

Sicherheitsanforderung

Microservice (Container basierter Web Service)

Deutsche Telekom Gruppe

Version	7.1
Datum	01.12.2025
Status	Freigegeben

Impressum

Herausgeber
Deutsche Telekom AG
Vorstandsbereich Technology & Innovation
Chief Security Officer

Reuterstrasse 65, 53315 Bonn
Deutschland

Dateiname	Dokumentennummer 3.84	Dokumententyp Sicherheitsanforderung
Version 7.1	Stand 01.12.2025	Status Freigegeben
Fachlicher Ansprechpartner Telekom Security psa.telekom.de	Gültigkeitsdauer 01.12.2025 - 30.11.2030	Freigegeben von Stefan Pütz, Leiter SEC-T-TST

Zusammenfassung

Unter Mikroservice ist typischerweise ein Softwarekomponente gemeint, die als Docker Container ein Application Programming Interface (API) in Form einer REST-API bereitstellt. Der Mikroservice interagiert nur mit anderen Services oder Systemen. Das Dokument fasst alle für diesen Mikroservice relevanten Anforderungen zusammen. Dadurch wird es ermöglicht, den Mikroservice mit genau einem SoC (Statement of Compliance) in einem SDK (Standardisierten Datenschutz- und Sicherheitskonzept) zu beschreiben.

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.	Container	5
3.	Application Server (Code Laufzeitumgebung)	7
4.	Web-Service	13
4.1.	Web-Service: allgemeine Anforderungen	13
4.2.	Web-Service: Authentifizierung und Autorisierung	15
4.3.	Web-Service: Validierung	17
4.4.	Web-Service: Payload Signaturen	18
4.5.	Web-Service: TLS (Transport Layer Security)	18
4.6.	Web-Service: REST	18
4.6.1.	API-Provider spezifische Anforderungen	22
4.6.2.	API-Consumer spezifische Anforderungen	25
4.6.3.	Spezialfall: API-Provider für Human to Machine (H2M, Mensch-zu-Maschine) Schnittstellen	25
5.	Protokollierung	27
6.	Web Services	33
6.1.	TLS	33

1. Einleitung

Unter Mikroservice ist typischerweise ein Softwarekomponente gemeint, die als Docker Container ein Application Programming Interface (API) in Form einer REST-API bereitstellt. Die Mikroservice interagiert nur mit anderen Services oder Systemen. Das Dokument fasst alle für diesen Mikroservice relevanten Anforderungen zusammen. Dadurch wird es ermöglicht, den Mikroservice mit genau einem SoC (Statement of Compliance) in einem SDSK (Standardisierten Datenschutz- und Sicherheitskonzept) zu beschreiben.

2. Container

Req 1 Container müssen als unveränderlich behandelt werden.

Container sollen unveränderlich sein, daher dürfen sie zur Laufzeit nicht verändert werden. Anstatt Container im Livebetrieb zu patchen, muss das Image gepatched und redeployed werden. Container Images müssen über eine vorhandene CI/CD Pipeline geändert werden. Diese Pipeline muss den Sicherheitsanforderungen für CI/CD folgen

Motivation: Container sollen unveränderlich sein, daher dürfen sie zur Laufzeit nicht verändert werden. Anstatt Container im Livebetrieb zu patchen, muss das Image gepatched und redeployed werden. Container Images müssen über eine vorhandene CI/CD Pipeline geändert werden. Diese Pipeline muss den Sicherheitsanforderungen für CI/CD folgen

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Modifikation von Daten

ID: 3.64-1/5.0

Req 2 Es müssen Fixe Tags verwendet werden um die Unveränderlichkeit zu gewährleisten.

Verwenden Sie das spezifischste verfügbare Tag. Wenn ein Image mehrere Tags hat, z.B. :8 und :8.0.1 oder: 8.0.1-alpine, müssen sie das letzte verwenden, da es die spezifischste Referenz ist.

Vermeiden Sie generische Tags wie "latest". [Bei Verwendung von unspezifischen Tags kann sich der Container bei einem Redeployment verändern und die Funktionsfähigkeit der Software einschränken.]

Da Tags im Container-Ökosystem grundsätzlich nicht unveränderlich (immutable) sind, kann ein Tag wie :8.0.1-alpine – so präzise er auch erscheinen mag – zukünftig durch ein anderes Image überschrieben werden. Eine tatsächliche Unveränderlichkeit und Nachvollziehbarkeit lässt sich daher nur durch die Angabe eines Digests erreichen. Sie können die Image-Version auf einen spezifischen Digest festlegen.

Motivation: Es kann von Vorteil sein, Images selbst zu erstellen, da Sie die Kontrolle über alle Komponenten behalten die mitgeliefert werden.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Modifikation von Daten
- Störung der Verfügbarkeit

ID: 3.64-9/5.0

Req 3 Container müssen auf Schwachstellen untersucht werden.

Images müssen in der Registry auf bekannte Sicherheitslücken gescannt werden.
Container müssen auch zur Laufzeit gescannt werden.

Motivation: Finden Sie bekannte Sicherheitslücken in Containern.

Umsetzungsbeispiel: Kann Beispielsweise mit DTSP erfüllt werden :<https://dtsp.telekom-dienste.de/login?redirect=%2F>

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: 3.64-13/5.0

Req 4 Ein Containerspezifisches Host-Betriebssystem muss verwendet werden.

Containerspezifische Host-Betriebssysteme müssen anstelle von Allzweck-Betriebssystemen verwendet werden.

Containerspezifische Host-Betriebssysteme verringern die Angriffsfläche, die ein System bietet.

Es gibt weniger Möglichkeiten, ein Container spezifisches Host-Betriebssystem wie z. B. : RedHat Atomic oder Core anzugreifen

Falls es nicht möglich ist ein Container spezifisches OS zu benutzen muss das Betriebssystem mindestens nach Linux/Windows SoCs gehärtet werden.

Motivation: Containerspezifische Betriebssysteme reduzieren die Angriffsfläche aufgrund ihres minimalistischen Ansatzes.

Umsetzungsbeispiel: Verwenden Sie minimale Container-Images.

Es gibt auch minimale Images innerhalb der DT <https://devsecops-tools.pages.devops.telekom.de/user-documentation/docs/mcc/baseimages/>, deren Verwendung bereits die Härtung gemäß SoCs zum Teil übernimmt und dazu beiträgt, mit einer soliden Grundlage zu beginnen.

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: 3.64-11/5.0

3. Application Server (Code Laufzeitumgebung)

Req 5 Auf dem System darf ausschließlich nur benötigte Software eingesetzt werden.

In den Hersteller-seitig bereitgestellten Installationsroutinen für Software sind oftmals Komponenten oder einzelne Bestandteile der Software als Standard-Installation vorausgewählt, die für den Betrieb und die Funktion eines spezifischen Systems nicht notwendig sind. Hierzu zählen auch Teile einer Software, die als Anwendungsbeispiele (z. B. Default-Web-Seiten, Beispieldatenbanken, Testdaten) mit installiert, aber typischerweise im Anschluss nicht verwendet werden.

Solche Komponenten müssen bei der Installation des Systems gezielt abgewählt (nicht mit installiert) werden oder - falls eine Abwahl während der Installation nicht möglich ist - unmittelbar im Anschluss entfernt werden.

Des Weiteren darf grundsätzlich keine Software auf einem System eingesetzt sein, die für den Betrieb, die Wartung oder die Funktion des Systems nicht erforderlich ist.

Motivation: Schwachstellen in der Software eines Systems sind Einfallstore für Angreifer. Durch die Deinstallation nicht benötigter Komponenten können die potentiellen Angriffsflächen deutlich reduziert werden.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Nutzung von Diensten oder Ressourcen
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.01-3/9.0

Req 6 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 Bereitstellungs Kanäle 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 7	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 verhindern) 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 8 Beispiel-Anwendungen müssen gelöscht werden.

Motivation: Beispiel-Anwendungen enthalten möglicherweise Schwachstellen und bieten dadurch Angriffspunkte.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.84-8/7.1

Req 9 Das System muss vor Überlastsituationen geschützt werden.

Ein System muss über Schutzmechanismen verfügen, die Überlastsituationen so weit wie möglich verhindern. Insbesondere ist eine partielle oder komplette Beeinträchtigung der Verfügbarkeit des Systems zu vermeiden.

Beispiele für mögliche Schutzmaßnahmen sind:

- Begrenzung des pro Anwendung verfügbaren Arbeitsspeichers (RAM)
- Begrenzung der maximalen Sessions einer Web-Anwendung
- Begrenzung der maximalen Größe eines Datensatzes
- Begrenzung von CPU-Ressourcen pro Prozess
- Priorisieren von Prozessen
- Begrenzung der Anzahl oder der Größe von Transaktionen eines Benutzers oder von einer IP-Adresse in einem bestimmten Zeitraum

Anmerkung:

vor Netzwerk-basierenden Angriffen mit extrem hohen Daten- oder Paketraten, den sogenannten "Distributed Denial of Service" (DDoS) Angriffen, kann ein System sich i. d. R. selbst nicht mehr schützen. Zur Abwehr von DDoS Angriffen wird eine vorgelagerte Lösung in der Netzwerk-Schicht benötigt.

Motivation: Angreifer können mit gezielten ressourcenlastigen oder großvolumigen Anfragen versuchen, die Ressourcen eines Systems aufzubrechen, so dass das System seine regulären Aufgaben bzw. vorgesehenen Aufgabenvolumen nicht mehr erfüllen kann und die Verfügbarkeit der angebotenen Dienste effektiv gestört ist. Die Begrenzung der Ressourcen, die je gestellter Anfrage an das System maximal in Anspruch genommen werden können, ist eine grundlegende Maßnahme, um die Auswirkungen solcher Denial-of-Service (DoS) Angriffe zu reduzieren.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Nutzung von Diensten oder Ressourcen
- Störung der Verfügbarkeit

ID: 3.01-12/9.0

Req 10 In Überlastsituationen muss sich das System berechenbar verhalten.

Selbst umfassende systemeigene Schutzmaßnahmen können in extremen Situationen möglicherweise nicht mehr verhindern, dass ein System in eine Überlastsituation gerät.

Es muss daher sichergestellt sein, dass das System in Überlastsituationen nicht in einen Zustand überwechselt, der sicherheitsrelevante Funktionen oder Eigenschaften des Systems außer Kraft setzt. Leistungseinbußen (bspw. die Verminderung des Durchsatzes von legitimen Netzwerk-Paketen oder der Anzahl beantworteter Server-Anfragen pro Zeitraum) sind in Überlastsituationen i. d. R. nicht vermeidbar, das reguläre funktionale Verhalten des Systems muss jedoch grundlegend erhalten bleiben.

