Arbeitsrichtlinie
Sichere Entwicklung

VAV Versicherungs-Aktiengesellschaft

Klassifikation: Intern
Version 21.0

VAV VERSICHERUNGEN

Dokumenteneigenschaften

Titel	Arbeitsrichtlinie Sichere Entwicklung
Version	21.0
Geltungsbereich	Siehe Geltungsbereich Richtlinie Informationssicherheit
Erstmalige Freigabe	31.03.2020
Verabschiedet durch	Daniel Fürdauer (Informationssicherheitsbeauftragter)
Klassifikation	Intern
Verantwortlicher Verantwortliche Abteilung	Daniel Fürdauer Datenschutz und Informationssicherheit
Fachlicher Ansprechpartner	Daniel Fürdauer (daniel.fuerdauer@vav.at)
Letztes Review	November 2021

Dokumentenhistorie

Version	Datum	Beschreibung der Änderung	Ersteller
20.0	31.03.2020	Ersterstellung in Anlehnung an die entsprechende Arbeitsrichtlinie der VHV in der Version 20.0.	Daniel Fürdauer
20.1	07.09.2020	Adaptierung Geltungsbereich	Daniel Fürdauer
21.0	12.11.2021	Änderungen in Anlehnung an die entsprechende Arbeitsrichtlinie der VHV in der Version 21.0: Kürzel „VHV-RSS-NEU“ in „VHV-RSE“ umbenannt; Referrer-Policy ist Muss-Kriterium; Klarstellung, dass generische Fehlermeldung dem Benutzer nicht angezeigt, aber protokolliert werden dürfen; RSA-Token gegen FIDO2-Token als Beispiel ersetzt; Automatisierungs-schutz beim Mailversand ergänzt.	Daniel Fürdauer

Hinweis zur Schreibweise

Die verwendete männliche Sprachform dient der leichteren Lesbarkeit und meint immer alle Geschlechter (m/w/d). Die verkürzte Sprachform hat nur redaktionelle Gründe und beinhaltet keine Wertung.

INHALTSVERZEICHNIS

Inhaltsverzeichnis 3

1 Einleitung 12

1.1 Zielsetzung 12

1.2 Geltungsbereich 12

1.2.1 Abgrenzung zu bestehenden Richtlinien 12
1.2.2 Abgrenzung zu verwandten Thematiken 13
1.2.3 Umgang mit nicht zutreffenden Vorgaben 13

1.3 Anwendung 13

1.4 Umgang mit Abweichungen 14

1.5 Arten von Vorgaben 14

1.6 Quellen 15

2 Implementierungsvorgaben 16

2.1 Allgemeine Grundsätze 16

2.1.1 Wiederverwendung von bestehendem Code 16
2.1.2 Minimale Rechteausweitung 16
2.1.3 Minimierung der Angriffsfläche 16
2.1.4 Misstrauensprinzip 16
2.1.5 Mehrschichtige Sicherheit 17
2.1.6 Konfigurierbare Sicherheitseinstellungen 17
2.1.7 Externalisierung von Sicherheitsfunktionen 17
2.1.8 Konsistenz von Sicherheitsprüfungen 17
2.1.9 Prinzip der ausgereiften Sicherheit 17
2.1.10 Testbarkeitsprinzip 18
2.1.11 Whitelisting-Ansatz 18
2.1.12 Bevorzugung von Open Source bei sicherheitskritischen Komponenten 18

2.2 Organisatorische und regulatorische Vorgaben 18

2.2.1 Berücksichtigung von Compliance-Anforderungen für die Software-Entwicklung . 18
2.2.2 Zugriffskonzept („Rollen und Rechte“) 19
2.2.3 Regelung für Behandlung vertraulicher Daten 19
2.2.4 Feststellen der Anwendbarkeit von Datenschutzvorgaben 19
2.2.5 Change-Management für Source Code 20
2.2.6 Keine Veröffentlichung von Source-Code ohne Freigabe 20
2.2.7 Auswahl vertrauenswürdiger Entwicklungswerkzeuge 20
2.2.8 Vier-Augen-Prinzip für sicherheitskritische Komponenten 20
2.2.9 Überprüfung von Third-Party-Code 20

2.2.10 Durchführung von Code-Reviews ... 21
2.2.11 Trennung von Entwicklungs- und Produktivumgebung ... 21
2.2.12 Zugriffsrechtebeschränkungen in Entwicklungsumgebungen ... 21
2.2.13 Sichere Verbindung verteilter Arbeitsplätze ... 21
2.2.14 Übermittlung von Source-Code nur über verschlüsselte Kanäle ... 22
2.2.15 Verwendung nur freigegebener 3rd-Party-Dependency Repositories ... 22
2.2.16 Sichere Installation der entwickelten Software ... 22

2.3 Übergreifende Programmiervorgaben ... 22

2.3.1 Anwendungsfluss entsprechend Geschäftslogik ... 22
2.3.2 Funktionstrennung für privilegierte Anwendungslogik ... 23
2.3.3 Verwendung unsicherer Funktionen vermeiden ... 23
2.3.4 Explizite Initialisierung von Variablen ... 23
2.3.5 Angemessene Kommentierung von Source Code ... 23
2.3.6 Vermeidung von Berechnungsfehlern ... 23
2.3.7 Vermeidung von Verwundbarkeiten durch weitere Anwendungskomponenten ... 24
2.3.8 Behandlung einzubindender Ressourcen ... 24
2.3.9 Zugriffsbeschränkung bei Einbindung externer und interner Ressourcen ... 24
2.3.10 Minimalprinzip für Verwendung externer Ressourcen ... 24
2.3.11 Sichere Ablage eingebundener Ressourcen ... 25
2.3.12 Code-Generierung oder -Modifizierung einschränken ... 25
2.3.13 Kapselung von Betriebssystemfunktionalität ... 25
2.3.14 Verwendung von Prüfsummen für sensible Dateien ... 25
2.3.15 Limitierung von Transaktionen ... 26
2.3.16 Definition der genutzten HTTP-Methoden ... 26
2.3.17 Sichere Update-Mechanismen ... 26

2.4 Eingabevalidierung ... 27

2.4.1 Validierung nicht vertrauenswürdiger Daten ... 27
2.4.2 Eingabeprüfung sämtlicher Parameter ... 27
2.4.3 Eingabedatenvalidierung in vertrauenswürdigen Systemen ... 27
2.4.4 Zentrale Eingabedatenvalidierung ... 27
2.4.5 Mindestanforderungen an die Eingabedatenvalidierung ... 28
2.4.6 Datenvalidierung: Verwendung eines Whitelisting-Ansatzes ... 28
2.4.7 Zurückweisung von nicht validierbaren Eingabedaten ... 28
2.4.8 Spezifizierung des Character-Sets ... 28
2.4.9 (Re)kodierung in verwendetes Character-Set ... 28
2.4.10 (Re)kodierung von Sonderzeichen ... 29
2.4.11 Bereinigung von formatierten Eingabedaten ... 29
2.4.12 Keine Ausführung nicht vertrauenswürdiger Daten ... 29
2.4.13 Berücksichtigung aller benutzerdefinierten Eingabedatenquellen bei Webanwendungen ... 29
2.4.14 Validierung des Header-Datenformats in Webanwendungen ... 30
2.4.15 Normalisierung von Pfadangaben ... 30
2.4.16 Sichere IDs für Resourcenreferenzierung ... 30

2.4.17 Validierung von HTTP-Variablen 30
2.4.18 Datenvalidierung: Behandlung von ungültigen Werten 31
2.4.19 Enkodierung von Kommandozeilenparametern 31

2.5 Ausgabevalidierung (Enkodierung & Escaping) 31

2.5.1 Kodierung von Ausgabedaten in vertrauenswürdigen Systemen 31
2.5.2 Zentrale Ausgabedatenvalidierung 31
2.5.3 Kodierung von Ausgabedaten über standardisierte Funktionen 32
2.5.4 Berücksichtigung des Kontexts bei Kodierung von Ausgabedaten 32
2.5.5 Kodierung aller möglicherweise unsicheren Zeichen 32
2.5.6 Implizite Ausgabevalidierung 32
2.5.7 Definierter Content-Type von HTTP-Antworten 33
2.5.8 Sichere HTTP-Weiterleitungen 33

2.6 Authentifizierung & Registrierung von Benutzern 33

2.6.1 Authentifizierung by Default 33
2.6.2 Authentifizierung durch vertrauenswürdige Systeme 34
2.6.3 Verwendung von standardisierten Authentifizierungsfunktionen 34
2.6.4 Zentrale Authentifizierungsimplementierung 34
2.6.5 Logische Trennung des Authentifizierungsmechanismus 34
2.6.6 Sichere Fehlerbehandlung bei der Authentifizierung 35
2.6.7 Kein Hinweis auf falsche Authentifizierungsdatenteile 35
2.6.8 Überprüfung aller für Authentifizierung benötigten Daten 35
2.6.9 Authentifizierung für sensible Drittsysteme 35
2.6.10 HTTP-Authentifizierung via POST 36
2.6.11 Brute Force-Gegenmaßnahmen 36
2.6.12 Keine „Anmeldedaten speichern“-Funktionalität 36
2.6.13 Benachrichtigung über letzten Anmeldeversuch 36
2.6.14 Zwei-Faktor-Authentifizierung bei Registrierung für Anwendungen mit hohem Schutzbedarf 37
2.6.15 Vollständige Benutzerregistrierung vor Anmeldung 37
2.6.16 Verwendung sicherer Benutzerkennungen 37
2.6.17 Benutzerpasswörter entsprechend Passwort-Policy 37
2.6.18 Keine ausschließliche Zugriffskontrolle via HTTP Basic 37
2.6.19 Sichere Fehlerbehandlung bei Authentisierungskontrollen 38
2.6.20 Reauthentisierung bei sensiblen Operationen 38
2.6.21 Abmeldefunktionalität 38

2.7 Benutzerpasswörter I: Stärke und Behandlung 38

2.7.1 Umsetzung der Passwortanforderungen 38
2.7.2 Standardvorgaben für Benutzerpasswörter 39
2.7.3 Verdeckte Eingabe in Passworteingabefeld 39
2.7.4 Keine Autovervollständigung für Passwortfelder 39
2.7.5 Keine unverschlüsselte Übertragung nicht-temporärer Passwörter 39
2.7.6 Ablaufzeit für temporäre Passwörter / Tokens 39

2.7.7 Salting von Passwort-Hashes ... 40
2.7.8 Zugriffsbeschränkung für Zugangsdaten ... 40
2.8 Benutzerpasswörter II: Änderung und Zurücksetzung ... 40
2.8.1 Änderbarkeit von Passwörtern ... 40
2.8.2 Änderung von Initialpasswörtern ... 40
2.8.3 Erneute Authentifizierung bei Passwortänderung ... 41
2.8.4 Passwortänderung erzwingen ... 41
2.8.5 Einheitliches Verfahren für Passwortänderung ... 41
2.8.6 Sicherheitsniveau der Zugangsdatenänderung ... 41
2.8.7 Verhindern von Passwortwiederverwendung ... 41
2.8.8 Anzeige der Passwortstärke ... 42
2.9 Authentifizierung am Backend ... 42
2.9.1 Sichere Authentifizierung gegen administrative Backends ... 42
2.9.2 Multi-Faktor-Authentifizierung für streng vertrauliche Konten ... 42
2.9.3 Erneute Authentifizierung vor Durchführung kritischer Aktionen ... 42
2.9.4 Verwendung verschlüsselter Kanäle ... 43
2.9.5 Minimalprinzip für Rechte von Service- und Support-Konten ... 43
2.9.6 Gegenseitige Authentifizierung verteilter Anwendungskomponenten ... 43
2.10 Dateiuploads und -downloads ... 43
2.10.1 Authentifizierung vor Upload ... 43
2.10.2 Whitelisting erlaubter Dateitypen bei Upload ... 44
2.10.3 Verifizierung des MIME-Types von Uploads ... 44
2.10.4 Kein Upload aktiver Dateien ... 44
2.10.5 Größenbeschränkung für Uploads ... 44
2.10.6 Rate Limiting für Uploads ... 44
2.10.7 Antivirenscan bei Upload ... 45
2.10.8 Konvertierung von Dateiuploads ... 45
2.10.9 Zugriffsgeschützte Ablage von Uploads ... 45
2.10.10 Keine Ausführungsberechtigungen in Upload-Verzeichnissen ... 45
2.10.11 Dateiablage für Uploads außerhalb der Applikationsverzeichnisse ... 46
2.10.12 Sichere Behandlung von Upload-Verzeichnissen in UNIX-Umgebungen ... 46
2.10.13 Security Header für Downloads ... 46
2.10.14 Referenzierung nur erlaubter Dateinamen und -typen ... 46
2.10.15 Downloads über separate Domain ... 46
2.11 Absicherung des Session-Managements ... 47
2.11.1 Zentrales Sitzungsmanagement ... 47
2.11.2 Erstellung von Sitzungs-Identifiern in vertrauenswürdigen Systemen ... 47
2.11.3 Geprüfte Sitzungsmanagement-Mechanismen mit ausreichend Entropie ... 47
2.11.4 Bindung der IP-Adresse an die Session ... 47
2.11.5 Keine Nutzung von Sitzungs-Identifiern außerhalb vorgesehener Kanäle ... 48
2.11.6 Serverseitiger Schutz der Sitzungsdaten ... 48

2.11.7 Sitzungs-Cookies mit Domain und Path ... 48
2.11.8 Sichere Übermittlung von Sitzungs-IDs ... 48
2.11.9 Cookie-„secure“-Attribut ... 49
2.11.10 Cookie-„HttpOnly“-Attribut ... 49
2.11.11 Bestehende Sitzung bei Anmelden terminieren ... 49
2.11.12 Mehrfache Anwendersitzungen nicht zulassen ... 49
2.11.13 Sitzung bei Abmeldung vollständig terminieren ... 50
2.11.14 Abmeldefunktionalität aus authentifizierten Bereichen erreichbar ... 50
2.11.15 Sitzungsablaufdauer ... 50
2.11.16 Keine persistenten Sitzungen ... 50
2.11.17 Neuer Sitzungs-Identifier bei Reauthentifizierung ... 51
2.11.18 Periodische Erstellung neuer Sitzungs-Identifier ... 51
2.11.19 Periodische erneute Zugriffskontrolle bei lang andauernden authentifizierten Sitzungen ... 51
2.11.20 Neuer Sitzungs-Identifier bei Wechsel von HTTP zu HTTPS ... 51
2.11.21 Tokens für sensible Funktionen ... 52

2.12 Zugriffskontrollen (Access Controls) ... 52

2.12.1 Autorisierung by Default ... 52
2.12.2 Zentraler Autorisierungsmechanismus ... 52
2.12.3 Prüfung von Rollen und Rechten ... 53
2.12.4 Protokollierung bei der Autorisierung ... 53
2.12.5 Autorisierung auf Basis vertrauenswürdiger Informationsobjekte ... 53
2.12.6 Zugriffsprüfungen auf mehreren Ebenen ... 53
2.12.7 Sichere Fehlerbehandlung bei der Autorisierung ... 54
2.12.8 Zugriffskontrolle bei allen Anfragen ... 54
2.12.9 Dedizierte Autorisierung sensibler Objektzugriffe ... 54
2.12.10 Zusätzliche Absicherung sicherheitskritischer Bereiche ... 54
2.12.11 Checkliste Zugriffskontrolle ... 55
2.12.12 Übereinstimmung serverseitiger Zugriffskontrollmechanismen mit Informationen in Präsentationsschicht ... 55
2.12.13 Keine Autorisierung auf Basis nicht vertrauenswürdiger Informationen ... 55
2.12.14 Kontoauditierung und Deaktivierung nicht genutzter Konten ... 55
2.12.15 Kontodeaktivierung und Sitzungsterminierung bei Autorisierungsentzug ... 56
2.12.16 Ablauf / Revalidierung von Benutzerberechtigungen ... 56
2.12.17 „Least Privilege“-Prinzip für Anwenderkonten ... 56
2.12.18 Vermeidung von Enumeration ... 56
2.12.19 Cross-Domain-Zugriffe ... 57
2.12.20 CSRF-Schutz ... 57
2.12.21 Minimalprinzip für Anwendungsrechte ... 57

2.13 Fehlerbehandlung & Protokollierung ... 58

2.13.1 Vermeidung einer zu hohen Fehlertoleranz ... 58
2.13.2 Abfangen von Fehlern ... 59
2.13.3 Vermeidung unsicherer Anwendungszustände ... 59

2.13.4 Sichere Fehlerbehandlung bei Systemvorgängen 59
2.13.5 Fehlerbehandlung in Sicherheitskontrollmechanismen 59
2.13.6 Ressourcen im Fehlerfall freigeben 60
2.13.7 Generische Fehlermeldungen und spezielle Fehlerseiten 60
2.13.8 Fehlerbehandlung ohne Ausgabe von Debug-Informationen oder Stack Traces 60
2.13.9 Keine vertraulichen Daten in Fehlermeldungen 60
2.13.10 Zentrale Protokollierungsfunktionalität 60
2.13.11 Protokollierungsfunktionen in vertrauenswürdigen Systemen 61
2.13.12 Monitoring gegen kontoübergreifende Brute Force-Angriffe 61
2.13.13 Protokollierung von Ergebnissen der Sicherheitskontrollmechanismen 61
2.13.14 Protokollierung relevanter Metadaten 61
2.13.15 Sicheres Escaping von Protokolldaten 62
2.13.16 Zugriffsbeschränkung für Protokolldaten 62
2.13.17 Keine vertraulichen Daten in Protokolleinträgen 62
2.13.18 Protokollanalyse 62
2.13.19 Checkliste Logging 63
2.13.20 Verwendung kryptographischer Hash-Funktion zur Validierung von Protokolleinträgen 63
2.13.21 Einhaltung von Datenschutzvorgaben bei der Protokollierung 63

2.14 Datensicherheit & Information Disclosure 63

2.14.1 Vermeidung von Race-Conditions 64
2.14.2 Vermeidung von gleichzeitigem Zugriff auf mehrfach genutzte Ressourcen 64
2.14.3 Enumerierung erlaubter und Deaktivierung nicht benötigter HTTP-Anfragemethoden 64
2.14.4 Keine vertraulichen Informationen in HTTP GET-Anfrageparametern 64
2.14.5 Verwendung von POST für einen Zustand ändernde Anfragen 65
2.14.6 Keine Weiterleitung zu benutzerdefinierten URLs 65
2.14.7 Character-Encoding für alle Verbindungen definieren 65
2.14.8 Kein Including benutzerdefinierter Daten 65
2.14.9 Keine Referenzierung von Dateien oder Verzeichnissen über Pfadnamen 66
2.14.10 Schutz temporärer Daten und Caches 66
2.14.11 Kein clientseitiges Caching vertraulicher Daten 66
2.14.12 Sicheres Löschen nicht mehr benötigter Daten 66
2.14.13 Minimalitätsprinzip für Informationen 67
2.14.14 Keine internen Informationen in Kommentaren 67
2.14.15 Vermeidung von Fingerprinting 67
2.14.16 Keine Ausgabe absoluter Systempfade 67
2.14.17 Keine Übermittlung vertraulicher Informationen in Links auf Drittsysteme 68
2.14.18 Integritätsprüfung bei Einbindung externer Inhalte 68
2.14.19 Sicherer Transport interner Daten 68
2.14.20 Weiterleitung auf HTTPS 68
2.14.21 Präventive Weiterleitung auf HTTPS 69
2.14.22 Kein Transport vertraulicher Daten über Metadaten 69

2.15 Speicher-, Prozess- und Ressourcenzugriffs-Management 69
2.15.1 Überprüfung der Buffer-Größen 69
2.15.2 Berücksichtigung des Speicherplatzbedarfs für den Null-Terminator 70
2.15.3 Buffer-Grenzen berücksichtigen 70
2.15.4 Freigabe von allokierten Ressourcen 70
2.15.5 Freigabe von allokierten Speicherbereichen 70
2.15.6 „Least Privilege“-Prinzip für Prozesse 71
2.16 Datenbanken 71
2.16.1 Verwendung von parametrisierten Datenbankanfragen (Prepared Statements) ... 71
2.16.2 Sichere Enkodierung für Backend-Zugriffe 71
2.16.3 Eingabedatenvalidierung und Ausgabekodierung bei Datenbankzugriff 71
2.16.4 Verwendung stark typisierter Variablen 72
2.16.5 Verwendung von Stored Procedures 72
2.16.6 Datenbankverbindung zeitnah schließen 72
2.16.7 Verwendung von Datenbankverbindungen mit unterschiedlichen Berechtigungsabstufungen 72
2.17 Kryptographie 73
2.17.1 Kryptographische Funktionen in vertrauenswürdigen Systemen 73
2.17.2 Verwendung sicherer kryptographischer Funktionen 73
2.17.3 Sichere Fehlerbehandlung bei kryptographischen Funktionen 73
2.17.4 Sichere Zufallswerte 74
2.17.5 Protokollierung von Fehlern in kryptographischen Funktionen 74
2.17.6 Transportverschlüsselung für Daten 74
2.17.7 TLS für authentifizierte Verbindungen 74
2.17.8 TLS für vertrauliche Verbindungen zu Drittsystemen 75
2.17.9 Zentrale, standardisiert konfigurierte TLS-Implementierung 75
2.17.10 Sichere TLS-Konfiguration 75
2.17.11 Perfect Forward Secrecy 75
2.17.12 Keine unsichere Verbindung bei fehlgeschlagener TLS-Verbindung 76
2.17.13 Sichere Ablage von Schlüsseln und Authentifizierungsdaten 76
2.18 X.509-Zertifikate („SSL/TLS-Zertifikate“) 76
2.18.1 Einsatz gültiger Zertifikate 76
2.18.2 X.509 / TLS: Schlüssellängen 77
2.18.3 X.509 / TLS: Anforderungen für Anwendungen mit hohem Schutzbedarf 77
2.18.4 Validierung von TLS-Zertifikaten 77
2.19 Clientseitige Sicherheit 77
2.19.1 Keine clientseitige Speicherung vertraulicher Daten 77
2.19.2 Clientseitige Speicherung vertraulicher Daten 78
2.19.3 Keine automatische Vervollständigung für Eingabefelder für vertrauliche Daten .. 78
2.19.4 Integritätsprüfung für Zustandsdaten 78
2.19.5 JavaScript: Sicherer Umgang mit JSON 78

2.19.6 JavaScript: Sichere Content-Manipulation 78
2.19.7 JavaScript: Sichere Transportschicht für APIs 79
2.19.8 Keine Offenlegung vertraulicher Geschäftslogik 79
2.19.9 Verwendung des „HttpOnly“-Attributs für Cookies 79
2.19.10 Verwendung des „secure“-Attributs für Cookies 79
2.19.11 Checkliste Vermeidung von UI-Spoofing 79
2.19.12 Verwendung signierter Werte 80

2.20 Webdienste und XML-Parser 80

2.20.1 Kapselung interner Services für externen Zugriff 80
2.20.2 Validierung nicht vertrauenswürdiger XML-Daten 80
2.20.3 Härtung von XML-Parsern für Validierung nicht vertrauenswürdiger XML-Daten .. 81
2.20.4 Authentifizierung bei REST-Services 81
2.20.5 Sichere Übermittlung von Skriptinhalten 81
2.20.6 CSRF-Schutz 81
2.20.7 Übermittlung vertraulicher Daten über WebSockets 81
2.20.8 Äquivalente Sicherheitsfunktionen für Webdienste 82
2.20.9 Verwendung authentisierter Verbindungen zu anderen Systemen 82

2.21 Datenschutzvorgaben 82

2.21.1 Feststellen der Anwendbarkeit von Datenschutzvorgaben 82
2.21.2 Verarbeitung besonderer Kategorien personenbezogener Daten 83
2.21.3 Datenschutzvorgabenrelevanz für Scoring 83
2.21.4 Abstimmung mit Datenschutzbeauftragtem beim Scoring 83
2.21.5 Verfahrensvorgaben für Scoring 83
2.21.6 Keine ausschließliche Verwendung von Anschriftendaten beim Scoring 84
2.21.7 Informations- und Dokumentationspflichten beim Scoring 84
2.21.8 Keine Verwendung besonderer Kategorien personenbezogener Daten beim Scoring 84
2.21.9 Minimalprinzip für Erhebung und Verarbeitung personenbezogener Daten 84
2.21.10 Minimalprinzip für Identifizierbarkeit durch personenbezogene Daten 85
2.21.11 Einwilligung in die Verarbeitung personenbezogener Daten 85
2.21.12 Widerruf der Verarbeitung personenbezogener Daten 85
2.21.13 Verarbeitung personenbezogener Daten durch Dritte 86
2.21.14 Datenaufbereitung für Auskunftsrecht 86
2.21.15 Berichtigung personenbezogener Daten 87
2.21.16 Löschung personenbezogener Daten 87
2.21.17 Einschränkung der Verarbeitung personenbezogener Daten 87
2.21.18 Datenübertragbarkeit personenbezogener Daten 87

2.22 Sichere Konfiguration 88

2.22.1 Minimalprinzip für Konfigurationen 88
2.22.2 Zugriffsbeschränkung für Konfigurationsinformationen 88
2.22.3 Sicherstellung vorhandener Sicherheitskonfiguration 88
2.22.4 Auditierbare Sicherheitskonfiguration 88

2.22.5 Keine Verwendung von Standardanmeldedaten ... 88
2.22.6 Begrenzung der Zugriffsrechte auf Zugangsdaten ... 89
2.22.7 Schutz des Quellcodes ... 89
2.22.8 Keine Veröffentlichung nicht öffentlich benötigter Dokumentation ... 89
2.22.9 Zugriffskontrollen für vertrauliche serverseitige Daten ... 89
2.22.10 Einsatz aktueller freigegebener Software-Versionen ... 89
2.22.11 Einsatz aller verfügbaren Patches ... 90
2.22.12 Kein Directory Listing ... 90
2.22.13 „Least Privilege“-Prinzip für Systemkonten ... 90
2.22.14 Entfernen aller Testdaten und -funktionen vor Produktivstellung ... 90
2.22.15 Entfernen nicht benötigter Informationen vor Produktivstellung ... 91
2.22.16 Kein Information Disclosure über robots.txt ... 91
2.22.17 Security Header zur Unterbindung von Cross-Site-Scripting (XSS) ... 91
2.22.18 Security Header zur Vermeidung von Clickjacking ... 92
2.22.19 Security Header zum Whitelisting von Inhalten ... 92
2.22.20 Kein Information Disclosure über HTTP-Header ... 92
2.22.21 Schreibschutz für Anwendungsdateien ... 92
2.22.22 „Least Privilege“-Prinzip für Datenbankkonten ... 93
2.22.23 Sichere Zugangsdaten für Datenbankzugriff ... 93
2.22.24 Sichere Authentifizierung für Datenbankadministration ... 93
2.22.25 Deaktivierung nicht benötigter Datenbankfunktionalität ... 93
2.22.26 Entfernen nicht benötigter Standarddatenbankinhalte ... 93
2.22.27 Deaktivieren nicht benötigter Datenbankkonten ... 94

3 Ergänzende Inhalte ... 95

3.1 HTTP-Security Header-Sicherheitsmechanismen ... 95

3.1.1 Content-Security-Policy ... 95
3.1.2 Referrer-Policy ... 95
3.1.3 Set-Cookie ... 95
3.1.4 Strict-Transport-Security ... 96
3.1.5 X-Content-Type-Options ... 96
3.1.6 X-Frame-Options ... 96
3.1.7 X-XSS-Protection ... 96
3.1.8 Content-Disposition ... 97
3.1.9 X-Download-Options ... 97

3.2 Häufige Schwachstellen in Webanwendungen (OWASP TOP 10) ... 97

3.3 Vorgaben der Richtlinie „Sichere Entwicklung“ in tabellarischer Form ... 98

4 Glossar ... 98

1 EINLEITUNG

1.1 Zielsetzung

Sinn und Zweck dieser Richtlinie ist es, die wachsenden Anforderungen an die bei der Softwareentwicklung zu berücksichtigenden Sicherheitsaspekte in deren Implementierung zu adressieren.

1.2 Geltungsbereich

Diese Richtlinie gilt 3 Monate nach dem jeweiligen Änderungsdatum (siehe „Dokumentenhistorie“), frühestens jedoch 2 Jahre nach erstmaliger Freigabe (siehe „Dokumenteneigenschaften“), für alle von der VAV neu- bzw. (teil-)entwickelten oder in Auftrag gegebenen Anwendungen, Anwendungskomponenten und insbesondere Web- bzw. Intranetanwendungen. Sie kann auch für die Bewertung von fremdentwickelter Software (z. B. [Standard-]Kaufsoftware oder Cloudprodukten) herangezogen werden.
Auf Basis dieser Richtlinie können spezifische Richtlinien für eingesetzte Technologien/Software erstellt werden. Diese können weitere Vorgaben regeln, aber keine Vorgaben dieser Richtlinie aufheben. Sie können allerdings beschreiben, warum bestimmte Vorgaben dieser Richtlinie für die eingesetzte Technologie/Software nicht zutreffend sind. Die nicht zutreffenden Vorgaben sind vorab mit der Stabstelle Datenschutz und Informationssicherheit abzustimmen. Ein Beispiel für eine solche Zusatzrichtlinie ist die SAP-Entwicklungsrichtlinie der VHV.

