Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Empfehlungen zum sicheren Einsatz von APIs in der Bundesverwaltung

Digitale Verwaltung vernetzt denken und gestalten

Klassifizierung	nicht klassifiziert
Status	genehmigt zur Nutzung
Programmname	Strategie Digitale Bundesverwaltung Schwerpunkt 1: Digitale Verwaltung vernetzt denken und gestalten
Initiativenleiter	Karen Dijkstra (BK)
Version	1.0
Datum	11.08.2025
Auftraggeber	Digitale Transformation und IKT-Lenkung (DTI)
Autor/Autoren	Andreas Grünert (NCSC), Karen Dijkstra (BK)
Beteiligte Fachgruppen	API Community, Architekturboard Bund (ABB)
Genehmigende Stelle	Andreas Spichiger (BK)

Inhaltsverzeichnis

1 Ausgangslage

Elektronische Softwareschnittstellen (Application Programming Interfaces, APIs) sind eine Voraussetzung, um Leistungen der Bundesverwaltung digital zur Verfügung zu stellen. Für eine Kommunikation von Maschine zu Maschine braucht es solche standardisierten elektronischen Schnittstellen. Dabei kommuniziert ein API-Client in einem definierten und maschinenlesbaren Format mit einem API-Endpoint. Der API-Endpoint ist der Standort der API und ist mit der Fachanwendung verbunden. Der API-Client baut im Normalfall[1] die Verbindung zum API-Endpoint auf. Dabei kann es sich um eine Mobile App, einen Browser oder auch eine Software, welche auf einem Server läuft, handeln.

Mit Inkrafttreten des Bundesgesetzes über den Einsatz elektronischer Mittel zur Erfüllung von Behördenaufgaben (EMBAG) ist die Bundesverwaltung verpflichtet sicherzustellen, dass der Datenaustausch innerhalb der Bundesverwaltung sowie mit den Kantonen, Gemeinden und Privaten über elektronische Schnittstellen abgewickelt werden kann[2]. Als Grundlage dazu wurden die API-Architektur Bund und ergänzende Guidelines erarbeitet und publiziert[3].

Das Design und die zu verwendenden API-Standards für diese Kommunikation müssen so geplant und ausgewählt werden, dass die Integration und Bereitstellung von APIs einfach und sicher sind. So müssen sich zum Beispiel die Komponenten gegenseitig sicher authentisieren können, um dem Zero-Trust Prinzip[4] gerecht zu werden. Dies stellt neue Ansprüche an das Credential und Secret Management[5]. Weiter muss die Architektur kompatibel mit der API-Architektur Bund und den Sicherheitsverfahren in der Bundesverwaltung[6] sein. Zurzeit gibt es keine detaillierte Bundesweite Empfehlung zum sicheren Einsatz von APIs in der Bundesverwaltung der diese Herausforderungen berücksichtigt. Diese Lücke soll mit diesem Dokument geschlossen werden.

2 Problemstellung

Das Informationssicherheitsgesetz verlangt, dass Behörden, Gerichte, die Armee und weitere Organisation nach Art. 2 ISG auf ihre Informatikmittel (und dazu gehören auch APIs) ein Sicherheitsverfahren anwenden. Dieses besteht aus (a) einer Schutzbedarfsanalyse, mit welcher vor der Inbetriebnahme beurteilt wird, ob ein Risiko bestehen könnte, welches einen erhöhten Schutzbedarf rechtfertigt, (b) einem IT-Grundschutz, der Basisanforderungen an das Informatikmittel definiert und (c) einem Vorgehen bei erhöhtem Schutzbedarf.[7]

Dieses Sicherheitsverfahren ist agnostisch zu den eingesetzten Technologien und unabhängig der gewählten Architektur. Dies gilt auch für die Entscheidung, welcher API-Typ (XML basiert wie SOAP, binäre Formate wie protobuf oder JSON basierte, wie REST) zum Einsatz kommt.

Der IT-Grundschutz verlangt, dass die Ausgestaltung der Informatikmittel gewissen Prinzipien folgt: Das Prinzip *Stand der Technik*, bedeutet, dass überholte und nicht mehr unterhaltene Technologien nicht eingesetzt werden sollten. Sofern sie eingesetzt werden, sollten sie zeitnah und unabhängig vom «Life Cycle» nachgebessert oder ersetzt werden. Auch die Einhaltung der Prinzipien *Security by Design und Security by Default* wird erwartet. Security *by Default* bedeutet, dass die Informatikmittel so entwickelt, konfiguriert und betrieben werden, dass alle – in einem spezifischen Umfeld sinnvollen – Sicherheitsmassnahmen standardmässig aktiviert sind und ihre Wirkung entfalten können, ohne dass sich die Benutzerinnen und Benutzer darum kümmern müssen. Security *by Design* erwartet, dass bei der Entwicklung die Sicherheit von Anfang an ein integraler Bestandteil ist. Auch die Prinzipien *Zero Trust* (die Komponenten eines Systems vertrauen sich nicht implizit. Dies bedeutet, dass jede Interaktion authentisiert sein muss) und *Least Privilege* (die Vergabe von Zugriffsrechten und Privilegien muss minimal erfolgen) sind wichtig.

---

[1] Es gibt auch asynchrone APIs, wobei die Fachanwendung eine Antwort auf die Anfrage des API-Clients von sich aus später liefert und dazu eine Verbindung zum API-Client aufbaut. Diese API-Clients werden auch Webhook genannt.

[2] Siehe https://www.fedlex.admin.ch/eli/cc/2023/682/de, Art. 13 Schnittstellen

[3] https://www.bk.admin.ch/bk/de/home/digitale-transformation-ikt-lenkung/bundesarchitektur/api-architektur-bund.html

[4] Bei Zero Trust wird vorausgesetzt, dass sich alle Komponenten eines Systems (Software, Prozesse, Benutzer) nicht implizit vertrauen. Jede Interaktion muss daher authentisiert sein.

[5] Bei Secret Management geht es darum, wie Schlüssel, die eine Software selbst benötigt, um sich zu authentisieren, gespeichert werden. Die Schwierigkeit dabei ist, diese Schlüssel so zu sichern, dass sie von einem Angreifer, der Zugang zum Server hat, nicht einfach ausgelesen werden können.

[6] https://www.ncsc.admin.ch/ncsc/de/home/dokumentation/sicherheitsvorgaben-bund/sicherheitsverfahren.html - Siehe dazu auch Kapitel 2 Problemstellung

[7] Die Informationssicherheitsverordnung konkretisiert, dass dieses Verfahren nicht für jedes Informatikmittel und API separat durchgeführt werden muss, sondern dass diese Mittel zu Schutzobjekten aggregiert werden sollen, um sie im Gesamten zu beurteilen.

