Security-Engineering nach Common Criteria
Methodik für eine *Security-by-Design* Praxis

Dr. Jörg Kebbedies
TrustKBB GmbH
10117 Berlin, Germany
Email: joerg.kebbedies@trustkbb.de

***Abstract*—Die rasante Entwicklung digitaler Baugruppen und Systeme erfordert neue Konzepte für ein ganzheitliches und methodisches Sicherheitsdesign. Erfolgreiche Cyber-Angriffe kennzeichnen fast täglich methodische Defizite im Sicherheitsdesign von IT-Architekturen.**

Das Kernproblem entwickelt sich aus einem wachsenden Bedarf an komplexen, funktional neuen, digitalen Lösungen, jedoch unter engen Entwicklungszeiten. Die gezielte Vermeidung schwer zu identifizierender Software- oder Systemschwachstellen zeigt sich als eine kaum zu beherrschende Herausforderung im digitalen Fertigungsprozess.

Der Begriff *Security-by-Design* formuliert die Forderung, die getrennte Betrachtungsweise über Funktion und Sicherheit in ein geschlossenes Praxismodell zu überführen. Die Frage nach einem durchgehenden Prozess für ein *Security-Engineering* stellt sich in den Vordergrund.

Wie lässt sich eine strukturierte Sicherheitsmethodik nach den *best practise*-Methoden einer *Common Criteria* in eine vertrauenswürdige Softwareentwicklungspraxis einbinden?

I. Informationssicherheit – Kein Selbstzweck

Informationssicherheit (Cyber-Security) wird in der gegenwärtigen Software-Entwicklungspraxis oft als zusätzliche *Notwendigkeit* empfunden. Für die Entwicklung von IT-Produkten, die unter den Bedingungen einer späteren Zertifizierung entstehen, sind jedoch solche qualifizierenden Prozesse erforderlich.

In der funktionsorientierten Softwareentwicklung folgt man bereits modernen Vorgehensmodellen wie *Srum* [9], *Kanban* [4]. Etablierte Prozesse wie *Continuous Integration*, *Continuous Delivery* bzw. *Continuous Deployment* [13] setzen technologische Standards für einen automatisierten und effizienten Life-Cycle zur Bereitstellung neuer Software-Releases.

Für das Paradigma *Security-by-Design* gibt es bisher keine vergleichbare methodische Systematisierung im Prozess der Softwareentwicklung.

Security-Engineering ist eine strukturierte Methodik, um digitale Systeme kontinuierlich im Entwicklungszyklus so zu gestalten, dass sie in der Interaktion mit sich ändernden Bedrohungsformen ihre Funktionsweisen zuverlässig bereitstellen und spezifische *Assets* schützen.

Das BSI etablierte dafür formale Rahmenprozesse, um Zertifizierungsprogramme zu durchlaufen [3]. Der Vorteil dieser Vorgehensweise liegt in der vertrauenswürdigen Zusicherung von definierten Sicherheitseigenschaften. Die dahinter liegende Methodik zeigt schnell ihre Grenzen, die einer hohen Entwicklungsdynamik einen limitierenden Faktor entgegensetzen.

A. Entkoppelte Sicherheitsbetrachtung

Die praktische Umsetzung einer formalen Sicherheitsbewertung, z. B. nach den Standards der Common Criteria [1], und ein modellbasiertes Software-Engineering erfolgt oft in einer fachlichen Aufteilung. Unterschiedliche Kompetenzen in Methodik, Denkart, vor allem in der Dokumentation, sind selten an eine spezifikative Zuständigkeit gebunden. Sicherheitsbetrachtungen werden gewöhnlich organisatorisch und damit zeitlich verlagert.

Die formale Aufteilung verhindert eine ganzheitliche, konsistente Durchdringung des Lösungsgegenstandes. Getrennte Dokumentationen teilen spezifikatives Wissen auf. Im Ergebnis führt es oft zu einer unvollständigen, unzureichenden und im Extremfall zu inkorrekten Sicherheitsimplementierungen.