1.2.1 Abgrenzung zu bestehenden Richtlinien

Es existiert eine Reihe interner Dokumente und Richtlinien, die bei der Erstellung dieser Richtlinie berücksichtigt wurden (Auszug):

- Basiert auf dem BSI / SecureNet-Dokument „Sicherheit von Webanwendungen – Maßnahmenkatalog und Best Practices“ (Version 1, August 2006), da dieses Dokument, die (Web-)Anwendungssicherheit nicht vollständig oder zeitgemäß adressiert, wurde diese entsprechend erweitert und abgeändert. Sinnvolle Maßnahmen des Quelldokuments wurden jedoch in dieser Richtlinie berücksichtigt.
- „Arbeitsrichtlinie Kryptographie“ und „Arbeitsrichtlinie Kennwörter“:
  Die vorliegende Richtlinie konkretisiert bewusst nicht Vorgaben zu kryptographischen Maßnahmen und Kennwörtern, um den Geltungsbereich zu diesen Thematiken klar abgegrenzt zu lassen und Redundanzen hinsichtlich von Vorgaben zu vermeiden.

1.2.2 Abgrenzung zu verwandten Thematiken

Die folgenden Punkte beziehen sich auf Thematiken und Phasen im SDLC, die nicht unmittelbar zu der Implementierung, also Programmierung, von Anwendungen gehören:

- Beheben von Schwachstellen in Anwendungen
- Sicherheit für den Betrieb von Anwendungen
- Sicherheit von Source- und Programmcode
- Sicherheit im Entwicklungsprozess
- Sicherheitstests (Code Audits, Vulnerability Scans, Penetrationstests usw.)
- Zulieferervorgaben (Vertragsgestaltungen, Zusagen & Garantien, Qualifikationen usw.)

Zu diesen Punkten sind in dieser Richtlinie lediglich vereinzelt allgemeingültige Vorgaben enthalten (z. B. zur „Sicherheit von Source- und Programmcode“). Da diese Punkte nicht im Fokus dieser Richtlinie stehen, besteht hier kein Anspruch auf Vollständigkeit. Es ist sicherzustellen, dass diese Punkte im SDLC berücksichtigt wurden oder noch werden; alternativ ad-hoc auf Basis von Konzeptdokumenten je Applikation.

1.2.3 Umgang mit nicht zutreffenden Vorgaben

Treffen einzelne Vorgaben dieser Richtlinie auf die eingesetzte Technologie/Software nicht zu, dann sollten (sofern die adressierten möglichen Schwachstellen für die eingesetzte Technologie/Software relevant sind) an Stelle der nicht zutreffenden Vorgaben wirksame und durchführbare Maßnahmen identifiziert und umgesetzt werden, deren Wirkungen der IT-Sicherheitszielsetzung der Vorgaben weitmöglichst entsprechen.

1.3 Anwendung

Diese Richtlinie gilt für Umsetzer (z. B. Anwendungsentwickler) – nachfolgend Entwickler genannt – von im Rahmen der Definition des Geltungsbereichs (siehe 1.2) definierten Produkten und ist daher unter anderem im SEP zu berücksichtigen.
Nachfolgend wird der Begriff des „Anwendungsverantwortlichen“ verwendet: Der Anwendungsverantwortliche ist zum Beispiel, aber nicht abschließend, ein Maßnahmen- oder Steckbriefverantwortlicher, in Projekten der IT-Projektleiter oder bei agilen Vorgehen der „Produkt Owner“, sowie ohne IT-Beteiligung der für die jeweilige Software verantwortliche (Fachbereichs-)Mitarbeiter.
Der Anwendungsverantwortliche stellt sicher, dass die auf die jeweilige Anwendung / das jeweilige Projekt zutreffenden Vorgaben dieser Richtlinie eingehalten bzw. umgesetzt werden.
Daher sollte der Anwendungsverantwortliche mindestens sicherstellen, dass beteiligten externen Entwicklern diese Arbeitsrichtlinie in der jeweils aktuellsten Version vorliegt und kann die Entwickler ggf. auf für das Projekt relevante Vorgaben explizit hinweisen.
Die Art und Weise, wie eine Einhaltung der Vorgaben innerhalb der Umsetzung durchgesetzt oder in welchem Umfang diese gar kontrolliert werden, obliegt dem Anwendungsverantwortlichen. Die Prüfung der Einhaltung dieser Richtlinie im Rahmen von Meilensteinabnahmen in der Umsetzung seitens der Stabstelle Datenschutz und Informationssicherheit bleiben hiervon unberührt. Der Anwendungsverantwortliche kann die Einhaltung der Vorgaben dieser Richtlinie durch Tracking mittels eines Questionnaires zu den Vorgaben (siehe Kapitel „Vorgaben der Richtlinie ‚Sichere Entwicklung'

in tabellarischer Form“) erfassen und so ggf. nachweisen – dieses Vorgehen ist jedoch nicht verpflichtend.

1.4 Umgang mit Abweichungen

Eine absehbare oder erkannte Abweichung von den für die Anwendung / das Projekt zutreffenden Vorgaben ist zunächst von dem jeweiligen Entwickler (bzw. der Person, welche die Abweichung identifiziert) mit dessen Gruppenleiter abzustimmen. Dieser entscheidet, wie die aus der Abweichung resultierenden Risiken zu bewerten sind und wie hiermit umgegangen werden soll.
Hierzu kann der Gruppenleiter die Stabstelle Datenschutz und Informationssicherheit einbeziehen. Dies sollte im Regelfall derart geschehen, dass der Gruppenleiter den Entwickler direkt an die Stabstelle Datenschutz und Informationssicherheit verweist – so dass die Kommunikation hierzu direkt zwischen Entwickler und Stabstelle Datenschutz und Informationssicherheit ablaufen kann.

1.5 Arten von Vorgaben

In diesem Standard werden gemäß RFC 2119, zwei grundsätzliche Arten von Vorgaben unterschieden:

- **Verbindliche Vorgaben:** Kennzeichnung durch Schlüsselworte „MUSS“, „MÜSSEN“, „DARF NICHT“ etc.
- **Empfehlungen:** Kennzeichnung durch die Schlüsselworte „SOLLTE“, „KANN“ etc.

Empfehlungen, die mit den entsprechenden Schlüsselwörtern gekennzeichnet sind, beziehen sich auf Anwendungen, die ein höheres Schutzniveau erfüllen müssen. Dies sind alle Anwendungen, welche einen Schutzbedarf von „erhöht“ oder „hoch“ aufweisen.

1.6 Quellen

Aus den folgenden Quellen wurden Vorgaben für diese Richtlinie aggregiert:

KÜRZEL	NAME	AUTOR	VERSION	DATUM
BSI-WEB-AN	Leitfaden zur Entwicklung sicherer Webanwendungen – Empfehlungen und Anforderungen an die Auftragnehmer	SEC Consult / BSI	20130903	03.09.2013
BSI-WEB-MBP	Sicherheit von Webanwendungen – Maßnahmenkatalog und Best Practices	SecureNet / BSI	1	01.08.2006
BSI-ITSGS-B5.27	BSI IT-Grundschutzkataloge – Baustein B 5.27 Software-Entwicklung	BSI	EL16	19.04.2016
EU-DSGVO	EU-Verordnung 2016 / 679 (DSGVO – Datenschutz-Grundverordnung)	Rat der EU	20160406	06.04.2016
DSG	Datenschutzgesetz	Nationalrat Ö	20190115	15.01.2019
OWASP-SCP	OWASP Secure Coding Practices Quick Reference Guide	OWASP Foundation	2.0	17.11.2010
TSS-WEB	Vorlage für Web-Security-Standard (TSS-WEB)	Secondis GmbH	1.5	15.08.2017
VHV-RSE	Richtlinie Sichere Entwicklung (dieses Dokument)	VHV Gruppe	(siehe Dokumenteigenschaften)	

Tabelle 1: Verwendete Quellen

Da viele Vorgaben aus unterschiedlichen Quellen zusammengefasst, konkretisiert, umgeschrieben usw. wurden, sowie aus Gründen der Vereinheitlichung und Vereinfachung, enthalten die einzelnen Vorgaben dieser Richtlinie nachfolgend (in Kapitel 2 „Implementierungsvorgaben“) jeweils einen Verweis auf die Primärquelle.

Aus dem Vorgabendokument in Tabellenform (siehe Kapitel „Vorgaben der Richtlinie ‚Sichere Entwicklung' in tabellarischer Form“) ist zudem anhand der ID die jeweilige originäre Maßnahmen-Nr. aus dem Quelldokument entnehmbar.

2 IMPLEMENTIERUNGSVORGABEN

2.1 Allgemeine Grundsätze

Diese allgemeinen Grundsätze gelten übergreifend bzw. sollten schon vor der eigentlichen Programmierung Beachtung finden. Aus ihnen leiten sich eine Vielzahl der nachfolgenden Vorgaben ab.

2.1.1 Wiederverwendung von bestehendem Code

Vorgabe: Für übliche Funktionen SOLLTE bereits getesteter und freigegebener Code verwendet werden, anstatt diesen jeweils neu zu erstellen.
Hinweise: Es SOLLTE getesteter Code verwendet werden, falls dies nicht der Fall ist, SOLLTEN Tests erstellt werden.
Quelle(n): OWASP-SCP

2.1.2 Minimale Rechteausweitung

Vorgabe: In Fällen in denen Anwendungsteile mit erhöhten Rechten ausgeführt werden müssen MUSS die Rechteausweitung weitmöglichst eingeschränkt werden, so spät wie nötig stattfinden und so früh wie möglich wieder aufgehoben werden.
Hinweise: Unter UNIX Betriebssystemen ist dies beispielsweise mittels „setuid“ bzw. „setgid“ möglich.
Quelle(n): OWASP-SCP

2.1.3 Minimierung der Angriffsfläche

Vorgabe: Nicht erforderliche oder benötigte Schnittstellen, Funktionen, Parameter, Dateien, Dienste und Protokolle MÜSSEN auf extern erreichbaren Systemen und SOLLTEN auf allen sonstigen Systemen nicht vorhanden sein bzw. deaktiviert oder entfernt werden.
Hinweise: Es SOLLTEN Tools zur statischen Codeanalyse verwendet werden, um tote Codestellen zu erkennen und zu entfernen.
Quelle(n): TSS-WEB

2.1.4 Misstrauensprinzip

Vorgabe: Daten, die von einem Client stammen, MUSS stets misstraut und diese daher entsprechend validiert werden.
Hinweise: Hierzu MÜSSEN vorrangig die im Kapitel „Eingabevalidierung“ enthaltenen Vorgaben beachtet werden.
Quelle(n): TSS-WEB

2.1.5 Mehrschichtige Sicherheit

Vorgabe: Es SOLLTE stets eine mehrschichtige Sicherheit (Defense-in-Depth-Prinzip) implementiert werden.
Hinweise: Dies bedeutet, dass ein Sicherheitsziel (z. B. Schutz gegen Cross-Site-Scripting-Angriffe) soweit möglich stets auf mehreren Ebenen sichergestellt wird (hier z. B. durch X-XSS-Protection Header [siehe auch Anhang] und sichere Kodierung von Ausgabedaten).
Quelle(n): TSS-WEB

2.1.6 Konfigurierbare Sicherheitseinstellungen

Vorgabe: Sicherheit SOLLTE stets deklarativ (mittels Konfigurationsanweisungen oder Annotationen) anstatt programmatisch (mittels Programmcode) parametrisiert werden.
Hinweise: Weitere konkrete Vorgaben hierzu MÜSSEN aus dem Kapitel „Sichere Konfiguration“ beachtet werden.
Quelle(n): TSS-WEB

2.1.7 Externalisierung von Sicherheitsfunktionen

Vorgabe: Wo dies sinnvoll möglich ist, SOLLTEN Sicherheitsfunktionen ausgelagert werden (z. B. durch Nutzung vorhandener Authentifizierungssysteme).
Hinweise: Es SOLLTE also, soweit möglich, z. B. ein Active Directory eingebunden werden, statt eine eigene Benutzerführung mit Rollen- und Rechtesystem zu implementieren.
Quelle(n): TSS-WEB

2.1.8 Konsistenz von Sicherheitsprüfungen

Vorgabe: Identische Sicherheitsprüfungen (z. B. einmal im Web Frontend und ein anderes Mal bei einer AJAX-Schnittstelle) SOLLTEN stets über dieselben Sicherheitsfunktionen (Programmcode) und entsprechend logisch identischer Policies durchgeführt werden.
Quelle(n): TSS-WEB

2.1.9 Prinzip der ausgereiften Sicherheit

Vorgabe: Sicherheitsrelevanter Programmcode SOLLTE stets über ausgereifte Technologien, Algorithmen und Implementierungen (APIs, Frameworks etc.) abgebildet werden.
Hinweise: Für Kryptochiffren oder Passwortstärke KANN sich hierbei bspw. an BSI-Richtlinien und MUSS sich ggf. an weiteren vorhandenen VAV-internen Vorgaben orientiert werden.
Quelle(n): TSS-WEB

2.1.10 Testbarkeitsprinzip

Vorgabe: Vor dem Einsatz neuer Technologien (Protokollen, Frameworks, APIs etc.) SOLLTE sichergestellt werden, dass sich diese auf mögliche Sicherheitsprobleme hin analysieren lassen (eingesetzte Werkzeuge etwa entsprechende Regeln besitzen).

Quelle(n): TSS-WEB

2.1.11 Whitelisting-Ansatz

Vorgabe: Ein positives Sicherheitsmodell (Whitelisting) SOLLTE stets anstelle eines negativen (Blacklisting) verwendet werden.

Quelle(n): TSS-WEB

2.1.12 Bevorzugung von Open Source bei sicherheitskritischen Komponenten

Vorgabe: Komponenten (Hardware, Software, Bibliotheken, ...) für sicherheitskritische Systeme (Authentifizierung, Passworthashing, ...) MÜSSEN offengelegte Verfahren (Kerckhoffs'sche Prinzip/Open Source) einsetzen.

Hinweise: Durch die bei Open Source sichergestellte öffentliche Zugänglichkeit des Programmquelltexts, kann geprüft werden, dass keine Hintertüren oder für Sicherheitsprobleme anfällige Programmierungen verwendet wurden, die dadurch die Gesamtsicherheit kritischer Systeme gefährden können. Dieser Open Source Ansatz MUSS proprietären Verfahren/Komponenten vorgezogen werden.

Quelle(n): VHV-RSE

2.2 Organisatorische und regulatorische Vorgaben

Obwohl der Fokus dieser Richtlinie auf Vorgaben für die Programmierung liegt, sind für Entwicklung sicherer Anwendungen auch organisatorische und regulatorische Maßnahmen wichtig. Die wichtigsten insbesondere für Programmierer relevanten Vorgaben organisatorischer Natur sind daher nachfolgend aufgeführt.

2.2.1 Berücksichtigung von Compliance-Anforderungen für die Software-Entwicklung

Vorgabe: Bei der Entwicklung MUSS darauf geachtet werden, dass rechtliche und normative Vorgaben eingehalten und nicht verletzt werden.

Hinweise: Beispielsweise DÜRFEN NICHT urheberrechtlich geschützte Programmteile, Bibliotheken oder Grafiken unlizenziert verwendet werden. Ebenfalls DARF NICHT gegen die Lizenzbedingungen der eingesetzten Entwicklungswerkzeuge verstoßen werden. Im Zweifelsfall MUSS ein entscheidungsfähiger Verantwortlicher für Stellungnahmen zu Vorgaben identifiziert und einbezogen werden.

Quelle(n): BSI-ITGS-B5.27

2.2.2 Zugriffskonzept („Rollen und Rechte“)

Vorgabe: Es MUSS ein Konzept vorliegen bzw. erstellt werden, aus dem hervorgeht, welche Anwendungsfälle, Geschäftsprozesse, Datentypen usw. sowie Zugriffskontrollkriterien existieren, sodass Berechtigungen entsprechend provisioniert und kontrolliert werden können. Die Vorgaben der Arbeitsrichtlinie Zugangssteuerung sind hierbei zu beachten.

Hinweise: Hierbei MÜSSEN sowohl Zugriffsanforderungen für Daten- wie auch für Systemressourcen ermittelt, Rollen/Rechten zugeordnet und dies dokumentiert werden.

Quelle(n): OWASP-SCP

2.2.3 Regelung für Behandlung vertraulicher Daten

Vorgabe: Es MUSS eine Liste der vertraulichen Daten, die von der Anwendung verarbeitet werden, existieren bzw. erstellt werden. Hierbei sind die definierten Schutzbedarfe für die durch die Anwendung unterstützten Geschäftsprozesse zu beachten. Eine explizite Regelung SOLLTE existieren, die regelt, wie Zugriffe auf diese Daten kontrolliert werden und wann diese zu verschlüsseln sind (sowohl für Speicherung, als auch Übertragung). Es SOLLTE sichergestellt werden, dass diese Regelung korrekt umgesetzt ist.

Hinweise: Hierbei MUSS sich die ggf. implementierte Verschlüsselung an der Arbeitsrichtlinie Kryptographie orientieren.

Quelle(n): BSI-WEB-AN

2.2.4 Feststellen der Anwendbarkeit von Datenschutzvorgaben

Vorgabe: Es MUSS festgestellt werden, ob die Anwendung personenbezogene Daten erhebt oder verarbeitet. Falls ja, MÜSSEN die im Kapitel „Datenschutz“ enthaltenen Vorgaben berücksichtigt werden (andernfalls nicht) und es MUSS ein Datenverarbeitungs-(DV-)Konzept diesbezüglich mit dem Datenschutzbeauftragten abgestimmt werden, woraus weitere Vorgaben resultieren können.

Hinweise: Personenbezogene Daten sind alle Informationen, die sich auf eine identifizierte oder identifizierbare Person beziehen. Identifizierbar bedeutet, dass die Person direkt oder indirekt, insbesondere mittels Zuordnung zu einer Kennung wie einem Namen, zu einer Kennnummer, zu Standortdaten, zu einer Online-Kennung oder zu einem oder mehreren besonderen Merkmalen, die Ausdruck der physischen, physiologischen, genetischen, psychischen, wirtschaftlichen, kulturellen oder sozialen Identität dieser natürlichen Person sind, identifiziert werden kann.

Quelle(n): EU-DSGVO

2.2.5 Change-Management für Source Code

Vorgabe: Es MUSS ein Change-Management-System für den Code sowohl aus Entwicklungs- als auch für Produktivumgebung existieren, in welchem Changes festgehalten und von Verantwortlichen freigegeben werden.

Hinweise: Im Change-Management-System MÜSSEN Unterschiede zwischen Versionen dokumentiert werden und eine Möglichkeit zur Rückkehr zu vorherigen Versionen möglich sein.

Quelle(n): OWASP-SCP

2.2.6 Keine Veröffentlichung von Source-Code ohne Freigabe

Vorgabe: Source- oder Programmcode DARF NUR nach expliziter Freigabe auf externen Plattformen (z. B. in Internet-Foren) veröffentlicht oder externen Personenkreisen auf sonstige Weise verfügbar gemacht werden.

Quelle(n): TSS-WEB

2.2.7 Auswahl vertrauenswürdiger Entwicklungswerkzeuge

Vorgabe: Es DÜRFEN NUR Entwicklungswerkzeuge verwendet werden, die von Datenschutz & Informationssicherheit als vertrauenswürdig eingestuft und freigegeben wurden. Diese DÜRFEN NUR aus ebenfalls als vertrauenswürdig eingestuften und freigegebenen Quellen bezogen werden.

Quelle(n): BSI-ITGS-B5.27

2.2.8 Vier-Augen-Prinzip für sicherheitskritische Komponenten

Vorgabe: Sicherheitskritische Anwendungskomponenten (z. B. Authentisierungssysteme) MÜSSEN während der Entwicklung von mehreren Entwicklern geprüft und getestet werden.

Quelle(n): BSI-ITGS-B5.27

2.2.9 Überprüfung von Third-Party-Code

Vorgabe: Verwendete externe Bibliotheken und Code aus Drittquellen MÜSSEN auf Schwachstellen und mögliche Konflikte mit anderen bereits verwendeten Komponenten überprüft werden.

Hinweise: Eine Recherche zu vorhandenen bekannten Schwachstellen KANN z. B. über cvedetails.com durchgeführt werden. Komponenten mit bekannten Schwachstellen und einem CVSS-Score größer oder gleich 7.0 DÜRFEN NICHT ohne Patch oder Workaround verwendet werden.

Quelle(n): BSI-ITGS-B5.27

2.2.10 Durchführung von Code-Reviews

Vorgabe: Der Anwendungsverantwortliche SOLLTE regelmäßig unabhängige Code-Reviews durchführen lassen, um die Integrität der Software sicherzustellen.

Hinweise: Die Code-Reviews müssen von unabhängigen Dritten oder zumindest von Entwicklern, die den Code nicht selbst geschrieben haben, durchgeführt werden. Hierbei ist zu prüfen, ob der Programmcode alle gewünschten Funktionen enthält und gleichzeitig keinerlei zusätzliche oder ungewollte Funktionen beinhaltet.

Quelle(n): BSI-ITGS-B5.27

2.2.11 Trennung von Entwicklungs- und Produktivumgebung

Vorgabe: Die Entwicklungsumgebung(en) MÜSSEN von der produktiven Umgebung isoliert sein. Zugang zu Netzwerksegmenten für die Entwicklung DARF NUR autorisierten Entwicklungs- und Testgruppen möglich sein. Auf dem Entwicklungssystem dürfen keine produktiven (Echtdaten) genutzt werden.

Hinweise: Ist eine Netzwerksegmentierung nicht möglich oder explizit nicht gewollt, so MUSS die beabsichtigte Trennung durch Rollen und Rechte für die jeweiligen Systeme / Anwendungen umgesetzt werden.

Quelle(n): OWASP-SCP

2.2.12 Zugriffsrechtebeschränkungen in Entwicklungsumgebungen

Vorgabe: Der Zugriff auf Entwicklungsdaten (Code-Repositories usw.) MUSS auf berechtigte Personengruppen beschränkt werden (Need-to-Know-Prinzip), damit diese nicht unautorisiert eingesehen oder manipuliert werden können. Dies schließt die Authentifizierung und Autorisierung entsprechender Zugriffe in Code Repositories (z. B. SVN, Git) Build-Systemen (z. B. Jenkins), Wikis und auf dem Dateisystem ein. Zugriffe auf Entwicklungsdaten MÜSSEN einzelnen Benutzern zugeordnet und dokumentiert werden können.

Quelle(n): BSI-ITGS-B5.27; TSS-WEB

2.2.13 Sichere Verbindung verteilter Arbeitsplätze

Vorgabe: Die Anbindung verteilter Arbeitsplätze an die Entwicklungsumgebung MUSS verschlüsselt erfolgen.

Hinweise: Hierfür KANN z. B. eine sichere VPN-Lösung wie OpenVPN mit Authentifizierung verwendet werden.

Quelle(n): BSI-ITGS-B5.27

2.2.14 Übermittlung von Source-Code nur über verschlüsselte Kanäle

Vorgabe: Der Austausch von Source- oder Programmcode mit Dritten DARF NUR über sichere Kanäle (z. B. S/MIME oder per TLS) erfolgen.
Hinweise: Hierbei MUSS der Sender verifiziert werden können (z. B. durch Verwendung von Signaturen bei PGP/GPG).
Quelle(n): TSS-WEB

2.2.15 Verwendung nur freigegebener 3rd-Party-Dependency Repositories

Vorgabe: Werden externe Dependency Repositories (z. B. Maven Repositories) angebunden, MUSS sichergestellt werden, dass diese vertrauenswürdig sind. Hierfür ist ein entsprechender Freigabeprozess festzulegen.
Quelle(n): TSS-WEB

2.2.16 Sichere Installation der entwickelten Software

Vorgabe: Für den Installationsprozess MUSS ein Installationsplan existieren bzw. erstellt werden, der alle durchzuführenden Schritte detailliert beschreibt und dabei auch mögliche Fehlerquellen oder Abweichungen zwischen verschiedenen Zielsystemen berücksichtigt. Für nach der Installation durchzuführende Tests, anhand dieser die korrekte Installation überprüft und das System für den Betrieb freigegeben werden SOLL, SOLLTE ein Testplan existieren bzw. erstellt werden.
Quelle(n): BSI-ITGS-B5.27

2.3 Übergreifende Programmiervorgaben

Die folgenden Vorgaben enthalten konkrete themenübergreifende Anforderungen, die bei der Programmierung zu beachten sind.

2.3.1 Anwendungsfluss entsprechend Geschäftslogik

Vorgabe: Es MUSS sichergestellt werden, dass die Anwendungsabläufe entsprechend und in Reihenfolge der definierten Geschäftslogik bzw. Prozesse erfolgen.
Hinweise: Insb. DARF NICHT clientseitig Einfluss auf die Reihenfolge des Ablaufs nachfolgender Prozessschritte genommen werden können. Nacheinander aufzurufende Funktionen MÜSSEN also überprüfen, ob die jeweils vorangegangenen Schritte erfolgt sind, wenn die Funktionen einzeln vom Client ausgelöst werden.
Quelle(n): OWASP-SCP

2.3.2 Funktionstrennung für privilegierte Anwendungslogik

Vorgabe: Besonders sensible Anwendungslogik, wie administrative Funktionen, MUSS von der übrigen Anwendungslogik getrennt sein.
Hinweise: Dies KANN beispielsweise in MVC-Anwendungen in separaten Controllern und separaten Dateien hierfür umgesetzt werden.
Quelle(n): OWASP-SCP

2.3.3 Verwendung unsicherer Funktionen vermeiden

Vorgabe: Die Verwendung von Funktionen, die als unsicher gelten (wie „sprintf“, „strcat“, „strcpy“ usw.), SOLLTE möglichst vermieden werden.
Hinweise: Als Ersatz eignen sich „snprintf“, „strncat“ und „strncpy“.
Quelle(n): OWASP-SCP

2.3.4 Explizite Initialisierung von Variablen

Vorgabe: Variablen und sonstige Datencontainer SOLLTEN explizit initialisiert werden, entweder im Rahmen der Deklaration oder unmittelbar vor erstmaliger Verwendung.
Hinweise: Für Java oder C SOLLTE entsprechend „int i = 0;“ anstelle von „int i;“ bei der Initialisierung verwendet werden.
Quelle(n): OWASP-SCP

2.3.5 Angemessene Kommentierung von Source Code

Vorgabe: Erstellter Code MUSS angemessen kommentiert sein. Dies bedeutet, dass Sourcecode für fachkundige Benutzer in angemessener Zeit verständlich sein MUSS.
Hinweise: Es SOLLTEN Kommentare entsprechend eines Standards genutzt werden (z. B. JavaDoc), aus dem automatisch eine Source Code-Dokumentation generiert werden kann.
Quelle(n): BSI-ITGS-B5.27

2.3.6 Vermeidung von Berechnungsfehlern

Vorgabe: Berechnungsfehler SOLLTEN weitmöglichst vermieden werden, indem die der verwendeten Programmiersprache zugrundeliegenden Datentypen und -repräsentationen sowie die Wirkungsweise darauf angewendeter (numerischer) Rechenoperationen verstanden werden.
Hinweise: Erhöhte Aufmerksamkeit SOLLTE auf Byte-Größenunterschiede von Datentypen, Präzision von Nachkommastellen, Unterschiede von signed / unsigned-Variablen, Rundung / Kürzung, Datenkonvertierung und Type Casting, „NaN“ (Not a Number)-Berechnungen und der Behandlung von Zahlen die zu groß oder klein für die vorgesehene Repräsentation sind, gelegt werden.
Quelle(n): OWASP-SCP

2.3.7 Vermeidung von Verwundbarkeiten durch weitere Anwendungskomponenten

Vorgabe: Für sämtliche genutzten sekundären Anwendungen, Code aus Drittquellen oder Bibliotheken MUSS eine Überprüfung hinsichtlich Notwendigkeit der Verwendung und sicherer Funktionalität erfolgen.
Hinweise: Hierbei MÜSSEN mindestens bekannte Sicherheitslücken (CVEs) für eingebundene sekundäre Anwendungen und Bibliotheken berücksichtigt werden.
Quelle(n): OWASP-SCP

2.3.8 Behandlung einzubindender Ressourcen

Vorgabe: Alle eingebundenen Ressourcen MÜSSEN definiert und in einer eigenen Liste bzw. einem eigenen Verzeichnis abgelegt werden.
Hinweise: Bspw. KANN definiert werden, dass eine Anwendung nur Dateiressourcen von bestimmten, aufgeführten Dateitypen einbinden soll. Hierbei SOLL sichergestellt werden, dass die entsprechenden Dateien in einem definierten, von sonstigen Programmdateien getrennten Verzeichnis abgelegt werden.
Quelle(n): BSI-WEB-AN

2.3.9 Zugriffsbeschränkung bei Einbindung externer und interner Ressourcen

Vorgabe: Der Zugriff MUSS auf Ressourcen beschränkt werden, die für die Funktion der Applikation notwendig sind.
Hinweise: Es DÜRFEN NICHT bspw. Ressourcen mit Dateitypen, deren Einbindung oder Verarbeitung nicht vorgesehen ist, eingebunden werden können.
Quelle(n): BSI-WEB-AN

2.3.10 Minimalprinzip für Verwendung externer Ressourcen

Vorgabe: Externe Ressourcen DÜRFEN NUR mit den minimalen (d. h. den notwendigen) Rechten verwendet werden.
Hinweise: Bspw. DÜRFEN NICHT Ausführungsrechte auf einzubindende Bilddateien gesetzt werden, wenn diese lediglich dargestellt (gelesen) werden sollen.
Quelle(n): BSI-WEB-AN

2.3.11 Sichere Ablage eingebundener Ressourcen

Vorgabe: Die einzubindenden Ressourcen MÜSSEN so abgelegt werden, dass sie nicht unbefugt gelesen oder verändert werden können.

