Standardkonforme Integration quelloffener Big Data-Lösungen
in existierende Netzleitsysteme

Sicherheitsanforderungen an netzleitsystemnahe Software für den Energienetzbetrieb und Empfehlungen für Security-by-Design und Sicherheitszertifizierung von Open Source-Lösungen am Beispiel von Open Source in openKONSEQUENZ

Technical Report

Autoren
André Göring (OFFIS), Tilo Mentler (IMIS)

8. Dezember 2020
Revision 1.0.0

Dokumentenhistorie

Version	Date	Author(s)	Description/Comments
0.0.1	27.03.2020	Andre Göring	Initiale Dokumentenstruktur eingefügt, Kapitel 1 Einleitung, Kapitel 2 Entwurfsphasensicherheit, Kapitel 3 Implementierungssicherheit verfasst.
0.0.2	30.03.2020	Tilo Mentler	Kapitel 4 zu Benutzungsschnittstellen verfasst.
0.0.3	09.05.2020	Andre Göring	Kapitel 5 zu Interoperabilitätsnachweis, Kapitel 6 zu Zertifizierung und Kapitel 7 Zusammenfassung verfasst.
1.0.0	08.12.2020	Andre Göring	Überarbeitung für Veröffentlichung

Abstract

Die Software in netzleitsystemnahen Umfeld beeinflusst kritische Infrastruktur und sollte deshalb unter sicherheitstechnischen Aspekten gestärkt sein. Die Berücksichtigung von Sicherheit bereits während der Entwurfsphase einer Software wird als Security-By-Design bezeichnet. Es ist dabei davon auszugehen, dass Angreifer mit hohem Budgeteinsatz (zeitlich, personell, monetär) eine Möglichkeit finden werden, ein System zu kompromittieren. Durch die Anwendung bekannter Sicherheitstaktiken und Sicherheitsmuster, soll dies erschwert werden. Die Hürden für Angriffe sollen so hoch gelegt werden, dass sich der Aufwand für die meisten Angreifer nicht lohnt und diese nach kurzem Versuch davon ablassen. Ein Sicherheitskonzept nur zur Entwurfsphase ist jedoch aus Sicht der am vorliegenden Bericht beteiligten Personen nicht ausreichend für die Gewährleistung der Sicherheit einer Software. Deshalb werden über Security-by-Design-Prinzipien während der Entwurfsphase von Software hinaus weitere Themen behandelt.

Während der Implementierung von Software können unabhängig vom umgesetzten Sicherheitskonzept der Entwurfsphase Angriffsflächen entstehen, die berücksichtigt werden müssen und durch bekannte und einfache Muster gering gehalten werden können. Auch bei der Verwendung von Software können sich durch Fehlbedienungen kritische Situationen ergeben, denen durch eine geeignete Darstellung vorzubeugen ist. Zudem sollte die Sicherheit von Open Source Software durch verschiedene Parteien nachvollziehbar bewertet werden können und die Einhaltung von Sicherheitsvorgaben ein Teil einer Zertifizierung von Software sein. Des Weiteren muss für Schnittstellen sichergestellt werden, dass ein Datenaustausch sowohl syntaktisch als auch semantisch korrekt durchgeführt wird, damit sich Software interoperabel in bestehende Systemwelten integrieren lassen.

Im vorliegenden Bericht werden Empfehlungen zur Sicherheit und ein Vorgehen zur Bewertung der Sicherheit von Software sowie die Entwicklung von Zertifikaten für Open Source Software im Umfeld von Netzleitsystemen vorgestellt.

Der Bericht greift die Sicherheitsanforderungen aus der IEC 62351 Reihe, dem BSI Grundschutz (BSI, 2020) und dem BDEW Whitepaper (BDEW, 2018) auf und vertieft diese Anforderungen für die Entwicklungsphasen des Software-Lebenszyklus (Anforderungsspezifikation, Entwurf, Entwicklung, Testen und Update/Patch-Management) bis zur Bereitstellung der Softwarequellen basierend auf Erfahrungen aus NetzDatenStrom und den Entwicklungsprojekten von openKONSEQUENZ. Der sichere IT-Betrieb sowie die Netzwerksicherheit obliegen den die Software einsetzenden/bereitstellenden Unternehmen. Sie sind in den oben genannten Quellen sowie der ISO 27000 und folgende detailliert aufbereitet und sind deswegen hier nicht weiter vertieft.

Inhalt

1 Problemstellung 1
2 Entwurfsphasensicherheit 2
2.1 IEC 62351 2
2.2 BSI Grundschutz 10
2.3 BDEW Whitepaper 14
2.4 Zusammenfassung Entwurfsphasensicherheit 15
3 Implementierungssicherheit 19
3.1 BSI Grundschutz 19
3.2 BDEW Whitepaper 21
3.3 Konsortiale Open Source Software-Entwicklung von openKONSEQUENZ 21
3.4 Zusammenfassung Implementierungssicherheit 26
4 Benutzungsschnittstellenvorgaben 30
5 Interoperabilitätsnachweis 33
6 Zertifizierung 36
6.1 Vorgehensweise zur Zertifizierung 36
6.2 Glaubwürdigkeit von Zertifikaten 37
6.3 Konsortiale Zertifizierung konsortialer Open Source Software 38
7 Zusammenfassung und Ausblick 41
8 Literaturverzeichnis 42

1 Problemstellung

Die Software in netzleitsystemnahen Umfeld beeinflusst kritische Infrastruktur und sollte deshalb unter sicherheitstechnischen Aspekten gestärkt sein. Die Berücksichtigung von Sicherheit bereits während der Entwurfsphase einer Software wird als Security-by-Design bezeichnet. Es ist dabei davon auszugehen, dass Angreifer mit hohem Budgeteinsatz (zeitlich, personell, monetär) eine Möglichkeit finden werden, ein System zu kompromittieren. Durch die Anwendung bekannter Sicherheitstaktiken und Sicherheitsmuster, soll dies erschwert werden. Die Hürden für Angriffe sollen so hoch gelegt werden, dass sich der Aufwand für die meisten Angreifer nicht lohnt und diese nach kurzem Versuch davon ablassen. Security-by-Design hebt sich hierbei von der bislang oft üblichen sogenannten Security-by-Obscurity ab, in welcher die Sicherheit aus dem Grund erwartet wird, dass ein geheim gehaltener Sicherheitsmechanismus Angreifer abschreckt. Bei einem geheimen Sicherheitskonzept ist es jedoch schwierig nachzuvollziehen, wie leicht ein System kompromittiert werden kann oder sogar bereits wurde. Darüber hinaus lässt sich die Sicherheit nur schwierig verstärken, da bisherige Maßnahmen nicht bekannt sind und gegebenenfalls sogar durch hinzugefügte Maßnahmen geschwächt werden. Das National Institute of Standards and Technology (NIST) sowie die IEC empfiehlt ein offenes Sicherheitskonzept (Scarfone, Jansen, & Tracey, 2008) (Cleveland, 2012). Ein Sicherheitskonzept nur zur Entwurfsphase ist jedoch nicht ausreichend für die Gewährleistung der Sicherheit einer Software. Deshalb werden über Security-by-Design-Prinzipien während der Entwurfsphase von Software hinaus auch weitere Themen behandelt.

Während der Implementierung von Software können unabhängig vom umgesetzten Sicherheitskonzept der Entwurfsphase Angriffsflächen entstehen, die berücksichtigt werden müssen und durch bekannte und einfache Muster gering gehalten werden können. Auch bei der Verwendung von Software können sich durch Fehlbedienungen kritische Situationen ergeben, denen durch eine geeignete Darstellung vorzubeugen ist. Zudem sollte die Sicherheit insbesondere von Open Source Software durch verschiedene Parteien nachvollziehbar bewertet werden können und die Einhaltung von Sicherheitsvorgaben ein Teil einer Zertifizierung von Software sein. Des Weiteren muss für Schnittstellen sichergestellt werden, dass ein Datenaustausch sowohl syntaktisch als auch semantisch korrekt durchgeführt wird, damit sich Software interoperabel in bestehende Systemwelten integrieren lassen.

Im Fokus des Berichts liegt die Sicherstellung von Sicherheit bis zur Auslieferung von Software an mögliche Kunden. Dazu greift der Bericht die Sicherheitsanforderungen aus der IEC 62351 Reihe, dem BSI Grundschutz (BSI, 2020) und dem BDEW Whitepaper (BDEW, 2018) auf und vertieft diese Anforderungen für die Entwicklungsphasen des Software-Lebenszyklus (Anforderungsspezifikation, Entwurf, Entwicklung, Testen und Update/Patch-Management) bis zur Bereitstellung der Softwarequellen basierend auf Erfahrungen aus NetzDatenStrom und den Entwicklungsprojekten von openKONSEQUENZ. Der Betrieb von Software unterliegt natürlich auch speziell im Umfeld kritischer Systeme Sicherheitsvorgaben welche beispielsweise auch im BSI Grundschutz-Kompendium (BSI, 2020), BDEW Whitepaper (BDEW, 2018) detailliert beschrieben sind oder dem Informationssicherheitsmangement (ISMS) ISO 27000 und folgende unterliegen. Der sichere Betrieb ist jedoch nicht Gegenstand des vorliegenden Berichts.

Im vorliegenden Bericht werden Empfehlungen zur Sicherheit und ein Vorgehen zur Bewertung der Sicherheit von Open Source Software-Entwicklungen sowie die Entwicklung von Zertifikaten für solche Software für das Umfeld von Netzleitsystemen vorgestellt. Die hardwarenahe Software/Firmware von Betriebsmitteln (industriellen Kontrollsystemen, Aktoren, Sensoren,…) liegt nicht im Fokus.

Dazu werden in Kapitel 2 die wesentlichen bestehenden Vorgaben bezüglich der Sicherheit für die Entwurfsphase vorgestellt. In Kapitel 3 werden Vorgaben aufgestellt hinsichtlich der Sicherheit während der Implementierung. Das Kapitel 4 enthält Vorgaben für die Erstellung von Benutzungsschnittstellen. In Kapitel 5 wird ein bestehendes Konzept des sogenannten Connectathon für Schnittstelleninteroperabilität vorgestellt und erweitert, um die Funktionalität von Schnittstellen und Software sicherzustellen und Interoperabilität nachzuweisen. In Kapitel 6 wird ein Verfahren zur Sicherstellung von Sicherheit und Funktionalität für eine gesamtheitliche Zertifizierung netzleitsystemnaher Software bis zur Auslieferung erarbeitet. Im abschließenden Kapitel 7 erfolgen Zusammenfassung und Ausblick.

2 Entwurfsphasensicherheit

Bei der Entwicklung von Software ist es sinnvoll, Sicherheit bereits während der Entwurfsphase zu berücksichtigen. Das heißt, dass bereits während der Planung der eigentlichen Softwareentwicklung das sichere Zusammenspiel von Benutzer und Software, von verschiedener Software sowie von Software und Hardware bedacht wird. Dies verhindert, neben der wesentlichen Gefahr unsichere Software einzusetzen, dass bereits umgesetzte funktionale Komponenten aufgrund von später berücksichtigten Sicherheitsaspekten verändert oder gar neu programmiert werden müssen und der bis dahin geleistete Entwicklungsaufwand im schlimmsten Fall nicht verwendet werden kann. Für Energiesystemstrukturen gibt die International Electrotechnical Commission (IEC) Technical Commity (TC) 57 Working Group (WG) 15 die Normenreihe „IEC 62351 Security Standards for the Power System Information Infrastructure“ heraus. Auch das Bundesamt für Sicherheit in der Informationstechnik (BSI) beschreibt den Grundschutz von Software insbesondere auch in kritischer Infrastruktur. Des Weiteren gibt auch der Bundesverband der Energie- und Wasserwirtschaft (BDEW) ein Whitepaper bezüglich der für Sicherheit zu berücksichtigenden Anforderungen heraus. Die genannten Normen, Pflichten und Empfehlungen werden im Folgenden vorgestellt. Sie können als Grundlage für einen sicheren Entwurf von Software insbesondere der kritischen Infrastruktur Energiesysteme herangezogen werden.

2.1 *IEC 62351*

Die Normenreihe IEC 62351 wurde durch die IEC TC57 WG15 mit dem Auftrag erstellt, Sicherheitsstandards für die durch die IEC TC57 definierten Kommunikationsprotokolle zu entwickeln sowie Standards / Berichte zu End-zu-End Sicherheitsfragen zu entwickeln. Der Bericht „IEC 62351 Security Standards for the Power System Information Infrastructure“ (Cleveland, 2012) gibt eine Übersicht über die Normenreihe und wird in diesem Kapitel zusammengefasst. Energienetz sowie Informations- und Kommunikationtechnologie (IKT) verschmelzen im Smart Grid zu einem sogenannten Cyber-Physical System mit starken Abhängigkeiten untereinander: Das IKT-Netz wird für die (effiziente und sichere) Steuerung des Energienetzes benötigt. Das Energienetz verbindet nicht nur Erzeuger mit Verbrauchern sondern versorgt auch die IKT mit der nötigen Energie. Physische Sicherheitskomponenten des Energienetzes werden nicht im IEC 62351 normiert. Der Fokus liegt auf die Übertragung von IKT-Sicherheit auf das Cyber-Physikalische System. Die Beeinträchtigung von IKT beeinflusst dementsprechend nicht nur IKT-Systeme, sondern kann auch direkte physische Auswirkungen haben, beziehungsweise physische Schutzschaltungen oder physische Schutzkaskaden auslösen. Zudem können IKT Schutzmaßnahmen negativen Einfluss auf das Gesamtsystem haben. Der IEC Bericht beschreibt Sicherheitsgefährdungen zwischen unbeabsichtigten Zwischenfällen (wie Sicherheitsausfälle, Ausfälle der Ausrüstung, Unachtsamkeit, Naturkatastrophen und absichtlich herbeigeführten Bedrohungen wie verärgerte Mitarbeiter, Industriespionage, Vandalismus/Diebstahl, Hacker, Terrorismus) und stellt diesen fünf Schichten von Sicherheitszielen (Abschreckung und Verzögerung, Aufdeckung von Angriffen, Bewertung von Angriffen, Kommunikation und Benachrichtigung über Angriffe sowie die Reaktion auf Angriffe) gegenüber. In der Energiedomäne gibt es hier besondere Herausforderungen (wie die Interaktion mit mehr und mehr Stakeholdern; große, zum Teil medienübergreifende Verbindungskomplexität; Einbindung internetbasierter Technologien; Einbindung bestehender (Alt-)Systeme (legacy systems); Erhöhte Attraktivität für Cyber-Angriffe), die aufgrund des geschäftlichen und technischen Umfelds entstehen.

Im Folgenden werden die Sicherheitsanforderungen, Attacken sowie Gegenmaßnahmen gegenübergestellt. Anschließend werden die wesentlichen Maßnahmen der Sicherheitsarchitektur, des Zertifikatsmanagements, der Zugriffskontrolle, der Daten- und Kommunikationssicherheit sowie des Netzwerk- und Systemmanagements vorgestellt.

2.1.1 Sicherheitsanforderungen, Gefahren, Angriffe und Gegenmaßnahmen

Die IEC beschreibt in ihrem Bericht (Cleveland, 2012) vier Grundlegende Sicherheitsanforderungen:

- Vertraulichkeit (Confidentiality): Die Verhinderung unautorisierten Zugriffs auf Information.
- Integrität (Integrity): Die Verhinderung unautorisierter Modifikation (inklusive des Löschens) von Information.

- Verfügbarkeit (Availability): Die Verhinderung von Dienstausfällen und Sicherstellung des autorisierten Zugriffs auf Information.
- Rechenschaftsplicht (Non-Repudiation/Accountability): Verhinderung der Abstreitbarkeit einer durchgeführten Aktion oder der Behauptung einer nicht durchgeführten Aktion.

Diesen Sicherheitsanforderungen stehen Gefahren mit verschiedenen Angriffsmöglichkeiten gegenüber, die in Abbildung 1 dargestellt werden.

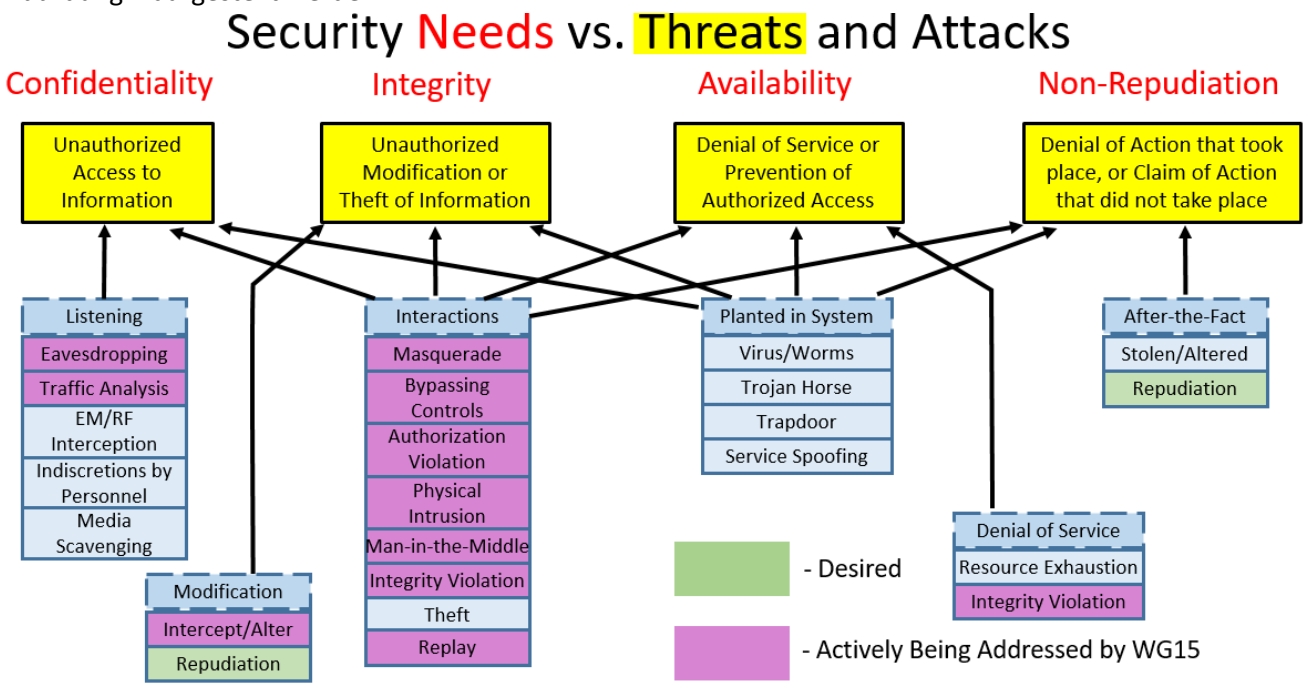


Abbildung 1: Sicherheitsanforderungen, Gefährdungen und mögliche Angriffe, welche durch das IEC TC57 WG15 adressiert werden (nach (Cleveland, 2012)).

Gegenmaßnahmen, wie sie in Abbildung 2 in lila dargestellt werden, bestehen ebenfalls aus einem Zusammenspiel von Technologien und Richtlinien. Jedoch wird nicht jede Sicherheitsmaßnahme für jedes System benötigt (Cleveland, 2012). In einem ersten Schritt ist zu identifizieren, welche Gegenmaßnahme zu welchen Anforderungen beiträgt. Hinsichtlich des dann anfallenden Sicherheitsmanagements sind einige Maßnahmen in der Abbildung in blau exemplarisch eingefügt.

Aufgrund der großen Vielzahl an Kommunikationsmethoden und Performanz Charakteristika, sowie aufgrund der Tatsache, dass eine einzelne Sicherheitsmaßnahme nicht allen Arten von Gefahren entgegenwirken kann, wird im IEC Bericht erwartet, dass mehrere Schichten von Sicherheitsmaßnahmen implementiert werden. Dies wird auch nochmal dadurch hervorgehoben, dass Sicherheit ein Querschnittsthema über alle Ebenen der Kommunikation im ISO/OSI Schichtenmodell oder im GWAC Stack darstellt und eine Ende-zu-Ende Sicherheit adressiert werden muss. Aus diesem Grund soll Cyber-Security als ein an die jeweilige Aufgabe angepasstes Maßnahmenbündel oder Profil verschiedener Sicherheitstechnologien und -prozeduren von Richtlinien verstanden werden.

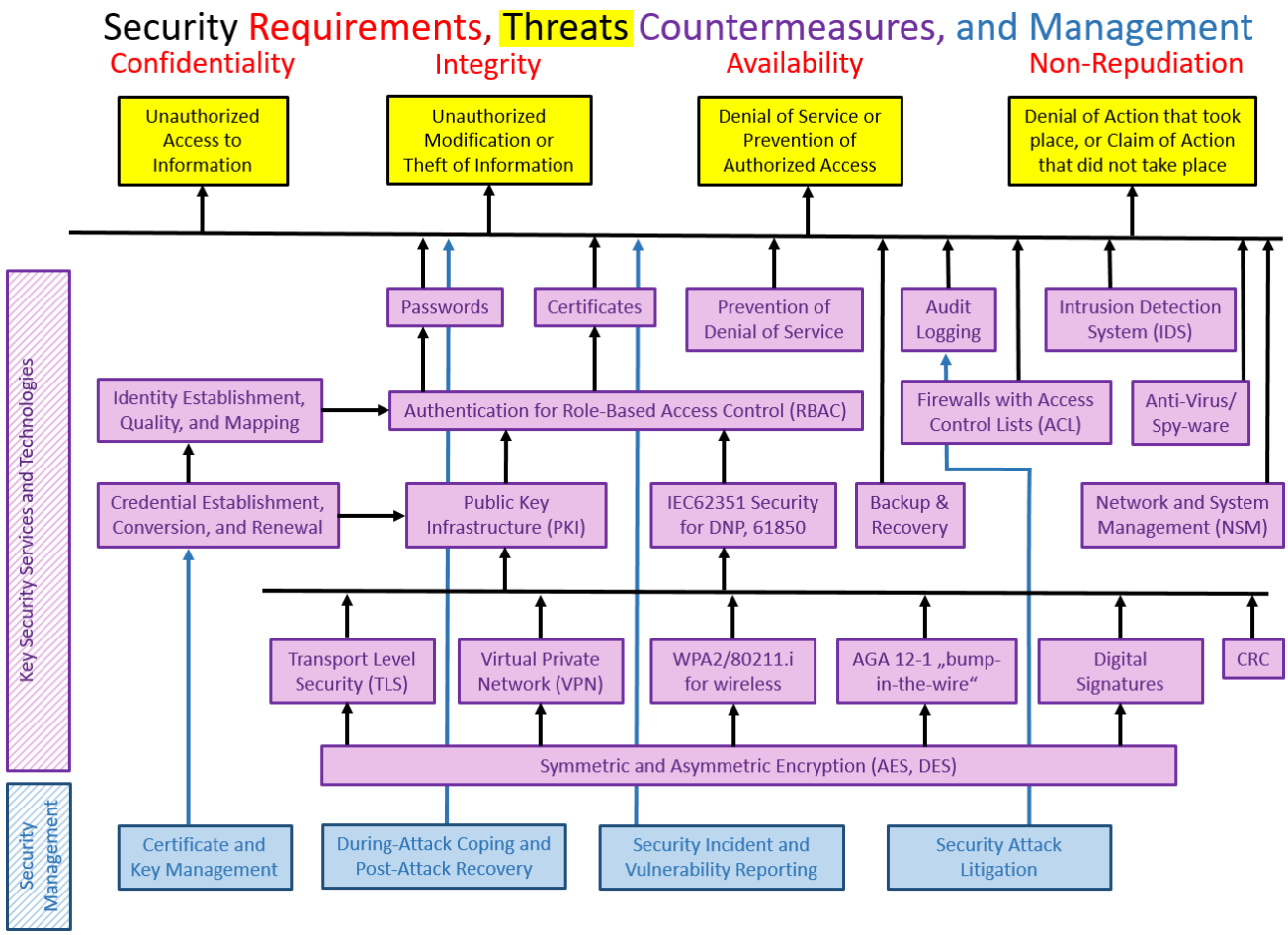


Abbildung 2: Sicherheitsübersicht: Sicherheitsanforderungen, -gefährdungen, -management-Beispiele und Gegenmaßnahmen (nach (Cleveland, 2012))

2.1.2 Systemweite Sicherheitsarchitektur

Die Norm IEC 62351-10 (IEC, IEC/TR 62351-10: Power systems managment and associated information exchange - Data and communications security - Part 10: Security architecture guidelines, 2012) bietet für die Gestaltung einer sicheren Netzwerkarchitektur Lösungen zur Realisierung an. Die Bereiche Netzwerksegmentierung, strenge Authentifizierung, Zugangskontrolle, Daten- und Kommunikationssicherheit sowie Sicherheitsüberwachung und vorbeugende Maßnahmen sind besonders hervorzuheben (Harner, 2019). Die Netzwerksegmentierung wird im Folgenden gemäß (Harner, 2019) vorgestellt.