B. Nicht nachvollziehbare Architekturentscheidungen

Security-Engineering ist durch einen Prozess kontinuierlicher Entscheidungsfindung geprägt. Sicherheitsziele beeinflussen und konstituieren die Sicherheitsarchitektur. Jede funktionale Weiterentwicklung fordert zu angemessenen Designentscheidungen heraus [8].

Die durch Formalismen künstlich differenzierte Entwicklungspraxis führte zu einem Defizit sicherheitsrelevanter Architekturentscheidungen. Die Beschreibung von Sicherheitszielen außerhalb eines Architekturmodells erfordert einen hohen Aufwand, um alle Entscheidungen für eine Sicherheitsarchitektur konsistent und nachvollziehbar zu dokumentieren.

C. Ineffiziente Sicherheitsbewertung (Evaluierung)

Die Bereitstellung notwendiger Nachweise zur Erlangung einer Sicherheitszulassung ist formal aufwändig, komplex und mit einem hohen Personalaufwand verbunden. Nachweise zur Evaluierung entstehen in Papierform. Die Erarbeitung z. B. eines *Security Targets* [5, Seite 64] erfolgt außerhalb der Entwicklung eines *Target-of-Evaluation*).

Autorisierte Prüfstellen bewerten manuell sehr aufwändig die Spezifikationen mit Sourcecode-Details. Die Bewertung stützt sich auf eine *Papierlage*, außerhalb von einer semantischen Modellgrundlage der Sicherheitsarchitektur. Der Überprüfungsaufwand bleibt schwer zu kalkulieren und steht im Widerspruch zur Dynamik moderner Software-Releasentwicklungen.

II. Konzeptansatz – Security-Engineering

Der vorgestellte Konzeptansatz für ein modellbasiertes *Security-Engineering* folgt den bewertenden, streng formalisierten und vertrauensbildenden Konzepten der *Common Criteria* (CC) [1]. Jedes zu entwickelnde IT-System (Target of Evaluation) (TOE) unterzieht sich einer strukturierten Methodik zur Sicherheitsbewertung und einem daraus abgeleiteten Sicherheitsdesign.

Die TOE-Modell-Methodik in Abbildung 1 identifiziert bestehende Sicherheitsprobleme (*Security Problem Definition*). Die Sicherheitsstrategie erhält mit der Problembeschreibung ihre Ausrichtung. Ein formaler Handlungsrahmen mit begründeten Sicherheitszielen (*Security Objectives*) ist Ausgangspunkt von Entscheidungen für eine angemessene Sicherheitslösung (*Security Solution Design*).

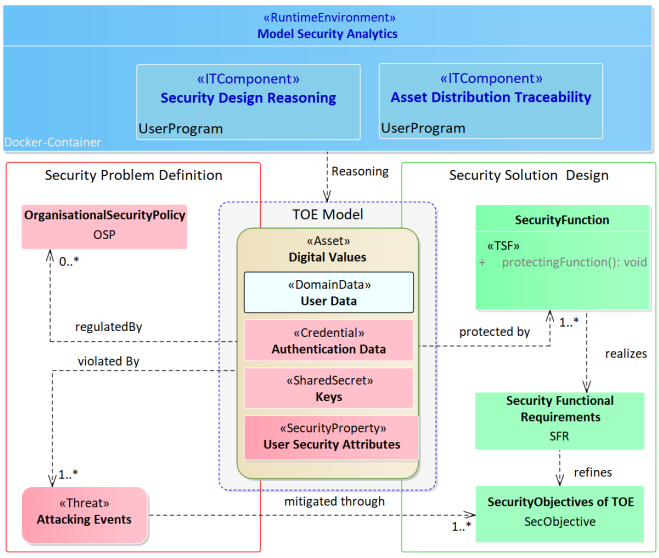

Abbildung 1: CC-basierte Sicherheitsmethodik

Der CC-Klassenkatalog für funktionale Sicherheitsanforderungen (*Security Functional Requirements*) [6] stellt einen vollumfassenden und *best-practise* erprobten Katalog möglicher Sicherheitsbausteine zur Verfügung.

