Automatisierte, minimalinvasive Sicherheitsanalyse und Vorfallreaktion für industrielle Systeme

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

M.Sc.

Florian Patzer

Tag der mündlichen Prüfung:	12.10.2021
Erster Gutachter:	Prof. Dr.-Ing. Jürgen Beyerer
Zweite Gutachterin:	Prof. Dr.-Ing. Anne Koziolek

Abstract

Industrial control and automation systems have been experiencing increasing networking in recent years and consist more and more of components using off-the-shelf software and open standards. In addition to the indisputable advantages that these developments bring with them, they also increase the attack surface of such systems. At the same time, the additional flexibility resulting from this evolution leads to additional complexity in configuration and an increase in exploitable vulnerabilities. The homogeneity of software and hardware also makes the exploitation of these vulnerabilities more attractive to attackers, since less effort has to go into individual attacks. It is therefore not surprising that the number of industrial systems affected by attacks has increased significantly over the last fifteen years. This is particularly worrying because, unlike in other areas (e.g. office-IT), successful attacks on these systems often have a dangerous impact on their environment.

As in other domains with high technological complexity, computer-based processes have become an important part of industrial systems. Among other things, they are used to ensure correct configuration, identification of vulnerabilities, threats and countermeasures, as well as attack detection and response. However, due to the guarantees of industrial systems and their networks regarding aspects, such as real-time processing, fail-safety and redundancy, there are limitations in the use of tools and measures. Thus, in order to interfere with the system as little as possible, security analysis and incident response, for example, must be performed in the least invasive way possible (minimally invasive). For automated security analyses, it has therefore become good practice to create models of the systems and analyse them computer-aided. Knowledge-based or ontology-based approaches have proven to be particularly suitable in this context. However, existing solutions suffer from

problems such as lack of configurability for different environments, lack of optimizability (since usually only certain inference mechanisms are applicable), lack of reusability and interchangeability of model extension steps and analyses, support for different stakeholders and multiple types of analyses such as threat, vulnerability, configuration, and compliance analyses, and lack of technical detail and component coverage to perform certain analyses at all. In the case of incident response, the aforementioned guarantees are even the reason for the lack of solutions deployable in industrial systems to date. This is because the majority of the responses that can be automated are in the area of containment and thus invade the corresponding system in a way that endangers the mentioned guarantees.

This dissertation addresses the outlined security analysis and incident response problems. Concepts and methods have been developed for security analysis that mitigate or solve each of the listed problems. For this purpose the following contributions are presented: a method for modeling and extracting network information from engineering tools based on the open standards AutomationML and OPC UA, results of an investigation on different mapping strategies for creating ontology-based digital twins, a concept for configuration-language-independent model generation for network access control instances, and concepts and methods for reusable, exchangeable, automated model processing and security analysis supporting multiple analysis types. A consistent separation-of-concerns-based framework for knowledge-based security analysis solutions has also been designed, prototyped, and evaluated for these and related concepts and methods. The framework, its implementation and the evaluation results are also presented in this dissertation. All these contributions lead to the first solution to the aforementioned problems and provide a basis for a new type of security analysis that can be collaboratively managed and optimized.

Furthermore, a concept for automated incident response based on the Software-Defined-Networking (SDN) network paradigm is presented. In this context, the playbook approach, which is known from various fields of security, is chosen, whereby a playbook defines predefined reactions to security-relevant events. In addition, explicit modelling of knowledge about endpoints, network

components, and connections to be protected is used in the form of restrictions to individually and automatically constrain the responses defined in the playbooks. The approach also takes advantage of the fact that the network controller has detailed data about the current network topology and leverages the SDN controller's optimized algorithms to reconfigure the data flows. Consequently, the concept presents an approach that, for the first time, makes it possible to use the advantages of automated incident response, such as the short response time and available topology knowledge, in industrial systems.

Kurzfassung

Industrielle Steuerungs- und Automatisierungssysteme erleben in den letzten Jahren eine zunehmende Vernetzung und bestehen mehr und mehr aus Komponenten, bei denen Off-the-Shelf-Software und offene Standards zum Einsatz kommen. Neben den unbestreitbaren Vorteilen, die diese Entwicklungen mit sich bringen, vergrößert sich damit jedoch auch die Angriffsfläche solcher Systeme. Gleichzeitig führt die, durch diese Evolution entstehende, zusätzliche Flexibilität zu zusätzlicher Komplexität in der Konfiguration und einer Zunahme von ausnutzbaren Schwachstellen. Die Homogenität der Soft- und Hardware macht die Ausnutzung dieser Schwachstellen für Angreifer zudem attraktiver, da weniger Aufwand in Individualangriffe fließen muss. Es ist so nicht verwunderlich, dass die Anzahl von Angriffen betroffener industrieller Systeme in den letzten fünfzehn Jahren einen deutlichen Zuwachs erfahren hat. Dies ist besonders bedenklich, weil erfolgreiche Angriffe auf diese Systeme, anders als in der Büro-IT, oft gefährliche Auswirkungen auf ihre Umwelt haben.

Wie auch in anderen Domänen mit hoher technologischer Komplexität, haben sich computergestützte Verfahren zu einem wichtigen Bestandteil industrieller Systeme entwickelt. Sie werden dabei u.a. zur Sicherstellung korrekter Konfiguration, Identifikation von Schwachstellen, Bedrohungen und Gegenmaßnahmen, sowie Angriffsdetektion und -reaktion eingesetzt. Allerdings bestehen aufgrund der Garantien industrieller Systeme und ihrer Netzwerke bezüglich Aspekten wie Echtzeitverarbeitung, Ausfallsicherheit und Redundanz, Einschränkungen im Einsatz von Werkzeugen und Maßnahmen. Um also möglichst wenig in das System einzugreifen, müssen beispielsweise Sicherheitsanalysen und Vorfallreaktionen so wenig invasiv wie möglich (minimalinvasiv) durchgeführt werden. Für automatisierte Sicherheitsanalysen hat es sich

daher zur guten Praxis entwickelt, Modelle der Systeme zu erstellen und diese computergestützt zu analysieren. Als besonders geeignet haben sich in der Forschung dabei wissensbasierte, bzw. ontologiebasierte, Ansätze erwiesen. Existierende Lösungen leiden jedoch unter Problemen wie der fehlenden Konfigurierbarkeit für unterschiedliche Umgebungen, der fehlenden Optimierbarkeit (da in der Regel nur bestimmte Inferenzmechanismen anwendbar sind), der fehlenden Wiederverwendbarkeit und Austauschbarkeit von Modellerweiterungsschritten und Analysen, der fehlenden Unterstützung verschiedener Akteure und mehrerer Analysearten wie Bedrohungs-, Schwachstellen-, Konfigurations- und Konformitätsanalysen, sowie der mangelnden technischen Detailtiefe und Komponentenabdeckung, um bestimmte Analysen überhaupt durchführen zu können. Bei der Vorfallreaktion sind die genannten Garantien sogar der Grund für den Mangel an Lösungen, die in industriellen Systemen eingesetzt werden können. Denn der Großteil der automatisierbaren Reaktionen liegt im Gebiet der Abschottung und greift somit garantiegefährdend in das entsprechende System ein.