Dies kann im Extremfall bedeuten, dass ein kontrolliertes Herunterfahren des Systems eher hinnehmbar ist, als ein Weiterbetrieb bei unkontrolliertem Versagen der Sicherheitsfunktionen und somit dem Verlust des Systemschutzes.

Motivation: Mittels eines Denial-of-Service-Angriffs kann ein Angreifer versuchen, ein System gezielt zu überlasten. Reagiert ein solches System dann unberechenbar oder versagt sein reguläres Verhalten, insbesondere hinsichtlich dessen Sicherheitsfunktionen, kann dies dem Angreifer eine erweiterte Angriffsfläche auf Funktionen und Daten des Systems eröffnen sowie potentiell weitere verknüpfte Systeme gefährden.

Umsetzungsbeispiel: Eine Firewall, die in Überlastsituationen ihr Filter-Regelwerk verwirft und sämtliche Pakete ohne Prüfung weiterleitet, würde die Anforderung nicht erfüllen. In diesem Fall wäre das Blockieren sämtlicher Pakete durch die Abschaltung der Firewall eher hinnehmbar als das Versagen ihrer regulär vorgesehenen Schutzaufgabe für nachgelagerte Systeme.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Nutzung von Diensten oder Ressourcen
- Störung der Verfügbarkeit
- Unbemerkt durchführbare Angriffe

ID: 3.01-13/9.0

Req 11 Die Nutzung und der Zugriff auf schutzbedürftige Funktionen und Informationen, dürfen nicht ohne erfolgreiche Authentifizierung und Autorisierung möglich sein.

Die Nutzung der Funktionen des Systems oder der Zugriff auf schutzbedürftige Daten die als intern oder vertraulich eingestuft sind, darf erst nach einer eindeutigen Identifizierung des Benutzers und dessen erfolgreichen Authentifizierung durch den Benutzernamen und mindestens eines Authentisierungsmerkmals möglich sein. Eine Ausnahme hiervon sind Funktionen, die öffentlich nutzbar sind, wie beispielsweise der Bereich eines Web-Servers im Internet, über den öffentliche Informationen zur Verfügung gestellt werden. Beispiele für Funktionen, die eine vorhergehende Authentifizierung erfordern, sind Netzdienste (wie SSH, SFTP, Web-Dienste), lokaler Zugriff auf die Management-Konsole, die lokale Nutzung des Betriebssystems und von Anwendungen. Die folgenden Beispiele sind Möglichkeiten, die für die Authentifizierung genutzt werden können:

- Abfrage von Benutzername und Passwort
- Benutzung von kryptographischen Schlüsseln und Zertifikaten (z. B. in Form von Smartcards)

Diese Anforderung ist auch für Konten anzuwenden, die nur für die Kommunikation zwischen Systemen (M2M) genutzt werden.

Motivation: Die Authentifizierung ist notwendig, um einen Benutzer zweifelsfrei identifizieren zu können, da hiervon die ihm zugeteilten Berechtigungen und somit der Zugriff auf Daten und Dienste eines Systems abhängig sind.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen

ID: 3.84-11/7.1

Req 12 Der Zugriff auf den Application Server muss protokolliert werden.

Das Protokoll muss die folgenden Informationen enthalten:

- Zugriffszeitpunkt
- Quelle (IP-Adresse)
- Konto (falls bekannt)
- URL
- HTTP-Status-Code der Antwort des Application Servers

Bei der Protokollierung müssen die jeweils geltenden gesetzlichen, tariflichen und betrieblichen Bestimmungen beachtet werden. Diese besagen unter anderem, dass die Protokollierung von Ereignissen nur zweckgebunden erfolgen darf. Die Protokollierung von Ereignissen, zum Zwecke der Arbeitskontrolle von Mitarbeitern, ist nicht zulässig.

sig.

Motivation: Für die Analyse von Sicherheitsvorfällen ist es sehr wichtig grundlegende Informationen darüber zu besitzen, wie ein Angriff erfolgt ist.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unbemerkt durchführbare Angriffe

ID: 3.84-12/7.1

4. Web-Service

4.1. Web-Service: allgemeine Anforderungen

Req 13	Grundlegende Sicherheitsfunktionen für extern zugängliche Web Services müssen in der DMZ implementiert werden und je nach Kritikalität mit einem Web Service Gateway geschützt werden.
--------	--

"Extern" in diesem Kontext meint hier außerhalb der eigenen Vertrauenszone (Trust Boundary). Wird der Service im Internet angeboten oder genutzt ist die DMZ notwendig (verpflichtend). Bei internen Lösungen besteht der Freiheitsgrad über den Schnitt der Vertrauenszonen abzuwägen. Aus der Zero-Trust-Perspektive ist die Trust Boundary die Systemgrenze. Je nach Situation ist es möglich diese zum Beispiel auf Kubernetes-Cluster-Ebene auszudehnen.

Eine DMZ ist eine Zone, in der sämtlicher externer Verkehr terminiert wird.

Datenbanken und Applikationen werden in der MZ aufgebaut, das heißt in einer Zone in der kein direkter externer Zugriff möglich ist.

Folgende Funktionen müssen auf der Zugangs- und Validierungsebene (DMZ) realisiert werden:

- Authentisierung und Autorisierung für den Zugriff (d.h., ist der Konsument des Web Services zum Zugriff berechtigt?)
- grundsätzliche Validierung auf Basis der vorliegenden Definition wie z. B. WSDL, JSON Schemata oder XSD
- Vor- und Nachbearbeitungsaufgaben (z. B. Terminierung einer TLS Verschlüsselung oder sicherheitsrelevante Protokollierung)
- Terminierung von TCP-Verbindungen und Blockieren aller direkten Zugriffe auf Web Service-Endpunkte in der MZ
- Limitieren der maximalen Verbindungen

Um diese Funktionalitäten zu gewährleisten sollte immer ein Web Service Gateway eingesetzt werden.

Motivation: In der DMZ müssen alle Maßnahmen ergriffen werden, um die Services in der MZ gegen Angriffe abzusichern.

Umsetzungsbeispiel: Die Nutzung von TARDIS mit Stargate für die DTAG internen Zonen und Spacegate für das Internet ist eine empfohlene Lösung. Historisch bildet die SOA-BP mit dem B2B-Gateway eine gleichwertige Lösung.

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

ID: 3.02-3/9.1

Req 14	Werden bei einem Web Service besonders schützenswerte Daten verarbeitet, so müssen diese Daten durch Einsatz von durchgängigen Mechanismen auf der Anwendungsebene (End-to-End), wie z. B. Verschlüsselung, individuell geschützt werden.
--------	---

Da Transportverschlüsselungen bereits auf äußeren Schichten terminiert werden, sowie möglicher Payloadprotokollierung und "Hop by Hop" Kommunikation bietet eine reine Transportverschlüsselung in den meisten Fällen keinen ausreichenden Schutz für besonders vertrauliche Daten.

Beispiele für besonders schützenswerte Daten sind Gesundheitsdaten, Führungszeugnis, Kontodaten von Personen,

Quartalszahlen vor Veröffentlichung und Vertragsentwürfe mit sehr hohem Volumen.

Motivation: Aufgrund des besonderen Schutzbedarfes ist für bestimmte Daten selbst dann Vertraulichkeitsschutz erforderlich, wenn sie transportverschlüsselt oder über sichere Netze übertragen werden.

Umsetzungsbeispiel: Es wird empfohlen End-to-End Verschlüsselungslösungen wie beispielsweise JSON Web Encryption (JWE, defined by RFC 7516) zu verwenden. Für das Schlüsselmanagement stehen Systeme wie T-PAM Conjur, Thales Key Manager oder Vault-Lösungen der jeweiligen Geschäftseinheit bereit.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten

ID: 3.02-4/9.1

Req 15	Ein Web Service muss vor Manipulation / Replay Attacken geschützt werden, wenn dieser über ungesicherte Transportwege übertragen wird.
--------	--

Das kann durch Einsatz von Payload-Signaturen auf der Anwendungsebene, Timestamps / Session IDs / Tokens bzw. TLS auf der Transportebene oder durch andere geeignete Mechanismen (z. B. VPN) erreicht werden.

Motivation: Es wird davon ausgegangen, dass die sog. „Non-Repudiation“-Mechanismen, d. h. Mechanismen, die sicherstellen, dass eine Transaktion nicht abgestritten werden kann in juristischen Verfahren, sowie auch für Rechnungslegungs- und Revisionszwecke zunehmend Akzeptanz finden.

Daher ist es äußerst wünschenswert, dass man sich nicht nur auf TLS als kurzlebigen Mechanismus für den Integritätsschutz auf der Transportschicht verlässt, sondern das Problem mit dauerhaften Payload-Signaturen auf der Anwendungsebene löst.

Im Vergleich zum TLS-Integritätsschutz decken Payload-Signaturen andere Zeitskalen und Ziele ab. Deshalb wird der Einsatz eines speziellen öffentlichen / privaten Schlüsselpaares für Signaturen unvermeidlich.

Payload-Signaturen sind der einzige Mechanismus, mit dessen Hilfe man einem externen Dritten (etwa einer Rechtsinstanz wie z. B. einem Gericht) beweisen kann, dass eine Nachricht von einer bestimmten Einheit empfangen wurde.

Mit dem Zeitstempel und / oder der Nachrichten-ID können Replay-Attacken verhindert werden.

Um vor ausgeklügelten Angriffen auf die Signaturen zu schützen ist es wichtig, die gesamte Nachricht bzw. den gesamten Body zu signieren.

Umsetzungsbeispiel: **XML-basierte Nachrichten:**

XML-Signature (XML-Sig, auch XML-DSig, XMLDsig) ist eine Spezifikation, um bestehende digitale Signaturen in XML-Schreibweise nutzen zu können. Sie findet Einsatz in vielen weiterführenden Web-Standards wie etwa SOAP, SAML oder dem deutschen Online-Banking-System FinTS.

JSON-basierte Dienste:

JSON Web Signature (JWS, RFC 7515).

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
- Unbemerkt durchführbare Angriffe

ID: 3.02-7/9.1

Req 16	Der Öffentlichkeit darf nur die kleinstmögliche Anzahl an Web Services zugänglich gemacht werden.
--------	---

Insbesondere bei der Weitergabe eines internen Web Services an einen externen Partner oder bei der Erweiterung einer Legacy Anwendung um ein Web Service Frontend muss darauf geachtet werden, dass nur erforderliche Web Services weiter gegeben werden.