Die Netzwerksegmentierung stellt eine Unterteilung eines Unternehmensnetzwerkes in unterschiedliche Bereiche dar, wobei Zugriffe zwischen unterschiedlichen Bereichen reglementiert oder unterbunden werden.

Die Trennung der unterschiedlichen Bereiche erfolgt durch Firewalls, filternde Router oder Gateways, welche ausschließlich die für die Kommunikation als nötig spezifizierten Kommunikationsprotokolle und Verbindungen erlauben. Es kann sowohl eine horizontale Segmentierung nach Zonen verschiedener Funktion und verschiedenem Schutzbedarf erfolgen, als auch eine vertikale Segmentierung nach den zugrundeliegenden Energienetzzonen / Standorten. Mittels Einrichtung einer demilitarisierten Zone (DMZ – ein eigenständiges Subnetz zur Trennung von Netzwerken) kann ein direkter Fernzugriff verhindert werden. In Abbildung 3 wird eine segmentierte Systemarchitektur mit DMZ für den Fernzugriff dargestellt.

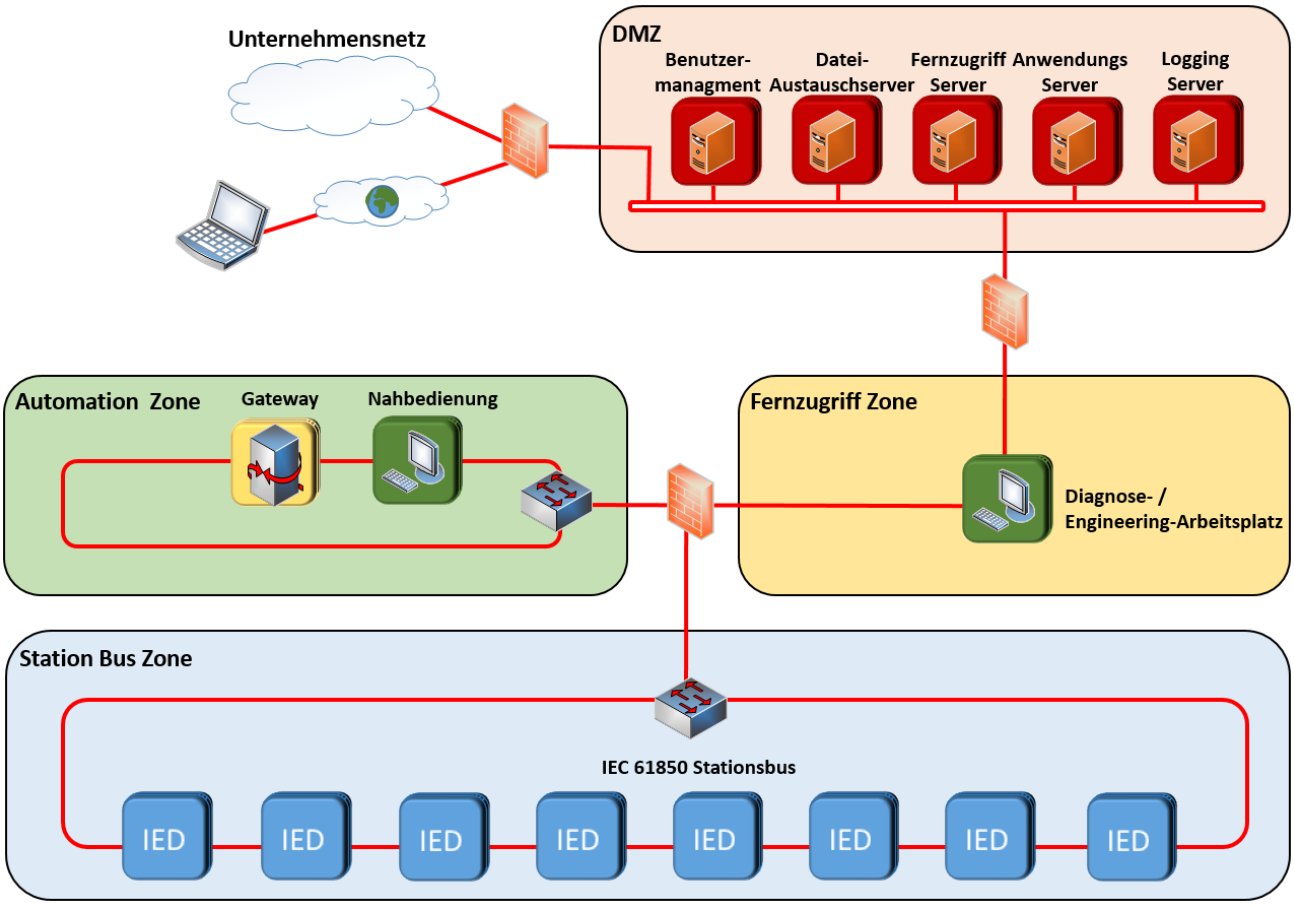


Abbildung 3: Sichere Systemarchitektur für den Fernzugriff (nach (Harner, 2019)).

2.1.3 Authentifizierung

Die IEC 62351-9 (IEC, IEC 62351-9: Power systems management and associated information exchange - Data and communications security - Part 9: Cyber security key management for power system equipment, 2017) behandelt das Zertifikats-/Schlüsselmanagement. Zertifikate beziehungsweise Schlüssel dienen als Grundlage für die Identitätsverifikation sowohl von Geräten als auch von Software oder Menschen sowie für die Verschlüsselung von Nachrichten.

Bei der Verschlüsselung von Daten wird zwischen symmetrischer und asymmetrischer Verschlüsselung unterschieden. Bei der symmetrischen Verschlüsselung Nutzen Empfänger und Sender den gleichen, nur ihnen bekannte Verschlüsselungsschlüssel zur En- und Decodierung der Nachrichten. Bei der asymmetrischen Verschlüsselung besteht der Schlüssel aus einem öffentlich bekannten (public) und einem korrespondierendem privaten (private) Schlüsselbestandteil, welcher jeweils nur einer Partei bekannt ist und zwingend geheim gehalten werden muss. Je mehr Komponenten miteinander kommunizieren, desto aufwändiger wird die Konfiguration und Pflege der Schlüssel insbesondere für die symmetrische Verschlüsselung (siehe Abbildung 4) (Harner, 2019).

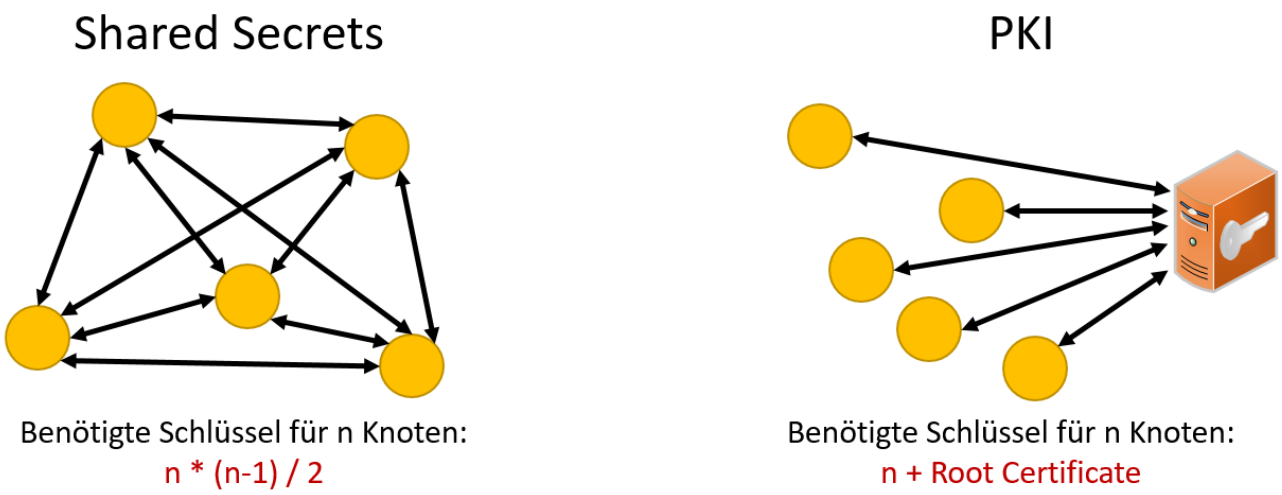
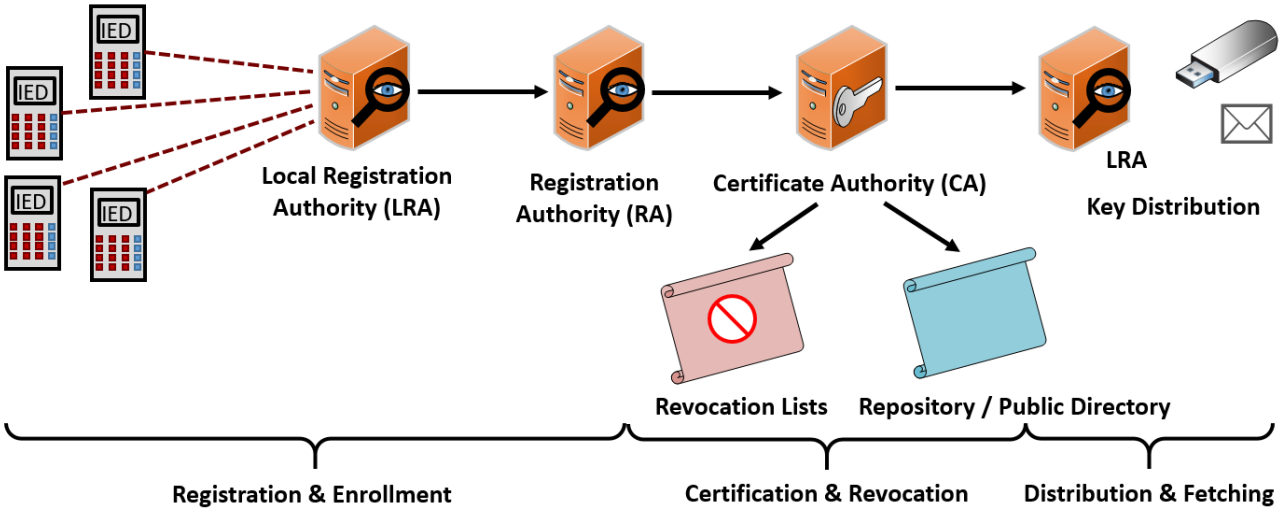


Abbildung 4: Komplexität der Schlüsselanzahl für symmetrische Verschlüsselung über bidirektional bekannte Schlüssel (Shared Secrets) sowie für die asymmetrische Verschlüsselung über eine zentrale Infrastruktur öffentlicher Schlüssel (Public Key Infrastructure (PKI)) (nach (Harner, 2019)).

Zertifikate sind von einer vertrauenswürdigen Stelle mit einer zeitlichen Beschränkung auszugeben.

Die Verwaltung der Zertifikate (siehe Abbildung 5 für eine Public Key Infrastructure (PKI)) umfasst unter anderem, die Registrierung von Nutzern, die Schlüsselpaargenerierung, Zertifikatsbeantragung, -generierung und -verteilung sowie die Bereitstellung von Gültigkeitsinformation.



Enrollment	Revocation	Distribution & Fetching
• manual	• manual (CRL)	• manual (configuration)
• automated (**EST, SCEP**, CMP, CMC)	• automated (**CRL**, OCSP)	• automated (LDAP, HTTP)

Abbildung 5: Zertifikatsmanagement (nach (Harner, 2019)).

Hierbei ist zu unterscheiden zwischen der initialen Gerätezertifizierung (sogenanntes Imprinting) in sicherer Umgebung durch den Gerätehersteller mit Geräteleben langer Laufzeit sowie der operativen Zertifizierung (sogenanntes Bootstrapping) bei Inbetriebnahme / während des Betriebes mit regelmäßiger Aktualisierung – sinnvollerweise unter Berücksichtigung des Herstellerzertifikats. Für das automatisierte Zertifikatsmanagment zwischen Geräten wird im IEC 62351-9 auf bereits existierende Protokolle wie Enrollment over Secure Transport (EST), Simple Certificate Enrollment Protocol (SCEP), Certificate Managment Protocol (CMP) unter Verwendung des X.509 Zertifikatformats verwiesen.

2.1.4 Zugangskontrolle

Die rollenbasierte Zugriffskontrolle (Role-based Access Control / RBAC) (INCITS, 2004) ist ein Entwurfsmuster und Verfahren zur Steuerung des Zugriffs auf IT-Systemressourcen (Harner, 2019). Hierbei werden Subjekten (Nutzern, Anwendungen, Geräte) nicht direkt einzelne Zugriffsrechte eingeräumt. Bei RBAC werden Berechtigungen an Rollen geknüpft und Subjekten ihre Rollen zugeordnet. Eine Rolle spiegelt hierbei eine übergeordnete Aufgabe eines Subjekts oder mehrerer Subjekte wider. Dies hat das Ziel, einzelne Berechtigungen zu gruppieren, und diese Subjekten schneller und eleganter gemäß ihrer Aufgaben gewähren/entziehen zu können und nach dem Prinzip geringster Rechte (least privilege) nur die Zugriffe zu erlauben, die für die Erledigung der Aufgaben nötig sind, ohne die Nutzer hierbei einzuschränken.

Die IEC 62351-8 (IEC, IEC/TS 62351-8: Power systems management and associated information exchange – Data and communications security – Part 8: Role-based access control , 2011) beschreibt den RBAC Mechanismus und dessen Integration in der Energieversorgung insbesondere auch mit dem Schwerpunkt der Interoperabilität in der Multi-Hersteller Umgebung der Umspannwerkautomatisierung (Cleveland, 2012). Das IEC 62351-8 spezifiziert hierzu

- das Format von Berechtigungsnachweisen, sogenannten Credentials oder Access Token,
- obligatorische Sicherheitsrollen für Administration, Auditierung und Wartung,
- die Übertragung von Rollen für TCP/IP und serielle Kommunikation,
- nötige Energiesystemdatenmodell-Erweiterungen für die Implementierung von RBAC
- die Verifikation von Berechtigungsnachweisen im Zielsystem zur Sicherstellung sicherer Zugriffskontrolle.

Für die Verifikation werden in der Spezifikation die generelle Zugriffskontrollarchitekturen nach dem PULL Modell sowie gemäß des PUSH Modells vorgestellt (siehe Abbildung 6 und Abbildung 7).

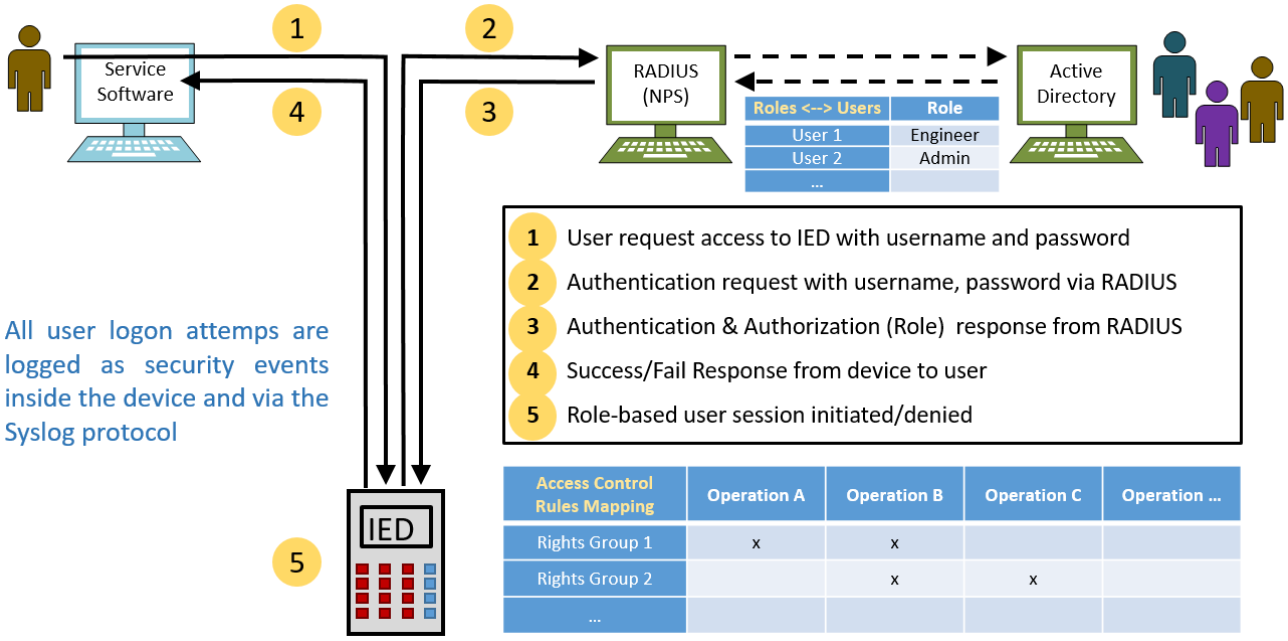


Abbildung 6: Zugriffskonzept von RBAC mittels PULL-Verfahren (nach (Harner, 2019)).

Im PULL-Verfahren authentifiziert sich das Subjekt bei dem Anfrageziel, wobei das Anfrageziel mit den Zugangsdaten des Subjekts dessen Rolle von einem zentralen Dienst abfragt, bevor es die Berechtigung verifiziert und die Anfrage durchführt.

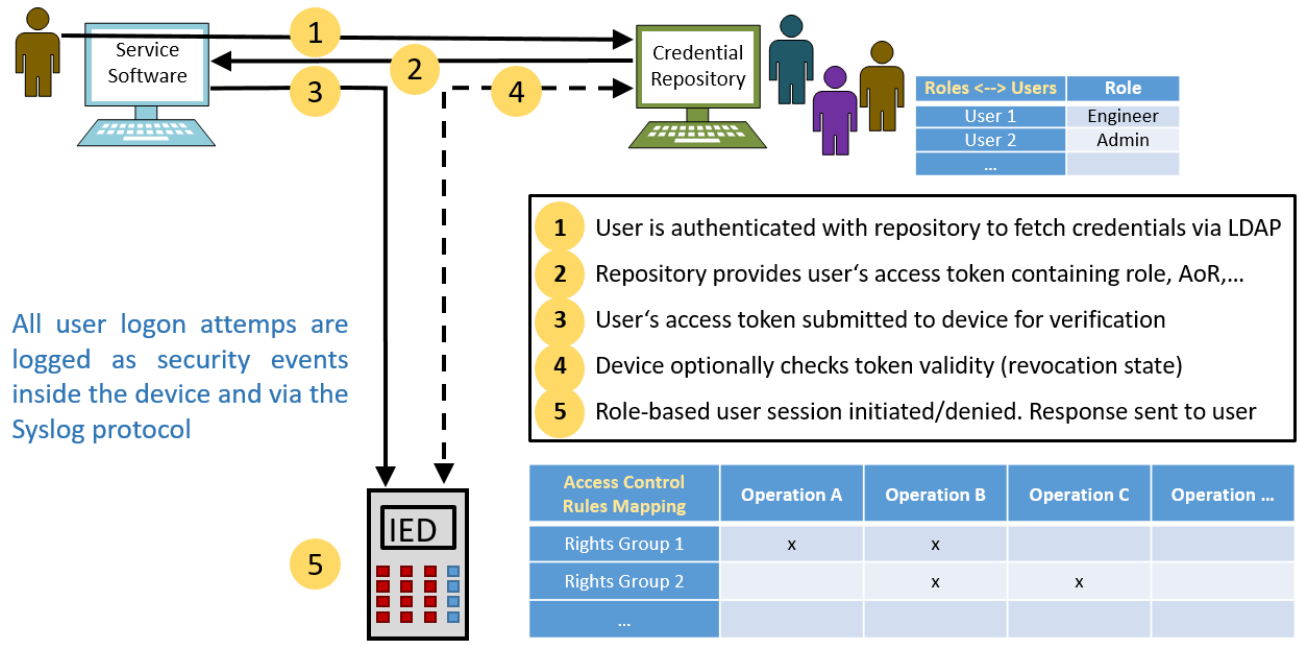


Abbildung 7: Zugriffskonzept von RBAC mittels PUSH-Verfahren (nach (Harner, 2019)).

Im PUSH-Verfahren authentifiziert sich das Subjekt beim zentralen Dienst und sendet die Authentifizierung sowie die Rollen und Berechtigungen (Credentials) anschließend zum Anfrageziel, welches die Autorisierung nach Validierung der Credentials am zentralen Dienst selbst verifizieren kann.

Zudem werden in der Spezifikation Kriterien für die Definition von Rollen sowie Rechten und ihre Verknüpfung im Allgemeinen sowie in der industriellen Prozesskontrolle aber auch speziell in der Energieversorgung unter Berücksichtigung des Gerätezugriffs, der Automatisierung, des Netzbetriebes, der Messinfrastruktur, dezentraler erneuerbarer Energien und der Marktintegration benannt.

Die Zugangskontrolle schließt dabei nahtlos an die Verwendung von Zertifikaten zur Identifikationsverifizierung aus dem vorherigen Abschnitt an und erweitert die öffentlichen Zertifikate durch Attributzertifikate mit kurzer Lebensdauer zur Gewährung von Rechten.

2.1.5 Daten- und Kommunikationssicherheit

Für die Ende-zu-Ende Sicherheit beim Datenaustausch geben die IEC 62351-3 bis IEC 62351-6 einen Sicherheitsrahmen mit verschiedenen Sicherheitszielen für die für den Energiesektor durch die TC 57 spezifizierten Schnittstellen vor. Insbesondere wird die Wiederverwendung von auch außerhalb der Energiebranche verbreiteten Sicherheitstechnologien wie TLS für die Sicherung TCP/IP basierter Verbindungen spezifiziert. Der Fokus dieser Normen liegt im Wesentlichen darauf, die Prozesskommunikation zwischen Leit- und Fernwirktechnik so abzusichern, dass diese den Anforderungen an einem sicheren IT-System entspricht. Die Diagnose- und Wartungsschnittstellen werden in der Normenreihe IEC 62351 nur unzureichend abgedeckt (Harner, 2019).

Die Norm „IEC 62351-3: Data and Communication Security – Profiles including TCP/IP“ deckt dabei die Sicherheitsmechanismen für den Datenaustausch mittels IEC 60870-6, IEC 60870-5-104, IEEE 1815 over TCP/IP, IEC 61850 over TCP/IP ab. Hierzu wird die Verwendung des Transport Layer Security (TLS) spezifiziert und für die Verwendung zu nutzende Parameter/Einstellungen beschrieben. Dadurch schützt die IEC 62351-3 vor Mithören und Wiederabspielen durch TLS-Verschlüsselung, Man-in-the-middle-Angriffe durch Nachrichten-Authentifizierung, die Verschleierung von Identitäten (Spoofing) durch Sicherheitszertifikate. Für Denial-of-Service (DoS) Angriffe, die niemals durch TLS abgefangen werden, wird jedoch auf Implementations-spezifische Maßnahmen verwiesen (Cleveland, 2012).