In dieser Dissertation werden die eben aufgezählten Probleme der Sicherheitsanalyse und Vorfallreaktion adressiert. Für die Sicherheitsanalyse wurden Konzepte und Methoden entwickelt, die jedes der aufgezählten Probleme mindern oder lösen. Dafür wird unter anderem eine auf den offenen Standards AutomationML und OPC UA basierende Methode zur Modellierung und Extraktion von Netzwerkinformationen aus Engineering-Werkzeugen, Untersuchungsergebnisse verschiedener Abbildungsstrategien zur Erstellung ontologiebasierter Digitaler Zwillinge, ein Konzept zur Sprachenunabhängigen Modellerzeugung für Netzwerkzugriffskontrollinstanzen und Konzepte und Methoden zur wiederverwendbaren, austauschbaren, automatisierten Modellverarbeitung und Sicherheitsanalyse für mehrere Analysearten vorgestellt. Für diese und damit verbundene Konzepte und Methoden wurde zudem ein konsistentes, auf Separation-of-Concerns basierendes Rahmenwerk für wissensbasierte Sicherheitsanalyselösungen entworfen, prototypisch implementiert und evaluiert. Das Rahmenwerk, die Implementierung und die Ergebnisse der Evaluationen werden ebenfalls in dieser Arbeit vorgestellt. Damit wird die erste Lösung für die zuvor genannten Probleme präsentiert und eine Basis für eine neue Art von kollaborativ verwalt- und optimierbaren Sicherheitsanalysen geschaffen.

Des Weiteren wird ein Konzept zur automatisierten Vorfallreaktion auf Basis des Netzwerkparadigmas Software-Defined-Networking (SDN) vorgestellt. Dabei wird ein Ansatz gewählt, der auf vordefinierten Reaktionen auf sicherheitsrelevante Ereignisse basiert und diese über Restriktionen individuell und automatisiert einschränkt. Wobei sich die Restriktionen auf explizit modelliertes Wissen über zu schützende Endgeräte, Netzwerkkomponenten und Verbindungen stützen. Das Konzept nutzt außerdem aus, dass die Netzwerksteuerung durch den SDN-Controller auf detaillierten Daten über die aktuelle Netzwerktopologie verfügt und verwendet die optimierten Algorithmen des SDN-Controllers zur Neukonfiguration. Mit dem Konzept wird ein Ansatz präsentiert, der es erstmals ermöglicht, auch in industriellen Systemen die Vorteile automatisierter Vorfallreaktion, wie die kurze Reaktionszeit und verfügbare Topologiekenntnis, zu nutzen.

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Optronik, Systemtechnik und Bildverarbeitung IOSB in Kooperation mit dem Kompetenzzentrum für angewandte Sicherheitstechnologie KASTEL.

Ich danke allen Personen, die mich zu dieser Arbeit ermutigt und mir während der Erstellung den Rücken gestärkt und freigehalten haben. Zunächst gilt mein Dank meinem Betreuer Herrn Prof. Dr.-Ing. Jürgen Beyerer für die Übernahme des Hauptreferates und seine wertvollen Anregungen bereits vor dem Verfassen der Dissertation. Ebenso gilt er meiner Korreferentin Frau Prof. Dr.-Ing. Anne Koziolek für die konstruktiven Diskussionen, die zu wertvollen Anregungen für die vorliegende Arbeit führten.

An dieser Stelle möchte ich mich auch bei meinem Gruppenleiter, Herrn Dr.-Ing. Christian Haas bedanken, der mir als Sparring-Partner zur Seite stand und mir ermöglichte, meine Forschungsthemen in verschiedenen Forschungsprojekte einzubringen und dort weiterzuentwickeln. Auch möchte ich mich bei allen Kolleginnen und Kollegen des Fraunhofer IOSB bedanken, mit denen ich in den letzten Jahren immer wieder an spannenden Themen forschen durfte und die stets mit offenen Worten und Anregungen zur Qualität unserer Forschung beigetragen haben. Besonders möchte ich mich bei meiner Kollegin Frau Anne Borcherding bedanken, die die Qualität der vorliegenden Arbeit durch detailliertes Hinterfragen und Korrekturlesen merkbar verbessert hat.

Mein herzlichster Dank gilt meiner Frau Isabel, die mir während der Erstellung dieser Arbeit nicht nur den Rücken freigehalten, sondern auch aktiv, mit ihrem eigenen wissenschaftlichen Hintergrund, durch Korrekturlesen, Hinterfragen und wertvolle Anregungen deutlich positiv zur Qualität dieser

Arbeit beigetragen hat. Nicht zuletzt danke ich meinem Hund Ludo, der stets dafür sorgte, dass ich auch in stressigen Phasen Pause- und Frischluftzeiten einhielt, sowie sportlichen Ausgleich nicht vernachlässigte.

Karlsruhe, im Mai 2021
Florian Patzer

Inhaltsverzeichnis

Anhang

Notation

Dieses Kapitel führt die Notationen und Symbole ein, die in dieser Thesis verwendet werden.

Beschreibungslogik

Zur Beschreibung von Ontologien wird in dieser Arbeit die Syntax für Beschreibungslogiken (DL-Syntax) verwendet (vgl. [Hit08]), welche eine kompakte Darstellung der Aussagen einer Ontologie ermöglichen. Die für diese Arbeit benötigten syntaktischen Konzepte werden in Abschnitt 2.7.1 eingeführt und verwenden die folgenden Darstellungsstile:

Klassen	Großer Anfangsbuchstabe mit romanischer Schriftfamilie	Klasse
Relationen/Rollen	Kleiner Anfangsbuchstabe mit romanischer Schriftfamilie in kursiv	*relation*
Individuen	Typewriter Schriftfamilie	`Individuum`
Werte	Typewriter Schriftfamilie kursiv	*`Wert`*
Datentypen	Kleiner Anfangsbuchstabe mit typewriter Schriftfamilie	`datentyp`

Generelle Notation

Skalare	Kleine Buchstaben romanischer Schriftfamilie	x
Mengen	Große griechische Buchstaben	Θ
AutomationML-Konzepte	Typewriter Schriftfamilie	`Element`
Referenzen auf fremde Arbeiten	Nicht kursiv	[Fit07]
Referenzen auf eigene oder betreute Arbeiten	Kursiv	*[Pat21]*

1 Einleitung

Spätestens seit der Entdeckung von Stuxnet[1] in 2010 [Chi10], ist klar, dass industrielle Steuerungs- und Automatisierungssysteme (fortan *industrielle Systeme*) zunehmend Ziel von komplexen Cyber-Angriffen sind. Hinzu kommt, dass die zunehmende Vernetzung industrieller Komponenten, sowie die voranschreitende Adaption von standardisierten Kommunikationsprotokollen und Standardhardware die Angriffsfläche dieser industriellen Systeme erweitert. Grund hierfür sind zusätzliche Eintrittspunkte für Angriffe, abnehmende Individualisierung der Lösungen und eine generelle, fortschreitende Adaption von Informationstechnologien (engl. *Information Technology* (*IT*)) in den von Betriebstechnik (engl. *Operational Technology* (*OT*)) geprägten, industriellen Systemen. Die Konvergenz von IT und OT führt dazu, dass Bedrohungen aus der IT zunehmend auch industrielle Systeme betreffen. Allerdings sind Auswirkungen erfolgreicher Angriffe auf industrielle Systeme besonders prekär, da diese, anders als beispielsweise in der Büro-IT, potenziell physischer Natur sind. So können diese Angriffe beispielsweise zu gefährlichen Fehlfunktionen von Robotern, Pressen oder ähnlichem führen, oder Brände und Explosionen auslösen [Per18].

Um sich bestmöglich vor diesen Bedrohungen zu schützen, setzen Unternehmen verschiedene Gegenmaßnahmen ein. Der sinnvolle, effektive und korrekte Einsatz dieser Gegenmaßnahmen ist jedoch eine große Herausforderung. So muss zunächst herausgefunden werden, welche *Assets* (tangible oder intangible Werte eines Unternehmens) das Unternehmen besitzt und welchen

[1] Stuxnet ist ein über mehrere Jahre unbemerkt gebliebener Cyber-Angriff, der nachweislich zur Manipulation von Urananreicherungszentrifugen eingesetzt wurde.