Motivation: Indem nur die minimale erforderliche Anzahl an Web Services angeboten wird, wird die Angriffsfläche reduziert.

Umsetzungsbeispiel: Auf Internet Web Service Gateways dürfen nur die Web Services verfügbar gemacht werden, die aus dem Internet heraus erreichbar sein müssen.

Gleiches gilt z. B. für die ausgehenden Verbindungen eines Container Namespaces.

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: 3.02-8/9.1

Req 17 Der Web-Service muss sich gegen Überlastsituationen schützen.

Ein Web-Service muss über Schutzmechanismen verfügen, die Überlastsituationen soweit wie möglich verhindern. Insbesondere ist eine partielle oder komplette Beeinträchtigung der Verfügbarkeit des HTTP-Servers zu vermeiden. Beispiele für mögliche Schutzmaßnahmen sind:

- Begrenzung der maximalen Anzahl von HTTP Sessions pro IP-Adresse
- Festlegen der maximalen Größe eines HTTP Requests
- Festlegen eines Timeouts für HTTP Requests

Begrenzungen müssen unter Berücksichtigung der zu schützenden Anwendung und ihrer Besonderheiten festgelegt werden. Als Richtlinie können hierbei die folgenden Werte dienen.

Falls über den HTTP-Server keine Uploads erfolgen sollen:

- Maximale Anzahl von HTTP Sessions pro IP Adresse: 50
- Maximale Größe eines HTTP Requests: 20000 Bytes
- Timeout für HTTP Requests: 30 Sekunden

Falls über den HTTP-Server auch Uploads erfolgen sollen:

- Maximale Anzahl von HTTP Sessions pro IP Adresse: 50
- Maximale Größe eines HTTP Requests: 10000000 Bytes oder, falls bekannt, maximale Größe des zu erwartenden Uploads
- Timeout für HTTP Requests: 60 Sekunden oder, falls bekannt, für maximalen Upload benötigte Zeit

Motivation: Angreifer versuchen oftmals mittels Denial-of-Service-Angriffen einen Webserver in eine Überlastsituation zu bringen. Ist ein solcher Angriff erfolgreich kann dies zur Beeinträchtigung der Verfügbarkeit oder Integrität des HTTP-Servers führen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Störung der Verfügbarkeit

Für diese Anforderung sind folgende Gewährleistungsziele relevant:

- Verfügbarkeit
- Integrität

ID: 3.84-17/7.1

4.2. Web-Service: Authentifizierung und Autorisierung

Req 18 Anbieter und Konsumenten eines Web Service müssen sich gegenseitig authentifizieren und autorisieren, wenn schutzbedürftige Daten übertragen werden.

Der Anbieter eines Web Services muss sicherstellen, dass der Konsument berechtigt ist auf diesen Web Service zuzugreifen und diese Daten erhalten bzw. diese Funktionen verwenden darf.

Der Konsument muss sicherstellen, dass er mit dem korrekten Anbieter kommuniziert.

Berechtigungsprüfungen müssen in jeder Funktion berücksichtigt werden. Es darf nicht auf einen besonderen Vertrauensstatus aufgrund der Authentisierung eines Gerätes auf Ebene des Netzwerkes oder von Verbindungen aus z. B. dem gleichen Netzwerk oder Namespace vertraut werden.

Die Berechtigungsprüfung muss sicherstellen, dass der Konsument nur die Daten erhält, die er erhalten darf.

Erläuterung "schutzbedürftige Daten", meint hier Daten die nicht "öffentlich" sind, also als "intern" oder "vertraulich" bewertet sind. Innerhalb der DTAG ist intern/vertraulich die Regel.

Motivation: Um eine unerlaubte Nutzung von Ressourcen oder Ausgabe von Daten zu verhindern, ist eine angemessene Authentisierung und Autorisierung erforderlich.

Ein Zugriff auf einen nicht schutzbedürftigen Web Service muss nicht authentifiziert oder autorisiert werden.

Umsetzungsbeispiel: Um generelle Berechtigungen für den Zugriff auf einen Web Service zu erteilen, eignet sich z. B. eine zertifikatsbasierte Authentifizierung / Autorisierung.

Um eine Berechtigung auf Datenfeldebene zu erteilen, eignen sich z. B. JW-Tokens mit individuellen "Claims", in Verbindung mit Filtern, um nur die für den Konsumenten berechtigte Datenfelder herauszugeben.

Der JW-Token oder das Zertifikat muss durch jede Funktion / Container / Microservice überprüft 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: 3.02-9/9.1

Req 19 Der Mechanismus zur Authentifizierung und Autorisierung muss auf starken kryptografischen Algorithmen / Frameworks basieren.

Starke kryptografische Frameworks/Algorithmen in diesem Sinne sind zum Beispiel XML-Signaturen oder TLS-Client-Zertifikate mit geeigneten Algorithmen wie SHA3- / SHA2 Hashes und ECDSA/RSA-Signaturen.

Weitere Beispiele sind JW-, OAuth- und STS-Tokens, sofern über TLS übertragen.

Motivation: Schwache Algorithmen können von Angreifern kompromittiert und Identitäten vorgetäuscht werden.

Umsetzungsbeispiel: Zertifikate gemäß der Zertifikatsanforderungen für HTTPS

JW Tokens:

- Hash Verfahren SHA 256 oder höher
- keine "none" Algorithmen ({"alg":"none"})
- ES, RS Algorithmen gegenüber HS bevorzugen
- möglichst kurze Lebenszeiten

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: 3.02-10/9.1

Req 20	Wenn die Authentisierung auf Token basiert, müssen diese durch das Web Service-Gateway bzw. einem zugelassen M2M IDPs generiert werden.
--------	---

Die Verwendung eines (förderierten) zentralisierten IDP (Identity Providers) als einziger vertrauenswürdiger Identitätsanbieter ist ein wesentlicher Teil des Konzepts eines API-Security.

Motivation: API-Security@DTAG - Prinzip 1: Identifizierung, Authentifizierung, Autorisierung (Identitätsmanagement)
 Die Implementierung robuster und anpassungsfähiger Authentifizierungsmechanismen ist von entscheidender Bedeutung. Genutzt werden Industriestandard-Protokolle wie OAuth 2.0 zur Validierung der Identität von M2M- und menschlichen Akteuren, die versuchen, auf APIs zuzugreifen. Dabei muss der gewählte Mechanismus auf das erforderliche Schutzniveau der Daten und die Schutzwürdigkeit der API innerhalb unserer Systemarchitektur abgestimmt sein. Statische API-Tokens sollten niemals eine Option für interne Dienste sein.
 In Verbindung mit der Authentifizierung sind zuverlässige Autorisierungskontrollen unverzichtbar. Es muss die Möglichkeit bestehen, fein abgestufte Zugriffskontrollen zu definieren und durchzusetzen, um sicherzustellen, dass authentifizierte Entitäten nur auf die Ressourcen und Funktionalitäten Zugriff haben, die für ihre Rollen und Verantwortlichkeiten relevant sind. Dies bedingt eine präzise Zuweisung von Benutzerrollen zu bestimmten API-Berechtigungen sowie eine präzise abgestimmte Kontrollebene.
 Ein effizientes Management von kurzlebigen Tokens spielt für die sichere (konzerninterne) API-Interaktion eine entscheidende Rolle. Die Lebensdauer der Tokens ist auf der Grundlage der Schutzwürdigkeit der Daten und der Geschäftsanforderungen zu definieren. Es müssen Mechanismen für den Widerruf und die Erneuerung von Tokens eingerichtet werden, um die Kontrolle über die Zugriffsberechtigungen zu wahren.

Umsetzungsbeispiel: TARDIS mit seiner Kombination aus Gateway und IRIS.

ID: 3.02-11/9.1

4.3. Web-Service: Validierung

Req 21	Alle Web Service Anfragen und Antworten müssen durch den Web Service Anbieter und Konsumenten gegen eine detaillierte Spezifikation validiert werden.
--------	---

Jeder durch einen Web Service übermittelte Datensatz muss mit den erwarteten Datenfeldern, der erwarteten Länge und / oder dem erwarteten Bereich übereinstimmen, sowie ggf. mit einer formellen Spezifikation zur Beschreibung der akzeptablen Daten.

Sollten die erwarteten Datenfelder nicht immer fest definierbar sein, dürfen nur erwartete Datenfelder verarbeitet werden.

Motivation: Die detaillierte Spezifikation (einschließlich formelle Spezifikation) ermöglicht eine weitaus bessere Beschreibung der Daten, die in Datenfeldern eines Web Services erwartet werden.

Umsetzungsbeispiel: Eine komplette Web Service Beschreibung kann durch z. B. WSDL 2.0 oder Swagger erreicht werden.

Beispiele für Datenformatbeschreibungen sind JSON Schemata oder XML Schemata Definitions.

Ein Beispiel für eine formelle Spezifikation sind reguläre Angaben wie z. B. "[A-Za-z0-9]+[._-#%/=~])*[A-Za-z0-9]+@[A-Za-z0-9]+[.]+[A-Za-z]{2,6}" zur Spezifizierung einer gültigen E-Mail-Adresse.

Der Basistyp „int“ sollte durch einen anwendungsspezifischen Subtyp ersetzt werden, der den benötigten Ganzzahlbereich genau definiert.

Entsprechend muss der Basistyp „string“ durch einen anwendungsspezifischen Subtyp ersetzt werden, der die Länge der Zeichenfolge mittels „maxLength“ und „minLength“ begrenzt.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Störung der Verfügbarkeit

ID: 3.02-12/9.1

Req 22	Enthält der Web Service keine formelle Spezifikation der Eingabedaten muss mittels Black- / Whitelists verhindert werden, dass illegalen Zeichen akzeptiert werden.
--------	---

Daten, die von den nachfolgenden Teilen der Anwendung nicht erwartet werden, können unerwartete und unerwünschte Auswirkungen haben.

Wenn die Eingabedaten in der DMZ ordnungsgemäß gefiltert werden, werden solche Angriffe erheblich erschwert.

Motivation: Durch Black- / Whitelisting kann effektiv vor gewissen Angriffsarten, wie z. B. SQL-, LDAP-, XML-, XPath-, XQuery-, Code-, Kommandoinjektion, geschützt werden.

Der Einsatz von Whitelists ist Blacklists vorzuziehen, da Blacklists im Laufe der Zeit tendenziell obsolet werden.

Umsetzungsbeispiel: Für Web Services, die über einen Webserver geführt werden, stellt das Open-Source-Tool mod_security eine gute Lösung dar.