Obengenannte Prinzipien sind grundsätzlich einzuhalten. Es gibt im IT-Grundschutz keine Anforderungen an die Sicherheit, die ausschliesslich bei APIs relevant sind. Folgende richten sich aber insbesondere auch an APIs:

- Alle sicherheitsrelevanten Komponenten, Funktionen und Einstellungen müssen dokumentiert sein (Anforderung O2 aus IT-Grundschutz Si001).
- Der Ausfall von APIs muss beim IT-Service Continuity Management berücksichtigt werden (Anforderung O3).
- APIs müssen vor der Inbetriebnahme vor unberechtigtem Zugriff geschützt sein; wichtige sicherheitsrelevante Aktivitäten müssen protokolliert und ausgewertet werden (Anforderung T2.1). Protokollierung ist auch aufgrund von Datenschutz-Anforderungen notwendig.
- APIs müssen periodisch und wenn möglich automatisiert auf Schwachstellen geprüft werden (Anforderung T4).
- Die Authentisierung benötigt (mit Ausnahme von anonymen Zugriffen für welche die Anforderung Z2.1 des IT-Grundschutzes greift) mindestens ein Authentifikations- und Identitätsnachweismittel der Stufe «mittel» nach Anhang B des IT-Grundschutzes. Bei Messsystemen reicht die Stufe «tief». Das heisst, es sind auch bei einer Kommunikation mit APIs oft mehrere Authentisierungsfaktoren notwendig, ein API-Key allein genügt dann nicht[8] (Anforderung T6).
- Ein externer administrativer Zugriff auf das Informatikmittel ist nur unter bestimmten Voraussetzungen erlaubt (Anforderung T8.3). Dies hat Auswirkungen auf APIs mit administrativen Funktionen wie zum Beispiel die Konfiguration von Rollen und Benutzerrechten.
- Die Kommunikation über ein API umfasst oft auch einen Zonenübergang. Dieser muss geschützt sein und vor allem ist sicherzustellen, dass die Vorgaben der Zonenpolicy zum Zonenübergang umgesetzt werden (Anforderung Z1-3, Z5.1). Lösungen wie eine Web Application Firewall zum Schutz der API sind dabei entweder Teil der Policy Enforcement Zone (Anforderung Z4) oder ein Policy Enforcement Point innerhalb derselben Zone wie der API (nach Anforderung Z3).
- Die Vertraulichkeit, Integrität und Verfügbarkeit von geschäftsrelevanten Informationen müssen jederzeit ihrem Schutzbedarf entsprechend geschützt sein. Wenn erforderlich ist auch der physische Schutz zu gewährleisten (Anforderung I2, I3).
- Als Teil der Anforderungen aus dem IT-Grundschutz ist auch die Web-Proxy Policy (Si004) wichtig. Die Web-Proxy Richtlinie verlangt, dass ein Zugang vom IT-Netz des Bundes ins Internet über eine Proxy-Infrastruktur laufen muss. Ein API-Aufruf vom Bundesnetz auf eine API im Internet muss auch den entsprechenden Regeln folgen.

Für eine Organisation stellt sich also die Frage wie diese Anforderungen und Erwartungen in die Planung und Implementation von APIs bei neuen Informatikmitteln einfliessen kann. Die Empfehlungen und Best Practices in Kapitel 3 dieses Dokumentes sollen die IT-Sicherheitsverantwortlichen und Entwicklerteams dabei unterstützen.

Die Empfehlungen erklären dabei nicht wie eine API implementiert werden soll, dazu wird auf die Dokumente und weiteren Hilfsmittel der API-Architektur Bund verwiesen, sondern ausschliesslich was für die Sicherheit bei der Implementation berücksichtigt werden sollte. Die Empfehlungen berücksichtigen auch nicht wie die Sicherheit innerhalb der Fachanwendung oder bei der Bearbeitung durch den Client berücksichtigt werden soll. Weiter werden Anforderungen die bei erhöhtem Schutzbedarf (P042) zur Geltung kommen nicht spezifisch beleuchtet. Hier muss der Leistungsbezüger als Teil des Sicherheitsverfahrens eine Risikoanalyse durchführen und geeignete Massnahmen planen.

3 Empfehlungen entlang der Systemlandschaft API-Architektur Bund

Die folgenden Überlegungen basieren auf der Systemlandschaft der API-Architektur Bund[9] (Abbildung 1). Dabei wird für jede Komponente der Systemlandschaft beschrieben, welche Sicherheitsfunktionen diese normalerweise erfüllen, welche Anforderungen an die Sicherheit gelten sowie Best Practices wie diese umgesetzt werden können. Die Empfehlungen zeigen dabei mögliche Umsetzungen auf, diese sind aber selbst keine Vorgabe. Auch alternative

[8] Mehr Informationen zur Bedeutung dieser Stufen, sie SI001, Anhang B, sowie Umsetzungsempfehlung zur SI001 Authentifikation https://intranet.ncsc.admin.ch/ncscintra/de/home/dokumentation/empfehlung_technologiebetrachtung.html
[9] https://www.bk.admin.ch/bk/de/home/digitale-transformation-ikt-lenkung/bundesarchitektur/api-architektur-bund.html

Umsetzungen, welche die Anforderungen erfüllen, können eingesetzt werden. Es wird empfohlen bei der Umsetzung auch die jeweiligen technische Vorgaben des Leistungserbringers zu berücksichtigen.

Nicht jede API-Architektur wird alle Komponenten nutzen. Der Verzeichnisdienst und das Monetization System wird zum Beispiel nicht immer benötigt. Auch der API-Gateway selbst kann im Fachservice integriert sein und keine separate Komponente darstellen. Die Logging, Reporting und Montoring Komponente sowie die IAM-Komponente sind bei der Bundesverwaltung jedoch im Normfallfall notwendig.

Die Empfehlungen gelten insbesondere bei Software-Schnittstellen, welche für die Kommunikation über IT-Systemgrenzen hinaus benötigt werden. Es kann sich dabei um öffentliche Schnittstellen wie auch private Schnittstellen[10] handeln.