Bedrohungen diese ausgesetzt sind (festzustellen in einer *Bedrohungsanalyse*). Erst dann können als sinnvoll erachtete technische und organisatorische (Gegen-)Maßnahmen gewählt und umgesetzt werden. Hierzu gehören auch vom Unternehmen definierte Richtlinien, die vorgeben, was von technischen Umsetzungen erwartet wird. Dies können beispielsweise Passwortanforderungen, die Deaktivierung nicht benötigter Software-Dienste oder Einschränkungen des Netzwerkzugriffs sein. Ob diese Richtlinien umgesetzt werden, muss regelmäßig überprüft werden (dieser Vorgang wird *Konformitätsanalyse* genannt). Konformität zu diesen Richtlinien ist jedoch noch kein hinreichender Schutz, da in einem Produktivsystem ggf. Schwachstellen, wie Software- oder Konfigurationsfehler existieren, oder zu einem späteren Zeitpunkt durch Aktualisierung oder Rekonfiguration eingebracht werden. Die Sicherheitsberatungsfirma Applied Risk BV stellte im Laufe des Jahres 2018, während ihrer Untersuchungen verschiedener kritischer industrieller Steuerungs- und Automatisierungssysteme, 768 Schwachstellen fest, von denen 26,7% auf Softwareschwachstellen und 34,7% auf Konfigurationsfehler zurückzuführen waren [App19]. Dies zeigt, dass solche Schwachstellen regelmäßig gesucht (mit Hilfe von *Schwachstellen-* und *Konfigurationsanalyse*) und behoben werden müssen. Die trotz der implementierten Gegenmaßnahmen möglicherweise auftretenden Angriffe, müssen zudem so schnell wie möglich als Angriffe erkannt (durch *Angriffserkennung/-korrelation*) und eingedämmt werden (im Rahmen der *Vorfallreaktion* (auch bekannt als *Incident Response*)).

Die eben beschriebenen Vorgänge finden sich auch in einem typischen Sicherheitslebenszyklus[1] wieder. Einen solchen Sicherheitslebenszyklus zeigt Abbildung 1.1. Er wurde im NIST-Standard 800-37 [Nat18] für Informationssysteme definiert, gilt jedoch ebenso für industrielle Systeme.
Wie die Zuordnung aus Tabelle 1.1 zeigt, finden die zuvor beschriebenen Vorgänge in vier der Lebenszyklus-Phasen statt. In dieser Dissertation werden alle Vorgänge aus Tabelle 1.1 adressiert.

[1] Dabei ist festzuhalten, dass sich der Begriff *Sicherheit* in dieser Dissertation immer auf IT/OT-Sicherheit (bzw. IT/OT-Security) und nicht auf Betriebssicherheit (bzw. Safety) bezieht.

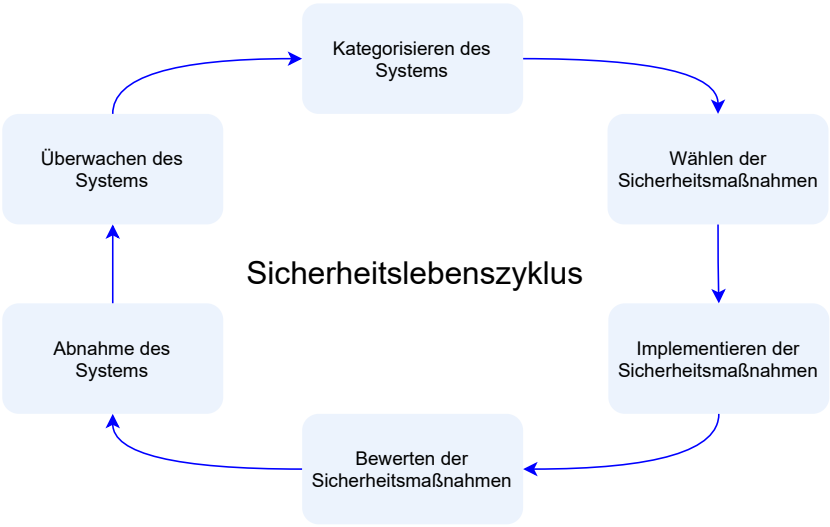


Abbildung 1.1: Sicherheitslebenszyklus für (Informations-)Systeme nach NIST 800-37 [Nat18].

Tabelle 1.1: Zuordnung der Lebenszyklus-Phasen zu Vorgängen.

Phase	Vorgang
Wählen der Sicherheitsmaßnahmen	Schwachstellenanalyse, Bedrohungsanalyse
Bewerten der Sicherheitsmaßnahmen	Konformitätsanalyse, Konfigurationsanalyse
Abnahme des Systems	Konformitätsanalyse
Überwachen des Systems	Schwachstellenanalyse, Angriffserkennung/-korrelation, Vorfallreaktion

Die beschriebenen Sicherheitsanalysen stellen die wesentlichen Untersuchungen der Systemsicherheit und der Ableitung von Handlungsmöglichkeiten dar.

Manchmal werden einige dieser Analysen auch heute noch manuell ausgeführt (beispielsweise basierend auf manueller Durchsicht von Konfigurationen und manuell erstellten Netzwerkplänen). Jedoch ist dieser Ansatz nur für oberflächliche Betrachtungen praktikabel, weshalb für die aufgezählten Analysen auf maschinelle Unterstützung zurückgegriffen werden sollte [Mon11].
Für Sicherheitsanalysen mit hohem Automatisierungsgrad haben sich dabei modellbasierte Ansätze, insbesondere ontologiebasierte Expertensysteme, bewährt [Bla11, Sic15, Fra17b]. Eine *Ontologie* kann dabei als „explizite Spezifikation einer Konzeptualisierung“ betrachtet werden [Gru93].

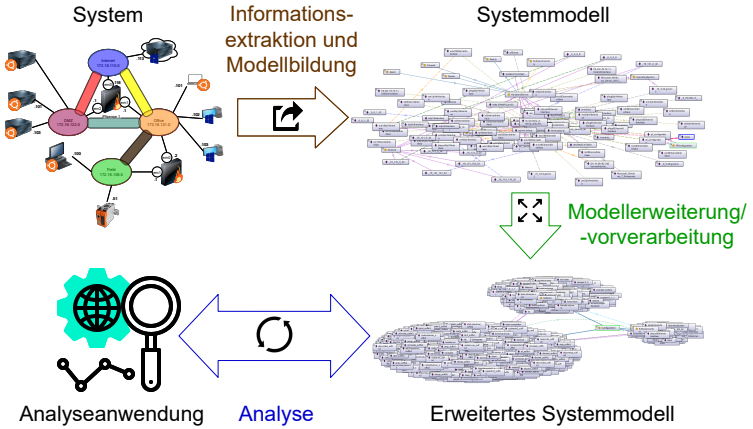


Abbildung 1.2: Generalisierter Ablauf modellbasierter Sicherheitsanalyse.

Wie in Abbildung 1.2 dargestellt, sind bei modellbasierten Ansätzen im Allgemeinen mindestens vier Schritte durchzuführen: *Informationsextraktion* und *Modellbildung*, die zu einem Systemmodell führen, *Modellerweiterung/-vorverarbeitung* zum Anpassen des Modells für die spätere Analyse und die eigentliche *Analyse*.
Gegenüber datengetriebenen Ansätzen, die keine Modelle der Systeme erzeugen, sondern direkt die vorliegenden Daten analysieren, haben modellbasierte Ansätze den Vorteil, Analyselogik in die Modellbildung und -erweiterung zu

transferieren. Dies ermöglicht zum Beispiel den Einsatz von generischen Analysen, leichtgewichtigen Analyseanwendungen, wiederverwendbarer Logik und Daten-agnostischen Programmen zum Ableiten neuen Wissens (sogenannten *Reasonern*).
Sind die eingesetzten Modelle aus Ontologien zusammengesetzt, sind sie besonders für die Erweiterbarkeit und automatisierte Ableitung neuen Wissens durch Reasoner geeignet. Zudem bilden Ontologien verschiedene Domänen ab und können zur semantischen Verknüpfung des Wissens aus verschiedenen Domänen eingesetzt werden.