Für die Entwicklung qualifizierter IT-Sicherheitskomponenten und Architekturen wurden formale CC-Terminologien in semantisch interpretierbare CC-Modellklassen überführt.

Als Instrument für ein *Security-by-Design* werden funktionale Designstrukturen direkt mit einer Sicherheitsorientierten Perspektive verknüpft, begründet und bleiben nachvollziehbar.

A. Modellklassen – Common Criteria (CC)

Die CC-Terminologie wurde strukturiert für die Sicherheitsmodellierung und Nachweiserzeugung in ein eigenständiges CC-Profil [12] für die UML-Sprache überführt und damit semantisch formalisiert.

Die Transformation der CC-Terminologie erfolgt durch die Anwendung spezifischer (*Stereotypes*), Beziehungen (*Relations*) und grundlegenden *Security-Pattern*. Die Übernahme formalisierter SFR-Klassen (CC-Klassenkatalog) in Modellelemente ist Teil der Transformation. Der technische Import stützt sich auf eine XML-basierte Analyse der CC-Terminologie und der Extrahierung von SFR-Klassen aus den publizierten CC-Releases [10].

Security Functional Requirements (SFRs) bilden das Herzstück für eine formalisierte Funktionsbeschreibung und ein strukturiertes Sicherheitsdesign.

Unterteilt in Sicherheitsklassen, beschreibt jede Komponente Sicherheitsanforderungen (*Security Functional Requirements* (SFRs) in Form von Sicherheitseigenschaften (*security properties*) und sicherheitsrelevanten Verhaltensformen (defined rules).

SFR-Sicherheitsklassen beinhalten eine axiomatisierte, regelbasierte Beschreibungsform. Sie umfassen *Operations*, inklusive formalisierter Ausführungsattribute (z. B. *iteration*, *assignment*, *selection*) und *Security Attributes*. Die Formalisierung gewährleistet eine vergleichende Interpretierbarkeit des Evaluationsgegenstandes. Die Abstraktionsform nimmt keinen Bezug auf eine konkrete Implementierung.

Einem Sicherheitsarchitekten stehen für den Sicherheitsentwurf *best-practise* CC-Modellbausteine zur Verfügung, um Sicherheitsanforderungen in eine Architektur einzubeziehen.

B. Modellklassen – Security-Domain

Der Gegenstand einer CC-Betrachtung sind IT-Strukturen einer digitalen Security-Domain. Die CC-Terminologie unterscheidet in *user*, *subjects*, *objects*, *resources*, *information* und die darauf anzuwendenden *operations*. Die Konzeptklassen bilden den *Scope* zur Entwicklung einer Sicherheitspolicy (*Security Policy Definition*).

In einem eigenständigen Security-Domain-Profil wurden adäquate Konzeptklassen für *Entity*, *Data*, *Interfaces*, *Network* und *Actor* abgebildet. Konzeptklassen wie *Organisation* und *Roles* erweitern konzeptionelle Informationsbeziehungen um einen fachlichen Bezug für Zuständigkeit und Verantwortung.

Die modellbasierte Verlinkung der CC-Terminologie über ein Security-Domain-Profil verbindet die CC-Methodik mit dem funktionalen Entwurf beliebig komplexer IT-Architekturen.

C. Modellklassen – Kryptographie

Funktionale Konzepte für Sicherheitsziele wie z. B. *Vertraulichkeit*, *Verbindlichkeit* oder *Zurechenbarkeit* benötigen häufig kryptographische Methoden.

Ein Cryptography-Profil stellt Konzeptklassen zur Beschreibung kryptographischer Funktionen (*TOE Security Functions*) bereit. Darüber hinaus lassen sich kryptographische Beziehungen zwischen TOE spezifischen Security-Daten (*TSF-Data*) und zu schützende Daten der Security-Domain (*Assets*) präzise beschreiben.

Der Funktionsumfang umfasst u. a. die Themengebiete *Vertrauenswürdige Identitäten* (eID), *Credential* (X.509 Zertifikate), *Verschlüsselung* (symmetrisch, asymmetrisch) und *Digitale Signatur*.