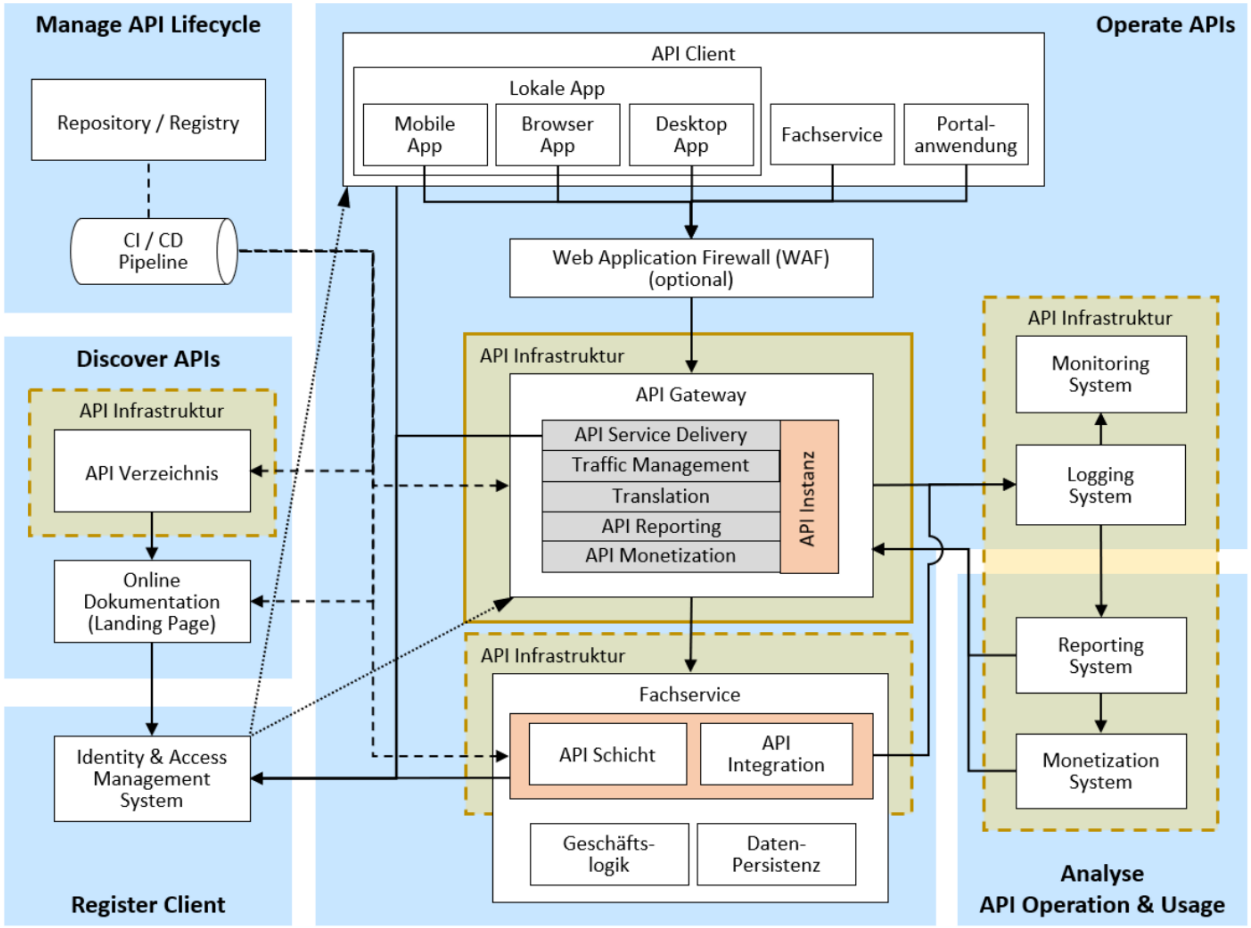

Abbildung 1: Systemlandschaft API-Architektur Bund

3.1 Komponente: API-Client

Funktion	Sicherheitsempfehlungen
Der API-Client authentisiert das API-Gateway, um die Authentizität des Gateways sicherzustellen.	Der API-Client verifiziert die Authentizität des API-Gateways und baut die Verbindung nur auf, wenn diese Verifikation stattgefunden hatte. **Best Practice** • Der API-Client verlangt für die Verbindung TLS 1.2 oder TLS 1.3. Bei APIs, welche ihre eigenen Kommunikationsprotokolle mitbringen kann, eine TLS Enkapsulierung verwenden werden.

[10] Mit privaten Schnittstellen sind in diesem Dokument und in der API-Architektur Bund APIs gemeint, die nur von API-Clients des gleichen Schuttobjektes verwendet werden können. Sicherheitsanforderungen gelten auch bei privaten Schnittstellen.

Funktion	Sicherheitsempfehlungen
STRIDE[11] Verhindern von Zugriffen des API-Clients auf gespoofte API-Gateways	• Bei langlebigen Zertifikaten sollte ein Prüfmechanismus bestehen, um revozierte Zertifikate zu erkennen (z. B. CRL, OCSP, Certificate Transparency oder auch CRLite). **Mapping zu Anforderungen** • Si001: Das Prinzip Zero Trust ist hier wichtig, der API-Client darf dem Server nicht implizit vertrauen.
Der API-Client authentisiert sich gegenüber dem API-Gateway, respektive dem Fachservice *STRIDE: Verhindern von Spoofing des API-Client gegenüber dem API-Gateway.*	Diese Funktion ist nur dann wichtig, wenn der API-Zugang nicht-anonyme API-Clients beinhaltet. Wenn ausschliesslich anonyme Zugriffe erlaubt sind, wird keine Authentisierung benötigt. Beim Verbindungsaufbau wird im Normalfall eine Multi-Faktor-Authentisierung (MFA) zwischen dem API-Client und dem API-Gateway, respektive dem Fachservice erwartet, um Identitätsdiebstahl (Spoofing) zu erschweren. Dies trifft bei interaktiven sowie nicht-interaktiven Authentisierungen zu. Findet die Authentisierung indirekt über Tokens statt, sollte die Gültigkeitsdauer dieser Tokens begrenzt sein und nicht verwendete oder kompromittierte Token widerrufen werden. Authentifikationsmittel oder Zugangsdaten müssen sicher generiert, gespeichert und erneuert werden. Session Informationen sollten so gespeichert werden, dass ein Cross Side Request Forgery (CSRF) nicht möglich ist. Die Folgenden Best Practices sollen bei der Planung unterstützen, sind aber nicht als abschliessend zu betrachten. Zudem ist es bei Authentisierungsmethoden wichtig diese im Einzelfall zu prüfen. Die Sicherheitsgarantien hängen immer auch von der jeweiligen Implementation und Konfiguration ab. **Best Practice** • Token basierte Authentisierung ○ Für die indirekte Authentisierung sollen die Tokens kryptografisch abgesichert (signiert, um eine unautorisierte Veränderung zu erkennen und verschlüsselt, um z. B. ein Replay Angriff zu verhindern oder um mögliche sensitive Daten, die als Teil der Session übertragen werden, zu schützen). ○ Wenn möglich sollten die Tokens auf eine dem Stand der Technik entsprechende Art und Weise an den Anwenderkontext (z.B. die «Session») gebunden sein. Auch dies kann einen Replay-Angriff erschweren. ○ Zudem sollte sichergestellt werden, dass die Erstellung der Tokens durch den Identity Provider (IdP) selbst nach den Anforderungen des IT-Grundschutzes (Stufe «mittel» im Normalfall oder «tief» bei Messsystemen gemäss Anforderung T6) erfolgt, insbesondere wenn das Token für den Zugang zu geschützten Ressourcen verwendet wird. Beispiele sind Kerberos-Tickets, SAML-Tokens und JSON Web Tokens (JWTs), wie sie z. B. als ID oder Access Tokens im Rahmen von OIDC und OAuth2 eingesetzt werden.

[11] STRIDE ist ein Modell von Sicherheitsrisiken, das ursprünglich von Loren Kohnfelder und Praerit Garg für die Bedrohungsmodellierung bei Microsoft entwickelt worden ist, und das heute weltweit eingesetzt wird. Der Name ist ein Akronym (Kunstwort), das sich aus den Anfangsbuchstaben der ursprünglich 6 Kategorien von Sicherheitsbedrohungen zusammensetzt, die im Rahmen von STRIDE unterschieden werden: Spoofing (Identitätsverschleierung), Tampering (Manipulation), Repudiation (Verleugnung), Information disclosure (Verletzung der Privatsphäre oder Datenabfluss), Denial of service (Verweigerung des Dienstes) und Elevation of privilege (Rechteausweitung). Diese Empfehlung verwendet das STRIDE Modell um bei den Sicherheitsfunktionen die relevanten Bedrohungen zu erkennen und darauf aufbauend die Anforderungen, Empfehlungen und Best Practices abzubilden.