Die OWASP bietet verschiedene Cheatsheets zur Lösung an:

https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Störung der Verfügbarkeit

ID: 3.02-13/9.1

4.4. Web-Service: Payload Signaturen

In speziellen kann es notwendig sein den Inhalt (Payload) mit einer kryptographischen Signatur des Requests oder der Response gegen Veränderung und Abstreitbarkeit zu schützen.

Req 23	Wird ein Web Service-Request mit Signaturen geschützt, dürfen die Signaturdaten vor der Weitergabe nicht aus dem Request entfernt werden.
--------	---

Die Signatur muss über den ganzen Übertragungsweg hinweg validiert werden, um die Integrität und Authentizität der Web Service Anfragen / Antworten zu garantieren.

Motivation: Wird diese Signatur entfernt, ist die Integrität und Authentizität des Web Services nicht mehr gewährleistet.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen
- Abstreiten von durchgeführten Aktionen

ID: 3.02-14/9.1

4.5. Web-Service: TLS (Transport Layer Security)

4.6. Web-Service: REST

Das HTTP Protokoll liefert für jeden Status einer Verbindung standardisierte Codes.

Weitere zusätzliche Informationen wie z. B. die empfangenen Daten (Payload) oder technische Details zu dem Fehler dürfen nicht ausgegeben werden, sondern sind mit einer eindeutigen Fehlerreferenz im Protokoll abzuspeichern. Die Fehlerreferenz darf ausgegeben werden.

Kategorie	Beschreibung
1xx: Informational	Transfer Protokoll Informationen
2xx: Success	Zeigt an, dass die Anfrage erfolgreich angenommen wurde
3xx: Redirection	Zeigt an, dass der Anfragende zusätzliche Maßnahmen ergreifen muss, um seine Anfrage zu vervollständigen
4xx: Client Error	Der Client scheint einen Fehler zu haben
5xx: Server Error	Der Server scheint einen Fehler zu haben

Motivation: Das HTTP Protokoll liefert bereits für jeden Status einer Verbindung standardisierte Codes. Durch das Verwenden dieser Status Codes wird verhindert, dass schutzbedürftige Informationen, z. B. Fehler- und Systemmeldungen, in Ausgaben enthalten sind. Ebenso kann der Konsument des Web Services anhand dieser standardisierten Codes generisch feststellen, woran der Fehler liegt. Durch das Widerspiegeln der empfangen Daten (Payload) können Sicherheitslücken entstehen (besonders bei Webanwendungen).

Umsetzungsbeispiel: Ein Beispiel einiger HTTP Status codes und deren Bedeutung:

HTTP Status Code	Nachricht	Bedeutung
200	OK	Die Anfrage wurde erfolgreich bearbeitet
400	Bad Request	Die Anfrage-Nachricht war fehlerhaft aufgebaut
406	Not Acceptable	Der vom Client angefragte Inhaltstyp im Accept-Header wird vom Web Service nicht angeboten
413	Payload to large	Die gestellte Anfrage war zu groß

Alle HTTP Status Codes und deren verwendung sind im RFC7231 spezifiziert.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.02-23/9.1

Das HTTP Protokoll liefert standardisiert alle Methoden, die für sämtliche Funktionalitäten benötigt werden. Durch das Verwenden dieser Methoden wird eine einheitliche und transparente Berechtigungssteuerung ermöglicht,

die durch Web Server, Reverseproxys und Web Service Gateways unterstützt wird.

Motivation: Durch das Verwenden standardisierter Methoden wird vorgelagerten Schichten wie z. B. Web Servern, Reverseproxys und Web Service Gateways ermöglicht grundsätzliche Validierungen der Anfragen durchzuführen. Ebenso sorgt dies für ein einheitliches Verständnis für Konsumenten, wie ein Web Service zu verwenden ist.

Umsetzungsbeispiel: Beispiele für die gängigsten HTTP Methoden:

HTTP Methode	Beschreibung
GET	Die GET-Methode fordert die Übertragung der aktuell ausgewählten Ressource an
HEAD	Die HEAD-Methode ist identisch mit GET, außer dass der Server nicht den HTTP-BODY der Antwort sendet
POST	Die POST-Methode fordert die Verarbeitung der übermittelten Daten an
PUT	Die PUT-Methode fordert die Erstellung oder den Ersatz vom Zustand der Zielressource
DELETE	Die DELETE-Methode fordert das Entfernen der Zielressource an
PATCH	Die PATCH-Methode fordert eine Reihe von Änderungen an, die in der Anfrage beschrieben sind

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Unberechtigte Nutzung von Diensten oder Ressourcen

ID: 3.02-24/9.1

Req 26 Nicht benötigte HTTP-Methoden müssen deaktiviert werden.

Die TRACE/TRACK-Methode darf bei einem produktiven Webserver nicht verwendet werden. Standard-Requests an Webserver verwenden nur GET und POST. Werden andere Methoden benötigt, müssen diese sicher verarbeitet werden.

Motivation: HTTP TRACE könnte von einem Angreifer missbraucht werden. Diese Methode ermöglicht das Debugging und die Trace-Analyse für Verbindungen vom Client zum Webserver. Andere HTTP-Methoden könnten auch verwendet werden, um Informationen über den Server zu erhalten oder könnten direkt von einem Angreifer missbraucht werden.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unberechtigte Modifikation von Daten
- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.84-26/7.1

Req 27 Informationen über den Server in HTTP-Headern müssen auf ein Mindestmaß beschränkt werden.

Der HTTP-Header darf keine Informationen zur Version des Webserver und verwendeter Module/Erweiterungen enthalten.

Motivation: Alle Informationen über den HTTP-Server könnten über Sicherheitsschwachstellen Aufschluss geben.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.84-27/7.1

Req 28	Vertrauliche Daten dürfen nicht in der URL übertragen werden.
--------	---

Vertrauliche Daten dürfen ausschließlich außerhalb der URL übertragen werden (z. B. HTTP-BODY, HTTP-HEADER). Bei der Verwendung von HTTP-Headern dürfen nur Felder verwendet werden, die nicht in Protokolldateien erfasst werden.

Beispiele für vertrauliche Daten sind:

- API Keys
- Security Tokens (z. B. OAuth-, JW-Token)
- Passwörter

Neben den vertraulichen Daten aus der Perspektive Informationsschutz, ergibt sich auch eine Vertraulichkeit aus dem Bezug der Daten zu einer Person. Auch diese Daten dürfen nicht in der URL übertragen werden.

Im Kontext von Restful-APIs führt das zu Konflikten, da ein Ressourcenname (ID, Identität) einen Personenbezug haben kann.

Datenschutzrechtliche Bewertung:

Wird ein URL-Parameter bei der Kommunikation zwischen zwei IT-Systemen **innerhalb** der DTAG verwendet, dann ist ein URL-Parameter mit Personenbezug (PII-Parameter) akzeptabel, wenn:

1. API wird nur über TARDIS exponiert und verwaltet.
2. API ist nur intern exponiert (Stargate).
3. Es werden nur Daten bis max. Datenschutzklasse 2 übertragen.
4. In der URL wird max. ein, heißt genau ein personenbezogenes Attribut übertragen.

Positivbeispiel (ein Parameter):

- Im Suchrequest wird in einer Nachricht nur entweder die MSISDN oder die Kundennummer übertragen.

Negativbeispiele (zwei Parameter):

- Im Such-Request wird eine Kombination aus MSISDN + Kundennummer genutzt.
- Kundennummer + BKTO (sowieso eigentlich DSK3 bei TDG, weil Credential im telefonischen Service)
- MSISDN + Kundennachname

Motivation: Sämtliche in der URL enthaltenen Daten erscheinen in Protokolldateien von z. B. Web Server, Reverse Proxy und Web Service Gateway.

Protokolldateien werden oft an zentraler Stelle gesammelt oder auch an SIEM Systeme zur Überwachung übertragen. Um dies zu ermöglichen und gleichzeitig vertrauliche Daten zu schützen, dürfen diese nicht in der URL übertragen 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

ID: 3.02-25/9.1

Req 29 Inhaltstypen (Content-Type) müssen validiert werden.

Eine REST Abfrage und Antwort muss mit dem beabsichtigten Inhaltstyp in der Kopfzeile übereinstimmen. Anderenfalls könnte dies zu Fehlinterpretationen auf der Konsumenten- / Anbieterseite führen und eine Code-Injektion / Code-Ausführung zur Folge haben.

Motivation: REST Web Services erlauben oft mehrere Datenformate zur Anfrage und Antwort.

Der Consumer muss das Datenformat seiner Anfrage, sowie das gewünschte Datenformat für die Antwort des Web Services in den "Content-Type" und "Accept" Headern bestimmen.

Der Provider darf die Anfrage nur verarbeiten, sofern er das Datenformat für die Anfrage und für die Antwort unterstützt.

Umsetzungsbeispiel: Anfragen mit unerwarteten oder fehlenden Inhaltstyp-Headern mit HTTP Code 415 "Nicht unterstützter Medientyp" ablehnen.

ID: 3.02-26/9.1

Req 30 Informationen über den HTTP-Server in Fehlerseiten, die durch den HTTP-Server ausgeliefert werden, müssen entfernt werden.

Die Standardfehlerseiten müssen durch benutzerspezifisch definierte Fehlerseiten ersetzt werden. Benutzerspezifisch definierte Fehlerseiten dürfen keine Versionsinformationen des Webservers und verwendeter Module/Erweiterungen enthalten. Die Fehlermeldungen dürfen keine internen Informationen, wie z. B. interne Servernamen, Fehlercodes usw. enthalten.

Motivation: Alle Informationen über den HTTP-Server könnten Aufschluss über Sicherheitsschwachstellen geben.

Umsetzungsbeispiel: Eigene Fehlerseiten ohne Informationen über HTTP-Server-Produkt und -Version erstellen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Angriffe motiviert und erleichtert durch Informationspreisgabe oder erkennbare Sicherheitsschwachstellen

ID: 3.84-30/7.1

4.6.1. API-Provider spezifische Anforderungen

Diese Anforderungen sind nur relevant, wenn eine API als Provider angeboten wird. Systeme oder Services die API nur nutzen (Consumer) stellen den gesamten Abschnitt auf "Nicht Zutreffend" (=NA).

Req 31 Der API-Provider muss eine technische Dokumentation der API veröffentlichen/anbieten.

