

Sicherheitsanforderung

Client-Anwendungen

Deutsche Telekom Gruppe

Version	7.0
Datum	01.07.2025
Status	Freigegeben

Impressum

Herausgeber
Deutsche Telekom AG
Vorstandsbereich Technology & Innovation
Chief Security Officer

Reuterstrasse 65, 53315 Bonn
Deutschland

Dateiname	Dokumentennummer 3.54	Dokumententyp Sicherheitsanforderung
Version 7.0	Stand 01.07.2025	Status Freigegeben
Fachlicher Ansprechpartner Telekom Security psa.telekom.de	Gültigkeitsdauer 01.07.2025 - 30.06.2030	Freigegeben von Stefan Pütz, Leiter SEC-T-TST

Zusammenfassung

Unter die Bezeichnung „Client-Anwendung“ fällt jede Anwendung/Applikation oder auch kurz allgemein „App“ genannt, welche auf mobiler oder Desktop Hardware ausgeführt wird. Die Ausführung kann dabei autark oder in Kommunikationsbeziehung mit einem Backend nach dem Client-Server-Modell stattfinden.

Copyright © 2025 by Deutsche Telekom AG.

Alle Rechte, auch die des auszugsweisen Nachdrucks, der fotomechanischen Wiedergabe (einschließlich Mikrokopie) sowie der Auswertung durch Datenbanken oder ähnliche Einrichtungen, vorbehalten.

Inhaltsverzeichnis

1.	Einleitung	4
2.	Allgemeine Sicherheitsanforderungen für Apps	5
3.	Anforderungen in der Planungs- und Designphase	10
4.	Anforderungen in der Entwicklungsphase	17
5.	Installations- und Updateprozesse	21
6.	Anforderungen an die Deinstallation	22
7.	Allgemeine Anforderungen	23
7.1.	Schutz der Verfügbarkeit und Integrität	23
7.2.	Schutz von Sitzungen	23
7.3.	Authentisierungsmerkmal Passwort	24
7.4.	Protokollierung	29

1. Einleitung

Dieses Dokument wurde auf Basis der Vorgaben aus den im Konzern gültigen Sicherheitsrichtlinien erstellt. Die Sicherheitsanforderung dient u.a. als Grundlage zur Freigabe im PSA-Prozess. Sie dient darüber hinaus als Umsetzungsempfehlung in Einheiten, die nicht am PSA-Verfahren teilnehmen. Diese Anforderungen müssen bereits während der Planungs- und Entscheidungsprozesse berücksichtigt werden. Bei der Umsetzung der Sicherheitsanforderung ist das jeweils vorrangige nationale, internationale und supranationale Recht zu beachten.

2. Allgemeine Sicherheitsanforderungen für Apps

Req 1	Eingesetzte Software muss aus vertrauenswürdigen Quellen bezogen und auf Integrität geprüft werden.
-------	---

Die auf dem System eingesetzte Software muss aus vertrauenswürdigen Quellen bezogen und vor der Installation auf ihre Integrität geprüft werden.

Diese Anforderung bezieht sich auf sämtliche Arten von Software:

- Firmware sowie Microcode für Hardware-Komponenten
- Betriebssysteme
- Software-Bibliotheken
- Applikationen und Anwendungssoftware
- vorintegrierte Anwendungslösungen, wie z. B. Software Appliances oder Container

sowie weitere Software, die ggf. zum Einsatz kommt.

Vertrauenswürdige Quellen

Als vertrauenswürdige Bezugsquellen gelten allgemein:

- die offiziellen Vertriebs- und Bereitstellungs Kanäle des Herstellers
- dritte Distributoren, sofern sie durch den Hersteller autorisiert und legitimer Bestandteil von dessen Bereitstellungs Kanälen sind
- Internet-Downloads, sofern diese von offiziellen Bereitstellungs-Servern des Herstellers oder autorisierter Distributoren erfolgen

(1) Bietet der Bereitstellungs-Server verschiedene Formen von Downloads an, müssen durch Verschlüsselung bzw. kryptografische Signaturen geschützte Downloads jenen ohne einen solchen Schutz vorgezogen werden.

(2) Sichert der Bereitstellungs-Server die Transport-Schicht mittels kryptografischer Protokolle (z. B. HTTPS, sftp), müssen die zugehörig präsentierten Server-Zertifikate oder Server-Keys/Fingerprints bei jedem Download validiert werden, um die Identität des Bereitstellungs-Servers zu bestätigen; schlägt die Validierung fehl, ist der Download abzubrechen und der Bereitstellungs-Server als nicht vertrauenswürdige Quelle zu bewerten.

Integritätsprüfung

Die Integritätsprüfung soll sicherstellen, dass die erhaltene Software frei von Manipulationen und Schadsoftware-Befall ist. Sofern vorhanden, müssen zur Prüfung die vom Hersteller implementierten Mechanismen verwendet werden.

Valide Mechanismen sind:

- physikalische Siegel bzw. fest aufgebrachte Echtheitszertifikate (bei Bereitstellung der Software auf physikalischen Datenträgern)
- Abgleich von kryptografischen Hash Werten (z. B. SHA256, SHA512) der erhaltenen Software gegen Soll-Werte, die der Hersteller separat benennt
- Prüfung von kryptografischen Signaturen (z. B. GPG, Zertifikate), mit denen der Hersteller seine Software versieht

Zusätzlich ist eine Überprüfung der Software mittels eines Anti-Virus- bzw. Anti-Malware Scanners empfohlen (falls der Hersteller keinen der zuvor erwähnten Integritätsschutz-Mechanismen für seine Software implementiert hat, ist diese Überprüfung verpflichtend).

Erweiterte Integritätsprüfung bei Bezug von Software aus öffentlichen Registries

Öffentliche Registries ermöglichen es Entwicklern, beliebige eigene Software-Projekte zur Nutzung bereitzustellen. Die Bandbreite umfasst sowohl Projekte von namhaften Anbietern mit kontrollierten Entwicklungsprozessen als auch

von kleineren Anbietern oder Amateur-Entwicklern.
Beispiele für solche Registries sind:

- Code Registries (z. B. GitHub, Bitbucket, SourceForge, Python Package Index)
- Container Registries (z. B. Docker Hub)

Software aus öffentlichen Registries muss vor dem Einsatz einer erweiterten Integritätsprüfung unterzogen werden. Zusätzlich zu den im vorherigen Abschnitt beschriebenen Bestandteilen der Integritätsprüfung soll die erweiterte Prüfung explizit sicherstellen, dass die Software ihre Funktion tatsächlich wie beschrieben erfüllt, keine inhärenten Sicherheits-Risiken wie etwa gezielt implementierte Malware-Funktionen enthält und nicht von bekannten Sicherheits-Schwachstellen betroffen ist. Sofern die Software direkte Abhängigkeiten zu dritten Software-Projekten aufweist (sog. "Dependencies", sehr typisch in Open Source Software), die zum Einsatz der Software ebenfalls bezogen und installiert werden müssen, sind diese in die erweiterte Integritätsprüfung mit einzubeziehen.

Methoden für eine erweiterte Integritätsprüfung können beispielsweise sein:

- strikte Validierung der Projekt-/Paket-Namen (Vermeidung der Verwechslung mit maliziösen, gezielt nachgeahmten Software-Projekten)
- dynamische Code Analyse / strukturierte Funktionsprüfung in einer Test-Umgebung
- statische Code Analyse mittels eines Linters (z. B. Splint, JSLint, Pylint)
- Untersuchung mittels eines Sicherheitsschwachstellen-Scanners (z. B. Qualys, Nessus)
- Untersuchung mittels eines Container Security Scanners (z. B. JFrog Xray, Harbor, Clair, Docker Scan)
- Untersuchung mittels eines SCA (Software Composition Analysis) Tools bzw. Dependency-Scanners (z. B. OWASP Dependency Check, Snyk)

Die Prüfungsmethoden sind passend zur exakten Form der Software-Bereitstellung (Quellcode, Binaries/Artefakte, Container) auszuwählen und angemessen zu kombinieren.

Motivation: Softwarelieferketten enthalten diverse Angriffsvektoren. Ein Angreifer kann an verschiedenen Stellen ansetzen, um Software zu manipulieren oder gezielt eigene Routinen einzubringen und die Zielumgebung, in der die Software anschließend zum Einsatz gebracht wird, zu schädigen oder unter seine Kontrolle zu bringen. Der Angriff kann auf dem Transport- bzw. Übertragungsweg geschehen oder auch auf die Bereitstellungsquelle selbst ausgeübt werden. Entsprechend erleichtert wird ein Angriff, falls Software nicht über offizielle und kontrollierte Quellen bezogen wird oder eine Integritätsprüfung ausgelassen wird.

Besonderes Risiko zeigt sich für Software, die aus öffentlichen Registries bezogen wird, da diese jedermann für die Bereitstellung von Software-Projekten offenstehen. Perfide Angriffsmethoden sind inzwischen bekannt, in denen der Angreifer zunächst eine Zeitlang sicherheitstechnisch völlig unauffällige, funktionale Software bereitstellt und sobald diese sich etabliert und eine gewisse Verbreitung gefunden hat, in zukünftigen Versionen gezielt versteckten Schadcode integriert. Andere Methoden setzen auf ähnlich klingende Projektnamen zu weit verbreiteten Bestandsprojekten oder überstimmende Versionsnummern, um manipulierte Software in darauf aufbauende Lösungen zu injizieren.

Umsetzungsbeispiel: Bezug der Software über die offiziellen Bereitstellungskanäle des Herstellers. Nach Erhalt der Software umgehend Prüfung auf Integrität mittels kryptografischer Checksummen, wie vom Hersteller bereitgestellt, sowie Freiheit von bekannter Schadsoftware mittels Anti-Malware- / Anti-Virus-Scanner. Ablage der geprüften Software auf einer internen, geschützten Dateiablage und weitere Nutzung (z. B. Rollout auf die Zielsysteme) nur noch von dort.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Modifikation von Daten
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-2/9.0

Req 2 Bekannte Schwachstellen in der Software oder Hardware des Systems müssen behoben oder gegen Missbrauch abgesichert werden.

Bekannte Schwachstellen in Software- und Hardware Komponenten müssen mittels Installation von verfügbaren Systemaktualisierungen des Herstellers (z. B. Patches, Updates/Upgrades) behoben werden. Alternativ ist der Einsatz von Workarounds (Akut-Lösungen, welche die Schwachstelle nicht beheben, aber die Ausnutzung wirksam verhin-

dern) zulässig. Workarounds sollten grundlegend nur temporär Verwendung finden und sobald möglich durch eine reguläre Systemaktualisierung ersetzt werden, um die Schwachstellen vollständig zu schließen.