Funktion	Sicherheitsempfehlungen

	• Direkte Authentisierung
	  ○ Für die direkte Authentisierung können API-Keys zusätzlich über einen weiteren Sicherheitsmechanismus wie Time based OTP (TOTP) ergänzt werden.
	  ○ Alternativ kann auch mTLS (mutual TLS) für die Authentisierung des API-Clients verwendet werden.
	  ○ Eine weitere Option stellen Software basierte FIDO2-Implementierungen wie Passkeys dar.
	  ○ Sofern beim Verbindungsaufbau keine genügende Authentisierung möglich ist, kann diese mit einer Verschlüsselung und Signierung des Payload kombiniert werden um zwei Faktoren zu erreichen.
	• Speicherung von Zugangsdaten:
	  ○ Ein Keychain, KMS oder Vault System sollte hierzu verwendet werden. Dieses gibt die Schlüssel nur wenn benötigt an den API-Client weiter oder macht die kryptographischen Aktionen gleich selbst, ohne die Schlüssel jemals preiszugeben.
	  ○ Um Sicherheitsrisiken wie Cross-Site Request Forgery zu vermeiden, sollte auf unsichere Methoden zur Übertragung von Zugangsdaten – etwa über URLs oder automatisch gesendete Cookies – verzichtet werden.
	  ○ Stattdessen ist, wo immer möglich, der Einsatz von kurzlebigen, temporären Zugangsdaten sinnvoll, da sie das Risiko bei einem allfälligen Datenabfluss deutlich reduzieren.
	Mapping zu Anforderungen
	• Si001: Es gelten Minimalanforderungen an die Authentifkationsstufen nach Anforderung T6. Sofern es sich um einen anonymen Zugriff handelt, gelten die Regeln nach Z2.1 a). In Jedem Fall müssen die Anforderungen ausgehend aus den Zonenpolicies (Z1-3) umgesetzt werden.
	• Bezüglich kurzlebiger Zugriffsrechte siehe auch T8.2, insbesondere wenn es sich um Administrative Zugriffe handelt.

Austausch von Daten über den API-Gateway *STRIDE: Verhindern von Tampering und Information Disclosure bei der Kommunikation zwischen API-Client und Gateway.*	Die Datenübertragung zwischen API-Client und dem API-Gateway soll verschlüsselt und gegen Integritätsangriffe geschützt sein. Wenn der API-Gateway innerhalb Bundesverwaltung mit einem Webhook (ein API-Endpoint, welcher vom API-Client für eine asynchrone Antwort verwendet wird) ausserhalb der Bundesverwaltung kommuniziert, muss die Proxy Infrastruktur, welche die TLS-Verbindung aufbricht (TLS Inspection) berücksichtigt werden. **Best Practice** • Beim Einsatz von TLS für die Verschlüsslung und den Integritätsschutz sollten Adversary in the Middle (AitM) und Session Hijacking Angriffe mitberücksichtigt und bestmöglich verhindert werden. Je nach Anwendungsfall können dazu Technologien, wie z. B. Mutual TLS, Token Binding oder im Umfeld von OAuth 2.0 auch Demonstrating Proof-of-Possession (DPoP) eingesetzt werden. Bei sensiblen Daten kann auch eine Payload Verschlüsselung und Signierung (z. b. JSON Web Signatures) in Betracht gezogen werden. • Wenn möglich ist eine Verbindung über ein Overlay oder über ein internes Netzwerk zu bevorzugen. **Mapping zu Anforderungen** • Si001: I2: Die Vertraulichkeit und Integrität von geschäftsrelevanten Informationen müssen jederzeit ihrem Schutzbedarf entsprechend und unter

Funktion	Sicherheitsempfehlungen
	Berücksichtigung der physischen Gegebenheiten mit Hilfe kryptografischer Verfahren geschützt sein. • Si004: Kommunikation von einer Zone innerhalb der Bundesverwaltung mit dem Internet muss über die Proxy Infrastruktur laufen.

3.2 Komponente: API-Gateway

Funktion	Sicherheitsempfehlungen
Das API-Gateway stellt sicher, dass nur authentisierte Clients Daten senden oder empfangen können *STRIDE: Verhindern von Spoofing des API-Clients*	Diese Funktion ist nur dann wichtig, wenn der API-Zugang nicht-anonyme API-Clients beinhaltet. Wenn ausschliesslich anonyme Zugriffe erlaubt sind, wird keine Authentisierung benötigt. Um zu verhindern, dass unautorisierte API-Clients Zugang zu den API-Dienstleistungen bekommen, soll das API-Gateway die Authentisierung von API-Clients systematisch überprüfen, bevor es das Senden oder Empfangen von Daten zulässt. Authentifikationsmittel oder Zugangsdaten müssen sicher generiert, gespeichert und erneuert werden. **Best Practice** • Hier gelten dieselben Best Practices zu Authentisierung und Speicherung von Zugangsdaten wie bei der Funktion «Der API-Client authentisiert sich gegenüber dem API-Gateway» weiter oben. • Bei TLS-Verbindungen muss zusätzlich berücksichtigt werden, an welcher Stelle die TLS-Verbindungen terminiert werden sollen: Sofern dies bereits bei einer Web Application Firewall (WAF) oder einem Reverse Proxy geschieht, sollte zwischen WAF respektive Reverse Proxy und API-Gateway auch TLS eingesetzt werden. • Bei sensiblen Daten, welche von Administratoren der WAF, Reverse Proxys oder des API-Gateways nicht eingesehen werden dürfen, sollte eine Payload Verschlüsselung und Signierung in Betracht gezogen werden. **Mapping zu Anforderungen** • Si001 T6: Mindestauthentifikationsstufen. • Si001 Z3 und Z4: API-Gateway ist ein Policy Enforcement Point (PEP) und befindet sich entweder in der Policy Enforcement Zone (PEZ) (Z4) oder als PEP in der Zielzone (Z3).
Der API-Gateway prüft die Rechte des API-Clients zuhanden des Fachservices. *STRIDE: Verhindern von Elevation of Privilege Angriffen des API-Clients zuhanden des Fachservices*	Das Gateway soll sicherstellen, dass jedem API-Client bei der Authentisierung die richtigen Rollen zugewiesen werden. Die Zuweisung soll zudem vor Integritätsangriffen geschützt sein. Sofern das API-Gateway auch eine Zugriffsteuerung vorzunehmen hat (d. h. Es entscheidet auf welche Fachservices der Client zugreifen darf), soll sichergellt werden, dass die Rollen für diese Entscheidung bekannt sind. Es ist möglich, dass das API-Gateway nicht über Rolleninformationen verfügt, da z.B. der zentrale Authentisierungsdienst diese nicht kennt. In diesem Fall muss die Verknüpfung der Rollen mit dem Client im Fachservice sichergestellt werden. **Best Practice** • Die Rollen sollen als Teil der geschützten Session, respektive den integritätsgeschützten Authentisierungstokens aufgenommen und so dem Fachservice weitergegeben werden.