Die Norm „IEC 62351-4: Data and Communication Secuirty – Profiles including MMS and derivatives“ deckt die Sicherheitsmechanismen für den Datenaustausch mittels IEC 60870-6 sowie IEC 61850 over MMS ab. Es unterscheidet zwischen T-Sicherheitsprofilen und A-Sicherheitsprofilen für die Sicherheit auf den

unterschiedlichen Ebenen des ISO-Schichtenmodells. Auch hier wird TLS verwendet um die Transport-Sicherheit zu gewährleisten. Es erlaubt die gleichzeitige Verwendung von sicherer und ungesicherter Kommunikation, benötigt zudem jedoch weitere Sicherheitsmaßnamen wie Association Control Service Elements (ACSE) in den Präsentations- und Anwendungsschichten. Darüber hinaus macht die Norm Vorgaben für die End-zu-end Applikationssicherheit sowie dessen Fehlerbehandlung und beschreibt die End-zu-End-Sicherheit für OSI Umgebungen und für XMPP.

Die technische Spezifikation „IEC 62351-5: Data and Communication Security – Security for IEC 60870-5 and Devrivates (i.e. DNP 3.0)“ deckt die Sicherheitsmechanismen sowohl für den seriellen als auch netzwerkbasierten Datenaustausch mittels IEEE 1815 und allen Teilen des IEC 60870-5 ab. Insbesondere für den seriellen Datenaustausch werden übliche Beschränkungen des Kommunikationsmediums und sich daraus ergebende Einschränkungen bei der Umsetzung der Sicherheit benannt. Daraus ergibt sich im Wesentlichen eine Konzentration auf Sicherheitsanforderungen hinsichtlich Authentifizierung, Verhinderung von Identitätsverschleierung, Wiederabspielen oder Modifikation und einige DoS-Angriffe und lässt Ressourcenhungrige Sicherheitsanforderungen wie solche, die Verschlüsselung benötigen außen vor, benennt aber Alternativen wie VPN, in Abhängigkeit verwendbarer Ressourcen.

Die technische Spezifikation „IEC 62351-6: Data and Communication Security – Secutiry for IEC 61850 Peer-to-Peer Profiles (e.g. GOOSE)“ deckt die Sicherheitsmechanismen für den Datenaustausch mittels IEC 61850 Peer-to-Peer (P2P) Datenpaketen ab. Das Hauptprotokoll für P2P ist das sogenannte GOOSE-Protokoll, welches für Schutzeinrichtungen Nachrichten innerhalb von 4 Millisekunden zwischen Controllern austauschen können muss. Verschlüsselung oder jegliche weitere Sicherheitsmaßnahme, die die Übertragungsrate signifikant beeinflusst, ist deshalb nicht akzeptabel, weshalb nur Authentifizierung als Anforderung übrigbleibt. Das IEC 62351-6 stellt einen Mechanismus zur Verfügung, um unter Verwendung minimaler Ressourcen Nachrichten digital zu signieren (Cleveland, 2012).

Die Norm „IEC 62351-11: Data and Communication Security - Security for XML files“ deckt die Sicherheitsmechanismen für die Präsentations-/Anwendungsschicht des ISO/OSI Schichtenmodells für XML Nachrichten ab und übernimmt Teile dieser Mechanismen aus der W3C. Diese Sicherheitsmechanismen spielen insbesondere bei der Übertragung von IEC 61970/61968/62325 CIM sowie IEC 61850 XML-Nachrichten eine Rolle, können aber auch für die Sicherung jeder weiteren XML Nachricht verwendet werden. Dazu definiert die Norm einen Header, in der die ursprüngliche Nachricht im Klartext oder verschlüsselt eingebettet wird. Zudem finden sich im Header weitere Informationen hinsichtlich der Verschlüsselung und der Signatur des Dokuments. Dies geschieht mit dem Ziel, unautorisierten Zugang und unautorisierte Modifikation von Nachrichten zu verhindern.

2.1.6 Netzwerk- und System-Management der IT-Infrastruktur

Der Energienetzbetrieb ist zunehmend abhängig von IT-Infrastruktur. Deshalb ist das Management dieser Infrastruktur kritisch für die Bereitstellung eines hohen Sicherheitslevels und der Zuverlässigkeit im Energienetzbetrieb (Cleveland, 2012). Gemäß des genannten Berichts, können ISO CMIP und der IETF SNMP Standard dieses Management in Teilen unterstützen, indem in SMNP sogenannte Management Information Base (MIB) Daten genutzt werden um den Zustand von Netzwerk und Systemen zu überwachen. Für den Energiesystembetrieb werden SMNP MIB jedoch nur für allgemeine Netzwerkgeräte wie Router bereitgestellt.

Mit der IEC 62351-7 (IEC, 62351-7: Power systems management and associated information exchange - Data and communications security - Part 7: Network and System Management (NSM) data object models, 2017) werden MIB-ähnliche Datenobjekte, sogenannte Network and System Management (NSM) Datenobjekte für die IT-Infrastruktur der Energiedomäne spezifiziert, deren Übertragung nicht an SMNP gekoppelt ist, sondern unabhängig davon erfolgen kann. Zudem wird der sichere Umgang mit SNMP und Zeitsynchronisation beschrieben.

2.2 *BSI Grundschutz*

Das BSI Grundschutz-Kompendium (BSI, 2020) beschreibt standardisierte Sicherheitsanforderungen für typische Geschäftsprozesse, Anwendungen, IT-Systeme, Kommunikationsverbindungen und Räume in einzelnen Bausteinen mit dem Ziel, einen angemessenen Schutz für alle Informationen einer Institution zu erreichen. Das in der IT-Grundschutz-Methodik verwendete Baukastenprinzip soll den heterogenen Bereich der Informationstechnik und ihrer Einsatzumgebung strukturieren und besser planbar machen. Für den sicheren Betrieb von Software/Hardware werden im BSI Grundschutz-Kompendium Vorgaben gemacht, die jedoch nicht Teil dieser Betrachtung sind. Für diese Betrachtung wird davon ausgegangen, das für den Betrieb ausreichend gehärtete Geräte / und grundlegende Betriebssystemeinstellungen / Virtualisierung zur Verfügung gestellt und gepflegt werden. Für die Entwurfsphasensicherheit spielen die im Folgenden benannten Bausteine des BSI Grundschutz-Kompendiums im Allgemeinen eine wesentliche Rolle.

2.2.1 ORP.4 Identitäts- und Berechtigungsmanagement

Wie bereits in der IEC 62351 spielt auch im BSI Grundschutz-Kompendium ein Baustein für das Identitäts- und Berechtigungsmanagement eine wichtige Rolle. Benutzer und IT-Komponenten müssen zweifelsfrei identifiziert und authentifiziert werden und der Zugang zu schützenswerten Ressourcen ist auf berechtigte Benutzer und IT-Komponenten einzuschränken (BSI, 2020).

2.2.2 CON.1 Kryptokonzept

Ein wichtiger Baustein für die Einhaltung der Schutzziele Vertraulichkeit, Integrität und Authentizität ist Verschlüsselung. Der BSI Grundschutz-Baustein CON.1 zielt darauf ab, ein institutionsweites Kryptokonzept aufzustellen. Für die Entwicklung von Software spielen in der Entwurfsphase insbesondere die Anforderungen

- CON.1.A1 Auswahl geeigneter kryptografischer Verfahren [Fachverantwortliche]
- CON.1.A3 Verschlüsselung der Kommunikationsverbindungen

eine Rolle.

2.2.3 CON.3 Datensicherungskonzept

Datensicherung soll Auswirkungen von Datenverlust minimieren. Für die Entwurfsphase von Software ist hier relevant, dass Daten sicher gesichert und wiedereingespielt werden können.

2.2.4 CON.5 Entwicklung und Einsatz von Individualsoftware

Für die Entwicklung von Individualsoftware beschreibt das BSI-Grundschutz-Kompendium einen Baustein. Hierbei wird auf die essentielle Bedeutung hingewiesen, bereits bei der Planung und Konzeptionierung der Individualsoftware benötigte Sicherheitsfunktionen zu bedenken und Informationssicherheit des gesamten Lebenszyklus der Software zu berücksichtigen (BSI, 2020).

Das BSI-Grundschutz-Kompendium nennt hinsichtlich der Entwicklung von Softwarekomponenten in der Entwurfsphase im Wesentlichen die Anforderung CON.5.A1: Festlegung benötigter Sicherheitsfunktionen der Individualsoftware.

2.2.5 CON.8 Software-Entwicklung

Auch im BSI-Grundschutz-Kompendium wird die Berücksichtigung der Informationssicherheit über den gesamten Softwae-Entwicklungsprozess hinweg als essentiell betrachtet und von zusätzlichen Aufwänden ausgegangen, falls diese erst in späten Phasen beachtet wird. Für die Software-Entwicklung gibt das BSI-Grundschutz-Kompendium Anforderungen schon während der Entwurfsphase von Software vor. Diese sind:

- CON.8.A5 Sicheres Systemdesign
- CON.8.A6 Verwendung von Bibliotheken aus vertrauenswürdigen Quellen
- CON.8.A9 Berücksichtigung von Compliance-Anforderungen
- CON.8.A14 Schulung des Projektteams zur Informationssicherheit

Weitere Anforderungen beziehen sich auf die Phase der Quellcode-Implementierung und der Inbetriebnahme.

2.2.6 OPS.1.1.5 Protokollierung

Für die Gewährleistung eines verlässlichen IT-Betriebes sollen dem BSI-Grundschutz-Kompendium zufolge entweder alle oder zumindest ausgewählte betriebs- und sicherheitsrelevante Ereignisse protokolliert werden. Für die zentrale Auswahl, Filterung und systematische Auswertung kann eine dedizierte zentrale Protokollinfrastruktur genutzt werden. Folgende Anforderungen sind bereits während der Entwurfsphase von Software relevant:

- OPS.1.1.5.A1 Erstellung einer Sicherheitsrichtlinie für die Protokollierung
- OPS.1.1.5.A4 Zeitsynchronisation der IT-Systeme
- OPS.1.1.5.A5 Einhaltung rechtlicher Rahmenbedingungen
- OPS.1.1.5.A10 Zugriffsschutz für Protokollierungsdaten
- OPS.1.1.5.A11 Steigerung des Protokollierungsumfangs
- OPS.1.1.5.A12 Verschlüsselung der Protokollierungsdaten

2.2.7 OPS.1.1.6 Software-Tests und –Freigaben

Von einer weitestgehend fehlerfreien Datenverarbeitung durch IT-Systeme kann nicht ohne weiteres ausgegangen werden. Aus diesem Grund muss Software schon vor Inbetriebnahme durch Software-Test geprüft werden und dadurch nachgewiesen werden, dass die erforderlichen Funktionen zuverlässig bereitgestellt werden (BSI, 2020). Je nach verwendetem Entwicklungsprozess finden sich bereits in der Entwurfsphase von Software folgende Anforderungen im BSI-Grundschutz-Kompendium, deren Ausprägungen sicherlich aber auch während der Implementierung von Software vervollständigt werden müssen.

- OPS.1.1.6.A1 Planung der Software-Tests
- OPS.1.1.6.A9 Beschaffung von Test-Software
- OPS.1.1.6.A10 Erstellung eines Abnahmeplans
- OPS.1.1.6.A11 Verwendung von anonymisierten oder pseudonymisierten Testdaten
- OPS.1.1.6.A14 Durchführung von Penetrationstests

2.2.8 APP.2.1 Allgemeiner Verzeichnisdienst, APP.2.2 Active Directory, APP.2.3 OpenLDAP

Soll ein allgemeiner Verzeichnisdienst, Active Directory oder OpenLDAP für die Benutzerverwaltung und Authentisierung angebunden werden, so sind insbesondere die Anforderungen hinsichtlich eines geeigneten Berechtigungskonzepts

- APP.2.1.A3 Einrichtung von Zugriffsberechtigungen auf Verzeichnisdienste

beziehungsweise

- APP.2.2.A3 Planung der Gruppenrichtlinien unter Windows

beziehungsweise

- APP.2.3.A5 Sichere Vergabe von Zugriffsrechten auf dem OpenLDAP

sowie hinsichtlich einer geeigneten technischen Kommunikation zum Verzeichnisdienst

- APP.2.1.A13 Absicherung der Kommunikation mit Verzeichnisdiensten

beziehungsweise

- APP.2.2.A9 Schutz der Authentisierung beim Einsatz von Active Directory

beziehungsweise

- APP.2.3.A3 Sichere Konfiguration von OpenLDAP
- APP.2.3.A6 Sichere Authentisierung gegenüber OpenLDAP

für die Software-Entwurfsphase zu berücksichtigen.

2.2.9 Weitere relevante BSI-Grundschutz-Komponenten in Abhängigkeit der Funktionalität der zu entwickelnden Software

Zu den allgemeinen Anforderungen hinsichtlich der Entwurfsphase von Software kommen besondere Anforderungen an die zu entwickelnde Software in Abhängigkeit ihrer Funktionalität, die im Folgenden Aufgelistet werden, aber nur im Spezialfall benötigt werden. Für einige der Bausteine des BSI-Grundschutz Kompendiums (wie beispielsweise APP.3.2 Webservice oder APP.4.3 Relationale Datenbanksysteme) scheint eine traditionelle Systemarchitektur der zentralen Bereitstellung Grundlage zu sein. Entsprechende Vorgaben sind aber insbesondere auch bei moderneren Herangehensweisen der verteilten oder individuellen Bereitstellung (beispielsweise in einer Micro-Service-Architektur) nicht nur global, sondern auch im Einzelfall zu berücksichtigen.

2.2.9.1 CON.2 Datenschutz

Werden durch die Software personenbezogene Daten verarbeitet, spielt Datenschutz eine wichtige Rolle. Datenschutz ist der Schutz personenbezogener Daten, dessen Rahmenbedingungen durch europäische und nationale Gesetze vorgegeben werden. Dies beinhaltet unter anderem,

- dass Recht einer jeden Person über die Verwendung ihrer eigenen personenbezogenen Daten zu bestimmen,
- die Nutzung von personenbezogenen Daten durch Dritte so zu verhindern, dass Personen vor Beeinträchtigung der Ausübung ihrer Grundrechte geschützt sind,
- dass eine Rechtgrundlage zur Datenverarbeitung notwendig ist,
- dass die Überwachung der Einhaltung von Datenschutzvorschriften durch unabhängige Stellen erfolgt.

Die gesetzlichen Bestimmungen zum Datenschutz (DSGVO, BDSG und LDSG) sind einzuhalten (BSI, 2020).

2.2.9.2 CON.9 Informationsaustausch

Hinsichtlich eines etwaigen Informationsaustausches mit externen Empfängern gibt das BSI-Grundschutz-Kompendium Anforderungen für die Weitergabe von Informationen. Die folgenden Anforderungen sind bereits während der Entwurfsphase von Software zu berücksichtigen, falls ein Austausch mit externen Empfängern erfolgt:

- CON.9.A2 Regelung des Informationsaustausches
- CON.9.A5 Beseitigung von Restinformationen in Dateien vor Weitergabe
- CON.9.A6 Kompatibilitätsprüfung des Sender- und Empfängersystems
- CON.9.A7 Sicherungskopie der übermittelten Daten
- CON.9.A8 Verschlüsselung und Signatur

2.2.9.3 OPS.1.2.2 Archivierung

Für einige Daten/digitale Dokumente gibt es Aufbewahrungsfristen. Die Archivierung dieser Daten unterliegt also besonderen Anforderungen an die zeitliche Verfügbarkeit, aber auch hinsichtlich ihrer Vertraulichkeit und Integrität. Zudem muss gemäß des BSI Grundschutz-Kompendiums der Kontext erhalten werden, damit der jeweilig archivierte Vorgang rekonstruiert werden kann. Dies Umfasst für die Entwurfsphase von Software im Wesentlichen die Anforderungen

- OPS.1.2.2.A2 Entwicklung eines Archivierungskonzepts
- OPS.1.2.2.A20 Geeigneter Einsatz kryptografischer Verfahren bei der Archivierung

des BSI Grundschutz-Kompendiums und hier im Besonderen die Festlegung, welche Daten wie zu Archivieren sind. Im Spezialfall der Entwicklung eines Archivsystems gelten neben den im BSI-Grundschutz-Kompendium genannten archivierungsspezifischen Anforderungen natürlich auch alle weiteren Anforderungen.

2.2.9.4 OPS.1.2.4 Telearbeit

Wird die entwickelte Software ganz oder teilweise außerhalb der Geschäftsräume und Gebäude des Arbeitgebers verrichtet und dazu externe Informations- und Kommunikationstechnik verwendet, sind

diesbezüglich auch bereits während der Entwurfsphase der Software die Folgenden im BSI-Grundschutz-Kompendium benannten Anforderungen zu berücksichtigen.

- OPS.1.2.4.A3 Sicherheitstechnische Anforderungen an die Kommunikationsverbindung
- OPS.1.2.4.A4 Datensicherheit bei der Telearbeit
- OPS.1.2.4.A6 Erstellung eines Sicherheitskonzeptes für Telearbeit

2.2.9.5 OPS.1.2.5 Fernwartung

Unter Fernwartung versteht das BSI Grundschutz-Kompendium den „räumlich getrennten Zugriff auf IT-Systeme und die darauf laufenden Anwendungen“ und weist auf die besondere Bedeutung der Absicherung der Fernwartungskomponenten aufgrund der tiefgreifenden Eingriffsmöglichkeiten in IT-Systeme und vielen Abhängigkeiten zu weiteren BSI Grundschutz-Bausteinen hin (BSI, 2020). Im Wesentlichen bezieht das BSI-Grundschutz-Kompendium dies jedoch auf entfernte Monitorausgaben und - bei der aktiven Fernwartung - auf den Zugriff/Übernahme unterliegender Schichten eines Systems, sodass beispielsweise Maus- und Tastatureingaben von entfernt vorgenommen werden können. Sind in der zu entwickelnden Komponente direkt Fernwartungsschnittstellen vorgesehen, so spielen alle Anforderungen des OPS.1.2.5 Fernwartung eine wichtige Rolle.

2.2.9.6 OPS.2.1 Outsourcing für Kunden und OPS.2.2 Cloud-Nutzung

Wird die zu entwickelnde Software ganz oder in Teilen durch einen externen Dienstleister betrieben sind in Abhängigkeit des Hostings, der Bereitstellung, der SLAs und der technischen Fähigkeiten der entsprechenden Plattform besondere Sicherungsmaßnahmen nötig, die das Grundschutz-Kompendium auflistet und die alle bereits zur Entwurfsphase mit berücksichtigt werden sollten, damit die Komponente auch schlussendlich sicher betrieben und eingebunden werden kann.

2.2.9.7 APP.3.1 Webanwendungen und APP.3.2 Webserver

Stellt die zu entwickelnde Komponente eine Webanwendung dar, so sind alle im BSI Grundschutz-Kompendium genannten Anforderungen hinsichtlich APP.3.1 Webanwendungen auf der Client-Seite und Server-Seite sowie APP.3.2 Webserver auf der Server-Seite zu berücksichtigen. Letzteres spielt insbesondere auch dann eine Rolle für die Bereitstellung der Komponente, wenn in (Micro-) Service orientierter Architektur jeder Dienst über einen eigenen Webservice bereitstellt wird.

2.2.9.8 APP.4.3: Relationale Datenbanksysteme

Verwendet die zu entwickelnde Komponente Datenbanksysteme beispielsweise zur Persistierung von Daten, so gibt das BSI Grundschutz-Kompendium für relationale Datenbanksysteme Anforderungen vor, welche zu erfüllen sind. Diese spielen insbesondere auch dann eine Rolle für die Bereitstellung der Komponente, wenn in (Micro-) Service orientierter Architektur jede Komponente eigene Datenbanken bereitstellt und verwendet. In Ergänzung zum BSI Grundschutz-Kompendium sollte angemerkt werden, dass die Anforderungen sich jedoch nicht nur auf relationale Datenbanken beschränken sollten. Die Anforderungen sind weitestgehend direkt anwendbar oder ohne Schwierigkeiten übertragbar auf weitere Datenbankentypen und sollten dort auch Anwendung finden.

2.2.9.9 Weitere Spezialfälle

Setzt die Software spezielle funktionale Anforderungen um, so gibt das BSI Grundschutz Kompendium zum Teil weitere Sicherheitsanforderungen vor, wie beispielsweise für die DER.1 Detektion sicherheitsrelevanter Ereignisse. Soll die zu entwickelnde Software zur Detektion von sicherheitsrelevanten Ereignissen beitragen, sind diese Funktionen zu aktivieren und deren Ergebnisse zu protokollieren (DER.1.A5) und gegebenenfalls die Integrität der Software zu kontrollieren (DER.1.A18).

Wird die zu entwickelnde Software an bestehende Systeme angebunden, so sind deren spezielle Sicherheitsanforderungen weiterhin zu erfüllen und die Sicherheit nicht durch die neu hinzugefügten Komponenten zu schwächen. Dies betrifft beispielsweise die Anforderungen zu APP.3.3 Fileserver, APP.3.4

Samba-Server, APP.3.6 DNS-Server, APP.4.2 SAP-ERP-System, APP.4.6 SAP ABAP-Programmierung, APP.5.1 Allgemeine Groupware, APP.5.2 Microsoft Exchange und Outlook.

2.2.10 Weitere relevante BSI Grundschutz-Komponenten in Abhängigkeit der Domäne der technischen Umsetzung

Für unterschiedlichen Domänen der technischen Umsetzung wie die Betriebs- und Steuertechnik, die Gerätetechnik, sowie die Netzwerktechnik gibt das BSI Grundschutz-Kompendium Sicherheitsanforderung vor. Diese Wiederholen beziehungsweise präzisieren die vorgenannten Anforderungen betreffend der Sicherheit in der Entwurfsphase von Software im Wesentlichen. Für Entwurfsphase von Software, die in der Umgebung von Netzleitsystemen eingesetzt werden soll, sind sicherlich die im Folgenden genannten Anforderungen aus den Domänen relevant, in denen eine technische Umsetzung erfolgt.

2.2.10.1 IND.1 Betriebs- und Steuertechnik

In den Bereich industrieller Betriebs- und Steuertechnik fallen gemäß des BSI Grundschutz-Kompendium die Kritischen Infrastrukturen und damit insbesondere deren Steuerungssysteme und Automationslösungen (BSI, 2020). Damit gehörten auch Netzleitsysteme und die angebundene Betriebstechnik dieser Domäne an. Dementsprechend sind auch alle Anforderungen dieses BSI Grundschutz-Bausteins im Umfeld von Netzleitsystemen umzusetzen. Hinsichtlich der Entwurfsphasensicherheit spielen über die vorab genannten allgemeinen Anforderungen besonders die folgenden Anforderungen eine Rolle.

- IND.1.A1 Einbindung in die Sicherheitsorganisation
- IND.1.A5 Entwicklung eines geeigneten Zonenkonzepts / IND.1.A16 Stärkere Abschottung der Zonen
- IND.1.A13 Notfallplanung

2.2.10.2 IND.2 ICS-Komponenten

Gerätenahe Software ist nicht Teil dieser Betrachtung

2.2.10.3 NET.1.1 Netzarchitektur und -design

Das BSI-Grundschutz-Kompendium beschreibt umfänglich Anforderungen für die Planung und den Betrieb von IT-Netzwerken, Netzwerkmedien und IT-Netzwerkkomponenten. Für die Sicherheit sind in der Entwurfsphase von Software im Wesentlichen die schon in IND.1 Betriebs- und Steuerungstechnik benannten Zonenkonzepte sowie die verwendeten Übertragungswege zu berücksichtigen. Die Berücksichtigung der Zonen ist sicherlich bei einem Software-Entwicklungsprojekt schwierig, wenn nicht für ein konkretes Unternehmen entwickelt wird. In diesem Fall könnte ein konkretes oder fiktives Unternehmenszonenkonzept als Grundlage dienen.