Komponenten, die bekannte, nicht behebbare Schwachstellen enthalten, dürfen in einem System nicht eingesetzt werden.

Die Behandlung von neu bekannt werdenden Schwachstellen muss zudem für die gesamte Einsatzphase des Systems fortlaufend sichergestellt und in den kontinuierlichen Betriebsprozessen eines Security Patch Managements umgesetzt werden.

Motivation: Der Einsatz von Komponenten ohne eine Behebung enthaltener Schwachstellen erhöht deutlich das Risiko einer erfolgreichen Kompromittierung. Der Angreifer wird zusätzlich dadurch begünstigt, dass i. d. R. nicht nur detaillierte Informationen zu bereits bekannt gewordenen Schwachstellen offen verfügbar sind, sondern oft auch bereits angepasste Angriffswerkzeuge, die eine aktive Ausnutzung erleichtern.

Umsetzungsbeispiel: Im Anschluss an die initiale Installation eines Betriebssystems von einem offiziellen Installationsmedium werden alle aktuell verfügbaren Patches und Sicherheits-Updates eingespielt.

Zusätzliche Informationen:

Primäre Quellen für bereits bekannt gewordene Schwachstellen in einer Software/Hardware sind Auflistungen in den Versionshinweisen sowie die Sicherheitshinweise aus den offiziellen Meldekanälen des Herstellers oder unabhängiger CERTs. Insbesondere die Meldekanäle werden sinnvollerweise fest in kontinuierliche Prozesse des Security Patch Managements für ein System eingebunden, damit neu bekanntwerdende Schwachstellen zeitnah registriert und in operative Behebungsmaßnahmen geführt werden können.

Als ergänzende Maßnahme zur Aufdeckung von potenziell noch enthaltenen, grundsätzlich bereits bekannt gewordenen Arten von Schwachstellen können gezielte Schwachstellenuntersuchungen des Systems durchgeführt werden. Zu diesem Zweck eignen sich besonders spezialisierte Werkzeuge wie automatisierte Schwachstellen-Scanner. Beispiele hierfür sind: Tenable Nessus, Qualys Scanner Appliance.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Störung der Verfügbarkeit
- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-10/9.0

Req 3	Software-Komponenten, für die es keine Wartung oder Pflege durch den Lieferanten, Hersteller oder Entwickler gibt, dürfen nicht verwendet werden.
-------	---

Es dürfen auf einem System nur Middleware- und Anwendungs-Software eingesetzt werden, für die ein Support durch Lieferanten, Hersteller, Entwickler (dies kann auch eine Open Source Community sein) oder anderen Vertragspartnern der Deutschen Telekom AG besteht. Komponenten die End-of-Life oder End-of-Support sind dürfen nicht eingesetzt werden. Ausgenommen hiervon sind Komponenten für die ein spezieller Support Vertrag abgeschlossen wurde, durch den auch über den Lebenszyklus des Produktes hinaus die Behebung von Sicherheitsschwachstellen gewährleistet ist.

Motivation: Software-Komponenten, die End-of-Life oder End-of-Support sind, stellen ein Risiko für ein System dar. Dies bedeutet, dass beim Auftreten von Schwachstellen in einer Komponente keine Aktualisierungen (z. B. Update, Patch) mehr bereit gestellt werden. Hieraus resultiert, dass möglicherweise bekannt gewordene Schwachstellen nicht behoben werden, so dass diese von einem Angreifer ausgenutzt werden können, um das entsprechende System zu kompromittieren oder um es in seiner Verfügbarkeit zu beeinträchtigen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten

- Unberechtigte Modifikation von Daten
- Störung der Verfügbarkeit

Für diese Anforderung sind folgende Gewährleistungsziele relevant:

- Verfügbarkeit

ID: 3.54-3/7.0

Req 4 Für die Laufzeitumgebung vorhandene Best Practices zur Security müssen eingehalten werden.

Best-Practice-Empfehlungen sind Lösungsvorschläge für wiederkehrende Probleme. Dabei wird erklärt, wie – bezogen auf die jeweilige Laufzeitumgebung – Sicherheitsprobleme vermieden und Sicherheitsfunktionen (z. B. das Sandbox-Prinzip) genutzt werden können. Außerdem werden Hinweise gegeben, wie typische Programmierfehler, die zu Sicherheitsschwachstellen führen können, vermieden werden.

Empfehlungen zur Vermeidung von Schwachstellen:

Verwendung sicherer Bibliotheksfunktionen.

Eingabedaten einer App, insbesondere solche, die nicht aus einer vertrauenswürdigen Quelle stammen, sollten immer hinsichtlich ihrer Länge und Kodierung überprüft und nicht als Format-Strings verwendet werden, um Buffer-Overflow-, Null-Byte-Poisoning-, Format-String- oder ähnliche Angriffe zu verhindern. Im Fall von Apps zählen zu nicht-vertrauenswürdigen Datenquellen neben Nutzereingaben auch Teile einer URL, SMS- oder E-Mail-Inhalte, Umgebungsvariablen, lokal gelesene Daten (z. B. von einer SD-Card) oder von einem Server empfangene Daten. Es muss sichergestellt sein, dass Eingabedaten nur als Daten und nicht als Programmcode interpretiert werden.

Buffer-Overflows vermeiden.

Zur Vermeidung von Buffer-Overflows, insbesondere in maschinennahen Programmiersprachen wie bspw. C, C++ oder Objective-C, dürfen keine Bibliotheksfunktionen ohne Längensvalidierung verwendet werden. Selbstgeschriebener Code, der Strings oder Speicherinhalte verarbeitet, muss die Länge der verarbeiteten Daten explizit überprüfen.

Race-Conditions vermeiden.

Fast alle Betriebssysteme und Laufzeitumgebungen, auch für kleinere Endgeräte, unterstützen heutzutage Multitasking und Multithreading, d. h. sie ermöglichen die parallele Verarbeitung mehrerer Prozesse. Der Nachteil davon ist, dass es keine Garantie dafür gibt, dass zwei aufeinanderfolgende Operationen in einem Programm hintereinander ausgeführt werden, ohne dass nicht andere Operationen in der Zwischenzeit ausgeführt werden. Dies kann zu Fehlern und im schlimmsten Fall zu Sicherheitsschwachstellen in einer App führen.

Motivation: Im Allgemeinen ist in Best-Practice-Empfehlungen nicht nur viel Security-Know-How, sondern auch Erfahrung aus zurückliegenden Fehlern und Sicherheitsproblemen eingeflossen. Anleitungen und beispielhafter Quellcode zeigen auch, wie die Security-Features der Laufzeitumgebung genutzt werden können.

Ohne Längensvalidierung oder bei Verwendung unsicherer Bibliotheksfunktionen können einfache Programmierfehler zu Sicherheitslücken führen, durch die vertrauliche Informationen kompromittiert oder Sicherheitsmechanismen der App oder der Laufzeitumgebung (Sandbox) ausgehebelt werden. Auch in Java-Apps (z. B. Android oder OSGi), können solche Sicherheitslücken entstehen, wenn Eingabedaten ungeprüft über eine Java-Bibliotheksfunktion an Low-Level-Funktionen übergeben werden. Bei der Verwendung von JNI in Java-Apps ist besondere Vorsicht geboten.

Buffer-Overflows sind eine der Hauptursachen zur Ausführung von applikationsfremden Code. Die Methoden zum Jail-breaking oder Rooting von Endgeräten nutzen i. d. R. Buffer-Overflows in Teilen des Betriebssystems eines Endgeräts aus. Durch eine Buffer-Overflow-Schwäche in einer App kann ein ggf. vorhandenes Sandbox-Modell der Laufzeitumgebung umgangen und fremder Code im Kontext der App ausgeführt werden.

Race Conditions sind nicht nur schwer auffindbare Fehler, sondern können heutzutage Gründe für ernstzunehmende Sicherheitslücken sein. Zur Vermeidung können Semaphoren bzw. Flags genutzt werden, d. h. Variablen werden Signalzustände zugeordnet, die sie für Änderungen sperren oder freigeben.

Umsetzungsbeispiel: Apple iOS: http://developer.apple.com/library/ios/#documentation/Security/Conceptual/SecureCodingGuide/Introduction.html#//apple_ref/doc/uid/TP40002415

ID: 3.54-4/7.0

Req 5	Die Berechtigungen von Benutzern und Anwendungen müssen auf den zur Erfüllung ihrer Aufgaben benötigten Umfang beschränkt werden.
-------	---

Die Berechtigungen auf einem System müssen so weit eingeschränkt werden, dass ein Benutzer nur auf Daten zugreifen und Funktionen nutzen kann, die er im Rahmen seiner Arbeit benötigt. Entsprechende Berechtigungen sind auch für den Zugriff auf Dateien zu vergeben, die Bestandteil des Betriebssystems oder von Anwendungen sind oder von diesen erzeugt werden (z. B. Konfigurations- und Protokollierungsdateien).

Neben dem Zugriff auf Daten muss auch die Ausführung von Anwendungen und deren Bestandteilen mit möglichst niedrigen Berechtigungen erfolgen. Anwendungen sollten nicht mit Administrator- oder Systemberechtigungen ausgeführt werden.

Motivation: Werden einem Benutzer zu weitreichende Berechtigungen auf einem System gewährt, kann er in einem Umfang auf Daten und Anwendungen zugreifen, der zur Erfüllung der zugewiesenen Aufgaben nicht erforderlich ist. Dies erzeugt ein unnötig erhöhtes Risiko im Falle eines Missbrauchs, insbesondere auch, falls der Benutzer oder dessen Benutzerkonto durch einen Angreifer kompromittiert werden.

