest vollumfänglich an den Auftraggeber zu übergeben beziehungsweise zu vernichten.

Der Bericht sollte neben einer Auflistung der gefundenen Schwachstellen auch Maßnahmenempfehlungen enthalten, wie mit den entdeckten Schwachstellen umgegangen werden sollte. Empfehlenswert ist hierbei zudem die Erstellung eines Umsetzungsplans für die in dem Bericht aufgeführten Maßnahmenempfehlungen einschließlich einer Priorisierung. Für das Management sollte der Abschlussbericht außerdem eine Zusammenfassung enthalten, in der die wesentlichen Prüfungsergebnisse und ein Überblick über die empfohlene weitere Vorgehensweise dargestellt sind. Der Abschlussbericht muss dem IT-Sicherheitsbeauftragten und den verantwortlichen Führungskräften vorgelegt werden.

Begleitend zu allen Phasen eines Penetrationstests ist eine gemeinsame Dokumentation der einzelnen Vereinbarungen und Ergebnisse durch den Auftraggeber und den Auftragnehmer empfehlenswert.

Prüffragen:

- Wird für Penetrationstests ausschließlich zuverlässiges und qualifiziertes Personal eingesetzt?
- Ist sichergestellt, dass ausschließlich vertrauenswürdige und qualifizierte Dienstleister mit Penetrationstests beauftragt werden?
- Ist sichergestellt, dass die Ergebnisse von Penetrationstests ausreichend geschützt und vertraulich behandelt werden?
- Werden die Abschlussberichte über Penetrationstests dem IT-Sicherheitsbeauftragten und den verantwortlichen Führungskräften vorgelegt?
- Wurden mit allen Auftragnehmern für Penetrationstests vorab detaillierte Vereinbarungen zur Durchführung und Auswertung von Penetrationstests abgeschlossen?

- Wurde im Vorfeld der Penetrationstests das Einverständnis aller zuständigen Stellen eingeholt?
- Wurden die Ansprechpartner und deren Erreichbarkeit für den Zeitraum der Durchführung von Penetrationstests verbindlich festgelegt?

M 5.168 Sichere Anbindung von Hintergrundsystemen an Webanwendungen und Web-Services

Verantwortlich für Initiierung: Leiter IT, Verantwortliche der einzelnen Anwendungen
Verantwortlich für Umsetzung: Administrator

Webanwendungen und Web-Services verwenden häufig Hintergrundsysteme, zum Beispiel für die Datenhaltung in einer Datenbank oder für die Authentisierung durch einen Identitätsspeicher. Die Daten sind auch bei der Übermittlung und Speicherung in Hintergrundsystemen ausreichend zu schützen. Dazu müssen die Hintergrundsysteme sicher an die Webanwendung oder den Web-Service angebunden sein.

Typische Hintergrundsysteme von Webanwendungen und Web-Services sind:

- Datenbanken,
- Verzeichnisdienste,
- Middleware,
- Web-Services und
- Legacy-Systeme.

Zur sicheren Anbindung von Hintergrundsystemen sollten folgende Punkte beachtet werden:

Platzierung von und Zugriff auf die Hintergrundsysteme

Die Benutzer der Webanwendung beziehungsweise Aufrufer des Web-Service sollten nicht direkt auf die Hintergrundsysteme zugreifen können, da so gegebenenfalls Schutzmaßnahmen umgangen werden. Stattdessen sollte der Zugriff ausschließlich über vordefinierte Schnittstellen und Funktionen der Webanwendung oder des Web-Service möglich sein.

Darüber hinaus sollte bei hohem Schutzbedarf die Verbindung zu den Hintergrundsystemen zusätzlich geschützt werden. Hierzu sollten sich die Systeme vor der Datenübertragung authentisieren und die übertragenen Daten verschlüsseln, sodass sie nicht unbemerkt mitgelesen oder geändert werden können (zum Beispiel mittels SSL/TLS; siehe auch M 5.66 *Clientseitige Verwendung von SSL/TLS* und M 5.177 *Serverseitige Verwendung von SSL/TLS*).

Werden die beteiligten IT-Systeme über unsichere Kanäle angebunden, so sollte in jedem Fall ein kryptographisch abgesicherter Tunnel mit entsprechender Verschlüsselung und Authentisierung verwendet werden.

Zugriffe auf Hintergrundsysteme sollten mit minimalen Rechten erfolgen. Hierfür sollten Dienstekonten auf dem jeweiligen Hintergrundsystem eingerichtet werden.

Wird für den Zugriff auf ein Hintergrundsystem ein einziges Dienstekonto verwendet, werden alle Anfragen im Sicherheitskontext dieses Dienstekontos bearbeitet. Dies gilt dann sowohl für Zugriffe von Benutzern mit eingeschränkten Zugriffsberechtigungen als auch für die Zugriffe administrativer Benutzer. Um dies zu verhindern, sollten mehrere Dienstekonten mit unterschiedlichen Zugriffsrechten für ein Hintergrundsystem verwendet werden.