2.3 *BDEW Whitepaper*

Der Bundesverband der Energie- und Wasserwirtschaft e.V. (BDEW) sowie der Interessenvertretung Österreichs E-Wirtschaft geben zusammen Anforderungen an kritische Informationssysteme für den Bereich der Energieversorgung heraus, welche im Wesentlichen der Normenreihe ISO/IEC 27000 folgen jedoch unter Umständen erweitert wurden (BDEW, 2018). Diese Anforderungen werden im Folgenden hinsichtlich der Software-Entwurfsphase kurz vorgestellt.

Auch das BDEW Whitepaper benennt für eine sichere Systemarchitektur Security-By-Design als wichtiges Prinzip, um Einzelkomponenten und das Gesamtsystem auf einen sicheren Betrieb hin zu entwickeln. Dazu werden noch die Prinzipien der Begrenzung auf die zur Erledigung von Arbeiten notwenigen Rechte, die Implementierung gestaffelter Sicherheitsmaßnahmen sowie die Redundanz zur Verringerung von Auswirkungen von Problemen zentral hervorgehoben.

Zudem wird auf die Anforderungen zur Verschlüsselung vertraulicher Daten hinsichtlich Informationssicherheits- und Datenschutzaspekten, die sichere Datenspeicherung und Übertragung sowie auf kryptografische Verfahren hingewiesen.

Weiterhin macht das BDEW Whitepaper Vorgaben hinsichtlich einzusetzender Netzwerk-Protokolle und -Technologien, insbesondere bezüglich Integrität, Authentifizierung und Verschlüsselung und benennt für

einzelne Technologien beziehungsweise Protokolle deren Verwendbarkeit. Auch wird im BDEW Whitepaper eine sichere Netzwerkstruktur mit Netzwerksegmentierung sowie sicherere Fern-Zugänge für Administration, Wartung und Konfiguration aller Komponenten verlangt.

Interessant für die Softwareentwicklung im Umfeld von Netzleitsystemen sind besonders auch die Anforderungen des BDEW Whitepaper hinsichtlich Anwendungen und Fachapplikationen. Hier wird insbesondere die granulare Zugriffskontrolle auf Daten und Ressourcen mittels Rollen- und Rechtekonzept, die personenspezifische Benutzer-Authentifizierung und Anmeldung sowie die Autorisierung von Aktionen auf Benutzer- und Systemebene genannt. Hinsichtlich Web-Anwendungen, -Schnittstellen und -Dienste wird auf den BSI-Leitfaden zur Entwicklung sicherer Webanwendungen verwiesen und darüber hinaus der OWASP Application Security Verification Standard (siehe (OWASP, 2019)) als Anforderungsgeber im Level L2 (Standard) für die Prozesssteuerung der Energieversorgung und Level L3 (Advanced) im Bereich kritischer Infrastrukturen sowie die OWASP Top 10 (OWASP, 2020) genannt (BDEW, 2018). Hinzu kommt die Integritätsprüfung von Daten vor der Verarbeitung sowie Anforderungen hinsichtlich Logging / Protokollierung.

Der Entwicklung von Hard- und Software wird im BDEW Whitepaper ein eigener Abschnitt zugeordnet. Die Anforderungen zur Entwurfsphasensicherheit sind im Wesentlichen die Schulung der Entwickler sowie die Entwicklung nach anerkannten Standards insbesondere unter Berücksichtigung sicherheitsrelevanter Entwicklungsschritte. Die Definition der Sicherheitsanforderungen, die Bedrohungsmodellierung und Risikoanalyse, die Ableitung von Anforderungen an das Systemdesign und an die Implementierung werden für die Entwurfsphase genannt.

2.4 *Zusammenfassung Entwurfsphasensicherheit*

Sowohl aus dem IEC 62351, als auch dem BSI Grundschutz-Kompendium und dem BDEW Whitepaper ergeben sich Anforderungen an die Sicherheit während der Entwurfsphase einer Software Entwicklung. Alle drei Quellen weisen zudem auf die Notwendigkeit hin, Sicherheit von Anfang an bei der Software-Entwicklung mitzudenken.

Das BSI Grundschutz-Kompendium gibt im Allgemeinen Anforderungen für verschiedene Sicherheitslevel vor, die Insbesondere auch für den hohen Schutzbedarf Kritischer Infrastrukturen abzudecken sind.

Das IEC 62351 wurde im speziellen für den Kontext Kritische Infrastruktur Energieversorgung erarbeitet. Es bietet im Hinblick auf Authentizität, Integrität und Vertraulichkeit eine signifikante Verbesserung der Sicherheit. Obwohl der Standard mit einer Rückwärtskompatibilität nicht alle Möglichkeiten in Sachen Sicherheit ausschöpft, stellt der Standard einen ausgeglichenen Ansatz mit angemessenem Aufwand ein angemessenes Sicherheitsniveau zu erreichen dar, wenn er vollumfänglich umgesetzt wird (Schlegel, Obermeier, & Scheider, 2015).

Das BDEW Whitepaper zu Anforderungen an sichere Steuerungs- und Telekommunikationssysteme konkretisiert Anforderungen zur Informationssicherheit hinsichtlich der Energieversorgung.

Die drei genannten Publikationen(-reihen) machen Vorgaben sowohl für einzelne Komponenten als auch die gesamtsystemische Kombination dieser einzelnen Komponenten in einer Systemlandschaft einer Organisation. Die Sicherstellung der Informationssicherheit für die gesamte Systemlandschaft sollte in jedem der betroffenen Energieversorgungsunternehmen bereits als Konzept erarbeitet sein welches entsprechend umgesetzt ist. Deshalb wird an dieser Stelle darauf nicht tiefergehender eingegangen. Die erarbeitete Informationssicherheit sollte/darf durch neu hinzugefügte Software nicht geschwächt werden. Insofern sollte sich neu zu integrierende Software einen mindestens ebensolchen Informationssicherheitsanspruch erfüllen und sich zudem in das vorhandene Informationssicherheitskonzept einbetten.

Für die Entwurfsphase von Open Source Software im Umfeld von Netzleitsystemen ergeben sich folgende wesentliche Aufgaben hinsichtlich der Informationssicherheit:

1. Erfassung der funktionalen Aufgaben und qualitativen Anforderungen unter Berücksichtigung verwendeter Daten, zu implementierender und zu verwendender vorhandener Schnittstellen und deren technischen Einschränkungen.

2. Einordung/Einbettung in eine vorhandene/beispielhafte Sicherheitsarchitektur/Netzsegmentierung und Feststellung der Auswirkungen der Sicherheitsarchitektur auf die Anforderungen der Software unter Berücksichtigung der BSI-Grundschutz-Bausteine
    a. NET1.1 Netzarchitektur und -design
    b. IND.1 Betriebs- und Steuertechnik
    c. OPS.2.1 Outsourcing für Kunden – falls Teile der neuen Software outgesourced werden sollen
    d. OPS.2.2 Cloud-Nutzung – falls Teile der neuen Software aus der Cloud angeboten werden sollen

    sowie Priorisierung der Anforderungen.
3. Erfassung, Beurteilung und Umsetzungsplanung der sicherheitstechnischen Anforderungen **jeder Mensch-Machine- und Maschine-Maschine-Schnittstelle und der Anwendung oder jedem Anwendungsbaustein selbst**. Dies kann für Software-Bausteine oder Schnittstellen zusammengefasst erfolgen, wenn exakt die gleichen Anforderungen zugrunde liegen. Dazu sollte eine Tabelle als Grundlage dienen, in der die nötigen Anforderungen markiert, begründet und die zu verwendende Lösung/Technologie angegeben wird. Ein Vorschlag für die Einträge in einer solchen Tabelle unter Verwendung der Struktur des BSI Grundschutz-Kompendiums ist im Folgenden für die Softwareentwurfsphase aufgelistet:
    a. Identitäts- und Berechtigungsmanagement
        i. Zweifelsfreie Identifikation von Benutzern und Komponenten
        ii. Rollen- und Rechtekonzept für die Software(-Schnittstellen und Nutzer)
        iii. Sichere Anbindung bei Übertragung von Identitätsdaten / -Bestätigungen
        iv. Einbindung bestehender Verzeichnisdienste / Public Key Infrastructure / Zertifikatsmanagement
        v. Sichere Konfiguration der Rollen- und Rechte
        vi. Weitere Anforderungen bzgl. Identitäts- und Berechtigungsmanagement
    b. Kryptokonzept
        i. Geeignetes kryptografisches Verfahren für Verschlüsselung
        ii. Geeignetes kryptografisches Verfahren für Signierung
        iii. Weitere Anforderungen bzgl. Kryptokonzept
    c. Datensicherungskonzept
        i. Konzept zur sicheren Datensicherung
        ii. Konzept zur sicheren Datenwiedereinspielung
        iii. Weitere Anforderungen bzgl. Datensicherungskonzept
    d. Software-Entwicklung
        i. Festlegung einer geeigneten Open Source Lizenz (s.u.)
        ii. Verwendung von Bibliotheken aus vertrauenswürdigen Quellen
        iii. Berücksichtigung von Compliance-Anforderungen
        iv. Schulung des Projektteams zur Informationssicherheit
        v. Weitere Anforderungen bzgl. Software-Entwicklung
    e. Protokollierung / Logging / Monitoring
        i. Erstellung einer Sicherheitsrichtlinie für die Protokollierung / Logging / (IKT)-Monitoring (im Folgenden als Protokollierung zusammengefasst) und Einbettung in eine entsprechend bestehende Sicherheitsrichtlinie mit Anbindung an (zentrale) Protokollierungsdienste
        ii. Einhaltung rechtlicher Rahmenbedingungen
        iii. Zugriffsschutz für Protokollierungsdaten
        iv. Steigerung des Protokollierungsumfangs
        v. Verschlüsselung der Protokollierungsdaten
        vi. Signierung der Protokollierungsdaten
        vii. Zeitsynchronisierung
        viii. Weitere Anforderungen bzgl. Protokollierung

f. Software-Tests und -Freigaben
    i. Planung von Software-Tests
    ii. Berücksichtigung von Test-Software-Eigenschaften
    iii. Erstellung eines Abnahmeplans
    iv. Verwendung anonymisierter / pseudonymisierter Testdaten
    v. Berücksichtigung von Penetrationstests
    vi. Weitere Anforderungen bzgl. Software-Tests und –Freigaben
g. Weitere spezifische Anforderungen
    i. Datenschutz für personenbezogene Daten
        1. Verschlüsselung von Daten
        2. Rechtsgrundlage zur Datenverarbeitung
        3. Einhaltung gesetzlicher Anforderungen an den Datenschutz
        4. Überwachung der Einhaltung gesetzlicher Anforderungen an den Datenschutz
    ii. Informationsaustausch mit externen Empfängern
        1. Regelung des Informationsaustausches
        2. Beseitigung von Restinformationen vor Weitergabe
        3. Kompatibilitätsprüfung Sender / Empfänger
        4. Sicherungskopie der übermittelten Daten
        5. Verschlüsselung und Signatur
    iii. Archivierung für Dokumente / Daten mit Aufbewahrungsverpflichtung und -fristen
        1. Erfassung eines Archivierungskonzepts für Daten und Kontextdaten mit Archivierungsfristen
        2. Geeigneter Einsatz kryptografischer Verfahren zur Sicherstellung von Vertraulichkeit und Integrität
    iv. Telearbeit unter Verwendung externer IKT
        1. Sicherheitstechnische Anforderungen an die Kommunikationsverbindung
        2. Datensicherheit bei der Telearbeit
        3. Erstellung eines Sicherheitskonzepts für die Telearbeit und Einbettung in das Gesamtkonzept
    v. Detektion sicherheitsrelevanter Ereignisse
        5. Aktivierung von Detektoren sicherheitsrelevanter Ereignisse und Protokollierung der Ereignisse
        6. Integritätsprüfung
    vi. Fernwartung bei Weiterleitung von Monitorausgaben- sowie Tastatur-/Maus-Eingaben
        7. Einbettung in bestehendes Fernwartungskonzept oder falls dieses nicht besteht, Erstellung des entsprechenden Fernwartungskonzepts (s. folgenden Punkt)
    vii. Fernwartung bei direkter Integration von Fernwartungs-Schnittstellen
        8. Falls erforderlich: Übernahme aller Anforderungen aus OPS.1.2.5 Fernwartung in eine externe Tabelle
    viii. Webanwendungen
        9. Falls erforderlich: Übernahme aller Anforderungen aus APP.3.1 Webanwendungen in eine externe Tabelle
    ix. Webserver / Webservices
        10. Falls erforderlich: Übernahme aller Anforderungen aus APP.3.2 Webserver in eine externe Tabelle
    x. Datenbanksysteme
        11. Falls erforderlich: Übernahme aller Anforderungen aus APP.4.3 Relationale Datenbanken und gegebenenfalls unter Anpassung an das Datenbankparadigma in eine externe Tabelle

xi. Neue Komponenten / Schnittstellen in / an bestehende(n) Systeme(n)
    12. Einhaltung bisheriger Sicherheitsanforderungen des bestehenden Systems
    13. Falls erforderlich: Übernahme nötiger Anforderungen aus APP.3.3 Fileserver, APP.3.4 Samba-Server, APP3.6 DNS-Server, APP4.2 SAP-ERP-Systeme, APP4.6 SAP ABAP-Programmierung, APP5.1 Allgemeine Groupware oder/und APP.5.2 Exchange und Outlook in eine externe Tabelle
xii. Weitere spezifische Anforderungen

h. Weitere Anforderungen

Die genannten Hauptthemen der Tabelle unter Punkt 3 ergeben sich aus der Überschneidung der genannten Quellen. Die Unterpunkte sind hochgradig abhängig von der spezifischen Aufgabenstellung. Für weitere sich ergebende Anforderungen und eine individuelle Risikoeinschätzung und -behandlung können zusätzliche Felder eingefügt werden. Die nicht-Beachtung einer Anforderung muss begründet dokumentiert werden. Die Nicht-Beachtung kann sich aus spezifischen Anforderungen aber auch beispielsweise aus der Art der Komponente (Software-Baustein, Mensch-Maschine-Schnittstelle oder Maschine-Maschine-Schnittstelle) ergeben.

Die *„Festlegung einer geeigneten Open Source Lizenz“* wurde als Anforderung zur Software-Entwicklung hinzugenommen, damit die Anforderungen der Software geeignet und für spätere Phasen wiederverwendbar als Open Source Dokumente festgehalten werden können.

Die Tabelle ist nicht zwangsläufig vollumfänglich in der Entwurfsphase vor dem Start der eigentlichen Entwicklung zu vervollständigen, es sollte jedoch berücksichtigt werden, dass zusätzliche funktionale Anforderungen den Schutzbedarf erhöhen können und eine bis dahin durchgeführte Umsetzung auch diesem erhöhten Schutzbedarf gerecht werden muss. Spätestens bei Abnahme der Software, sollten die Dokumentationen zu Punkt 1 und 2 erstellt und die Matrix der Sicherheitsanforderungen zu Punkt 3 jedoch vollständig ausgefüllt sein, weitere referenzierte Tabellen / Dokumente vorliegen und die darin referenzierten Lösungen/Technologien auch in der Software um-/eingesetzt sein.

3 Implementierungssicherheit

Die Implementierungssicherheit beschreibt die Beachtung der Sicherheit während der Implementierungsphase der Software. Die Implementierung liegt zwischen der vorhergehenden Entwurfsphase und der Auslieferung sowie der anschließenden Inbetriebnahme. Zum Start der Implementierungsphase sollten wesentliche Anforderungen aus der Entwurfsphase (siehe vorheriges Kapitel) bereits detailliert vorliegen, beziehungsweise Zusammenhänge zwischen funktionalen Anforderungen und Sicherheitsanforderungen bekannt sein. Ein wichtiger Baustein bezüglich der Sicherheit von Open Source-Software ist die Sicherheit während der Implementierungsphase. Im folgenden Abschnitt werden zunächst wesentliche Vorgaben aus dem BSI Grundschutz-Kompendium und dem BDEW Whitepaper für die Implementierungssicherheit benannt, die auch für Closed Source-Software beziehungsweise für Eigenentwicklungen gültig sind, unabhängig davon ob diese offen oder geschlossen implementiert werden. Darauf folgend werden die Vorgaben der openKONSEQUENZ e.G. (oK) für konsortiale Open Source-Software-Entwicklung vorgestellt und anschließend die sich ergebenen wesentlichen Vorgaben an die Sicherheit während der Open Source Software-Implementierung zusammengefasst.

3.1 *BSI Grundschutz*

Das BSI Grundschutz-Kompendium (BSI, 2020) bietet neben den im vorherigen Kapitel näher benannten Anforderungen für die Sicherheit in der Entwurfsphase auch für die Sicherheit während der Implementierungsphase Anforderungen. Die wesentlichen Anforderungen für die Implementierungsphase werden im Folgenden genannt.

3.1.1 CON.5 Entwicklung und Einsatz von Individualsoftware

Das BSI Grundschutz-Kompendium führt ein Baustein für die kundenindividuelle Entwicklung von Software zur Lösung von Herausforderungen, die durch Standardsoftware nicht hinreichend abgedeckt sind. Die entsprechenden Anforderungen sollten jedoch auch in Open Source-Implementierungen Anwendung finden. Die hervorzuhebenden Anforderungen während der Implementierungsphase sind:

- CON.5.A3 Sichere Installation von Individualsoftware
- CON.5.A4 Heranführen von Benutzerinnen und Benutzern an Individualsoftware
- CON.5.A6 Dokumentation der Anforderungen an die Individualsoftware
- CON.5.A8 Geeignete Steuerung der Anwendungsentwicklung
- CON.5.A12 Treuhänderische Hinterlegung

Hierbei geht es im Wesentlichen um eine ordentliche Dokumentierung für die Installation, für die Administration und die Anwendung der Individualsoftware. Zudem wird ein geeignetes Steuerungs- und Projektmanagementmodell für die Entwicklung vorausgesetzt. In Open Source Projekten entfällt die Treuhänderische Hinterlegung dokumentierten Codes, da die Ressourcen von vornherein offen zugreifbar sind. Zudem sollte aber auch sichergestellt sein, dass Konstruktionspläne entweder öffentlich gemacht sind oder treuhänderisch hinterlegt werden.

3.1.2 CON.8 Softwareentwicklung

Der BSI-Grundschutz-Baustein Softwareentwicklung konkretisiert den Baustein CON.5 Entwicklung und Einsatz von Individualsoftware hinsichtlich der Eigenentwicklung von Software. Die Anforderungen sind im Wesentlichen durch den Entwickler selbst umzusetzen und gelten dabei sowohl für beauftragte Entwicklungen als auch für Eigenentwicklungen. Auch hier sind sämtliche Anforderungen für die Sicherheit in der Implementierungsphase wichtig. Diese Anforderungen sind

- CON.8.A1 Definition von Rollen und Verantwortlichkeiten
- CON.8.A2 Auswahl eines Vorgehensmodells
- CON.8.A3 Auswahl einer Entwicklungsumgebung
- CON.8.A4 Einhaltung einer sicheren Vorgehensweise
- CON.8.A5 Sicheres Systemdesign

- CON.8.A6 Verwendung von Bibliotheken aus vertrauenswürdigen Quellen
- CON.8.A7 Anwendung von Testverfahren
- CON.8.A8 Bereitstellung von Patches, Updates und Änderungen
- CON.8.A9 Berücksichtigung von Compliance-Anforderungen
- CON.8.A10 Versionsverwaltung des Quellcodes
- CON.8.A11 Erstellung einer Richtlinie für die Software-Entwicklung
- CON.8.A12 Ausführliche Dokumentation
- CON.8.A13 Beschaffung von Werkzeugen
- CON.8.A14 Schulung des Projektteams zur Informationssicherheit
- CON.8.A15 Sicherer Einsatz der Test- und Entwicklungsumgebungen
- CON.8.A16 Geeignete Steuerung der Software-Entwicklung
- CON.8.A17 Auswahl vertrauenswürdiger Entwicklungswerkzeuge
- CON.8.A18 Regelmäßige Sicherheitsaudits für die Entwicklungsumgebung
- CON.8.A19 Regelmäßige Integritätsprüfung der Entwicklungsumgebung

3.1.3 OPS.1.1.3 Patch und Änderungsmanagement

Um Software auf den neuen Stand zu bringen und insbesondere auch zur Schließung auftauchender Sicherheitslücken muss Software auf neue Versionen aktualisiert werden. Hier sollten während der Entwicklungsphase von Software insbesondere die folgenden Anforderungen des BSI Grundschutz-Kompendiums berücksichtigt werden.

- OPS.1.1.3.A9 Test- und Abnahmeverfahren für neue Hard- und Software
- OPS1.1.3.A10 Sicherstellung der Integrität und Authentizität von Softwarepaketen
- OPS1.1.3.A13 Erfolgsmessung von Änderungsanforderungen

3.1.4 OPS.1.1.6 Software-Tests und –Freigaben

In der Entwurfsphase sollte bereits für Software-Tests und -Freigaben die Grundlage gelegt sein. Während der Implementierungsphase nimmt dieser BSI Grundschutz-Baustein jedoch eine wesentliche Rolle zur Feststellung beziehungsweise dem Nachweis der Einhaltung funktionaler sowie nicht-funktionaler Anforderungen ein. Fast alle Anforderungen des Bausteins sind wichtig und werden deshalb nochmal genannt:

- OPS.1.1.6.A1 Planung der Software-Tests
- OPS.1.1.6.A2 Durchführung von funktionalen Software-Tests
- OPS.1.1.6.A3 Auswertung der Testergebnisse
- OPS.1.1.6.A4 Freigabe der Software
- OPS.1.1.6.A5 Durchführung nicht-funktionaler Software-Tests
- OPS.1.1.6.A6 Geordnete Einweisung der Software-Tester
- OPS.1.1.6.A7 Personalauswahl der Software-Tester
- OPS.1.1.6.A8 Fort- und Weiterbildung der Software-Tester
- OPS.1.1.6.A9 Beschaffung von Test-Software
- OPS.1.1.6.A10 Erstellung eines Abnahmeplans
- OPS.1.1.6.A11 Verwendung von anonymisierten oder pseudonymisierten Testdaten
- OPS.1.1.6.A12 Durchführung von Regressionstests
- OPS.1.1.6.A13 Trennung von Test- und Qualitätsmanagement-Umgebung von der Produktivumgebung

Im BSI Grundschutz-Kompendium wird eine externe Qualitätssicherung nicht explizit benannt. Im Optimalfall werden die Tests jedoch durch einen unabhängig von der Implementierung beauftragten Qualitätssicherer durchgeführt und ausgewertet. Insbesondere wird dies durch die Entwicklung als Open Source ermöglicht und stellt einen erheblichen potentiellen Sicherheitszugewinn dar beziehungsweise ein unabhängiges Urteil über den Testausgang sicher.

3.2 *BDEW Whitepaper*

Auch das BDEW Whitepaper (BDEW, 2018), welches im Wesentlichen auf den sicheren Betrieb von Software abzielt, wird neben Sicherheitsanforderungen zur Entwurfsphase auch auf solche aus der Implementierungsphase eingegangen. Diese werden im Folgenden kurz benannt.

Zunächst müssen Standard-Konfigurationen für Wieder- oder Inbetriebnahme einen betriebssicheren Zustand erzeugen und es wird die Prüfbarkeit auf Integrität für Systemdateien, Anwendungen, Konfigurationsdateien und Anwendungs-Parameter verlangt.