Hinweise: Dies SOLLTE bspw. sowohl auf Dateisystemebene für Anwender mit Schreib-/Lesezugriff hierauf gelten, wie auch für die Rechte der Anwendung den Ressourcen gegenüber. D. h. auch die Anwendung SOLLTE nicht über weitreichendere als benötigte Rechte für die Ressourcen/Dateien verfügen.

Quelle(n): BSI-WEB-AN

2.3.12 Code-Generierung oder -Modifizierung einschränken

Vorgabe: Es MUSS sichergestellt werden, dass Anwender nicht neuen Anwendungscode generieren oder bestehenden modifizieren können.

Quelle(n): OWASP-SCP

2.3.13 Kapselung von Betriebssystemfunktionalität

Vorgabe: Für Verwendung benötigter Betriebssystemfunktionalität SOLLTE eine anwendungsfallspezifische API verwendet werden. Befehle DÜRFEN NICHT direkt auf Betriebssystemebene ausgeführt werden, insbesondere nicht auf Basis von nicht vertrauenswürdigen (Eingabe)daten oder von der Anwendung ausgehenden Command-Shells.

Hinweise: Es KANN z. B. eine Klasse bereitgestellt werden, welche die benötigte Betriebssystemfunktionalität durch parametrisierbare Methoden bereitstellt.

Quelle(n): OWASP-SCP

2.3.14 Verwendung von Prüfsummen für sensible Dateien

Vorgabe: Soweit möglich und sinnvoll, SOLLTEN Prüfsummen bzw. Hashes verwendet werden, um die Integrität von zu interpretierenden Code-Dateien, Bibliotheken, ausführbaren Dateien und Konfigurationsdateien zu verifizieren.

Hinweise: Kryptographisch sichere Hashfunktionen SOLLTEN anstelle von Prüfsummen und eigenen Hashfunktionen verwendet werden – hierbei SOLLTE die Arbeitsrichtlinie Kryptographie berücksichtigt werden.

Quelle(n): OWASP-SCP

2.3.15 Limitierung von Transaktionen

Vorgabe: Je Benutzer oder Endgerät SOLLTE die Anzahl der durchführbaren Transaktionen innerhalb eines bestimmten Zeitraums limitiert werden.

Hinweise: Zeitraum und erlaubte Transaktionsanzahl je Typ SOLLTE konfigurierbar sein. Durchführbare Transaktionen SOLLTEN über den tatsächlichen Geschäftsanforderungen liegen, jedoch niedrig genug gewählt sein, um automatisierte Angriffe einzuschränken. Bei Überschreitung des Limits KANN eine sog. „Tar Pit“ eingesetzt werden, die die Zugriffszeiten bei folgenden Zugriffen zeitlich begrenzt sukzessive verzögert.

Quelle(n): OWASP-SCP

2.3.16 Definition der genutzten HTTP-Methoden

Vorgabe: Es MUSS spezifiziert werden, welche HTTP-Methoden die Anwendung unterstützt. Weiterhin SOLLTE spezifiziert und dokumentiert werden, ob / wie diese Anfragemethoden ggf. von unterschiedlichen Seiten der Anwendung unterschiedlich behandelt werden.

Hinweise: In der Regel SOLLTE nur eine Verwendung von GET, POST und ggf. HEAD vorgesehen und zulässig sein.

Quelle(n): OWASP-SCP

2.3.17 Sichere Update-Mechanismen

Vorgabe: Es MÜSSEN sichere Update-Mechanismen für die Anwendung definiert und implementiert sein.

Hinweise: Bei Verwendung automatischer Updates MÜSSEN kryptographische Signaturen genutzt und überprüft werden. Die Übertragung von Code DARF NUR über sicher verschlüsselte Kanäle erfolgen.

Quelle(n): OWASP-SCP

2.4 Eingabevalidierung

Validierung von Eingabedaten und speziellen Benutzereingaben ist ein essenzieller Punkt bei der Erstellung sicherer Software. Viele, insbesondere auch angreifbare, Sicherheitslücken entstehen durch eine unzureichende Eingabedatenvalidierung. Die wichtigste Faustregel zur Eingabevalidierung lautet: Alle Daten aus externen Quellen (dazu gehören sämtliche Benutzereingaben und Metadaten von Clients, wie HTTP-Header) sind als möglicherweise bösartig einzustufen und müssen daher auf Kriterien für sichere Verwendung im jeweiligen Anwendungskontext validiert werden.

2.4.1 Validierung nicht vertrauenswürdiger Daten

Vorgabe: Alle Datenquellen MÜSSEN identifiziert und SOLLTEN in „vertrauenswürdig“ und „nicht vertrauenswürdig“ klassifiziert werden. Datenquellen die als „nicht vertrauenswürdig“ eingestuft wurden MÜSSEN validiert werden.
Hinweise: Nicht vertrauenswürdige Quellen aus Sicht der Anwendung können nicht nur Formulardaten oder Parameter, sondern auch Datenbank- oder Dateiinhalte sein.
Quelle(n): OWASP-SCP; TSS-WEB

2.4.2 Eingabeprüfung sämtlicher Parameter

Vorgabe: Eingabeprüfungen MÜSSEN auf sämtliche Parameter angewendet werden. Dies schließt neben Benutzerparametern auch Anwendungsparameter (z. B. mittels Hidden Form Fields oder Cookies) mit ein.
Quelle(n): TSS-WEB

2.4.3 Eingabedatenvalidierung in vertrauenswürdigen Systemen

Vorgabe: Es MUSS sichergestellt werden, dass Eingabedatenvalidierung in einem kontrollierbaren, vertrauenswürdigen Kontext erfolgt (i. d. R. serverseitig).
Hinweise: Eine clientseitige Eingabedatenvalidierung ist als Usability und nicht als Sicherheitsfunktion zu verstehen und nur als zusätzliche Maßnahme zulässig.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.4.4 Zentrale Eingabedatenvalidierung

Vorgabe: Soweit möglich, SOLLTE die Datenvalidierung zentral erfolgen, z. B. in einer globalen Validierungsfunktion.
Hinweise: Beispielsweise KANN für Java-Anwendungen Java Bean Validation verwendet werden.
Quelle(n): OWASP-SCP

2.4.5 Mindestanforderungen an die Eingabedatenvalidierung

Vorgabe: Die Eingabedatenvalidierung MUSS mindestens, soweit zutreffend, den erwarteten Datentyp, den Wertebereich, die Datenlänge, sowie die verwendeten Zeichen validieren. Die Einschränkungen für erlaubte Daten MÜSSEN möglichst restriktiv gewählt sein.

Hinweise: Soweit möglich, SOLLTE eine Validierung von verwendeten Zeichen aus Eingabedaten gegen eine Whitelist erfolgen.

Quelle(n): OWASP-SCP; TSS-WEB; BSI-WEB-AN

2.4.6 Datenvalidierung: Verwendung eines Whitelisting-Ansatzes

Vorgabe: Soweit möglich, MUSS für Validierung von Daten ein Whitelisting-Ansatz verwendet werden, also gegen Muster für gültige (nicht: ungültige) Daten geprüft werden.

Quelle(n): BSI-WEB-MBP

2.4.7 Zurückweisung von nicht validierbaren Eingabedaten

Vorgabe: Eingabedaten, die die durchgeführte Eingabedatenvalidierung nicht bestehen, MÜSSEN zurückgewiesen werden bzw. DÜRFEN NICHT weiterverarbeitet werden.

Hinweise: Sollte eine anwendungsseitige Korrektur von nicht validierbaren Eingabedaten verwendet werden, so MÜSSEN die resultierenden Daten erneut validiert werden.

Quelle(n): OWASP-SCP

2.4.8 Spezifizierung des Character-Sets

Vorgabe: Das verwendete Character-Set für Eingabedatenquellen MUSS jeweils spezifiziert werden (z. B. UTF-8).

Hinweise: Dies KANN z. B. in C# via Parametrisierung von „StreamReader“-Objekten mit „System.Text.Encoding.UTF8“ geschehen, in PHP über die Funktion „mb_internal_encoding“, in Java über Parametrisierung beim Programmaufruf mit „-Dfile.encoding=UTF-8“.

Quelle(n): OWASP-SCP; BSI-WEB-AN

2.4.9 (Re)kodierung in verwendetes Character-Set

Vorgabe: Eingabedaten aus nicht vertrauenswürdigen Quellen MÜSSEN vor Verwendung und vor Validierung in ein verwendetes Standard-Character-Set (z. B. UTF-8) überführt bzw. rekodiert werden.

Hinweise: Eine Eingabedatenüberprüfung, die ein Character-Set voraussetzt ohne dieses zu forcieren, kann gegenüber sogenannten Unicode Encoding Attacks verwundbar sein.

Quelle(n): OWASP-SCP

2.4.10 (Re)kodierung von Sonderzeichen

Vorgabe: Wenn die Anwendung (potenziell gefährliche) Sonderzeichen in Eingabedaten zulässt, so MÜSSEN diese vor Verarbeitung und vor Ausgabe entsprechend des jeweiligen Kontexts sicher (re)kodiert werden.
Hinweise: Hierbei SOLLTEN Standardfunktionen Verwendung finden.
Quelle(n): OWASP-SCP

2.4.11 Bereinigung von formatierten Eingabedaten

Vorgabe: Eingabedaten, die in bestimmten Datenformaten erwartet werden (z. B. JSON, XML, HTML), MÜSSEN über eine ausgereifte Sanitizer- oder Parser-API restriktiv verifiziert und bereinigt werden.
Hinweise: Insbesondere DARF NICHT ein komplexes Datenformat mithilfe ausschließlich unzureichender Mechanismen (z. B. mit regulären Ausdrücken) validiert oder bereinigt werden.
Für HTML-Eingabedaten SOLLTE nach Möglichkeit ein Zwischenformat wie beispielsweise BBCode verwendet werden.
Quelle(n): TSS-WEB; BSI-WEB-MBP

2.4.12 Keine Ausführung nicht vertrauenswürdiger Daten

Vorgabe: Nicht vertrauenswürdige Daten, insbesondere anwenderspezifizierte Eingabedaten, DÜRFEN NICHT ungeprüft an Funktionen weitergegeben werden, die diese (dynamisch) ausführen.
Hinweise: Datenbankzugriffe zählen hierbei insbesondere als Ausführung (siehe SQL Injection Angriffe).
Quelle(n): OWASP-SCP

2.4.13 Berücksichtigung aller benutzerdefinierten Eingabedatenquellen bei Webanwendungen

Vorgabe: Webanwendungen MÜSSEN sämtliche verwendeten und vom Benutzer manipulierbaren Daten bei der Eingabedatenvalidierung berücksichtigen. Hierzu gehören GET- und POST-Parameter, URLs / Pfade, Dateinamen und MIME-Types, HTTP-Header (u. a. Cookies usw.).
Hinweise: Auch üblicherweise automatisiert ausgelöste Anfragen, z. B. aus JavaScript-, Flash- oder sonstigem Embedded-Kontext, sowie Daten aus Redirects, können von Angreifern manipuliert werden und MÜSSEN hierbei berücksichtigt werden.
Quelle(n): OWASP-SCP

2.4.14 Validierung des Header-Datenformats in Webanwendungen

Vorgabe: Webanwendungen SOLLTEN verifizieren, dass HTTP-Header in Anfragen und Antworten lediglich aus ASCII-Zeichen bestehen.

Hinweise: Wird eine eigene Funktion implementiert, SOLLTE diese überprüfen, ob alle Zeichen einen numerischen Wert <128 aufweisen. Für Python-Anwendungen kann überprüft werden, ob s.decode('ascii') auf den Header-String einen Fehler wirft.

Quelle(n): OWASP-SCP; BSI-WEB-AN

2.4.15 Normalisierung von Pfadangaben

Vorgabe: Dateipfade MÜSSEN stets normalisiert und kanonisiert werden, bevor gegen diese eine Validierung durchgeführt wird. Hierbei MÜSSEN Pfadtraversierungen („../../“) bereinigt bzw. aufgelöst werden.

Hinweise: Insbesondere MUSS hierbei auf die Unterschiede zwischen Unix- und Windows-Pfaden geachtet werden.

Quelle(n): TSS-WEB

2.4.16 Sichere IDs für Resourcenreferenzierung

Vorgabe: Es DARF KEINE Ressource eingebunden werden, deren Identifikator durch ungefilterte Benutzereingaben definiert wird.

Quelle(n): BSI-WEB-AN

2.4.17 Validierung von HTTP-Variablen

Vorgabe: Bei HTTP-Eingabedaten SOLLTE zwischen Request-Variablen je nach HTTP Verb (i. d. R. GET oder POST) unterschieden werden und nur diejenigen herangezogen werden, die für die jeweilige Anfrage erwartet werden (z. B. Berücksichtigung von POST-Variablen bei Verarbeitung einer Eingabe via Formular). Namen von Request-Variablen, die ggf. nicht vor Eingang bekannt sind, MÜSSEN validiert werden – Variablen mit ungültigem Namensschema MÜSSEN abgelehnt werden.

Hinweise: Der explizite Zugriff auf HTTP-Variablen zu einem bestimmten Verb (hier: POST) KANN z. B. in Java (Spring MVC) via Parametrisierung des „@RequestMapping“ mit „method = RequestMethod.POST“ erfolgen, in ASP.Net über Zugriff via „Request.Form“, in PHP via „$_POST“. Ein gültiges Variablennamenschema KANN mit regulären Ausdrücken wie „^[a-z_]\\w*$“ erfolgen.

Quelle(n): BSI-WEB-MBP

2.4.18 Datenvalidierung: Behandlung von ungültigen Werten

Vorgabe: Fehlerhafte Eingaben MÜSSEN zurückgewiesen werden. Es DARF KEINE Weiterverarbeitung stattfinden. Es DÜRFEN KEINE fehlerhaften Eingaben korrigiert werden.

Hinweise: Fehlermeldungen bei Zurückweisung ungültiger Daten DÜRFEN KEINE vertraulichen Informationen (z. B. Systemdaten wie Konfigurationseinstellungen, Systempfade und Programmcode, personenbezogene Daten Dritter oder interne Prozesslogik) beinhalten.

Quelle(n): BSI-WEB-MBP

2.4.19 Enkodierung von Kommandozeilenparametern

Vorgabe: Aufgerufene Kommandozeilenbefehle, welche Parameter aus nicht vertrauenswürdigen Quellen enthalten, DÜRFEN NUR nach sicherer Enkodierung verwendet werden.

Quelle(n): BSI-WEB-AN

2.5 Ausgabevalidierung (Enkodierung & Escaping)

Auch bei der Ausgabe von Daten muss sichergestellt sein, dass diese im jeweiligen Kontext wie gewünscht interpretiert werden. Andernfalls können nicht nur Fehler auftreten, sondern auch Sicherheitslücken entstehen.

2.5.1 Kodierung von Ausgabedaten in vertrauenswürdigen Systemen

Vorgabe: Es MUSS sichergestellt werden, dass die Ausgabedatenkodierung in einem kontrollierbaren, vertrauenswürdigen Kontext erfolgt (i. d. R. serverseitig).

Hinweise: Für Java KANN hierfür die Methode escapeHtml der StringEscapeUtils aus Apache Commons Lang verwendet werden. Insbesondere SOLLTEN zusätzlich die aus Drittquellen für Ausgabe genutzten Daten auf tatsächliche Kodierung im erwarteten Format hin überprüft werden.

Quelle(n): OWASP-SCP

2.5.2 Zentrale Ausgabedatenvalidierung

Vorgabe: Für jeden akzeptierten Ein- und Ausgabedatentyp SOLLTE ein zentraler Mechanismus verwendet werden, der die Daten gemäß ihrem Verwendungszweck validiert.

Quelle(n): BSI-WEB-AN

2.5.3 Kodierung von Ausgabedaten über standardisierte Funktionen

Vorgabe: Für die Kodierung von Ausgabedaten SOLLTEN getestete Standardfunktionen der Programmiersprache oder des verwendeten Frameworks oder ausgereifte APIs bzw. Bibliotheken verwendet werden.

Hinweise: Beispiele für Standardkodierungsfunktionen sind: „StringEscapeUtilities.escapeXml“ aus Apache, „Commons“ für XML-Ausgabekontext in Java oder „mysqli_real_escape_string“ für MySQL in PHP.

Quelle(n): OWASP-SCP; TSS-WEB

2.5.4 Berücksichtigung des Kontexts bei Kodierung von Ausgabedaten

Vorgabe: Bei der Kodierung von Ausgabedaten MUSS die Kodierung entsprechend des vorgesehenen Zielkontexts erfolgen (z. B. JavaScript, JSON, XML, HTML, SQL, LDAP, Shell, ...).

Hinweise: Hierbei MUSS sichergestellt sein, dass Ausgabedaten nicht aus dem vorgesehenen Kontext ausbrechen können, was bei nicht / falsch kodierten Steuerzeichen des jeweiligen Kontexts der Fall sein kann.

Quelle(n): OWASP-SCP; TSS-WEB; BSI-WEB-AN

2.5.5 Kodierung aller möglicherweise unsicheren Zeichen

Vorgabe: Sämtliche bei der Ausgabe verwendeten Zeichen, die nicht im jeweiligen Kontext oder für den jeweiligen Interpreter als sicher bekannt sind, MÜSSEN sicher kodiert werden.

Hinweise: Ein Beispiel für Encoding ggf. unsicherer Zeichen ist HTML Entity Encoding, das jedoch nicht in jedem Kontext anwendbar ist.

Quelle(n): OWASP-SCP; TSS-WEB

2.5.6 Implizite Ausgabevalidierung

Vorgabe: Für die Ausgabevalidierung SOLLTE stets eine implizite Validierung durchgeführt werden. Im Frontend wird eine solche etwa durch viele Webframeworks (bzw. Template-Technologien) zur Verfügung gestellt, im Backend z. B. durch den Einsatz eines ORM-Frameworks.

Hinweise: Ein Beispiel für implizite Ausgabevalidierung ist die Verwendung von Zope Page Templates.

Quelle(n): TSS-WEB

2.5.7 Definierter Content-Type von HTTP-Antworten

Vorgabe: Jede HTTP-Antwort MUSS einen Content-Type enthalten, der einen sicheren Zeichensatz festlegt (z. B. UTF-8).

Hinweise: Hierfür MUSS z. B. ein Header „Content-Type: text/html; charset=utf-8“ für HTML-Seiten gesetzt werden. Bei Ausgabe in HTML, XML oder andere Datenformate, die dies unterstützten, SOLLTE zusätzlich der Content-Type in den Ausgabedaten selbst festgelegt werden (z. B. via „<meta charset="UTF-8" />“.

Quelle(n): BSI-WEB-AN

2.5.8 Sichere HTTP-Weiterleitungen

Vorgabe: Insbesondere auch bei HTTP-Weiterleitungen DÜRFEN NICHT unvalidierte Daten für die Weiterleitung (z. B. als Teil der URL) verwendet werden. Besonders beachtet werden MUSS hierbei, dass die Parametrisierung von HTTP-Weiterleitungen nicht zur Injektion oder Manipulation des Antwort-HTTP-Headers führen darf.

Hinweise: Noch besser SOLLTE Parametrisierung von HTTP-Weiterleitungen nach Möglichkeit nicht mit einer (Teil)adresse, sondern durch Übergabe eines Indexwerts erfolgen, zu dem die Ziel-URI hinterlegt ist.

Quelle(n): BSI-WEB-AN; BSI-WEB-MBP

2.6 Authentifizierung & Registrierung von Benutzern

Dieses Kapitel enthält Vorgaben zu den Prozessen Benutzerregistrierung und -Account-Provisionierung sowie Identifizierung, Authentisierung und Authentifizierung von Benutzern gegenüber einer Anwendung, die zentral bei der Entwicklung von Anwendungen zu berücksichtigen sind. Die Umsetzung der nachfolgenden Vorgaben dient der Vermeidung logischer Fehler und einem hohen Sicherheitsniveau der Anwendung.

2.6.1 Authentifizierung by Default

Vorgabe: Für den Zugriff auf alle Ressourcen bzw. Seiten und Funktionen, sofern diese nicht explizit als öffentlich verfügbar klassifiziert wurden, MUSS vorab eine Authentifizierung erfolgen.

Hinweise: Bei der Authentifizierung MUSS sichergestellt sein, dass jeder Benutzer mithilfe eindeutiger Sitzungsidentifikatoren, beispielsweise ein eindeutiger und zufällig generierter Sitzungsschlüssel, der über einen vertraulichen Kanal transferiert wird, identifiziert werden kann.

Quelle(n): OWASP-SCP; BSI-WEB-AN

2.6.2 Authentifizierung durch vertrauenswürdige Systeme

Vorgabe: Es MUSS sichergestellt werden, dass sämtliche Authentifizierungsmechanismen in einem kontrollierbaren, vertrauenswürdigen Kontext erfolgen (i. d. R. serverseitig).

Hinweise: Ein häufiger Fehler ist hierbei z. B. die Übernahme eines „isAuthenticated“ o. ä.-Flags vom Client. Stattdessen DÜRFEN NUR die für die Authentifizierung tatsächlich benötigten Merkmale validiert vom Client herangezogen werden, um damit eine sichere (serverseitige) Validierung durchzuführen.

Quelle(n): OWASP-SCP; TSS-WEB

2.6.3 Verwendung von standardisierten Authentifizierungsfunktionen

Vorgabe: Für die Authentifizierung SOLLTEN getestete Standardfunktionen der Programmiersprache und / oder des verwendeten Frameworks verwendet werden.

Hinweise: Bei dem Prüfen von Passwörtern in z. B. PHP sollten die Funktion „password_verify“ genutzt werden. Zum Hashen von Passwörtern „password_hash“.

Quelle(n): OWASP-SCP

2.6.4 Zentrale Authentifizierungsimplementierung

Vorgabe: Für alle Authentifizierungsmechanismen, auch jene die hierfür externe Services anbinden, SOLLTE eine zentrale Implementierung verwendet werden.

Hinweise: Dies entspricht dem DRY (don’t repeat yourself)-Konzept – Authentifizierungscode SOLLTE zentral implementiert sein, statt an sämtlichen benötigten Codestellen ggf. wiederholt zu werden.

Quelle(n): OWASP-SCP; BSI-WEB-AN

2.6.5 Logische Trennung des Authentifizierungsmechanismus

Vorgabe: Die Authentifizierungslogik SOLLTE von den angefragten Ressourcen getrennt werden und eine Weiterleitung zu und von der zentralen Implementierung erfolgen.

Hinweise: Bei einer Weiterleitung MUSS sichergestellt sein, dass die Information über eine vorhandene, erfolgreiche Authentifizierung nicht vom Client eingeschoben bzw. gefälscht werden kann (Negativbeispiel: „isAuthenticated“-Flag via HTTP-Parameter oder Cookie).

Quelle(n): OWASP-SCP

2.6.6 Sichere Fehlerbehandlung bei der Authentifizierung

Vorgabe: Bei Auftreten von Fehlern in der Authentifizierung MUSS sichergestellt werden, dass die Authentifizierung standardmäßig fehlschlägt, keine vertraulichen Informationen ausgegeben werden und ein Abfangen des Fehlers in einen sicheren Zustand führt.

Hinweise: Fehlermeldungen DÜRFEN NICHT Informationen darüber enthalten, ob der Benutzername nicht bekannt war oder das Passwort ungültig ist. Webanwendungen DÜRFEN NICHT unterschiedlichen Fehlercodes abhängig von dem Fehler in der Authentifizierung zurückgeben – bspw. MUSS für einen falsch eingegebenen Benutzernamen sowie für ein falsch eingegebenes Passwort dieselbe Fehlermeldung ausgegeben werden (z. B. „Benutzername oder Passwort falsch“).

Quelle(n): OWASP-SCP

2.6.7 Kein Hinweis auf falsche Authentifizierungsdatenteile

Vorgabe: Fehlermeldungen bei Angabe unzulässiger Authentifizierungsangaben DÜRFEN NICHT darauf hinweisen, welche angegebene Information falsch ist.

Hinweise: Im Falle eines falschen / ungültigen angegebenen Benutzernamens MUSS dementsprechend dieselbe Fehlermeldung, wie für ein falsches eingegebenes Passwort ausgegeben werden (z. B. „Benutzername oder Passwort falsch“).

Quelle(n): OWASP-SCP; TSS-WEB

2.6.8 Überprüfung aller für Authentifizierung benötigten Daten

Vorgabe: Es MUSS sichergestellt werden, dass eine angefragte Authentifizierung nur nach Angabe aller benötigten Daten durchgeführt werden kann und andernfalls zurückgewiesen wird.

Hinweise: Dies ist besonders relevant bei Verwendung zusätzlicher Mechanismen oder sequentieller Authentifizierungsverfahren (z. B. mit CAPTCHA, PIN oder SMS-Token).

Quelle(n): OWASP-SCP

2.6.9 Authentifizierung für sensible Drittsysteme

Vorgabe: Bei Nutzung von sensiblen Funktionen, wie administrativen Funktionen oder Daten aus Drittsystemen MUSS vor Zugriff jeweils eine Authentifizierung erfolgen und eine angemessene Autorisierung überprüft werden.

Hinweise: Die Überprüfung der Autorisierung beim Zugriff auf Daten erfolgt im Idealfall zentralisiert über das Sitzungsmanagement der Anwendung.

Quelle(n): OWASP-SCP

2.6.10 HTTP-Authentifizierung via POST

Vorgabe: Für Authentifizierung via HTTP DARF NUR die HTTP-POST-Methode verwendet werden.

Hinweise: Dies stellt (neben möglichen anderen Mechanismen) sicher, dass Authentifizierungsdaten nicht via HTTP-Referer geleaked oder via Access Logs Dritten zugänglich gemacht werden können – deshalb DARF NICHT HTTP GET für Authentifizierung verwendet werden.

Quelle(n): OWASP-SCP

2.6.11 Brute Force-Gegenmaßnahmen

Vorgabe: Es SOLLTEN geeignete Mechanismen implementiert werden, mit denen Brute Force-Angriffe auf die Authentisierungsmechanismen verhindert werden können.

Hinweise: Dies KANN beispielsweise mit CAPTCHAs realisiert werden oder über eine sog. „Teergrube“ (engl. „Tarpit“): Hierbei wird die Anzahl der möglichen Authentisierungsversuche in einem Zeitraum begrenzt und mit fehlgeschlagenen Versuchen fortlaufend reduziert. Es MUSS hierbei sichergestellt werden, dass die Anmeldung für administrative Konten zumindest durch einen definierten sicheren Zugriffsweg (z. B. Firmen-IP-Block) nicht eingeschränkt wird, um einen DoS für administrative Konten zu verhindern.

Quelle(n): BSI-WEB-AN; OWASP-SCP; TSS-WEB

2.6.12 Keine „Anmeldedaten speichern“-Funktionalität

Vorgabe: Die Speicherung von Anmeldedaten, insbesondere Passwörtern, durch Anwendung oder den verwendeten Client SOLLTE unterbunden werden.

Hinweise: Für Webanwendungen KANN dies über Setzen des Attributs „autocomplete“ auf den Wert „off“ und für Eingabefelder („input“) vom Typ („type“) „password“ realisiert werden.

Quelle(n): OWASP-SCP

2.6.13 Benachrichtigung über letzten Anmeldeversuch

Vorgabe: Anwender SOLLTEN nach erfolgreicher Anmeldung über den letzten (erfolgreichen oder fehlgeschlagenen) Anmeldeversuch informiert werden.

Hinweise: Idealerweise SOLLTEN Anwender ihre Anmelde- und Anmeldeversuchshistorie einsehen und überprüfen können. Dies MUSS jedoch bei Implementierung mit der / dem Datenschutzbeauftragten abgestimmt werden.

Quelle(n): OWASP-SCP

2.6.14 Zwei-Faktor-Authentifizierung bei Registrierung für Anwendungen mit hohem Schutzbedarf

Vorgabe: Im Rahmen einer Benutzerregistrierung SOLLTE ein zusätzlicher Faktor (z. B. Handy) zum Einsatz kommen, wenn für die Anwendung ein hoher Schutzbedarf bezüglich Vertraulichkeit und/oder Authentizität festgestellt wurde.

Hinweise: Ein zusätzlicher Faktor, neben einem Passwort, KANN z. B. ein FIDO2-Token oder eine SMS-TAN sein.

Quelle(n): TSS-WEB

2.6.15 Vollständige Benutzerregistrierung vor Anmeldung

Vorgabe: Die Identifikation eines Benutzers MUSS abgeschlossen sein, bevor sich dieser an der Anwendung anmelden darf.

Quelle(n): TSS-WEB

2.6.16 Verwendung sicherer Benutzerkennungen

Vorgabe: Benutzernamen SOLLTEN frei wählbar und personenspezifisch sein. Für die Identifizierung von Anwendern SOLLTE die Benutzerkennung als nicht öffentliche Information behandelt werden.