Bei einer geeigneten Systemumgebung (zum Beispiel bei der Verwendung eines Verzeichnisdienstes, der sowohl von der Webanwendung als auch für das Hintergrundsystem zur Verwaltung der Benutzer verwendet wird) können die Benutzerkonten an das Hintergrundsystem weitergeleitet werden. Auf diese Weise können die Privilegien auf die notwendigen Rechte des jeweils an der Webanwendung angemeldeten Benutzers limitiert werden.

Es ist darauf zu achten, dass für unauthentisierte Zugriffe auf die Webanwendung ein eigenes Dienstekonto im Verzeichnisdienst mit eingeschränkten Berechtigungen verwendet wird.

Enterprise Service Bus

Im Kontext sogenannter Service-Orientierter Architekturen (SOA) werden Webanwendungen und Web-Services häufig über einen Enterprise Service Bus (ESB) als zentrale Kommunikationsinfrastruktur an Hintergrundsysteme angebunden. Dadurch wird erreicht, dass für jede Anwendung jeweils nur die Schnittstelle zum ESB definiert und realisiert werden muss, und nicht viele separate Schnittstellen zu anderen Anwendungen und Diensten. In einem eigenen Verzeichnis ("Repository") speichert der ESB Metainformationen über die angeschlossenen Dienste.

Zusätzlich kann der ESB auch zentrale Sicherheitsfunktionen realisieren, um die angeschlossenen Anwendungen weiter zu schützen. Solche Sicherheitsfunktionen können beispielsweise Replay-Attacken erkennen und abwehren oder XML-Daten auf potenziell schädliche Inhalte prüfen, aber auch den Nachrichtenaustausch zentral und revisionssicher protokollieren.

Beim Einsatz eines ESB muss sichergestellt werden, dass sich alle Dienste gegenüber dem ESB authentisieren, bevor ihnen ein Zugriff erlaubt wird. Dies gilt auch für Zugriffe auf das ESB-Repository. Der ESB muss so in die Netzarchitektur integriert werden, dass ein Zugriff nur von den Servern der angeschlossenen Anwendungen und Dienste möglich ist und ein Zugriff von außen auf den ESB ausgeschlossen wird. Dazu sollte der ESB ein eigenes logisches Netzsegment erhalten.

Der ESB muss eine eigene Berechtigungsprüfung durchführen, um zu prüfen, ob ein Zugriff auf den angefragten Dienst durch den anfragenden Dienst beziehungsweise die anfragende Anwendung zulässig ist. Dabei muss insbesondere sicher ausgeschlossen werden, dass Anwendungen oder Dienste mit Außenkontakt auf interne Dienste zugreifen, die dafür nicht vorgesehen sind. Solche Anwendungen dürfen auch nicht über das ESB-Repository Kenntnis von internen Diensten und ihren Schnittstellen erlangen.

Sofern die service-orientierte Architektur mehrere Sicherheitsdomänen umspannt, zum Beispiel eine DMZ mit extern aufrufbaren Services und ein internes Netz mit Backend-Systemen, so muss auch der ESB in entsprechende Sicherheitsdomänen mit kontrollierten Übergängen aufgeteilt werden, oder es müssen mehrere ESB für die einzelnen Sicherheitszonen realisiert werden.

Sofern der ESB nicht ausschließlich lokal in einem geschützten RZ-Netz kommuniziert, muss die Kommunikation zwischen ESB und den angeschlossenen Anwendungen geeignet gesichert werden (Authentisierung und Verschlüsselung).

Durch die Bündelung der Kommunikation vieler Anwendungen und Dienste kommt der Verfügbarkeit des ESB eine besondere Bedeutung zu. Dies ist bei

der Realisierung und beim Betrieb des ESB entsprechend durch Redundanzen und eine geeignete Überwachung des Dienstes zu berücksichtigen.

Prüffragen:

- Ist der Zugriff auf Hintergrundsysteme von Webanwendungen oder Web-Services ausschließlich über definierte Schnittstellen und von definierten Systemen aus möglich?
- Wird der Datenverkehr zwischen den Benutzern und der Webanwendung beziehungsweise den Anwendungen, Web-Services und weiteren Diensten sowie den Hintergrundsystemen durch Sicherheitsgateways (Firewalls) reglementiert?
- Werden die Verbindungen zwischen Webanwendungen oder Web-Services und Hintergrundsystemen bei hohem Schutzbedarf durch eine Transportverschlüsselung geschützt?
- Ist sichergestellt, dass Anfragen der Webanwendung oder des Web-Service an Hintergrundsysteme nur mit minimalen Rechten auf diesen ausgeführt werden?
- Ist beim Einsatz eines ESB ein eigenes logisches Netzsegment für den ESB vorgesehen? Ist der Zugriff auf den ESB ausschließlich durch die angeschlossenen Anwendungen und Dienste möglich?
- Ist eine Segmentierung nach Zonen entsprechend den vorhandenen Sicherheitsdomänen im ESB durchgehalten, nötigenfalls bis hin zur Auftrennung in mehrere ESB?
- Werden alle Zugriffe auf den ESB authentisiert und bei der Kommunikation über Standort- und Netzgrenzen hinweg ausreichend gesichert/verschlüsselt?
- Sind bei der Realisierung und beim Betrieb des ESB geeignete Maßnahmen zur Sicherstellung einer angemessenen Verfügbarkeit umgesetzt?