Das BDEW Whitepaper enthält Anforderungen an die Dokumentation. Spätestens zum Abschluss der Implementierungsphase erwartet das BDEW Whitepaper eine projektspezifische Dokumentation aller sicherheitsbezogenen Systemeinstellungen, Parameter, Log-/Audit-Meldungen mit passenden Gegenmaßnahmen, Konfigurationseinstellungen, Implementierungsdetails sowie verwendete Verfahren für Einzelkomponenten und das Gesamtsystem sowie die Voraussetzungen für den sicheren Systembetrieb. Zudem sollen die Systemarchitektur und die Interaktion der Komponenten dokumentiert sein sowie getrennte Benutzerhandbücher für Administratoren und Nutzer existieren.

Hinsichtlich der Projektorganisation ist gemäß des BDEW Whitepapers ein Ansprechpartner seitens des Entwicklers zu benennen, der für den Bereich IT-Sicherheit in der System-Entwicklung zuständig ist.

Das BDEW Whitepaper enthält zudem umfangreiche Anforderungen hinsichtlich Sicherheits- und Abnahmetests. Einzelne Komponenten und wesentliche Funktionen des Gesamtsystems sind durch eine vom Entwickler-Team unabhängige Organisationseinheit einem Sicherheits-, Last- und Stresstest zu unterziehen. Der Umfang und die Testtiefe für Abnahme und Funktionsprüfung sollen abhängig von Kritikalität und Komplexität der Software gestaltet sein. Eine Sicherheitsüberprüfung soll die vollständige und effektive Umsetzung der Sicherheitsmaßnahmen prüfen. Zudem sind alle Tests, der Testaufbau und die Ergebnisse zu dokumentieren.

Im BDEW Whitepaper werden Anforderungen in der Software-Entwicklung auch an die Implementierungssicherheit gestellt. Diese sind, der Einsatz zuverlässiger und geschulter Entwickler sowie die Entwicklung anhand anerkannter Vorgehensmodelle, Entwicklungsstandards und Qualitätsmanagement/-sicherungs-Prozessen. Für die Implementierungsphase werden an dieser Stelle besonders eine sichere Programmierung, Anforderungstests sowie Sicherheitsprüfungen vor der Inbetriebnahme genannt. Zudem sind Tests durch andere Personen als die Entwickler durchzuführen und die Testpläne, -prozeduren, sowie erwartete und eingetretene Ergebnisse sollen dokumentiert werden und nachvollziehbar sein. Die Programmierung hat entlang einer Programmierrichtlinie zu erfolgen in der explizit auf sicherheitsrelevante Anforderungen eingegangen wird. Die Freigabe eines Systems / von Updates / Patches muss einem spezifizierten und dokumentierten Freigabe-Prozess folgen.

Darüber hinaus stellt das BDEW Whitepaper auch Anforderungen an Entwicklungs- und Test-Systeme sowie die Integritätsprüfung. Die Entwicklungssysteme müssen sicher sein und nach aktuellem Stand der Technik gehärtet sein, auf aktuellem Stand gepatcht sein und über aktuellen Schutz gegen Schadsoftware verfügen. Die Entwicklungsumgebung als auch der Quellcode und Binärdateien müssen gegen fremden Zugriff geschützt sein und für die letztgenannten beiden muss die Integrität auf unerlaubte Änderungen überprüfbar sein. Entwicklung und Tests dürfen nicht auf Produktivsystemen erfolgen und auf Produktivsystemen darf kein Quellcode gespeichert werden.

3.3 *Konsortiale Open Source Software-Entwicklung von openKONSEQUENZ*

Die openKONSEQUENZ e.G. (oK) ist die Genossenschaft für konsortiale Software-Entwicklung von Open Source-Software für den Betrieb von Energie- und Wassernetzen. Sie setzt sich aus einer Reihe von Organisationen des Verteilnetzbetriebs, Dienstleister, Software-Anbieter und -Entwickler sowie der Wissenschaft zusammen mit der Vision, die im Markt anerkannte Plattform für konsortial entwickelte, offene, modulare und sichere Software zu sein. Die Software-Entwicklung selbst wird von oK nicht selbst durchgeführt. Es werden Software-Unternehmen beauftragt, für oK Open Source-Software zu entwickeln, mit einem Rahmen, den oK vorgibt.

Wird durch die entsprechenden oK-Gremien eine Software-Entwicklung angestoßen so erfolgt ein mehrstufiges, kombiniertes Wasserfall- / V-Modell- / SCRUM-Vorgehen. Zunächst werden grobe funktionale Anforderungen gesammelt. Um die Komplexität einer Gesamtentwicklung zu kapseln, wird nach Möglichkeiten gesucht, funktionale Anforderungen auf mehrere Stufen aufzuteilen, die nacheinander abgearbeitet werden können. Ist dies beispielsweise in eine Basisumsetzung und mehrere anschließende Erweiterungsprojekte möglich, erfolgt die Fokussierung auf die erste / eine Stufe. Das in der Umsetzung erarbeitete Wissen kann dann in die Umsetzung weiterer Projekte oder Stufen wieder einfließen. Vor der Ausschreibung der Entwicklung der Software wird zunächst ein Rahmen für die Software gesteckt. Dieser besteht neben groben Vorgaben für den Funktionsumfang und nicht-funktionalen Anforderungen aus Vorgaben hinsichtlich der Architektur, der Qualität, der Software-Oberfläche, der kaufmännischen sowie die Entwicklung betreffende Prozesse. Bereits vor der Ausschreibung können sich interessierte Auftragnehmer bei der Erstellung des Rahmens beteiligen. Nach Ausschreibung der eigentlichen Software-Entwicklung wird auch die Qualitätssicherung unabhängig von der Entwicklung ausgeschrieben/vergeben. Die QS hilft bei der Sicherstellung, den beschriebenen Rahmen einzuhalten.

Die schlussendliche Entwicklung selbst erfolgt agil mittels SCRUM. Der Ablauf folgt dem in Abbildung 8 vereinfacht dargestellten Prozess, der pro Sprint iterativ durchgeführt werden kann und zu einer schlussendlichen Abnahme durchgeführt werden muss. Dokumente aus vorherigen Arbeitsschritten dienen in allen nachfolgenden Arbeitsschritten als Grundlage. Aus Gründen der Übersichtlichkeit wurden entsprechende Abhängigkeiten in der Darstellung weggelassen. Der Scrum Product Owner wird von Seiten der Netzbetreiber in oK gestellt. Er ist Hauptanforderungsgeber für domänenbezogene funktionale und betriebstechnische Anforderungen und bereitet das Product-Backlog vor. Der Scrum Master wird seitens der Entwicklerorganisation gestellt. Die Sprint-Planung erfolgt durch Product Owner und Entwicklerteam – bei Bedarf mit Unterstützung des Qualitätssicherers. Der Qualitätssicherer hilft dem Product Owner während der Sprint-Abnahme und der schlussendlichen Abnahme bei der Einschätzung der Umsetzung qualitätstechnischer Anforderungen.

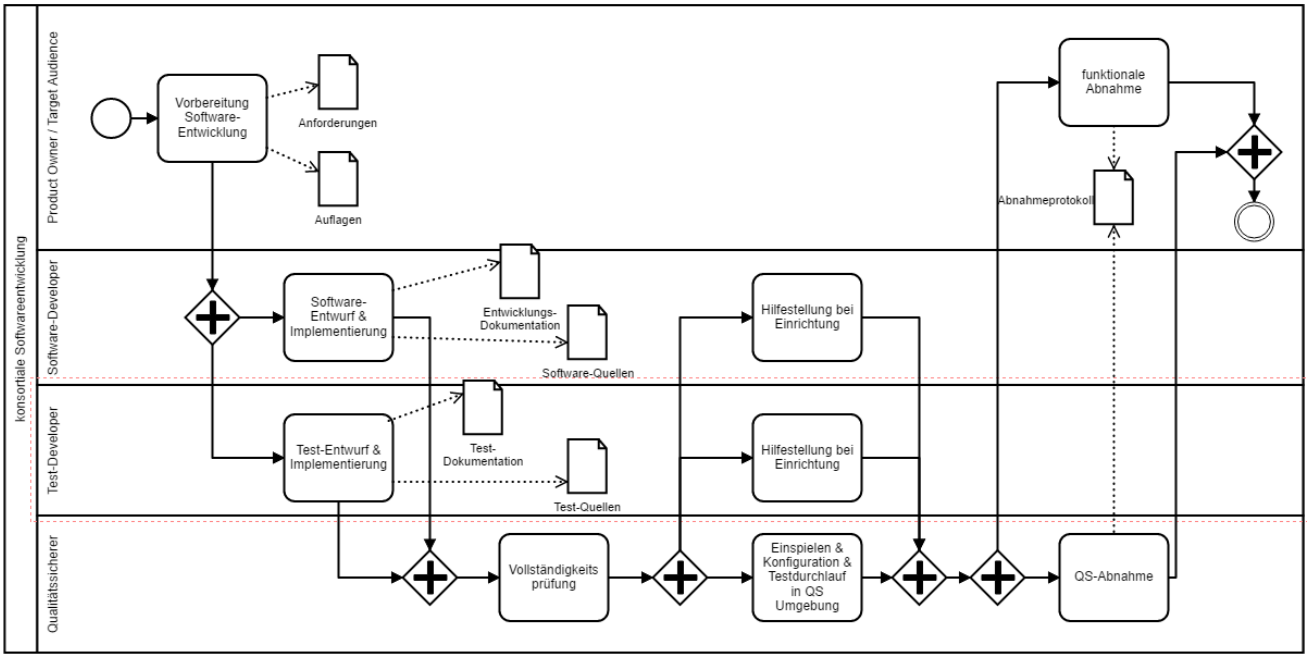


Abbildung 8: Vereinfachte Darstellung des iterativ durchgeführten Entwicklungsprozesses von openKONSEQUENZ für die konsortiale Open Source Software-Entwicklung und Abnahme.

Im Folgenden werden die Handbücher der oK Gremien zu Architektur und Qualität sowie die Entwicklerprozesse & -Tools der von oK zugrunde gelegten Eclipse Foundation Projekte, sowie derzeitig in Entwicklung befindliche Sicherheitsmaßnahmen, des Entwicklungsprojekt-übergreifenden Rahmens vorgestellt, dessen Einhaltung der Qualitätssicherer in oK sicherzustellen hat und der als Richtlinie für die Software-Entwicklung gilt. An dieser Stelle wird der Fokus jedoch auf allgemein anwendbare Anforderungen

in der Open Source-Software Implementierung für das Umfeld von Netzleitsystemen gelegt und oK-Spezifika ausgeblendet. Letztere finden sich in der entsprechend angegebenen Literatur. Der kaufmännische Rahmen wird hier ebenfalls ausgeblendet. Vorgaben zur Benutzungsoberfläche, die auch Teil des Entwicklungsprojekt-übergreifenden Rahmens von oK sind, werden in Kapitel 4 angesprochen.

3.3.1 oK Architekturkomitee-Handbuch

Das Handbuch des oK Architekturkomitees (AC) (oK Architecture Committee, 2019) enthält die für oK wesentlichen Anforderungen an die Software-Architektur der für oK zu entwickelnden Software-Module. Die technischen Entwurfsentscheidungen des Architektur-Handbuchs sind oK spezifisch und für die allgemeine Entwicklung von Open Source Software nicht von Bedeutung und werden deshalb hier ausgeblendet. An dieser Stelle steht der Fokus darauf, wie wesentliche Anforderungen an die Implementierungssicherheit von Open-Source Software-Entwicklungen im Umfeld von Netzleitsystemen umgesetzt werden.

Zum Thema Sicherheit wird im AC Handbuch festgestellt, dass dort ein Sicherheitskonzept noch nicht vollständig spezifiziert ist und dies eine offene Aufgabe des AC ist. Die Anforderung an Sicherheit wird aber allgemein hervorgehoben: Die zu entwickelnde Software muss sicher sein. Zudem wird auf einige Anforderungen der Entwurfsphasensicherheit eingegangen.

Hinsichtlich der BSI-Anforderungen zur *Entwicklung und dem Einsatz von Individualsoftware* werden insbesondere umfangreiche Dokumentationsaufgaben bezüglich der technischen und sicherheitstechnischen Ausprägungen der Software, der Inbetriebnahme / des Betriebs und der Verwendung benannt, SCRUM als Entwicklungsprozess festgehalten und die Ablage sowohl des Quell-Codes als auch der Dokumentationen im öffentlich zugänglichen Open Source Code Repository bestimmt.

Bezüglich des BSI Bausteins *Softwareentwicklung* geht das AC Handbuch darüber hinaus insbesondere auf die Rollen und Verantwortlichkeiten der oK Gremien und der SCRUM Beteiligten, auf die Verwendung von Open Source-Tools für die Dokumentation sowie die Trennung verschiedener Staging-Umgebungen ein.

Letztere Anforderung wird auch im BSI unter *Software-Tests und –Freigaben* explizit benannt. In oK wird zwischen verschiedenen Umgebungen für Entwickler, für die Integration, für Qualitätssicherung, für die Demonstration sowie zwischen Kunden-Umgebungen unterschieden. Nach Fertigstellung der Entwicklung der Software übernimmt ein Integrator kundenindividuelle Anpassungen (falls nötig) und die Inbetriebnahme der Software in die Kundenumgebungen, der zusätzlich dann entsprechend nötige Umgebungen vor der Produktiv-Umgebung des Kunden einzurichten hat. Dies geht jedoch über den Punkt der eigentlichen Open Source-Software-Entwicklung hinaus.

3.3.2 oK Qualitätskomitee-Handbuch

Das Handbuch des oK Qualitätskomitees (oK Quality Committee, 2020) definiert Qualitätsrichtlinien für die Projekte, die im Kontext von oK entwickelt werden. Die Überwachung der Qualität der Codebasis der Software und die Überwachung der Einhaltung der Vorgaben durch das AC Handbuch liegt in der Verantwortung des Qualitätskomitees. Die für die Qualitätssicherung verwendeten Werkzeuge des Qualitäts-Handbuchs sind spezifisch für die von oK gewählte Architektur und nur für technisch ähnlich gelagerte Open Source Software von Bedeutung. Deshalb werden diese hier ausgeblendet – eine entsprechende Beschaffung der Werkzeuge zur Qualitätssicherung sollte jedoch sicher sein und diese sichere Beschaffung allgemein berücksichtigt werden. An dieser Stelle steht der Fokus darauf, wie wesentliche allgemeine Anforderungen an die Implementierungssicherheit von Open-Source Software-Entwicklungen im Umfeld von Netzleitsystemen durch das QC von oK umgesetzt werden.

Hinsichtlich des BSI Bausteins *Entwicklung und Einsatz von Individualsoftware* verlangt das QC Handbuch die öffentliche Ablage der Nutzer-, Administrator-, Entwickler-, Architektur-, Kompilierungs-, Konfigurations-, Einrichtungs-, Test-Anleitungs-, Test-Durchführungs- und Test-Ergebnis-Dokumentationen, Konfigurationsdaten sowie Test-Daten im Open Source-Code-Repository. Zudem werden Prozesse für den Ablauf einer qualitätsgesicherten Steuerung der Anwendungsentwicklung vorgegeben.

Bezüglich des BSI Bausteins zur *Softwareentwicklung* finden sich im QC Handbuch über die vorgenannten Punkte hinaus, die Definition der Rollen und Verantwortlichkeiten der unterschiedlichen oK Gremien und

insbesondere des QC hinsichtlich der Qualität der Software. Die Einhaltung einer sicheren Vorgehensweise wird durch eine gemeinsame Abnahme durch Produkt-Owner, Qualitätssicherer und Entwickler sichergestellt. Zudem erfolgen Einstufungen der Kritikalität und der Komplexität der Software und eine dementsprechende Festlegung der Tiefe manueller Inspektionen, wie sie auch durch das BDEW Whitepaper gefordert wird. Externe Komponenten/Bibliotheken werden durch eine automatisierte Sicherheitsüberprüfung der Abhängigkeiten bei jedem Build-Prozess in der Integrations- beziehungsweise Qualitätssicherungsumgebung geprüft. Zur Sicherstellung langfristig sicher verwendbarer externer Bibliotheken kann ein regelmäßiger Bauvorgang angestoßen werden, entsprechende Log-Files gesichtet und auf Sicherheitslücken der verwendeten externen Bibliotheken eingegangen werden. Für die Anwendung von Testverfahren sind Werkzeuge auf verschiedenen Ebenen einzusetzen, die beispielsweise die Einhaltung von Kodierungsrichtlinien, die statische Code-Analyse die technische Qualität von Quellcode, aber auch den automatisierten Zusammenbau einzelner Komponenten zu einem Anwendungsprogramm durchführen beziehungsweise überwachen. Zudem sind regelmäßige Unit Tests mit einem festgelegten beziehungsweise transparent festzulegenden Mindestmaß an Zweigüberdeckung als auch der Code-Abdeckung durchführbar und zu protokollieren. Ein wesentlicher Baustein der Tests stellen Tests der technischen Lösung gegenüber den funktionalen und qualitativen Anforderungen dar. Diese Tests sind entsprechend zu definieren und deren Anforderungen und Akzeptanzkriterien zu benennen. Einzubindende Test-Daten und Mock-Funktionalität sind zu spezifizieren und falls möglich soweit zu formalisieren, dass Test automatisiert durchgeführt werden können oder als Arbeitsanweisungen für manuelle Tests vorliegen. Einen Teil der Richtlinien zur Softwareentwicklung stellt das QC Handbuch selbst dar und verweist auf weitere Richtlinien wie beispielsweise Codierungsrichtlinien zu unterschiedlichen Programmiersprachen und Datenformaten.

Die Ausführungen zu Tests beziehen sich insbesondere auch dementsprechend auf die BSI Bausteine zum *Patch und Änderungsmagement* sowie zu *Software-Tests und –Freigaben.* Zu letzterem wird insbesondere auch wie im AC Handbuch zwischen verschiedenen Entwicklungs-, Integrations- und Qualitätssicherungsumgebungen unterschieden, die sämtlich vor den Test- und Produktivumgebungen der Anwender liegen. Da die Qualitätssicherung in oK durch eine Entwicklerfirma-externe Person/Organisation durchgeführt wird und aufgrund der Vorgaben zur Veröffentlichung von Quellcode und Dokumenten auch durchgeführt werden kann, kann von einer größeren Unabhängigkeit zwischen Qualitätssicherer und Entwickler ausgegangen werden und eine mindestens gleich hohe, wenn nicht gar höhere Verlässlichkeit der externen Qualitätsprüfung gegenüber einer internen Qualitätsprüfung erwartet werden. Zudem werden dementsprechend auch Anforderungen des BDEW an die Unabhängigkeit des Test-Durchführenden Personals gegenüber dem Entwicklerteam erfüllt.

3.3.3 oK Eclipse Foundation Project

Projekte von openKONSEQUENZ werden als Eclipse Foundation Projekte unter Eclipse Public License (EPL) (1.0 beziehungsweise 2.0) entwickelt. Für ihre Projekte stellt die Eclipse verschiedene Dienstleistungen und Regeln zur Verfügung, die im Folgenden kurz vorgestellt werden.

3.3.3.1 IP Check

Ein Intelectual Property (IP) Check soll vermeiden, dass Besitzrechte Dritter (Entwickler) durch Übernahme oder Einbindung von Quellcode oder ungeeigneter Bibliotheken in ein Software Projekt verletzt werden und so sicherstellen, dass die (Weiter-)Entwicklung und Verwendung der geprüften Software nicht durch die Ausübung des Urheberrechts oder Besitzrechts gefährdet sind.

Der Eclipse Intelectual Property (IP) Prozess unterstützt Entwickler bei der Sicherstellung, dass das Projekt kompatibel mit der EPL ist (Wittek, 2018). Dies umfasst die vertragliche Bindung von Committer an Sorgfaltspflichten, regelmäßig durchgeführte IP Checks für das Software Projekt, IP Checks für Bibliotheken Dritter (für die eigentliche Software und für die Test & Build-Software) sowie die Überprüfung der rechtlichen Dokumentation (unter anderem der EPL, Copyright-Vermerk).

3.3.3.2 Eclipse Entwicklerwerkzeuge

Für die Entwicklung von Software Projekten stellt die Eclipse für Eclipse Foundation Projekte eine Infrastruktur zur Verfügung. Diese umfasst neben Projekt-Homepages Werkzeuge wie Quellcode-Repositories, kollaborative Review-Systeme für Quellcode-Repositories, Bug-/Feature-Tracker für die Verwaltung von Fehlermeldungen und Erweiterungen, ein Verwaltungstool für IP-Anfragen sowie ein Werkzeug zur kontinuierlichen Integration.

3.3.3.3 Committing- / Committer-Prozess

In Eclipse Projekten werden bei Projektinitiierung Committer bestimmt. Nur diese haben schreibenden Zugriff auf die Code-Repositories des Projekts (Eclipse, 2020). Nach Projektinitiierung gibt es einen mehrphasigen Prozess, bei dem Dritte sich auf unterschiedlichen Ebenen an der Projektarbeit beteiligen können. Werden die Arbeiten durch die bisherigen Committer als sinnvoll erachtet, können diese auf Wunsch eine Wahl anstoßen, um den Externen in den Committer-Kreis aufzunehmen. So wird ausgeschlossen, dass fremde Personen Inhalte der Repositories direkt manipulieren können und so kann sichergestellt werden, dass nur Personen, die sich gut mit der Software auskennen auf die Software-Quellen zugreifen können.

Für das Committing selbst, gibt es ein Review-System für Quellcode-Repositories. Hier können auch externe zum Code beitragen und Änderungsvorschläge machen. Jeder solcher Beitrag muss aber durch einen Committer innerhalb des Projekts freigegeben werden. Halten Committer ihre dementsprechenden Sorgfaltspflichten ein, sollte dadurch kein Schadcode eingefügt werden. Da Committer selbst unter Umständen an dem Review-System vorbei direkt auf den Quellen arbeiten können ergibt sich hier ein Risiko, falls sich Committer vom Projekt abwenden oder fahrlässig handeln. Im Besten Fall wird der direkte Zugriff von Committern auf das Repository jedoch nicht zugelassen und auch Committer-Beiträge müssen wie die von Externen durch einen zweiten oder mehrere weitere Committer geprüft werden, bevor sie in die Code-Basis gelangen.

3.3.4 Derzeitig in Bearbeitung befindliche sicherheitsbezogene oK-Themen

In openKONSEQUENZ wird derzeit ein Rahmen für die Softwarepflege festgelegt. Nach Abschluss der Entwicklung eines oK-Moduls ist ein Entwicklerunternehmen nur noch in der gesetzlichen Gewährleistung. Ein darüber hinausgehender Support ist in Diskussion, der auftretende Bugs und Sicherheitslücken fixt und Sicherheits-Patches externer Bibliotheken zeitnah in den Code eingespielt, sodass daraus Updates und Patches erzeugt werden können und so die grundlegende Verwendbarkeit der Software aufrecht erhalten wird. Für die Umsetzung umfangreicher Feature-Requests ist eine neue Beauftragung angedacht, wie sie auch zur Softwareentwicklung selbst erfolgt und dementsprechenden Prozessen folgen.