Anwendungen, die über zu weitreichende Berechtigungen verfügen, können von einem Angreifer missbraucht werden, um unbefugte Zugänge zu erhalten oder zu erweitern und damit einen Zugriff auf schutzbedürftige Daten und Systembereiche zu erlangen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen

ID: 3.01-23/9.0

Req 6	Die Applikation muss mit unprivilegierten Benutzerrechten auskommen, um auf dem Endgerät lauffähig zu sein.
-------	---

Jailbreaking und Rooting von mobilen Endgeräten erlaubt es Entwicklern, die Hardware und Systemressourcen des Endgeräts vollständig auszunutzen. Dadurch wird die Sicherheit der Apps und Informationen auf dem Endgerät gefährdet, da i. d. R. das Sandbox-Modell der Laufzeitumgebung ausgehebelt und ein Vollzugriff auf das Betriebssystem und die Hardware des Endgeräts möglich wird. Eine App soll daher in einem nicht privilegierten Benutzerkonto lauffähig sein. Die Nutzung von privilegierten Benutzerkonten ermöglicht es Entwicklern, die Funktionen des Betriebssystems voll auszuschöpfen. Dem Nutzer soll jedoch die Möglichkeit gegeben werden, die App in einem nicht privilegierten Konto zu betreiben, da dies sicherer ist. Andererseits könnte die Installation Administrationsrechte erfordern, die angefordert werden können.

Motivation: Durch Jailbreaking und Rooting werden bestehende Sicherheitsmechanismen umgangen. Die gesamte Laufzeitumgebung wird unsicher und die darin laufenden Apps und gespeicherten Informationen sind Gefährdungen ausgesetzt. Sicherheitslücken in einer PC-App, die mit Administrationsrechten läuft, können die Sicherheit des Betriebssystems kompromittieren. Daher sollte der Nutzer die Möglichkeit haben, die App in einem nicht privilegierten Konto zu betreiben.

ID: 3.54-6/7.0

3. Anforderungen in der Planungs- und Designphase

Req 7	Benutzerkonten müssen mit mindestens zwei Authentisierungsmerkmalen aus unterschiedlichen Faktoren geschützt werden.
-------	--

Sämtliche Benutzerkonten des Systems müssen vor einer unberechtigten Nutzung geschützt werden. Dies ist durch Verwendung von Authentisierungsmerkmalen umzusetzen, die eine eindeutige Identifizierung des legitimen Benutzers ermöglichen.

Authentisierungsmerkmale werden in übergeordneten Gruppen, sogenannten "Faktoren", differenziert:

- Faktor WISSEN: "etwas, das nur der legitime Benutzer weiss"
- Faktor BESITZ: "etwas, das nur der legitime Benutzer besitzt"
- Faktor INHÄRENZ: "etwas, das nur der legitime Benutzer ist"

Es müssen mindestens zwei Authentisierungsmerkmale miteinander kombiniert werden.

Diese müssen aus unterschiedlichen Faktoren stammen ("Multi-Faktor-Authentisierung"; kurz: "MFA").

Eine Kombination von Authentisierungsmerkmalen desselben Faktors (Beispiel: zwei unterschiedliche PINs) ist nicht zulässig.

Beispiele für zulässige Authentisierungsmerkmale

- aus dem Faktor WISSEN: PINs, Passphrases (Passwörter*)
- aus dem Faktor BESITZ: Smartcards, FIDO2 Level 2 Security Keys, Device-bound Passkeys, in einem TPM/HSM gespeicherte kryptografische Schlüssel, OTPs (One-Time Passwords) generiert in einer zum Einsatz in der Deutschen Telekom freigegebenen Authenticator App
- aus dem Faktor INHÄRENZ: biometrische Merkmale wie Fingerabdrucke, Iris- bzw. Retina-Scans, Gesichtserkennung

*) Wichtiger Hinweis zum Einsatz von Passwörtern

Passwörter müssen als Authentisierungsmerkmal strikt vermieden werden.

Sie sind nur dann zulässig, falls sicherere Alternativen nicht umgesetzt werden können.

Grundsätzlich nicht zulässig sind Authentisierungsmerkmale, die leicht gefälscht oder manipuliert werden können.

Dazu gehören:

- Rufnummern und Anrufer-IDs
- IP-Adressen
- geografische Standorte
- VPN-Zugehörigkeiten und Netzwerkinformationen
- Status von Clients, etwa von Arbeitsplatzsystemen oder mobilen Endgeräten

Hinweis zu technischen Benutzerkonten

Für den maschinellen Zugriff eingesetzte technische Benutzerkonten (auch unter dem Begriff "Machine-to-Machine" bzw. "M2M"-Konten bekannt) sind von der Verpflichtung zur Multi-Faktor-Authentisierung ausgenommen, da eine solche dort i.d.R. nicht umsetzbar ist.

Motivation: Benutzerkonten, die nicht über angemessene Authentisierungsmerkmale geschützt sind, können von einem Angreifer missbraucht werden, um unberechtigt auf ein System und die darauf gespeicherten Daten und Anwendungen zuzugreifen. Aufgrund vielfältiger gängiger Angriffsmethoden ist der Schutz von Benutzerkonten durch ein einzelnes Authentisierungsmerkmal mittlerweile als schwach zu bewerten, insbesondere, falls hier das klassische Passwort zum Einsatz kommt. Eine Kombination von Authentisierungsmerkmalen aus unterschiedlichen Faktoren erhöht die Sicherheit für Benutzerkonten signifikant.

MFA Kombination	Beschreibung	Anmerkung zur Anwendung
Telekom "myCard" plus PIN-Eingabe	1. Faktor: in Hardware implementiertes Zertifikat ("Besitz") 2. Faktor: PIN ("Wissen")	(keine)
FIDO2 Level 2 Security Key plus Fingerabdruck	1. Faktor: in Hardware gespeicherter Crypto-Schlüssel ("Besitz") 2. Faktor: Biometrisches Merkmal ("Inhärenz")	Die Biometrie-Funktion muss vorab konfiguriert werden, da der Biometrie-Sensor sonst nur als "Taste" funktioniert und somit der 2. Faktor ("Inhärenz") nicht gegeben wäre.
Device-bound Passkey in WH4B* auf Nucleus PC plus PIN	1. Faktor: in Hardware gespeicherter Crypto-Schlüssel ("Besitz") 2. Faktor: PIN ("Wissen")	(keine)
Device-bound Passkey in Secure Enclave auf iOS Device plus FaceID	1. Faktor: in Hardware gespeicherter Crypto-Schlüssel ("Besitz") 2. Faktor: Biometrisches Merkmal ("Inhärenz")	(keine)
Passwort plus freigegebene OTP App (z. B. MS Authenticator)	1. Faktor: Passwort ("Wissen") 2. Faktor: spezifisch generiertes One Time Password ("Besitz")	Passwörter sind nur dann zulässig, falls sicherere Alternativen nicht umgesetzt werden können. Die Synchronisation der Authenticator App über einen Clouddienst sowie das Teilen (z. B. über einen Passwortmanager) des OTP ist nicht zulässig.

*) WH4B: "Windows Hello for Business"

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen

ID: 3.01-20/9.0

Req 8	Falls die Anwendung Passwörter zur Benutzerauthentisierung verwendet, muss sie diese bei der Speicherung schützen, indem ausschließlich Passwort-Hashes gespeichert werden. Das Hashing der Passwörter muss mit einem Passwort-Hashing-Verfahren wie bcrypt, bcrypt oder Argon2 erfolgen oder, wenn dies nicht möglich ist, durch die Verwendung einer kryptographischen Hashfunktion mit Salt.
-------	---

Motivation: Durch eine sichere Speicherung wird verhindert, dass Benutzerpasswörter kompromittiert und missbräuchlich verwendet werden. Selbst bei einem erfolgreichen Angriff kann ein Angreifer lediglich die Passwort-Hashes abgreifen. Durch die Wahl eines geeigneten Hashverfahrens und bei ausreichend langen Passwörtern ist zudem der Aufwand, die Passwörter aus den erbeuteten Passwort-Hashes durch Brute Forcing zu ermitteln, so hoch,

dass dies nicht in annehmbarer Zeit erfolgen kann. Passwort-Hashing-Verfahren werden mit dem Ziel entwickelt, das Hashing möglichst aufwändig zu gestalten und so Brute-Force-Angriffe erheblich zu verlangsamen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System

ID: 3.54-8/7.0

Req 9	Für sicherheitsrelevante Protokollierungsdaten in Anwendungen, die lokal erfasst werden, müssen geltende Aufbewahrungszeiträume und Löschfristen eingehalten werden, falls die Anwendung durch die Telekom betrieben/gemanagt wird.
-------	---

Aus Sicht der IT-Sicherheit wird auf einem System die lokale Erfassung von sicherheitsrelevanten Protokollierungsdaten nicht zwingend gefordert. Da lokal erfasste Protokollierungsdaten bei Systemstörungen beschädigt oder von einem erfolgreichen Angreifer gezielt manipuliert werden können, sind sie für sicherheitstechnische bzw. forensische Analysen nur bedingt belastbar. Entsprechend ist für die IT-Sicherheit relevant, dass Protokollierungsdaten an einen separaten Log Server weitergeleitet werden.

Eine lokale Erfassung kann dennoch erfolgen; beispielsweise falls bei der Erzeugung der Protokollierungsdaten eine lokale Speicherung aufgrund technischer Abläufe zunächst unabdingbar ist oder es begründete betriebliche Interessen gibt, Protokollierungsdaten auch lokal vorzuhalten.

Folgende Grundregelung ist bei einer lokalen Erfassung von Protokollierungsdaten zu berücksichtigen:

- sicherheitsrelevante Protokollierungsdaten müssen für einen Zeitraum von 90 Tagen aufbewahrt werden. [1*]
- nach Ablauf von 90 Tagen muss eine umgehende Löschung von gespeicherten Protokollierungsdaten erfolgen.

[1*] Diese Anforderung gilt nur, falls auf dem System keine zusätzliche Weiterleitung an einen separaten Log Server umgesetzt ist und Protokollierungsdaten daher ausschliesslich lokal erfasst werden.

Abweichungen

Abweichende Aufbewahrungszeiträume und Löschfristen können durch gesetzliche oder regulatorische Anforderungen (insbesondere im Zusammenhang mit personenbezogenen Daten) bestehen oder durch vertragliche Vereinbarungen definiert sein. In diesen Fällen sind die geltenden Zeiträume individuell mit einem Project Security Manager (PSM) bzw. Datenschutzberater (DSB) abzustimmen oder werden durch diesen vorgegeben.

Motivation: Protokollierungsdaten sind ein immens wichtiges Werkzeug der IT Sicherheit zur Abwendung, Detektierung und Aufklärung von Systemstörungen, Sicherheits- und Datenschutzvorfällen. Auf der anderen Seite unterliegt die Erfassung von Protokollierungsdaten wie jede andere Datenverarbeitung auch gesetzlichen und regulatorischen Auflagen. Entsprechend müssen Richtlinien eingehalten werden, die beides in Einklang bringen.