III. Sicherheitsproblembeschreibung (Gefährdungssicht)

Der Schutz digitaler Werte (*Assets*) setzt den Ausgangspunkt einer CC-Problembeschreibung (*Security Problem Definition* (SPD)) und wird auf Grundlage einer durchgehenden Gefährdungsbewertung geführt.

Ein qualifiziertes Verständnis zu identifizierten Sicherheitsproblemen lenkt Designentscheidungen für eine Sicherheitsarchitektur auf angemessene Sicherheitsmaßnahmen .

A. Identifikation von Assets

Der Begriff informationelles *Asset* kennzeichnet einen *Wertzustand* (*Digital Value*) identifizierbarer Datenstrukturen. Der Asset-Zustand wird ausgewählten Datenstrukturen als Eigenschaft dynamisch zugeordnet und entwickelt sich aus einem Prozesskontext heraus.

Die CC-Terminologie unterscheidet zwischen fachbezogenen Datenobjekten der Anwender (*User-Data*) und spezifische sicherheitsrelevante Datenstrukturen (*TSF-Data*), die für die Ausführung von TOE-Sicherheitsfunktionen erforderlich sind und von vornherein einen grundlegenden Schutzbedarf erzeugen.

1) Asset-Kategorien: Im allgemeinen Diskurs zur Identifikation von *Assets* werden neben digitalen Informationswerten auch technische Ressourcen mit in die *Asset*-Betrachtung einbezogen.

Die Praxis zeigt, das sowohl die Identifikation als auch ihre strukturierte Erhebung und Kategorisierung mit Schwierigkeiten verbunden sind. Ausgehend von den im Modell gekennzeichneten informationellen *Assets* sichert ein hierarchisches Vorgehen eine vollständige Erhebung.

2) Asset-Identifikation: Informations-*Assets* verteilen sich prozessbedingt auf verschiedene Bereiche einer IT-Architektur. Prozessänderungen und Architekturerweiterungen verlangen eine *adaptive* Nachführung eines bestehenden Sicherheitsniveaus auf die betroffenen Architekturbereiche.

Die modellgestützte CC-Methodik nutzt Methoden für die Verfolgung von Asset-Verteilungen (*Asset-Distribution-Traceability*). Die Analyse dynamischer Sichten (Interaction Views) führt zu einer automatisierten Sicht auf die Verteilung von *Assets* über Komponenten und Netzwerkbereiche komplexer, verteilter Sicherheitsarchitekturen.

3) Asset-Reasoning: Die modellgestützte CC-Methodik prüft bestehende Bindungen von Informations-*Assets* an Entitäten eines domänenspezifischen Architekturmodells.

Ressourcen-*Assets* sind Entitäten, welche zur Repräsentation, Verarbeitung und Speicherung beitragen [7, Seite 30]. Methoden für ein (*Security Design Reasoning*) identifizieren, kennzeichnen und kategorisieren dynamisch einbezogene Architekturbereiche als Ressourcen-*Asset*.

B. Threat-Analyse – (Attacking Events)

IT-Systeme interagieren über Schnittstellen (*TOE Security Functionality Interface*) (TSFI) u. a. mit einer nicht-vertrauenswürdigen Umgebung und werden mit bedrohenden Ereignissen (*Attacking Event*) konfrontiert.

Bedrohende Ereignisse lassen sich nicht abschaffen, jedoch wird ihre Wirksamkeit beeinflusst. Die Beobachtung und Bewertung existierender Bedrohungen (kurz *Threat*) ermöglicht die Aufdeckung von Schwachstellen (*WeakPoint*) innerhalb einer TOE-Architektur. In Abbildung 2 wird eine mögliche und noch nicht identifizierte Schwachstelle mit eingebunden.

Das funktionale TOE-Design wird mit jeder Schnittstellenspezifikation in Beziehung zu bekannten Bedrohungstypen gesetzt. Das Modell stellt dafür nach STRIDE [11] strukturierte Bedrohungsklassen bereit, welche einer kontinuierlichen Aktualisierung und Erweiterung unterliegen.