M 5.169 Systemarchitektur einer Webanwendung

Verantwortlich für Initiierung: Verantwortliche der einzelnen Anwendungen
Verantwortlich für Umsetzung: Administrator

Webanwendungen verwenden im Allgemeinen mehrere IT-Systemkomponenten wie z. B. Webserver, Web-Applikationsserver und Hintergrundsysteme. Als Grundlage für den sicheren Betrieb einer Webanwendung muss eine geeignete Systemarchitektur gewählt werden.

Beim Entwurf der Systemarchitektur der Webanwendung und der Vernetzung der beteiligten IT-Systeme sollten die folgenden Punkte berücksichtigt werden.

Trennung nach Server-Rollen

Die Serverdienste der Webanwendung (z. B. Webserver, Applikationsserver, Datenbankserver) sollten jeweils auf separaten IT-Systemen betrieben werden. Kann bei diesem Ansatz eine Schwachstelle im System einer exponierten Komponente (z. B. im Webserver) ausgenutzt werden, so sind die auf anderen Systemkomponenten (z. B. der Datenbank) gespeicherten Daten hiervon nicht direkt betroffen.

Eine Trennung der Server-Rollen kann auch durch eine Servervirtualisierung umgesetzt werden. Wird von der Servervirtualisierung Gebrauch gemacht, so ist bei der Umsetzung der Baustein B 3.304 *Virtualisierung* zu berücksichtigen.

Eingeschränkte Konten für Serverprozesse der Systemkomponenten

Es sollten jeweils eigene Konten für die unterschiedlichen Serverprozesse der Systemkomponenten verwendet werden (z. B. ein eigener Systembenutzer für den Webserverprozess). Dabei sind die Rechte dieser Dienstekonten auf Betriebssystemebene soweit einzuschränken, dass nur auf die erforderlichen Ressourcen und Dateien des Betriebssystems zugegriffen werden kann. Auf diese Weise verfügt ein Angreifer auch nach einer erfolgreichen Kompromittierung eines Server-Prozesses nur über eingeschränkte Rechte, sodass der Zugriff auf Betriebssystemebene erschwert wird.

Mehrschichtige Netzwerkarchitektur

Die IT-Systemkomponenten der Webanwendung sollten im Sicherheitsgateway in demilitarisierten Zonen (DMZ) entsprechend des Schutzbedarfs entkoppelt werden (siehe M 2.73 *Auswahl geeigneter Grundstrukturen für Sicherheitsgateways*).

Die Netzwerkarchitektur sollte einen mehrschichtigen (Multi-Tier) Ansatz verfolgen. Dabei sollten mindestens die folgenden Sicherheitszonen berücksichtigt werden:

- Webschicht
  Diese Schicht grenzt an das nicht vertrauenswürdige Netz (z. B. Internet) und stellt die exponierte Schicht mit direkten Zugriffen durch Benutzer dar. Paketfilter zwischen angrenzenden Netzen (z. B. Anwendungsschicht und Internet) sollten den Datenverkehr filtern, sodass kein direkter Zugriff aus dem nicht vertrauenswürdigen Netz über die Netzgrenzen der Webschicht hinaus möglich ist. In dieser Schicht sollten Systeme wie der Webserver

platziert werden, die eine exponierte Stellung einnehmen und z. B. den direkten Zugriff durch Benutzer erfordern.
- Anwendungsschicht
  Die Anwendungsschicht sollte zum einen an die Webschicht und zum anderen an die Datenschicht angrenzen. Der Netzverkehr zu den angrenzenden Netzen sollte durch Paketfilter gefiltert werden, sodass kein direkter Zugriff zwischen den angrenzenden Netzen möglich ist. In diesem Netzsegment sollten Systeme und Server mit der Anwendungslogik (z. B. der Applikationsserver mit der Webanwendung) platziert sein. Die Systeme greifen auf Daten aus der angrenzenden Datenschicht zu (z. B. Datenbanken), bereiten diese auf und stellen sie Systemen in der Webschicht (z. B. dem Webserver) zur Verfügung.
- Datenschicht
  Die Datenschicht ist die vertrauenswürdigste Zone der mehrschichtigen Architektur. Paketfilter zwischen den angrenzenden Netzen sollten den Datenverkehr reglementieren. In dieser Schicht sollten die Hintergrundsysteme der Webanwendung wie z. B. Datenbanken, Verzeichnisdienst und Legacy-Systeme aufgestellt sein. Der Zugriff auf diese Systeme sollte ausschließlich von angrenzenden Netzen aus möglich sein (z. B. Anwendungsschicht). Die Datenschicht ist als separate Zone umzusetzen und sollte nicht in andere Zonen integriert werden (z. B. Intranet).