Umsetzungsbeispiel: Unter Berücksichtigung der gültigen Gesetzeslage und zutreffenden Datenschutz-Regularien werden folgende Löschfristen für lokal erfasste sicherheitsrelevante Protokollierungsdaten auf einem beispielhaften TK-System umgesetzt:

- Standard System Logs: Löschung nach spätestens 90 Tagen
- Protokollierungen von öffentlichen IP-Adressen: Löschung (oder Anonymisierung) nach spätestens 7 Tagen
- Protokollierungen der Vergabe von dynamischen öffentlichen IP Adressen durch die TK-Lösung: Löschung nach spätestens 7 Tagen
- Protokollierungen nicht abrechnungsrelevanter Call Detail Records: Löschung nach spätestens 7 Tagen
- Protokollierungen der Kommunikations-Inhalte von eMail und SMS: Löschung nach spätestens 24 Stunden
- Protokollierungen der behandelten Domain-Abfragen des DNS-Servers der TK-Lösung: Löschung nach spätestens 24 Stunden

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe

ID: 3.54-9/7.0

Req 10	Falls ein dauerhaftes Login benötigt wird, dann müssen OAuth-2.0-Refresh-Token oder ein vergleichbarer Mechanismus verwendet werden.
--------	--

OAuth 2.0 ist der Entwurf eines offenen Protokolls, das eine standardisierte, sichere Authentifizierung und Autorisierung z.B. auch für mobile Apps erlaubt. Der Nutzer kann mit Hilfe dieses Protokolls einer App den nutzerbezogenen Zugang zu Backend-Systemen ermöglichen, ohne sein Passwort offen zu legen oder eine vollständige Zugangsbe-
rechtigung zu seinen Daten zu erteilen.

Motivation: OAuth-2.0-Refresh-Tokens oder vergleichbare Mechanismen (allg. „Security Tokens“) bieten einen größeren Schutz bei einem dauerhaften Login: Security-Tokens haben eine begrenzte Gültigkeit und einen eng begrenzten Anwendungsbereich. Sie gelten i. d. R. nur für die Authentifizierung gegenüber dem Backend-System, für das sie ausgestellt wurden, und die Berechtigungen für den Zugriff können auf die Funktionen der App eingeschränkt werden. Darüber hinaus können sie ungültig gemacht werden („Revocation“), z. B. wenn der Nutzer das Endgerät verliert. Bei OAuth-Refresh-Tokens ist zusätzlich der Token-Tausch-Mechanismus hervorzuheben: Bei jeder Nutzung, d. h. bei jedem Login-Vorgang gegenüber dem Backend-System verliert das aktuelle Token seine Gültigkeit und wird durch ein neues ersetzt. Falls beispielsweise ein Token kompromittiert wurde, dann wird es spätestens beim nächsten Login-Vorgang automatisch ungültig, ohne dass etwa ein Passwort geändert werden müsste.

Umsetzungsbeispiel: OAuth 2.0 RFC: <https://tools.ietf.org/html/rfc6749> (Letzer Aufruf: 01.10.2019)

ID: 3.54-10/7.0

Req 11	Falls eine App die Möglichkeit bietet, ein Login dauerhaft zu speichern (z. B. mit OAuth-Refresh-Tokens), so muss diese Option für den Benutzer an- und abwählbar sein.
--------	---

Die Option, ein Login dauerhaft zu speichern, sollte in der Standardeinstellung nicht aktiv sein.

Motivation: Sicherheitsbewussten Nutzern und solchen, die sich ein Endgerät mit anderen Nutzern teilen, muss die Möglichkeit gegeben werden, die Speicherung des nutzerbezogenen Logins zu unterbinden.

ID: 3.54-11/7.0

Req 12	Die Nutzung von schutzbedürftigen Funktionen des Systems sowie der Zugriff auf interne oder vertrauliche Daten darf nicht ohne vorhergehende Authentifizierung und Autorisierung ermöglicht werden.
--------	---

Die Nutzung von schutzbedürftigen Funktionen des Systems sowie der Zugriff auf Daten, die als intern oder vertraulich eingestuft sind, darf erst nach einer eindeutigen Identifizierung und erfolgreichen Authentifizierung des Benutzers anhand eines Benutzerkontos ermöglicht werden. Ein Benutzerkonto besteht aus einem eindeutigen Benutzernamen und zugehörigen Authentisierungsmerkmalen. Zudem muss verifiziert werden, dass der Benutzer innerhalb der ihm im System zugeordneten Benutzerrolle für den Zugriff auf die betroffenen Funktionen und Daten legitimiert ist.

Eine Ausnahme hiervon sind Funktionen und Daten, die uneingeschränkt öffentlich genutzt werden dürfen; beispielsweise der Bereich einer Web-Seite im Internet, in dem nur öffentliche Informationen bereitgestellt werden.

Beispiele für Funktionen, die eine vorhergehende Authentifizierung erfordern:

- Fernzugriff auf Netzdienste (wie SSH, SFTP, Web-Dienste)
- lokaler Zugriff auf die Management-Konsole
- lokale Nutzung von Betriebssystem und Anwendungen

Diese Anforderung gilt ohne Einschränkung auch für jegliche maschinellen Zugriffe auf das System (Hier erfolgt die Umsetzung in der Regel durch den Einsatz von sogenannten "technischen Benutzerkonten", auch "Machine-2-Machine" oder "M2M"-Konten genannt).

Motivation: Die eindeutige Authentifizierung und Autorisierung von Zugriffen auf ein System sind elementar, um Funktionen und Daten vor Missbrauch zu schützen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen

ID: 3.01-19/9.0

Req 13	Ein Bezahl- oder Buchungsvorgang innerhalb einer App muss durch eine zusätzliche Eingabe des Authentisierungsmerkmals bestätigt werden.
--------	---

Bei einem Bezahlprozess innerhalb der App („In-App-Payment“) oder bei der Buchung zusätzlicher Produkte oder Produktoptionen darf die Bezahl- bzw. Buchungsaktion erst dann ausgelöst werden, wenn der Benutzer seine Identität durch eine zusätzliche Eingabe des Passworts oder eines anderen Geheimnisses, das nur der Benutzer kennt, nachgewiesen hat. Der Bezahl- oder Buchungsprozess darf nicht anhand eines dauerhaften Logins einem Benutzer zugeordnet werden.

Motivation: So ist gewährleistet, dass nicht ein illegitimer Benutzer Kauf- oder Buchungsaktionen mit Hilfe eines gefundenen oder gestohlenen Endgeräts durchführen kann. Der legitime Benutzer kann den Kauf oder die Buchung nicht ohne Weiteres abstreiten.

ID: 3.54-13/7.0

Req 14	Das System muss es Benutzern ermöglichen, sich von ihrer laufenden Sitzung abzumelden.
--------	--

Das System muss über eine Funktion verfügen, die es dem angemeldeten Benutzer ermöglicht, sich jederzeit abzumelden. Die Wiederaufnahme einer abgemeldeten Sitzung darf nicht ohne erneute und erfolgreiche Authentifizierung des Benutzers möglich sein.

Motivation: Ein Benutzer muss die vollständige Kontrolle über von ihm aufgebaute Sitzungen behalten, um jederzeit situationsgerecht seinen Zugriff auf ein System zu beenden und damit über diesen Zugriff exponierte Daten und Funktionen schützen zu können. Zudem muss der Benutzer davon ausgehen können, dass durch ihn gezielt beendete Sitzungen nicht im Anschluss durch unberechtigte Dritte wieder aufgenommen und weitergeführt werden können.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen

ID: 3.01-17/9.0

Req 15	Falls eine App, die nutzerbezogene Zugangsinformationen (z. B. Security-Tokens für Backend-Systeme) dauerhaft speichert, dann muss diese über eine Logout-Funktion verfügen, mit der gespeicherte Zugangsinformationen gelöscht werden (Logout).
--------	--

Die Nutzer müssen die Möglichkeit haben, das Speichern eines Logins rückgängig zu machen und ein vollständiges Logout durchzuführen. Die vollständige Löschung des Endgeräte-Speichers („Factory-Reset“) oder das Löschen der App ist zu aufwendig, falls ein Nutzer nur einen gespeicherten Login rückgängig machen möchte.

Motivation: Ein Nutzer könnte z. B. ein Endgerät mit einem anderen Nutzer teilen, diesem aber nicht den Zugang zu seinen Informationen im Backend zur Verfügung stellen wollen. Oder ein sicherheitsbewusster Nutzer, der ein Login versehentlich gespeichert hat, möchte dies rückgängig machen.

ID: 3.54-15/7.0

Req 16	In Ausgaben und Meldungen dürfen keine Informationen über interne Strukturen des Systems preisgegeben werden.
--------	---

Informationen über die internen Strukturen eines Systems einschließlich der dort verwendeten Komponenten und entsprechende Implementationsdetails gelten grundsätzlich als schutzbedürftig.

Im Allgemeinen betrifft dies Informationen über

- Produktnamen und Produkt-Identifizierer implementierter System-Komponenten
- eingesetzte Betriebssysteme, Middleware, Backend-Software, Software-Bibliotheken und interne Applikationen sowie deren Software-Versionen
- installierte Service-Packs, Patches, Hotfixes
- Seriennummern von Komponenten sowie hinterlegte Produkt-Lizenzen
- Datenbankstrukturen

Typische Beispiele für Ausgaben und Meldungen, in denen eine Preisgabe solcher System-Informationen potenziell auftreten kann:

- Anmelde-Fenster und -dialoge
- Fehlermeldungen
- Statusausgaben
- Banner von aktiven Netzwerk-Diensten
- System-Protokolle und Log Dateien
- Debug-Protokolle, Stack Traces

Soweit es technisch umsetzbar ist, ohne die Funktion und den Betrieb des Systems zu beeinträchtigen, muss die Ausgabe betroffener System-Informationen grundsätzlich deaktiviert sein.

Ein Zugriff auf betroffene System-Informationen darf nur dazu autorisierten Benutzern des Systems möglich sein. In der Regel ist der Kreis autorisierter Benutzer auf Administratoren und Betreiber des Systems zu beschränken. Ebenfalls zulässig ist ein Zugriff für autorisierte Monitoring- und Inventarisierungs-Systeme innerhalb der Einsatzumgebung.

Eine zulässige Ausnahme dieser Einschränkungen besteht für spezifische einzelne System-Informationen, deren Preisgabe für die vorgesehene Funktion des Systems in Verbindung mit dritten Systemen technisch zwingend erforderlich ist; beispielsweise die Präsentation unterstützter Protokolle und deren Versionen während der initialen Parameter-Aushandlung in Sitzungs-Aufbauten zwischen einem Client und einem Server.