Die Bereitstellung von Software / Patches / Updates / Änderungen erfolgt derzeit auf Ebene des Quellcodes. Das heißt ein Anwenderunternehmen oder dessen Integrator bezieht den Quellcode aus den Eclipse-Repositories und baut die Software selbst anhand der mitgelieferten Anleitung. Da zu diesem Zweck sämtliche Entwickler- und Testwerkzeuge vorhanden seien müssen, ist dies ein aufwändiger Prozess. Zudem ist die Expertise zu Beurteilung der individuellen Tests möglicherweise beim Endanwender oder Integrator nicht vorhanden. Um diesen Aufwand für Anwender zu verringern sollen zukünftig vorgebaute Software-Bausteine sowie eingerichtete Container zur Verfügung gestellt werden. Für diese können Tests und Abnahmeverfahren sowie die Erfolgsmessung von Änderungsmanagement wie gewohnt durch den Qualitätssicherungsexperten, den Product-Owner und das Entwicklerteam durchgeführt werden, sodass die grundlegende Funktionalität sichergestellt ist. Hierzu soll ein zentraler Dienst eingerichtet werden, der alle verwendeten externen Bibliotheken und Komponenten hostet, und bei Änderungen der Codebasis einen neuen Bauvorgang anstößt, die entsprechende Software testet und bei Erfolg die vorgebauten Software-Bausteine selbst sowie eingebettet und konfiguriert in lauffähige Container zur Verfügung stellt. Durch die Hinzunahme der externen Komponenten in den zentralen Dienst, werden diese nicht mehr von extern abgeholt und zwischenzeitige Manipulationen an externen Komponenten werden nicht automatisch mit eingespielt und kompromittieren dadurch die Software nicht. Für die Aufnahme dieser externen Komponenten muss jedoch sichergestellt sein, dass diese aus vertrauenswürdigen Quellen stammen und dass sie integer und authentisch sind. Hier muss für eine externe Komponente nur einmalig eine vertrauenswürdige Quelle gefunden werden, und die Integrität und Authentizität beispielsweise mittels Checksummen und Zertifikaten überprüft werden. Die

Vertrauenswürdigkeit muss nicht zu jedem Bau innerhalb des zentralen Dienstes erneut geprüft werden. Für jeglichen Zugriff von außerhalb auf den zentralen Dienst – sei es für den Zugriff auf dort hinterlegte externe Bibiotheken zum externen zusammenbauen von Software oder für den Zugriff auf fertig gebaute Bausteine oder Container – sollten jedoch Checksummen und Zertifikate bereitgestellt werden, damit die Vertrauenswürdigkeit geprüft werden kann.

3.3.5 Offene Sicherheitsanforderungen zur Entwicklungsumgebung in oK

Sowohl das BSI Grundschutzkompendium als auch das BDEW Whiterpaper benennen die Auswahl und Überprüfung der Entwicklungsumgebung als sicherheitskritische Anforderungen für die Entwicklung von Software und detaillieren diese Anforderungen weiter aus. Da durch oK die Entwicklung der Software externalisiert wird hat der Entwickler / die Entwicklerorganisation die Einhaltung der entsprechenden Anforderungen sicherzustellen. Entsprechende Auditierungsergebnisse sind sicherlich auch sinnvoll bei der Auftragsvergabe durch oK zu berücksichtigen. Die Entwicklung der Open Source Software endet nach dem abschließenden Bau und erfolgreichem Test auf der Qualitätssicherungsumgebung mit der Auslieferung des Quellcodes auf das Eclipse-Repository. Verwenden Anwender oder Integratoren die entsprechenden Quellen, so haben sie für ihre eigenen Entwicklungsumgebungen auch die gleichen Sicherheitsanforderungen einzuhalten. Dies geht aber über die Entwicklung der eigentlichen Open Source-Anwendung im oK Kontext hinaus.

3.4 *Zusammenfassung Implementierungssicherheit*

An die Implementierungssicherheit von Open Source-Software im Umfeld von Netzleitsystemem sind, unter der Prämisse Security-By-Design und nicht Security-By-Obscurity, die gleichen Ansprüche wie für entsprechende Closed Source-Umsetzungen zu stellen. Aus Überschneidung der genannten BSI und BDEW Anforderungen können folgende wesentlichen Punkte für die Implementierungssicherheit von Open Source Software für den Einsatz im Umfeld von Netzleitsystemen festgehalten werden:

1. Open Source Software-Entwicklung
    a. *Festlegung einer geeigneten Open Source Entwickler-Plattform (s.u.)*
    b. *Festlegung einer geeigneten Open Source Software-Lizenz und Überprüfung der Einhaltung (s.u.)*
    c. Definition von Rollen und Verantwortlichkeiten
    d. Auswahl eines Vorgehensmodells
    e. Einhaltung einer sicheren Vorgehensweise
    f. Sicheres Systemdesign
    g. Verwendung von Bibliotheken aus vertrauenswürdigen Quellen
    h. Anwendung von Testverfahren
    i. Bereitstellung von Patches, Updates und Änderungen
    j. Sicherstellung der Integrität und Authentizität von Softwarepaketen
    k. Berücksichtigung von Compliance-Anforderungen
    l. Versionsverwaltung des Quellcodes
    m. Erstellung einer Richtlinie für die Software-Entwicklung
    n. Ausführliche, öffentlich zugängliche Dokumentation
    o. Schulung des Projektteams zur Informationssicherheit
    p. Installation von Software mit sicherer initialer Konfiguration
    q. Open Source Quellablage
    r. Auswahl vertrauenswürdiger Entwicklungswerkzeuge / Entwicklungsumgebung
    s. Regelmäßige Sicherheitsaudits für die Entwicklungsumgebung
    t. Regelmäßige Integritätsprüfung der Entwicklungsumgebung
    u. Sicherer Einsatz der Test- und Entwicklungsumgebungen
2. Open Source Software-Tests und -Freigaben
    a. *Einordnung von Kritikalität, Komplexität und Festlegung der Inspektionstiefe (s.u.)*
    b. Planung der Software-Tests

c. Durchführung von funktionalen Software-Tests
d. Auswertung der Testergebnisse
e. Freigabe der Software
f. Durchführung nicht-funktionaler Software-Tests
g. Geordnete Einweisung der Software-Tester
h. Personalauswahl der Software-Tester
i. Fort- und Weiterbildung der Software-Tester
j. Beschaffung von Test-Software
k. Erstellung eines Abnahmeplans
l. Verwendung von anonymisierten oder pseudonymisierten Testdaten
m. Durchführung von Regressionstests
n. Trennung von Test- und Qualitätsmanagement-Umgebung von der Produktivumgebung

Für einige Anforderungen reicht es, Regeln einmalig aufzustellen, die Einhaltung wäre dann jedoch wie die weiteren Anforderungen bei Abnahme zu überprüfen.

Hierbei wurden die oben genannten Anforderungen des BSI zur *Entwicklung und Einsatz von Individual-Software* zur *Software-Entwicklung* und zum *Patch und Änderungsmanagement* unter dem Titel *OS Software-Entwicklung* verschränkt und die Anforderungen des BSI zu *Software-Tests und Freigaben* übernommen.

Für die Entwicklung von Open Source Software im Umfeld von Netzleitsystemen ergeben sich sowohl aus der Quellverwaltung als Open Source als auch dem Umfeld besondere Herausforderungen, die im Folgenden hervorgehoben werden.

In der kritischen Infrastruktur Energieversorung muss gemäß BDEW Whitepaper für Software im Umfeld von Netzleitsystemen die Kritikalität, Komplexität der Software mit berücksichtigt werden bei der Tiefe der Inspektion. Dies wurde in die obige Checkliste hinsichtlich der Open Source Software-Tests und -Freigaben mit aufgenommen.

Für die konsortiale Open Source Software-Entwicklung wurde openKONSEQUENZ als Beispiel einer adäquaten Vorgehensweise benannt, bei der eine Entwicklerfirma beauftragt wird Open Source zu entwickeln und dementsprechend auch die Entwickler-Firma entwicklerübergreifend Regeln festlegen kann, die ein extern beauftragter Qualitätssicherer überprüfen kann. Dies ist im Open Source Umfeld sicherlich ungewöhnlich, da sich im Allgemeinen auch unterschiedliche Partner(-Firmen) an der Entwicklung beteiligen können. Es gibt also viele weitere Arten von Open Source Software-Entwicklung, bei denen die genannten Anforderungen anders umzusetzen sind, damit BSI- / BDEW-konform implementiert werden kann.

Als Beispiel dient nun die „Extreme“, dass völlig unabhängige Entwickler auf freiwilliger Basis zu einem Projekt beitragen, welches anschließend Dritte verwenden können. Hier gibt es insbesondere folgende Problemstellungen für Zielanwender mit möglichen, kaskadierenden Lösungsansätzen:

Problemgruppe 1 – Einbindung möglicherweise infizierter Code:

- Die Auswahl der Entwicklungswerkzeuge ist individuell, Entwickler-Systeme sind nicht gehärtet, werden nicht auf Integrität überprüft und nicht auditiert.
- Die Sicherstellung der Verwendung von Bibliotheken aus verlässlichen Quellen, der Integrität und der Authentizität für eingesetzte und für von Entwicklern ausgelieferte gebaute Software, Patches/Updates/Änderungen/Dokumentation ist nicht möglich.

Lösungsansatz: Diese Problemstellungen sind möglicherweise zu umgehen, wenn von den Entwicklern nur noch zu verifizierender Quellcode beigetragen wird, der noch nicht kompiliert ist, so dass keine Schadsoftware eingebunden / veröffentlicht wird.

Problemgruppe 2 – Mangelnde Qualität von Beiträgen / ungeprüfte Beiträge

- Es werden fahrlässig oder bewusst Sicherheitslücken eingespielt.
- Verwendete Bibliotheken sind nicht konform zur gewählten Lizenz.
- Ausführliche Dokumentationen werden nicht erstellt.

- Abnahmepläne sind nicht ausreichend erweitert vorhanden.
- Es werden keine annonymisierten und pseudonymisierten Testdaten zur Verfügung gestellt.
- Ein sicheres Systemdesign wird nicht eingehalten.

Lösungsansatz: Diese Problemstellungen sind möglicherweise zu lösen, wenn alle Beiträge ohne Kompilierung eingestellt werden und jegliche Beiträge in einem Review-System einzuspielen sind. Diese Beiträge können dann von einem oder gar mehreren im Projekt etablierten, weiteren Committern **ohne Bias** unter Aufrechterhaltung von Sorgfaltspflichten auf Sicherheit sowie Vollständigkeit geprüft und getestet und nur bei Bestehen aufgenommen und das entsprechende Vorgehen dokumentiert werden. Das Review-System sollte zusätzlich den Code erst dann in die Code-Basis aufnehmen, wenn automatisierte Tests erfolgreich verlaufen (siehe auch folgender Punkt). Zur Code-Basis dürfte niemand an dem Review-System vorbei direkten schreibenden Zugriff haben.

Problemgruppe 3 – Fehlende offizielle End- und Zwischenabnahmen:

- Die Durchführung von (nicht-funktionalen und funktionalen) Tests erfolgt nicht vollständig oder die Auswertung der Testergebnisse erfolgt nicht adäquat.
- Regressionstests/Testverfahren werden nicht oder nur teilweise und mit unterschiedlichen Tools durchgeführt, sodass Ergebnisse nicht vergleichbar sind.
- Trennung von Test- und Qualitätsmanagement-Umgebung wird nicht eingehalten.

Lösungsansatz: Diese Problemstellungen sind unter Umständen zu lösen, wenn ein vom Entwickler und dessen Umgebung unabhängiges Test-System etabliert wird, in dem die Software gebaut, verteilt, soweit möglich automatisiert Tests durchgeführt werden, als auch durch einen Qualitätssicherer in einer externen Qualitätssicherungsumgebung geprüft wird. Der Test wäre adäquat zu dokumentieren. Dieser Qualitätssicherer könnte ein weiterer Committer ohne Bias sein oder ein beauftragter und trotzdem notwendigerweise unabhängiger Dritter sein.

Problemgruppe 4 – Weiteres:

- Nötige Sicherheitsupdates werden nicht umgesetzt und nicht kommuniziert.

  Lösungsansatz: Dieses Problem könnte durch Bereitstellung eines Meldesystems für sicherheitskritische Meldungen mit regelmäßiger automatisierter Überprüfung auf eingehende sicherheitskritische Meldungen hinsichtlich des selbst entwickelten Software-Bestandteils gelöst werden. Zudem muss eine regelmäßige Überprüfung der Sicherheit der verwendeten externen Bibliotheken und Komponenten erfolgen. Diese Überprüfung sollte im bestenfalls automatisiert bei jedem Bau der Software durchgeführt werden. Der Bau sollte bei Fehlschlagen der Prüfung unterbrochen und der Fehlschlag öffentlich dokumentiert werden. Wird der Bau mit Überprüfung regelmäßig auch zwischen neuen Releases der Software durchgeführt, können neu erkannte Fehler verwendeter Komponenten auch als Sicherheitslücken der Software festgestellt werden und geeignete Maßnahmen eingeleitet werden.
- Verwendete Bibliotheken sind nicht aus vertrauenswürdigen Quellen und / oder nicht auf Integrität und Authentizität geprüft.

  Lösungsansatz: Verwendete Bibliotheken werden mit fester Versionsnummer einmalig aus verlässlichen Quellen abgeholt und auf Integrität sowie Authentizität geprüft in einem lokalen Repository gespiegelt und nicht für jeden Bau neu eingesammelt.
- Zur Verfügung gestellte ausführbare Daten kommen von Entwicklungsumgebungen und -systeme die nicht gehärtet und nicht dem Stand der Technik nach gesichert sind, werden nicht auditiert und sind nicht auf Integrität und Autentizität überprüfbar.

  Lösungsansatz: Eine zentrale, gehärtete und mit dem Stand der Technik gesicherte und dementsprechend auditierte Umgebung übernimmt den Bau der Software und erzeugt Artefakte zur Integritäts- und Autentitätsüberprüfung für Anwender, die das Software-Produkt aus dem sicheren System beziehen können.

Viele dieser Probleme für die Software-Implementierung im Bereich des kritischen Infrastruktur Energiesysteme können also mit der Bereitstellung beziehungsweise der Auswahl einer geeigneten Plattform technischer Hilfsmittel und strengen Regeln/Prozessen für die Open Source Software Entwicklung gelöst werden. Deshalb wurde der Punkt *„Festlegung einer geeigneten Open Source Entwickler-Plattform“* in der Checkliste zur Open Source Software-Entwicklung mit aufgenommen.
Für eine Entwicklung von Open Source ist für den Quellcode zudem spätestens zu Beginn der Implementierungsphase festzulegen unter welcher Open Source Lizenz er erstellt wird, damit die Implementierung, Teile davon oder weitere Artefakte wie Dokumentationen nicht später für das Entwicklungs-Projekt wegfallen, wenn beispielsweise Urheber das Projekt verlassen. Damit einhergeht, dass nur zu der gewählten Lizenz kompatible Bibliotheken, Quellen und Komponenten verwendet werden und dieses während des Entwicklungsprozesses sicherzustellen ist. Letztere Problematik wurde auch im Problempunkt 2 angesprochen und ein Lösungsansatz aufgezeigt. Dieser Punkt ist als *„Festlegung einer geeigneten Open Source Software-Lizenz und Überprüfung der Einhaltung“* in der Checkliste zur Open Source Software-Entwicklung aufgenommen.
Es bleiben aber auch offene Punkte, beispielsweise dass Schulungen des Projektteams / Qualitätssicherer zur Informationssicherheit nicht für Open Source Projekte sicherzustellen sind. Hier kann nur eine adäquate Dokumentation während der Entwicklung und der Tests zeigen, wie vertieft das Thema Informationssicherheit berücksichtigt wurde. Die schlussendliche Auswertung der Erfüllung aller oben genannten Anforderungen obliegt jedoch dem Anwender selbst.

Auch schwimmende Rollengrenzen und Verantwortlichkeiten können zu Problemen führen. Entwickler können nicht gezwungen werden, sich weiter zu beteiligen oder sich an (zeitliche) Vorgaben zu halten. Es gibt gegebenenfalls keine festen Abnahmetermine und keinen oder keinen festen Qualitätssicherer ode es stehen keine dedizierten Software-Tester zur Verfügung. Hier kann eine Beteiligung der Anwender(-unternehmen) bei der Entwicklung als Committer und Tester Abhilfe schaffen.

Vollständig kann eine qualitätssichernde Überprüfung dem Endanwender für allgemeine Open Source Software nicht abgenommen werden, solange es zwischen Entwicklung und Anwendung keine (unabhängige) Instanz gibt, die die Sicherheit zertifiziert.

4 Benutzungsschnittstellenvorgaben

Bei der Gestaltung von Benutzungs- bzw. Mensch-Maschine-Schnittstellen können Empfehlungen sowohl hinsichtlich des Prozesses als auch des Ergebnisses/Produktes getroffen werden. Die Ausführungen basieren maßgeblich auf der DIN 9241-Normenfamilie sowie u.a. den Arbeiten von Herczeg (Herczeg, 2014), Hollnagel & Woods (Hollnagel & Woods, 2005), Mentler (Mentler, 2018), Norman (Norman, 1995) und Reason (Reason, 1990).

Die Entwicklung von Benutzungsschnittstellen setzt ein umfassendes Verständnis der Nutzenden, ihrer Aufgaben, Ziele sowie der sozialen und technischen Rahmenbedingungen voraus. Ziel muss es sein, sicherheitsrelevante Vorkommnisse im Zusammenwirken *„von Mensch, Technik, Team, Organisation, Gesetzgebung und Kultur“* (Herczeg, 2014) zu vermeiden bzw. Ihre Auswirkungen zu begrenzen. Ausgehend von der Analyse dieser Aspekte sind Benutzungsschnittstellen iterativ und auf Grundlage von Nutzerfeedback weiterzuentwickeln. Hierbei sind insbesondere Sicherheitsmechanismen und sicherheitsrelevante Teilfunktionalitäten auf Verständnis und praktische Anwendbarkeit durch die späteren Nutzenden zu prüfen. Andernfalls besteht die Gefahr der Entwicklung von Lösungen, die zwar aus rein technischer Sicht zuverlässig erscheinen, aber im zuvor genannten und letztlich entscheidenden Zusammenspiel von Menschen, Technik und Organisation als unbrauchbar eingestuft werden müssen. Eine dann spätere, viel zu einfache und eindimensionale Bewertung als „Menschlicher Fehler“/ „Menschliches Versagen“ hilft im Endeffekt weder den Nutzenden und Systembetreibern noch den Betroffenen. Vielmehr sollte das Konzept „Menschlicher Fehler“ / „Menschliches Versagen“ oftmals zugunsten einer ganzheitlichen Betrachtung von Interaktions- und Organisationsversagen aufgegeben werden. Dies gilt umso mehr, da bereits seit Jahrzehnten ein umfassendes Verständnis unsicherer menschlicher Handlungen vorliegt (siehe z.B. Abbildung 9).

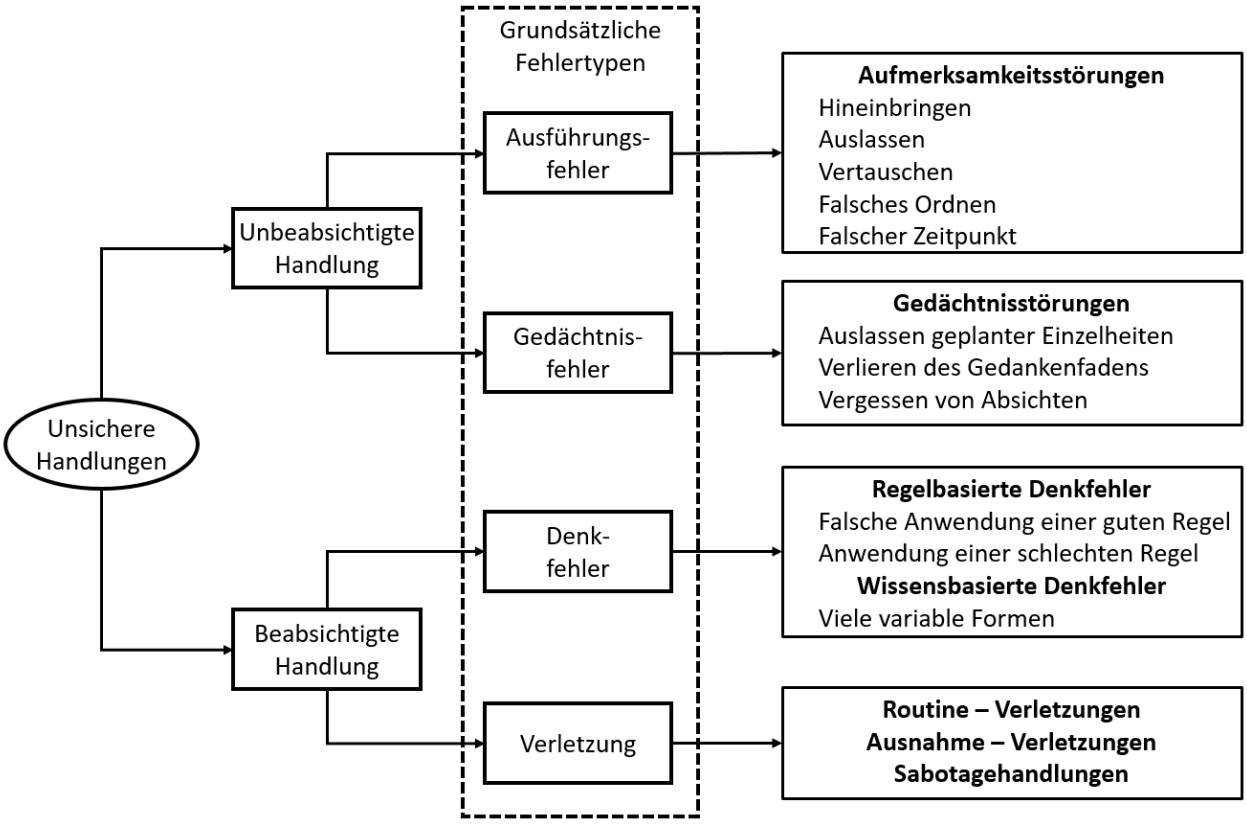


Abbildung 9: Fehlertypen (nach (Reason, 1990))

Mit dem definierten Maß der Gebrauchstauglichkeit und seinen Teilaspekten Effektivität, Effizienz und Zufriedenstellung der Nutzenden lassen sich Benutzungsschnittstellen und insbesondere ihre sicherheitsrelevanten Bereiche unabhängig von persönlichen Urteilen der an der Entwicklung beteiligten prüfen. Weiterhin ist darauf zu achten, dass Benutzungsschnittstellen nicht weitestgehend losgelöst oder

nachgelagert zur Realisierung von Sicherheitsmechanismen auf anderen (tieferen) Systemschichten erfolgt. Visualisierungslösungen, Konfigurationsmöglichkeiten und Hilfefunktionen sind für ein erfolgreiches Sicherheitskonzept unabdingbar und müssen technische Zusammenhänge in geeigneter Form widerspiegeln. Dabei sind komplexe Transformations- und Deformationsprozesse zu beachten (siehe Abbildung 10).