Der Entscheidung zur Umsetzung von Sicherheitsmaßnahmen geht immer eine Abwägung zwischen dem Risiko einer Bedrohung und dem Aufwand der Umsetzung voraus. Daher sind Systeme nie vollständig sicher. So kommt es in der Realität auch bei besonders gut abgesicherten Systemen zu Angriffen, welche so schnell wie möglich entdeckt werden müssen und auf die im Rahmen der Vorfallreaktion entsprechend reagiert werden muss. Drei der wichtigsten Faktoren bei der Vorfallreaktion sind die Zeiten von der Kompromittierung zur Erkennung des Angriffs (a), von der Erkennung zur Einschränkung des Angriffs (b) und von der Kompromittierung zur Durchsetzung von Gegenmaßnahmen (c) [Bro17]. Verglichen mit der immer noch üblichen manuellen Vorfallreaktion, bietet automatisierte Vorfallreaktion insbesondere eine signifikante Reduktion von (b) (ohne (a) und (c) negativ zu beeinflussen). Die automatisierte Vorfallreaktion basiert in der Regel auf vorkonfigurierten Reaktionsanweisungen. Hierzu gehört beispielsweise das Isolieren einer betroffenen Komponente. Wie in Abschnitt 1.1.2 genauer erläutert wird, können solche Anweisungen in industriellen Systemen jedoch nicht bedingungslos umgesetzt werden. Dieses Problem wird im zweiten Themenblock der vorliegenden Dissertation behandelt.

Im folgenden Abschnitt werden die Probleme, Fragestellungen und Hypothesen dieser Dissertation konkretisiert.

1.1 Probleme, wissenschaftliche Fragen und Hypothesen

Den hier vorgestellten Forschungsarbeiten liegen Problemstellungen zugrunde, welche in den folgenden beiden Themenabschnitten erläutert werden. In diesen Abschnitten werden zudem wissenschaftliche Fragestellungen abgeleitet und Ansätze zu deren Beantwortung in Form von Hypothesen vorgestellt. Belege der Hypothesen werden im Rahmen der Evaluationsbeschreibungen (Abschnitte 3.7 und 4.6) erbracht und diskutiert.

1.1.1 Automatisierte Sicherheitsanalyse

Wie bereits erwähnt, befasst sich der erste Themenblock mit automatisierter, modellbasierter Sicherheitsanalyse. Passend dazu, identifizierten De Franco Rosa und Jino in ihrem 2017 erschienenem Survey zu ontologiebasierten Sicherheitsanalysen[1] [Fra17b, Fra17a] Forschungsbedarfe in den folgenden Bereichen:

1. Analysieren, Verifizieren oder Testen von Sicherheit
2. Identifikation von Schwachstellen
3. Automatisierung von Prozessen
4. Wiederverwendung von Wissen
5. Erweiterung der Analyseabdeckung
6. Sicherer Austausch von Informationen
7. Messung der Sicherheit
8. Schutz von Assets
9. Definition von Sicherheitsstandards

[1] Im Original wird von *Security Assessments* statt Sicherheitsanalysen gesprochen, bei genauerer Betrachtung erschließt sich den Lesenden jedoch, dass die Autoren die entsprechenden in einem Assessment verwendeten Analysen meinen und nicht die, ebenfalls zu einem Assessment gehörende, Bewertung.

Im Rahmen von industriellen Systemen, muss diese Liste um *Nicht-Gefährdung der zu analysierenden Systeme* ergänzt werden (vgl. Abschnitt 3.3). Die behandelten Konzepte und Methoden dieser Dissertation fokussieren offene Probleme der Analyse und Verifikation von Sicherheit im Rahmen automatisierter Sicherheitsanalyse und leisten somit einen direkten Beitrag zu Forschungsbedarf 1. Außerdem werden Schwierigkeiten der Abdeckung von Konfigurations- und Schwachstellenanalysen adressiert, welche unter anderem die Identifikation von Konfigurationsproblemen und bekannter Softwareschwachstellen behindern, wodurch ein Beitrag zu Forschungsbedarf 2 geleistet wird. Weitere Aspekte dieser Dissertation befassen sich zudem intensiv mit den Forschungsbedarfen 3-5 und der Nicht-Gefährdung der zu analysierenden Systeme. Forschungsbedarfe 6-9 liegen nicht im Fokus dieser Dissertation.
In den folgenden Themenabschnitten werden diese Bedarfe in entsprechenden Paragraphen anhand zugrundeliegender Probleme näher beschrieben. Dabei werden entsprechende Forschungsfragen und Hypothesen abgeleitet, die in dieser Dissertation beantwortet und belegt werden. In Abschnitt 3.3.12 werden die genannten Problemstellungen zudem konkretisiert, um den Lesenden die Nachvollziehbarkeit der Probleme mit dem dann verfügbaren Hintergrundwissen zum Stand von Wissenschaft und Technik zu erleichtern.

Automatisierung von Prozessen

Einer der entscheidendsten Nachteile modellbasierter, gegenüber nichtmodellbasierter Ansätze für die Sicherheitsanalyse ist der, vorwiegend manuelle, Aufwand der Informationsextraktion und Modellierung. Für den Erfolg von modellbasierten Ansätzen ist also die Reduktion dieses Mehraufwands in der Informationsextraktion und Modellierung entscheidend. Aus diesem Grund bildet die Automatisierung dieser Schritte eine Grundanforderung an die hier präsentierten Konzepte und Methoden.
Entscheidend für den Erfolg einer automatisierten Lösung ist auch deren Skalierbarkeit, die besonders unter redundanten Modellerweiterungs- und Analyseschritten leidet, die meist einen hohen Ressourcenverbrauch haben.

Aus diesen zwei Problem ergibt sich folgende Forschungsfrage:

Forschungsfrage 1. *Kann eine automatisierte Lösung gefunden werden, die es erlaubt, die Anzahl der durchzuführenden Modellerweiterungen für eine Analyse zu minimieren und durchgeführte Modellerweiterungen gleichzeitig so gut wie möglich für weitere Analysen wiederzuverwenden?*

Zur Beantwortung dieser Frage wir die folgende Hypothese getroffen:

Hypothese 1. *Die Identifikation und Trennung typischer Phasen der Modellerweiterung und -analyse führt sowohl zur Wiederverwendbarkeit abgeleiteter Modellerweiterungen, als auch zur Reduktion nötiger, durchzuführender Modellerweiterungen je Analyse.*

Wiederverwendung von Wissen

Wissen, bei dem eine Wiederverwendbarkeit denkbar und insbesondere zur Ausnutzung verteilter Expertise wünschenswert ist, umfasst unter anderem die folgenden Domänen:

- Strategien zur Erzeugung maschinenlesbarer Darstellung von Systemen und ihrer Sicherheitsinformationen für Sicherheitsanalysen
- Domänenspezifische Regeln zur Ableitung neuen Wissens
- Verfahren zur komplexen Ableitung neuen Wissens
- Best-Practices für Sicherheitsmaßnahmen, -strategien und -architekturen
- Technologische Umsetzung von Richtlinien und Gegenmaßnahmen
- Bekannte Konfigurationsprobleme
- Bekannte Schwachstellen
- Bekannte Bedrohungen
- Bekannte Angriffsmuster
- Semantische und logische Zusammenhänge zwischen den zuvor genannten Wissensdomänen

Die Wiederverwendbarkeit solchen Wissens ist für die automatisierte Sicherheitsanalyse in der Regel auf eine bestimmte Analyseanwendung und eine

bestimmte Analyseart beschränkt oder sogar Teil eines Produktgeheimnisses (vgl. Abschnitt 3.3).
Außerdem kann das Erstellen und Erweitern von Systemmodellen zur Sicherheitsanalyse eine annähernd beliebige Komplexität erreichen und somit tausende Verarbeitungsschritte erfordern. Gleichzeitig eignen sich existierende Ansätze nicht für beliebige Komplexität, weshalb alternative Lösungen benötigt werden.
Zudem müssen jene Personen, welche konkrete Analysen erstellen, bei existierenden Lösungen gleichzeitig Experten aller Domänen sein, die im Modell abgebildet sind und zudem die entsprechende Modellierungsexpertise mitbringen. Dies schränkt den Nutzerkreis drastisch ein und ist nicht für den praktischen Einsatz geeignet.
Bezogen auf die genannten Probleme dieses Abschnittes stellt sich die folgende Forschungsfrage:

Forschungsfrage 2. *Kann eine wissensbasierte Lösung für automatisierte Sicherheitsanalysen gefunden werden, welche die Wiederverwendbarkeit von Wissen zu den zuvor gelisteten Wissensdomänen sicherstellt?*

Die folgende Hypothese wurde zur Beantwortung der Forschungsfrage 2 aufgestellt:

Hypothese 2. Separation-of-Concerns[1] *und der Einsatz von Ontologien kann automatisierte, austauschbare und wiederverwendbare Modellbildung, -erweiterung und -analyse ermöglichen, welche unterschiedliche Domänen für System- und Sicherheitswissen unterstützen.*

Betrachtet man die Informationsextraktion und anschließende Weiterverarbeitung der gerätespezifischen Netzwerkinformationen, die zu einem Gesamtmodell des Systems und seiner Netzwerke führen sollen, ist dies nach wie vor eine große Herausforderung. Insbesondere bleibt, nach aktuellem Stand

[1] Entwurfsprinzip u.a. für Modelle und Programme, bei dem diese in separate, in sich abgeschlossene Teile unterteilt werden, die jeweils unterschiedliche Anliegen adressieren.

von Wissenschaft und Technik, das Ableiten eines solchen Modells, unter der Betrachtung verschiedener, sich ggf. widersprechender, Konfigurationen von Endgeräten und Netzwerkzugriffssteuerungskomponenten (*Network Access Control (NAC)*), ein ungelöstes Problem. Bei gegenwärtigem Stand der Wissenschaft und Technik muss die Analyse die nötige Logik zur Interpretation der NAC-Konfigurationen enthalten. So ergibt sich sowohl ein Skalierungsproblem, als auch eine Vermischung der Wissensdomänen der Sicherheitsanalyse und der NAC-spezifischen Konfigurationssprache, sodass die Wiederverwendbarkeit der Analysen und Modellerweiterungen stark begrenzt wird (vgl. Abschnitt 3.4.2.1).
Hieraus lässt sich die folgende Forschungsfrage ableiten:

Forschungsfrage 3. *Lässt sich durch alternative, automatisierte Interpretations- und Modellierungsverfahren ein Netzwerkmodell erzeugen, welches sowohl die Netzwerkendpunkt- und NAC-spezifischen Konfigurationen, als auch deren Beziehungen zueinander enthält und dabei die Wiederverwendbarkeit von Sicherheitsanalysen und Modellerweiterungen steigert?*

Die folgende Hypothese wurde zur Beantwortung der Forschungsfrage 3 aufgestellt:

Hypothese 3. *Durch die automatisierte Vorinterpretation von NAC-Konfigurationen und die Modellierung der so abgeleiteten, effektiven Konfiguration kann ein Netzwerkmodell erzeugt werden, welches sowohl die Endpunkt- und NAC-spezifischen Konfigurationen, als auch deren Beziehungen zueinander enthält und dabei die Wiederverwendbarkeit und Effizienz von Sicherheitsanalysen und Modellerweiterungen erhöht.*

Nicht-Gefährdung der zu analysierenden Systeme

Die wünschenswerte automatisierte Informationsextraktion für industrielle Systeme wirft weitere Probleme auf. Der klassische Ansatz des Scannens, der im Büro-IT-Bereich u.a. oft eingesetzt wird, wenn es an aktuellen Inventar- und Konfigurationsquellen fehlt (vgl. nächsten Paragraphen „Erweiterung der

Analyseabdeckung“) oder Komponenten unbekannt sind, ist in industriellen Systemen nicht sicher einsetzbar [Dug05].
Folglich wird ein weniger invasiver Ansatz benötigt, der die zu untersuchenden Systeme nicht gefährdet und deren Beeinträchtigung minimiert. Ein solcher Ansatz wird im Rahmen dieser Arbeit als *minimalinvasiv* bezeichnet. Der Terminus berücksichtigt dabei, dass auch Alternativen zu Scanning in das Netzwerk eingreifen, falls sie Datenübertragungen auf dem schützenswerten Netzwerk oder Rechenaufwand auf den Assets erfordern. Auch die Minimalinvasivität der Analysen und vorbereitender Prozesse ist eine Grundanforderung, die den Rahmen der in dieser Dissertation vorgestellten Arbeiten schärft.

Erweiterung der Analyseabdeckung

Gerade durch die noch junge Digitalisierung bei industriellen Systemen und die meist sehr eingeschränkte Software der entsprechenden Komponenten, ist die Verfügbarkeit maschinenlesbarer Informationen zu den Komponenten und ihren Konfigurationen sehr beschränkt. Gerade Informationsquellen, die von Programmen und Betriebssystemen aus der Büro-IT angeboten und von entsprechenden Analyselösungen in dieser Domäne eingesetzt werden, sind für industrielle Systeme nicht verfügbar. Dazu zählen Konfigurationsschnittstellen und *Selbstauskunft*-Anwendungen, also Software, deren Zweck es ist, die Zustände und Konfigurationen der Komponenten, auf denen sie ausgeführt wird, preiszugeben. Diese Einschränkung hat deutlich negative Auswirkungen auf die Analyseabdeckung, da Analysen nicht ohne die zugrundeliegenden Komponenteninformationen durchgeführt werden können, bzw. mit abnehmender Informationsabdeckung schwindende Aussagekraft haben.
Aus diesem Problem lässt sich eine weitere Forschungsfrage ableiten:

Forschungsfrage 4. *Wie lassen sich sicherheitsrelevante Informationen über industrielle Komponenten automatisiert sammeln, die von minimalinvasiven Extraktionsmöglichkeiten bisher nicht abgedeckt werden?*

Zu Forschungsfrage 4 wurde die folgende Hypothese aufgestellt:

Hypothese 4. *Die offene, werkzeugübergreifende Sprache AutomationML kann genutzt werden, um die automatisierte Extraktion sicherheitsrelevanter Informationen für industrielle Komponenten zu ermöglichen, die weder Selbstauskunft noch standardisierte Konfigurationsschnittstellen bieten.*

Selbst durch den Nachweis der Hypothese 4 verbleibt das Problem, dass die hohe Heterogenität der Informationsrepräsentation eine breite Abdeckung industrieller Komponenten derzeit noch behindert. Neue Konzepte wie die Verwaltungsschale (vgl. Abschnitt 2.1.1), die ein standardisiertes, digitales Abbild eines Assets zur Verfügung stellen soll, adressieren dieses Problem.
Somit stellt sich die folgende Forschungsfrage:

Forschungsfrage 5. *Wie kann die Verwaltungsschale als Informationsextraktionskonzept zur Sicherheitsanalyse in industriellen Systemen eingesetzt werden?*

Dieser Frage folgt keine Hypothese da sie eine generelle Untersuchung erfordert. In Abschnitt 3.4.1.2 wird eine solche Untersuchung vorgestellt.