Hinweise: Die Benutzerkennung SOLLTE daher nicht für andere Anwender ohne administrative Rechte einsehbar sein. Als Benutzerkennung SOLLTE NICHT eine bekannte Information, wie z. B. eine E-Mail-Adresse, verwendet werden.

Quelle(n): BSI-WEB-MBP; TSS-WEB

2.6.17 Benutzerpasswörter entsprechend Passwort-Policy

Vorgabe: Benutzerpasswörter MÜSSEN der Passwort-Policy entsprechen. Dies MUSS sowohl bei der Registrierung, als auch beim Neusetzen des Passworts durch den Benutzer geprüft werden.

Hinweise: Die in den folgenden beiden Kapiteln (Benutzerpasswörter I und II) enthaltenen Vorgaben zu Passwortanforderungen MÜSSEN auch bei der Registrierung, soweit zutreffend, eingehalten werden.

Quelle(n): TSS-WEB

2.6.18 Keine ausschließliche Zugriffskontrolle via HTTP Basic

Vorgabe: HTTP Basic Auth SOLLTE nur als zusätzliche Zugriffskontrolle zum Einsatz kommen und DARF ausschließlich über HTTPS verwendet werden.

Quelle(n): TSS-WEB

2.6.19 Sichere Fehlerbehandlung bei Authentisierungskontrollen

Vorgabe: Alle Authentisierungskontrollen MÜSSEN bei einem Fehlerfall in einen sicheren Zustand fallen.
Quelle(n): BSI-WEB-AN

2.6.20 Reauthentisierung bei sensiblen Operationen

Vorgabe: Eine erneute Authentisierung (Reauthentisierung) SOLLTE erfolgen, bevor der Aufruf einer sensiblen Operation (Änderung des Benutzernamens, der E-Mail-Adresse, die auch für die Kennwortzurücksetzung verwendet wird etc.) zugelassen wird. Bei Anwendungen mit hohem Schutzbedarf MUSS dies bei sensiblen Operationen erfolgen.
Quelle(n): BSI-WEB-AN

2.6.21 Abmeldefunktionalität

Vorgabe: Findet eine Authentifizierung statt, so MUSS eine Abmeldefunktionalität umgesetzt werden. Diese MUSS jederzeit erreichbar und einfach erkennbar sein.
Hinweise: Weiterhin SOLLTEN Benutzer darauf hingewiesen werden, sich nach Nutzung der Anwendung abzumelden. Nach automatischem Abmelden durch ein Timeout SOLLTE nach erneutem Anmelden ein Hinweis angezeigt werden, dass die vorige Sitzung aufgrund von Inaktivität deaktiviert wurde und darum gebeten werden, die manuelle Abmeldefunktionalität zu nutzen.
Quelle(n): BSI-WEB-MBP

2.7 Benutzerpasswörter I: Stärke und Behandlung

Diese Richtlinie regelt nicht die allgemeinen Anforderungen hinsichtlich Passwortstärke, da hierfür eine separate VAV eigene Arbeitsrichtline Kennwörter existiert. Dieses Kapitel enthält jedoch zusätzliche Anforderungen im Umgang mit Passwörtern und sonstigen Authentisierungsfaktoren.

2.7.1 Umsetzung der Passwortanforderungen

Vorgabe: Die Passwortanforderungen (Länge, Komplexität usw.) MÜSSEN entsprechend der aktuell gültigen Kennwortrichtlinie umgesetzt werden.
Hinweise: Dies ist derzeit die Arbeitsrichtlinie Kennwörter in ihrer jeweils aktuell gültigen Version. Die geforderte Kennwortkomplexität SOLLTE konfigurierbar sein.
Quelle(n): OWASP-SCP

2.7.2 Standardvorgaben für Benutzerpasswörter

Vorgabe: Sofern durch eine Passwort-Policy nicht anders vorgegeben, MÜSSEN Benutzerpasswörter: Mindestens 8 Zeichen Länge besitzen, Buchstaben, Zahlen und Sonderzeichen enthalten, sowie nicht identisch mit dem Benutzernamen sein.
Hinweise: Sofern durch eine Passwort-Policy nicht anders vorgegeben, SOLLTEN Passwörter keine Maximallänge besitzen und mindestens 50 Zeichen lang sein dürfen.
Quelle(n): TSS-WEB

2.7.3 Verdeckte Eingabe in Passworteingabefeld

Vorgabe: Eingabefelder für Passwörter MÜSSEN Benutzereingaben verdecken (bspw. durch Anzeige von Sternchen [„*“] anstelle der eingegebenen Zeichen).
Hinweise: Für Webanwendungen MUSS dies über Eingabefelder („input“) vom Typ („type“) „password“ realisiert werden.
Quelle(n): OWASP-SCP

2.7.4 Keine Autovervollständigung für Passwortfelder

Vorgabe: Alle Passwortfelder (sowie Formulare, die diese enthalten) MÜSSEN die Autocomplete-Funktion deaktiviert haben.
Quelle(n): BSI-WEB-AN

2.7.5 Keine unverschlüsselte Übertragung nicht-temporärer Passwörter

Vorgabe: Nicht-temporäre Passwörter DÜRFEN NICHT über unverschlüsselte Kanäle (unverschlüsselte E-Mail, HTTP ohne TLS, SMS) kommuniziert werden.
Hinweise: Temporäre Passwörter bzw. Anmelde-Tokens (idealerweise mit zeitlich eingeschränkter Gültigkeit), die nach erstmaliger Verwendung eine Passwortänderung erzwingen, DÜRFEN bei nicht kritischen Anwendungen für normale Benutzerkonten unverschlüsselt übertragen werden.
Quelle(n): OWASP-SCP

2.7.6 Ablaufzeit für temporäre Passwörter / Tokens

Vorgabe: Verwendete temporäre Passwörter und Tokens DÜRFEN NUR über einen kurzen Zeitraum gültig sein.
Hinweise: Der Gültigkeitszeitraum für temporäre Passwörter und Tokens SOLLTE konfigurierbar sein.
Quelle(n): OWASP-SCP

2.7.7 Salting von Passwort-Hashes

Vorgabe: Passwörter von Benutzern DÜRFEN NUR als Hash mit Salt gespeichert werden. Passwort-Hashes MÜSSEN mit Verwendung eines zufälligen, sicher generierten Salt-Werts erstellt werden.

Hinweise: Hierzu SOLLTE standardmäßig der PBKDF2-Algorithmus eingesetzt werden, falls FIPS-Kompatibilität / Zertifizierbarkeit bestehen soll oder dies nicht bekannt ist. Andernfalls KANN Argon2, scrypt oder bcrypt verwendet werden. Als unsicher geltende Hash-Algorithmen (MD5 oder älter, SHA-1, LM, GOST, Panama, Snefru) DÜRFEN NICHT verwendet werden. Es SOLLTE weiterhin Key Stretching (mehrfache Passes des Hashing-Algorithmus) erfolgen; 100.000 Iterationen werden derzeit empfohlen.

Quelle(n): BSI-WEB-AN; OWASP-SCP

2.7.8 Zugriffsbeschränkung für Zugangsdaten

Vorgabe: Alle Arten von Zugangsdaten, die im Klartext lokal gespeichert sind und für den Zugriff auf sicherheitsrelevante Konfigurationsdaten verwendet werden (z. B. geheime Schlüssel, Passwörter), MÜSSEN gegen nicht autorisierten Zugriff geschützt sein.

Quelle(n): BSI-WEB-AN

2.8 Benutzerpasswörter II: Änderung und Zurücksetzung

Bei der Implementierung von Funktionalität für die Änderung und Zurücksetzung von Benutzerpasswörtern können leicht Sicherheitslücken entstehen. Die nachfolgenden Maßnahmen adressieren diese Thematik.

2.8.1 Änderbarkeit von Passwörtern

Vorgabe: Benutzerpasswörter MÜSSEN durch den Benutzer geändert werden können.

Quelle(n): TSS-WEB

2.8.2 Änderung von Initialpasswörtern

Vorgabe: Initialpasswörter MÜSSEN beim ersten Anmelden durch den Benutzer geändert werden.

Hinweise: Auch temporäre Passwörter einer „Passwort vergessen“-Funktion MÜSSEN beim ersten Anmelden durch den Nutzer geändert werden.

Quelle(n): TSS-WEB

2.8.3 Erneute Authentifizierung bei Passwortänderung

Vorgabe: Beim Ändern seines Passworts MUSS ein Benutzer sein altes Passwort zur Bestätigung eingeben.
Hinweise: Ausnahme ist die Verwendung eines Tokens stattdessen bei Nutzung einer „Passwort vergessen“-Funktion.
Quelle(n): TSS-WEB; OWASP-SCP

2.8.4 Passwortänderung erzwingen

Vorgabe: Die Forcierung von Passwortänderungen nach definierten Ablaufzeiträumen MUSS entsprechend der geltenden Kennwortrichtlinie umgesetzt werden.
Hinweise: Dies ist derzeit die Arbeitsrichtlinie Kennwörter in ihrer jeweils aktuell gültigen Version. Die Passwortablaufzeiträume SOLLTEN konfigurierbar sein.
Quelle(n): OWASP-SCP

2.8.5 Einheitliches Verfahren für Passwortänderung

Vorgabe: Sollte ein Benutzerpasswort an mehreren Stellen geändert oder gesetzt werden können, MUSS die Prüfung im Hintergrund stets mit demselben technischen Verfahren erfolgen.
Hinweise: Grundsätzlich SOLLTEN Passwörter an so wenig Stellen wie möglich verwendet werden, stattdessen SOLLTE ein einheitliches Login Verfahren verwendet werden.
Quelle(n): TSS-WEB

2.8.6 Sicherheitsniveau der Zugangsdatenänderung

Vorgabe: Die Änderung von Zugangsdaten DARF NUR über einen Mechanismus erfolgen, der mindestens über den gleichen Schutzlevel verfügt, wie die primäre Authentisierung.
Quelle(n): BSI-WEB-AN

2.8.7 Verhindern von Passwortwiederverwendung

Vorgabe: Die Verwendung von zuvor bereits durch den Anwender genutzten Passwörtern SOLLTE unterbunden werden.
Hinweise: Die letzten x Passwörter des Anwenders SOLLTEN (als sichere kryptographische Hashes mit Salt, entsprechend der sicheren Ablage des geltenden Passworts) abgelegt werden und gegen diese Liste verglichen werden, wobei „x“ ein konfigurierbarer Wert sein SOLLTE. Ein empfohlener Wert für x ist 10.
Quelle(n): OWASP-SCP

2.8.8 Anzeige der Passwortstärke

Vorgabe: Bei Änderung eines Passworts durch einen Benutzer SOLLTE diesem die aktuelle Stärke des eingegebenen Passworts visuell angezeigt werden (Passwortstärkefunktion).
Quelle(n): TSS-WEB

2.9 Authentifizierung am Backend

Für Anwendungs-Backends, insbesondere eine Eigenentwicklung dieser, sollten besondere Anforderungen an die Sicherheit bestehen, da hierüber schlimmstenfalls eine weitreichende und zentralisierte Kompromittierung erfolgen kann. Dieser Umstand wird bei der Programmierung durch die nachfolgenden Vorgaben berücksichtigt.

2.9.1 Sichere Authentifizierung gegen administrative Backends

Vorgabe: Für die Authentifizierung zur Nutzung von administrativen Funktionen bzw. Funktionen, die über die eines normalen Anwenders hinausgehen, MÜSSEN mindestens gleichwertige Sicherheitsmechanismen bei der Authentifizierung umgesetzt werden wie für die normale Benutzerauthentifizierung.
Hinweise: Dieser Punkt ist automatisch erfüllt, wenn eine zentralisierte Implementierung für alle Authentifizierungsmechanismen verwendet wird (siehe „Zentrale Authentifizierungsimplementierung“).
Quelle(n): OWASP-SCP

2.9.2 Multi-Faktor-Authentifizierung für streng vertrauliche Konten

Vorgabe: Bei Authentifizierung von Konten, die für Zugriff auf streng vertrauliche Informationen autorisiert sind bzw. Transaktionen mit hohen monetären Volumina auslösen können, MUSS (mindestens) eine Zwei-Faktor-Authentifizierung vorgenommen werden.
Hinweise: Zusätzliche Faktoren für die Authentifizierung sind beispielsweise ein SMS-, E-Mail- oder Mobile App-Token, ein FIDO2-Token oder eine Smart Card.
Quelle(n): OWASP-SCP

2.9.3 Erneute Authentifizierung vor Durchführung kritischer Aktionen

Vorgabe: Vor Durchführung von kritischen Aktionen SOLLTEN Anwender erneut authentifiziert werden.
Hinweise: Kritische Aktionen können z. B. die Vergabe von administrativen Rechten oder der Download bzw. das Einspielen eines Backups sein.
Quelle(n): OWASP-SCP

2.9.4 Verwendung verschlüsselter Kanäle

Vorgabe: Für alle externen sowie Backend-Verbindungen, die entweder authentisiert sind oder vertrauliche Daten bzw. sensible Funktionen betreffen, MUSS ein verschlüsselter Kanal verwendet werden. Ausgenommen hiervon sind interne Verbindungen wie lokal laufende Services, auf die der Zugriff ausschließlich lokal erfolgt und die dementsprechend abgeschottet sind.
Quelle(n): BSI-WEB-AN

2.9.5 Minimalprinzip für Rechte von Service- und Support-Konten

Vorgabe: Service- und Support-Konten sowie Konten für Zugriff von Drittsystemen MÜSSEN lediglich die minimalen für die jeweiligen Anwendungsfälle benötigen Rechte besitzen.
Hinweise: Hierbei SOLLTEN soweit möglich nicht nur die Objekte, Informationen und Methoden eingeschränkt werden, sondern auch die Art des Zugriffs (z. B. nur lesend).
Quelle(n): OWASP-SCP

2.9.6 Gegenseitige Authentifizierung verteilter Anwendungskomponenten

Vorgabe: Verteilte Anwendungskomponenten SOLLTEN sich stets gegenseitig authentifizieren.
Hinweise: Dies SOLLTE auf mehreren Ebenen durchgeführt werden (z. B. IP-Adressen, X.509-Zertifikate, Passwörter). Hierfür SOLLTE ein Verfahren zum Einsatz kommen, das nicht auf Basis geheimer Schlüssel (Passwörter) arbeitet. Besser geeignet sind etwa X.509-Zertifikate (Public-Key-Authentifizierung oder der Einsatz von Ticket-basierten Authentifizierungssystemen wie Kerberos).
Quelle(n): TSS-WEB

2.10 Dateiuploads und -downloads

Wenn Dateidownloads oder gar -uploads anwendungsseitig ermöglicht werden sollen, ist eine Reihe von Sicherheitsaspekten diesbezüglich zu berücksichtigen – sonst haben gerade hierbei Angreifer leichtes Spiel.

2.10.1 Authentifizierung vor Upload

Vorgabe: Bevor ein Upload ermöglicht wird, SOLLTE eine Authentifizierung erfolgen.
Hinweise: Es SOLLTE weiterhin festgehalten oder protokolliert werden, welche Uploads welchen Anwenderkonten zuzuordnen sind.
Quelle(n): OWASP-SCP

2.10.2 Whitelisting erlaubter Dateitypen bei Upload

Vorgabe: Die erlaubten Dateitypen DÜRFEN NUR den in den vorgesehenen Anwendungsfällen benötigten Typen entsprechen.

Hinweise: Dateitypen, die clientseitig ausführbaren Code enthalten können (z. B. „.html“, „.js“, „.exe“ oder „.bat“) DÜRFEN NICHT hochgeladen werden können. Die Prüfung MUSS auf Basis von Whitelisting (nur erlaubte Dateitypen werden zugelassen) durchgeführt werden. Die erlaubten Dateitypen SOLLTEN konfigurierbar sein.

Quelle(n): OWASP-SCP; BSI-WEB-AN.DV5; TSS-WEB

2.10.3 Verifizierung des MIME-Types von Uploads

Vorgabe: Zusätzlich zur Dateierweiterung MUSS verifiziert werden, dass der angegebene MIME-Type erlaubt ist und zu der Dateinamenerweiterung passt.

Hinweise: Hierbei MUSS der MIME-Type auf Basis des Dateiinhalts (Datei-Header) ermittelt werden, es DARF NICHT der vom Client übermittelte MIME-Type ungeprüft übernommen werden. In UNIX-Umgebungen KANN dies über das Werkzeug „file“ geschehen. Eine ausschließliche Überprüfung der Dateinamenerweiterung DARF NICHT erfolgen.

Quelle(n): TSS-WEB; BSI-WEB-AN

2.10.4 Kein Upload aktiver Dateien

Vorgabe: Der Upload von Dateien, die vom Anwendungsserver interpretiert und ausgeführt werden könnten, MUSS verhindert oder eingeschränkt werden.

Hinweise: Insbesondere Webserver DÜRFEN NICHT in Uploadverzeichnissen Perl oder PHP Skripte ausführen.

Quelle(n): OWASP-SCP

2.10.5 Größenbeschränkung für Uploads

Vorgabe: Die Größe hochgeladener Dateien MUSS auf einen sinnvollen Wert (z. B. 5 MB) beschränkt werden.

Hinweise: Neben Überprüfung in der Anwendung SOLLTE dies auch in der Webserverkonfiguration erfolgen, für Tomcat bspw. Via ”maxPostSize”.

Quelle(n): TSS-WEB

2.10.6 Rate Limiting für Uploads

Vorgabe: Die Anzahl hochgeladener Dateien SOLLTE auf einen sinnvollen Wert beschränkt werden (z. B. 8 Dateien pro Stunde und Benutzer).

Quelle(n): TSS-WEB

2.10.7 Antivirenscan bei Upload

Vorgabe: Von Anwendern hochgeladene Dateien MÜSSEN von einer Antivirensoftware auf schädliche Inhalte überprüft werden.

Hinweise: Die Antivirenüberprüfung SOLLTE unmittelbar nach dem Upload, vor dem finalen Ablegen der hochgeladenen Dateien erfolgen. Von Ergebnissen der Virenprüfung ausgehende Meldungen DÜRFEN NICHT Rückschlüsse auf die eingesetzte Antivirensoftware erlauben. Die eingesetzte Antivirensoftware und die verwendete Signaturdatenbank MUSS regelmäßig aktualisiert werden. Die Antivirensoftware SOLLTE von der Anwendung getrennt betrieben werden, um eine Kompromittierung der Anwendung über Ausnutzung von Schwachstellen in der Antivirensoftware zu erschweren.

Quelle(n): OWASP-SCP

2.10.8 Konvertierung von Dateiuploads

Vorgabe: Bilder und Dokumente SOLLTEN konvertiert werden, um Schadcode zu entfernen und Fehlformatierungen zu korrigieren.

Quelle(n): BSI-WEB-AN

2.10.9 Zugriffsgeschützte Ablage von Uploads

Vorgabe: Hochgeladene Dateien SOLLTEN in einer zugriffsgeschützten Datenbank abgelegt werden.

Hinweise: Eine zugriffsgeschützte Datenbank ist häufig gegeben, allerdings gibt es auch Datenbanken, wie MongoDB, bei der erst die Access Controls aktiviert werden müssen, damit ein Zugriffsschutz sichergestellt ist. Die Ablage kann auch in einem Dateisystem erfolgen, dann ist die sichere und fehlerfreie Verwendung korrekter Dateinamen erforderlich (z. B. „../../datei.txt“ ist kein sicherer Dateiname unter UNIX).

Quelle(n): TSS-WEB

2.10.10 Keine Ausführungsberechtigungen in Upload-Verzeichnissen

Vorgabe: Für Upload-Verzeichnisse DÜRFEN KEINE Ausführungsrechte gesetzt werden.

Hinweise: Unter UNIX-Betriebssystemen ist dies mittels des „chmod“-Befehls möglich.

Quelle(n): OWASP-SCP

2.10.11 Dateiablage für Uploads außerhalb der Applikationsverzeichnisse

Vorgabe: Die Ablage von hochgeladenen Dateien SOLLTE sich außerhalb der Anwendungsverzeichnisse befinden.
Hinweise: Idealerweise SOLLTEN hochgeladene Dateien in separaten, von einem Content-Server verwendeten Verzeichnisse oder Laufwerke abgelegt werden oder in einer Datenbank.
Quelle(n): OWASP-SCP

2.10.12 Sichere Behandlung von Upload-Verzeichnissen in UNIX-Umgebungen

Vorgabe: Bei Uploads in UNIX-Umgebungen SOLLTE das Zielverzeichnis als logisches Laufwerk über den assoziierten Pfad oder die chroot-Umgebung gemountet werden.
Quelle(n): OWASP-SCP

2.10.13 Security Header für Downloads

Vorgabe: Beim Download von Dateien SOLLTEN entsprechende Security Header gesetzt werden, um damit z. B. das MIME Type Sniffing im Browser zu deaktivieren (siehe Anhang „HTTP Security Header-Sicherheitsmechanismen“).
Hinweise: Insbesondere MUSS der Header „X-Content-Type-Options: nosniff“ in Kombination mit einem passenden „Content-Type“-Header gesetzt werden. Weiterhin SOLLTE der „Content-Disposition“-Header für Downloads mit dem Attribut „attachment“, sowie einem sicheren Wert für „filename“ verwendet werden.
Quelle(n): TSS-WEB

2.10.14 Referenzierung nur erlaubter Dateinamen und -typen

Vorgabe: Bei der Referenzierung existierender Dateien MUSS ein Whitelisting-Ansatz für erlaubte Dateinamen und -typen verwendet werden.
Hinweise: Der Wert des übergebenen Parameters MUSS überprüft werden. Wenn dieser nicht einem erwarteten / erlaubten Schema entspricht, SOLLTE die Anfrage abgewiesen werden.
Quelle(n): OWASP-SCP

2.10.15 Downloads über separate Domain

Vorgabe: Dateidownloads SOLLTEN über ein separates Origin erfolgen (z. B. „files.example.com“), um dadurch die Auswirkungen von ausgeführtem Skriptcode zu begrenzen.
Hinweise: Über die Haupt-Domain (hier z. B. „example.com“) SOLLTE NICHT weiterhin zusätzlich direkt Zugriff auf die entsprechenden Dateidownloads gegeben sein.
Quelle(n): TSS-WEB

2.11 Absicherung des Session-Managements

Die Session stellt die Identität eines Benutzers nach erfolgreicher Authentifizierung sicher. Zum Umgang mit ihr sind die nachfolgenden Vorgaben zu relevanten Sicherheitsaspekten essenziell.

2.11.1 Zentrales Sitzungsmanagement

Vorgabe: Soweit möglich, SOLLTE das Sitzungsmanagement zentral erfolgen, i. d. R. durch Verwendung der Standardfunktionalität des verwendeten Servers oder Frameworks hierfür.
Hinweise: Gängig hierfür ist zum Beispiel die Verwendung von LDAP. Diese Eigenschaft ist auch als „Single Sign-On“ oder SSO bekannt.
Quelle(n): OWASP-SCP

2.11.2 Erstellung von Sitzungs-Identifiern in vertrauenswürdigen Systemen

Vorgabe: Es MUSS sichergestellt werden, dass die Erstellung von Sitzungs-Identifiern (Session-IDs) in einem kontrollierbaren, vertrauenswürdigen Kontext erfolgt (i. d. R. serverseitig).
Hinweise: Clientseitiges Festlegen von Session-IDs kann zu sog. Session Fixation-Angriffen führen und DARF NICHT verwendet werden.
Quelle(n): OWASP-SCP

2.11.3 Geprüfte Sitzungsmanagement-Mechanismen mit ausreichend Entropie

Vorgabe: Die verwendeten Sitzungsmanagement-Mechanismen SOLLTEN hinreichend erprobt bzw. geprüft sein, insbesondere MUSS sichergestellt sein, dass die generierten Session-IDs (pseudo)zufällig generiert werden, nicht vorhersagbar sind und über ausreichend Entropie verfügen.
Hinweise: Hierfür MÜSSEN Sitzungs-Identifier mit mindestens 64 Bit Entropie verwendet werden.
Quelle(n): OWASP-SCP; TSS-WEB; BSI-WEB-AN

2.11.4 Bindung der IP-Adresse an die Session

Vorgabe: Eine Session KANN an die jeweilige IP-Adresse eines Clients geknüpft werden. Dies SOLLTE NICHT bei offenen Internetinhalten oder mobil zu erreichenden Anwendungsinhalten geschehen, da die Verfügbarkeit hierdurch einschränkt wird.
Hinweise: Es SOLLTE dennoch ein Limit geben, wie viele Sessions pro IP-Adresse eines Clients möglich sind, wenn der Anwendungsfall dies zulässt.
Quelle(n): BSI-WEB-MBP

2.11.5 Keine Nutzung von Sitzungs-Identifiern außerhalb vorgesehener Kanäle

Vorgabe: Sitzungs-Identifier DÜRFEN NUR über vorgesehene Kanäle kommuniziert werden, insbesondere MUSS sichergestellt werden, dass Session-IDs (bzw. Session Tokens) nur über den HTTP-Cookie-Header kommuniziert werden.
Hinweise: Session-IDs (bzw. Session Tokens) DÜRFEN NICHT in URLs, als GET-Parameter, in Fehlermeldungen oder Protokolldateien usw. verwendet werden.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.11.6 Serverseitiger Schutz der Sitzungsdaten

Vorgabe: Die Sitzungsdaten MÜSSEN serverseitig vor unautorisiertem Zugriff geschützt sein.
Hinweise: Hierfür MUSS sichergestellt werden, dass nicht autorisierte Systembenutzerkonten nicht über Berechtigungen verfügen, mit denen die Sitzungsdaten eingesehen werden können.
Quelle(n): OWASP-SCP

2.11.7 Sitzungs-Cookies mit Domain und Path

Vorgabe: Für Sitzungs-Cookies MÜSSEN die „Domain“- und „Path“-Attribute sinnvoll auf jeweils minimal benötigte Werte gesetzt werden.
Hinweise: Sitzungs-Cookies DÜRFEN NUR für ihren FQDN (Fully Qualified Domain Name, d. h. inklusive vollständiger Subdomain) sowie für den Anwendungspfad (z. B. „/webapp/“) gesetzt werden.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.11.8 Sichere Übermittlung von Sitzungs-IDs

Vorgabe: Wird ein Benutzer mit Hilfe der Sitzung authentifiziert, MUSS der Sitzungsidentifikator ausschließlich über verschlüsselte Kanäle übertragen werden.
Hinweise: Hierfür SOLLTE die Client-Server-Kommunikation ab einschließlich der Authentifizierung ausschließlich über TLS erfolgen.
Quelle(n): BSI-WEB-AN

2.11.9 Cookie-„secure“-Attribut

Vorgabe: Cookies, die über TLS (HTTPS) übertragen werden, MÜSSEN das „secure“-Attribut gesetzt haben, um ein Auslesen der Sitzungsinformationen über Man in the Middle (MitM)-Angriffe an dieser Stelle zu unterbinden.

Hinweise: Ein Cookie muss also die folgende Form aufweisen: "Set-Cookie: CookieName=Wert; path=/; HttpOnly; secure". In PHP kann das secure-Attribut mit dem Aufruf "session.cookie_secure = True" gesetzt werden.

Quelle(n): OWASP-SCP

2.11.10 Cookie-„HttpOnly“-Attribut

Vorgabe: Cookies SOLLTEN das „HttpOnly“-Attribut gesetzt haben, um ein Auslesen der (Sitzungs)informationen über (ggf. durch Angreifer eingeschleusten) Script-Code zu unterbinden. Cookies mit vertraulichem Inhalt, wie z. B. einer Session-ID, MÜSSEN dieses Flag gesetzt haben.

Quelle(n): OWASP-SCP

2.11.11 Bestehende Sitzung bei Anmelden terminieren

Vorgabe: Sollte beim erfolgreichen Anmelden bereits eine Sitzung zum authentifizierten Benutzer existieren, so MUSS diese vollständig terminiert und eine neue Sitzungs-ID (bzw. Session Token) erstellt werden, sofern nicht Mehrfachanmeldungen für die Anwendung explizit gewollt sind.

Hinweise: Bei Verwendung eines Frameworks MUSS die Sitzungsverwaltung des Frameworks hierauf überprüft werden.

Quelle(n): OWASP-SCP

2.11.12 Mehrfache Anwendersitzungen nicht zulassen

Vorgabe: Es MUSS sichergestellt werden, dass zu jedem Anwenderkonto zu jedem Zeitpunkt jeweils nur eine gültige Sitzung existiert, sofern nicht Mehrfachanmeldungen für die Anwendung explizit gewollt sind.

Hinweise: Ein erneutes Anmelden (beispielsweise auf einem anderen Client) MUSS vor Erstellung der neuen Sitzung zur Terminierung der bestehenden Sitzung des Anwenders führen. Ein Hinweis auf den Grund der Sitzungsterminierung SOLLTE für den jeweiligen Client erfolgen, idealerweise einschließlich sinnvoller Meta-Informationen (Zeitpunkt, IP-Adresse, ...).