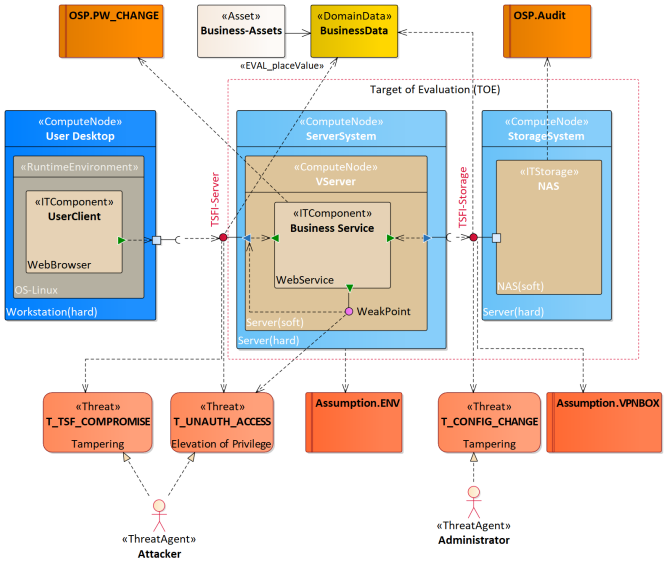

Abbildung 2: Security-Problem-Definition im Modell

Bedrohungen werden als spezifische *Aktivitäten* abgeleitet. Jeder Bedrohung ist ein Akteur (*ThreatAgent*) zugeordnet und ermöglicht die Abbildung von strukturierten Angreiferkonzepten (z. B. Innentäter, Externe Angreifer, Technische Akteure).

C. Annahmen und regulative Bedingungen

Operational Security Policies (OSP) formulieren die Einhaltung gesetzlicher Anforderungen oder branchenspezifischer Prozess- oder Verfahrensvorgaben und sind Teil der Problembeschreibung. Daraus leiten sich regulatorische Anforderungen ab, deren Umsetzung und Einhaltung in Form eigenständiger Sicherheitsziele sicher nachzuweisen sind (z. B. Nachweisführung, Zurechenbarkeit und Revisionssicherheit). *Governance*-Strategien führen ebenfalls zu regulatorischen Anforderungen und bestimmen maßgeblich den Aspekt der Prozesssicherheit.

CC-Klassen beschreiben Sicherheitseigenschaften, die durch die Einsatzumgebung vorausgesetzt werden können (*Assumption*). Methoden der CC-Domänenmodellierung erlauben die konkrete Beschreibung einer zu Grunde gelegten operativen Einsatzumgebung.

IV. Sicherheitslösung und Analyse

A. Sicherheitsziele – Ausgangspunkt der Sicherheitsstrategie

Mit dem Wissen einer vorliegenden Sicherheitsproblembeschreibung erfolgt die Ableitung und Darstellung von Sicherheitszielen (*Security Objective*). Die Bestimmung von Sicherheitszielen setzt gleichzeitig den Ausgangspunkt einer strukturiert geplanten Sicherheitsstrategie.

Sicherheitsziele verteilen sich dabei auf die Sicherheitsumgebung (Operational Environment) und auf den zu entwickelnden Sicherheitsbaustein (TOE) selbst. Mit den Instrumenten der CC-Modellklassen erfolgt eine Verknüpfung von definierten Sicherheitszielen mit Einzelaspekten der Sicherheitsproblembeschreibung