Es sollte aus den oben genannten Zonen nicht auf Systeme im Intranet zugegriffen werden können. Falls z. B. für die Authentisierung an der Webanwendung ein Verzeichnisdienst eingesetzt wird, sollte hierfür nach Möglichkeit eine eigene Domäne auf dedizierter Hardware verwendet werden.

Eine Filterung des Datenverkehrs sollte durch getrennte Filterkomponenten erfolgen (z. B. Paketfilter). Bei hohem Schutzbedarf sollten die Filterkomponenten durch Systeme mit Filterfunktionen auf höheren Protokollebenen (z. B. Application Level Gateway) ersetzt oder ergänzt werden. Das Application Level Gateway sollte dabei in einer eigenen Sicherheitszone integriert werden, die noch vor den Systemen der Webschicht die Anfragen der Benutzer entgegennimmt.

Einsatz von Web Application Firewalls

Bei der Filterung auf höheren Protokollebenen kann auf den Einsatz von Web Application Firewalls (WAF) zurückgegriffen werden. Da eine WAF das HTTP-Protokoll und die darüber übertragenen Daten analysiert, können Angriffsmuster auf Anwendungsebene bereits an der WAF gefiltert werden. Auf diese Weise werden Angriffsversuche frühzeitig erkannt und nicht mehr an die Webanwendung weitergeleitet.

Die Filterung an der WAF kann üblicherweise auf zwei Arten erfolgen.

- An eine Webanwendung gesendete Daten werden auf bekannte Angriffsmuster untersucht. Die Angriffsmuster werden vom Hersteller der WAF zur Verfügung gestellt und umfassen sowohl typische Zeichenketten, die bei allgemeinen Angriffen gegen Webanwendungen (z. B. SQL-Injection) verwendet werden, als auch spezifische Angriffsmuster, die Standard-Softwareprodukte betreffen. Damit bekannte Angriffe zuverlässig erkannt werden, müssen die Angriffssignaturen ähnlich wie bei einem Virenscanner regelmäßig aktualisiert werden.
- Wird keine Standardsoftware eingesetzt oder soll ein zusätzlicher Schutz erreicht werden, können für WAF üblicherweise auch eigene Filterregeln erstellt werden. Dabei wird beispielsweise definiert, welche Eingabedaten für die Webanwendung zugelassen werden. Diese Methode erfordert

einen hohen Konfigurationsaufwand und eine genaue Kenntnis über die von der Webanwendung verarbeiteten Daten.

Prüffragen:

- Ist eine Trennung der Serverdienste bei Webanwendungen auf jeweils separate IT-Systeme vorgesehen (Trennung nach Server-Rollen)?
- Werden eingeschränkte Konten für Serverprozesse der Systemkomponenten von Webanwendungen verwendet?
- Wird für die Webanwendung ein mehrschichtiger (Multi-Tier) Ansatz bei der Netzwerkarchitektur umgesetzt?
- Bei dem Einsatz von Web Application Firewalls: Ist die Konfiguration der WAF auf die zu schützende Webanwendung angepasst worden?
- Bei dem Einsatz von Web Application Firewalls: Werden die Angriffssignaturen für die WAF regelmäßig aktualisiert?

M 5.177 Serverseitige Verwendung von SSL/TLS

Verantwortlich für Initiierung: IT-Sicherheitsbeauftragter, Leiter IT
Verantwortlich für Umsetzung: Administrator

Transport Layer Security (TLS) ist eine Weiterentwicklung von Secure Sockets Layer (SSL) und wird dazu verwendet, Informationen während der Übertragung in Netzen, in der Regel zwischen Serverdiensten und Clients oder zwischen Serverdiensten untereinander kryptographisch abzusichern. Konfigurationshinweise, wie SSL/TLS auf Clients verwendet werden sollte, und allgemeine Informationen zur Funktionsweise von SSL/TLS sind in M 5.66 *Clientseitige Verwendung von SSL/TLS* zu finden. Clients können die Verschlüsselung über SSL/TLS nur dann nutzen, wenn diese von den Serverdiensten unterstützt wird. SSL/TLS kann dazu eingesetzt werden, Informationen aus der Anwendungsschicht (z. B. HTTP, LDAP, POP3, IMAP und SMTP) verschlüsselt über TCP/IP zu übertragen. Überdies können mittels SSL/TLS auch sichere VPNs (Virtuelle Private Netze) aufgebaut werden. Mit OpenVPN, einer unter der GNU GPL (General Public License) frei verfügbaren Software, können VPNs mittels SSL/TLS verschlüsselte Verbindungen realisiert werden. Vertiefende Informationen zu VPNs sind in B 4.4 *VPN* zu finden.