Motivation: Informationen über die internen Strukturen eines Systems können von einem Angreifer genutzt werden, um Angriffe auf das System äußerst effektiv vorzubereiten. Beispielsweise kann ein Angreifer aus der Software-Version eventuell bekannte Schwachstellen eines Produktes ableiten, um diese gezielt während des Angriffs auf das System auszunutzen.

Umsetzungsbeispiel: [Beispiel 1]

Deaktivierung der Anzeige des Produktnamens und der installierten Version in den ausgelieferten Fehler-Webseiten eines eingesetzten Web-Servers.

[Beispiel 2]

Entfernung des Produktnamens und des zugehörigen Versions-Strings aus dem Login-Banner eines eingesetzten SSH Servers.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-9/9.0

Req 17	Falls schutzbedürftige Daten mittels des HTTP-Protokolls übertragen werden, muss die POST-Methode für Requests verwendet werden.
--------	--

Stellt der Webservice einer REST-API sicher, dass wiederholte Requests immer den gleichen Effekt auf eine bestimmte URI haben, kann auch PUT als Ersatz zu POST verwendet werden. Die GET-Methode darf nur verwendet werden, um Inhalte vom Server anzufordern. Zu schutzbedürftigen Daten gehören auch Identifier wie z.B. Session-IDs oder auch die MSISDN eines Mobilfunkteilnehmers, die nicht über GET-Requests übertragen werden dürfen.

Motivation: Bei der POST-Methode werden Parameter im HTTP-Request-Body übertragen. Dadurch wird sichergestellt, dass keine schutzbedürftigen Daten von Webservern, Security Information and Event Management-Systemen (SIEM) und Clients protokolliert werden oder innerhalb der URL leicht ablesbar sind. Bei der GET-Methode werden Parameter über den HTTP-Header übertragen: Die Parameter sind in der URL sichtbar und können von Servern und Clients protokolliert werden. Diese Protokollierung findet oft in Zonen statt, die für die notwendige Schutzklasse nicht freigegeben sind. Es besteht zudem die Möglichkeit, dass eine URI (URL oder URN) über das Referrer-Feld im HTTP-Header dem Server und anderen Anwendungen zur Verfügung gestellt wird. Diese Information kann z.B. zur Auflistung von Link-Listen sowie zum Logging usw. verwendet werden.

Umsetzungsbeispiel: OWASP REST Security Cheat Sheet (Letzter Aufruf: 04.10.19): https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen

ID: 3.54-17/7.0

4. Anforderungen in der Entwicklungsphase

Req 18 Für kryptographische Operationen müssen Standardbibliotheken verwendet werden.

Um Seitenkanalattacken und Fehler bei der Software-Implementierung von kryptographischen Operationen vorzubeugen, müssen die Standardbibliotheken der Betriebssysteme oder quelloffene Bibliotheken verwendet werden. Quelloffene Bibliotheken sind Eigenentwicklungen vorzuziehen. Falls quelloffene Bibliotheken verwendet werden, sollten diese nur aus vertrauenswürdigen Quellen bezogen und hinsichtlich ihrer Softwarelizenzen geprüft werden. Darüber hinaus muss der Support sichergestellt sein. Bibliotheken jeglicher Art mit bekannten Schwachstellen dürfen nicht verwendet werden. Sicherheitspatches sind zeitnah nach deren Verfügbarkeit einzuspielen.

Motivation: Die Updatezyklen von Standardbibliotheken der Betriebssysteme sowie öffentlicher Bibliotheken (z.B. OpenSSL) sind oft sehr kurz und Sicherheitsschwachstellen können schnell behoben werden. Öffentliche Bibliotheken werden meist einem kontinuierlichen Review-Prozess durch die Community unterzogen. Sicherheitspatches stehen nach dem Fund der Schwachstelle schnell bereit. Beispiel: Heartbleed in OpenSSL - <https://de.wikipedia.org/wiki/Heartbleed>

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Störung der Verfügbarkeit
- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.54-18/7.0

Req 19 Für die verschlüsselte Speicherung vertraulicher Informationen müssen individuelle (d. h. für die jeweilige App-Instanz, Endgerät oder Benutzer gebildete) Krypto-Schlüssel verwendet werden.

In Abhängigkeit von der Art der vertraulichen Informationen müssen nutzer- oder endgerätespezifische oder sowohl nutzer- als auch endgerätespezifische Krypto-Schlüssel verwendet werden. Für verschiedene Instanzen der App auf unterschiedlichen Endgeräten darf nicht der gleiche Schlüssel verwendet werden, insbesondere dann nicht, wenn damit nutzerbezogene vertrauliche Informationen unterschiedlicher Nutzer verschlüsselt werden. Auch bei einer Speicherung vertraulicher Informationen unterschiedlicher Nutzer auf demselben Endgerät darf nicht der gleiche Schlüssel verwendet werden (der Schlüssel muss in diesem Fall nutzerspezifisch sein).

Motivation: Werden vertrauliche Informationen auf unterschiedlichen Endgeräten mit demselben Schlüssel verschlüsselt gespeichert, dann genügt einem Angreifer die Kenntnis eines Schlüssels, um auch die Informationen auf anderen Endgeräten zu entschlüsseln. Wenn auf demselben Endgerät verschlüsselte Informationen unterschiedlicher Nutzer mit demselben Schlüssel verschlüsselt gespeichert sind, dann könnte ein Nutzer bei Kenntnis „seines“ Schlüssels auch die vertraulichen Informationen anderer Nutzer entschlüsseln.

ID: 3.54-19/7.0

Req 20 Krypto-Schlüssel dürfen auf dem Gerät nur dann gespeichert werden, wenn die Laufzeitumgebung dafür einen sicheren Datenspeicher zur Verfügung stellt. Andernfalls müssen Krypto-Schlüssel jeweils zur Laufzeit deterministisch berechnet werden und dabei einen beim ersten Start oder der Installation der App hinterlegten Faktor verwenden.

Manche Laufzeitumgebungen bieten sichere Speicher für Krypto-Schlüssel und Zugangsdaten (Keychain ab iOS 5.x, Security-Account-Manager, Secure-Local-Storage, ...). Sofern vorhanden sollen diese Mechanismen zur sicheren Speicherung von Schlüsseln und Security-Tokens benutzt werden. Falls die Ausführungsumgebung keinen sicheren Schlüsselspeicher bereitstellt, dann dürfen Krypto-Schlüssel nicht gespeichert werden. Stattdessen müssen die

Schlüssel während der Laufzeit (Applikationsausführung) aus geeigneten Daten deterministisch abgeleitet werden (z.B. Hardware-IDs o.ä.). Die Nutzung eindeutiger IDs trägt dazu bei, dass unterschiedliche Instanzen der App (auf unterschiedlichen Geräten) nicht denselben Schlüssel ableiten. Da solche Daten aber auch für andere Applikationen oder für einen Angreifer zugreifbar sind, muss die Ableitung eines Krypto-Schlüssels in der Anwendung geheim gehalten und, falls möglich, obfuskiert werden, damit ein Reverse Engineering der Schlüsselableitung erschwert wird. Die Ableitung des Krypto-Schlüssels muss auch einen weiteren Faktor ("Master-Schlüssel") beinhalten, der während der Installation (z.B. mit Hilfe eines Zufallszahlengenerators) erzeugt und dann auf dem Gerät gespeichert wird. Darüber hinaus sollte ein vom Benutzer eingegebene Passwort oder eine Passphrase bei der Ableitung des Krypto-Schlüssels verwendet werden (in Abhängigkeit vom Nutzungs-Szenario und dem Wert der Daten, die die App verarbeitet).

Motivation: Vorteile der Verwendung eines sicheren Speichers: Die sicheren Speicher werden i.d.R. durch Mechanismen des Betriebssystems vor unberechtigtem Zugriff geschützt und bieten damit einen guten Schutz z. B. vor Schadsoftware. Sofern das Betriebssystem dem Nutzer die Möglichkeit bietet, das Endgerät mit einem Zugangspasswort zu schützen, wird der sichere Speicher i. d. R. auch mit diesem Passwort geschützt. Falls kein sicherer Speicher verwendet wird: Verschlüsselte Informationen sind nur so sicher wie der zur Verschlüsselung benutzte Schlüssel. Ist der Schlüssel bekannt, dann stellt auch eine verschlüsselte Speicherung keinen Schutz mehr vor unberechtigtem Zugriff auf vertrauliche Informationen dar.

ID: 3.54-20/7.0

Req 21	Falls das HTTP-Protokoll zur verschlüsselten Übertragung von Informationen verwendet wird, dann muss TLS ab Version 1.2 verwendet werden.
--------	---

Bei TLS handelt es sich um ein allgemein als sicher anerkanntes Protokoll zur verschlüsselten Datenübertragung, das unter dem Namen TLS (Transport Layer Security) von der IETF standardisiert wurde. TLS 1.0 und 1.1 werden allgemein als unsicher betrachtet, da sie die Verwendung veralteter Algorithmen und Hashfunktionen wie SHA-1 und MD5 erlauben. Zudem gibt es Angriffe wie BEAST und POODLE die effektiv gegen TLS v1.0 und v1.1 eingesetzt werden. Es muss demnach TLS ab der Version 1.2 verwendet werden. Insbesondere bei der Übertragung über das HTTP-Protokoll, das üblicherweise von Apps verwendet wird, bietet sich TLS an (HTTPS). Aber auch für die Übertragung von und zu Mailservern mit den Protokollen SMTP, POP3 oder IMAP gibt es Standards, diese Protokolle über TLS zu verwenden. Die Verwendung von standardisierten und allgemein als sicher anerkannten Verschlüsselungsprotokollen bietet den größtmöglichen Schutz vor fehlerhaften Implementierungen oder protokollinhärenten Sicherheitslücken bei der verschlüsselten Übertragung. Die Auswahl der richtigen Ciphersuites ist dem Anforderungsdokument "Kryptographische Algorithmen" zu entnehmen und bei einer Abweichung davon mit dem PSM zu besprechen.

Motivation: Die Benutzung von standardisierten Verschlüsselungsprotokollen, welche allgemein als sicher gelten, bietet einen optimalen Schutz vor Fehlimplementierungen und Fehlern, insbesondere während einer verschlüsselten Übertragung.