Die technische Dokumentation sollte auf allgemeine Standards wie beispielsweise [Open API Specification](https://developer.telekom.de/docs/src/api_standards_conventions/2_basics/) (OAS) basieren. Nähere Erläuterungen und weitere Informationen zur allgemeinen API-Standardisierung, Spezifikation und -Dokumentation finden sich beispielsweise hier: https://developer.telekom.de/docs/src/api_standards_conventions/2_basics/ oder im TM-Forum: <https://www.tmforum.org/oda/open-apis/>

Die technische Dokumentation muss die folgenden Informationen, Felder und Methoden bezüglich der verfügbaren API-Endpunkte enthalten:

- Spezifikationsstandard
- API-Name
- API-Beschreibung
- API-Version
- Verantwortliche Org.- Einheit
- Kontakt-E-Mail-Adresse des API-Verantwortlichen
- Sicherheits-Scopes (bezüglich Autorisierung) für die Endpunkte

- Kundensegment
- IT-Anwendung, z.B. ICTO-Nummer
- Version des API-Dienstes
- Ggf. Restlaufzeithinweise (deprecation indicator)
- Endpunkt-Spezifikationen
- Datenspezifikationen
- Genehmigte und bereitgestellte Media-Typs
- HTTP-Status-Codes
- Error Response Object
- Differenzierte Fehlerfälle
- Datenschutzklasse (DSK) oder Informationsschutzklasse (VS)

Motivation: (Technische) Dokumentation: Eine umfassende Dokumentation hilft bei Sicherheitsaudits und -überprüfungen und gewährleistet, dass während des gesamten Entwicklungs- und Integrationsprozesses bewährte Verfahren (Best Practices) eingehalten werden. Außerdem kann sie dazu dienen, versteckte APIs aufzudecken oder zu verbieten. Eine Implementierung ist beispielsweise die OpenAPI Specification (OAS).

Umsetzungsbeispiel: Umsetzungsbeispiele finden sich im API-Catalog von TARDIS: <https://developer.telekom.de/catalog> oder im TARDIS Rover Handbuch: https://developer.telekom.de/docs/src/tardis_customer_handbook/rover/

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigte Modifikation von Daten
- Unbemerkt durchführbare Angriffe

ID: 3.02-27/9.1

Req 32 APIs (M2M) müssen über einen API-Gateway exponiert sein, das als Proxy fungiert.

- Eine direkte Exposition ist nicht zulässig.
- Innerhalb desselben Sicherheitskontextes eines Anwendungsmoduls kann eine direkte Kommunikation zu strikt internen APIs innerhalb einer Vertrauenszone stattfinden, doch ist dies nicht erwünscht.

Unter gewissen Umständen kann eine Ausnahme zu dieser Anforderung gerechtfertigt sein.

Triftige Gründe für ein Ausnahme sind:

- Echtzeit- oder Niedrig-Latenz-Verbindungen sind notwendig. Wenn die Latenz entscheidend ist, kann es sinnvoll sein, Systeme direkt zu verbinden.
- Kunden-Implementierung: In einem Szenario, in dem eine API für einen externen Kunden des Konzerns betrieben wird, hängt es von dem jeweiligen Vertrag mit dem Kunden ab, ob die Nutzung eines zentralen API-Gateway erforderlich oder optional ist. Die Verwendung eines zentralen Gateway wird jedoch in jedem Fall empfohlen.
- Die Datenschutzgesetzgebung schreibt u.U. eine Ende-zu-Ende-Verschlüsselung vor. Die Software kann aber nicht vom Konzern oder dem Kunden zwecks Implementierung der Payload-Verschlüsselung geändert werden, dann ist ggf. eine direkte Kopplung zwischen den Diensten unumgänglich.
- Kleinstmögliche Trust Zone: In einem Szenario, in dem Mikroservices im selben Kubernetes-Namespace laufen und vom gleichen, kleinen Team verwaltet werden.

Motivation: API-Security@DTAG - Prinzip 2a: Isolierung nach Bereichen - Zero Trust – Trust Boundary

Durch die Definition von Vertrauenszonen auf verschiedenen Ebenen (Kubernetes, Team, Organisation, Legaleinheit, Datenschutzbestimmungen, NatCos usw.) kann die Entwicklung kleinerer Service-APIs optimiert und unternehmenskritische Umgebungen können zum Schutz gegen menschliche Fehler isoliert und abgeschirmt werden. Die Umset-

zung der Isolierung mit einem API-Gateway reduziert die Angriffsfläche auf ein Minimum. Die Auswirkungen eines Sicherheitsverstoßes wird auf isolierte Bereiche eingegrenzt und eine Seitwärtsbewegung von Angreifern verhindert. Standardmäßig muss ein neu eingeführter Dienst immer isoliert werden; Kommunikation ist ausdrücklich durch Whitelisting zu erlauben.

Zero Trust ist extrem wichtig für APIs: Es zwingt sie dazu, innerhalb segmentierter Netzwerke als ihre eigenen strikten Vertrauensgrenzen zu fungieren. Das heißt, dass APIs die Identität jedes Anfragenden streng authentifizieren müssen, und zwar über die reine Maschine-zu-Maschine-Interaktion hinaus. Dies geschieht durch Überprüfen von zusätzlichem Kontext wie beispielsweise des Gerätezustands. Auf diese Weise gewährleisten APIs ein hohes Maß an Sicherheit: Sie setzen kein inhärentes Vertrauen voraus und reduzieren auf diese Weise potenzielle Bedrohungen (von inner- und außerhalb). Der Aufwand für die Überprüfung der erforderlichen Kontextparameter sollte sich nach der Schutzwürdigkeit der verwalteten Daten richten.

Umsetzungsbeispiel: Durch die Nutzung von TARDIS und seinen Services kann das Ziel erreicht und die Umsetzung erfolgen. Gleichzeitig hilft TARDIS die Komplexität und Verwaltung der Isolation zu vereinfachen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unberechtigter Zugang zum System
- Unberechtigte Einsichtnahme oder Abgriff von Daten
- Unbemerkt durchführbare Angriffe

ID: 3.02-28/9.1

Req 33	Der API-Provider muss das vom API-Gateway übermittelte Authentifizierungstoken verifizieren.
--------	--

Die folgenden Parameter sind zu verifizieren:

- Scope (Berechtigungsumfang) oder Identität,
- Ablauf-Zeitstempel,
- Signatur und Aussteller,
- Algorithmus (nicht „keiner“)

Motivation: Die Token bilden den Vertrauensanker und ermöglichen dem Provider den Consumer zu identifizieren und zu autorisieren. Eine fehlende oder mangelhafte Token-Verifikation ermöglicht den Abfluss von Daten oder den Missbrauch von Ressourcen.

Umsetzungsbeispiel: Die TARDIS Dokumentation bietet Hilfe und Anmerkungen dazu.

Quelle: https://developer.telekom.de/docs/src/tardis_customer_handbook/stargate/#to-do-for-api-provider

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: 3.02-29/9.1

Req 34	Payload-Verschlüsselung: Wenn der Datenschutz oder Informationsschutz es erfordern, müssen Consumer und Provider die Payload mit Ende-zu-Ende-Verschlüsselung sichern.
--------	--

Der Provider/Consumer muss sicherstellen, dass die beteiligten API-Gateways keinen Zugriff auf die Inhalte der Übertragung (Payload) bekommen. Das kann daran liegen, dass die Daten einer bestimmten VS-IT Kategorie oder einer bestimmten Datenschutzesklasse unterliegen und die Gateways diese Kategorie nicht verarbeiten bzw. keinen Zugriff darauf haben dürfen.

Quelle zur VS-IT-Kategorie sind die NOR-Betriebsmodelle: <https://yam-united.telekom.com/pages/it-security-privacy-dt-it/apps/wiki/wiki/list/view/34291ced-1e0f-475b-b785-7aa8fab3f2b8>.

Motivation: E2E Verschlüsselung ermöglicht es, dass System mit hohen Betriebsmodellen wie DE4 oder höher über

Transportsysteme wie TARDIS mit einem niedrigeren Betriebsmodell genutzt werden können. Auch die Verknüpfung von Systemen mit unterschiedlichen Betriebsmodellen wie DE4 zu DE3 wird einfacher umsetzbar, da die Verschlüsselung die höherwertigen Daten vor dem Zugriff durch das niederwertige System schützt..

ID: 3.02-30/9.1

4.6.2. API-Consumer spezifische Anforderungen

Diese Anforderungen sind nur relevant, wenn durch das System oder Service eine API genutzt/konsumiert wird. Wird durch das System oder den Service eine API ausschließlich angeboten (Provider), wird der Abschnitt als auf "Nicht Zutreffend" (=NA) markiert.

Req 35	Die Identität des Consumer muss durch einen zugelassenen Maschine-zu-Maschine Identity Provider (M2M-IDP) der API-Gateway Infrastruktur verwaltet und als Token beglaubigt werden.
--------	--

Motivation: API-Security@DTAG - Prinzip 1: Identifizierung, Authentifizierung, Autorisierung (Identitätsmanagement)

Die Implementierung robuster und anpassungsfähiger Authentifizierungsmechanismen ist von entscheidender Bedeutung. Genutzt werden Industriestandard-Protokolle wie OAuth 2.0 zur Validierung der Identität von M2M- und menschlichen Akteuren, die versuchen, auf APIs zuzugreifen. Dabei muss der gewählte Mechanismus auf das erforderliche Schutzniveau der Daten und die Schutzwürdigkeit der API innerhalb unserer Systemarchitektur abgestimmt sein. Statische API-Tokens sollten niemals eine Option für interne Dienste sein.

In Verbindung mit der Authentifizierung sind zuverlässige Autorisierungskontrollen unverzichtbar. Es muss die Möglichkeit bestehen, fein abgestufte Zugriffskontrollen zu definieren und durchzusetzen, um sicherzustellen, dass authentifizierte Entitäten nur auf die Ressourcen und Funktionalitäten Zugriff haben, die für ihre Rollen und Verantwortlichkeiten relevant sind. Dies bedingt eine präzise Zuweisung von Benutzerrollen zu bestimmten API-Berechtigungen sowie eine präzise abgestimmte Kontrollebene.

Ein effizientes Management von kurzlebigen Tokens spielt für die sichere (konzerninterne) API-Interaktion eine entscheidende Rolle. Die Lebensdauer der Tokens ist auf der Grundlage der Schutzwürdigkeit der Daten und der Geschäftsanforderungen zu definieren. Es müssen Mechanismen für den Widerruf und die Erneuerung von Tokens eingerichtet werden, um die Kontrolle über die Zugriffsberechtigungen zu wahren.

Die Verwendung eines (förderierten) zentralisierten IDP (Identity Providers) als einziger vertrauenswürdiger Identitätsanbieter ist ein wesentlicher Teil des Konzepts eines API-Gateways.