In der Regel ist es bei vielen Serverdiensten nur ein geringer Mehraufwand, diese so zu konfigurieren, dass eine SSL/TLS-Verschlüsselung unterstützt wird, oder so, dass diese für einen Informationsaustausch ausschließlich genutzt wird. Daher ist für alle Serverdienste zu prüfen, ob mit vertretbarem Aufwand eine Verschlüsselung über SSL/TLS möglich und praktikabel ist. Ist dies mit vertretbarem Aufwand möglich, sollte die SSL/TLS-Verschlüsselung aktiviert werden. Generell sollte der interne und externe Nachrichtenstrom von und zu LDAP-, E-Mail- und Webservern mit SSL/TLS verschlüsselt werden.

Auswahl einer vertrauenswürdigen Zertifizierungsstelle

Zu Beginn eines neuen mit SSL/TLS abgesicherten Kommunikationsaufbaus findet ein sogenannter Handshake zwischen Client und Server statt. Hierbei verständigen sich Client und Server über die kryptographischen Algorithmen, die für Schlüsselaustausch, Verschlüsselung und Integritätssicherung eingesetzt werden. Außerdem einigen sich Client und Server über die SSL-Version, die verwendet wird. Zusätzlich dazu sendet der Server sein X.509-Zertifikat an den Client. Optional kann der Server auch so konfiguriert werden, dass auch der Client aufgefordert wird, dem Server sein X.509-Zertifikat zu übermitteln.

Die Identität der Kommunikationspartner wird hierbei über diese Zertifikate geprüft. X.509-Zertifikate enthalten die öffentlichen Schlüssel sowie eine Bestätigung einer weiteren Instanz, der Zertifizierungsstelle oder auch Trustcenter oder Certificate Authority (CA) genannt, über die korrekte Zuordnung des öffentlichen Schlüssels zu dessen "Besitzer". Der Wert eines Zertifikates hängt davon ab, welche Felder des X.509-Zertifikats von der Zertifizierungsstelle geprüft werden, bevor das Zertifikat ausgestellt wird, und wie vertrauenswürdig die Zertifizierungsstelle selbst ist. Daher spielt die Auswahl einer vertrauenswürdigen Zertifizierungsstelle eine wichtige Rolle.

Aufgrund der Vielzahl von Zertifizierungsstellen auf dem Markt sollte eine Institution die Zertifizierungsstelle sorgfältig auswählen. Es ist ratsam, die für

den späteren Betrieb wesentlichen Auswahlkriterien im Vorfeld festzulegen. Zu diesen können beispielsweise gehören:

- ob das Root-Zertifikat schon in CA-Listen der Clients, wie dem Browser, enthalten ist,
- wo sich Sitz und Rechtsstand der Zertifizierungsstelle befinden, und auch wo der Sitz des technischen Betriebs sich befindet,
- was die geschäftliche Ausrichtung der Zertifizierungsstelle ist (Ist CA-Betrieb ein zentrales Geschäftsfeld?), was die angebotenen CA-Dienste umfassen (z. B. OSCP, CRL),
- welches Sicherheitsniveau die Zertifizierungsstelle nachweisen kann,
- wie gut Umfang und Qualität des technischen Supports sind,
- wie hoch die Zertifikatskosten sind.

Grundsätzlich sollten die Kosten eines Zertifikats jedoch keinesfalls das allein ausschlaggebende Kriterium darstellen. Wird der angebotene Serverdienst von einem beschränkten Benutzerkreis verwendet, z. B. nur innerhalb eines LANs, kann ein Zertifikat auch ohne die Beteiligung einer Zertifizierungsstelle selbst erstellt und signiert und auf alle Clients eingespielt werden, auf denen der Serverdienst genutzt werden soll,

Extended Validation Zertifikate

Um Angriffe mit gefälschten Webseiten zu erschweren und der Problematik entgegen zu wirken, dass diverse Zertifizierungsstellen SSL/TLS-Anträge nicht immer zuverlässig prüfen, wurden Extended Validation Zertifikate zum Umgang mit Zertifikaten mit höheren Sicherheitsanforderungen eingeführt. Diese sollen verhindern, dass, wenn ein Zertifikat ausgestellt wird, eine CA nur den Domainnamen prüft. Darüber hinaus soll die CA außerdem noch eindeutig nachvollziehen, von wem die betreffende Domain registriert wurde. Im Unterschied zu den normalen X.509 SSL/TLS-Zertifikaten wird bei diesen erweiterten Zertifikaten (Extended Validation SSL-Zertifikate, EV-SSL) die Identität des Antragstellers ausführlicher überprüft. Hierbei verpflichten sich die beteiligten Zertifizierungsstellen und Browser-Hersteller, die "Guidelines for the Issuance and Management of Extended Validation Certificates" des CA/Browser Forums einzuhalten. Danach sind unter anderem folgende Kriterien vom Antragsteller zu erfüllen:

- Identitätsnachweis und Adresse des Antragstellers,
- Nachweis, dass der Antragsteller alleiniger Eigentümer der Domain ist,
- Bestätigung, dass antragstellende Person überhaupt berechtigt ist, den Antrag zu stellen und
- Hauptkontaktperson.