Ein weiteres Problem ist die Tatsache, dass bisher für alle Sicherheitsanalysearten spezifische monolithische, spezialisierte Lösungen entwickelt wurden. Die Informationsextraktion, Modellbildung und Modellerweiterung ist jedoch für alle hier behandelten Analysearten durchzuführen und das zu modellierende und auszuwertende Expertenwissen aller Analysearten birgt große Überschneidungen. Solche Synergien zwischen den Analysearten können bisher nicht ausgenutzt werden.
Folglich stellt sich die folgende Forschungsfrage:

Forschungsfrage 6. *Lassen sich die verschiedenen Analysearten Schwachstellenanalyse, Konfigurationsanalyse, Konformitätsanalyse, Angriffserkennung, Angriffskorrelation und Bedrohungsanalyse so zusammenfassen, dass deren Synergien ausgenutzt werden können?*

Zur Beantwortung dieser Frage wurde die folgende Hypothese aufgestellt:

Hypothese 5. *Durch Separation-of-Concerns, den Einsatz von Ontologien und einer geschickten logikbasierten Umsetzung von Analysen, kann ein Analysesystem für mehrere gängige Analysearten geschaffen werden, welches die Ausnutzung von Synergien unterstützt.*

1.1.2 Automatisierte Vorfallreaktion für industrielle Systeme

Eine weitere, in dieser Dissertation adressierte, Problematik bezieht sich auf automatisierte Vorfallreaktion.
Existierende Lösungen für die Vorfallreaktion können nicht unmittelbar für industrielle Umgebungen eingesetzt werden. Der Grund hierfür ist die kontextabhängige Gefahr, die von automatisierten Eingriffen in das Netzwerk ausgeht. Konkret unterliegen industrielle Systeme einer Vielzahl zu sichernder Eigenschaften, wie Echtzeit-, Betriebssicherheits-, Verfügbarkeits- und Redundanzgarantien. In das Netzwerk eingreifende Maßnahmen führen in vielen Fällen zur Verletzung solcher Garantien und können somit nicht einfach von IT-Umgebungen übernommen werden.
In der Leitline für die Sicherheit in industriellen Steuersystemen (*Industrial Control Systems (ICS)*) NIST 800-82 [Nat11] steht über Lösungen zur automatisierten Vorfallreaktion:

> *"These systems have the ability to automatically reconfigure systems if an intrusion attempt is identified. This automated and fast reaction is designed to prevent successful exploits; however, an automated tool such as this could be used by an adversary to adversely affect the operation on an ICS by shutting down segments of a network or server. False positives can also hinder ICS operation.",* [Nat11]

Sinngemäß bedeutet die Aussage, dass automatisierte Vorfallreaktion von Angreifern ausgenutzt werden kann, den Betrieb von industriellen Umgebungen zu beeinflussen und dass fälschlich detektierte Angriffe ebenso den Betrieb stören können.

Gleichzeitig sind Lösungen für automatisierte Vorfallreaktion für industrielle Systeme weitgehend unerforscht (vgl. Abschnitt 4.1). Somit stellt sich die folgende Forschungsfrage:

Forschungsfrage 7. *Wie können automatisierte Vorfallreaktionslösungen angepasst werden, um deren Einsatz in industriellen Umgebungen bei korrekter Konfiguration zu ermöglichen?*

Im Rahmen dieser Dissertation wurde bezüglich Forschungsfrage 7 die folgende Hypothese untersucht:

Hypothese 6. *Eine Kombination aus der Modellierung von Systemgarantien und deren Beziehungen zu Netzwerkteilnehmern, Kommunikationsverbindungen und Netzwerkkomponenten, sowie durch die Verwendung von Drehbüchern*[1] *(engl. Playbooks) und restriktiven Regeln, ermöglicht den Einsatz automatisierter Vorfallreaktion in industriellen Systemen.*

1.2 Technologische Zielsetzung und eigene Beiträge

In diesem Abschnitt werden die technologischen Ziele zur Erarbeitung der hier adressierten Konzepte und Methoden, sowie die eigenen Beiträge zur Beantwortung der Fragestellungen und zum Nachweis der Hypothesen aus Abschnitt 1.1 beschrieben.

1.2.1 Technologische Ziele

Abgeleitet aus den Fragestellungen und Hypothesen aus Abschnitt 1.1 standen im Forschungsverlauf die folgenden technologischen Ziele im Vordergrund, deren Erreichung in den Abschnitten 3.7.8 und 4.6.5 diskutiert wird:

[1] vgl. https://www.incidentresponse.com/playbooks/, zuletzt zugegriffen: 19.08.2020.

1. Erhöhung der Anzahl auffindbarer Schwachstellen, Fehlkonfigurationen, Inkonsistenzen, Bedrohungen, Konformitätsprobleme und Angriffe.
    a) Erweiterung der Geräteabdeckung für die Informationsextraktion.
    b) Austauschbarkeit von Netzwerkstrategien.
    c) Austausch- und Wiederverwendbarkeit von Analysemodellerweiterung.
    d) Austausch- und Wiederverwendbarkeit von Analysen.
2. Steigerung der Effizienz von Analyselösungen.
    a) Steigerung der Automatisierung der Analysemodellerzeugung.
    b) Steigerung der Automatisierung der Analysemodellerweiterung.
    c) Austausch- und Wiederverwendbarkeit von Analysemodellerweiterung.
    d) Austausch- und Wiederverwendbarkeit von Analysen.
3. Konzeptionierung eines flexiblen, wissensbasierten Rahmenwerks mit hohem Automatisierungsgrad für die Unterstützung der heterogenen Analyselandschaft automatisierter Angriffserkennung und -korrelation, sowie automatisierter Schwachstellen-, Konfigurations-, Konformitäts- und Bedrohungsanalysen, mit Bezug auf die Sicherheit industrieller Systeme.
4. Ermöglichung des Einsatzes von automatisierter Vorfallreaktion in industriellen Systemen.

Dabei ist zu berücksichtigen, dass sowohl Ziele 1.c) und 2.c) als auch Ziele 1.d) und 2.d) unterschiedliche Aspekte der selben Teilziele beschreiben, die entsprechender separater Evaluation bedürfen. Zudem adressiert Ziel 3 sowohl die Problematik der starken Beschränktheit bisheriger Lösungen auf spezifische Modellerweiterungen, Analysearten und Informationsquellen, als auch die fehlende Möglichkeit der Rekonfiguration und Ausrichtung am Sicherheitslebenszyklus.

1.2.2 Eigene Beiträge

Die folgenden eigenen Beiträge wurden zum Erreichen der aufgeführten Ziele und zum Nachweis der in Abschnitt 1.1 gelisteten Hypothesen geleistet. Eigene Veröffentlichungen sind den Themenabschnitten zugeordnet und werden, wie auch die vom Autor dieser Dissertation betreuten Abschlussarbeiten, kursiv referenziert, sodass den Lesenden die Unterscheidung zwischen eigenen und fremden Arbeiten erleichtert wird. Die Beiträge werden im Verlauf dieser Dissertation genau erläutert und durch die Evaluationen in Abschnitten 3.7 und 4.6 detailliert mit den zuvor beschriebenen Fragestellungen, Hypothesen und Zielen in Bezug gesetzt.

Informationsextraktion aus Engineering- und Projektierungswerkzeugen [Pat17, Pla18]

Es wurde eine Methode und Vorlage zur Modellierung von Netzwerkeigenschaften in der als offener Standard verfügbaren Modellierungssprache *AutomationML* (vgl. Abschnitt 2.2) entwickelt und evaluiert. Die Methode bildet die, nach bestem Wissen des Autors, erste Engineering-werkzeugübergreifende Exportiermöglichkeit von Netzwerkinformationen industrieller Komponenten, welche keiner anderen Quelle als Engineering- und Projektierungswerkzeugen entnommen werden können.

Abbildungsverfahren [Pat19d]

Es wurden verschiedene Abbildungsverfahren zur praxistauglichen Übersetzung von Informationsmodellen des industriellen Kommunikationsprotokolls *Open Platform Communications Unified Architecture (OPC UA)*

(vgl. Abschnitt 2.3) in *OWL-2-DL*[1]-Ontologien (vgl. Abschnitt 2.7.1) entwickelt und untersucht. Dies beinhaltet die Entwicklung eines *YAML*[2]-Konfigurationsschemas, mit dessen Hilfe ein eigens entwickeltes Python-Modul komplexe Abbildungen von OPC UA (sowie verschiedener weiterer Quellen) auf OWL 2 DL durch eine Kombination von Schemaabbildung, regulären Ausdrücken und Vor- bzw. Nachbearbeitungsroutinen realisiert. Damit wurde die erste Analyse von praxisrelevanten Strategien zur Abbildung von OPC UA zu OWL 2 DL präsentiert und eine Lösung für automatisierte, komplexe OPC UA zu OWL 2 DL Abbildungen vorgestellt. Diese ermöglicht unter anderem die automatisierte Erzeugung Digitaler Zwillinge mithilfe der populären, als sicher angesehenen [Asc16], industriellen Schnittstelle OPC UA.