Umsetzungsbeispiel: TARDIS bietet alle notwendigen Funktionen und ist auch zugelassener ID Provider für M2M Identitäten. Darüber hinaus besteht die Möglichkeit der IDM Föderation. Nähere Hinweise für Consumer finden sich hier: https://developer.telekom.de/docs/src/tardis_customer_handbook/stargate/#to-do-for-api-consumer

ID: 3.02-31/9.1

Req 36	Der Consumer muss prüfen, dass er mit dem API-Gateway direkt verbunden ist, beispielsweise durch Prüfung des TLS Zertifikats.
--------	---

Die Prüfung kann durch weitere Mechanismen unterstützt werden, beispielsweise:

- Filterung der IP-Adresse,
- Security Policies in der jeweiligen (Kubernetes-)Umgebung,

Motivation: Eine große Gefahr für die Vertraulichkeit und den Missbrauch von Ressource geht von Man-in-the-Middle (MiM) Attacken aus. Der fundamentale Schutz der Authentisierungsmerkmale (Token) und Geheimnisse (Credentials) liegt in der TLS (Transport Layer Security) Ebene. Um die Komplexität niedrig zu halten, wird in der Regel auf eine beidseitige, zertifikatsbasierte Authentisierung verzichtet. Nur das Gateway weist sich gegenüber dem Consumer als HTTPS-Entität aus. Dadurch kann nur der Client, durch eine Überprüfung des Gateway-TLS-Zertifikats einen MitM Angriff erkennen und die Verbindung ablehnen.

ID: 3.02-32/9.1

4.6.3. Spezialfall: API-Provider für Human to Machine (H2M, Mensch-zu-

Maschine) Schnittstellen

Verweis auf die Guiding Principles Front End: <https://wiki.telekom.de/pages/viewpage.action?pagelId=886833714>

Req 37 APIs zwischen Browser und Backend müssen durch ein Backend-for-Frontend (BFF) terminiert werden.

Benutzerorientierte APIs: Je nach NatCo oder LBU verlangen die Architekturrichtlinien die Nutzung eines Backend-for-Frontends (BFF) als eine Abstraktionsebene anstelle eines API-Gateways ([Erläuterung von BFF in DTAG](#)). Das BFF kapselt die Web-Session des Menschen und trennt die gegenüber den Backend-Aufrufen auf der Maschine zu Maschine-Ebene ab.

Motivation: Durch die Trennung wird zum einen ermöglicht die Anmeldung und Nutzung durch den Menschen (Web-Session) zu überwachen und zu beenden. Gleichzeitig wird sichergestellt, dass Zugriffs-Credentials/Token nicht zu einem unsicheren Client übertragen werden und ein Angreifer Zugriff auf die Backend-Systeme erhält.

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: 3.02-33/9.1

Req 38 Das BFF darf die Backend-M2M-Tokens nicht zu den Nutzenden (Browser) weitergeben.

Das Backend-M2M-Token ermöglicht den Zugriff auf die internen Services zur Verarbeitung der Geschäftsprozesse. Ist das Rollen- und Berechtigungskonzept im BFF umgesetzt, führt das zu einer großen Angriffsfläche und der Gefahr von Datenlecks oder Ressourcenmissbrauch.

Motivation: API Security @ DTAG Abschnitt 4.4 und 4.5

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: 3.02-34/9.1

Req 39 Das BFF muss zur Interaktion mit den Backend-Services die API-Infrastruktur (M2M) als regulärer Consumer oder Provider nutzen.

Das Backend-For-Frontend (BFF) trennt die Web-Session zum Browser / Mensch, von der fachlichen Datenverarbeitung im Backend. Auf der Backend-Service Seite nutzt es die API-Gateway Infrastruktur zur Interaktion mit API-Providern oder zum Anbieten eigener APIs.

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: 3.02-35/9.1

5. Protokollierung

Req 40	Ereignisse müssen mit genauem Zeitstempel protokolliert werden und je nach ihrer Kritikalität Alarme auslösen. Die Protokollierung von Benutzern / Benutzeraktionen, sowie die Payloads sind mit dem Datenschutz und der Datensicherheit abzustimmen.
--------	--

Ereignisse müssen entsprechend ihrer Ereignisart eine Ereignis-Kennzeichnung erhalten.

Sicherheitsrelevante Ereignisse müssen besonders gekennzeichnet werden um bedarfsgerecht Alarme auslösen zu können.

Als Ereignis-Kennzeichnung eignen sich z. B. „info“, „error“, „warning“, „alert“, „emergency“.

Anhand dieser Ereignis-Kennzeichnungen können automatisiert Alarme ausgelöst werden und Fehlerbehebungen erleichtert werden.

Ebenso ist es möglich, anhand dieser Ereignis-Kennzeichnungen Löschfristen und Zugriffsberechtigungen zu steuern.

Motivation: Damit ein sicherer Betrieb sowie eine Fehleranalyse möglich ist, muss ein "Logging, Monitoring und Alarming" Konzept erstellt werden.

Sicherheitsrelevante Ereignisse müssen sofort an einen geeigneten Betrieb weitergeleitet werden, der das Problem analysieren und Gegenmaßnahmen ergreifen kann.

Umsetzungsbeispiel: Die Ereignis-Kennzeichnung sowie die Notwendigkeit eines Alarms richtet sich nach dem Schutzbedarf und den Schnittstellen, über die der Web Service erreichbar ist.

Bei einem Web Service, der aus dem Internet erreichbar ist, muss eher mit Validierung und Autorisierungsfehlern gerechnet werden. Deshalb ist hier ein Alarm nicht immer zielführend. Sollte dies jedoch bei einem Webservice auftreten, der nur aus vertrauenswürdigen Bereichen erreichbar ist, z. B. aus dem eigenen Namespace, administrative Sondernetze oder M2M Verbindungen, kann das ein Indiz für einen vorgedungenen Angreifer sein.

Ebenfalls kann die Notwendigkeit für einen Alarm sich nach der Häufigkeit des Ereignis richten.

Ein Zugriffsversuch auf eine Ressource ohne entsprechende Berechtigung oder ein Validierungsfehler kann bei einmaligen Auftreten irrelevant sein. Sollte dies gehäuft auftreten, kann das ein Indiz für einen gezielten Angriff auf die Applikation sein oder es liegen Fehlfunktionen der Applikation vor.

Die folgende Tabelle gibt Beispiele für Ereignisse, Ereignis-Kennzeichnungen und auch den zu speichernden Logkontext.

Kategorie	Ereignis	Kennzeichnung	Logkontext
Zugriff	Zugriff auf den Web Service	info	Applikations-, Container-ID, Authentizität, IP
Validierungsfehler	falsches Datenformat	error, debug	Applikations-, Container-ID, Authentizität, IP, Payload
	Falsches Encoding	error, debug	Applikations-, Container-ID, Authentizität, IP, Payload
	Falsche Datenfeldnamen	error, debug	Applikations-, Container-ID, Authentizität, IP, Payload
	Prüfung gegen formelle Spezifikation schlägt fehl	error, debug	Applikations-, Container-ID, Authentizität, IP, Payload
	Falscher Wertebereich	error, debug	Applikations-, Container-ID, Authentizität, IP, Payload

Ausgabefehler	Datenbank / zuliefernde Systeme nicht erreichbar	error, debug	Applikations-, Container-ID, Authentizität, Anfrage-ID, IP
	Fehler in den Datensätzen der Datenbank	error, debug	Applikations-, Container-ID, Authentizität, Anfrage-ID, IP
	Ausgabe unvollständig	error, debug	Applikations-, Container-ID, Authentizität, Anfrage-ID, IP
Autorisierungsfehler	Zugriff auf Ressourcen ohne entsprechende Autorisierung	warning	Applikations-, Container-ID, Authentizität, IP, Payload
	Signaturvalidierungsfehler bei JW Token	alert	Applikations-, Container-ID, Authentizität, IP, Payload
	Durchführen von Aktionen die nicht auf die Rolle / ACL passen	alert	Applikations-, Container-ID, Authentizität, IP, Payload
Authentifizierung	Fehlschläge	warning	Applikations-, Container-ID, IP
	Erfolg	info	Applikations-, Container-ID, IP
Privilegierte Aktionen	Hinzufügen / Löschen von Accounts	trace	Applikations-, Container-ID, Authentizität, IP
	Ändern von Privilegien	trace	Applikations-, Container-ID, Authentizität, IP
	Exportieren von schützenswerten Daten	trace	Applikations-, Container-ID, Authentizität, IP
	Import von Daten	trace	Applikations-, Container-ID, Authentizität, IP
	Ändern von Konfigurationen	trace	Applikations-, Container-ID, Authentizität, IP
Laufzeitfehler	Absturz / Neustart der Applikation / Container	emergency	Applikations-, Container-ID, Stacktrace
	Konnektivitätsprobleme	emergency	Applikations-, Container-ID
	Performance Probleme	alert	Applikations-, Container-ID
	Viren Erkennung z. B. File Upload	alert	Applikations-, Container-ID, Authentizität, IP

Für diese Anforderung sind folgende Bedrohungen relevant:

- Abstreiten von durchgeführten Aktionen
- Unbemerkt durchführbare Angriffe

ID: 3.02-37/9.1

Req 41	Für sicherheitsrelevante Protokollierungsdaten, die lokal erfasst werden, müssen geltende Aufbewahrungszeiträume und Löschfristen eingehalten werden.
--------	---

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.
(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.)
- Nach Ablauf von 90 Tagen muss eine umgehende Löschung von gespeicherten Protokollierungsdaten erfolgen.

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 E-Mail 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: 3.01-34/9.0

Req 42	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

Req 43	Sicherheitsrelevante Protokollierungsdaten müssen echtzeitnah an eine Erkennungs- und Reaktionslösung für Sicherheitsvorfälle bereitgestellt werden.
--------	--

Eine zulässige Erkennungs- und Reaktionslösung im Sinne dieser Anforderung muss die folgenden Kriterien erfüllen:

- Es erfolgt eine kontinuierliche Analyse bzw. Korrelation der aus angebundenen IT/NT-Systemen bereitgestellten Protokollierungsdaten.
- Die Identifizierung bedrohlicher Aktivitäten und laufender Angriffe findet auf Basis einer fortlaufend aktuell ge-

haltenen Erkennungsmuster-Datenbank statt.

- Integrierte Alarmierungsprozesse lösen im Fall eines Fundes die Einleitung von Gegenmaßnahmen aus.
- Sicherheitsexperten sind aktiv involviert in den Betrieb der Lösung sowie die Auswertung von Alarmen.