ID: 3.54-21/7.0

Req 22	Vor der Übertragung vertraulicher Informationen an Server oder an andere Instanzen müssen diese authentisiert werden.
--------	---

Bei der Übertragung vertraulicher Informationen darf das Ziel der Übertragung nicht nur anhand einer Netzwerkidentifikation (z. B. Domainname, IP-Adresse, Bluetooth-MAC, ...) identifiziert werden, sondern muss zusätzlich authentisiert werden. Zur Authentisierung des Ziels können X.509-Zertifikate (bei TLS oder IPSec), Pre-Shared Secrets oder die bei einem vorausgehenden Pairing gebildeten Schlüssel (bei Bluetooth) dienen.

Motivation: Wird das Ziel einer Übertragung nicht zuverlässig identifiziert und authentisiert, können vertrauliche Informationen z. B. durch Man-in-the-Middle-, ARP-Request-Poisoning- oder DNS-Spoofing-Angriffe fälschlicherweise an einen Angreifer übertragen werden. Eine verschlüsselte Übertragung wäre in diesen Fällen nutzlos, da ein Sitzungsschlüssel mit dem Angreifer ausgehandelt worden wäre.

ID: 3.54-22/7.0

Req 23	Falls TLS-Server-Zertifikate verwendet werden, müssen diese von einer App auf Gültigkeit und Revocation-Status überprüft werden. Bei Ungültigkeit oder Verbindungs-Timeout zum Revocation- oder OCSP-Server muss der Verbindungsaufbau abgebrochen und der Benutzer informiert werden.
--------	--

Wird das Ziel bei der verschlüsselten Übertragung vertraulicher Informationen mit TLS durch ein X.509 Zertifikat authentisiert, muss das Zertifikat auf Gültigkeit geprüft werden. Dabei muss die Zertifikatskette bis zu einer vertrauenswürdigen Wurzelzertifizierungsinstanz (Root-CA) nach RFC5280, geprüft werden. Die in RFC6818, RFC8389 und RFC8399 definierten Updates des RFC5280 müssen ebenfalls beachtet und umgesetzt werden.

Ein Zertifikat ist dann gültig, wenn

- der Zeitpunkt der Gültigkeitsprüfung innerhalb des Gültigkeitszeitraums des Zertifikats liegt,
- die Zertifikatskette auf eine vertrauenswürdige Wurzelzertifizierungsinstanz (Root-CA) zurückzuführen ist (d. h. das Zertifikat muss von einer Zertifizierungsinstanz ausgestellt worden sein, deren Zertifikat wiederum mittelbar oder unmittelbar von einer vertrauenswürdigen Wurzelzertifizierungsinstanz ausgegeben wurde),
- die Signatur der Zertifizierungsinstanz korrekt ist und
- das Zertifikat von der Zertifizierungsinstanz (CA) nicht zurückgezogen wurde (Revocation). Um zu prüfen, ob ein Zertifikat zurückgezogen wurde, muss eine aktuelle CRL (Certificate Revocation List) der CA abgerufen, ihre Signatur geprüft und die Seriennummer des Zertifikats mit der CRL verglichen werden. Alternativ kann OCSP (Online Certificate Status Protocol) nach RFC6960 und OCSP stapling nach RFC6961 dafür verwendet werden. Alle Zertifikate müssen entlang der Zertifikatskette gültig sein (dabei ist der Zeitpunkt der Ausstellung eines Zertifikats maßgeblich für die Prüfung des Gültigkeitszeitraums des zugehörigen CA-Zertifikats: das CA-Zertifikat muss zum Zeitpunkt der Ausstellung des Zertifikats gültig gewesen sein). ##Wenn OCSP genutzt wird, darf das Verhalten "fail gracefully" nur nach Absprache mit dem Project Security Manager und unter Berücksichtigung des Schutzbedarfs der von der Applikation verarbeiteten Daten implementiert werden.

Anmerkung zur Gültigkeitsprüfung: Die meisten Laufzeitumgebungen vereinfachen die Gültigkeitsprüfung von Zertifikaten, indem sie entsprechende Bibliotheken oder APIs zur Verfügung stellen. Aber auch bei der Verwendung solcher bereitgestellter Bibliotheken oder APIs muss sichergestellt sein, dass alle Schritte der Gültigkeitsprüfung durchgeführt werden. Unter Umständen müssen mehrere API-Funktionen in der entsprechenden Reihenfolge aufgerufen werden. Weitere Informationen finden sich in den Security-How-Tos oder Best-Practices zur entsprechenden Laufzeitumgebung.

Motivation: Durch die Prüfung können kompromittierte oder abgelaufene Server-Zertifikate erkannt werden. Im schlimmsten Fall werden kompromittierte Zertifikate einer Zertifizierungsstelle (CA-Zertifikate) dazu benutzt, illegitime Server-Zertifikate auszustellen (siehe den Fall der DigiNotar CA im Jahr 2011). Eine lückenlose Gültigkeitsprüfung des Zertifikats und der Zertifizierungskette kann solche Angriffsszenarien verhindern. Ein weiteres Angriffsszenario wird ermöglicht, falls "fail gracefully" bei der Verwendung von OCSP erlaubt ist, d. h., falls keine Antwort vom OCSP-Responder (Server) kommt, wird angenommen, das Zertifikat sei weiterhin gültig: In diesem Fall kann ein Angreifer dafür sorgen, dass keine Antwort vom OCSP-Responder an den Client gelangt und so dafür sorgen, dass ein kompromittiertes Zertifikat weiterhin akzeptiert wird.

ID: 3.54-23/7.0

Req 24	Falls ein TLS-Server-Zertifikat verwendet wird und dieses als ungültig erkannt wurde, so muss die Verbindung unterbrochen werden.
--------	---

Bei einem Man-in-the-Middle- oder Impersonation-Angriff (z. B. DNS-Spoofing) könnte ein kompromittiertes oder abgelaufenes Server-Zertifikat dazu verwendet werden, den Anschein zu erwecken, dass es sich bei einem Server des Angreifers um einen legitimen Server handelt, an den vertrauliche Informationen übertragen werden sollen.

Motivation:

ID: 3.54-24/7.0

5. Installations- und Updateprozesse

Req 25 Apps müssen über (halb-) automatische Update-Verfahren verfügen.

Einige Laufzeitumgebungen bieten automatische Update-Mechanismen, die auch die entsprechenden Verteilplattformen (Google Play Store, AppStore, Microsoft Store, Marketplace, Corporate App Store für Geschäftskunden) nutzen. Sind keine automatischen Update-Mechanismen vorhanden, dann muss eine App in regelmäßigen Zeitabständen prüfen, ob eine neue Version verfügbar ist und diese nach einer entsprechenden Bestätigung des Nutzers automatisch herunterladen und installieren.

Motivation: Fehler und Sicherheitslücken können durch Updates behoben und beseitigt werden. Ein automatischer Update-Mechanismus oder entsprechende Hinweise und Aufforderungen an den Nutzer, gewährleistet, dass Updates zügig verbreitet und alte, fehleranfällige und unsichere Versionen schnell ersetzt werden.

ID: 3.54-25/7.0

6. Anforderungen an die Deinstallation

Req 26	Die Deinstallation muss so gestaltet sein, dass vertrauliche Nutzerdaten und zugehörige anwendungsspezifische Informationen aus der Laufzeitumgebung, vom Endgerät, aus Sicherheitsmodulen und von anderen Speichermedien gelöscht werden.
--------	--

Entscheidet sich ein Nutzer zu einer Deinstallation der App, müssen vertrauliche Nutzerdaten und zugehörige App-spezifische Daten wie z. B. die Krypto-Schlüssel endgültig gelöscht werden. In einigen Fällen übernimmt das Betriebssystem bzw. die Laufzeitumgebung das Löschen aller Anwendungsdaten bei der Deinstallation und die App selbst hat keinen Einfluss auf den Deinstallationsprozess; in solch einem Fall ist diese Anforderung nicht zutreffend. In anderen Fällen wollen Nutzer, dass ihre Daten auch nach der Installation gespeichert bleiben, z.B. bei der Speicherung von Mediendateien auf der SD-Card. Wenn der Nutzer sich über dieses Umstands bewusst ist und diese Daten auch manuell löschen kann, dann können diese Daten gespeichert bleiben. Krypto-Schlüssel und Security-Tokens müssen jedoch gelöscht und überschrieben werden, falls die App Einfluss auf die Deinstallation hat.

Motivation: Die endgültige Löschung von vertraulichen Nutzerdaten, Krypto-Schlüsseln und Security-Tokens bei der Deinstallation der App verhindert eine nachträgliche Kompromittierung vertraulicher Informationen, die auf einem Endgerät zurückgelassen wurden.

ID: 3.54-26/7.0

7. Allgemeine Anforderungen

7.1. Schutz der Verfügbarkeit und Integrität

Req 27	Das System muss robust gegenüber unerwarteten Eingaben implementiert werden.
--------	--

An das System übergebene Daten müssen vor der Weiterverarbeitung zunächst validiert werden, um sicherzustellen, dass die Daten dem erwarteten Daten-Typ und -Format entsprechen. Dies soll das Risiko einer Manipulation von Systemabläufen und -Zuständen durch entsprechend konstruierte Dateninhalte eliminieren. Eine Validierung ist für jegliche Daten durchzuführen, die an das System übergeben werden. Beispiele sind Benutzereingaben, Werte in Datenfeldern und Protokollinhalte.

Die folgenden typischen Implementierungsfehler müssen vermieden werden:

- Fehlende Validierung der Länge von übergebenen Daten
- Unzutreffende Annahmen über das Format von Daten
- Fehlende Validierung empfangener Daten auf Konformität mit der Spezifikation
- Unzureichende Behandlung von Protokollabweichungen in empfangenen Daten
- Unzureichende Begrenzung der Rekursion beim Parsing komplexer Datenformate
- Unzureichende Umsetzung von Whitelisting oder Escaping zum Schutz vor Eingaben außerhalb des validen Wertebereiches

Motivation: Ein Angreifer kann mittels gezielt konstruierter Dateninhalte versuchen, ein System, das empfangene Daten vor der internen Weiterverarbeitung nicht ausreichend validiert, in einen instabilen Zustand zu versetzen oder nicht autorisierte Aktionen innerhalb des Systems auszulösen. Das Schadenspotential entsprechender Angriffe ist individuell systemabhängig, besitzt jedoch eine theoretische Bandbreite von unkontrollierten Systemabstürzen bis hin zu einer kontrollierten Ausführung von eigens eingeschleustem Code und der daraus resultierenden vollständigen Kompromittierung eines Systems.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Störung der Verfügbarkeit
- Unbemerkt durchführbare Angriffe

ID: 3.01-11/9.0

7.2. Schutz von Sitzungen

Req 28	Sitzungen müssen gegen eine unautorisierte Übernahme ("Session Hijacking") geschützt werden.
--------	--

In Schnittstellen, die Funktionen für Sitzungen zum System bereitstellen, müssen technische Maßnahmen implementiert werden, die verhindern, dass die Sitzung eines legitimen Benutzers von einem nicht autorisierten Dritten übernommen und weitergeführt werden kann.