Zusätzlich darf der Antragsteller oder die antragstellende Person auf keiner Liste mit verbotenen Organisationen oder Personen stehen. Außerdem darf das Land, in dem sich der Sitz oder der Rechtsstand des Antragstellers befindet, weder Handelsembargos oder irgendwelchen anderen Sanktionen ausgesetzt sein, die durch dasjenige Land verhängt wurden, dessen Gesetzgebung die Zertifizierungsstelle unterliegt.

Für die Anwender sind EV-SSL-Zertifikate daran zu erkennen, dass in den unterstützten Browsern bestimmte Bereiche, wie die URL im Adressfeld oder das von vielen Browsern verwendete Vorhängeschlosssymbol, das eine verschlüsselte Seite kennzeichnet, grün hinterlegt ist. Je nach Konfiguration des Sicherheitsgateways (Firewall), hinter dem die Benutzer auf Webseiten mit EV-SSL-Zertifikaten zugreifen, kann es aber vorkommen, dass diese Markierungen in den Browsern der Clients nicht angezeigt werden. Wird beispielsweise der Nachrichtenfluss zwischen Client und Webserver von einem Proxy

ent- und wieder neu verschlüsselt, wird im Browser lediglich das SSL/TLS-Zertifikat des Sicherheitsgateways angezeigt.

Neben den höheren finanziellen Kosten, die für die Ausstellung eines EV-SSL-Zertifikats entstehen können, dauert die Antragstellung in der Regel auch länger, da zusätzliche Informationen von der Zertifizierungsstelle überprüft werden. Wenn es möglich ist, wird empfohlen, diesen zusätzlichen Aufwand in Kauf zu nehmen. Insbesondere in Bereichen, in denen Informationen mit höherem Schutzbedarf bezüglich Vertraulichkeit und Integrität übertragen werden, sollten EV-SSL-Zertifikate bevorzugt eingesetzt werden.

Common Name Eintrag

Browser zeigen immer eine Sicherheitswarnung an, wenn der im Zertifikat einer Webseite eingetragene Common Name (Allgemeiner Name) nicht mit dem vollständigen DNS-Name (Fully Qualified Domain Name) übereinstimmt, über den der Server im Web erreichbar ist. Daher sollte sichergestellt sein, dass der Common Name zu der URL passt, die tatsächlich verwendet wird, um mit dem Server zu kommunizieren. Wenn es möglich ist, sollten Wildcard-Zertifikate (z. B. *.example.de) vermieden werden. Diese werden häufig eingesetzt, um mit einem einzelnen Zertifikat mehrere Subdomains abzusichern.

Vollständige Zertifikatskette

Da für die Prüfung der hierarchischen Zertifikatskette durch den Browser auch alle Zwischen-Zertifikate benötigt werden, reicht das SSL-Zertifikat des Servers alleine nicht aus. Deshalb sollte der Server so konfiguriert werden, dass beim Verbindungsaufbau alle erforderlichen Zertifikate an den Client gesendet werden. Dazu sollte die Zertifikatskette im Webserver entsprechend hinterlegt werden.

Zu beachten ist außerdem, dass neben Zertifikate, die fehlen, auch abgelaufene oder gesperrte Zertifikate die Prüfung der Zertifikatskette fehlschlagen lassen. Nur wenn alle Zertifikate gültig sind und beim Verbindungsaufbau übertragen wurden, kann die Zertifikatskette erfolgreich geprüft werden.

Auswahl einer SSL/TLS Protokollversion

Derzeit existieren fünf SSL/TLS-Protokollversionen: SSL v2, SSL v3, TLS v1.0, TLS v1.1 und TLS v1.2. SSL v1 wurde nicht veröffentlicht. Um eine sichere Verbindung zwischen Client und Server zu gewährleisten, sollte TLS 1.2 verwendet werden. TLS 1.1 bietet ausreichende Sicherheit, aber im Vergleich zu TLS 1.2 weist es jedoch einige Schwächen auf, z. B. sind in TLS 1.1 noch Cipher-Suites vorhanden, die auf IDEA und DES basieren, in TLS 1.2 nicht mehr. TLS 1.0 kann in bestehenden Anwendungen übergangsweise weiter eingesetzt werden, falls eine sofortige Migration zu TLS 1.1 oder vorzugsweise TLS 1.2 nicht möglich ist und geeignete Maßnahmen gegen Chosen-Plaintext-Angriffe (z. B. BEAST) auf die CBC-Implementierung getroffen werden. Generell sollte jedoch eine Migration zu TLS 1.2 schnellstmöglich erfolgen.SSL v2 und SSL v3 dürfen nicht mehr eingesetzt werden.