Quelle(n): OWASP-SCP

2.11.13 Sitzung bei Abmeldung vollständig terminieren

Vorgabe: Beim Abmelden MUSS die Sitzung des Anwenders vollständig terminieren, sodass die verwendete Sitzungs-ID nicht mehr gültig ist.
Hinweise: Bei Verwendung eines Frameworks sind dafür die entsprechenden Funktionen zu verwenden.
Quelle(n): OWASP-SCP; TSS-WEB; BSI-WEB-AN

2.11.14 Abmeldefunktionalität aus authentifizierten Bereichen erreichbar

Vorgabe: Aus allen Teilen der Anwendung, die eine Autorisierung überprüfen, MUSS für angemeldete Anwender eine Abmeldefunktionalität („Logout“) genutzt werden können.
Hinweise: Hierbei MUSS sichergestellt sein, dass nicht bloß der Client eine neue Sitzungs-ID erhält, sondern die abgemeldete Sitzung auch invalidiert wird.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.11.15 Sitzungsablaufdauer

Vorgabe: Es MUSS sichergestellt werden, dass bei Inaktivität die Sitzung nur für einen definierten, sinnvollen und minimalen Zeitraum weiterhin Gültigkeit besitzt, unter Berücksichtigung von Risiken und funktionalen Anforderungen.
Hinweise: Der Gültigkeitszeitraum für inaktive Sitzungen SOLLTE konfigurierbar sein. Empfohlen wird, dass eine Sitzung für eine Anwendung mit normalen Schutzbedarf nach maximal der üblichen Zeitspanne eines Arbeitstags (etwa 9 Stunden) ungültig wird. Ein unbegrenzter Gültigkeitszeitraum (keine Ablaufdauer) ist zulässig, wenn explizit gewollt – wie z. B. bei manchen Portalen ohne sicherheitsrelevante Funktionalität oder Datenführung ggf. sinnvoll.
Quelle(n): OWASP-SCP; TSS-WEB; BSI-WEB-AN

2.11.16 Keine persistenten Sitzungen

Vorgabe: Anmeldungen insbesondere für kritische Systeme DÜRFEN NICHT persistent (= unbegrenzt langfristig gültig) sein, Sitzungen MÜSSEN somit selbst bei Aktivität periodisch terminiert werden.
Hinweise: Der maximale Gültigkeitszeitraum für Sitzungen SOLLTE konfigurierbar sein und die Anforderungen der Use-Cases der Anwendung berücksichtigen. Ein empfohlener Wert hierfür sind 12 Stunden. Anwender SOLLTEN hinreichend auf die bevorstehende Sitzungsterminierung hingewiesen werden, um negative Auswirkungen zu vermeiden.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.11.17 Neuer Sitzungs-Identifier bei Reauthentifizierung

Vorgabe: Bei jeder erneuten Authentifizierung innerhalb der Anwendung MUSS ein neuer Sitzungs-Identifier erstellt und fortan verwendet werden und die alte Session-ID (bzw. Session Token) vollständig terminiert werden.

Hinweise: Bei Verwendung eines Frameworks MUSS die Sitzungsverwaltung des Frameworks hierauf überprüft werden.

Quelle(n): OWASP-SCP

2.11.18 Periodische Erstellung neuer Sitzungs-Identifier

Vorgabe: Sitzungs-Identifier (IDs bzw. Tokens) SOLLTEN nach einem definierten Zeitraum erneuert werden.

Hinweise: Der Zeitraum, nach dem aktive Sitzungen erneuert werden, SOLLTE konfigurierbar sein. Bei Überschreiten des Zeitraums kann bei der nächsten Anfrage eine neue Sitzungs-ID (bzw. Session Token) erstellt und diese fortan verwendet werden. Ein empfohlener Wert für den zu konfigurierenden Zeitraum ist 15 - 30 Minuten. Nach Wechsel auf die neu erstellte Session-ID (bzw. Session Token) MUSS die zuvor verwendete Sitzung terminiert werden.

Quelle(n): OWASP-SCP; TSS-WEB

2.11.19 Periodische erneute Zugriffskontrolle bei lang andauernden authentifizierten Sitzungen

Vorgabe: Falls lang andauernde authentifizierte Sitzungen zulässig sind, SOLLTE periodisch eine Revalidierung der Benutzerautorisierung erfolgen.

Hinweise: Falls hierbei der Entzug einer Berechtigung festgestellt werden sollte, SOLLTE der Benutzer abgemeldet, seine Sitzung terminiert und er zu einer erneuten Authentifizierung aufgefordert werden. Das Zeitintervall für die Revalidierungsperiode SOLLTE konfigurierbar sein. Ein empfohlener Wert ist 10 - 60 Minuten.

Quelle(n): OWASP-SCP

2.11.20 Neuer Sitzungs-Identifier bei Wechsel von HTTP zu HTTPS

Vorgabe: Falls ein Wechsel von HTTP auf HTTPS erfolgt, MUSS die bestehende Sitzung terminiert und eine neue Sitzungs-ID erstellt und weiterverwendet werden.

Hinweise: Es wird empfohlen, ausschließlich HTTPS zu verwenden und von einer Verwendung von unverschlüsseltem HTTP vollständig abzusehen.

Quelle(n): OWASP-SCP

2.11.21 Tokens für sensible Funktionen

Vorgabe: Für sensible Funktionen (wie Konto- und Rechtemanagement, Nachrichten- oder Geldversand usw.) MÜSSEN jeweils sitzungsspezifische, zufällige Tokens mit ausreichend Entropie verwendet werden, um sog. Cross Site Request Forgery (CSRF)-Angriffe zu unterbinden.

Hinweise: Hierbei SOLLTE die Standard-Funktionalität des verwendeten Frameworks genutzt werden, sofern vorhanden. Dieser Mechanismus SOLLTE auch für die Ausloggen-Funktion implementiert werden um Denial of Service (DoS)-Angriffe gegen einzelne Nutzer hierüber auszuschließen. Bei Verwendung eines Frameworks MUSS die Sitzungsverwaltung des Frameworks dahingehend überprüft werden ob CSRF-Token unterstützt werden und aktiviert sind.

Quelle(n): OWASP-SCP; TSS-WEB

2.12 Zugriffskontrollen (Access Controls)

Nach erfolgter Authentifizierung ist insbesondere eine technisch und logisch einwandfrei implementierte, sowie vollständig angewandte Autorisierung nach einem durchdachten Rollen- und Rechtekonzept für die Sicherheit einer Anwendung ausschlaggebend. Die folgenden Vorgaben sollen dies sicherstellen. Weiterhin enthält dieses Kapitel Vorgaben, die den Zugriff auf Ressourcen aus Sicherheitsgründen gezielt einschränken, um Thematiken wie (D)DoS, Enumeration & Brute Force zu adressieren.

2.12.1 Autorisierung by Default

Vorgabe: Anwender DÜRFEN NUR auf geschützte Funktionen, Ressourcen oder Daten zugreifen können, wenn dieser Zugriff zuvor erfolgreich autorisiert wurde. Gemäß dem Prinzip der Zugriffskontrolle, MUSS jeder Zugriff auf ein beliebiges Objekt auf seine Befugnis hin überprüft werden.

Hinweise: Die Rechte DÜRFEN NUR anhand der serverseitig dem Anwender zugeordneten Rollen und Rechte ermittelt werden. Der Anwender selbst DARF NUR anhand einer authentifizierten Sitzungs-ID identifiziert werden.

Quelle(n): BSI-WEB-AN; BSI-WEB-MBP

2.12.2 Zentraler Autorisierungsmechanismus

Vorgabe: Für alle Arten von Zugriffen auf geschützte Ressourcen SOLLTE ein zentraler Mechanismus verwendet werden (inklusive für Bibliotheken, die externe Autorisierungsdienste aufrufen).

Quelle(n): BSI-WEB-AN; OWASP-SCP

2.12.3 Prüfung von Rollen und Rechten

Vorgabe: Zugriffsprüfungen SOLLTEN die Rollen des genutzten Benutzerkontos und damit verknüpfte Rechte, gegen die zum Zugriff auf bestimmte Datenobjekte und Objektmethoden benötigten Rechte, überprüfen.
Quelle(n): TSS-WEB

2.12.4 Protokollierung bei der Autorisierung

Vorgabe: Die Protokollierung sämtlicher Entscheidungen von Zugriffskontrollen MUSS möglich sein und alle sicherheitsrelevanten Entscheidungen (Verweigerung von Zugriff) SOLLTEN standardmäßig protokolliert werden.
Hinweise: Insbesondere SOLLTEN alle „Indicators of Compromise" geloggt werden können und standardmäßig geloggt werden (jeweils erfolgreiche und erfolglose Login-Versuche, Zugriffe auf sicherheitsrelevante Funktionen und Informationen, Logout).
Quelle(n): BSI-WEB-AN

2.12.5 Autorisierung auf Basis vertrauenswürdiger Informationsobjekte

Vorgabe: Es MUSS sichergestellt werden, dass Entscheidungen bezüglich Autorisierung auf Basis von vertrauenswürdigen Informationsobjekten erfolgt (i. d. R. serverseitige Sitzungsobjekte).
Hinweise: Entsprechend DÜRFEN NICHT Informationen zu erfolgreicher Autorisierung ungeprüft vom Client (via z. B. GET-Parameter) übernommen werden – hier darf lediglich die Session-ID bzw. ein Session Identifier überprüft werden.
Quelle(n): OWASP-SCP; TSS-WEB; BSI-WEB-AN

2.12.6 Zugriffsprüfungen auf mehreren Ebenen

Vorgabe: Zugriffsprüfungen SOLLTEN sowohl auf URL-, also auch auf Datei-, Methoden- und Objekt-Ebene durchgeführt werden.
Hinweise: Beispielsweise SOLLTE eine Webanwendung sowohl Rechte bei Zugriff auf eine Route, bei Zugriff auf ein Objekt und bei Zugriff auf zugrundeliegende Daten überprüfen.
Quelle(n): TSS-WEB

2.12.7 Sichere Fehlerbehandlung bei der Autorisierung

Vorgabe: Bei Auftreten von Fehlern in der Autorisierung MUSS sichergestellt werden, dass die Autorisierung standardmäßig fehlschlägt, dass keine vertraulichen Informationen ausgegeben werden und ein Abfangen des Fehlers in einen sicheren Zustand führt.

Hinweise: Ebenfalls MUSS die Autorisierung verweigert werden, falls die Konfiguration der Anwendung hinsichtlich Autorisierung nicht geladen werden kann oder die Verbindung eines hierfür angebundenen Systems (z. B. AD) nicht funktioniert oder fehlerhaft ist.

Quelle(n): OWASP-SCP; BSI-WEB-AN

2.12.8 Zugriffskontrolle bei allen Anfragen

Vorgabe: Für alle Anfragen, einschließlich jene die von serverseitigen Scripts, Includes oder Rich Client-Technologien wie AJAX oder Flash ausgelöst werden, MÜSSEN Zugriffskontrollen durchgeführt werden.

Hinweise: Zum Beispiel DARF NICHT eine vorhandene Authentifizierung als ausreichende Zugriffskontrolle genutzt werden, wenn mehrere Benutzerrollen/-rechte vorgesehen sind.

Quelle(n): OWASP-SCP

2.12.9 Dedizierte Autorisierung sensibler Objektzugriffe

Vorgabe: Jeder sensible Objektzugriff (z. B. auf ein Datenbankobjekt) SOLLTE stets dediziert autorisiert werden (Complete-Mediation-Prinzip).

Hinweise: Das heißt, dass für jeden sensiblen Objektzugriff einzeln überprüft werden SOLLTE, ob für die gewünschte Aktion ausreichende Rechte vorliegen.

Quelle(n): TSS-WEB

2.12.10 Zusätzliche Absicherung sicherheitskritischer Bereiche

Vorgabe: Vor Anzeige von vertraulichen Informationen oder vor Ausführung sicherheitskritischer Funktionen SOLLTE eine erneute Authentifizierung stattfinden. Für Anwendungen mit erhöhtem Schutzbedarf MUSS dies umgesetzt werden.

Hinweise: Beispielsweise SOLLTE vor einer Passwortänderung das aktuelle Passwort erneut abgefragt und überprüft werden.

Quelle(n): BSI-WEB-MBP

2.12.11 Checkliste Zugriffskontrolle

Vorgabe: Mindestens für folgende Aspekte der Anwendung MUSS eine Zugriffskontrolle erfolgen: Zugriffsbeschränkte URLs; Zugriffsbeschränkte Funktionen; Direkte Objektreferenzen; Services; Anwendungsdaten; Anwender- und Datenattribute; Policy-Informationen des Zugriffskontrollmechanismus; Sicherheitsrelevante Konfigurationsinformationen

Quelle(n): OWASP-SCP

2.12.12 Übereinstimmung serverseitiger Zugriffskontrollmechanismen mit Informationen in Präsentationsschicht

Vorgabe: Die über die Präsentationsschicht dem Anwender bereitgestellten Informationen MÜSSEN entsprechend der durchgeführten Zugriffskontrolle aufbereitet werden.

Hinweise: Hierbei MÜSSEN z. B. Views den Zugriff auf Informationen 1:1 entsprechend der Logik der Zugriffskontrolle abbilden.

Quelle(n): OWASP-SCP

2.12.13 Keine Autorisierung auf Basis nicht vertrauenswürdiger Informationen

Vorgabe: Nicht vertrauenswürdige Informationen sind Angaben, die beispielsweise durch Angreifer gefälscht werden können. Diese DÜRFEN NICHT als alleinige Kriterien für Autorisierungsüberprüfungen herangezogen werden.

Hinweise: Einfach durch Angreifer fälschbare Informationen sind z. B. HTTP-Header, wie „Referrer“ oder „User-Agent“.

Quelle(n): OWASP-SCP

2.12.14 Kontoauditierung und Deaktivierung nicht genutzter Konten

Vorgabe: Es SOLLTE ein Kontoauditierungsprozess durchgeführt werden, indem nicht mehr benötigte Konten identifiziert und deaktiviert werden. Zudem SOLLTEN automatisch nicht mehr genutzte Konten nach einem definierten Zeitraum deaktiviert werden.

Hinweise: Die Ablaufzeit für die automatische Kontodeaktivierung SOLLTE konfigurierbar sein. Ein empfohlener Wert ist maximal 30 Tage nach Ablauf des Kontopassworts.

Quelle(n): OWASP-SCP

2.12.15 Kontodeaktivierung und Sitzungsterminierung bei Autorisierungsentzug

Vorgabe: Die Anwendung MUSS bei Entzug der Autorisierung (z. B. durch Veränderung der Rollen oder des Beschäftigungsverhältnisses des Anwenders) die Deaktivierung des Anwenderkontos und die Terminierung etwaiger vorhandener Sitzungen unterstützen.

Hinweise: Hierbei MUSS sichergestellt sein, dass nicht durch Entzug der Autorisierung für sämtliche Administratoren der administrative Zugang zu der Anwendung vollständig deaktiviert ist.

Quelle(n): OWASP-SCP

2.12.16 Ablauf / Revalidierung von Benutzerberechtigungen

Vorgabe: Jede privilegierte Benutzerberechtigung SOLLTE mit einer zeitlichen Gültigkeit versehen werden.

Hinweise: Nach Ablauf der Gültigkeitsdauer MUSS erneut auf vorhandene Rechte überprüft werden. Die Gültigkeitsdauer SOLLTE konfigurierbar sein.

Quelle(n): TSS-WEB

2.12.17 „Least Privilege“-Prinzip für Anwenderkonten

Vorgabe: Das „Least Privilege“-Prinzip MUSS für Anwenderkonten umgesetzt werden: Anwender DÜRFEN NUR auf die Funktionalität, Daten und Systeminformationen zugreifen, die sie für ihre vorgesehenen Arbeitsabläufe benötigen.

Hinweise: Hierbei SOLLTEN soweit möglich nicht nur die Objekte, Informationen und Methoden eingeschränkt werden, sondern auch die Art des Zugriffs (z. B. nur lesend).

Quelle(n): OWASP-SCP

2.12.18 Vermeidung von Enumeration

Vorgabe: „Resource Enumeration“ SOLLTE weitmöglichst ausgeschlossen werden. Hierfür SOLLTEN nicht-fortlaufende, nicht-numerischer IDs mit erhöhter Entropie für den Zugriff auf Ressourcen verwendet werden.

Hinweise: Weiterhin KÖNNEN zusätzliche Mechanismen entsprechend der „Brute Force-Gegenmaßnahmen“ aus dem Kapitel „Authentifizierung & Registrierung von Benutzern“ umgesetzt werden (CAPTCHAs, Teergrube).

Quelle(n): BSI-WEB-MBP

2.12.19 Cross-Domain-Zugriffe

Vorgabe: Cross-Domain-Zugriffe MÜSSEN über sichere Verfahren (z. B. „Cross-Origin Resource Sharing“ [kurz: „CORS“]) und minimaler Berechtigungen durchgeführt werden und MÜSSEN serverseitig im Hinblick auf ihre Herkunft autorisiert werden (z. B. durch Auswerten des Origin-Headers).

Quelle(n): TSS-WEB

2.12.20 CSRF-Schutz

Vorgabe: Es MUSS jeweils ein kryptographisch starker Zufallstoken generiert werden, der Teil sämtlicher Links und Formulare ist, die Transaktionen oder Zugriffe auf Daten bereitstellen. Zudem MUSS überprüft werden, ob der korrekte Token des jeweiligen Benutzers vorhanden ist, bevor eine Anfrage ausgeführt wird.

Quelle(n): BSI-WEB-AN

2.12.21 Minimalprinzip für Anwendungsrechte

Vorgabe: Applikationen DÜRFEN NUR die minimalen Rechte besitzen, die für die Ausführung der erforderlichen Funktionalität unbedingt benötigt werden.

Quelle(n): BSI-WEB-AN

2.12.22 Automatisierungsschutz beim Mailversand

Vorgabe: Beim Versand von Mails über Webservices, Formulare etc. MUSS an geeigneter Stelle sichergestellt werden, dass nicht zu viele Nachrichten in zu kurzer Zeit gesendet werden können, um ein Blacklisting der Mailserver zu verhindern.

Hinweise: Bei Formularen MUSS ein Grundschutz bestehen. Mögliche Methoden nach Benutzererfahrung (beste Erfahrung am Anfang) sortiert: Verzögerung des Absendens nach Aufwand des Ausfüllens des Formulars („Form Submit Delay“); Verwendung eines Honeypot-Felds (z. B. über JavaScript/CSS „display:none“ bzw. weiße Felder auf weißem Untergrund) – hierbei ist mit „tabindex="-1"“ und „autocomplete="off"“ eine Fehl-eingabe zu verhindern; Nutzung von externen Blacklist-Angeboten (stopforumspam.com, protecthoneypot.org); Erstellung eines einfachen CAPTCHA (z. B. einfache Matheaufgabe, friendlycaptcha.com), dass lokal zur Verfügung gestellt wird. Wenn die Maßnahmen des Grundschutzes umgangen wurden, sollte ein komplexeres CAPTCHA eingesetzt werden (z. B. hcaptcha.com).
Der Einsatz externer Angebote (Blacklist/CAPTCHA) ist vor Einsatz datenschutzrechtlich zu prüfen.
Der Schutz von Webservices MUSS mindestens eine pro Client (Vhv.de-Webseite oder Tarifrechner „KFZ“ etc.) authentifizierte Verbindung, an die ein Rate Limit geknüpft ist, enthalten. Weiter sollte ein Soft Limit definiert werden, dass zu einer Verlangsamung („Tar Pit“) der Anfragen pro Kontext eingesetzt wird.

Quelle(n): VHV-RSE

2.13 Fehlerbehandlung & Protokollierung

Die Themen Fehlerbehandlung und Protokollierung (Logging) liegen logisch eng beieinander, weshalb zugehörige Maßnahmen in diesem Kapitel gemeinsam enthalten sind.

2.13.1 Vermeidung einer zu hohen Fehlertoleranz

Vorgabe: Fehlerzustände MÜSSEN sicher abgefangen werden, jedoch DARF NICHT eine automatische Fehlerkorrektur dazu führen, dass valide Fehlerfälle ignoriert und nicht behandelt werden. Weiterhin SOLLTEN Fehlerzustände oder Eingaben, die auf einen möglichen Missbrauch schließen lassen, nicht korrigiert werden – diese KÖNNEN zum Beenden der jeweiligen Sitzung führen.

Quelle(n): BSI-WEB-MBP

2.13.2 Abfangen von Fehlern

Vorgabe: Anwendungsfehler MÜSSEN abgefangen werden. Es DARF NICHT von einer ausreichenden Fehlerbehandlung außerhalb der Anwendung (z. B. durch den Server oder den Client) ausgegangen werden.

Hinweise: Beispielsweise MÜSSEN alle Exceptions abgefangen und SOLLTEN geloggt werden. Die Fehlermeldungen, die durch Clients einsehbar sein sollen, MÜSSEN explizit via Whitelisting festgelegt werden.

Quelle(n): OWASP-SCP; BSI-WEB-AN

2.13.3 Vermeidung unsicherer Anwendungszustände

Vorgabe: Es MUSS sichergestellt werden, dass die Anwendung beim Auftreten eines Fehlers niemals in einen unsicheren Zustand fallen kann und Benutzern in einem solchen Fall keine technischen Details (z. B. durch Ausgabe von Stack Traces) angezeigt werden.

Quelle(n): TSS-WEB

2.13.4 Sichere Fehlerbehandlung bei Systemvorgängen

Vorgabe: Bei Auftreten von Fehlern auf Systemebene und beim Umgang mit der Systemkonfiguration MUSS sichergestellt werden, dass Fehler sicher abgefangen werden.

Hinweise: Häufige auf Systemebene auftretende Fehler, die sicher abgefangen werden MÜSSEN, sind z. B. Fehlschlagen beim Erstellen eines File-Handles oder Fehler beim Schreiben von Dateien aufgrund von nicht ausreichendem Speicherplatz.

Quelle(n): OWASP-SCP

2.13.5 Fehlerbehandlung in Sicherheitskontrollmechanismen

Vorgabe: Bei der Behandlung von Fehlern aus Sicherheitskontrollmechanismen (wie Anmeldung, Zugriffskontrollen, Rollen / Rechte-Überprüfung) MUSS sichergestellt sein, dass standardmäßig der Zugriff verweigert wird.

Hinweise: Technisch bedeutet dies, dass die Ausgabe einer Information oder das Ausführen einer Methode z. B. innerhalb einer If-Abfrage auf ausreichende Rechte stattfindet, oder nicht ausreichende Rechte vorab zu einem Funktionsabbruch führen.

Quelle(n): OWASP-SCP; BSI-WEB-AN

2.13.6 Ressourcen im Fehlerfall freigeben

Vorgabe: Es MUSS sichergestellt werden, dass etwaige gesperrte Ressourcen, Zugriffs-Handles und allokierte Speicherbereiche im Fehlerfall wieder sicher freigegeben werden.
Hinweise: Geöffnete Dateien sowie Datenbankverbindungen MÜSSEN geschlossen werden, um die Verfügbarkeit sicherzustellen.
Quelle(n): OWASP-SCP

2.13.7 Generische Fehlermeldungen und spezielle Fehlerseiten

Vorgabe: Fehlermeldungen, die dem Benutzer durch die Anwendung zugänglich gemacht werden, MÜSSEN generisch sein und damit so wenig Information wie möglich und nur so viel Information wie nötig enthalten. Es SOLLTEN eigene Fehlermeldungen / Seiten verwendet werden.
Hinweise: Idealerweise SOLLTEN Fehlerseiten Verweise zu Anwendungshilfe/-dokumentation und/oder technischem Support beinhalten. Das Schreiben ausführlicher Meldungen in ein (technisches) Protokoll, unter Einhaltung des Datenschutzes, wird durch diese Vorgabe nicht eingeschränkt.
Quelle(n): OWASP-SCP

2.13.8 Fehlerbehandlung ohne Ausgabe von Debug-Informationen oder Stack Traces

Vorgabe: Bei der Fehlerbehandlung DÜRFEN NICHT vertrauliche Daten wie Debug-Informationen oder Stack Trace-Informationen an den Benutzer ausgegeben werden.
Hinweise: Dies DARF in Entwicklungsumgebungen jeweils geschehen, in Abnahme- und Produktivumgebungen jedoch nicht. Wird die Ausgabe von Debug-Informationen oder Stack Traces unterstützt, so MUSS dies konfigurierbar sein. Das Schreiben ausführlicher Debug-Informationen oder Stack Traces in ein (technisches) Protokoll außerhalb von Entwicklungsumgebungen, unter Einhaltung des Datenschutzes, wird durch diese Vorgabe nicht eingeschränkt.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.13.9 Keine vertraulichen Daten in Fehlermeldungen

Vorgabe: Dem Benutzer angezeigte Fehlermeldungen DÜRFEN NICHT vertrauliche Daten wie Systemdetails, Sitzungs-IDs oder erweiterte Kontoinformationen enthalten.
Quelle(n): OWASP-SCP

2.13.10 Zentrale Protokollierungsfunktionalität

Vorgabe: Soweit möglich, SOLLTE das Protokollieren zentral erfolgen, z. B. in einer globalen Protokollfunktion oder über den Aufruf einer Standardprotokollierungsmethode des verwendeten Frameworks.

Hinweise: Logging SOLLTE in Klartextformaten geschehen und nicht in Binaryformaten. Weiterhin SOLLTEN Logs sofort auf das Speichermedium geschrieben werden und nicht erst gecached werden.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.13.11 Protokollierungsfunktionen in vertrauenswürdigen Systemen

Vorgabe: Es MUSS sichergestellt werden, dass Protokollierungsfunktionen von vertrauenswürdigen Systemen durchgeführt werden (i. d. R. serverseitig).
Hinweise: Clientseitiges Caching von zu protokollierenden Informationen ist entsprechend unzulässig und DARF NICHT verwendet werden.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.13.12 Monitoring gegen kontoübergreifende Brute Force-Angriffe

Vorgabe: Es SOLLTEN Anmeldeversuche gegen mehrere Anwenderkonten unter Verwendung derselben Passwörter identifiziert und protokolliert werden.
Hinweise: Es SOLLTEN grundsätzlich alle Anmeldeversuche protokolliert werden, zusätzlich KANN ein Passwort-Hash unter Verwendung eines anwendungsspezifischen Salts mitprotokolliert werden, um eine entsprechende Auswertung hinsichtlich account-übergreifendem Brute-Force sicherzustellen.
Quelle(n): OWASP-SCP

2.13.13 Protokollierung von Ergebnissen der Sicherheitskontrollmechanismen

Vorgabe: Es MUSS jeweils das Ergebnis (positiv, wie negativ) von Sicherheitskontrollmechanismen (wie Anmeldung, Zugriffskontrollen, Rollen / Rechte-Überprüfung) protokolliert werden.
Quelle(n): OWASP-SCP; TSS-WEB

2.13.14 Protokollierung relevanter Metadaten

Vorgabe: Protokolleinträge MÜSSEN jeweils mindestens die folgenden Metadaten beinhalten: Zeitstempel (aus vertrauenswürdiger Systemkomponente; i. d. R. Zeitstempel des Servers); Log-Level des Ereignisses; Markierung sicherheitsrelevanter Ereignisse, wenn diese im gemeinsamem Protokoll mit anderen Protokolleinträgen protokolliert werden; Identität des Kontos / Anwenders, der das Ereignis ausgelöst hat (i. d. R. User-ID); IP-Adresse, die die Anfrage ausgelöst hat; Ergebnis des Ereignisses (Erfolg oder Misserfolg); Beschreibung des Ereignisses
Hinweise: Für „Log-Level des Ereignisses“ SOLLTEN die folgenden Syslog- und RFC 5425-konformen Abstufungen verwendet werden: Emergency (0/"emerg"), Alert (1/"Alert"), Critical (2/"crit"), Error (3/"err"), Warning (4/"warn"), Notice (5/"notice"), Informational (6/"info"), Debug (7/"debug").
Quelle(n): OWASP-SCP

2.13.15 Sicheres Escaping von Protokolldaten

Vorgabe: Protokolleinträge, die Daten aus nicht vertrauenswürdigen Quellen enthalten, DÜRFEN NICHT Code enthalten, der in einem für die Protokollanzeige vorgesehenen Programm zur ungewollten Ausführung gebracht werden kann.
Hinweise: Weiterhin MUSS sichergestellt sein, dass Protokolleinträge, die nicht vertrauenswürdige Daten beinhalten, nicht in-line gefälschte Protokolleinträge enthalten können, die letztlich im Protokoll für echte Protokolleinträge gehalten werden könnten („Log Injection“). Beide Problematiken MÜSSEN durch entsprechendes sicheres Escaping von Sonder- und Steuerzeichen umgesetzt werden.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.13.16 Zugriffsbeschränkung für Protokolldaten

Vorgabe: Zugriff auf Protokolle DARF NUR für vorgesehene, autorisierte Rollen- bzw. Gruppenbenutzer mit den jeweils minimal benötigten Rechten erfolgen.
Hinweise: Dies lässt sich beispielsweise einfach mittels gängigen Dateirechten durchsetzen.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.13.17 Keine vertraulichen Daten in Protokolleinträgen

Vorgabe: Protokolleinträge DÜRFEN NICHT vertrauliche Daten, wie Systemdetails, Sitzungs-IDs oder erweiterte Kontoinformationen enthalten.
Quelle(n): OWASP-SCP

2.13.18 Protokollanalyse

Vorgabe: Es SOLLTE sichergestellt sein, dass ein Mechanismus zur Auswertung aller Protokolle existiert. Andernfalls MÜSSEN Protokolle in Formaten vorliegen, die mit Standardwerkzeugen ausgewertet werden können (i. d. R. in Plaintext).
Hinweise: Es SOLLTE hierfür eine Anbindung an ein zentrales Protokollierungssystem erfolgen. Dieses und die Authentifizierung hiergegen SOLLTE konfigurierbar sein.
Quelle(n): OWASP-SCP

2.13.19 Checkliste Logging

Vorgabe: Mindestens die folgenden Events MÜSSEN geloggt werden: Fehlschlagen von Eingabedatenvalidierungen; Authentifizierungsversuche, insbesondere die fehlgeschlagenen; Fehlgeschlagene Zugriffsberechtigungsprüfungen; Sämtliche möglichen Manipulationsversuche, einschließlich unerwarteter Datentypen und Veränderung von Zustandsdaten (Erkennung über Prüfsummen); Zugriffsversuche mit ungültigen oder abgelaufenen Session-IDs oder Tokens; Sämtliche System Exceptions; Sämtliche Ereignisse administrativer Funktionen, einschließlich Änderungen an der Sicherheitskonfiguration; Sämtliche TLS-Verbindungsfehler im Backend; Sämtliche Fehler aus kryptographischen Modulen

Quelle(n): OWASP-SCP; BSI-WEB-AN

2.13.20 Verwendung kryptographischer Hash-Funktion zur Validierung von Protokolleinträgen

Vorgabe: Protokolleinträge SOLLTEN mit kryptographischen Hashes abgelegt werden, um eine unbefugte Manipulation von diesen zu vermeiden bzw. erkennen zu können.