Repräsentation effektiver Konfiguration [Pat21]

Es wurde ein Algorithmus und eine Modellierungsstrategie zur Ableitung einer generischen OWL-2-DL-basierten Repräsentation von Netzwerkzugriffskontrollkonfigurationen entwickelt und evaluiert. Durch diese Repräsentation werden NAC-übergreifende Analysen stark vereinfacht und somit nachweislich effizienter. Auch fördert die Repräsentation die Wiederverwendbarkeit von Modellerweiterungen und Analysen, da sie eine individuelle Interpretation der konkreten NAC-Konfigurationen innerhalb der Erweiterungen und Analysen obsolet macht.

Regelbasierte Sicherheitsanalyse für verschiedene Analysearten[3]

Es wurde ein Konzept zur Vereinheitlichung verschiedener Sicherheitsanalysetypen entwickelt und evaluiert. Dieses ermöglicht, zusammen mit der im

[1] OWL 2 DL ist eine Ontologiesprache, die in Abschnitt 2.7.1 erläutert wird.
[2] YAML ist ein, für Lesbarkeit entwickelter Standard zur Datenserialisierung (vgl. https://yaml.org zuletzt zugegriffen: 10.10.2020).
[3] Eine Veröffentlichung zu Analyse-Konzept und Rahmenwerk war bei Fertigstellung dieser Dissertation in Arbeit, jedoch noch nicht eingereicht.

nächsten Paragraphen angesprochenen Gesamtlösung, die Ausnutzung von Synergien zwischen den Analysearten.

Rahmenwerk für ontologiebasierte Systemanalysen [Pat19b]

In dieser Dissertation werden zusätzlich zu den bereits aufgezählten Beiträgen, weitere Teilkonzepte und Methoden zu den in Abschnitt 1.1.1 beschriebenen Problemen vorgestellt. All diese Konzepte und Methoden wurden verwendet, um eine konsistente Gesamtlösung zu konzipieren. Die Gesamtlösung wird in Form eines Rahmenwerks, mit dem Namen *SyMP* (das für *System Model Processing* steht), vorgestellt. Dieses wird in Abschnitt 3.5 beschrieben und in Abschnitt 3.7 evaluiert. Mithilfe des SyMP-Rahmenwerks können darauf basierende Implementierungen effizient und strukturiert für beliebige Umgebungen konfiguriert werden. Durch die konsistente Unterstützung der zuvor beschriebenen Beiträge, durch die im Rahmenwerk beschriebene Architektur und Methodik, wird die Austauschbarkeit (bzw. Teilbarkeit) von System-/Konfigurationsinformationen, Modellerweiterungen, Standard- und Best-Practice-Interpretationen und Analysen erreicht und somit deren Abdeckung, Optimierung und Qualität erhöht. Auch werden durch das Rahmenwerk und das zuvor erwähnte Konzept der regelbasierten Sicherheitsanalyse verschiedene systemmodellbasierte Analysearten gleichzeitig unterstützt und die Ausnutzung ihrer Synergien ermöglicht. Zudem fallen durch das Rahmenwerk verschiedene Integrationsprobleme weg (vgl. Abschnitt 3.4.2.1) und Werkzeuge zum Sicherheitsmanagement können stark vereinfacht werden, da sich die Analysen eine Informationsbasis teilen können.

Kontext-sensitive Vorfallreaktion [Pat19c, Pat20, Pat19a]

Um den verschiedenen Einschränkungen automatisierter Vorfallreaktionen in industriellen Systemen gerecht zu werden, wurde eine Methode zur expliziten Definition des Kontextes eines Systems auf Basis von Asset- und Restriktionsklassen erarbeitet, mit deren Hilfe ein ebenfalls entwickeltes Regel-basiertes Konzept frei definierbare Reaktionsabläufe (definiert in einem sogenannten

Drehbuch) filtert. So werden unerwünschte und das betreffende System gefährdende Reaktionen vermieden und Drehbuch-basierte Ansätze zur automatisierten Vorfallreaktion können in industriellen Systemen eingesetzt werden. Dieses Konzept wurde auf Basis des Paradigmas *Software-Defined Networking (SDN)* (vgl. Abschnitt 2.9) entwickelt und sowohl alleinstehend, als auch als Kernkonzept in einer Sicherheitsstatus-basierten Netzwerkmanagementlösung evaluiert.

1.2.3 Abgrenzung

In der Natur der hier behandelten Themen liegt die Existenz einer Vielzahl von Schnittstellen zu und Überlappungen mit verschiedenen Forschungsthemen. Die folgenden Abgrenzungen sind daher zu treffen.

Ziel der Forschungsarbeiten war es nicht, existierende ontologiebasierte Ansätze vollständig zu ersetzen. Stattdessen lag der Fokus auf der Lösung der bereits vorgestellten Probleme (vgl. Abschnitt 1.1.1) und der Erarbeitung einer ersten Gesamtlösung, welche die in Abschnitt 3.2 abgeleiteten Anforderungen erfüllt. Dabei können viele der existierenden Ansätze wiederverwendet, mit der resultierenden Lösung umgesetzt und damit verbessert werden.

In Abschnitt 3.4.3.1 werden Gemeinsamkeiten von Kernaspekten verschiedener Analysearten genutzt, um eine Generalisierung abzuleiten, welche die Entwicklung von Werkzeugen ermöglicht, die diese verschiedenen Analysetypen gleichzeitig unterstützen. Das dabei resultierende Konzept von regelbasierten Analysen abstrahiert von formalen Regelsprachen, welche die Aussagen abbilden müssten, die sonst nur der natürlichsprachigen Definition durch Sicherheitsexperten zu entnehmen sind. Eine passende Sprache für diesen Anwendungsfall zu definieren lag nicht im Fokus der hier präsentierten Forschungsarbeiten. Im Gegenteil - die hier vorgestellten Konzepte sind so entworfen, dass der Einsatz beliebiger solcher Sprachen und beispielsweise auch Ansätzen zur automatischen Überführung natürlicher Sprache in Regeln (z.B. mittels *Natural Language Processing (NLP)*) unterstützt wird, da die Regeldefinitionen in maschineninterpretierbarer Form definiert werden (vgl. Abschnitt 3.4.3.1).

In dieser Dissertation werden nicht-verhaltensbasierte Analysen und Analysen zur Erkennung von Angriffen betrachtet. Verhaltensbasierte Analysen, wie Fuzzing, Scanning und Probing sind nicht Teil der Arbeit, da diese nicht für industrielle Systeme im Produktivbetrieb eingesetzt werden sollten (vgl. Abschnitt 1.1.1).

Da die Arbeit auf Deutsch verfasst wird, ist es zudem wichtig hier noch einmal darauf hinzuweisen, dass in dieser Dissertation mit dem Begriff *Sicherheit* immer IT/OT-Sicherheit (engl. IT/OT-Security) und nicht die *Betriebssicherheit* (engl. Safety) gemeint ist.