Sichere Cipher-Suites

SSL/TLS nutzt Cipher-Suites, die bestimmen, wie sicher eine HTTPS-Verbindung ist. Jede Suite besteht aus spezifischen Modulen. Wenn ein bestimmtes Modul als unsicher oder schwach eingestuft wird, kann durch die Veränderung der Cipher Suite zu einem sichereren Modul gewechselt werden.

Da die Verwendung schwacher Cipher Suites clientseitig erzwungen werden kann, ist es erforderlich, serverseitig nur solche anzubieten, die Authentisierung und Verschlüsselung mit einer ausreichenden Stärke einsetzen. Darüber hinaus sollten die verwendeten Cipher-Suites Perfect Forward Secrecy (PFS) unterstützen (siehe TR-02102-2).

Weitere Hinweise zu kryptografischen Algorithmen und Schlüssellängen sind in der Technischen Richtlinie des BSI Kryptographische Verfahren: Empfehlungen und Schlüssellängen - Teil 2 Verwendung von TLS (TR-02102-2) und M 2.164 *Auswahl eines geeigneten kryptographischen Verfahrens* enthalten.

Session Renegotiation/TLS-Kompression

Mittels der sogenannten Session Renegotiation (Session-Neuverhandlung) können sowohl Client als auch Server die Parameter einer bestehenden HTTPS-Sitzung neu aushandeln. Aufgrund eines Fehlers in der Spezifikation des TLS-Protokolls (RFC 5246) ist es einem Man-in-the-Middle-Angreifer möglich, die Session Renegotiation zu missbrauchen, um beliebige Inhalte in eine existierende HTTPS-Sitzung einzufügen. Mittlerweile wurde das TLS-Protokoll erweitert (RFC 5746) und dieser Designfehler behoben. Generell sollte überlegt werden, ob serverseitig die Session Renegotiation erforderlich ist. Ist dies der Fall, dann sollte diese sicher konfiguriert werden, also auf Basis des RFC 5746. Eine Renegotiation, die durch den Client initiiert wird, sollte vom Server abgelehnt werden.

Darüber hinaus sollte die TLS-Kompression deaktiviert werden.

Webserverspezifische Aspekte

Generell wird empfohlen, die auf Webservern zur Verfügung gestellten Inhalte bei der Übertragung vom Server zum Client und umgekehrt mittels SSL/TLS zu schützen.

Wenn möglich, sollte darauf verzichtet werden, Webseiten mit gemischten Inhalten anzubieten. Als Webseite mit gemischtem Inhalt wird eine Seite bezeichnet, die zwar Verschlüsselung nutzt, dabei aber auch unverschlüsselte Inhalte (z. B. JavaScript-, CSS-Dateien oder Bilder) einbindet. Ein Man-in-the-Middle-Angreifer kann die Übertragung einer einzelnen unverschlüsselten Datei ausnutzen, um eine HTTPS-Session zu übernehmen. Da Webseiten mit gemischten Inhalten zudem üblicherweise Browser-Warnungen erzeugen, wird dadurch die Benutzerfreundlichkeit verschlechtert.

HTTP Strict Transport Security (HSTS) ist eine weitere Methode, die gegen bekannte Schwächen von SSL schützt. Damit wird erschwert, dass ein Besucher durch einen Angriff oder serverseitige Konfigurationsprobleme von einer gesicherten auf eine ungesicherte Seite umgeleitet wird. Befindet sich ein Angreifer beispielsweise in demselben WLAN wie das Opfer, könnte er so die Session Cookies mitlesen und die HTTPS-Session übernehmen. Um HSTS zu aktivieren, muss der HSTS-Header auf dem Server konfiguriert werden.

Schutz des privaten Serverschlüssels

Ein besonders wichtiger Sicherheitsaspekt beim Einsatz von SSL/TLS ist der Schutz des privaten Serverschlüssels. Daher ist es ratsam, den Server so zu konfigurieren, dass der private Serverschlüssel beim Start des Servers durch Passworteingabe freigegeben werden muss. Besteht der Verdacht, dass der private Schlüssel kompromittiert wurde, so muss das zugrunde liegende Zerti-

fikat widerrufen werden. Weitere Hinweise zum Umgang mit kryptografischen Schlüsseln sind in M 2.46 *Geeignetes Schlüsselmanagement* zu finden.

Validierung

Die Auswirkungen von Konfigurationsänderungen auf dem Server lassen sich nicht immer mit Bestimmtheit vorhersagen. Auch Software Updates können mitunter zu überraschenden Änderungen führen. Es wird daher empfohlen, die SSL/TLS Konfiguration vor der Freigabe zur Nutzung auf Fehler zu prüfen und den Status in periodischen Abständen (regelmäßig) zu validieren.