Hinweise: Der kryptographische Hash MUSS über den vollständigen Protokolleintrag einschließlich Metadaten gebildet werden. Es SOLLTE ein geheimer Schlüssel / Salt oder ein Hash Chain-Verfahren verwendet werden.

Quelle(n): OWASP-SCP

2.13.21 Einhaltung von Datenschutzvorgaben bei der Protokollierung

Vorgabe: Es DÜRFEN KEINE nicht unbedingt notwendigen Daten protokolliert werden. Dies beinhaltet beispielsweise Sitzungsidentifikatoren, personenbezogene oder andere vertrauliche Informationen, z. B. Berechtigungsnachweise.

Hinweise: Konkret werden Datenschutzvorgaben im Kapitel „Datenschutz“ behandelt. Im Zweifelsfall MUSS der Datenschutzbeauftragte eingebunden werden.

Quelle(n): BSI-WEB-AN

2.14 Datensicherheit & Information Disclosure

Die folgenden Vorgaben dienen dem grundsätzlichen Schutz von Daten und Informationen, sowie der Vermeidung von Datenlecks („Information Disclosure“).

2.14.1 Vermeidung von Race-Conditions

Vorgabe: Mögliche Race-Conditions MÜSSEN identifiziert werden und durch Unterbindung von synchronem Zugriff auf entsprechende Ressourcen, Locking oder einen Synchronisationsmechanismus ausgeschlossen werden.
Hinweise: Im Java-Kontext KANN dies beispielsweise durch Verwendung von Klassen aus „java.util.concurrent.locks“ erfolgen. Deadlocks MÜSSEN hierbei ausgeschlossen werden.
Quelle(n): OWASP-SCP

2.14.2 Vermeidung von gleichzeitigem Zugriff auf mehrfach genutzte Ressourcen

Vorgabe: Mehrfach genutzte Ressourcen MÜSSEN vor unzulässigem synchronen Zugriff geschützt werden.
Hinweise: Im Java-Kontext KANN dies beispielsweise durch Verwendung des „synchronized“-Keywords erfolgen.
Quelle(n): OWASP-SCP

2.14.3 Enumerierung erlaubter und Deaktivierung nicht benötigter HTTP-Anfragemethoden

Vorgabe: Es DÜRFEN NUR die von der Anwendung benötigten HTTP-Methoden (nach dem Minimalprinzip) erlaubt sein, wie etwa GET und POST. Nicht benötigte HTTP-Methoden, wie ggf. WebDAV-Erweiterungen, MÜSSEN (i. d. R. webserverseitig) deaktiviert werden. Insbesondere MÜSSEN die TRACE- und DEBUG-Methoden deaktiviert werden, sofern diese nicht explizit benötigt werden und dies begründet wird.
Hinweise: Bei HTTP Anfragen mit verbotener HTTP-Methode SOLLTE die Anfrage geloggt und danach gedroppt werden.
Quelle(n): BSI-WEB-AN; OWASP-SCP

2.14.4 Keine vertraulichen Informationen in HTTP GET-Anfrageparametern

Vorgabe: HTTP GET-Anfragen DÜRFEN NICHT mit vertraulichen Daten parametrisiert werden.
Hinweise: Dies DARF NICHT geschehen, da u. a. standardmäßig alle HTTP GET-Parameter in Zugriffsprotokolle geschrieben werden und somit ggf. für Unbefugte einsehbar sein können.
Quelle(n): OWASP-SCP

2.14.5 Verwendung von POST für einen Zustand ändernde Anfragen

Vorgabe: Für alle Anfragen, die eine Aktion auslösen sollen, die über einen einfachen lesenden Zugriff hinausgeht (Erstellen, Ändern oder Löschen von Datensätzen) MÜSSEN POST-Anfragen verwendet werden.

Hinweise: Serverseitig MUSS zudem überprüft werden, ob die jeweilige Anfrage auch unter Verwendung des erwarteten HTTP POST-Verbs eingegangen ist – falls nicht, MUSS die vorgesehene Aktion abgebrochen werden. Bei REST-Anfragen MÜSSEN der jeweiligen Anfrage zugehörige HTTP-Verben entsprechend der Spezifikation verwendet werden.

Quelle(n): BSI-WEB-MBP

2.14.6 Keine Weiterleitung zu benutzerdefinierten URLs

Vorgabe: Benutzerdefinierte Daten DÜRFEN NICHT ungeprüft an dynamische Weiterleitungsfunktionen übergeben werden.

Hinweise: Falls bei dynamischen Weiterleitungen benutzerdefinierte Daten genutzt werden müssen, SOLLTE die Weiterleitungsfunktion lediglich validierte, relative Pfade / URLs akzeptieren.

Quelle(n): OWASP-SCP

2.14.7 Character-Encoding für alle Verbindungen definieren

Vorgabe: Für alle Verbindungen MUSS das verwendete Character-Encoding richtig spezifiziert und i. d. R. mit angegeben werden.

Hinweise: Im HTTP-Kontext SOLLTE dies über korrekte Verwendung des „Content-Type“-Headers umgesetzt werden.

Quelle(n): OWASP-SCP

2.14.8 Kein Including benutzerdefinierter Daten

Vorgabe: Es MUSS sichergestellt werden, dass benutzerdefinierte Eingaben nicht direkt dynamische Include-Funktionen parametrisieren.

Hinweise: Insbesondere MUSS sichergestellt werden, dass nicht beliebige Dateien interpretiert und ausgeführt werden können, für die dies nicht vorgesehen ist (Local / Remote File Inclusion).

Quelle(n): OWASP-SCP

2.14.9 Keine Referenzierung von Dateien oder Verzeichnissen über Pfadnamen

Vorgabe: Dateien oder Verzeichnisse DÜRFEN NICHT über relative oder absolute Datei- oder Verzeichnisnamen referenziert werden.

Hinweise: Hierzu SOLLTEN Download-Funktionen stets mit erlaubten Datei-IDs parametrisiert und eine Autorisierung überprüft werden. Eine Parametrisierung mit relativen oder absoluten serverseitigen Dateipfaden SOLLTE NICHT erfolgen.

Quelle(n): OWASP-SCP

2.14.10 Schutz temporärer Daten und Caches

Vorgabe: Alle gecachten und temporär angelegten Daten (z. B. auch im Memory-Cache usw.) mit vertraulichem Inhalt MÜSSEN vor unbefugtem Zugriff geschützt werden und MÜSSEN unverzüglich gelöscht werden, sobald sie nicht länger benötigt werden.

Hinweise: Speziell für die Ablage temporärer Dateien durch Intranet-/Web-Applikationen bedeutet dies: Ein Zugriff auf temporäre Dateien (auch für einen minimalen Zeitraum) über den Webserver DARF NICHT erfolgen. Dies gilt ebenso für lokale Benutzeraccounts im lokalen Verzeichnisbaum.

Quelle(n): OWASP-SCP; TSS-WEB

2.14.11 Kein clientseitiges Caching vertraulicher Daten

Vorgabe: Serverantworten, die ggf. vertrauliche Daten enthalten, DÜRFEN NICHT gecachet bzw. zwischengespeichert werden.

Hinweise: Dies SOLLTE im HTTP-Kontext beispielsweise durch sinnvolles Setzen des „Cache-Control“-Headers („no-store“) umgesetzt werden und KANN gemeinsam mit dem HTTP-Header „Pragma“ („no-cache“) umgesetzt werden (letzteres ist weniger effektiv, jedoch abwärtskompatibel zu HTTP/1.0). Für Frontend-Frameworks SOLLTE sichergestellt werden, dass vertrauliche/personenbezogene Daten nach Abschluss des jeweiligen Vorgangs aus dem Model bereinigt werden.

Quelle(n): OWASP-SCP

2.14.12 Sicheres Löschen nicht mehr benötigter Daten

Vorgabe: Die Anwendung SOLLTE das Löschen nicht mehr benötigter vertraulicher Daten, wie personenbezogener Daten oder bestimmter buchhalterischer Daten, ermöglichen.

Hinweise: Hierfür SOLLTEN insbesondere vertrauliche Daten, sofern jeweils möglich, über Meta-Informationen zur minimalen und maximalen Lebensdauer verfügen.

Quelle(n): OWASP-SCP

2.14.13 Minimalitätsprinzip für Informationen

Vorgabe: Bereitgestellte Informationen, insbesondere auch Fehlermeldungen, DÜRFEN NUR die minimal für den jeweiligen Anwendungszweck des Anwenders benötigten Informationen enthalten. Für Benutzer nicht relevante Attribute von Datenobjekten SOLLTEN NICHT angezeigt werden können.

Hinweise: Insbesondere DÜRFEN NICHT bei Fehlermeldungen zu Anmeldungen Hinweise zu den falschen Authentisierungskomponenten ausgegeben werden. Weiterhin DÜRFEN NICHT interne Systeminformationen oder Versionsnummern, die nicht explizit benötigt werden, ausgegeben werden.

Quelle(n): BSI-WEB-MBP

2.14.14 Keine internen Informationen in Kommentaren

Vorgabe: Clientseitig einsehbare Code-Dateien DÜRFEN NICHT Kommentare mit internen Informationen (wie Backend-Systeme, API-Keys, interne Ticket-Informationen usw.) beinhalten.

Hinweise: Dies MUSS z. B. für Views / HTML, CSS, JS, XML-Dateien beachtet werden.

Quelle(n): OWASP-SCP; BSI-WEB-AN; BSI-WEB-MBP

2.14.15 Vermeidung von Fingerprinting

Vorgabe: Sogenanntes Fingerprinting der Anwendung SOLLTE durch Vermeidung der Ausgabe von Versionsinformationen und Umbenennen von typischen Merkmalen wie Meta-Tags, (Form-)Variablennamen usw. unterbunden werden.

Quelle(n): BSI-WEB-MBP

2.14.16 Keine Ausgabe absoluter Systempfade

Vorgabe: Absolute Systempfade von Dateien und / oder Verzeichnissen der Anwendung oder des Systems DÜRFEN NICHT an Anwender ausgegeben werden.

Hinweise: Idealerweise SOLLTEN Dateipfadinformationen, sofern nicht unbedingt benötigt, gar nicht ausgegeben werden. Falls doch benötigt, MÜSSEN relative Pfadangaben verwendet werden – also der Teil des Pfades ab dem Anwendungsverzeichnispfad.

Quelle(n): OWASP-SCP

2.14.17 Keine Übermittlung vertraulicher Informationen in Links auf Drittsysteme

Vorgabe: Bei Verlinkung von Inhalten auf Drittsystemen MUSS sichergestellt werden, dass bei Aufruf keine vertraulichen Informationen z. B. über den HTTP Referrer-Header mit übergeben werden.

Hinweise: Hierfür MUSS grundsätzlich sichergestellt werden, dass Webanwendungen keine URLs verwenden, die vertrauliche Informationen beinhalten. Der Security-Header „Referrer-Policy“ (z. B. mit dem restriktivsten Wert „no-referrer“) MUSS verwendet werden. Da dieser nicht durch alle Browser unterstützt wird, SOLLTEN weiterhin Links zu externen Inhalten mit „rel="noreferrer"“ und „rel="noopener"“ eingerichtet werden.

Quelle(n): OWASP-SCP

2.14.18 Integritätsprüfung bei Einbindung externer Inhalte

Vorgabe: Werden externe Skriptinhalte eingebunden, SOLLTE deren Integrität daraufhin überprüft werden, ob der Inhalt einem bekannten erwarteten Inhalt entspricht. Hierfür SOLLTEN sichere kryptographische Prüfsummen verwendet werden.

Hinweise: Bei Einbindung von externen Skript- oder Stylesheet-Inhalten in Webapplikationen SOLLTE Subresource Integrity (SRI) entsprechend der Dokumentation (https://www.w3.org/TR/SRI/) mit mindestens SHA256-Hashes verwendet werden.

Quelle(n): VHV-RSE

2.14.19 Sicherer Transport interner Daten

Vorgabe: Die Übertragung interner Daten über unsichere Netze (z. B. dem Internet) MUSS verschlüsselt (mittels TLS/HTTPS) erfolgen (auch bei Intranet bzw. internen Übertragungen SOLLTE dies der Fall sein).

Quelle(n): TSS-WEB

2.14.20 Weiterleitung auf HTTPS

Vorgabe: Erfordert eine Anwendung die Verwendung von HTTPS, MÜSSEN alle Zugriffe mittels HTTP auf den entsprechenden HTTPS-Kontext weitergeleitet werden. Hierzu SOLLTE eine permanente Weiterleitung verwendet werden.

Hinweise: Für eine permanente Weiterleitung SOLLTEN die HTTP Status Codes 301 bzw. 308 verwendet werden.

Quelle(n): TSS-WEB

2.14.21 Präventive Weiterleitung auf HTTPS

Vorgabe: Erfordert eine Anwendung die Verwendung von HTTPS, SOLLTE der „Strict-Transport-Security“-(HSTS)-Header verwendet werden, um Man-in-the-Middle-(MitM)-Angriffe (ohne gültige x.509-Zertifikate) weitestgehend auszuschließen.

Hinweise: Für den Header SOLLTE eine ausreichend lange Ablaufzeit (z. B. ein Jahr) wie folgt konfiguriert werden. Wird HTTPS für alle Subdomains verwendet, SOLLTE ebenfalls das „includeSubDomains“-Attribut verwendet werden. Idealerweise SOLLTE nur für extern erreichbare Domains HSTS Preloading verwendet werden (durch Setzen des „preload“-Attributs und Eintragung der Domain auf https://hstspreload.org). Die o. g. Umsetzungsdetails KÖNNEN durch Setzen des Headers wie folgt implementiert werden: „Strict-Transport-Security: max-age=31536000; includeSubDomains; preload“.

Quelle(n): VHV-RSE

2.14.22 Kein Transport vertraulicher Daten über Metadaten

Vorgabe: Die Übertragung vertraulicher Daten MUSS über den (HTTP) Request Body (z. B. mittels HTTP-POST-Methode) erfolgen und DARF NICHT innerhalb der URL (Ausnahme: Objekt-IDs) erfolgen.

Hinweise: Metadaten, die anfällig für Disclosure oder unbeabsichtigtes Logging vertraulicher Daten sind, sind bspw. der HTTP „Referer“-Header.

Quelle(n): TSS-WEB

2.15 Speicher-, Prozess- und Ressourcenzugriffs-Management

Bei Verwendung von Programmiersprachen ohne automatisches Speichermanagement, die manuelle Allokierung und Freigabe von Arbeitsspeicher erfordern, sind die folgenden Vorgaben zu beachten (C/C++). Bei Verwendung von Skript-/Hochsprachen wie Java, ASP.net, Python usw. sind die in diesem Kapitel enthaltenen Vorgaben i. d. R. nicht relevant.

2.15.1 Überprüfung der Buffer-Größen

Vorgabe: Es MUSS überprüft werden, dass allokierte Speicherbereiche groß genug für den jeweilig vorgesehenen Verwendungszweck sind.

Hinweise: Es SOLLTEN Tools zur statischen Codeanalyse verwendet werden, um insbesondere die Größe der Speicherbereiche zu überprüfen.

Quelle(n): OWASP-SCP

2.15.2 Berücksichtigung des Speicherplatzbedarfs für den Null-Terminator

Vorgabe: Bei Verwendung von Funktionen, bei denen die Menge der zu kopierenden Bytes angegeben werden muss, wie beispielsweise „strncpy()“, MUSS berücksichtigt werden, dass wenn die Größe des Zielspeicherbereichs der Länge eines zu kopierenden Strings entspricht, dieser ggf. nicht null-terminiert werden kann.

Hinweise: Es SOLLTEN daher stets Buffer allokiert werden, die Platz für einen Null-Terminator berücksichtigen und die fortlaufende Verwendung von Null-Terminatoren sichergestellt werden.

Quelle(n): OWASP-SCP

2.15.3 Buffer-Grenzen berücksichtigen

Vorgabe: Insbesondere bei Funktionsaufrufen in Schleifen MUSS sichergestellt werden, dass keine Gefahr besteht, außerhalb vorgesehener Speicherbereiche zu lesen oder zu schreiben.

Hinweise: Unter UNIX Betriebssystemen ist es möglich mit Programmen wie „valgrind“ zu überprüfen, ob Speicherlecks auftreten.

Quelle(n): OWASP-SCP

2.15.4 Freigabe von allokierten Ressourcen

Vorgabe: Allokierte Ressourcen (wie Datei- oder Stream-Handles) MÜSSEN nach Verwendung (unabhängig von ggf. verwendeten Garbage Collection-Mechanismen) wieder freigegeben werden.

Hinweise: Unter Java KANN z. B. jeweils close() auf im Kontext einer Datenbankverbindung allokierte Objekte verwendet werden.

Quelle(n): OWASP-SCP

2.15.5 Freigabe von allokierten Speicherbereichen

Vorgabe: Allokierte Speicherbereiche MÜSSEN nach Verwendung (vor Funktionsrückgabe und bei allen Exit-Punkten) wieder freigegeben werden (unabhängig von ggf. verwendeten Garbage Collection-Mechanismen).

Hinweise: Unter C KANN hierfür z. B., sofern dies nicht durch Nutzung einer funktionalen Bibliothek umgesetzt ist, free() verwendet werden.

Quelle(n): OWASP-SCP

2.15.6 „Least Privilege“-Prinzip für Prozesse

Vorgabe: Jeder Prozess DARF NUR die notwendigen Berechtigungen auf Ressourcen (Dateisystem, Datenbank etc.) besitzen („Least Privilege“-Prinzip).
Quelle(n): TSS-WEB

2.16 Datenbanken

Für Datenverarbeitung und -ablage in Datenbanken sind die nachfolgenden Vorgaben umzusetzen, um datenbankspezifische Sicherheitslücken auszuschließen und die Anwendungssicherheit zu erhöhen.

2.16.1 Verwendung von parametrisierten Datenbankanfragen (Prepared Statements)

Vorgabe: Für Datenbankabfragen MÜSSEN codeseitig Befehle und Daten durch Platzhalter voneinander getrennt werden. Hierzu MÜSSEN Prepared Statements oder äquivalente Kapselungstechnologien (wie ORM [Object Relational Mapping]) verwendet werden, die sicheres Escaping von Parametern durchführen.
Hinweise: Hierbei MÜSSEN Standardmethoden der verwendeten Programmiersprache oder des verwendeten Frameworks verwendet werden, die ein sicheres Escaping der Datenkomponenten garantieren. Diese Vorgabe ist bei ausschließlicher Verwendung von einer Abstraktionsschicht z. B. Spring JPA für Datenbankzugriffe umgesetzt, wenn nicht native SQL-Queries abgesetzt werden.
Quelle(n): OWASP-SCP; TSS-WEB; BSI-WEB-AN; BSI-WEB-MBP

2.16.2 Sichere Enkodierung für Backend-Zugriffe

Vorgabe: Sollte keine API zur Parametrisierung des Backend-Zugriffs zur Verfügung stehen oder deren Einsatz nicht möglich sein, MÜSSEN Parameter mit einer geeigneten API enkodiert werden (z. B. SQL-Encoding).
Hinweise: In Ruby KANN z. B. Dude.sanitize für SQL-Parameter verwendet werden.
Quelle(n): TSS-WEB

2.16.3 Eingabedatenvalidierung und Ausgabekodierung bei Datenbankzugriff

Vorgabe: Insbesondere beim Datenbankzugriff MUSS eine Validierung der Eingabedaten sowie eine sichere Kodierung der Ausgabedaten erfolgen.
Hinweise: Hierbei MÜSSEN Sonder- und Steuerzeichen berücksichtigt werden. Schlägt die Validierung bzw. Kodierung fehl, MUSS die Ausführung des Datenbankbefehls bzw. die Ausgabe der Daten abgebrochen werden.
Quelle(n): OWASP-SCP

2.16.4 Verwendung stark typisierter Variablen

Vorgabe: Insbesondere die bei der Erstellung von Datenbankanfragen verwendeten Variablen MÜSSEN stark typisiert sein, d. h. es MÜSSEN restriktive Variablentypen mit definiertem Wertebereich verwendet werden („strong typing“).
Hinweise: Auch in nicht stark typisierten Sprachen wie PHP SOLLTE so viel wie möglich das Typsystem genutzt werden.
Quelle(n): OWASP-SCP

2.16.5 Verwendung von Stored Procedures

Vorgabe: Es SOLLTEN Stored Procedures verwendet werden, um den Datenbankzugriff zu abstrahieren und Rechte auf die zugrundeliegenden Datenbanktabellen einschränken zu können.
Hinweise: Äquivalent sichere Abstraktionsschichten, wie Prepared Statements oder Object Relational Mapping (ORM) KÖNNEN alternativ verwendet werden.
Quelle(n): OWASP-SCP

2.16.6 Datenbankverbindung zeitnah schließen

Vorgabe: Nicht mehr benötigte Datenbankverbindungen SOLLTEN direkt anschließend codeseitig geschlossen werden.
Hinweise: Bspw. SOLLTE bei C#-Anwendungen .Close() auf das SqlConnection-Objekt erfolgen, oder dieses in einem using () { … } -Block verwendet werden, so dass dies automatisch geschieht.
Quelle(n): OWASP-SCP

2.16.7 Verwendung von Datenbankverbindungen mit unterschiedlichen Berechtigungsabstufungen

Vorgabe: Es SOLLTEN Datenbankzugangsdaten mit unterschiedlichen Rechteabstufungen existieren und entsprechend dem jeweiligen Kontext verwendet werden (z. B. Admin, Anwender, Gast).
Hinweise: Es MÜSSEN User ausschließlich auf die Datenbanken zugreifen dürfen, die sie benötigen.
Quelle(n): OWASP-SCP

2.17 Kryptographie

Unter dieses Kapitel fallen Vorgaben zur Kryptographie. Geregelt wird, was zu verschlüsseln ist und teilweise wie eine Verschlüsselung zu erfolgen hat. Zu dieser Thematik muss beachtet werden, dass hierzu in der VAV bereits eine dedizierte Kryptographierichtlinie (Arbeitsrichtlinie Kryptographie) existiert, die durch die vorliegende Richtlinie nicht abgelöst wird. Dort enthaltene Vorgaben sind somit ebenfalls, soweit zutreffend, zu berücksichtigen bzw. zur Konkretisierung der hier enthaltenen Vorgaben heranzuziehen.

2.17.1 Kryptographische Funktionen in vertrauenswürdigen Systemen

Vorgabe: Es MUSS sichergestellt werden, dass kryptographische Funktionen, die geheime Informationen vor Zugriff durch Anwender schützen, von vertrauenswürdigen Systemen durchgeführt werden (i. d. R. serverseitig).
Hinweise: Ebenfalls MUSS Passwort-Hashing in einem kontrollierbaren, vertrauenswürdigen Kontext (serverseitig) erfolgen.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.17.2 Verwendung sicherer kryptographischer Funktionen

Vorgabe: Es DÜRFEN nur ausgereifte und standardisierte kryptographische Algorithmen, Betriebsmodi, Schlüsselstärken, SSL/TLS-Ciphers und Implementierungen eingesetzt werden.
Hinweise: Dies ist entsprechend der aktuell gültigen Arbeitsrichtline Kryptographie umzusetzen. Die konkret verwendeten Verfahren, Ciphers usw. SOLLTEN konfigurierbar sein.
Quelle(n): TSS-WEB

2.17.3 Sichere Fehlerbehandlung bei kryptographischen Funktionen

Vorgabe: Bei Auftreten von Fehlern in kryptographischen Funktionen, wie Verschlüsselung oder Hashing, MUSS sichergestellt werden, dass keine vertraulichen Informationen ausgegeben werden, eine etwaige Weiterverarbeitung – insbesondere bei der Authentifizierung, Passwortüberprüfung usw. – sicher abgebrochen wird und ein Abfangen des Fehlers in einen sicheren Zustand führt.
Hinweise: Diese Fehlerfälle MÜSSEN in den Tests abgedeckt und geprüft werden, sowie durch beispielsweise Fuzzing o. Ä. gesondert gehärtet werden.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.17.4 Sichere Zufallswerte

Vorgabe: Sämtliche Zufallszahlen, sowie zufällig generierte Dateinamen, GUIDs, Zeichenketten usw. MÜSSEN über Zufallsgeneratoren eines Standardkryptographiemoduls der verwendeten Programmiersprache oder des verwendeten Frameworks generiert werden, wenn ein Erraten der generierten Werte ausgeschlossen werden soll.

Hinweise: In PHP sollte beispielsweise die Funktion „random_bytes“ und niemals die Funktion „mt_rand“ verwendet werden.

Quelle(n): OWASP-SCP

2.17.5 Protokollierung von Fehlern in kryptographischen Funktionen

Vorgabe: Fehler von kryptographischen Modulen MÜSSEN protokolliert werden.

Quelle(n): BSI-WEB-AN

2.17.6 Transportverschlüsselung für Daten

Vorgabe: Der Transport von Daten zwischen Systemen DARF NICHT unverschlüsselt erfolgen, um eine Manipulation der Daten zu erschweren.

Hinweise: Dies bedeutet, dass eine Verschlüsselung der Transportschicht (i. d. R. mit TLS) erfolgen MUSS. Zusätzliche Verschlüsselung von beispielsweise übertragenen Dateien KANN zudem in bestimmten Anwendungsfällen sinnvoll sein und DARF als zusätzlicher Schutz verwendet werden.

Quelle(n): OWASP-SCP; BSI-ITGS-B5.27

2.17.7 TLS für authentifizierte Verbindungen

Vorgabe: Für Verbindungen, die Authentifizierungsdaten beinhalten, authentifiziert sind oder vertrauliche Informationen enthalten, MUSS eine Transportschichtverschlüsselung (i. d. R. mit TLS) erfolgen.

Hinweise: Um MitM-Angriffe sinnvoll zu unterbinden, MUSS zudem natürlich auch überprüft werden, ob die Gegenstelle über ein Zertifikat verfügt, dem – ggf. via dessen CA – die Anwendung vertraut.

Quelle(n): OWASP-SCP

2.17.8 TLS für vertrauliche Verbindungen zu Drittsystemen

Vorgabe: Für Verbindungen zu weiteren Systemen, bei denen vertrauliche Informationen übermittelt oder sensible Funktionen aufgerufen werden, MUSS eine Transportschichtverschlüsselung (i. d. R. mit TLS) erfolgen.
Hinweise: Um MitM-Angriffe sinnvoll zu unterbinden, MUSS zudem natürlich auch überprüft werden, ob die Gegenstelle über ein Zertifikat verfügt, dem – ggf. via dessen CA – die Anwendung vertraut.
Quelle(n): OWASP-SCP

2.17.9 Zentrale, standardisiert konfigurierte TLS-Implementierung

Vorgabe: Es SOLLTE eine zentrale und einem vorgegebenen Standard entsprechend konfigurierte TLS-Implementierung verwendet werden.
Hinweise: Dies ist entsprechend der aktuell gültigen Arbeitsrichtlinie Kryptographie umzusetzen. Die konkret verwendeten Verfahren, Ciphers usw. SOLLTEN konfigurierbar sein.
Quelle(n): OWASP-SCP

2.17.10 Sichere TLS-Konfiguration

Vorgabe: Ein HTTPS-Server MUSS aktuelle Cipher und Protokolle mit sehr hoher Sicherheit unterstützen. Unsichere HTTPS-Cipher oder Protokolle (z. B. SSLv2, SSLv3 oder RC4) MÜSSEN deaktiviert werden.
Hinweise: Generell SOLLTEN lediglich so wenig Ciphern und Protokolle wie unbedingt nötig unterstützt werden.
Quelle(n): TSS-WEB

2.17.11 Perfect Forward Secrecy

Vorgabe: Perfect Forward Secrecy (PFS) MUSS auf allen HTTPS-Servern unterstützt werden.
Quelle(n): TSS-WEB

2.17.12 Keine unsichere Verbindung bei fehlgeschlagener TLS-Verbindung

Vorgabe: Für den Fall, dass eine sichere TLS-Verbindung nicht aufgebaut werden kann und fehlschlägt, DARF NICHT eine unsichere Verbindung verwendet werden.