Funktion	Sicherheitsempfehlungen
	• Die Rolle darf nicht als Attribut vom Client übernommen werden (z. B. ein Flag in einem TLS-Zertifikat), sondern darf nur über den Authentisierungsdienst bereitgestellt werden oder im API-Gateway oder Fachservice vorkonfiguriert sein. • Vor der Weiterleitung der Anfrage an den Fachservice soll verifiziert sein, ob der API-Client berechtigt ist auf den Fachservice zuzugreifen. **Mapping zu Anforderungen** • Si001: Das Prinzip Least Privilege, respektive die Sicherheitsanforderung T5.2 verlangen, dass Autorisierungsinformationen vorhanden sind und eine Autorisierung vorgenommen wird. • Si001 Z3 und Z4: Das API-Gateway ist eine PEP und befindet sich entweder in der PEZ (Anforderung Z4) oder als PEP in der Zielzone (Anforderung Z3).
Das API-Gateway steuert und überwacht den Netzwerkdatenverkehr und schützt dabei sich selbst sowie die anderen Komponenten der Systemlandschaft vor Angriffen. *STRIDE: Verhindern von Denial of Service sowie Information Discloure des API-Gateways und der Kommunikation vom API-Client und anderen Komponenten zum Gateway.*	Das API-Gateway soll den Netzwerkverkehr zu sich selbst und den Komponenten dahinter überwachen und steuern. Dazu gehört die Erkennung und Blockierung von bösartigen Übertragungsdaten. Bei der Erkennung von potenziell bösartigem Netzwerkverkehr soll auch die interne Kommunikation (z. B. zwischen dem Monetization System und dem Gateway) bei den Schutzmassnahmen berücksichtigt werden. Diese Steuerung des Netzwerkverkehrs wird oft auch von einem Load Balancer gemacht. In der API-Architektur Bund sind API spezifische Aspekte des Load Balancer Teil der API-Gateway Komponente. **Best Practice** • Begrenzung der Angriffsfläche, indem nur die notwendigen API-Endpoints des API-Gateways von ausserhalb des Systems oder des Schutzobjektes zugänglich sind. • Mit einer Web Application Firewall (WAF) können fehlerhafte Anfragen erkannt und blockiert werden, so dass sie die dahinter liegenden Fachservices gar nie erreichen. • Redundanz des API-Gateways und ein Verteilen der Last durch ein Load Balancing System, z. B. mit vorgelagerten Reverse Proxies, welche die eingehenden Daten auf verschiedene API-Gateways verteilen. Für das Load Balancing eignet sich oft DNS Round Robin mit einer sehr kurzen DNS Time-to-Live (TTL) Wert. • Regeln zur Ratenbegrenzung (Rate Limiting) können eine Überlastung des Fachservices verhindern. Ein dynamisches Rate Limiting kann bei laufenden Angriffen die Anzahl Anfragen pro API-Client einschränken oder Anfragen aus gewissen Gebieten blockieren und so die Verfügbarkeit für die autorisierten API-Clients gewährleisten. Rate Limiting kann auf den API-Gateways selbst, auf vorgelagerten Reverse Proxies (oder anderen Load Balancing Systemen) oder auch mithilfe eines vorgelagerten DDoS Prevention Systems umgesetzt werden. • Ein Network Intrusion Detection und Prevention System (NIDS/NIPS) kann Angriffsmuster erkennen und vor allem auch alarmieren und gegebenenfalls blockieren. Diese Echtzeitüberwachung und Erkennung von Anomalien ist wichtig, damit rechtzeitig reagiert werden kann. Ein NIDS kann auch einen möglichen (unautorisierten) Datenabfluss erkennen und rechtzeitig davor warnen. Weiter können Listen von bekannten bösartigen Mustern bei der Erkennung und automatische Blockierung berücksichtigt werden (z.B. GovCERT RPZ, MISP)[12]. Ein Intrusion Detection System ist normalerweise Teil der Logging Komponente der API, die Blockierung von Zugriffen Aufgrund einer Entscheidung dieses Systems (durch das Intrusion Prevention System) wird aber vom API-Gateway vorgenommen.

[12] https://www.ncsc.admin.ch/ncsc/de/home/infos-fuer/infos-it-spezialisten/informationen-govcert.html

Funktion	Sicherheitsempfehlungen
	• Wenn anonyme Zugriffe erlaubt sind, sollte eine Lösung eingesetzt werden, welche die Anzahl Anfragen pro (anonymen) API-Client einschränkt. • Um die eigenen Schutzmassnahmen zu prüfen, sollte regelmässig ein Schwachstellenscan auf das API-Gateway durchgeführt werden. **Mapping zu Anforderungen** • Si001 Z2.1: Anonyme Zugriffe müssen vor automatisierten Angriffen geschützt werden, das Hochladen von Dokumenten muss befristet sein. • Si001 Z5 - Überwachung: Die Kommunikation muss dahingehend überwacht werden, dass Angriffe möglichst zuverlässig erkannt werden können. Und der Betreiber im Bedarfsfall zeitnah und adäquat reagieren kann. • Si001 I3.1: Die Verfügbarkeit von geschäftsrelevanten Informationen muss jederzeit dem Schutzbedarf entsprechend sichergestellt sein. Dazu muss auch der Netzwerkverkehr gesteuert und überwacht werden. • Si001 T4: Da das API-Gateway normalerweise im Internet exponiert ist, muss auch eine, vorzugsweise automatisierte, regelmässige Prüfung der Sicherheit und möglicher Schwachstellen durchgeführt werden. • Si001 T2.1c: Wichtige sicherheitsrelevante Aktivitäten und Ereignisse werden aufgezeichnet und zeitnah ausgewertet. Dies beinhaltet auch den Datenverkehr zum API-Gateway. • Si004: Die Web Proxy Richtlinie verlangt, dass ein Gateway den Datenverkehr überwacht und steuert.
Das API-Gateway nimmt Anfragen des API-Client entgegen, und gibt sie an den Fachservice weiter (und vice versa). Er parst und übersetzt diese Nachrichten, wenn notwendig, in ein Format, welches der Fachservice, respektive der API-Client versteht. *STRIDE: Verhindern von Tampering der Kommunikation zwischen dem API-Client und dem Fachservice*	Das API-Gateway soll sicherstellen, dass die Anfragen validiert und, wenn notwendig, in ein Format übersetzt werden, dass der Fachservice versteht, bevor sie an die Fachservices weitergeleitet werden. Die Eingabevalidierungen sollten Anti-Injection- und Anti-Korruptions-Datenprüfungen umfassen. Dies beinhaltet nicht nur die Kommunikation vom API-Client zum Fachservice, sondern auch vom Fach-Service zum API-Client. Sofern die Übertragung eine Form von Payload-Verschlüsselung verwendet, wird das API-Gateway, diese Funktion nicht immer wahrnehmen können (insbesondere, wenn die Payload Entschlüsselung nur im Fachservice gemacht werden darf). **Best Practice** • Alle Eingaben sollen validiert werden, um mögliche bösartigen Anfragen zu erkennen und zu filtern. Dazu gehören alle Arten von SQL, System oder anderweitigen Injections. • Ein definiertes Mapping zwischen dem Client- und dem Fachservice soll die Konsistenz der Kommunikation sicherstellen. Abfragen, die nicht den vorgesehenen Schemata entsprechen (z. B. JSON-Schema oder XML-Schema) sollen verworfen werden. • Bei Anfragen in einem falschen Format handelt es sich oft um bösartige Bots, welche nach Schwachstellen scannen. Solche Bots sollten automatisch für eine gewisse Zeit z.B. über eine dynamische Firewall-Regel blockiert werden. Eigene Schwachstellenscans müssen von dieser Regel natürlich ausgenommen werden. • Sofern binäre Eingaben validiert werden müssen, muss dies durch Komponenten geschehen, die eine sichere Speicherverwaltung nutzen (auch Memory-Safety genannt). • Alle Eingaben sollten, wenn möglich auf mögliche Malware geprüft werden. • Ausgaben sollten validiert werden, so dass ein Abfluss möglicher sensibler Daten erkannt und verhindert werden kann.

Funktion	Sicherheitsempfehlungen
	• Eine Zentralisierung der Validierungsregeln im API-Gateway kann deren Verwaltung und Aktualisierung vereinfachen. **Mapping zu Anforderungen** • Si001: Das Prinzip Security by Design (und auch Anforderung A1.1) verlangen, dass einschlägige Sicherheitsempfehlungen, wie z. B. OWASP Top 10 berücksichtigt werden. Das bedeutet auch, dass insbesondre alle Eingaben und Ausgaben validiert werden müssen.
Verhindern, dass ein Ausfall oder die Kompromittierung einer Komponente die gesamte API-Infrastruktur in Mitleidenschaft zieht.	Hierbei ist zu prüfen, ob die Vertraulichkeit und Integrität wichtiger sind (bei einem Ausfall einer anderen Komponente soll das API-Gateway deaktiviert werden) als die Verfügbarkeit (z. B. beim Ausfall des Monetization Systems sollen trotzdem alle (autorisierten) Anfragen durchgelassen werden, und auf die Verrechnung der Anfragen wird verzichtet).