Eine solche Lösung ist in ihrer Funktion nicht mit einem Log Server gleichzusetzen. Entsprechend muss die Bereitstellung an eine Erkennungs- und Reaktionslösung stets zusätzlich zu einer vorhandenen Weiterleitung an einen Log Server erfolgen. Eine Ausnahme besteht, falls der genutzte Log Server seinerseits die Protokollierungsdaten umgehend nach Erhalt zusätzlich an eine Erkennungs- und Reaktionslösung bereitstellt. In diesem Fall ist es ausreichend, dass das System die Protokollierungsdaten nur an den Log Server weiterleitet.

Die Protokollierungsdaten müssen ab dem Zeitpunkt ihrer Erzeugung im System innerhalb von 5 Minuten an die Erkennungs- und Reaktionslösung bereitgestellt werden ("echtzeitnah").

Die konkret zu unterstützenden technischen Schnittstellen und Protokolle für die Bereitstellung werden durch die im Einsatzumfeld des Systems verfügbare Erkennungs- und Reaktionslösung sowie die technische Basis des Systems bestimmt.

Motivation: IT/NT-Systeme sind permanent sicherheitstechnischen Bedrohungen und Angriffen ausgesetzt. Mittels einer zentralen Analyse und der Korrelation erfasster System-Ereignisse ganzer IT-Landschaften in nahezu Echtzeit können aktuelle Bedrohungen und laufende Angriffe auch über Systemgrenzen hinaus erkannt und zeitnah passende Gegenmaßnahmen eingeleitet werden. Die unverzügerte Bereitstellung von Protokollierungsdaten an die Erkennungs- und Reaktionslösung ist essenziell für deren Effektivität.

Umsetzungsbeispiel: **Beispiel 1:** Ein System auf Basis eines klassischen Servers erfasst sicherheitsrelevante Protokollierungsdaten im lokalen Syslog oder Windows Event Log des Betriebssystems. Die Protokollierungsdaten werden kontinuierlich zur weiteren Analyse, Angriffserkennung und Alarmbehandlung an ein SIEM (Security Information and Event Management) weitergeleitet.

Beispiel 2: Auf einem System wird ein EDR ("Endpoint Detection and Response") Agent ausgeführt. Der EDR-Agent leitet die von ihm auf dem System erfassten sicherheitsrelevanten Protokollierungsdaten kontinuierlich an das zentrale Backend der EDR-Lösung aus, auf dem die weitere Analyse, Angriffserkennung und Alarmbehandlung erfolgt.

Beispiel 3: Netzwerk-Komponenten stellen Ereignisse an ein SIEM bereit, die es ermöglichen, verdächtiges Kommunikationsverhalten wie Netzwerk-Scans, unzulässige Datenflüsse zwischen Netz-Segmenten oder aktive Verbindungen bzw. Verbindungsversuche zu verdächtigen Zielen im Internet zu entdecken. Im speziellen liefern Gateway Router aus dem CNDTAG "Netflow" Logs, während Forward Proxies Zugriffe auf Domänen und/oder IP-Adressen melden und Firewalls blockierte Verbindungsversuche loggen.

Beispiel 4: Auf Basis von Cloud-nativen Funktionen entwickelte Systeme erzeugen Protokollierungsdaten mittels der Cloud-eigenen Protokollierungs- und Überwachungsinfrastruktur direkt aus diesen Funktionen heraus. In einem solchen System wird die Erzeugung sicherheitsrelevanter Protokollierungsdaten in den entsprechenden Konfigurationsoptionen der eingesetzten Cloud-nativen Funktionen gezielt aktiviert. Die entstehenden Protokollierungsdaten werden kontinuierlich zur weiteren Analyse, Angriffserkennung und Alarmbehandlung an ein SIEM (Security Information and Event Management) übertragen. Einige Cloud-Anbieter leisten eine gewisse Angriffsmustererkennung bereits in der Cloud-eigenen Überwachungsinfrastruktur und erzeugen daraus spezifische Cloud-Alarme. Diese werden zusätzlich an das SIEM übertragen.

Beispiel 5: Ein System basiert auf einer SaaS ("Software as a Service") Applikation eines Cloud-Anbieters. Typischerweise verantwortet im SaaS Modell der Cloud-Anbieter alle operativen Aspekte der Applikation, inkl. des Sicherheitsmanagements. Der Cloud-Kunde selbst hat keinen Einfluss auf die zugrunde liegende technische Infrastruktur oder den Betrieb der Applikation. Anhand von Vertragsvereinbarungen, anerkannten Zertifikaten und dokumentierten Selbstauskünften des Cloud-Anbieters wird formell überprüft, dass der Cloud-Anbieter eine Erkennungs- und Reaktionslösung für Sicherheitsvorfälle implementiert hat. Einige Cloud-Anbieter erzeugen zusätzlich dedizierte Protokollierungsdaten über sicherheitsrelevante Aktivitäten innerhalb der separaten Kunden-Mandanten ihrer SaaS Applikation (beispielsweise zu Änderungen von Konfigurationseinstellungen, Berechtigungsvergaben oder Benutzer-Konten innerhalb der Mandanten) und stellen diese den Cloud-Kunden zu Verfügung. Diese sogenannten Audit-Logs werden kontinuierlich zur weiteren Analyse, Angriffserkennung und Alarmbehandlung an ein SIEM (Security Information and Event Management) übertragen.

Für diese Anforderung sind folgende Bedrohungen relevant:

- Unbemerkt durchführbare Angriffe

6. Web Services

6.1. TLS

Req 44	Es muss TLS in der Version 1.2 oder 1.3 verwendet werden.
--------	---

Nutzerrollen: Betrieb, Entwicklung, Integration

TLS (Transport Layer Security) ist ein Protokoll zur sicheren Übertragung von Daten über TCP/IP-basierte Verbindungen und der Nachfolger von SSL (Secure Sockets Layer). Es schützt die Vertraulichkeit, Integrität und Authentizität der Daten bzw. der Kommunikationspartner.

TLS in der Version 1.2 [RFC-5246] und Version 1.3 [RFC-8446] bietet Cipher Suites mit Authenticated Encryption Associated Data (AEAD). AEAD bietet neben dem Schutz der Vertraulichkeit auch Integrität sowie Authentizität der übertragenen Daten.

Quellen:

[RFC-5246] The Transport Layer Security (TLS) Protocol Version 1.2

[RFC-8446] The Transport Layer Security (TLS) Protocol Version 1.3

Motivation: Die aktuelle Version von TLS schließt bisherige Sicherheitsschwachstellen und Angriffsflächen auf den TLS-Protokoll-Handshake.

Umsetzungsbeispiel: OpenSSL> protocol = tlsv1_3

ID: 3.02-16/9.1

Req 45	Es müssen die folgenden Cipher Suites mit Perfect Forward Secrecy (PFS) verwendet werden: für TLS v1.3: IANA-Nr. "0x13,0x01", "0x13,0x02", "0x13,0x03" or "0x13,0x04", sowie für TLS v1.2: IANA-Nr. "0xC0,0x2B", "0xC0,0x2C", "0xC0,0xAC", "0xC0,0xAD", "0xC0,0x2F", "0xC0,0x30", "0xCC,0xA9" oder "0xCC,0xA8".
--------	---

Nutzerrollen: Betrieb, Entwicklung, Integration

Cipher Suites spezifizieren die kryptographischen Verfahren einer TLS-Verbindung.

Perfect Forward Secrecy (kurz PFS, auch Forward Secrecy) bedeutet, dass Verbindungsinhalte auch bei Kenntnis der Langzeit-Schlüssel der Kommunikationspartner nicht nachträglich entschlüsselt werden können.

In TLS v1.2 werden Cipher Suites wie folgt definiert: *TLS_AKE_WITH_Enc_Hash*.

Nachfolgend wird die Bedeutung der einzelnen Bestandteile erläutert:

- *AKE (Authenticated Key Exchange)*: Authentisiertes Schlüsseleingangsverfahren für das Handshake-Protokoll
- *Enc (Encryption)*: Verschlüsselungsverfahren mit Betriebsart für das Record-Protokoll
- *Hash*: Hashfunktion für einen HMAC zur Schlüsselableitung. Sofern *Enc* kein AEAD-Verschlüsselungsverfahren ist, wird der HMAC außerdem zur Integritätssicherung verwendet.

Die nachfolgende Tabelle listet die zulässigen Cipher Suites mit PFS in TLS v1.2 sowie die Referenzspezifikationen auf. Dabei wurde der Designphilosophie von TLS v1.3 gefolgt, weshalb in dieser Tabelle lediglich AEAD-Konstruktionen aufgelistet sind.

Cipher Suite	IANA-Nr.	Referenzspezifikation
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	0xC0,0x2B	[RFC-5289]

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	0xC0,0x2C	[RFC-5289]
TLS_ECDHE_ECDSA_WITH_AES_128_CCM	0xC0,0xAC	[RFC-7251]
TLS_ECDHE_ECDSA_WITH_AES_256_CCM	0xC0,0xAD	[RFC-7251]
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	0xC0,0x2F	[RFC-5289]
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	0xC0,0x30	[RFC-5289]
TLS_ECDHE_ECDSA_WITH_CHACHA20POLY1305_SHA256	0xCC,0xA9	[RFC-7905]
TLS_ECDHE_RSA_WITH_CHACHA20POLY1305_SHA256	0xCC,0xA8	[RFC-7905]

Die folgenden Cipher Suites mit PFS in TLS v1.2 mit AES-CBC (CBC-Modus) und / oder DHE Cipher Suites sind nur in Bestandsystemen erlaubt:

Cipher Suite	IANA-Nr.	Referenzspezifikation	Hinweis
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	0xC0,0x23	[RFC-5289]	CBC
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	0xC0,0x24	[RFC-5289]	CBC
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	0xC0,0x27	[RFC-5289]	CBC
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	0xC0,0x28	[RFC-5289]	CBC
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	0x00,0xA2	[RFC-5288]	DHE
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	0x00,0xA3	[RFC-5288]	DHE
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	0x00,0x9E	[RFC-5288]	DHE
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	0x00,0x9F	[RFC-5288]	DHE
TLS_DHE_RSA_WITH_AES_128_CCM	0xC0,0x9E	[RFC-6655]	DHE
TLS_DHE_RSA_WITH_AES_256_CCM	0xC0,0x9F	[RFC-6655]	DHE

TLS_DHE_RSA_WITH_CHACHA20POLY1305_SHA256	0xCC,0xAA	[RFC-7905]	DHE
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	0x00,0x40	[RFC-5246]	CBC / DHE
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	0x00,0x6A	[RFC-5246]	CBC / DHE
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	0x00,0x67	[RFC-5246]	CBC / DHE
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	0x00,0x6B	[RFC-5246]	CBC / DHE

Hinweis zu DHE Cipher Suites:

Für das authentifizierte Schlüsseleinigungsverfahren (AKE) werden Verfahren basierend auf elliptischen Kurven (ECDHE_ECDSA) bevorzugt. DHE (diskreter Logarithmus) Schlüsselvereinbarungen sind anfälliger gegen DoS (DHeater) Angriffe als ECDHE (elliptische Kurven). Sie werden voraussichtlich vom IETF abgekündigt und vom BSI [TR-02102-2] nur bis zum Jahr 2029 empfohlen.