Hinweise: Für Webanwendungen SOLLTE hierzu der „Strict-Transport-Security“-Header verwendet werden, um sicher verschlüsselte Verbindungen zu forcieren. Auch für verschlüsselte Verbindungen aus dem Backend DARF NICHT die Zertifikatsprüfung übergangen oder ignoriert werden. Bei ggf. noch vorhandenen Legacy-Anwendungen, die SSLv3 unterstützen, MUSS weiterhin der TLS_FALLBACK_SCSV-Mechanismus implementiert werden.

Quelle(n): OWASP-SCP; BSI-WEB-AN

2.17.13 Sichere Ablage von Schlüsseln und Authentifizierungsdaten

Vorgabe: Vertrauliche kryptographische Schlüssel (= geheime Schlüssel bei symmetrischer oder private Schlüssel bei asymmetrischer Kryptographie sowie Credentials technischer Benutzer, API-Keys, Passwörter usw.) MÜSSEN getrennt vom Programmcode (z. B. in einem KeyStore) abgelegt und vor unberechtigtem Zugriff geschützt werden (restriktive Zugriffsrechte, passwortgeschützter KeyStore).

Hinweise: Schlüssel dürfen nicht im Programmcode hinterlegt sein, sondern müssen bei Installation oder zur Laufzeit erzeugt werden.

Quelle(n): OWASP-SCP; TSS-WEB

2.18 X.509-Zertifikate („SSL/TLS-Zertifikate“)

Die nachfolgenden Vorgaben in diesem Kapitel beziehen sich lediglich auf Anforderungen an zu nutzende Zertifikate bei Einsatz von TLS. Weitere zu beachtende Aspekte hinsichtlich sicherer Transportverschlüsselung, sowie Konfiguration dieser enthalten die Kapitel „Kryptographie“ und „Konfiguration“. Zu dieser Thematik muss ebenfalls beachtet werden, dass hierzu in der VAV bereits eine dedizierte Kryptographierichtlinie existiert, die durch die vorliegende Richtlinie nicht abgelöst wird. Dort enthaltene Vorgaben sind somit ebenfalls, soweit zutreffend, zu berücksichtigen.

2.18.1 Einsatz gültiger Zertifikate

Vorgabe: Eingesetzte (TLS-)Zertifikate MÜSSEN gültig sein. Sie MÜSSEN weiterhin den korrekten Domainnamen bzw. FQDN verwenden, DÜRFEN NICHT abgelaufen sein und SOLLTEN die benötigten Intermediate-Zertifikate beinhalten.

Hinweise: Es SOLLTEN Zertifikate mit Schlüssellängen von mindestens 4096 Bit verwendet werden bzw. die jeweils aktuell gültige Kryptographierichtlinie der VAV umsetzen.

Quelle(n): OWASP-SCP; TSS-WEB

2.18.2 X.509 / TLS: Schlüssellängen

Vorgabe: Alle X.509-Zertifikate MÜSSEN eine Schlüssellänge von mindestens 2048 Bit besitzen.
Hinweise: Zertifikate SOLLTEN eine Schlüssellänge von mindestens 4096 Bit besitzen.
Quelle(n): TSS-WEB

2.18.3 X.509 / TLS: Anforderungen für Anwendungen mit hohem Schutzbedarf

Vorgabe: Nach extern (Internet / Dritte) exponierte Applikationen ab der Schutzklasse „Hoch“: MÜSSEN eine Schlüssellänge von 4096 Bit besitzen; MÜSSEN Extended Validation (EV) Zertifikate verwenden; MÜSSEN Certificate Transparency (CT) verwenden; MÜSSEN einen Certification Authority Authorization (CAA) Eintrag im DNS setzen; MÜSSEN eigene Zertifikate verwenden (keine Wildcard-Zertifikate etc.)
Quelle(n): TSS-WEB; VHV-RSE

2.18.4 Validierung von TLS-Zertifikaten

Vorgabe: Bei Verwendung von TLS-Zertifikaten MUSS geprüft werden, ob die „Chain of Trust“ zu einer vertrauenswürdigen CA zurückverfolgt werden kann und ob jedes Serverzertifikat gültig ist. Weiterhin MUSS geprüft werden, ob das jeweilige Zertifikat nicht abgelaufen ist und zum aufgerufenen Domainnamen passt.
Hinweise: In Java KANN dies beispielsweise via „java.security.cert“ durch Überprüfung mit „.checkValidity()“ auf alle Serverzertifikate der Verbindung („.getServerCertificates()“) erfolgen.
Quelle(n): BSI-WEB-AN

2.19 Clientseitige Sicherheit

Für die Sicherheit des Clients und der dort erhobenen, verarbeiteten und seltener auch gespeicherten Daten sind die folgenden Vorgaben zu beachten.

2.19.1 Keine clientseitige Speicherung vertraulicher Daten

Vorgabe: Vertrauliche bzw. streng vertrauliche Daten, wie Passwörter, Verbindungs-Strings, kryptographische Schlüssel, personenbezogene Daten usw. DÜRFEN NICHT unverschlüsselt clientseitig gespeichert werden.
Hinweise: Dies schließt Einbettung in unsichere, clientseitig verwendete Dateiformate wie Adobe Flash, ASP.NET View State oder kompilierten Code ein. Für alle Formulare, die vertrauliche Daten enthalten, MUSS clientseitiges Caching unterbunden werden (inklusive AutoComplete-Funktionen [autocomplete="off"]). Für Frontend-Frameworks SOLLTE sichergestellt werden, dass vertrauliche/personenbezogene Daten nach Abschluss des jeweiligen Vorgangs aus dem Model bereinigt werden.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.19.2 Clientseitige Speicherung vertraulicher Daten

Vorgabe: Clientseitig (z. B. im Browser) DÜRFEN vertrauliche Daten nur dann abgelegt werden, wenn diese dem Benutzer selbst gehören (z. B. eigene personenbezogene Daten).
Quelle(n): TSS-WEB

2.19.3 Keine automatische Vervollständigung für Eingabefelder für vertrauliche Daten

Vorgabe: Für Eingabefelder, in die ggf. vertrauliche Daten eingegeben werden, DARF NICHT eine Autovervollständigung aktiviert sein.
Hinweise: Für Webanwendungen KANN dies über Setzen des Attributs „autocomplete“ auf den Wert „off“ und für Eingabefelder („input“) vom Typ („type“) „password“ realisiert werden.
Quelle(n): OWASP-SCP

2.19.4 Integritätsprüfung für Zustandsdaten

Vorgabe: Falls der Anwendungszustand clientseitig gespeichert werden soll, MUSS serverseitig eine Verschlüsselung und Integritätsüberprüfung der zugehörigen Datenobjekte erfolgen.
Hinweise: Dies KANN z. B. über Cookies bzw. JWE-Token erfolgen, die serverseitig erstellt und zur Sicherstellung der Integrität signiert und verschlüsselt werden.
Quelle(n): OWASP-SCP

2.19.5 JavaScript: Sicherer Umgang mit JSON

Vorgabe: JSON-Code DARF ausschließlich mit einer sicheren JavaScript-API interpretiert werden.
Hinweise: Im Browser SOLLTE „JSON.parse()“ verwendet werden. Hier DARF NICHT „eval()“ verwendet werden.
Quelle(n): TSS-WEB

2.19.6 JavaScript: Sichere Content-Manipulation

Vorgabe: Anstelle der JavaScript-APIs mit „.innerHTML“ im Namen, SOLLTEN stets sichere APIs wie „.innerText“ und „.textContent“ genutzt werden, sofern nicht explizit HTML-Markup ausgegeben werden soll. Gleiches gilt auch für Webframeworks, die entsprechende APIs zur Verfügung stellen.
Quelle(n): TSS-WEB

2.19.7 JavaScript: Sichere Transportschicht für APIs

Vorgabe: HTTP-Security-Header, wie Strict Transport Security (HSTS) oder Content Security Policy (CSP) MÜSSEN (entsprechend der Spezifikation in Anhang) für von JavaScript genutzte APIs implementiert werden.
Quelle(n): TSS-WEB

2.19.8 Keine Offenlegung vertraulicher Geschäftslogik

Vorgabe: Clientseitig DARF KEINE vertrauliche Geschäftslogik (z. B. Berechnung von Tarifen, Generierung von IDs) offengelegt werden.
Quelle(n): TSS-WEB

2.19.9 Verwendung des „HttpOnly“-Attributs für Cookies

Vorgabe: Das „HttpOnly“-Attribut MUSS für alle Cookies verwendet werden, auf die nicht mittels JavaScript zugegriffen werden muss.
Quelle(n): BSI-WEB-AN

2.19.10 Verwendung des „secure“-Attributs für Cookies

Vorgabe: Das „secure“-Attribut MUSS für alle Cookies verwendet werden, die vertrauliche Informationen enthalten.
Quelle(n): BSI-WEB-AN

2.19.11 Checkliste Vermeidung von UI-Spoofing

Vorgabe: Um u. a. Angriffsszenarien via User Interface-Spoofing zu erschweren, MUSS die Authentizität der Anwendungsinhalte stets überprüfbar sein.
Hinweise: Dies ist je nach Anwendungstyp über passende Maßnahmen zu adressieren, für Internet / Intranetanwendungen gilt: Es SOLLTEN KEINE Pop-up-Fenster verwendet werden; Zertifikate MÜSSEN gültig sein (nicht abgelaufen, von valider CA signiert); Zertifikate SOLLTEN über Extended Validation (EV) verfügen; Abmeldungen oder erneute Authentisierungsaufforderungen MÜSSEN für Anwender nachvollziehbar sein und DÜRFEN NICHT ohne besonderen Grund (Timeouts, Aktion mit höheren Rechten) stattfinden; Werden aus der Anwendung E-Mails versendet, SOLLTEN diese stets dieselbe(n) Absenderadressen (FROM und REPLY-TO, falls verwendet) haben und SOLLTEN von derselben Domain stammen, über die Anwender die Anwendung erreichen.
Quelle(n): BSI-WEB-MBP

2.19.12 Verwendung signierter Werte

Vorgabe: Werden Werte von der Anwendung an einen Client transportiert, die dieser oder von welchen dieser eine Auswahl unverändert zurück an den Server transportieren muss, so MUSS der Server diese Werte auf Gültigkeit überprüfen. Die Gültigkeitsüberprüfung MUSS, wenn (z. B. aus Performancegründen) nicht serverseitig gegen eine Liste zulässiger Werte überprüft werden soll, anhand einer Überprüfung eines jeweils zu dem Wert zusätzlich erstellten Hashs erfolgen.

Hinweise: Der Hash MUSS unter Verwendung eines zusätzlichen Salts generiert werden. Der Salt DARF NICHT den Serverkontext verlassen. Der Hash MUSS unter Verwendung einer sicheren kryptographischen Funktion generiert werden (siehe Arbeitsrichtlinie Kryptographie).

Quelle(n): BSI-WEB-MBP

2.20 Webdienste und XML-Parser

Dieses Kapitel behandelt die Anforderungen an Webservices und zugehörige Technologien wie SOAP, REST und XML-Parser.

2.20.1 Kapselung interner Services für externen Zugriff

Vorgabe: Werden interne Webdienste von extern zugreifbar gemacht, so MUSS dies stets über eine Façade erfolgen, über die ausschließlich die erforderliche Funktionalität der internen Webdienste gekapselt und von extern zugreifbar ist.

Quelle(n): TSS-WEB

2.20.2 Validierung nicht vertrauenswürdiger XML-Daten

Vorgabe: XML-Daten aus nicht vertrauenswürdigen Quellen (z. B. empfangen von externen Webservices) MÜSSEN stets restriktiv auf Modelebene (z. B. Bean Validation) oder durch ein XML-Schema validiert werden. Das verwendete Schema SOLLTE hierzu so weit wie möglich eingeschränkt werden.

Hinweise: Beispiele zur Einschränkung des Schemas sind: Integer statt String-Datentyp; Min/Max-Einschränkung für numerische Datentypen; Eingeschränkte String-Typen.

Quelle(n): TSS-WEB; BSI-WEB-AN

2.20.3 Härtung von XML-Parsern für Validierung nicht vertrauenswürdiger XML-Daten

Vorgabe: XML-Parser, die Daten aus nicht vertrauenswürdiger Quelle parsen, MÜSSEN wie folgt gehärtet werden, um gängige XML-basierte Angriffe zu unterbinden: Setzen restriktiver Limits (z. B. im Hinblick auf Verschachtelungstiefe oder Dokumentgröße); Deaktivierung externer XML Entitäten („External XML Entities“).

Quelle(n): TSS-WEB

2.20.4 Authentifizierung bei REST-Services

Vorgabe: Die Authentifizierung bei REST-Services MUSS auf eine der folgenden Arten durchgeführt werden: Mittels generiertem, vollständig zufälligem Schlüssel mit ausreichender Entropie („API Key“), der durch den Client im HTTP-Header (z. B. als „X-Header“) übertragen wird (siehe Anforderungen in Kapitel „Authentifizierung am Backend“); Mittels OAuth-Protokoll; Mittels X.509 Client Zertifikaten; Mittels HTTP Basic Auth und der Verwendung von HTTPS.

Hinweise: Die Übertragung von API Keys in URLs hingegen DARF NICHT erfolgen.

Quelle(n): TSS-WEB

2.20.5 Sichere Übermittlung von Skriptinhalten

Vorgabe: Enthält die Antwort eines Webdienstes Skriptcode (z. B. JSON bzw. JavaScript), so MUSS sichergestellt werden, dass dieser nicht direkt ausführbar ist.

Hinweise: Falls zum Cross Origin Resource Sharing (CORS) ein JSONP-Ansatz verwendet wird, so MUSS besonders darauf geachtet werden, eine JavaScript Code Injection durch Validierung bzw. sicheres Escaping von Eingabedaten zu verhindern.

Quelle(n): TSS-WEB

2.20.6 CSRF-Schutz

Vorgabe: Für alle Webdienste (inkl. HTTP-basierter APIs), bei denen über eine authentifizierte Session ändernde Anfragen durchgeführt werden, MUSS ein CSRF-Schutz implementiert werden (vgl. Kapitel „Absicherung des Session-Managements“).

Quelle(n): TSS-WEB

2.20.7 Übermittlung vertraulicher Daten über WebSockets

Vorgabe: WebSockets, über die vertrauliche Daten übermittelt werden, MÜSSEN das „wss://“-Schema verwenden und eine Prüfung des Origin-Headers durchführen.

Quelle(n): TSS-WEB

2.20.8 Äquivalente Sicherheitsfunktionen für Webdienste

Vorgabe: Webdienste MÜSSEN dieselben Zugriffe stets auch mit denselben Sicherheitsvorgaben (im Hinblick auf Authentifizierung, Validierung etc.) durchführen, wie für das Web-GUI-Frontend. Hierfür SOLLTEN dieselben Sicherheitsfunktionen (Programmcode, APIs) verwendet werden.
Quelle(n): TSS-WEB

2.20.9 Verwendung authentisierter Verbindungen zu anderen Systemen

Vorgabe: Alle Verbindungen zu externen Systemen und Diensten MÜSSEN authentisiert sein, wenn diese interne Informationen oder sensible Funktionen betreffen.
Hinweise: Es SOLLTEN bei fehlender menschlicher Interaktion generierte Tokens genutzt werden und keine Passwörter.
Quelle(n): BSI-WEB-AN

2.21 Datenschutzvorgaben

Die folgenden Vorgaben basieren größtenteils auf rechtlichen Anforderungen zum Datenschutz, also dem Schutz insbesondere personenbezogener Daten.

2.21.1 Feststellen der Anwendbarkeit von Datenschutzvorgaben

Vorgabe: Es MUSS festgestellt werden, ob die Anwendung personenbezogene Daten verarbeitet. Falls ja, MÜSSEN die folgenden den Datenschutz betreffenden Vorgaben berücksichtigt werden (andernfalls nicht).
Hinweise: Die Begrifflichkeiten „Verarbeitung“ und „personenbezogene Daten“ sind im Art. 4 EU-DSGVO definiert. Für die Entscheidung SOLLTE im Zweifelsfall der Datenschutzbeauftragte hinzugezogen werden.
Quelle(n): DSG; EU-DSGVO

2.21.2 Verarbeitung besonderer Kategorien personenbezogener Daten

Vorgabe: Bei Verarbeitung personenbezogener Daten MUSS festgestellt werden, ob auch „besondere Kategorien personenbezogener Daten“ verarbeitet werden. Falls ja, so DARF NUR eine Verarbeitung entsprechend einer zuvor erfolgten Abstimmung mit dem Datenschutzbeauftragten erfolgen.

Hinweise: Besondere Kategorien personenbezogener Daten sind (vgl. Art. 4 Nr. 13-15 und Art. 9 Abs. 1 EU-DSGVO): Daten, aus denen die rassische oder ethnische Herkunft, politische Meinungen, religiöse oder weltanschauliche Überzeugungen oder die Gewerkschaftszugehörigkeit hervorgehen, genetische Daten, biometrische Daten zur eindeutigen Identifizierung einer natürlichen Person, Gesundheitsdaten und Daten zum Sexualleben oder zur sexuellen Orientierung. Eine getrennte Datenverarbeitung KANN beispielsweise durch separate Datenobjekte und -strukturen, wie auch eigene Datenbanken, Tabellen oder Verarbeitung auf besonders geschützten Backend-Systemen sein.

Quelle(n): DSG; EU-DSGVO

2.21.3 Datenschutzvorgabenrelevanz für Scoring

Vorgabe: Findet sog. Scoring statt (d. h., wenn Wahrscheinlichkeitswerte über zukünftiges Verhalten einer Person berechnet werden, die für Entscheidungen über Begründung, Durchführung oder Beendigung von Vertragsverhältnissen mit dieser Person herangezogen werden), dann MÜSSEN die nachfolgenden fünf Vorgaben zum Scoring berücksichtigt werden (andernfalls nicht).

Hinweise: Herrscht Unklarheit darüber, ob bestimmte Verfahren als Scoring zu werten sind, so MUSS zur Klärung der Datenschutzbeauftragte einbezogen werden.

Quelle(n): DSG; EU-DSGVO

2.21.4 Abstimmung mit Datenschutzbeauftragtem beim Scoring

Vorgabe: Findet Scoring statt, MÜSSEN Datenschutzvorschriften mit dem Datenschutzbeauftragten abgestimmt und eingehalten werden.

Hinweise: Die Verfahren sowie die Abstimmung mit dem Datenschutzbeauftragten SOLLTEN – bspw. im Sicherheits- oder Datenschutzkonzept – nachvollziehbar dokumentiert werden.

Quelle(n): EU-DSGVO; DSG

2.21.5 Verfahrensvorgaben für Scoring

Vorgabe: Findet Scoring statt, MUSS zur Berechnung des Wahrscheinlichkeitswerts ein wissenschaftlich anerkanntes mathematisch-statistisches Verfahrens verwendet werden und die verwendeten Daten MÜSSEN nachweisbar für die Berechnung der Wahrscheinlichkeit des bestimmten Verhaltens erheblich sein.

Quelle(n): DSG; EU-DSGVO

2.21.6 Keine ausschließliche Verwendung von Anschriftendaten beim Scoring

Vorgabe: Findet Scoring statt, DÜRFEN NICHT ausschließlich Anschriftendaten Grundlage zur Berechnung des Scoring-Werts sein.

Hinweise: Es MÜSSEN also zur Bildung eines Scores weitere Daten, die keinen Bezug zur Anschrift haben, verwendet werden. Ein ausschließliches Subset aus Straße+Nr., Ort, PLZ, Land, Geokoordinaten usw. DARF NICHT verwendet werden.

Quelle(n): DSG; EU-DSGVO

2.21.7 Informations- und Dokumentationspflichten beim Scoring

Vorgabe: Findet Scoring statt, MUSS bei Nutzung von Anschriftendaten die betreffende Person vor Berechnung des Scoring-Werts über die vorgesehene Datennutzung informiert werden, was weiterhin protokolliert werden muss.

Hinweise: Dies KANN über die Arbeitsabläufe der Anwendung geschehen oder über vorgelagerte (auch nicht-EDV) Prozesse.

Quelle(n): DSG; EU-DSGVO

2.21.8 Keine Verwendung besonderer Kategorien personenbezogener Daten beim Scoring

Vorgabe: Findet Scoring statt, DARF KEINE Entscheidung auf Basis von „besonderen Kategorien personenbezogener Daten“ getroffen werden.

Hinweise: Besondere Kategorien personenbezogener Daten sind (entspr. EU-DSGVO Art. 9 Abs. 1): Daten, aus denen die rassische oder ethnische Herkunft, politische Meinungen, religiöse oder weltanschauliche Überzeugungen oder die Gewerkschaftszugehörigkeit hervorgehen, genetische Daten, biometrische Daten zur eindeutigen Identifizierung einer natürlichen Person, Gesundheitsdaten und Daten zum Sexualleben oder zur sexuellen Orientierung.

Quelle(n): EU-DSGVO; DSG

2.21.9 Minimalprinzip für Erhebung und Verarbeitung personenbezogener Daten

Vorgabe: Es DÜRFEN NUR personenbezogene Daten erhoben und gespeichert werden, die für die vorgesehene, festgelegte Bearbeitung benötigt werden.

Hinweise: Sämtliche verwendeten Datenstrukturen DÜRFEN NUR diejenigen Daten / Attribute mit Personenbezug enthalten, die für die vorgesehene Bearbeitung benötigt werden.

Quelle(n): EU-DSGVO; DSG

2.21.10 Minimalprinzip für Identifizierbarkeit durch personenbezogene Daten

Vorgabe: Personenbezogene Daten DÜRFEN NICHT länger als für die vorgesehene Verarbeitung nötig in einer Form gespeichert werden, die die Identifizierung der betroffenen Personen ermöglicht.

Hinweise: Nach vorgesehener Verarbeitung MÜSSEN die personenbezogenen Daten entweder gelöscht oder pseudonymisiert werden.

Quelle(n): EU-DSGVO; DSG

2.21.11 Einwilligung in die Verarbeitung personenbezogener Daten

Vorgabe: Findet entsprechend dem mit dem Datenschutzbeauftragten abgestimmten DV-Konzept eine Verarbeitung von Daten aufgrund einer Einwilligung der betroffenen Person statt und wird diese Einwilligung über die Anwendung erhoben, so MUSS diese Einwilligung nachgewiesen werden können. Hierzu MUSS ein entsprechender Protokolleintrag oder etwas Äquivalentes erstellt werden, dessen Speicherung mindestens für die Gesamtdauer des Geschäftsverhältnisses sichergestellt werden MUSS.

Hinweise: Entsprechende Protokolleinträge über die Einwilligung DÜRFEN NICHT Teil üblicher Zugriffsprotokolle (Access-Logs) sein, die ggf. regelmäßig gelöscht werden könnten. Protokolleinträge MÜSSEN mindestens einen Zeitstempel und die betreffende Account-ID (oder die Remote IP-Adresse) beinhalten. Die angezeigten Einwilligungs- und Widerrufstexte MÜSSEN mit dem Datenschutzbeauftragten abgestimmt werden. Eine vorhandene Einwilligung KANN als Attribut zum betreffenden Benutzerkonto vermerkt werden.

Quelle(n): EU-DSGVO; DSG

2.21.12 Widerruf der Verarbeitung personenbezogener Daten

Vorgabe: Findet eine Verarbeitung von Daten auf Basis einer Einwilligung statt, so MUSS die Möglichkeit eines Widerrufs auch applikationsseitig vorgesehen werden.

Hinweise: Hierbei MUSS der Widerruf, wenn er über die Anwendung erfolgt ist, unter Berücksichtigung derselben Protokollierungsvorgaben, die auch für die entsprechende Einwilligung gelten, erfolgen. Ein Widerruf KANN z. B. über ein Attribut im betreffenden Benutzerkonto vermerkt werden bzw. das Attribut für die Einwilligung zurücksetzen. Auf die Möglichkeit des Widerrufs MUSS hingewiesen werden, die Form des Hinweises MUSS mit dem Datenschutzbeauftragten abgestimmt werden.

Quelle(n): EU-DSGVO; DSG

2.21.13 Verarbeitung personenbezogener Daten durch Dritte

Vorgabe: Bei Anbindung von Systemen Dritter via (Web-)APIs oder (Web-)Services für die Verarbeitung personenbezogener Daten MUSS eine vorherige Freigabe des Datenschutzbeauftragten eingeholt werden.

Hinweise: Die Freigabe SOLLTE im Rahmen der Dokumentation (Sicherheitskonzept / DV-Konzept o. ä.) der Anwendung nachvollziehbar mit abgelegt oder auf diese verwiesen werden.

Quelle(n): EU-DSGVO; DSG

2.21.14 Datenaufbereitung für Auskunftsrecht

Vorgabe: Werden personenbezogene Daten verarbeitet, SOLLTE die Bereitstellung der Metadaten zur Erteilung angefragter Auskünfte über gespeicherte und verarbeitete personenbezogene Daten vorgesehen werden. Hierfür KÖNNEN geeignete, ggf. als „für Auskunft relevant“ markierte Datenstrukturen eingesetzt werden, weiterhin SOLLTEN Funktionalitäten bestehen, die die relevanten Daten zu einer Anfrage aggregieren.

Hinweise: Die folgenden (Meta)daten zu den personenbezogenen Daten MÜSSEN berücksichtigt werden und aggregiert werden können: Die Zwecke, zu denen die personenbezogenen Daten verarbeitet werden; die Kategorien personenbezogener Daten, die verarbeitet werden; die Empfänger oder Kategorien von Empfängern, gegenüber denen die personenbezogenen Daten offengelegt worden sind oder noch offengelegt werden; falls möglich die geplante Dauer, für die die personenbezogenen Daten gespeichert werden oder, falls dies nicht möglich ist, die Kriterien für die Festlegung dieser Dauer; wenn die personenbezogenen Daten nicht bei der betroffenen Person erhoben werden, alle verfügbaren Informationen über die Herkunft der Daten; ggf. das Bestehen einer automatisierten Entscheidungsfindung einschließlich Profiling, einschließlich aussagekräftiger Informationen über die involvierte Logik, sowie die Tragweite und die angestrebten Auswirkungen einer derartigen Verarbeitung für die betroffene Person. Die Form der Übergabe der personenbezogenen Daten MUSS mit dem Datenschutzbeauftragten abgestimmt werden.

Quelle(n): EU-DSGVO; DSG

2.21.15 Berichtigung personenbezogener Daten

Vorgabe: Für alle Datenstrukturen oder Felder von Datenstrukturen, die sich auf personenbezogene Daten beziehen, MUSS eine Möglichkeit zur Korrektur implementiert werden.

Hinweise: Dies beinhaltet z. B. Name, Geschlecht, Alter und alle weiteren Attribute mit Personenbezug. Es MUSS unter Berücksichtigung der vorhandenen Rollen- und Rechtevergaben mindestens einen technischen und/oder prozessualen Weg geben, diese Angaben jeweils zu ändern.

Quelle(n): EU-DSGVO; DSG

2.21.16 Löschung personenbezogener Daten

Vorgabe: Für alle Datenstrukturen oder Felder von Datenstrukturen, die sich auf personenbezogene Daten beziehen, MUSS eine Möglichkeit zur Löschung implementiert werden.

Hinweise: Es MUSS unter Berücksichtigung der vorhandenen Rollen- und Rechtevergaben mindestens einen technischen und/oder prozessualen Weg geben, diese Angaben jeweils zu löschen.

Quelle(n): EU-DSGVO; DSG

2.21.17 Einschränkung der Verarbeitung personenbezogener Daten

Vorgabe: Für alle Datenstrukturen oder Felder von Datenstrukturen, die sich auf personenbezogene Daten beziehen, MUSS eine Möglichkeit zur Einschränkung / Unterbindung der weiteren Verarbeitung implementiert werden.

Hinweise: Dies bedeutet natürlich auch, dass die entsprechenden Flags zur Unterbindung/Einschränkung der Weiterverarbeitung vor Weiterverarbeitung geprüft und berücksichtigt werden MÜSSEN.

Quelle(n): EU-DSGVO; DSG

2.21.18 Datenübertragbarkeit personenbezogener Daten

Vorgabe: Für personenbezogene Daten MUSS eine Funktionalität zum Exportieren aller personenbezogenen Daten zu einer Person existieren.

Hinweise: Hierbei MÜSSEN portable Datenformate gewählt werden, die eine Datenübernahme in andere Anwendungen ermöglichen (z. B. XML, CSV).

Quelle(n): EU-DSGVO; DSG

2.22 Sichere Konfiguration

Für viele Klassen von Anwendungen sind neben Berücksichtigung sicherheitsrelevanter Aspekte bei der Programmierung auch ebendiese bei der Konfiguration erforderlich. Auch soll die Entwicklung eine spätere Konfiguration von Sicherheitseinstellungen ermöglichen. Diese Thematik wird durch die nachfolgenden Vorgaben abgedeckt.