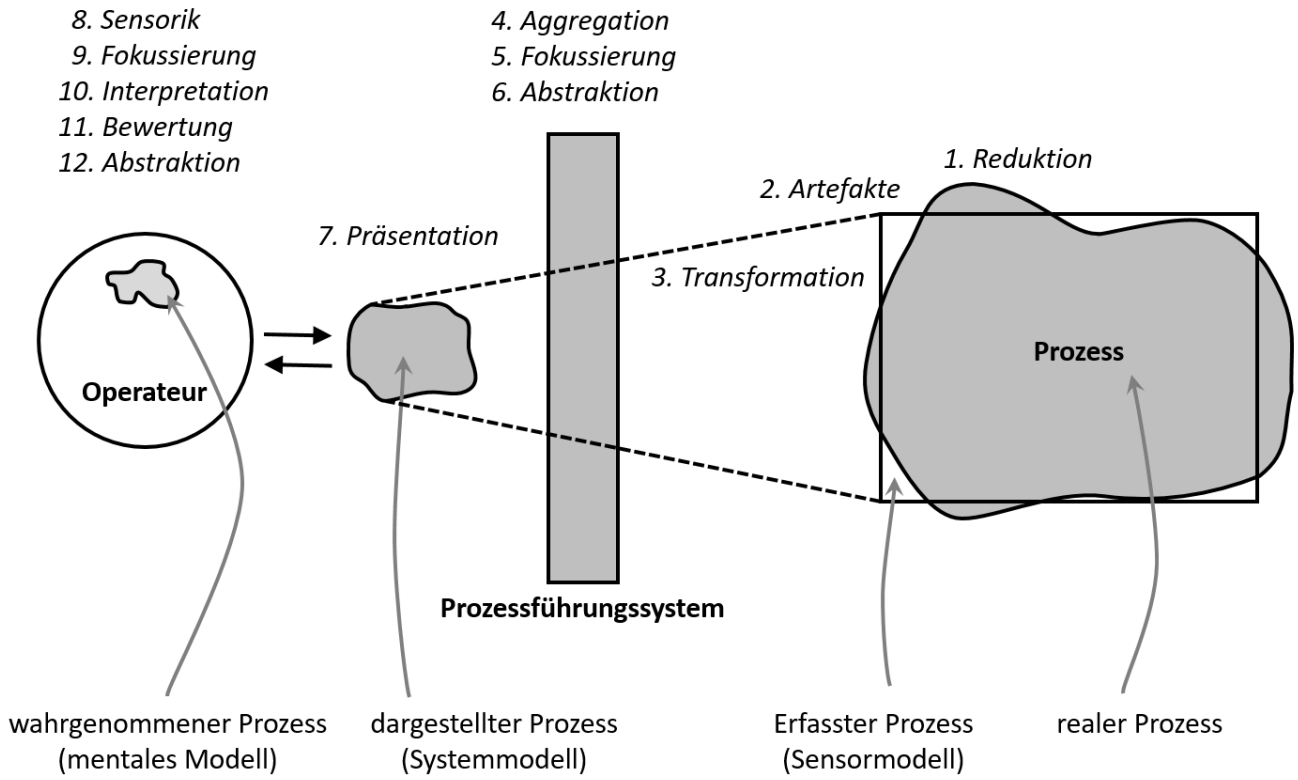


Abbildung 10: Transformation und Deformation des zu überwachenden und zu steuernden Prozesses auf das mentale Modell des Operateurs (nach (Herczeg, 2000 in (Herczeg, 2014)))

Neben prozessorientierten Empfehlungen für die Entwicklung konkreter sicherheitsrelevanter Benutzungsschnittstellen können auch allgemeine gestalterische Prinzipien benannt werden, deren Berücksichtigung zur Systemsicherheit beitragen. Hier sind insbesondere Fehlertoleranz und Konsistenz zu nennen.

Eine Benutzungsschnittstelle ist dabei laut DIN EN ISO 9241-110 fehlertolerant, wenn *„das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand durch den Benutzer erreicht werden kann“*. Es kann zwischen Ansätzen zur Fehlervermeidung, zur Fehlerkorrektur sowie zum Fehlermanagement unterschieden werden. Die Auswahl von Werten aus einem in Hinblick auf die Sicherheit zulässigen Wertebereich statt freier Eingaben sowie die Formulierung von aussagekräftigen Fehlermeldungen, idealerweise mit Möglichkeiten zur unmittelbaren Korrektur, stellen dabei nur zwei von vielen Möglichkeiten dar, Fehlertoleranz zu realisieren.

Neben der Fehlertoleranz stellt Konsistenz, im Sinne *„gleiche Information wird innerhalb der Anwendung entsprechend den Erwartungen des Benutzers stets auf gleiche Art dargestellt“* (DIN EN ISO 9241-12), ein wichtiges Gestaltungskriterium dar. Zu ihrer besseren Gewährleistung empfiehlt es sich einen Styleguide zu pflegen, der wesentliche Gestaltungsentscheidungen für alle an der Entwicklung Beteiligten nachvollziehbar macht. Styleguides können von einfachen Dokumenten mit Farbtabellen etc. bis hin zu eigenständigen interaktiven Systemen reichen, die die Einbindung von auf Konsistenz geprüften Entwicklungsbausteinen in eine Entwicklungsumgebung ermöglichen.

Dementsprechend ergeben sich folgende wesentliche Anforderungen für die Entwicklung sicherer Benutzungsschnittstellen:

1. Verwendung eines definierten Prozesses, um Benutzungsschnittstellen unter Beachtung soziotechnischer Randbedingungen iterativ zu entwickeln.
2. Technische Zusammenhänge in geeigneter Form widerspiegeln
    - Visualisierungslösungen
    - Konfigurationsmöglichkeiten
    - Hilfefunktionen
3. Gebrauchstauglichkeit
    - Effektivität
    - Effizienz
    - Zufriedenstellung der Nutzenden
4. Gestalterische Prinzipien
    - Fehlertoleranz
    - Konsistenz (ggf. nach einem Styleguide)

5 Interoperabilitätsnachweis

Für den Einsatz einer Software ist es wichtig, dass deren Schnittstellen zu externen Systemen nicht nur sicher sind, sondern Daten auch korrekt austauschen und der nötigen Funktion entsprechen, wo Interoperabilität nötig ist. Dabei wird im Folgenden davon ausgegangen, dass die Schnittstellen gemäß der Entwurfs- und Implementierungsphasensicherheit abgesichert sind. Eine Eigen- oder beauftragte Implementierung zur Anpassung von Schnittstellen einer Software an die existierende Systemlandschaft ist indivduell, aufwändig und fehleranfällig und sollte deshalb vermieden werden. Dies würde über die Zeit zu kontinuierlichen Kosten und einem möglicherweise unüberschaubaren Wildwuchs an proprietären Schnittstellen führen.

Zum Nachweis der Interoperabilität von Schnittstellen wurde in der Domäne Healthcare ein sogenannter Connectathon eingeführt – eine Wortneuschöpfung aus den englischen Wörtern für Verbinden (Connect) und Marathon (IHE, 2013). Die Methodik wurde bereits auf die Energiedomäne übertragen (Franzl, Pasteka, & Gottschalk, 2019) und wird im Folgenden vorgestellt.

Der Connectathon ist ein regelmäßig stattfindendes Event, an dem Hersteller von Software ihrer Domäne einen festgelegten Zeitraum zusammenkommen, um die Interoperabiltiät ihrer Software zu Fremdsoftware zu testen und weiterzuentwickeln. Dazu wird das Zusammenspiel von Software verschiedener Hersteller zu vorab definierten, öffentlich zugänglichen, sogenannten Integrationsprofilen und Tests mithilfe eines zentralen Test-Werkzeuges festgestellt. Der Connectathon hat hierbei mehrere Ziele: Neue Interoperabiltätsbedürfnisse festzustellen, Interpretationsspielräume von unreifen Integrationsprofilen zu beseitigen und implementierte Schnittstellen dementsprechend testweise anzupassen, um den Reifegrad von Integrationsprofilen zu erhöhen und für reife Integrationsprofile, eine Interoperabilität verschiedener Systeme nachzuweisen.

Zur Erstellung eines Integrationsprofils wird zunächst durch die Experten in der Anwendungsdomäne der Interoperabilitätsbedarf festgestellt, indem die Interaktion zwischen Akteuren zu einer betriebswirtschaftlichen Funktion beschrieben wird. Zudem werden Interoperabilitätsanwendungsfälle beschrieben. Daraufhin werden durch Systemarchitekten mögliche Lösungen evaluiert, indem Akteure und deren Transaktionen fein granular, funktional zerlegt, Kommunikationsrestriktionen und Anforderungen festgehalten und mit einer zu erstellenden Sammlung bestehender Praktiken und internationaler Normen verglichen werden. Löst keines der bestehenden Praktiken oder Normen das Problem, so wird ein neues Integrationsprofil als Open Access Dokument erarbeitet. Ein Integrationsprofil wird durch Implementierungsexperten definiert und beschreibt genau einen Interoperabilitätsanwendungsfall. Dieser Interoperabilitätsanwendungsfall wird benannt, das Interoperabilitätsproblem beschrieben und Akteure und Transaktionen aufgelistet und detailliert beschrieben. Hierbei fließen unter anderem auch die Informationsflüsse, -schritte und -sequenzen sowie technische und sicherheitstechnische Bedingungen und Maßnahmen mit ein. Zudem wird jede zu verwendete Norm und Best Practises zur Lösung des Interoperabilitätsproblems beschrieben. Anschließend wird die Spezifikation durch Domänenexperten veröffentlicht. Während eines Connectathons festgestellte Probleme werden sofort oder im Nachhinein bearbeitet, sodass ein Integrationsprofil mit der Zeit reift, bis es endgültige Reife erreicht hat.

Zur Durchführung des Connectathons muss das Test-Event vorbereitet werden. Dabei erfolgt eine Registration der Teilnehmer mit Angabe der Integrationsprofil-Akteure und -Rollen, welche zu Testen gewünscht sind. Anschließend wird unter Abgleich der registrierten Fälle bekannt gegeben, welche der Testfälle während des Connectathon zu welchen Zeitpunkten geprüft werden. Zudem müssen die beteiligten Software-Anbieter die entsprechenden Schnittstellen vorab implementieren und sollten – falls vorhanden – bereits vorab mit Simulatoren / virtueller Test-Partner / Mock-Ups grundlegende Fehlerquellen ausschließen.

Die Testfälle werden anhand der Anwendungsfälle und den Spezifikationen aus den Integrationsprofilen durch Test-Experten festgelegt. Hierzu werden Test-Szenarien definiert, Evaluationskriterien festgelegt und die Randbedingungen für die Testumgebung festgehalten. Die Tests sollten sowohl positiv-Tests („etwas funktioniert wie geplant“) als auch negativ-Tests („etwas negatives wurde verhindert“), sogenannte Misuse Cases, umfassen. Unter letzterem können auch insbesondere Sicherheitstests fallen. Zudem werden Test-Sequenzen sowie -Berichte spezifiziert, optional Simulatoren / virtuelle Test-Partner / Mock-Ups für Schnittstellengegenseiten erstellt sowie die Validierungswerkzeuge für den Connectathon vorbereitet.

Während des Connectathons werden alle zu testenden Komponenten an ein zentrales Test-System (die sogenannte IHE Gazelle Testplattform) angeschlossen. Diese Testplattform wird auf die durchzuführenden Testsequenzen voreingestellt und protokolliert die Durchführung des Tests und macht Schnittstellendaten sichtbar und validierbar. Auf dem Connectathon wird nun für die Software jeweils zwei verschiedener Anbieter durch die Entwickler untersucht, ob sie miteinander interoperabel sind, indem die Testsequenzen durchgegangen werden und vom zentralen Tool protokolliert werden. Scheint der Test erfolgreich, wird eine Dritte, unabhängige Instanz zur Verifizierung der Ergebnisse hinzugezogen. Ist der Test nicht erfolgreich aber ein Erfolg steht in Aussicht, kann ad hoc an den Implementierungen der Systeme gearbeitet werden, um erneut zu Testen. Ein bestandener Test wird dementsprechend protokolliert, ein offizielles Statement zur Integration kann für abgeschlossene Produktentwicklungen beantragt werden und verfügbare Produkte können in einer zentralen, öffentlichen Liste geführt werden. Weitere Hinweise zum Integrationsprofilmanagement und zur Durchführung von Connectathons finden sich in (Franzl, Pasteka, & Gottschalk, 2019).

Mit diesem Vorgehen kann zuverlässig für zwei oder mehr Softwares nachgewiesen werden, dass Daten hinsichtlich eines Integrationsprofils interoperabel ausgetauscht werden können und es hat sich in der Domäne Healthcare seit 2001 in einem jährlichen Event etabliert (IHE, 2020).

Diese Vorgehensweise hat zudem folgende Vorteile:

- Die Weiterentwicklung/Präzisierung von Schnittstellendefinitionen ist bei Bedarf möglich
- Möglichkeit, direkte Erfahrung auszutauschen und einzubinden.
- Nachweis der syntaktischen Interoperabilität.

Das Vorgehen hat jedoch auch Schwächen:

- Bei nötiger Weiterentwicklung/Präzisierung von Schnittstellendefinitionen, ist nur der Interoperabiltitätsnachweis zu alten Schnittstellenständen erfolgt.
- Lange Vorlaufzeit und hoher Vorbereitungsaufwand.
- Mehrere Anbieter müssen sich (jedesmal) aktiv beteiligen, daraus folgen hohe und laufende Kosten.
- Semantische Interoperabilität ist nicht gezeigt.
- Sicherungsmaßnahmen (z.B. Ende-zu-Ende Verschlüsselungen) sind für die zentrale Einbindung der Testplattform gegebenenfalls auszuschalten und können dementsprechend nicht mitgeprüft werden.
- Die Interoperabilität beschränkt sich auf die getesteten Integrationsprofile. Ein falsches Branding von Software mit nicht getesteten Integrationsprofilen ist regelmäßig zu überprüfen, um das Gütesiegel nicht zu schwächen.

Für einen allgemeinen Interoperabilitätsnachweis können folgende wesentliche Schritte abgeleitet werden, auf die im Folgenden kurz eingegangen wird.

1. Festlegung einer geeigneten Schritttiefe für einen Interoperabilitätsnachweis
2. Erstellung einer genauen und stabilen Schnittstellenspezifikation
3. Erstellung einer genauen und stabilen Testspezifikation
4. Implementierung von projektindividuellen Mockups/Simulatoren, deren Anbindung und Testdurchführung

5a. Austausch Mockups/Simulatoren verschiedener Projekte und deren Anbindung und Testdurchführung

5b. Umsetzung von Referenzimplementierungen, Anbindung und Testdurchführung

6. Direkte Anbindung unterschiedlicher Systeme und Testdurchführung
7. Zusammenschluss von Systemen über eine zentrale Testplattform und Testdurchführung
8. Test der semantischen Interoperabilität

Eine genaue und stabile Schnittstellen- und Testspezifikation kann dabei wie für die Connectathons mittels Integrationsprofilen und der Spezifikation von Test-Use- und -Misuse-Cases, Erstellung von Testdatensätzen, und Testsequenzen erfolgen (Schritt 2, 3).

Für die Umsetzung im Connectathon sind Mockups, Referenzimplementierungen und Simulatoren optional. Diese können jedoch bei der Entwicklung einer Software wesentlich zur Findung von stabilen Lösungen betragen und erfolgreiche Tests bereits als erste Interoperabilitätsnachweise dienen. Dies kann wie aufgelistet gestaffelt erfolgen. Zunächst sollten in der Softwareentwicklung des Projekts/Produkts selbst Mockups für

Systemgrenzen überschreitende Kommunikation erstellt und angebunden werden, um den erfolgreichen Datenaustausch über Schnittstellen zu prüfen (Schritt 4). Sind diese Mockups jeweils in individuellen Softwareprojekten entwickelt, können diese – insbesondere wenn sie Open Source entwickelt wurden – den jeweils anderen Testpartnern zur Verfügung gestellt und dort verwendet werden, um damit festzustellen, ob Schnittstellenspezifikation und Testspezifikation von unterschiedlichen Entwicklern gleich interpretiert wurden und ein Austausch der Daten zwischen den Mockups von Dritten und der Software erfolgreich ist (Schritt 5a). Gemäß der Schnittstellen- und Testspezifikationen kann unabhängig von Softwareprojekten, deren Interoperabilität gezeigt werden soll, durch einen Dritten eine entsprechende Mockup-/Simulationsimplementierung als Referenzimplementierung erfolgen und im Detail überprüft werden. Alternativ kann durch eine unabhängige Prüfung eine Softwareprojektindividuelle und verfügbare Mockup/Simulationsimplementierung als Referenzimplementierung bestimmt werden. Eine solche Referenzimplementierung kann dann zur Überprüfung der Interoperabilität angebunden und der Datenaustausch geprüft werden (Schritt 5b). Die Schritte 5a und 5b sind in ihrer logischen Reihenfolge vertauschbar, wenn es der konkrete Kontext erfordert – beispielsweise, weil noch keine zweite Software vorhanden ist. Die Verwendung solcher Referenzimplementierungen für Interoperabilitätstests wird beispielsweise auch im Kontext der OpenInterface-Initiative mit der Zurverfügungstellung von „Basis-Schnittstellen“ angedacht (OpenInterface, 2020).

Die nächste Stufe (Schritt 6) des Nachweises von Interoperabilität zwischen zwei Systemen kann durch ihre direkte Kopplung und das Abfahren der spezifizierten Tests erreicht werden.

Der Zusammenschluss verschiedener Systeme an eine zentrale Testplattform (Schritt 7) entspricht der Vorgehensweise auf den Connectathon-Events. Hier können die Zwischenschritte der Mockups/Simulatoren optional bleiben, da die Interoperabitltät der Software auch ohne diese im Connectathon in ausreichender Weise dargestellt werden können. Nichtsdestotrotz bietet sich auch in Entwicklungsprojekten, deren Interoperabilität in einem Connectathon nachgewiesen werden soll, die Verwendung von eigenen und/oder fremd- bzw. referenzimplementierten Mockups/Simulatoren während der Entwicklungsphase an, um entwicklungsprojektintern von Anfang an methodisch auf eine Interoperabilität hinzuarbeiten und prüfen zu können. Insofern erscheint es sinnvoll, alle Schritte während der Entwicklungsphase zu berücksichtigen.

Über das Connectathon-Verfahren hinaus ist für die Interoperabilität nicht nur wichtig, ob Daten syntaktisch korrekt ausgetauscht werden, sondern auch, dass sie semantisch korrekt verwendet werden. Diese Überprüfung kann nur von Fachanwendern im Austausch mit Schnittstellenspezifizierern und den Softwareentwicklerteams unterschiedlicher beteiligter Komponenten geleistet werden. Diese Überprüfung ist in Schritt 8 aufgelistet, sollte jedoch sinnvollerweise abschließender/paralleler Teil der Tests sein, die von Schritt 4 an erfolgen.

Für einen allgemeinen Interoperabilitätsnachweis erscheint es sinnvoll vorauszuwählen, für welche Interoperabilitätsfragen ein Nachweis mittels eines Connectathon sinnvoll und möglich ist und wie weit die benannten Schritte abgearbeitet werden sollen (Schritt 1). In den bisherigen Connectathon-Formaten erfolgte die Auswahl in Abhängigkeit der Anzahl daran interessierten Partner/einzubringenden Softwares. Eine Entscheidung könnte aber auch anhand der Wichtigkeit oder der Kritikalität von Software gefällt werden. Für die Stabilität von Schnittstellen- und deren Testspezifikationen ist es sicherlich sinnvoll, vorab bereits Beteiligte unterschiedlicher Softwareprodukte/-projekte zu gewinnen. Ab Schritt 5 sind jedoch spätestens mindestens zwei unterschiedliche beteiligte Entwicklerteams von Nöten, da auch eine Referenzimplementierung von Mockups/Simulationen individuell erfolgen sollte, wenn nur eine einzige Software entwickelt/angepasst wird. Da bis einschließlich Schritt 4 Interoperabilität noch nicht für Software unabhängiger Teams getestet wurde, sollte für allgemeine Interoperabilitätsnachweise mindestens Schritt 5a oder Schritt 5b sowie Schritt 8 erfolgreich durchgeführt worden sein.

6 Zertifizierung

Eine Zertifizierung von Software für eine grundsätzlich sichere und interoperable Verwendbarkeit erleichtert potentiellen Nutzern die Auswahl von Software. Insbesondere dann, wenn die Software in kritischer Infrastruktur oder kritischem Umfeld eingesetzt werden soll, sollte/muss ein Nutzer sicher sein, sichere und interoperable Softwaresysteme zu verwenden. Eine Zertifizierung kann dabei helfen, die Vertrauenswürdigkeit von Software einzuschätzen und diese bei erfolgreichem Abschluss mittels Zertifikat zu attestieren. Für die weitere Beschreibung wird der Begriff *Zertifizierer* für Personen oder Organisationen, die Zertifikate ausstellen und der Begriff *Softwareentwickler* für Personen oder Organisationen, die die zu zertifizierende Software herstellen, genutzt. Im Folgenden wird eine allgemeine Vorgehensweise für die Zertifizierung von Software auf Basis der vorangegangenen Kapitel vorgestellt. Auch an dieser Stelle wird nicht auf die Zertifizierung für den Betrieb von Software eingegangen. Dieses obliegt möglicherweise weiteren Zertifizierungen beispielsweise nach ISO 27000 und folgende. Darauf folgend wird die Glaubwürdigkeit von Zertifikaten angesprochen und anschließend eine Möglichkeit konsortialer Zertifizierung von konsortial entwickelter Open Source Software bis zur Bereitstellung für das Umfeld von Netzleitsystemen vorgestellt.

6.1 *Vorgehensweise zur Zertifizierung*

Eine allgemeine Vorgehensweise zur Zertifizierung der sicherheitsrelevanten Aspekte und der Interoperabilität von Software bis zu ihrer Auslieferung ist in Abbildung 11 für das Umfeld von Netzleitsystemen dargestellt. Dokumente aus vorherigen Arbeitsschritten dienen in allen nachfolgenden Arbeitsschritten als Grundlage. Aus Gründen der Übersichtlichkeit wurden entsprechende Abhängigkeiten in der Darstellung weggelassen. Zunächst wird die Zertifizierung durch die Software-Entwickler und die Zertifizierer vorbereitet und die entsprechenden Dokumente bzw. (Rand-)Systeme/Software beigebracht. Der Software-Entwickler bringt dazu die Anforderungen an die Software, die Entwicklungsdokumentationen sowie die Software und auf die Software bezogene Testcode und -dokumentation ein. Der Zertifizierer hat die Aufgabe sich den Kontext Netzleitsystem-Umfeld für den zertifiziert wird zu erarbeiten und dessen Auflagen (z.B. gesetzlich) vollumfänglich zu kennen, und für die angedachten Interoperabilitätsnachweise entsprechende Schnittstellendefinitionen, Testdefinitionen zu sammeln und die benötigten Randsysteme verfügbar zu machen.

Im nächsten Schritt erfolgt eine Prüfung („Vollständigkeitsprüfung“), ob die Entwicklungsdokumentation die Punkte der Checklisten zur Entwurfsphasensicherheit (siehe Abschnitt 2.4), zur Implementierungssicherheit (siehe Abschnitt 3.4) und zu den Benutzungsschnittstellen (siehe Abschnitt 4) aufführt. Anschließend können Software und Tests in einer Qualitätssicherungsumgebung eingerichtet werden. Ist die Software Open Source Software, kann an dieser Stelle kompiliert und Tests durchgeführt und protokolliert werden. Ist die Software Closed Source, kann der Quelltext gegebenenfalls nicht durch den Zertifizierer eingesehen werden und er muss den zu liefernenden Protokoll-Ergebnissen der Tests der Software durch den Softwareentwickler vertrauen – zu denen er jedoch Auflagen machen können sollte.

Im folgenden Schritt erfolgt die sicherheitstechnische Abnahme nach den Fragestellungen der genannten Checklisten gemäß der Auflagen. Hierbei kann der Zertifizierer prüfen, ob die in der Entwicklungsdokumentation benannten Umsetzungen die Auflagen erfüllen. Je nach Einsicht in den geschlossenen Quellcode ist gegebenenfalls jedoch nur eine oberflächliche aber keine tiefergehende Kontrolle möglich, ob die in der Entwicklerdokumentation genannten Lösungsstrategien auch „unter der Haube“ so umgesetzt wurden, wie es benannt wurde. Für Open Source Software kann dahingegen, abhängig von der vom Kontext verlangten nötigen Tiefe, sowohl eine zufällige stichprobenartige oder/und tiefe Inspektion des Codes erfolgen. Damit kann festgestellt werden, ob die in der Entwicklerdokumentation beschriebenen Lösungsansätze tatsächlich auch „unter der Haube“ so wie benannt umgesetzt wurden und ob alle Teile der Software dokumentiert sind.

Nachfolgend können Interoperabilitätstests (IOP-Test) (wie in Abschnitt 5 beschrieben) für die zu zertifizierenden Schnittstellen auf Basis von offiziellen/konsolidierten Definitionen der Interoperabilitäts-Schnittstellen (IOP-API) durchgeführt werden. Je nach Anzahl vergleichbarer Werkzeuge oder Gegenseiten, zu denen Schnittstellen überprüft werden, ist hierbei eine unterschiedlich weit fortgeschrittene Umsetzung

sinnvoll, die jedoch immer mindestens Implementierungen zweier verschiedener Entwicklergruppen berücksichtigen sollte. Zudem sollten Zertifizierern stets Fachanwender zur Seite stehen, um zur Einschätzung der semantischen Korrektheit der übertragenen Daten beitragen zu können.