Hinweis zu Cipher Suites mit AES-CBC:

Cipher Suites mit AES-CBC sind nur mit der TLS-Erweiterung "Encrypt-then-MAC" vom BSI empfohlen ([TR-02102-2]). Die Erweiterung ist aber in den meisten Implementierungen nicht aktivierbar bzw. eine Überprüfung, ob sie aktiv ist, ist nicht möglich.

In TLS v1.3 werden Cipher Suites wie folgt definiert: *TLS_AEAD_Hash*.

Nachfolgend wird die Bedeutung der einzelnen Bestandteile erläutert:

- *AEAD*: Authentisiertes Verschlüsselungsverfahren für das Record-Protokoll
- *Hash*: Hashfunktion für HMAC und HKDF im Handshake-Protokoll

Die nachfolgende Tabelle listet die empfohlenen Cipher Suites mit PFS in TLS v1.3 sowie die Referenzspezifikationen auf.

Cipher Suite	IANA-Nr.	Referenzspezifikation
TLS_AES_128_GCM_SHA256	0x13,0x01	[RFC-8446]
TLS_AES_256_GCM_SHA384	0x13,0x02	[RFC-8446]
TLS_CHACHA20_POLY1305_SHA256	0x13,0x03	[RFC-8446]
TLS_AES_128_CCM_SHA256	0x13,0x04	[RFC-8446]

Hinweis zu CHACHA20 Cipher Suites:

CHACHA20 Cipher Suites: werden nicht (explizit) vom BSI empfohlen, aber bisher sind keine Schwachstellen bekannt, so dass sie als sicher einzustufen sind.

Hinweis zu PQC:

Wenn eine hybride Implementierung von klassischer und Post-Quanten-Kryptographie verwendet werden (erfordert TLS 1.3), dann muss der klassische Teil dieser Anforderung entsprechen.

Alle zukünftig spezifizierten Cipher Suites, die den in diesem Dokument definierten Anforderungen entsprechen, können ebenfalls verwendet werden. Dies gilt beispielsweise für Cipher Suites, die eine Hash-Funktion aus der SHA-3-Familie verwenden.

Quellen:

[RFC-5246] The Transport Layer Security (TLS) Protocol Version 1.2

[RFC-5288] AES Galois Counter Mode (GCM) Cipher Suites for TLS
[RFC-5289] TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)
[RFC-6655] AES-CCM Cipher Suites for Transport Layer Security (TLS)
[RFC-7251] AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS
[RFC-7905] ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)
[RFC-8446] The Transport Layer Security (TLS) Protocol Version 1.3
[TR-02102-2] BSI-Technische Richtlinie: Kryptographische Verfahren - Empfehlungen und Schlüssellängen - Teil 2
TLS, Version 2025-01

Motivation: Der Einsatz von modernen Cipher Suites mit Perfect Forward Secrecy schützt die Transport-Sicherheit in TLS.

Umsetzungsbeispiel: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

ID: 3.02-18/9.1

Req 46 Für TLS müssen die Diffie-Hellman Gruppen mit IANA-Nr. 23-33 verwendet werden.

Nutzerrollen: Betrieb, Entwicklung, Integration

Die Diffie-Hellman Gruppe wird zum Schlüsselaustausch mit Perfect Forward Secrecy (PFS) verwendet.

Die folgende Tabelle beinhaltet die zulässigen Diffie-Hellman Gruppen:

Diffie-Hellman Gruppe	IANA-Nr.	Referenzspezifikation	Hinweis
secp256r1	23	[RFC 8422]	
secp384r1	24	[RFC 8422]	
secp521r1	25	[RFC 8422]	
brainpoolP256r1	26	[RFC 7027]	TLS 1.2
brainpoolP384r1	27	[RFC 7027]	TLS 1.2
brainpoolP512r1	28	[RFC 7027]	TLS 1.2
x25519	29	[RFC-8422]	
x448	30	[RFC-8422]	
brainpoolP256r1tls13	31	[RFC 8734]	TLS 1.3
brainpoolP384r1tls13	32	[RFC 8734]	TLS 1.3
brainpoolP512r1tls13	33	[RFC 8734]	TLS 1.3
ffdhe3072	257	[RFC 7919]	
ffdhe4096	258	[RFC 7919]	

Hinweis zu x448 und x25519:

x448 und x25519 werden nicht (explizit) vom BSI empfohlen, aber bisher sind keine Schwachstellen bekannt, so dass sie als sicher einzustufen sind.

Hinweis zu Brainpool-Kurven:

Für die Brainpool-Kurven in TLS v1.2 [RFC-7027] wurden die IANA-Nummern 26, 27 und 28 reserviert, in TLS v1.3 [RFC-8446] die IANA-Nummern 31, 32 und 33.

Hinweis zu DH Gruppe 256:

Die Diffie-Hellman Gruppe 256 (IANA-Nr. 256) besitzt eine Schlüssellänge von 2048 Bit und darf nicht mehr eingesetzt werden.

Hinweis zu DH Gruppen 257 und 258:

Diese Diffie-Hellman Gruppen sind anfälliger gegenüber dem DHeater (DoS) Angriff als elliptische Kurven Diffie-Hellman. Daher wird empfohlen Brainpool oder NIST (secp) Kurven für die Diffie-Hellman Schlüsselvereinbarung zu nutzen.

Hinweis zu PQC:

Wenn eine hybride Implementierung von klassischer und Post-Quanten-Kryptographie verwendet werden (erfordert TLS 1.3), dann muss der klassische Teil dieser Anforderung entsprechen.

Quellen:

[RFC-7027] Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS)

[RFC-7919] Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)

[RFC-8422] Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier

[RFC-8734] Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS) Version 1.3

[TR-02102-2] BSI-Technische Richtlinie: Kryptographische Verfahren - Empfehlungen und Schlüssellängen - Teil 2 TLS, Version 2025-01

Motivation: Standardisierte Diffie-Hellman Gruppen verwenden sichere Parameter und beschleunigen den Schlüsselaustausch.

ID: 3.02-20/9.1

Req 47	Für TLS müssen digitale Zertifikate von einer Zertifizierungsstelle mit ausreichender Schlüssellänge und begrenzter Gültigkeit verwendet sowie korrekt validiert werden.
--------	--

Nutzerrollen: Betrieb, Entwicklung, Integration

Digitale Zertifikate werden in TLS zur Authentisierung während des TLS-Handshakes verwendet. TLS-Server müssen ein TLS-Zertifikat verwenden, Clients benötigen ein Zertifikat falls gegenseitige Authentisierung gefordert ist (mutual authentication).

TLS-Zertifikate für Webserver, die aus dem Internet erreichbar sind, müssen von einer öffentlichen Zertifizierungsstelle ausgestellt werden, die von Browsern und Betriebssystemen als Vertrauenswürdig eingestuft ist. Diese können z.B. über den Dienst „TeleSec ServerPass“ beauftragt werden (siehe <https://www.telesec.de/de/produkte/serverpass/ueberblick>).

Bei Schlüssellängen, Gültigkeit und weiteren Konfigurationsmöglichkeiten muss die Certificate Policy der Zertifizierungsstelle beachtet werden.

Bei Webservern, die ausschließlich für interne Anwendungen verwendet werden und die nicht über das Internet erreichbar sind können digitale Zertifikate von einer privaten (internen) Public Key Infrastruktur verwendet werden.

Die Mindestanforderungen aus der folgenden Tabelle müssen für jede Form von TLS-Zertifikaten, d.h. auch für Client-zertifikate, beachtet werden:

Algorithmenfamilie	Schlüssellänge	Hashfunktion
Elliptic Curve	250 Bit	SHA-3, SHA-2 mit einer Ausgabelänge 256 Bit
Digital Signature Algorithm (DSA)	3000 Bit	SHA-3, SHA-2 mit einer Ausgabelänge 256 Bit
RSA	3000 Bit	SHA-3, SHA-2 mit einer Ausgabelänge 256 Bit

Einschränkungen zu SHA-224/SHA-3-224:

SHA-224/SHA-3-224 darf nur noch in Bestandssystemen verwendet werden und muss bei nächster Gelegenheit gegen einen Hashalgorithmus mit einer Output-Länge von mindestens 256 Bit ausgetauscht werden.

Die Zertifikatslaufzeit von öffentlichen TLS-Serverzertifikaten (ausgestellt von einer Zertifizierungsstelle, die Zertifikate entsprechend der Vorgaben des [CA/Browser Forum] ausstellt) darf maximal 397 Tage betragen. Für andere, interne TLS-Zertifikate sollte eine Laufzeit von maximal 3 Jahren nicht überschritten werden.

Die Zertifikatslaufzeit für automatisch ausgestellte und automatisch erneuerbare Zertifikate, z.B. bei ACME (Automatic Certificate Management Environment), sollte möglichst kurz gewählt werden. Zu empfehlen ist eine Laufzeit von maximal 90 Tagen.

Zertifikats-Validierung (vgl. [RFC-5280]):

- Die Zertifikate müssen gegen eine wohldefinierte Trust-Anchor-Liste validiert werden, die auf die nötigen Trust-Anchors reduziert sind. Dabei können Intermediate Zertifikate als Trust-Anchor verwendet werden.
- Bei der Validierung von Serverzertifikaten muss eine Hostname Validation durchgeführt werden.
- Die Zertifikate müssen gültig sein.
- Wenn in der Zertifikatskette „Name Constraints“ definiert sind, müssen alle nachfolgenden Zertifikate in der Kette diese „Name Constraints“ einhalten.

Quellen:

[CA/Browser Forum] <https://cabforum.org/working-groups/server/baseline-requirements/documents/>

[RFC5280] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, chapter 6

Motivation: Digitale Zertifikate bilden die Basis der Authentisierung und schaffen Vertrauen. Ohne ausreichend starke Authentisierung sind Man-in-the-Middle Angriffe möglich.

ID: 3.02-22/9.1