2.22.1 Minimalprinzip für Konfigurationen

Vorgabe: Konfigurationen MÜSSEN standardmäßig dem Minimalprinzip entsprechen.
Hinweise: Daher DÜRFEN NUR die standardmäßig benötigten Funktionen und Features standardmäßig aktiviert sein bzw. Informationen per Default angezeigt werden.
Quelle(n): BSI-WEB-AN

2.22.2 Zugriffsbeschränkung für Konfigurationsinformationen

Vorgabe: Alle sicherheitsrelevanten Konfigurationsinformationen DÜRFEN NUR an Orten gespeichert werden, die vor unbefugtem Zugriff geschützt sind.
Quelle(n): BSI-WEB-AN

2.22.3 Sicherstellung vorhandener Sicherheitskonfiguration

Vorgabe: Alle Zugriffe auf die Anwendung MÜSSEN abgelehnt werden, wenn diese keinen Zugriff zu ihrer Sicherheitskonfiguration herstellen kann.
Quelle(n): BSI-WEB-AN

2.22.4 Auditierbare Sicherheitskonfiguration

Vorgabe: Die sicherheitsrelevanten Konfigurationseinstellungen SOLLTEN in einem menschenlesbaren und verständlichen Format abgelegt sein, sodass diese auditiert werden können.
Hinweise: Es SOLLTE, soweit möglich, eine Kommentierung und / oder Dokumentation der sicherheitsrelevanten Konfigurationseinstellungen existieren bzw. erstellt werden.
Quelle(n): OWASP-SCP

2.22.5 Keine Verwendung von Standardanmeldedaten

Vorgabe: Alle produkt- bzw. herstellerspezifischen Standardpasswörter MÜSSEN (und Kontonamen / IDs SOLLTEN) geändert oder die betreffenden Standardkonten deaktiviert werden.
Hinweise: Nutzer sollten nach der ersten Anmeldung gezwungen sein herstellerspezifische Passwörter zu ändern. Wenn Standardkonten existieren, die im Regelfall nicht verwendet werden, müssen diese dokumentiert sein.
Quelle(n): OWASP-SCP; TSS-WEB

2.22.6 Begrenzung der Zugriffsrechte auf Zugangsdaten

Vorgabe: Die Ablage / Datenbank für Zugangsdaten SOLLTE Leserechte und MUSS Schreibrechte ausschließlich für die jeweilige Anwendung (bzw. den Systembenutzer der Anwendung) besitzen.

Hinweise: Die Anwendung sollte also mit den Rechten eines eigenen technischen Nutzers laufen, damit die Lese- und Schreiberechte des Betriebssystems genutzt werden können.

Quelle(n): OWASP-SCP

2.22.7 Schutz des Quellcodes

Vorgabe: Serverseitiger Quellcode und zugehörige Konfigurationsdateien MÜSSEN vor Einsicht durch Unbefugte geschützt sein und DÜRFEN NICHT von Unbefugten heruntergeladen werden können.

Hinweise: Hierzu SOLLTEN Download-Funktionen stets mit erlaubten Datei-IDs parametrisiert und eine Autorisierung überprüft werden. Eine Parametrisierung mit relativen oder absoluten serverseitigen Dateipfaden SOLLTE NICHT erfolgen.

Quelle(n): OWASP-SCP

2.22.8 Keine Veröffentlichung nicht öffentlich benötigter Dokumentation

Vorgabe: Dokumentation zu Anwendung, Konfiguration, Komponenten usw., die nicht öffentlich benötigt wird, DARF NICHT öffentlich einsehbar sein.

Hinweise: Diese Dokumentationsdateien SOLLTEN NICHT Bestandteil von Produktivanwendungen bzw. Systemen sein.

Quelle(n): OWASP-SCP

2.22.9 Zugriffskontrollen für vertrauliche serverseitige Daten

Vorgabe: Für vertrauliche Daten, die serverseitig abgelegt werden, MUSS ein Zugriffskontrollmechanismus umgesetzt sein.

Hinweise: Hierbei MÜSSEN auch gecachte Daten, temporäre Dateien sowie Daten, die nur für bestimmte Systembenutzer einsehbar sein sollten, berücksichtigt werden.

Quelle(n): OWASP-SCP

2.22.10 Einsatz aktueller freigegebener Software-Versionen

Vorgabe: Es MUSS sichergestellt werden, dass die verwendete Software, insbesondere Server-, Framework- und Systemkomponenten, jeweils der aktuellen vom Hersteller/Autor noch mit Sicherheitsaktualisierungen unterstützenden, freigegebenen Version entsprechen.

Quelle(n): OWASP-SCP

2.22.11 Einsatz aller verfügbaren Patches

Vorgabe: Es MUSS sichergestellt werden, dass die für die eingesetzten Softwarekomponenten verfügbaren Patches, insbesondere Security-Patches, eingespielt sind.
Hinweise: Bekannte Sicherheitslücken (CVEs) ab hohem Impact (CVSS Rating >= 7.0), für die (noch) keine Sicherheitspatches/-updates existieren, MÜSSEN durch Workarounds, Konfigurationsanpassungen oder Risikoübernahme adressiert werden.
Quelle(n): OWASP-SCP

2.22.12 Kein Directory Listing

Vorgabe: Es MUSS sichergestellt werden, dass Directory Listing (Auflistung von Verzeichnisinhalten) für serverseitige Pfade deaktiviert ist.
Hinweise: Dies SOLLTE als Standardeinstellung für eingesetzte Webserver konfiguriert werden.
Quelle(n): OWASP-SCP; BSI-WEB-AN

2.22.13 „Least Privilege“-Prinzip für Systemkonten

Vorgabe: Das „Least Privilege“-Prinzip MUSS für (Web)server, Prozesse, Daemons, Servicekonten usw. umgesetzt werden: Systemkonten DÜRFEN NUR auf die Funktionalität, Daten und Systeminformationen zugreifen, die jeweils benötigen werden.
Hinweise: Hierbei SOLLTEN soweit möglich nicht nur die Objekte, Informationen und Methoden eingeschränkt werden, sondern auch die Art des Zugriffs (z. B. nur lesend).
Quelle(n): OWASP-SCP

2.22.14 Entfernen aller Testdaten und -funktionen vor Produktivstellung

Vorgabe: In die Produktivumgebung SOLLTEN NICHT Daten, Dateien und Funktionen übernommen werden, die nur für den Entwicklungs- oder Testbetrieb benötigt werden. Insbesondere DÜRFEN NICHT erweiterte Diagnose- oder Administrationswerkzeuge oder -funktionen, die produktiv nicht benötigt werden, mit auf Produktivumgebungen migriert werden bzw. dort vorhanden sein.
Hinweise: Es SOLLTE – bspw. anhand von Dokumentation oder durch Abgleich mit einem Versionskontrollsystem – einfach möglich sein, herauszufinden, welche Dateien tatsächlich für den produktiven Anwendungsbetrieb benötigt werden.
Quelle(n): OWASP-SCP

2.22.15 Entfernen nicht benötigter Informationen vor Produktivstellung

Vorgabe: Kommentare und weitere für den Produktionsbetrieb nicht relevante Informationen SOLLTEN aus Quelltexten, Konfigurationsdateien und ausführbaren Dateien entfernt werden, bevor diese im Produktionssystem verwendet werden. Wenn die Entwicklungsumgebung es unterstützt, SOLLTE sie so konfiguriert werden, dass bei der Erstellung von Programmpaketen automatisch alle für den vorgesehenen Verwendungszweck nicht relevanten Informationen entfernt werden. Es SOLLTE sichergestellt werden, dass hierfür genau dokumentiert ist, welche Informationen im Interesse des Auftraggebers im fertigen Programmpaket enthalten sein sollen. Relevante Kommentare, Dokumentationen und Zusatzinformationen in den Entwicklungsdaten MÜSSEN stets erhalten bleiben, um sicherzustellen, dass die Software jederzeit geprüft und gewartet werden kann.

Hinweise: Für Binärdateien KANN dies u. a. unter Zuhilfenahme des Tools „strip“ umgesetzt werden, für C-Source-Files mit GCC durch Parametrisierung mit „-fpreprocessed -dD -E“, für Java-Source-Files u. a. mit dem Tool „java-comment-preprocessor (JCP)“. Hierbei MUSS sichergestellt werden, dass nicht erklärende Kommentare aus Konfigurationsdateien entfernt werden.

Quelle(n): BSI-ITGS-B5.27; BSI-WEB-AN

2.22.16 Kein Information Disclosure über robots.txt

Vorgabe: Die robots.txt-Datei DARF NICHT interne Informationen, insbesondere interne Verzeichnisstrukturen von Unterverzeichnissen, enthalten.

Hinweise: Verzeichnispfade, die nicht für öffentliche Indizierung bestimmt sind, SOLLTEN sich in gemeinsamen Stammverzeichnissen befinden, die in der robots.txt aufgeführt werden DÜRFEN.

Quelle(n): OWASP-SCP

2.22.17 Security Header zur Unterbindung von Cross-Site-Scripting (XSS)

Vorgabe: Browserseitige Erkennung und Unterbindung von Cross-Site-Scripting (XSS) bei Webapplikationen SOLLTE mit der sichersten möglichen Konfiguration verwendet werden (Header: „X-XSS-Protection: 1; mode=block“). Das Feature DARF NICHT deaktiviert werden (Header-Wert: „0“).

Hinweise: Dies KANN durch die Anwendung selbst oder – falls nicht möglich – in der Webserver-Konfiguration gesetzt werden.

Quelle(n): VHV-RSE

2.22.18 Security Header zur Vermeidung von Clickjacking

Vorgabe: Bei Webapplikationen MÜSSEN sog. Clickjacking-Angriffe unterbunden werden, indem das Einbetten von (Teilen) der Webapplikation deaktiviert wird. Hierzu MUSS der Security Header „X-Frame-Options“ mit einem sicheren Wert (z. B. „deny“) gesetzt werden.

Hinweise: Dies KANN durch die Anwendung selbst oder – falls nicht möglich – in der Webserver-Konfiguration gesetzt werden.

Quelle(n): VHV-RSE

2.22.19 Security Header zum Whitelisting von Inhalten

Vorgabe: Für Webapplikationen SOLLTE restriktiv konfiguriert werden, aus welchen Quellen unterschiedliche Inhalte eingebunden werden dürfen. Hierfür SOLLTE der „Content-Security-Policy“ (CSP)-Header mit möglichst restriktiven Einstellungen (z. B. „default-src 'self'“) verwendet werden.

Hinweise: Wenn Legacy-Clients unterstützt werden sollen, SOLLTEN die CSP-Einstellungen ebenfalls über die „X-Content-Security-Policy“, sowie „X-Webkit-CSP“-Header gesetzt werden. Als Test für einen restriktiv konfigurierten CSP-Header KANN der Online-Service https://csp-evaluator.withgoogle.com herangezogen werden.

Quelle(n): VHV-RSE

2.22.20 Kein Information Disclosure über HTTP-Header

Vorgabe: Es MUSS sichergestellt werden, dass ausgegebene HTTP-Header keine vertraulichen Informationen enthalten. Hierzu gelten auch die eingesetzten OS- und Softwarekomponenten und deren jeweilige Versionsstände, die wertvolle Informationen für Angreifer sind.

Hinweise: Dies KANN i. d. R. über Konfigurationsanpassungen des jeweiligen Web Application Servers, des eingesetzten Frameworks oder der Applikation erfolgen. Alternativ KÖNNEN die betreffenden Header auch z. B. von einem Proxy entfernt oder modifiziert werden – Dies betrifft beispielsweise die Header „Server“, „X-Powered-By“, „X-AspNet-Version“, „X-AspNetMvc-Version“.

Quelle(n): OWASP-SCP

2.22.21 Schreibschutz für Anwendungsdateien

Vorgabe: Anwendungsdateien und -ressourcen MÜSSEN mindestens für das die Anwendung ausführende Systemkonto schreibgeschützt sein.

Hinweise: Es SOLLTEN auf Windowssystemen Gruppenrichtlinien verwendet werden, die das Verändern der Anwendungsdateien für die entsprechenden Accounts über sinnvolle Rechtevergabe verhindern.

Quelle(n): OWASP-SCP

2.22.22 „Least Privilege“-Prinzip für Datenbankkonten

Vorgabe: Das „Least Privilege“-Prinzip MUSS für Datenbankkonten umgesetzt werden: Konten zum Zugriff auf Datenbanken DÜRFEN NUR Rechte für die Funktionalität, Daten und Systeminformationen besitzen, die tatsächlich jeweils benötigt werden.
Hinweise: Hierbei SOLLTEN soweit möglich nicht nur die Objekte, Informationen und Methoden eingeschränkt werden, sondern auch die Art des Zugriffs (z. B. nur lesend).
Quelle(n): OWASP-SCP

2.22.23 Sichere Zugangsdaten für Datenbankzugriff

Vorgabe: Für verwendete Datenbankkonten MÜSSEN sichere Zugangsdaten gesetzt sein bzw. verwendet werden.
Hinweise: Die verwendeten Passwörter MÜSSEN die Anforderungen der aktuell gültigen Arbeitsrichtlinie Kennwörter erfüllen.
Quelle(n): OWASP-SCP

2.22.24 Sichere Authentifizierung für Datenbankadministration

Vorgabe: Es DÜRFEN KEINE Standardpasswörter für die Datenbankadministrationsschnittstellen existieren. Vorhandene Standardkonten SOLLTEN entweder entfernt oder deaktiviert werden oder mit einem sicheren Passwort ausgestattet werden.
Hinweise: Wenn möglich, SOLLTE (insbesondere für Anwendungen mit vertraulichen Daten) eine Zwei-Faktor-Authentifizierung für die Datenbankadministration eingerichtet werden.
Quelle(n): OWASP-SCP

2.22.25 Deaktivierung nicht benötigter Datenbankfunktionalität

Vorgabe: Nicht benötigte Datenbankfunktionalität (nicht genutzte Stored Procedures, Services, Module usw.) SOLLTE entfernt werden. Die Datenbankumgebung SOLLTE nur das jeweils minimal benötigte Feature-Set umfassen.
Quelle(n): OWASP-SCP

2.22.26 Entfernen nicht benötigter Standarddatenbankinhalte

Vorgabe: Nicht benötigte, ggf. herstellerspezifische Standarddatenbankinhalte (Beispieldatenbanken, -tabellen, -prozeduren usw.) MÜSSEN in produktiven Umgebungen entfernt werden.
Hinweise: In MySQL-Datenbanken ist es beispielsweise möglich mittels „SHOW DATABASES;“ sich sämtlichen Datenbanken anzeigen zu lassen.
Quelle(n): OWASP-SCP

2.22.27 Deaktivieren nicht benötigter Datenbankkonten

Vorgabe: Nicht benötigte, ggf. herstellerspezifische Standarddatenbankkonten, MÜSSEN deaktiviert (oder entfernt) werden.

Hinweise: In MySQL-Datenbanken ist es beispielsweise möglich mittels „SELECT User FROM mysql.user;“ sich alle User anzeigen zu lassen.

Quelle(n): OWASP-SCP

3 ERGÄNZENDE INHALTE

3.1 HTTP-Security Header-Sicherheitsmechanismen

Eine Vielzahl von möglichen Sicherheitsproblemen lässt sich bereits durch Verwendung und sinnvolle, restriktive Konfiguration bestimmter HTTP-Header, insbesondere der sogenannten Security Header, adressieren. Diese können auf Anwendungs-, Webserver- oder Proxy-Ebene gesetzt werden und sind nachfolgend aufgeführt.

3.1.1 Content-Security-Policy

Effektive Maßnahme zum Whitelisting von erlaubten eingebundenen Webinhalten. Einfachster und grundsätzlich zu empfehlender Anwendungsfall ist die Nutzung zur Einschränkung aller erlaubten Inhalte auf Dateien vom Anwendungshost wie folgt:
Content-Security-Policy: default-src 'self'; script-src 'strict-dynamic' 'nonce-rAnd0m123' 'unsafe-inline' http: https:; style-src 'self' 'unsafe-inline'; object-src 'none'; base-uri 'none'

3.1.2 Referrer-Policy

Der HTTP-Referrer-Header verrät einem Webserver, von welcher Seite Ressourcen angefragt wurden bzw. von welcher Seite ein Besucher einem Link gefolgt ist. Schlimmstenfalls können hierüber vertrauliche, in URLs enthaltene Informationen, Dritten übermittelt werden (Information Disclosure). Die folgende empfohlene Einstellung erlaubt die Übermittlung des „Referrer“ nur an den Host der Anwendung selbst:
Referrer-Policy: same-origin

3.1.3 Set-Cookie

Cookies werden primär zur Wiedererkennung von Anwendern über Session-IDs benutzt. Einige wichtige sicherheitsrelevante Attribute für Cookies sind:

- „secure“-Attribut
  Das „secure“-Attribut verhindert eine unverschlüsselte Übertragung des Cookies. Eine Übertragung des Cookies setzt somit eine verschlüsselte Verbindung (TLS) voraus.
- „HttpOnly“-Attribut
  Das „HttpOnly“-Attribut verhindert einen clientseitigen Zugriff auf das Cookie über JavaScript und erschwert somit das Auslesen von Session-IDs durch eine Script Injection enorm.
- „domain“-Attribut
  Das „domain“-Attribut spezifiziert den domainbezogenen Geltungsbereich des Cookies und der (Sub)domains, zu denen das Cookie übertragen wird. Die restriktivste Möglichkeit ist, das Attribut nicht zu setzen – dann dürfen Cookies nur zu dem exakten Host der originären Anwendung übertragen werden.
- „path“-Attribut
  Das „path“-Attribut schränkt ein, an welche Seiten das Cookie übertragen wird. Dies ist von Bedeutung, wenn mehrere Anwendungen unter derselben Domain eingerichtet sind und Cookies nicht anwendungsübergreifend benötigt werden. In diesem Fall sollte der Pfad beispielsweise auf „/anwendungspfad/“ gesetzt werden.

- „expires“- und „max-age“-Attribut
  Durch Verwendung des „expires“- oder „max-age“-Attribut (letzteres hat Präferenz vor dem ersteren) lässt sich die Gültigkeitsdauer von Cookies festlegen. Nach Überschreiten eines gesetzten Zeitwerts wird das Cookie clientseitig nicht mehr verwendet. Ohne „expires“- oder „max-age“-Attribut werden Cookies clientseitig nach Beenden der Browser-Sitzung nicht mehr verwendet.
- „SameSite“-Attribut
  Mit dem Setzen des „SameSite“-Attributes auf „Strict“, wird dem Browser befohlen nie Requests, die von einer Cross Site-Quelle stammen zu öffnen. Dies kann aber zu ungewolltem Verhalten führen, daher wird mindestens als Attribut „Lax“ empfohlen, wenn „Strict“ zu ungewünschten Nebeneffekten führt.

3.1.4 Strict-Transport-Security

Der Strict-Transport-Security-Header weist Clients an, nach erstmaligem Zugriff über eine gesicherte TLS-Verbindung ausschließlich diese zu verwenden. Ein Fallback auf ungesicherte Verbindungen zu der Anwendungsdomain (und optional auch Subdomains) ist dann für diesen Client für einen definierten Zeitraum nicht mehr möglich. Als zusätzliche Option ermöglicht das „preload“-Attribut die Anweisung an Browser-Hersteller, die Domain mit in eine Preload-Liste aufzunehmen – damit auch der erstmalige Verbindungsaufbau von Clients bereits via TLS erfolgt (hierfür ist ein Eintrag auf https://hstspreload.org notwendig).
Der folgende empfohlene Wert weist Clients an, für den Zeitraum von einem Jahr ab erstem gesicherten Verbindungsaufbau weiterhin nur noch gesicherte Verbindungen, auch zu allen Subdomains, durchzuführen und verwendet das zuvor beschriebene „preload“-Attribut.
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload

3.1.5 X-Content-Type-Options

Der X-Content-Type-Options-Header weist Clients bei Verwendung dazu an, den Content-Type von empfangenen Inhalten niemals selbst zu ermitteln, sondern Inhalte anhand des zwingend mit zu übergebenden Content-Type-Headers zu interpretieren. Dies ist empfohlen, da es bestimmte Angriffsszenarien mitigiert.
X-Content-Type-Options: nosniff

3.1.6 X-Frame-Options

Dieser Header ermöglicht es, das Einbinden von Anwendungsseiten in Frames zu verhindern. Empfohlen wird, sofern nicht explizit benötigt, wie folgt ein Einbinden in Frames von Dritten zu verhindern:
X-Frame-Options: deny

3.1.7 X-XSS-Protection

Der X-XSS-Protection Header verhindert Cross-Site-Scripting.
X-XSS-Protection: 1; mode=block

3.1.8 Content-Disposition

In Kombination mit dem Attribut „attachment“ sorgt der Header dafür, dass der Inhalt nicht im Browser ausgeführt wird, sondern herunterladen werden soll. Er sollte zusammen mit „X-Download-Options“ verwendet werden:
Content-Disposition: attachment; filename=<dateiname>

3.1.9 X-Download-Options

Dieser Header verhindert, dass alte Versionen von Internet Explorer die Ausführung bösartiger HTML-Downloads im Kontext der Website zulassen:
X-Download-Options: noopen

3.2 Häufige Schwachstellen in Webanwendungen (OWASP TOP 10)

Die folgenden „Top 10“ stellen die weitverbreitetsten Klassen von Sicherheitslücken für Webanwendungen dar. Für alle internet- oder intranetexponierten Anwendungen sollten diese berücksichtigt und adressiert werden:

- **A1: Injection**
  Injection-Angriffe nutzen die anwendungsseitig unzureichend implementierte Trennung von Eingabedaten und zu interpretierenden Daten aus, um unterschiedlichen Code in einen Anwendungskontext zu injizieren. Eine Injection kann schlimmstenfalls vollständige Systemkompromittierungen erlauben.
- **A2: Broken Authentication and Session Management**
  Logische und technische Fehler im Authentifizierungs- und Sitzungsmanagement können die Übernahme von Sitzungen und Ausweitung von Anwendungsprivilegien ermöglichen.
- **A3: Sensitive Data Exposure**
  Eine Vielzahl vertraulicher Daten wird fälschlicherweise häufig nicht als selbiges eingestuft oder inkorrekt vor nicht ausreichend authentifizierten Zugriffen geschützt. Dies kann dazu führen, dass vertrauliche Daten exfiltriert werden – was entweder bereits Ziel der Angreifer ist oder ggf. weitergehende Angriffe ermöglichen kann.
- **A4: XML External Entities (XXE)**
  Viele ältere oder schlecht konfigurierte XML-Prozessoren werten externe Entity-Referenzen in XML-Dokumenten aus. Externe Entitäten können verwendet werden, um interne Dateien mithilfe des Datei-URI-Handlers, interner Dateifreigaben, interner Port-Scans, Remotecodeausführung und Denial-of-Service-Attacken offenzulegen.
- **A5: Broken Access Control**
  Eine inkonsequente oder fehlerhafte Forcierung von Rollen- / Rechtekonzepten ermöglicht diese Klasse von Sicherheitslücken und damit u. a. Privilegienausweitungen für Angreifer.
- **A6: Security Misconfiguration**
  Viele Schwachstellen lassen sich bei möglichst sicherer Konfiguration und konsequentem Patch-Management erst gar nicht ausnutzen – andernfalls öffnen sich an dieser Stelle Sicherheitslücken.
- **A7: Cross-Site Scripting (XSS)**
  Eine unzureichende Validierung sowie unvollständiges Escaping nicht vertrauenswürdiger Eingabedaten, welche erneut ausgegeben werden, ermöglicht diese Klasse von Sicherheitslücken, welche das Einschleusen von bösartigem HTML und JavaScript-Code in Benutzersitzungen bis hin zu Persistenz des Codes in der Anwendung ermöglicht.

- **A8: Insecure Deserialization**
  Unsichere Deserialisierung führt oft zu Remotecodeausführung. Selbst wenn Deserialisierungsfehler nicht zu einer Remotecodeausführung führen, können sie zur Ausführung von Angriffen verwendet werden, einschließlich Replay-Angriffen, Injektionsangriffen und Privilegien-Eskalationsangriffen.
- **A9: Using Components with Known Vulnerabilities**
  Ein schlechtes Schwachstellen- oder Patch-Management führt dazu, dass bekanntwerdende Schwachstellen in Anwendungskomponenten nicht behoben werden und so schlimmstenfalls eine Systemkompromittierung durch Angreifer ermöglicht wird.
- **A10: Insufficent Logging & Monitoring**
  Unzureichende Protokollierung und Monitoring, gekoppelt mit fehlender oder ineffektiver Integration mit der Reaktion auf Vorfälle (Incident Response), ermöglicht es Angreifern, weitere Angriffe auf Systeme durchzuführen, die Persistenz aufrechtzuerhalten, weitere Systeme zu steuern und Daten zu manipulieren, zu extrahieren oder zu zerstören. Die meisten Untersuchungen zeigen, dass die Zeit bis zur Feststellung eines Verstoßes mehr als 200 Tage beträgt, was in der Regel eher von externen Parteien als von internen Prozessen oder dem Monitoring erkannt wird.

3.3 Vorgaben der Richtlinie „Sichere Entwicklung“ in tabellarischer Form

Die in dieser Richtlinie enthaltenen Vorgaben existieren in einer erweiterten tabellarischen Form („Maßnahmen Richtlinie Sichere Entwicklung.xlsm“), die die Vorgaben betreffend datenführend ist. Dieses Tabellendokument enthält neben den in dieser Richtlinie aufgeführten Vorgaben, Umsetzungshinweisen und Quellverweisen zusätzliche Metadaten, wie die technisch auswertbare Relevanz ab bestimmten Schutzbedarfskategorien („MUSS“ oder „KANN“) und eine ID, aus der die Referenz auf die Primärquelle ersichtlich ist.

4 GLOSSAR

API	Application Programming Interface. Schnittstelle zu Funktionen, die durch (dritte) Anwendungen genutzt werden können.
Blacklisting	Einschränkung zulässiger Werte anhand einer Liste verbotener Werte. Nicht explizit verbotene Werte sind erlaubt.
CORS	Cross-Origin Resource Sharing. Mechanismus, über den bestimmte (Web-)Inhalte bestimmten Drittanwendungen / Domains zur Verwendung freigegeben werden können.
CSV	Comma-Separated Value. Plain-Text-Dateiformat für tabellarische Daten.
CVE	Common Vulnerabilities and Exposures. Standard zur eindeutigen Referenzierung von Sicherheitslücken.
CVSS	Common Vulnerability Scoring System. Verfahren zur Berechnung eines numerischen Scores zu einer Sicherheitslücke, der deren Kritikalität abstrakt wiedergibt.
DBMS	Database Management System. Anwendung zur Verwaltung von Datenbanken. Umgangssprachlich wird ein DBMS selbst auch als
DSB	Datenschutzbeauftragter. Siehe Kapitel „Rollen“.
DV-Konzept	Datenverarbeitungskonzept. Ein Fachkonzept, dass die relevanten Daten und deren Verarbeitung beschreibt.
ITSB	IT-Sicherheitsbeauftragter. Siehe Kapitel „Rollen“.

JSON	JavaScript Object Notation. Einfache textbasierte Notationssprache zur Beschreibung von strukturierten Informationen.
LDAP	Lightweight Directory Access Protocol. Serviceprotokoll zum Anbinden, Durchsuchen und Bearbeiten von Internetverzeichnissen, wie beispielsweise einem Active Directory.
OWASP	Open Web Application Security Project. Community und de Facto-Industriestandard(s) für Webanwendungssicherheit.
Refer(r)er	Originäre Lokation, von der aus ein URL aufgerufen wurde.
RFC	Request for Comments. Technische und organisatorische Dokumente zum Internet, die Protokolle und andere Standards beschreiben.
RSA	Rivest-Shamir-Adleman. Für sicheren Datenaustausch verwendetes, asymetrisches Public-Key-Kryptosystem.
SDLC	Software Development Life Cycle. Beschreibt alle Phasen, die Software durchläuft (i. d. R. mindestens Anforderungsanalyse, Design & Architektur, Implementierung (Programmierung), Tests, Inbetriebnahme, Betrieb & Wartung, oft auch Training, Außerbetriebnahme, u. a.).
SEP	Software-Entwicklungsprozess. Der im Rahmen der Einführung (Anschaffung oder (tlw.) Entwicklung) einer Anwendung bei der VAV einzuhaltende Prozess.
SQL	Structured Query Language. Sprache zur Ansteuerung eines DBMS, z. B. zur Erstellung strukturierter Datenbankabfragen.
TLS	Transport Layer Security. TLS und der Vorgänger SSL (Secure Sockets Layer) sind kryptographische Protokolle zur sicheren Netzwerkkommunikation.
Vertrauliche Daten	Daten, die (1) vertrauliche Informationen enthalten bzw. enthalten könnten (z. B. Patente, sensible Rechenlogik, personenbezogene Daten, Passwörter), (2) explizit als solche gekennzeichnet wurden oder (3) allgemein nur einem eingeschränkten Personenkreis zugänglich sind bzw. sein sollen.
Whitelisting	Einschränkung zulässiger Werte anhand einer Liste erlaubter Werte. Nicht explizit erlaubte Werte sind verboten.
XML	eXtendible Markup Language. Sprache zur semantischen Strukturierung von Informationen.
XSS	Cross-Site Scripting. Angriff, der clientseitigen Script-Code in den Anwendungskontext von Anwendersitzungen einschleust.