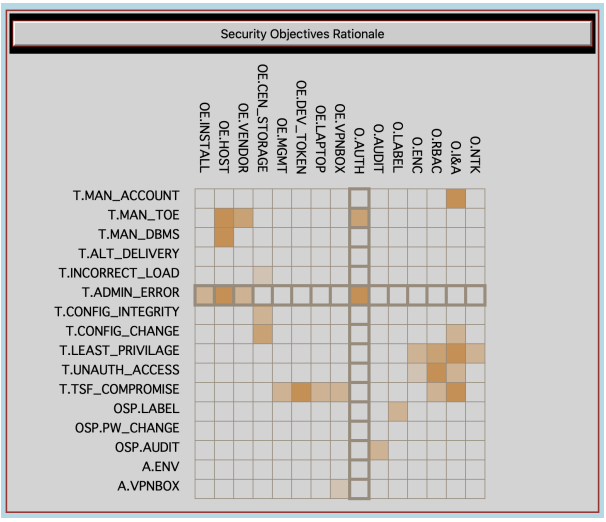

Abbildung 3: Security Objectives Rationale

Als Teil der *Model-Security-Analytic* lässt sich eine interaktive Beziehungsmatrix (Abbildung 3) bedarfsgerecht generieren. Die Visualisierung hebt sämtliche Sicherheitsziele hervor und stellt sie den Aspekten der Problembeschreibung gegenüber. Die farbliche Differenzierung kennzeichnet den Grad der Wirksamkeit eines Sicherheitsziels. Bestimmte Sicherheitsprobleme können nur reduziert werden und benötigen weitere Sicherheitsziele, um sie vollständig auszuräumen.

Einzelaspekte der Sicherheits-Problembeschreibung können geordnet nach *Threats*, *Assumptions* oder *OSPs* aufgelistet, gefiltert und einzeln eingesehen werden.

B. Sicherheitskomponenten – Basis für ein Security-by-Design

Die CC bietet ein strukturiertes Konzept für *Security Functional Components* an. CC-Sicherheitsklassen sind vollständig im Modell als Hierarchie aus *Class*, *Family*, *Component* und *Functional Elements* integriert. Die Modelltransformation kennzeichnet jede Sicherheitskomponente mit einem SFR-*Stereotype* und bezeichnet Modellelemente mit den von der CC verwendeten Akronymen.

Die Phase TOE-security-function specification leitet aus den SFRs konkrete Sicherheitsfunktionen (TSFs) ab. Im Modell

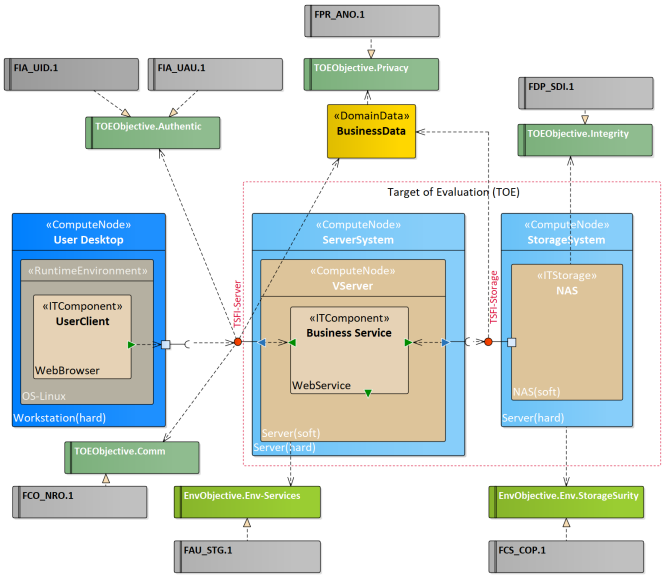

Abbildung 4: Sicherheitsdesign mit CC-Komponenten

erfolgt eine Fortschreibung der Beziehungen zu bestehenden Sicherheitszielen (siehe Nachweise in Abschnitt IV-D).

C. Modellgestützte Analyse und Nachweise zur Evaluation

Die Anwendung der CC-Methodik in einem geschlossenen TOE-Modell ist Grundlage für eine in Echtzeit ablaufende, sicherheitsorientierte Analyse (*Model Security Analytics*).

Ein *Security Design Reasoning* stellt sicher, dass funktionale Erweiterungen nicht *unbemerkt* einen bereits etablierten Sicherheitsstatus nachteilig verändern. Änderungen werden hervorgehoben. Die Auswertung sichert die Nachverfolgung nicht behandelter Sicherheitsprobleme.