Ein derartiger Schutz kann beispielsweise durch die Umsetzung einer für das spezifische System sinnvollen Kombination aus den folgenden Optionen erreicht werden:

- Auf der Transportschicht: Verwendung des TCP-Protokolls (mit dessen Sequenz-Nummern) und entsprechenden Filterlisten
- Auf der Sitzungsschicht: Verwendung des TLS-Protokolls
- Auf der Applikationsschicht: Aushandlung eines zufälligen geheimen Sitzungsschlüssels zwischen Sender und Empfänger zwecks Autorisierung allen Sitzungsverkehrs (z. B. Session-ID, Session Cookie, Session Token)

- Nutzung kryptographischer Verfahren zum Schutz von Sitzungsschlüsseln vor Ausspähung oder Veränderung

Motivation: Ungeschützte Sitzungen können potenziell von einem Angreifer gekapert und weitergeführt werden, um darüber im Kontext des betroffenen Benutzers unberechtigte Zugriffe auf das System auszuüben.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen

ID: 3.01-16/9.0

Req 29	Sitzungen müssen nach einer dem Verwendungszweck angepassten Zeit der Inaktivität automatisch beendet werden.
--------	---

Es ist notwendig, dass Sitzungen auf einem System nach einer vorgegebenen Zeit der Inaktivität automatisch beendet werden.

Aus diesem Grund muss ein Time-Out für Sitzungen gesetzt werden. Der hier zu wählende Zeitraum richtet sich nach der Verwendung des Systems und ggf. nach der physischen Umgebung. So ist beispielsweise der Time-Out für eine Anwendung, die in einer ungesicherten Umgebung genutzt wird, kürzer (wenige Minuten) zu wählen als der Time-Out für eine Anwendung, die von Betriebsmitarbeitern für System-Monitoring-Aufgaben in einem zugangsgeschützten Bereich genutzt wird (60 Minuten und länger).

Motivation: Für eine offene, aber ungenutzte Sitzung besteht ein Risiko, dass ein nicht legitimer Benutzer diese unbemerkt übernimmt und weiterführt, um im Namen des betroffenen Benutzers einen nicht autorisierten Zugriff auf das System und enthaltene Daten auszuüben.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen

ID: 3.01-18/9.0

7.3. Authentisierungsmerkmal Passwort

Req 30	Falls Passwörter als Authentisierungsmerkmal genutzt werden, müssen diese mindestens 12 Zeichen lang sein und drei der folgenden Zeichenkategorien beinhalten: Kleinbuchstaben, Großbuchstaben, Ziffern und Sonderzeichen.
--------	--

Ein System darf nur Passwörter akzeptieren, welche die folgende Komplexität erfüllen:

- Mindestlänge 12 Zeichen
- Bestehend aus mindestens drei der folgenden vier Zeichenkategorien:
 - Kleinbuchstaben
 - Großbuchstaben
 - Ziffern
 - Sonderzeichen

Die maximal mögliche Länge für Passwörter sollte 25 Zeichen oder mehr betragen. Dies unterstützt Endbenutzer bei Bedarf mit mehr Freiraum, um individuelle merkbare Passwörter bilden zu können, und hilft so, unerwünschte Verhal-

tensweisen im Umgang mit Passwörtern zu vermeiden, welche durch stark eingeschränkte maximale Passwortlängen provoziert werden können.

Das System muss bei der Vergabe eines Passwortes sicherstellen, dass das Passwort diesen Bildungsregeln entspricht; vorzugsweise mittels technischer Maßnahmen, falls solche aber nicht umsetzbar sein sollten, ersatzweise mittels organisatorischer Maßnahmen.

Bei Verwendung eines zentralen Systems für die Authentisierung der Benutzer [vgl. Wurzel-Anforderungsdokument "[i]3.69 IAM (Identity Access Management) - Framework"] darf diese Aufgabe an das zentrale System weitergereicht bzw. dorthin ausgelagert werden.

Zulässige Abweichung in der Passwort-Mindestlänge

Unter sicherheitsbezogenen Kriterien lassen sich für ein System potenziell Bedingungen identifizieren, die eine Reduzierung der Passwort-Mindestlänge ermöglichen:

- Allgemein zulässig ist eine Reduzierung der Passwort-Mindestlänge für Systeme, die innerhalb des Authentisierungsprozess zusätzlich zum Passwort weitere unabhängige Authentisierungsmerkmale verwenden (Umsetzung von 2-Faktor oder Multi-Faktor Authentisierung).
- Eine Reduzierung der Passwort-Mindestlänge muss grundsätzlich durch einen fachlichen Sicherheitsberater (z. B. ein PSM der Telekom Security) individuell bewertet und als zulässig bestätigt werden. In der Bewertung müssen umgebende technische, organisatorische und gesetzliche Rahmenparameter berücksichtigt sowie der systemspezifische Schutzbedarf und die potenzielle Schadenshöhe bei Sicherheitsvorfällen einbezogen werden.
- Der absolute Mindestwert von 8 Zeichen Länge darf für Passwörter grundsätzlich nicht unterschritten werden.

Motivation: Passwörter mit der genannten Komplexität bieten eine zeitgemäße Robustheit gegen Angriffe bei gleichzeitig akzeptabler Benutzerfreundlichkeit. In der Praxis haben sich Passwörter mit dieser Komplexität bewährt. Triviale und zu kurze Passwörter sind anfällig für Brute-Force- und Wörterbuch-Angriffe und von einem Angreifer leicht zu ermitteln. Ein ermitteltes Passwort kann dann von einem Angreifer für den unautorisierten Zugriff auf das System und dessen Daten genutzt werden

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen

ID: 3.01-26/9.0

Req 31	Falls Passwörter als Authentisierungsmerkmal genutzt werden, muss eine Änderung des eigenen Passwortes jederzeit durch den Benutzer möglich sein.
--------	---

Das System muss eine Funktion bieten, die es einem Benutzer ermöglicht, sein Passwort jederzeit zu ändern.

Bei der Verwendung eines zentralen Systems darf die Funktion auf dieses System weitergereicht oder ausgelagert werden.

Motivation: Nur durch die Möglichkeit, dass ein Benutzer sein Authentisierungsmerkmal jederzeit selbsttätig ändern kann, ist ein zeitnahe Änderung möglich, wenn der Benutzer den Verdacht hat, dass dieses in den Besitz Dritter gelangt sein könnte.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen

- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-29/9.0

Req 32	Falls Passwörter als Authentisierungsmerkmal genutzt werden, muss nach spätestens 12 Monaten ein Passwort-Wechsel erfolgen.
--------	---

Die maximal zulässige Nutzungsdauer für Passwörter beträgt 12 Monate.
Erreicht ein Passwort die maximal zulässige Nutzungsdauer, muss es geändert werden.

Das System muss hierzu den Benutzer automatisch bei dessen nächster Systemanmeldung auf die abgelaufene Nutzungsdauer hinweisen und ihn unmittelbar durch einen Passwort-Änderungsdialog führen. Der Zugriff auf das System darf ohne eine erfolgreich abgeschlossene Passwort-Änderung nicht weiter zugelassen werden.

Für technische Benutzerkonten (M2M bzw. Machine-2-Machine), die für die Authentifizierung und Autorisierung von Systemen untereinander oder von Anwendungen auf einem System genutzt werden, müssen ebenfalls automatisierte Lösungen zur Einhaltung der zulässigen Nutzungsdauer für Passwörter implementiert werden.

Falls eine automatisierte Abbildung des Prozesses zur Passwort-Änderung nicht umsetzbar ist, muss ersatzweise eine wirksame organisatorische Maßnahme angewendet werden, die eine verbindliche manuelle Passwort-Änderung bei Ablauf der zulässigen Nutzungsdauer sicherstellt.

Motivation: Passwörter sind im Gegensatz zu moderneren Authentisierungsmerkmalen leichter angreifbar. Ohne spezifische Maßnahmen einer verlässlichen technisch automatisierten Kompromittierungserkennung kann das Risiko, dass ein Passwort von einem Angreifer ermittelt oder gebrochen wird, über die Zeit deutlich steigen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-30/9.0

Req 33	Falls Passwörter als Authentisierungsmerkmal verwendet werden, muss die Wiederverwendung bereits eingesetzter Passwörter unterbunden werden.
--------	--

Für jedes Benutzerkonto muss eine Historie der bisher genutzten Passwörter aufgezeichnet werden. Wird eine Passwort-Änderung für ein Benutzerkonto initiiert, ist das neue Passwort gegen diese Passwort-Historie abzugleichen. Bei erkannter Wiederverwendung eines Passworts muss die Passwort-Änderung abgelehnt werden. Die Umsetzung dieses Validierungsvorgangs muss im System auf Basis von technischen Maßnahmen erfolgen. Bei Verwendung eines zentralen IAM Systems für die Authentisierung der Benutzer darf die Umsetzung an das zentrale IAM System weitergereicht bzw. dorthin ausgelagert werden [vgl. Wurzel-Anforderungsdokument "3.69 IAM (Identity Access Management) - Framework"].

Generell soll mittels der Passwort-Historie sichergestellt werden, dass ein schon einmal verwendetes Passwort niemals erneut einsetzbar ist.

Aufgrund technischer Limitierungen kann eine Passwort-Historie jedoch in vielen IT/NT-Produkten nicht uneingeschränkt aufgezeichnet werden. In diesem Fall ist die folgende Grundregelung zu berücksichtigen:

- ein bereits genutztes Passwort darf für den Zeitraum von mindestens 60 Tagen nicht erneut einsetzbar sein (gemessen ab dem Zeitpunkt, an dem das betroffene Passwort durch ein anderes ersetzt wurde)
- in Systemen, in denen der Zeitraum von mindestens 60 Tagen nicht abgebildet werden kann, muss der längstmögliche Zeitraum konfiguriert werden. Zusätzlich muss durch einen Project Security Manager (PSM) bestätigt werden, dass der konfigurierte Zeitraum im Gesamtkontext des Systems noch ausreichend im Sinne der Sicherheitsanforderung ist.