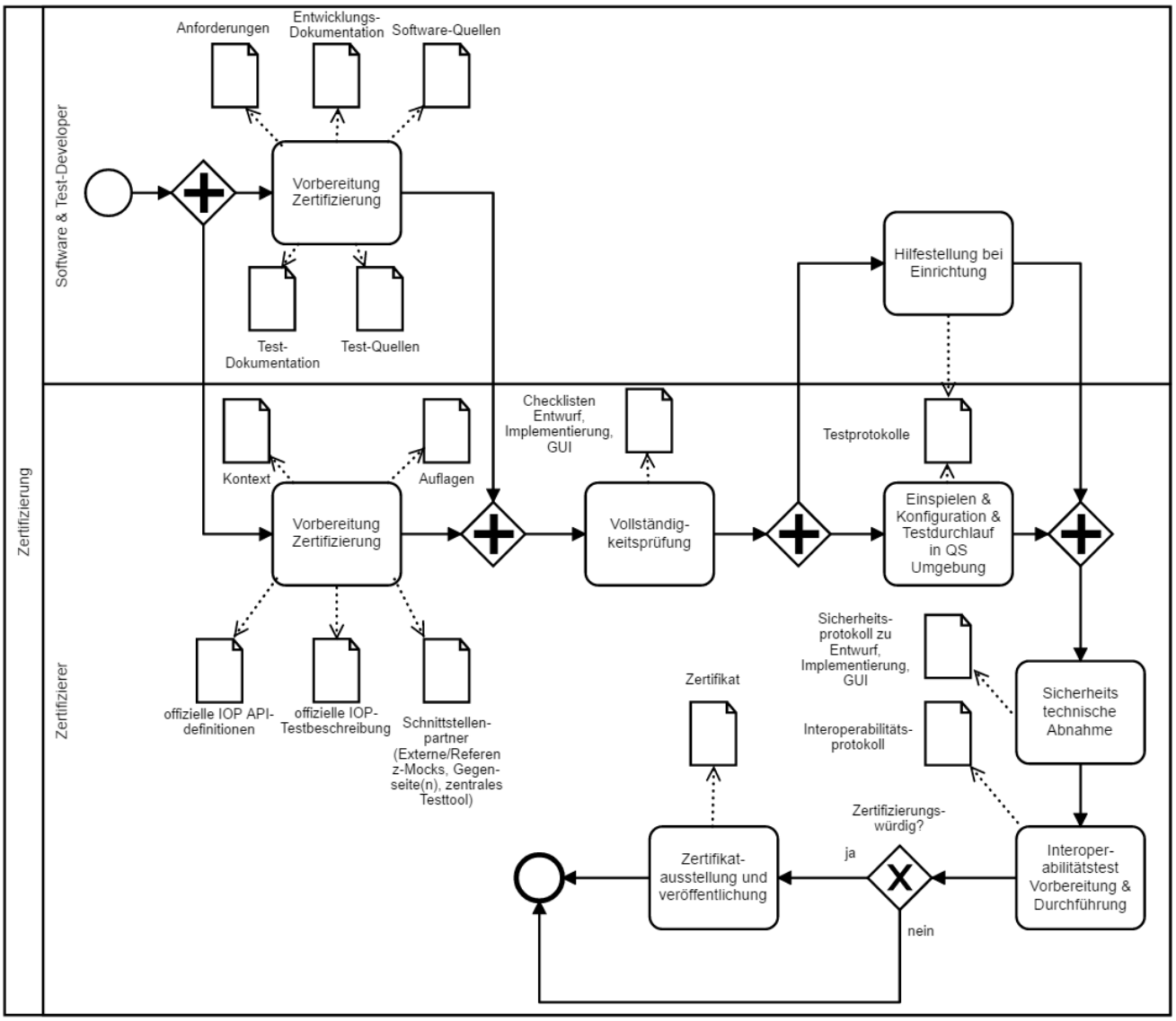


Abbildung 11: Prozess zur Zertifizierung von Software

Sind alle Überprüfungen erfolgreich abgeschlossen und protokolliert, so kann der Zertifizierer ein Zertifikat für die Sicherheit der Software und die Interoperabilität der überprüften Schnittstellen ausstellen.

6.2 *Glaubwürdigkeit von Zertifikaten*

Die Glaubwürdigkeit eines Zertifikats hängt in hohem Maße von folgenden Punkten ab:

1. Unabhängigkeit von Zertifizierer und Softwareentwickler,
2. Reputation des Zertifizierers und Konsequenzen bei Fehlverhalten,
3. Offenlegung der Zertifizierungsergebnisse / -methodik / -zwischenergebnisse,
4. Unterbindung von Missbrauch.

Hinsichtlich der Unabhängigkeit von Zertifizierer und Softwareentwickler sind eine möglichst große Unabhängigkeit und deren Sichtbarkeit entscheidend für das Vertrauen in ein Zertifikat. Ein Softwareentwickler selbst kann kaum seriös seine eigene Software zertifizieren. Besteht eine Unabhängigkeit von Zertifizierer und Softwareentwickler wird die Vertrauenswürdigkeit geschwächt, wenn diese nicht sichtbar ist oder Offensichtliches eine Abhängigkeit suggeriert. Dies könnten beispielsweise die

Organisationszugehörigkeit in der gleichen Abteilung / Organisation / Dachorganisation sein. Auch die Bezahlung eines Zertifizierers durch den Softwareentwickler kann die Vertrauenswürdigkeit schwächen.

Bezüglich der Reputation des Zertifizierers kann (Zertifizierungs-) Kompetenz eine Rolle spielen. Wesentlich ist jedoch auch die Integrität des Zertifizierers. Wird durch Fehlverhalten des Zertifizierers zum Beispiel ein Zertifikat trotz nicht bestandener wichtiger Prüfungen oder Auslassung von Prüfungen ausgestellt, ist seine Integrität eingeschränkt. Wird dieser Fall öffentlich, so erleidet der Zertifizierer einen Vertrauensverlust. So ein Vertrauensverlust kann die Reputation eines Zertifizierers dauerhaft zerstören. Möchte der Zertifizierer auch zukünftig verlässliche Zertifikate ausstellen oder ist er sogar darauf angewiesen, liegt es in seinem eigenen Interesse, Vertrauensverlust durch Fehlverhalten zu vermeiden. Da auch vergangene Zertifikate in der Glaubwürdigkeit leiden, besteht vielleicht von Betroffenen das Interesse die Veröffentlichung von Zertifikatproblemen zu unterbinden. Die Veröffentlichungsmöglichkeit von Zertifikatproblemen ist jedoch wesentlicher Bestandteil der Glaubwürdigkeit, da ein Zertifizierer gewissenhaft arbeiten muss, damit es nicht zu Beanstandungen des Zertifikats kommt und der Zertifizierer zukünftig weiter als seriöser Zertifizierer wahrgenommen werden und arbeiten kann.

Über die Ausstellung eines Zertifikats hinausgehend, kann auch die Offenlegung der Zertifizierung wesentlich zur Glaubwürdigkeit beitragen, wenn einzelne Fragestellungen der Zertifizierung nachvollzogen werden können. Über die Ausstellung des Zertifikats hinaus könnten auch die Ergebnisse der Prüfungen, die zur Zertifikatsausstellung führen veröffentlich werden. Auch die Veröffentlichung der Zertifizierungsmethoden ist sinnvoll, damit nachvollzogen werden kann, wie es zu Ergebnissen gekommen ist. Damit werden Prüfungen reproduzierbar und Ergebnisse überprüfbar. Werden auch Zwischenergebnisse offengelegt, so ist zusammen mit den offengelegten Methoden eine Nachprüfung noch einfacher. Je offener die Zertifizierung durchgeführt und dokumentiert wird, desto einfacher können Ergebnisse nachvollzogen werden und desto glaubwürdiger ist ein Zertifikat, da Probleme schneller aufgedeckt werden könnten und dadurch wieder die Reputation des Zertifizierers leiden könnte. Während bei Closed Source Entwicklungen gegebenenfalls Zertifizierungs(-zwischen-)ergebnisse nicht veröffentlicht werden können oder zumindest nicht nachvollzogen werden können, sollte bei Open Source Entwicklungen die Zertifizierung öffentlich dokumentiert, alle Schritte nachvollzogen und (Zwischen-)Ergebnisse reproduziert werden können, sodass hier eine höhere Glaubwürdigkeit von Zertifikaten möglich ist.

Zudem muss Missbrauch des Zertifikats unterbunden werden, um eine mögliche Schwächung des Zertifikats zu verhindern. Hierzu sollte regelmäßig die Korrektheit der Benennung von Zertifikaten durch den Herausgeber überprüft werden und eine geeignete Re-Zertifizierung für Software-Änderungen etabliert werden. Zudem könnte eine öffentlich einsehbare Liste erfolgreicher Zertifizierungen geführt werden, damit eine Zertifikatsauszeichnung von Software unabhängig überprüft werden kann.

6.3 *Konsortiale Zertifizierung konsortialer Open Source Software*

Im Allgemeinen kann nicht davon ausgegangen werden, dass Software einer adäquaten, unabhängigen sicherheitstechnischen Überprüfung vor Auslieferung unterliegt. Aufgrund dessen muss der Zertifizierer diese Aufgabe im Allgemeinen wie in Abschnitt 6.1 beschrieben, übernehmen. Wird eine Software jedoch als Open Source entwickelt und vollständig Open Source dokumentiert, kann aufbauend auf einer adäquaten Qualitätssicherung eine Zertifizierung vereinfacht durchgeführt werden, wenn der – im besten Fall von der Software-Entwicklung unabhängige – Qualitätssicherer die sicherheitstechnischen Aspekte der Zertifizierung mitberücksichtigt und die (Zwischen-)Ergebnisse offen protokolliert. Durch die Offenheit von Quellen und Dokumentationen sind die Ergebnisse leicht reproduzierbar und überprüfbar und durch die Unabhängigkeit von Software-Entwicklung und Qualitätssicherer glaubhaft.

In einem Konsortium unabhängiger Entwickler und Anwender (wie Beispielsweise openKONSEQUENZ) kann eine weitestgehend von der Softwareentwicklung unabhängige Qualitätssicherung und Zertifizierung von Open Source Software durchgeführt werden. Es kann sichergestellt werden, dass Qualitätssicherer, Zertifizierer und Software-Entwickler anderen Organisationen angehören und jeder das Interesse an möglichst hochqualitativer Software hat. Durch Open Source Code und offene Dokumentation der Entwicklung aber auch der Qualitätssicherung und Zertifizierung sind die Ergebnisse leicht nachzuvollziehen. Schlecht durchgeführte

Qualitätssicherung, Zertifizierung aber auch Software-Entwicklung tritt offen Zutage. Da die Reputation für weitere Aufträge wichtig ist, besteht für jeden zudem das Eigeninteresse Fehler zu vermeiden.

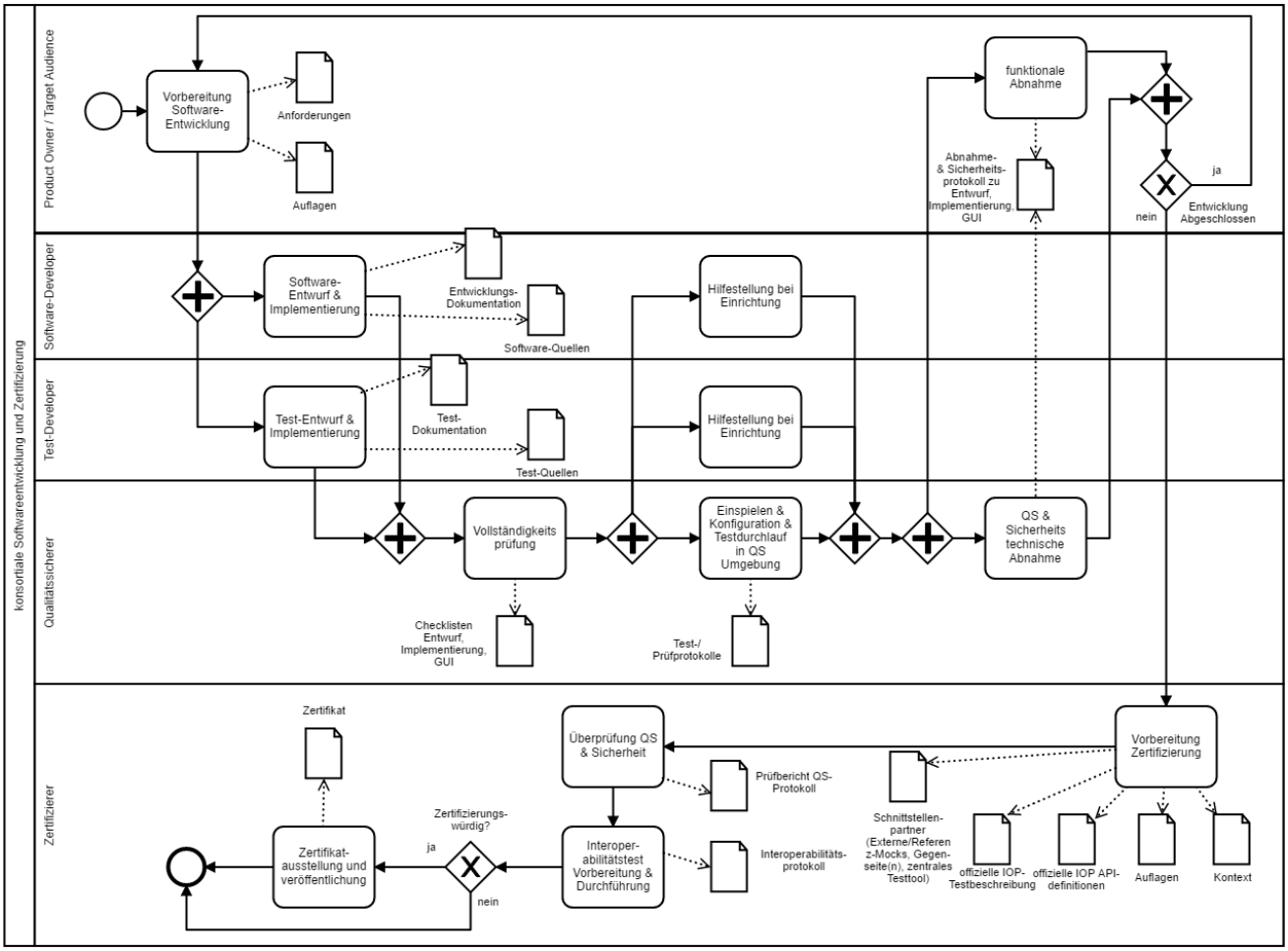


Abbildung 12: Prozess der konsortialen Open Source Software-Entwicklung und Zertifizierung

Die Prozesse der konsortialen Software-Entwicklung (siehe Abschnitt 3.3) und der Zertifizierung (siehe Abschnitt 6.1) können so zusammengeführt und vereinfacht werden. Insbesondere kann auch bereits während der Entwicklung positiv auf sicherheitstechnische Aspekte / Prozesse eingewirkt werden. So kann möglicherweise verhindert werden, dass erst im Nachhinein Mängel offengelegt werden, die eine Zertifizierung verhindern. Zudem sind im Nachhinein durch den Zertifizierer schwierig zu überprüfende Aussagen (beispielswiese über die Ausprägungen von Entwicklungsprozessen) möglicherweise durch den entwicklungsbegleitenden Qualitätssicherer überprüf- und belegbar, sodass sogar eine Stärkung des Zertifikats möglich ist. Der entsprechend zusammengefasste Prozess findet sich in Abbildung 12. Hierbei überprüft und dokumentiert bereits der Qualitätssicherer anhand der Checklisten die Vollständigkeit der Dokumentationen auch bezüglich der sicherheitstechnischen Fragestellungen. Zudem werden Test- und Prüfprotokolle während des Buildprozesses der Software generiert, untersucht sowie protokolliert und während der sicherheitstechnischen Abnahme auch die Checklisten inhaltlich berücksichtigt. Die Hauptaufgabe der Prüfung und Dokumentation sicherheitstechnischer Aspekte verschiebt sich also vom Zertifizierer zum Qualitätssicherer. Der Zertifizierer muss sich dennoch in den Kontext und deren Auflagen einarbeiten, muss aber nicht mehr alles von Neuem erarbeiten, sondern kann anhand der bestehenden Dokumentationen des Qualitätssicherers die Qualitätssicherung in der nötigen Tiefe beziehungsweise stichpunktartig nachvollziehen / kontrollieren. Eine vollständige Dopplung der Aufgaben für qualitätsgesicherte Software bei Qualitätssicherer und Zertifizierer entfällt somit. Darüber hinaus kümmert sich der Zertifizierer wie gehabt um die Tests zur Sicherstellung der Interoperabilität der Software zu den im Konsortium eingesetzten Systemen und der anschließenden Ausstellung des Zertifikats.

Die Vermeidung von Missbrauch von Zertifikaten liegt im Interesse aller Konsortium-Beteiligten, da ein solcher Missbrauch das Konsortiums-Zertifikat, die eigene Reputation, und die des gemeinsamen Konsortiums schwächt. Eine öffentlich einsehbare Infrastruktur für das Melden von Missbrauch und für die Zertifikatausstellung und -Dokumentation kann konsortial geteilt und betrieben werden, wodurch sich Einzelaufwände reduzieren und Missbrauch voraussichtlich schneller entdeckt wird.

7 Zusammenfassung und Ausblick

In der kritischen Infrastruktur Energiesysteme/Wasserversorgung sollte im Umfeld von Netzleitsystemen sicherheitstechnisch gestärkte Software für den Einsatz in die Systemlandschaft ausgewählt werden, damit der Betrieb der Infrastruktur nicht durch Sicherheitsvorfälle gefährdet ist und Sicherheitsrisiken weitest möglich reduziert sind. Hierzu werden in dem vorliegenden Dokument die sicherheitstechnischen Anforderungen für die Entwurfs- und Implementierungsphasen und an die Benutzungsschnittstellen gemäß der für Software im Umfeld der Netzleitsystem einschlägigen Literatur benannt und insbesondere für Open Source Software basierend auf Erfahrungen aus NetzDatenStrom und den Enwicklungsprojekten in openKONSEQUENZ diskutiert. Darüber hinaus wird ein Verfahren zur Sicherstellung von Interoperabilität vorgestellt. Zu jedem der angesprochenen Themen werden Checklisten zur Verfügung gestellt, die als Basis für eine umfassende Prüfung dienen können. Darauf aufbauend wird ein Zertifizierungsverfahren für Software im Umfeld von Netzleitsystemen vorgestellt und die besondere Eignung von konsortial entwickelter Open Source Software, für eine transparent als sicher verwendbar zertifizierte Software, hervorgehoben.

Das vorliegende Dokument beschränkt sich hierbei auf Software die vor der Auslieferung und dem Einsatz steht. Für die Sicherstellung von Sicherheit von Software im laufenden Betrieb sind jeweils weitere Überprüfungen (z.B. nach ISO 27000 und folgende) nötig. Darüber hinaus beschreibt (Cleveland, 2012) auch einen fünfphasigen Sicherheitsprozess (Bewertung, Richtlinienüberarbeitung, Umsetzung, Personalschulung und Auditierung) hinsichtlich der Sicherstellung von Sicherheit während des Betriebs des Gesamtsystems. Auch die Wartung von konsortial entwickelter Open Source Software wurde im Dokument nur angesprochen aber nicht tiefergehender diskutiert und ist ein offener Arbeitspunkt, der beispielsweise im Kontext von openKONSEQUENZ vorangetrieben wird.

8 Literaturverzeichnis

BDEW. (2018). *Whitepaper Anforderungen an sichere Steuerungs- und Telekommunikationssyssteme.* Berlin: BDEW.

BSI. (2020). *IT-Grundschutz-Kompendium.* Bonn: Bundesamt für Sicherheit in der Informationstechnik.

Cleveland, F. (2012). *IEC TC57 WG15: IEC 62351 Security Standards for the Power System Information Infrastructure.* International Electrotechnical Commision.

Eclipse. (26. 03 2020). *Commiter Members - The Eclipse Foundation*. Von https://www.eclipse.org/membership/become_a_member/committer.php abgerufen

Franzl, G., Pasteka, R., & Gottschalk, M. (2019). *The IES Cookbook.* Wien: IES - Integrating the Energy System.

Harner, A. (2019). *Anwendungshinweis für die Normenreihe IEC 62351: Informationssicherheit in der Netz- und Stationsleittechnik.* Frankfurt am Main: VDE/DKE.

Herczeg, M. (2014). *Prozessführungssysteme. Sicherheitskritische Mensch-Maschine-Systeme und interaktive Medien zur Überwachung und Steuerung von Prozessen in Echtzeit.* München: de Gruyter Oldenbourg.

Hollnagel, E., & Woods, D. (2005). *Joint cognitive systems: Foundations of cognitive systems engineering.* Boca Raton, FL: CRC Press / Taylor & Francis.

IEC. (2011). *IEC/TS 62351-8: Power systems management and associated information exchange –Data and communications security – Part 8: Role-based access control .* Genf: IEC.

IEC. (2012). *IEC/TR 62351-10: Power systems managment and associated information exchange - Data and communications security - Part 10: Security architecture guidelines.* Genf: International Electrotechnical Commision.

IEC. (2017). *62351-7: Power systems management and associated information exchange - Data and communications security - Part 7: Network and System Management (NSM) data object models.* Genf: IEC.

IEC. (2017). *IEC 62351-9: Power systems management and associated information exchange - Data and communications security - Part 9: Cyber security key management for power system equipment.* Genf: IEC.

IHE. (2013). *IHE-D Cookbook.* Deutschland : IHE - Integrating the Healthcare Enterprise.

IHE. (21. 04 2020). *IHE Gazelle - History of Connectathons*. Von https://gazelle.ihe.net/content/history-connectathons abgerufen

INCITS. (2004). *ANSI INCITS 359-2004: Information technology - Role Based Access Control.* Washington D.C.: INCITS.

Mentler, T. (2018). Usability Engineering und User Experience Design sicherhietskritischer Systeme. In R. C., *Sicherheitskritische Mensch-Computer-Interaktion.* Wiesbaden: Springer Vieweg.

Norman. (1995). Human error and the design of computer systems. In *Human-computer interaction: toward the year 2000* (S. 681-683). San Francisco, CA, USA: Kaufmann Publishers Inc.

oK Architecture Committee. (2019). *Architecture Committee Handbook - Version 1.6.0.* Web: openKONSEQUENZ.de.

oK Quality Committee. (2020). *Quality Committee Handbook - Release 2.0.2.* Web: openKONSEQUENZ.de.

OpenInterface. (21. 04 2020). *OpenInterface - Mission Statement*. Von https://www.openinterface.de/mission-statement/ abgerufen

OWASP. (2019). *Application Security Verification Standard 4.0.* owasp.org: OWASP.

OWASP. (09. 03 2020). *OWASP Top Ten*. Von https://owasp.org/www-project-top-ten/ abgerufen

Reason, J. (1990). *Human Error.* Cambridge: Cambridge University Press.

Scarfone, K., Jansen, W., & Tracey, M. (2008). *Guide to General Server Security - Special Publication 800-123 - Recommendations of the National Institute of Standars and Technology.* Gaithersburg: National Institute of Standards and Technology - U.S. Department of Commerce.

Schlegel, R., Obermeier, S., & Scheider, J. (2015). Assessing the Security of IEC 62351. *3rd International Symposium for ICS & SCADA Cyber Security Research* (S. 11-19). Swindon, UK: BCS Learning & Development Ltd.

Wittek, A. (2018). *Introduction to the Eclipse IP management process for Eclipse openK projects - Version 2.0.* Web: openKONSEQUENZ.de.