Mit Methoden der *Asset Distribution Traceability* entsteht eine transparente Sicht auf die *Asset*-Verteilung innerhalb der betrachteten Sicherheitsarchitektur. Für selektierte *Assets* lassen sich sowohl die zugehörigen SFRs als auch konkrete Sicherheitsfunktionen (SF) einblenden.

D. Generative Nachweise zur Evaluation

Im Rahmen eines BSI geführten Zulassungsverfahrens bestehen hohe Anforderungen an die zu erbringenden Nachweise [2]. Der Nachweiskatalog für die Zulassung von Sicherheitskomponenten für bis zu GEHEIM fordert verschiedene Kategorien zur Nachweiserbringung im Bereich der Architektur, der Entwicklung und in der Teststrategie.

Das *Security Target* (ASE) bildet den umfangreichsten Bestandteil der Nachweise. Die TOE Beschreibung und das CC-konforme Vorgehen zur Beschreibung von Sicherheitsvorgaben sind Bestandteile des Modells.

Das *Security-Target* oder auch Einzelaspekte wie *Security Requirements*, *TOE-summary-specification* oder die *TOE-summary-specification rationale (ASS_TSS)* können generiert und angezeigt werden. Optional besteht die Möglichkeit, ausgewählte Nachweise in Dokumentenform zu exportieren.

References

[1] Bundesamt für Sicherheit in der Informationstechnik. *Common Criteria Startseite*. 2021. URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Zertifizierung-und-Anerkennung/Zertifizierung-von-Produkten/Zertifizierung-nach-CC/IT-Sicherheiskriterien/CommonCriteria/commoncriteria_node.html.

[2] Bundesamt für Sicherheit in der Informationstechnik. *Nachweise für eine Evaluierung für eine Zulassung Teil2: Nachweise für eine Zulassung bis VS-VERTRAULICH / GEHEIM: Sicherheitsanforderungen*. 2015.

[3] Bundesamt für Sicherheit in der Informationstechnik. *Zertifizierung von Produkten*. 2021. URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Zertifizierung-und-Anerkennung/Zertifizierung-von-Produkten/zertifizierung-von-produkten_node.html.

[4] Mike Burrows. *Kanban: Verstehen, einführen, anwenden*. 1. Aufl. s.l.: dpunkt, 2015. ISBN: 978-3-86490-253-6. URL: http://gbv.eblib.com/patron/FullRecord.aspx?p=4350075.

[5] Common Criteria. "Part 1: Introduction and general model: Common Criteriafor Information Technology Security Evaluation". In: 2017 (). URL: https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf.

[6] Common Criteria. "Part 2: Security functional components: Common Criteriafor Information TechnologySecurity Evaluation". In: (). URL: https://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5.pdf.

[7] Claudia Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. 9., aktualisierte Aufl. Berlin: De Gruyter Oldenbourg, 2014. ISBN: 978-3-486-77848-9. DOI: 10.1515/9783486859164. URL: http://www.degruyter.com/search?f_0=isbnissn&q_0=9783486859164&searchTitles=true.

[8] Eduardo B. Fernandez. *Security patterns in practice: Designing secure architectures using software patterns*. 1. ed. Wiley Software Patterns Series. Chichester: Wiley, 2013. ISBN: 9781119970484. URL: http://proquest.safaribooksonline.com/9781119970484.

[9] Don McGreal and Ralph Jocham. *The professional product owner: Leveraging Scrum as a competitive advantage*. The professional scrum series. Boston, Amsterdam, and London: Addison-Wesley, 2018. ISBN: 0134686470.

[10] *Publications : CC Portal: CC v3.1. Release 5: cc3R5.XML*. URL: https://www.commoncriteriaportal.org/cc/.

[11] Adam Shostack. *Threat modeling: Designing for security*. Indianapolis, IN: Wiley, 2014. ISBN: 9781118809990.

[12] *Unified Modeling Language Specification Version 2.5.1: UML 2.5.1 Standard Profile*. URL: https://www.omg.org/spec/UML/2.5.1/.

[13] *Was ist CI/CD?* 2021. URL: https://www.redhat.com/de/topics/devops/what-is-ci-cd.