3.3 Komponente: Monetization System

Funktion	Sicherheitsempfehlungen
Die Monetization Komponente verrechnet die Zugriffe und informiert das API-Gateway, wenn ein Zugriff aufgrund fehlender Bonität nicht gestattet ist. *STRIDE: Verhindern von Tampering, Information Disclosure, Denial of Service der Komponente*	Die Monetization-Komponente bewertet das aktuelle Guthaben bzw. Nutzungskontingent eines Benutzers und übermittelt diese Informationen sicher an das API-Gateway. Dieses kann auf Basis dieser Informationen den Zugriff zulassen oder verweigern. Es sollten geeignete Validierungsmechanismen integriert werden, um Missbrauch oder Betrug zu verhindern **Best Practice** • Der Einsatz von TLS 1.2 oder höher für die Verschlüsslung und den Integritätsschutz bei der Übertragung der Daten. • Eine vollständige Protokollierung, so dass jede Veränderung der Guthaben nachvollzogen werden kann. Dies kann auch an die Logging und Reporting Komponente ausgelagert werden. • Verwenden eines sicheren Caches, um die Überprüfungsergebnisse vorübergehend zu speichern. Dies ist wichtig, um die Latenz bei wiederholten Abfragen zu verringern. **Mapping zu Anforderungen** • Si001 I2 und I3: Wenn diese Informationen, respektive die Monetarisierung für die Geschäftsprozesse wichtig ist, muss auch die Integrität und Verfügbarkeit dieser geschäftsrelevanten Informationen dem Schutzbedarf entsprechend geschützt sein.
Die Monetization Komponente kommuniziert mit dem Abrechnungssystem. *STRIDE: Verhindern von Tampering, Information Disclosure der Abrechnung von Zugriffen*	Die Kommunikation zwischen dem Monetization und dem externen System, welches für die Abrechnung verantwortlich ist, sollte verschlüsselt und gegen Integritätsangriffe geschützt sein. Bei der Entwicklung und Beurteilung der API dieses Dritt- Systems, kann man auch die Empfehlungen aus diesem Dokument anwenden. **Best Practices** • Der Einsatz von TLS 1.2 oder höher für die Verschlüsslung und den Integritätsschutz bei der Übertragung der Daten. **Mapping zu Anforderungen** • Si001 I2 und I3: Wenn diese Informationen, respektive die Monetarisierung für die Geschäftsprozesse wichtig ist, muss auch die Integrität und Verfügbarkeit der geschäftsrelevanten Informationen dem Schutzbedarf entsprechend geschützt sein.

3.4 Komponente: Logging, Reporting and Monitoring

Funktion	Sicherheitsempfehlungen
Das Überwachen der Protokolle soll es ermöglichen, Angriffe zu erkennen und rechtzeitig zu reagieren. *STRIDE: Erkennen und Verhindern von Tampering, Information Disclosure, Denial of Service der Kommunikation zwischen den Komponenten*	Die Protokolle und das Überwachungssystem sollen eine Echtzeiterkennung von Angriffen und ungewöhnlichem Verhalten ermöglichen. Sie sollen Korrelationsmechanismen zur Identifizierung von Angriffsmustern beinhalten. Für bekannte Angriffsmuster sollen Alerts konfiguriert sein. Verantwortliche sollen die Analyse der Protokolle regelmässig durchführen und mögliche Angriffe triagieren. Die Umsetzung dieser Anforderungen werden oft von einem Intrusion Detection System (IDS) oder einem Security Information and Event Management (SIEM) System unterstützt, welches ein Teil dieser Komponente sein kann. **Best Practices** • Als Input für die Analyse ist eine Kombination von API-Health-Checks (z. B. mit Hilfe von Prometheus oder bei Netzwerkgeräten mit SNMP) und Protokollen der verschiedenen Komponenten, insbesondere des Netzwerkverkehrs (z. B. mit NetFlow) und der Authentisierungen und Authentisierungsversuche, sinnvoll. • Die Protokolle sollten über alle Komponenten hinweg eine standardisierte Struktur aufweisen (inklusive Quell-IP Adresse, Ziel-IP Adresse, und wenn vorhanden die URL), und einen synchronisierten Zeitstempel beinhalten, um die Aufnahme in SIEM-Systemen zu erleichtern. Diese Systeme können verwendet werden um die verschiedenen Quellen (API-Gateway, API-Clients usw.) zu verknüpfen, und so komplexe Angriffe zu erkennen. • Für bekannte Phänomene sollen Alerts festgelegt damit über ein Benachrichtigungssystem das Betriebsteam alarmiert werden kann. Dazu gehören unter anderem privilegierte und administrative Funktionsausführungen, Änderungen von Zugriffsrechten und Benutzerrollen, Änderungen von Systemkonfigurationen, unautorisierte Zugriffsversuche auf die Protokolle selbst, ein möglicher Datenabfluss (z. B. ungewöhnlich grosse Mengen an Anfragen und Antworten) sowie zu viele Authentisierungen und Authentisierungsversuche. Dies kann bei allen gängigen Systemen (wie Grafana, Splunk, Graylog, DataDog etc.) umgesetzt werden. • Eine regelmässige Prüfung der Integrität der Protokolle, wenn möglich automatisiert, ist empfohlen. So kann schnell erkannt werden, wenn Protokolle verändert wurden oder fehlen. **Mapping zu Anforderungen** • Si001: Anforderung T2.1c erwartet, dass wichtige sicherheitsrelevante Aktivitäten und Ereignisse (mit Zeitangaben) aufgezeichnet und zeitnah ausgewertet werden.
Die Speicherung von Protokollen muss (gegebenenfalls) die Nachvollziehbarkeit der Aktivitäten der API-Clients gegenüber dem API-Gateway garantieren. *STRIDE: Sicherstellen der nicht Abstreitbarkeit (Non Repudiation) der Anfragen von API-Clients.*	Sofern die nicht-Abstreitbarkeit der Anfragen für die Sicherheit der API wichtig ist, sollten die Aufzeichnungen auch sicher und vor unbefugten Änderungen geschützt, aufbewahrt werden. Dies ermöglicht die Rückverfolgbarkeit der Handlungen der API-Kunden. **Best Practice** • Die Protokolle werden auf einem Write-Once-Read-Many (WORM) Medium gespeichert; • Alternativ kann das Fehlen von Protokollen über sequenzielle Nummerierungen festgestellt werden. Veränderungen können auch mithilfe von kryptographischen Checksummen festgestellt werden. Fehlende Protokolle sollten von einem Backup-System wiederhergestellt werden können. **Mapping zu Anforderungen** • Si001 I2.1: Die geschäftsrelevanten Informationen müssen jederzeit ihrem Schutzbedarf entsprechend geschützt sein.