Prüffragen:

- Bieten alle Serverdienste, bei denen es sinnvoll und möglich ist, die Informationen verschlüsselt über SSL/TLS an?
- Wurde sorgfältig eine Zertifizierungsstelle ausgewählt?
- Wurde der Common Name sorgfältig ausgewählt?
- Wurde die vollständige Zertifikatskette im Webserver hinterlegt?
- Unterstützen die eingesetzten Server-Produkte eine sichere Version von SSL/TLS?
- Ist sichergestellt, dass die eingesetzten Server kryptographische Algorithmen und Schlüssellängen verwenden, die dem Stand der Technik und den Sicherheitsanforderungen der Institution entsprechen?
- Wurde die Session Renegotiation deaktiviert oder erfolgt diese auf Basis des RFC 5746? Wurde die clientseitige Renegotiation deaktiviert?
- Wird bei Webseiten auf gemischten Inhalten verzichtet?
- Wurde die TLS-Kompression deaktiviert?
- Wird der private Serverschlüssel durch ein Passwort geschützt?
- Wurde die SSL/TLS Konfiguration vor der Freigabe zur Nutzung auf Fehler geprüft und wird der Status in periodischen Abständen validiert?

			G 2.1	G 2.4	G 2.7	G 2.22	G 2.27	G 2.67	G 2.87	G 2.103	G 2.157	G 2.158	G 2.159	G 3.16	G 3.38	G 3.43	G 4.22	G 4.33	G 4.35	G 4.84	G 4.85	G 4.86	G 4.87	G 5.18	G 5.19	G 5.20	G 5.28	G 5.87	G 5.88	G 5.131
M 2.1	PK	A	X		X																									
M 2.8	BT	A	X		X			X						X												X				
M 2.11	PK	A	X					X										X												
M 2.31	BT	A					X	X						X																X
M 2.34	BT	A					X																							
M 2.35	BT	B							X								X		X									X		
M 2.62	BE	B	X	X								X					X													
M 2.63	PK	A	X		X			X						X										X		X				
M 2.64	BT	A	X	X		X																			X	X				
M 2.80	PK	A					X				X						X													
M 2.110	BT	A						X					X																	
M 2.273	BT	A			X												X													
M 2.363	PK	B																		X										X
M 2.486	PK	A					X				X																			
M 2.487	PK	B		X							X	X		X			X		X											
M 2.488	PK	W											X																	
M 3.5	BT	A							X	X				X	X	X														
M 4.78	BT	A													X		X	X											X	
M 4.176	PK	B																	X						X					
M 4.392	UM	A																X												
M 4.393	UM	B																		X			X							X
M 4.394	UM	A																												
M 4.395	UM	B																			X		X							
M 4.396	UM	B																						X						
M 4.397	BT	C																				X			X	X				
M 4.398	UM	B			X								X		X								X							
M 4.399	UM	A																		X										
M 4.400	BT	B																					X				X			
M 4.401	UM	B																	X				X							
M 4.402	UM	A			X			X																	X	X				
M 4.403	UM	C																												
M 4.404	PK	A									X																		X	
M 4.405	UM	C																					X				X			
M 4.406	UM	Z																												
M 5.150	BT	Z		X					X			X					X		X	X			X					X		
M 5.168	PK	A									X				X			X												
M 5.169	PK	A									X																			
M 5.177	PK	B																										X		

			G 5.165	G 5.166	G 5.167	G 5.168	G 5.169	G 5.170	G 5.171	G 5.172	G 5.173	G 5.174	G 5.175
M 2.1	PK	A											
M 2.8	BT	A								X			
M 2.11	PK	A											
M 2.31	BT	A											
M 2.34	BT	A											
M 2.35	BT	B											
M 2.62	BE	B											
M 2.63	PK	A								X			
M 2.64	BT	A											
M 2.80	PK	A											
M 2.110	BT	A											
M 2.273	BT	A											
M 2.363	PK	B										X	
M 2.486	PK	A											
M 2.487	PK	B				X							
M 2.488	PK	W											
M 3.5	BT	A											
M 4.78	BT	A	X			X					X		
M 4.176	PK	B											
M 4.392	UM	A	X							X			

			G 5.165	G 5.166	G 5.167	G 5.168	G 5.169	G 5.170	G 5.171	G 5.172	G 5.173	G 5.174	G 5.175
M 4.393	UM	B						X				X	
M 4.394	UM	A				X	X						
M 4.395	UM	B			X								
M 4.396	UM	B		X									
M 4.397	BT	C	X										
M 4.398	UM	B	X							X			
M 4.399	UM	A									X		
M 4.400	BT	B											
M 4.401	UM	B	X										
M 4.402	UM	A	X			X				X			
M 4.403	UM	C							X				
M 4.404	PK	A			X								
M 4.405	UM	C											
M 4.406	UM	Z											X
M 5.150	BT	Z	X	X	X		X	X	X	X	X	X	X
M 5.168	PK	A											
M 5.169	PK	A											
M 5.177	PK	B											