Anmerkung:

Einige IT/NT-Produkte bieten keinen technischen Konfigurationsparameter an, mit dem die Passwort-Historie direkt an einen Zeitraum gebunden werden kann, sondern ermöglichen nur, die Anzahl der aufzuzeichnenden Passwörter zu definieren. In solchen Fällen darf der Zeitraum alternativ durch eine Verknüpfung der folgend beschriebenen, i. d. R. breit verfügbaren Konfigurationsparameter sichergestellt werden. Innerhalb der resultierenden Policy kann ein Benutzer nur einmal pro Tag sein Passwort ändern und aufgrund der Anzahl der aufgezeichneten Passwörter effektiv nach frühestens 60 Tagen ein altes Passwort erneut einsetzen.

- Minimum Passwort-Alter: 1 Tag
- Passwort-Historie: Aufzeichnung der letzten 60 benutzten Passwörter

In dieser Umsetzungsvariante ist zu beachten, dass das Minimum Passwort-Alter nicht mehr als einen Tag betragen sollte, um den Benutzer nicht unangemessen hinsichtlich der grundsätzlichen Notwendigkeit einzuschränken, das Passwort jederzeit selbstständig ändern zu können.

Motivation: Benutzer favorisieren in der Regel Passwörter, die sie sich gut merken können und verwenden diese in Folge oft wiederkehrend über lange Zeiträume, falls das System dies zulässt. Aus Sicht des Benutzers ist das Verhalten nachvollziehbar, führt aber effektiv zu einer erheblichen Reduzierung der Schutzwirkung dieses Authentisierungsparameters. Mit etwa entsprechenden Kenntnissen über den Benutzer oder erbeuteten Informationen aus zurückliegenden System-Kompromittierungen kann ein Angreifer so Zugriff auf vermeintlich geschützte Benutzerkonten erlangen. Insbesondere auch in Situationen, in denen im Rahmen einer akuten Risikobehandlung zentral neue Initialpasswörter vergeben werden, die Benutzer diese jedoch unmittelbar aus Bequemlichkeit auf ein vorheriges Passwort zurückändern, bestünde hohe Gefahr, dass ein Angreifer seinen illegalen Zugriff erneut aufnehmen kann. Es ist daher wichtig zu vermeiden, dass Benutzer alte Passwörter erneut einsetzen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-28/9.0

Req 34	Falls Passwörter als Authentisierungsmerkmal genutzt werden, muss ein Schutz gegen Online-Angriffe wie Wörterbuch- und Brute-Force-Attacken vorhanden sein, der das Erraten von Passwörtern stark erschwert.
--------	--

Online Wörterbuch- und Brute-Force Angriffe haben zum Ziel, durch automatisiertes Ausprobieren an einer regulären Zugangsschnittstelle des Systems die Passwörter von Benutzerkonten zu erraten.

Um dies zu verhindern, muss eine der folgenden Maßnahmen oder eine Kombination aus diesen umgesetzt werden:

- mittels technischer Maßnahmen abgebildete Erzwingung einer Wartezeit nach einem fehlgeschlagenen Anmeldeversuch, bevor eine erneute Anmeldung ermöglicht wird. Die Wartezeit sollte sich zudem mit jedem weiteren fehlgeschlagenen Anmeldeversuch automatisch deutlich erhöhen (z. B. durch Verdopplung).
- automatisiertes Sperren des Benutzerkontos nach einer vorgegebenen Anzahl von aufeinanderfolgenden fehlgeschlagenen Anmeldeversuchen (typischerweise 5). Allerdings ist bei dieser Lösung zu bedenken, dass diese einen Entsperrungs-Prozess erforderlich macht und ein Angreifer dies gezielt nutzen kann, um Konten zu sperren und somit vorübergehend unbrauchbar zu machen.
- Verwendung von CAPTCHA ("Completely Automated Public Turing test to tell Computers and Humans Apart"), um automatisierte Anmeldeversuche von Maschinen ("Robots" oder "Bots") bestmöglich zu unterbinden. Ein CAPTCHA stellt dazu eine kleine Aufgabe, i. d. R. basierend auf grafischen oder akustischen Elementen, die von einer Maschine nur schwer zu lösen ist. Es ist entsprechend zu berücksichtigen, dass CAPTCHA proble-

matisch bzgl. der Barrierefreiheit sein können.

Um eine höhere Sicherheit zu erreichen, ist es oftmals sinnvoll, zwei oder mehrere der genannten Maßnahmen miteinander zu kombinieren. Dies ist abhängig vom Einzelfall zu prüfen und entsprechend umzusetzen.

Motivation: Ohne entsprechenden Schutz kann ein Angreifer durch bloßes Durchprobieren von Wörterbuchlisten oder automatisch erzeugten Zeichenkombinationen versuchen ein Passwort zu ermitteln, um so das entsprechende Benutzerkonto missbräuchlich zu nutzen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen

ID: 3.01-25/9.0

Req 35	Falls Passwörter als Authentisierungsmerkmal genutzt werden, darf deren Darstellung während der Eingabe nicht im Klartext erfolgen.
--------	---

Passwörter dürfen während deren Eingabe nicht in lesbarem Klartext auf Bildschirmen oder sonstigen Ausgabegeräten dargestellt werden. Eine Darstellung während der Eingabe darf keine Rückschlüsse auf die tatsächlich im Passwort verwendeten Zeichen ermöglichen.

Diese Anforderung gilt für jegliche Arten von Eingabemasken und -feldern für Passwörter.

Beispiele hierfür sind Dialoge zur Passwort-Vergabe, zur Passwort-basierenden Anmeldung an Systemen oder zur Änderung von bestehenden Passwörtern.

Ausnahmeregelungen:

- Innerhalb eines Eingabefelds ist eine optionale Klartext-Darstellung eines Passworts zulässig, sofern diese Klartext-Darstellung einem validen Zweck dient, nur temporär anhält, dazu fallbezogen durch den legitimen Benutzer explizit zu aktivieren ist und von diesem umgehend auch wieder deaktiviert werden kann.
Ein valider Zweck wäre beispielsweise, dem legitimen Benutzer bei Bedarf eine unkomplizierte visuelle Überprüfung zu ermöglichen, sein Passwort in einem Anmelde-Dialog korrekt eingegeben zu haben, bevor er die Anmeldung letztendlich final vollzieht.
Eine solche optionale Klartext-Darstellung eines Passworts muss vollständig in der Kontrolle des legitimen Benutzers verbleiben, damit er über deren Aktivierung/Deaktivierung situationsgerecht entscheiden kann. In der Standard-Einstellung des Systems muss die Klartext-Darstellung deaktiviert sein.
- Das auf vielen mobilen Endgeräten (Smartphones) typische Verhalten, jedes einzelne Zeichen bei der Eingabe eines Passworts sehr kurz im Klartext anzuzeigen, um dem Benutzer die Eingabekontrolle zu erleichtern, ist dort grundlegend zulässig. Allerdings darf dabei niemals das vollständige Passwort im Klartext auf dem Bildschirm dargestellt werden.

Motivation: Im Fall einer Klartext-Darstellung besteht das Risiko, dass dritte Personen ein Passwort während der Eingabe zufällig oder gezielt über die Bildschirm-Ausgabe ausspähen können.

Umsetzungsbeispiel: In der Darstellung auf dem Bildschirm wird während der Eingabe eines Passworts jedes einzelne Zeichen einheitlich durch ein "*" ersetzt.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten

- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-31/9.0

7.4. Protokollierung

Req 36 Für sicherheitsrelevante Protokollierungsdaten, die an den Log Server weitergeleitet werden, muss die Einhaltung geltender Aufbewahrungszeiträume und Löschfristen sichergestellt werden.

Folgende Grundregelung ist zu berücksichtigen:

- sicherheitsrelevante Protokollierungsdaten müssen auf dem Log Server für einen Zeitraum von 90 Tagen aufbewahrt werden.
- nach Ablauf von 90 Tagen muss eine umgehende Löschung der auf dem Log Server gespeicherten Protokollierungsdaten erfolgen.

Abweichungen

Abweichende Aufbewahrungszeiträume und Löschfristen können durch gesetzliche oder regulatorische Anforderungen bestehen (insbesondere im Zusammenhang mit personenbezogenen Daten) oder durch vertragliche Vereinbarungen definiert sein. In diesen Fällen sind die geltenden Zeiträume individuell mit einem Project Security Manager (PSM) bzw. Datenschutzberater (DSB) abzustimmen oder werden durch diesen vorgegeben.

Log Server in Verantwortung eines Dritten

Sollte der ausgewählte Log Server nicht innerhalb derselben operativen Verantwortung liegen wie das Quellsystem der Protokollierungsdaten, muss sichergestellt werden, dass dem verantwortlichen Betreiber des Log Servers die gültigen Parameter für die zu empfangenden Protokollierungsdaten bekannt sind und durch diesen im Sinne der hier genannten Regelungen eingehalten werden.

Motivation: Protokollierungsdaten sind ein immens wichtiges Werkzeug der IT-Sicherheit zur Abwendung, Detektierung und Aufklärung von Systemstörungen, Sicherheits- und Datenschutzvorfällen. Auf der anderen Seite unterliegt die Erfassung von Protokollierungsdaten wie jede andere Datenverarbeitung auch gesetzlichen und regulatorischen Auflagen. Entsprechend müssen Richtlinien eingehalten werden, die beides in Einklang bringen.

Umsetzungsbeispiel: Unter Berücksichtigung der gültigen Gesetzeslage und zutreffenden Datenschutz-Regularien werden folgende Löschfristen für ausgeleitete sicherheitsrelevante Protokollierungsdaten eines beispielhaften TK-Systems auf dem separaten Log Server umgesetzt:

- Standard System Logs: Löschung nach spätestens 90 Tagen
- Protokollierungen von öffentlichen IP-Adressen: Löschung (oder Anonymisierung) nach spätestens 7 Tagen
- Protokollierungen der Vergabe von dynamischen öffentlichen IP-Adressen durch die TK-Lösung: Löschung nach spätestens 7 Tagen
- Protokollierungen nicht abrechnungsrelevanter Call Detail Records: Löschung nach spätestens 7 Tagen
- Protokollierungen der Kommunikations-Inhalte von E-Mail und SMS: Löschung nach spätestens 24 Stunden
- Protokollierungen der behandelten Domain-Abfragen des DNS-Servers der TK-Lösung: Löschung spätestens nach 24 Stunden

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe

ID: 3.01-36/9.0