Funktion	Sicherheitsempfehlungen
	• Si002 I3.1: Die Verfügbarkeit von geschäftsrelevanten Informationen muss jederzeit dem Schutzbedarf entsprechend sichergestellt sein.
Protokolle und das Monitoring sollen das Überwachen der Funktionalität der API ermöglichen. *STRIDE: Erkennen und Verhindern von Tampering, Information Disclosoure, Denial of Service der Komponenten*	Das Monitoringsystem soll wichtige Leistungsindikatoren zur Überwachung und Verbesserung der API-Funktionalität umfassen. **Best Practice** • Implementieren von API-Health-Checks, um den Status und die Leistung der Komponenten zu messen. • Automatische Warnschwellen für kritische Indikatoren und ein automatisiertes Benachrichtigungssystems (z. B. über Teams oder E-Mail), ermöglicht es dem Betriebsteam schnell zu reagieren. • Regelmässige End-to-End Tests der API mithilfe von Scripts. **Mapping zu Anforderungen** • Si002 I3.1: Die Verfügbarkeit von geschäftsrelevanten Informationen muss jederzeit dem Schutzbedarf entsprechend sichergestellt sein.
Eine Reporting-Komponente ermöglicht es Berichte über die Verwendung der APIs zu generieren.	Die Komponente soll es ermöglichen, Berichte über die Nutzung der APIs, sowie zu Anzahl von Zugriffen, Latenzzeiten, Fehlern und weiteren Aspekten in einem standardisierten Format zu erstellen. Dies ist keine spezifische operative Sicherheitsanforderung aber notwendig für die Überwachung der Systeme und der Prüfung der Wirksamkeit der Massnahmen, sowie dem Ausweisen der Funktionen bei einem Audit.

3.5 Komponente: API-Endpunkt des Fachservices

Funktion	Sicherheitsempfehlungen
Der API-Endpoint des Fachservices nimmt Anfragen vom API-Gateway entgegen, übersetzt sie und gibt sie weiter an die Anwendung. *STRIDE: Erkennen und Verhindern von Tampering der Kommunikation und in der Folge des Fachservices*	Der API-Endpoint des Fachservices soll sicherstellen, dass die Anfragen validiert werden. Die Eingabevalidierungen soll Anti-Injection- und Anti-Korruptions-Datenprüfungen umfassen. Auch wenn der API-Gateway bereits solche Validierungen vorgenommen hat, sollte der Fachservice, im Sinne von Zero-Trust, selbst auch Prüfungen durchführen. **Best Practice** • Alle Eingaben sollen validiert werden, um mögliche bösartige Anfragen zu erkennen und zu filtern. Dazu gehören alle Arten von SQL, System oder anderweitigen Injections. Dies sollte immer gemacht werden, auch wenn das API-Gateway oder eine Web Application Firewall (WAF) dies bereits tun. • Sofern binäre Eingaben validiert werden müssen, soll dies durch Komponenten geschehen, die eine sichere Speicherverwaltung nutzen (auch Memory Safety genannt). • Alle Eingaben sollten, wenn möglich, auch gescannt werden, um zu erkennen, ob es sich um Malware handeln könnte. **Mapping zu Anforderungen** • Si001: Das Prinzip Security by Design (und auch Anforderung A1.1) verlangen, dass einschlägige Sicherheitsempfehlungen, wie z. B. OWASP Top 10 berücksichtigt werden. Das bedeutet auch, dass insbesondere alle Eingaben und Ausgaben validiert werden müssen.[13]

[13] Zur Fragen der Implementierung der OWASP Best Practices gibt es auch das API Security Project: https://owasp.org/www-project-api-security/

Funktion	Sicherheitsempfehlungen
Der API-Endpoint des Fachservices gibt Antworten und sendet diese über das API-Gateway zurück zum API-Client. *STRIDE: Erkennen und Verhinden von Tampering des API-Gateways, Clients und anderen Komponenten*	Der API-Endpoint des Fachservices soll sicherstellen, dass die Antworten validiert werden. Die Ausgabevalidierung sollte insbesondere prüfen, ob sensible Daten (wie API-Keys) in der Antwort sind (dies kann z. B. aufgrund von erfolgreichen Server Side Request Forgery (SSRF) Angriffen geschehen), oder Scripts, welche beispielsweise von einem Reflected-Cross-Site-Scripting Angriff stammen (XSS). Die Kommunikation zwischen dem Fachservice und den anderen Komponenten sollte verschlüsselt und gegen Integritätsangriffe geschützt sein. **Best Practices** • Ausgabevalidierung mit Filterfunktionen oder Regular Expressions. Sofern für die Validierung Regular Expressions verwendet werden, muss unbedingt auf die Vermeidung von Evil Regex geachtet werden um ReDoS-Angriffe (Regular expression Denial of Service) zu verhindern. • Der Einsatz von TLS 1.2 oder höher für die Verschlüsslung und den Integritätsschutz bei der Übertragung der Daten. **Mapping zu Anforderungen** • Si001: Das Prinzip Security by Design (und auch Anforderung A1.1) verlangen, dass einschlägige Sicherheitsempfehlungen, wie z. B. OWASP Top 10 berücksichtigt werden. Das bedeutet auch, dass alle Ausgaben validiert und gegebenenfalls maskiert werden müssen. • Si001 I2.1: Die geschäftsrelevanten Informationen müssen jederzeit ihrem Schutzbedarf entsprechend geschützt sein
Der Fachservice prüft die Autorisierung des anfragenden API-Clients *STRIDE: Verhinden von Elevation of Privilege auf dem Fachservice*	Der Fachservice soll sicherstellen, dass jedem API-Client die richtigen Rollen zugewiesen werden. Sofern das API-Gateway auch eine Zugriffsteuerung vorgenommen hat, muss die Integrität der Zuweisung geprüft werden. Wenn nicht, muss die Verknüpfung der Rollen mit dem Client im Fachservice sichergestellt werden. **Mapping zu Anforderungen** • Si001: Das Prinzip Least Privilege, respektive die Sicherheitsanforderung T5.2 verlangen, dass Autorisierungsinformationen vorhanden sind und eine Autorisierung vorgenommen wird. • Si001 Anforderung Z3: Sofern die Rollenzuteilung im Fachservice gemacht werden, ist dieser auch ein Policy Enforcement Point (PEP).

3.6 Komponente: Registry / CI/CD Pipeline

Funktion	Sicherheitsempfehlungen
Die Registry liefert die API-Builds und Konfigurationen für die API-Instanzen (API-Gateway, Fachservice) *STRIDE: Verhindern von Tampering und Information Disclosure der Images in der Registry sowie nicht Abstreitbarkeit (Non Repudiation) von Veränderungen an den Images*	Die Registry soll gesichert sein, um die Integrität und Authentizität der API-Konfigurationen und -Builds vor ihrer Verwendung zu gewährleisten. **Best Practice** • Die CI/CD-Pipeline sollte verhindern, dass ungetesteter Quellcode eingeschleust werden kann. • Es sollte nicht möglich sein, Applikationen in der Registry zu publizieren, die nicht in einer geprüften Pipeline kompiliert wurden. Für die Verifikation können z. B. kryptografische Signaturen eingesetzt werden. • Bei Einbindung externer Werkzeuge, wie zum Beispiel von Kompilierservern, soll sichergestellt sein, dass die Pipeline selbst dadurch nicht kompromittiert werden kann und dass kein sensibler Quellcode abfliesst.

Funktion	Sicherheitsempfehlungen
	Mapping zu Anforderungen • Si001 A1.2e, respektive S4.1: Die Integrität der Software muss jederzeit sichergestellt sein. • Si001 A1.2b: Der Zugriff auf Repositories muss klar geregelt und nachvollziehbar kontrolliert werden; hierbei ist auch Anforderung T5 relevant, wonach diese Regelung anhand dem Least-Privilege-Prinzip erfolgen muss.
Die CI/CD Pipeline ermöglicht das automatisierte Prüfen der API-Builds auf Schwachstellen und Fehler. *STRIDE: Verhinderung von Tampering der API-Endpoints*	Die CI/CD-Pipeline sollte Sicherheitsanalyse-Tools enthalten, um automatisiert Schwachstellen und Verwundbarkeiten zu erkennen und die Konformität der Builds mit den Sicherheitsrichtlinien sicherzustellen. Die Prüfung soll bei der Entwicklung erstmals und danach regelmässig durchgeführt werden. Bei erkannten Schwachstellen sollte keine Installation auf die Produktion stattfinden. Auch die CI/CD-Pipeline selbst sollte regelmässig geprüft und auf dem aktuellen Stand gehalten werden. **Best Practice** • Verwenden von Tools in der Pipeline für die syntaktische Fehlerprüfung. • Verwendung von Tools für die statische Code-Analyse und Erkennung von bekannten semantischen Fehlern und sogenannten «Code Smells» (Code der verdächtig unsicher aussieht). • Ergänzend zur statischen Analyse (SAST) wird empfohlen, dynamische Analysen (DAST) in Form von automatisierten Scans auf Vorproduktionssystemen sowie sicherheitsorientierten Integrationstests durchzuführen, um das Verhalten der API in realistischen Umgebungen zu validieren. • Dependency Checker Tools helfen bei der Prüfung, ob verwendete Bibliotheken und Abhängigkeiten bekannte Sicherheitslücken aufweisen. **Mapping zu Anforderungen** • Si001 A1.2c: Die Build-Prozesse müssen überwacht werden und Änderungen in der Build Pipline dürfen nur kontrolliert erfolgen. • Si001 A1.2d: Die Software muss regelmässig getestet werden, sowie T4, wonach Schwachstellen und Verwundbarkeiten, wenn möglich, automatisiert erkannt werden müssen.

3.7 Komponente: IAM-System

Funktion	Sicherheitsempfehlungen
Das Identity- und Access-Management-System registriert Benutzer und API-Client und assoziiert sie mit API-Zugriffsrechten. *STRIDE: Verhindern von* Information *Disclosure in den Fachservices*	Das IAM soll eine sichere Registrierung von API-Clients ermöglichen. Es verwaltet dabei meistens auch die Berechtigungen, auf welchen API-Fachservice der Client zugreifen darf. Zu IAM gelten beim Bund die Anforderungen der Standarddienste.[14] **Mapping zu Anforderungen** • Si001 T5: Zugriffsrechte müssen dem Least-Privilege-Prinzip folgen. • Si001 S3: Dienstkonti müssen sicher verwaltet werden

[14] In der Bundesverwaltung beschreibt W008 - Weisungen zur Steuerung und Führung der Standarddienste gemäss VDTI, Anhang 4, welche IAM Systeme eingesetzt werden dürfen. Dies gilt auch für die API- und Prozess-Authentisierung

Funktion	Sicherheitsempfehlungen
Die Identity- und Access-Management-Komponente authentisiert API-Clients. *STRIDE: Verhindern von Tampering, Spoofing der API-Clients, respektive Elevation of Privilege des API-Clients gegenüber dem Fachservice*	Es gelten beim Bund die Anforderungen der Standarddienste. Mögliche Methoden zur indirekten Authentisierung sind in Kapitel 3.1 beschrieben.
Die Identity- und Access-Management-Komponente bestätigt die Authentizität eines Authentisierungstokens. *STRIDE: Verhindern von Spoofing und Tampering der Authentisierungs-tokens*	Die IAM-Komponente soll die Authentifizierungstoken validieren können und deren Authentizität, respektive deren Signatur, zeitliche Gültigkeit und Verwendungszweck kontrollieren und bestätigen. **Mapping zu Anforderungen** • Alle Anforderungen bezüglich Authentisierung benötigen die Möglichkeit die Korrektheit der Authentisierung zu validieren.

3.8 Komponente: API-Verzeichnisdienst

Funktion	Sicherheitsempfehlungen
Der API-Verzeichnisdienst ermöglicht es den API-Clients für bestehende APIs, die Endpoints und Dokumentation zu finden. *STRIDE: Verhindern von Denial of Service / Spoofing des API-Verzeichnisdienstes*	Das API-Verzeichnis soll eine sichere Suche nach API-Endpoints ermöglichen, das heisst unter anderem, es soll ein Spoofing des Verzeichnisdienstes zu verhindern versuchen, damit API-Clients (z. B. via Phishing) nicht an falsche API-Endpoints verwiesen werden. Zudem sollte der Verzeichnisdienst redundant ausgestaltet sein, damit ein Denial-of-Service-Angriff auf diesen die Verwendung von APIs nicht beeinträchtigt. Wenn notwendig, darf der Verzeichnisdienst Zugriff auf die Informationen darin nur nach vorangehender Authentisierung erlauben. Für das Publizieren von Informationen zu APIs, gibt es in der Bundesverwaltung auch den Metadatenkatalog der Interoperabilitätsplattform[15]. **Best Practices** • Das API-Verzeichnis klassifiziert APIs nach den erforderlichen Schutzstufen. **Mapping zu Anforderungen** • Si001 I2.1: Die geschäftsrelevanten Informationen müssen jederzeit ihrem Schutzbedarf entsprechend geschützt sein. • Si002 I3.1: Die Verfügbarkeit von geschäftsrelevanten Informationen muss jederzeit dem Schutzbedarf entsprechend sichergestellt sein.
Der API-Verzeichnisdienst ermöglicht es APIs ihre Endpoints zu registrieren und revozieren. *STRIDE: Verhindern von Tampering des Verzeichnisdiensts*	Das API-Verzeichnis soll eine sichere Verwaltung der Endpoints gewährleisten, einschliesslich ihrer Registrierung, Änderung und Löschung. Diese Registrierung sollte authentisiert erfolgen. **Mapping zu Anforderungen** • Si001 T5: Zugriffsrechte müssen dem Least-Privilege-Prinzip folgen.

[15] https://www.i14y.admin.ch/de/catalog/dataservices

3.9 Gesamtsystem: Sicherheitsüberwachung und IT-Sicherheit

Funktion	Sicherheitsempfehlungen
Bekannte Schwachstellen oder Fehlkonfigurationen werden erkannt. *STRIDE: Verhindern von allen Bedrohungen auf die API-System-landschaft*	Für die jeweiligen Komponenten wurden spezifische Empfehlungen gemacht. Damit aber das Gesamtbild nicht aus den Augen verloren geht, sollen die jeweiligen Massnahmen im Gesamtsystem geprüft werden. Zu diesem Zweck können die Sicherheitsannahmen auch mithilfe von Tests wie Simulationen, Lastentests, Latenztests, Backup-Restore Tests, Failover Tests, Penetration Tests, Schwachstellenscans oder in einem Bug Bounty Programm geprüft werden. Die Periodizität dieser Tests und Simulationen hängt von der Art und dem Schutzbedarf der API ab und sollte vom Leistungsbezüger in Absprache mit dem Leistungserbringer festgelegt werden. Fehlermeldungen, welche an den API-Client zurückgegeben werden, sollten zudem über alle Komponenten hinweg standardisiert sein. Dies auch damit keine System-internen Information über Fehlermeldungen preisgegeben werden. Wichtig ist hier auch, dass APIs für administrative Zugriffe nur unter den Voraussetzungen von Anforderung T8.3 des IT-Grundschutzes erlaubt sind. Dabei handelt es sich um API-Endpoints, welche administrative Funktionen anbieten (dazu gehört z. B. das Anpassen von Einstellungen der Anwendung oder die Konfiguration von Rollen und Benutzerrechten).