1.3 Aufbau der Dissertation

Hier soll nun beschrieben werden, wie der Rest dieser Dissertation strukturiert ist. Zunächst werden in Kapitel 2 einige Grundlagen vermittelt, die für das Verständnis der darauffolgenden Kapitel nötig sind.
Dem folgend, wird in Kapitel 3 das Thema der modellbasierten Sicherheitsanalyse adressiert. Dafür werden zunächst im Laufe der Dissertation eingesetzte Beispielumgebungen vorgestellt und ein Sicherheitslebenszyklus-basiertes Anwendungsszenario für die Sicherheitsanalyse beschrieben. Daraufhin werden Anforderungen für adäquate Lösungen, die auch solch ein Szenario unterstützen, erhoben (Abschnitt 3.2). Anschließend wird auf den Stand von Wissenschaft und Technik eingegangen (Abschnitt 3.3) und dessen Abdeckung der aufgestellten Anforderungen analysiert. Nach den verwandten Arbeiten werden einzelne erarbeitete Konzepte und Methoden zur Lösung vorgestellter Probleme der automatisierten, minimalinvasiven Sicherheitsanalyse erörtert (Abschnitt 3.4). Schließlich präsentiert Abschnitt 3.5 ein entsprechendes Gesamtkonzept in Form des SyMP-Rahmenwerks, welches die Einzelkonzepte auf konsistente Weise vereint und ergänzt. Die Implementierung dieses Rahmenwerks wird danach in Abschnitt 3.6 beschrieben. Unter Einsatz dieser Implementierung wird das Rahmenwerk schließlich in Abschnitt 3.7 evaluiert. Dabei wird gezeigt, wie die definierten Ziele erreicht, Anforderungen erfüllt, aufgestellten Hypothesen belegt und Forschungsfragen beantwortet wurden. Als Nächstes widmet sich Kapitel 4 der automatisierten Vorfallreaktion. Auch

hier werden zunächst verwandte Arbeiten diskutiert (Abschnitt 4.1) und im Anschluss Anforderungen für eine geeignete Lösung erhoben (Abschnitt 4.2). Danach werden wichtige Vorüberlegungen getroffen (Abschnitt 4.3), bevor in Abschnitt 4.4 ein Konzept für minimalinvasive, automatisierte Vorfallreaktion vorgestellt wird. Die Implementierung und Evaluation dieses Konzepts wird in den darauffolgenden Abschnitten 4.5 und 4.6 beschrieben.
Zuletzt wird die Dissertation zusammengefasst und ein Ausblick auf weiterführende Forschungsarbeiten gegeben (vgl. Kapitel 5).

2 Grundlagen

In diesem Kapitel werden die Grundlagen vermittelt, die zum Verständnis der weiteren Kapitel benötigt werden.

2.1 Industrielle Systeme und Industrie 4.0

Die dritte industrielle Revolution beschrieb die Phase der sich immer weiter ausbreitenden, computergestützten Automatisierung und der zunehmenden Verbreitung von IT in den Unternehmen. Die aktuelle, vierte industrielle Revolution (*Industrie 4.0*) bezieht sich technologisch hingegen auf die Ausbreitung von *Industrial Internet of Things* (*IIoT*) und die zunehmende Vernetzung von Maschinen, auch über Unternehmensgrenzen hinweg *[Pla20]*. Damit einhergehend bestehen moderne industrielle Systeme aus *Cyber-physischen Systemen* (*CPS*) und unterliegen einer zunehmenden Konvergenz von IT und OT. Typische Komponenten industrieller Systeme sind beispielsweise speicherprogrammierbare Steuerungen (*SPS*, engl. *PLC*), intelligente elektronische Geräte (*IED*), Buskoppler oder andere Gateways, Mensch-Maschine-Schnittstellen (engl. *Human Machine Interface* (*HMI*)) und Netzwerkkomponenten, wie Switches und Router. Aber auch Lösungen für die Planung der Unternehmensressourcen (engl. *Enterprise Resource Planning* (*ERP*)), Historiendatenbanken und die Anbindung von Gateways an Cloud-Lösungen für Analysen und Logistik sind in modernen industriellen Systemen bereits allgegenwärtig und zeigen die Konvergenz von IT und OT. Mehr Einblick in ein modernes, typisches Produktionssystem liefert die Beschreibung des Demonstrator-Aufbaus in Abschnitt 3.1.1.
Industrielle Systeme sind technologisch hochgradig heterogen und erleben

erst seit wenigen Jahren eine zunehmende offene Standardisierung von Protokollen und anderer Technologien, mit dem Ziel der Interoperabilität. Mit einer typischen Betriebsdauer industrieller Komponenten von über 20 Jahren benötigen Transformationen zu diesen interoperablen Lösungen wesentlich mehr Zeit, als in der Büro-IT, wo eine typische Betriebsdauer bei drei Jahren liegt. Zudem sind Aktualisierungen oder Software-technische Erweiterungen industrieller Komponenten nur selten möglich, da diese Systeme in der Regel dauerhaft im Einsatz sind und nur seltene und kurze Stillstandszeiten auftreten, z.B. durch geplante Wartungsarbeiten. Ausfälle sind anders als in der Büro-IT nicht tolerierbar. Auch haben industrielle Systeme besondere Anforderungen in der Betriebssicherheit und benötigen Garantien im Rahmen von Echtzeitkommunikation und -verarbeitung.

Industrie 4.0 führt zu einer Vielzahl neuer technologischer Konzepte und Entwicklungen, z.B. im Rahmen der Digitalisierung und Interoperabilität. Arbeitsgruppen der vom *Bundesministerium für Wirtschaft und Entergie* (*BMWi*) und vom *Bundesministerium für Bildung und Forschung* (*BMBF*) geleiteten und geförderten *Plattform Industrie 4.0*[1] haben sich der Entwicklung solcher Konzepte verschrieben. Zwei der populärsten Konzepte sind die *Verwaltungsschale* (engl. *Asset Administration Shell*) und der *Digitale Zwilling* (engl. *Digital Twin*), welche in folgendem Abschnitt näher erläutert werden.

2.1.1 Verwaltungsschale und Digitaler Zwilling

Die Verwaltungsschale ist ein Konzept unter dem eine digitale oder virtuelle Repräsentation eines Assets zu verstehen ist [Pla15]. Dafür definiert sie ein Metamodell, welches als Grundstruktur der Informationen in einer Verwaltungsschale dient, setzt für die Semantik von Daten auf Referenzen zu externen Semantiken, bspw. *ECLASS*[2], und spezifiziert, dass beliebige Domänen durch sogenannte Teilmodelle abgebildet werden [Pla19]. Zudem werden für die Verwaltungsschale verschiedene Modellierungs-/Darstellungsformen,

[1] https://www.plattform-i40.de/, zuletzt zugegriffen: 06.05.2021.
[2] https://www.eclass.eu/standard.html, zuletzt zugegriffen: 26.11.2020.

wie *OPC UA Nodesets* (vgl. Abschnitt 2.3), AutomationML (vgl. Abschnitt 2.2) oder *RDF*[1] und Übertragungsarten, wie HTTPS, E-Mail oder OPC UA vorgeschlagen [Pla19].

Die Grundidee hinter dem oft zitierten Terminus „Digitaler Zwilling“ ist hierzu recht ähnlich. Demnach besteht ein Digitaler Zwilling aus einem physischen Produkt im realen Raum, einem virtuellen Produkt im virtuellen Raum und den Verbindungen von Daten und Informationen, welche das physische und virtuelle Produkt aneinander binden [Gri15]. Diese ursprüngliche Definition von Dr. Michael Grieves aus dem Jahr 2003, wurde bis zum Jahr 2020 in verschiedener Weise weiterentwickelt. So bezieht man mittlerweile oft auch die Simulationen im virtuellen Raum und die automatisierte Synchronisation inklusive entsprechender Schnittstellen zwischen physischem und virtuellen Produkt, sowie beliebige „Dinge“, statt lediglich Produkte, in den Begriff mit ein [Bos20, Sta20].

In der Umsetzung wurden Digitale Zwillinge oft mit proprietären Informationsmodellen und Formaten entwickelt und bilden so Informationssilos, verhindern also die Verwendung der Informationen über die Grenzen der initialen Umsetzung hinaus [Bos20]. Genau dieses Interoperabilitätsproblem Digitaler Zwillinge wird durch die Verwaltungsschale adressiert, was nach [Bos20] ihren Einsatz als interoperable Informationsgrundlage für Digitale Zwillinge motiviert.

2.2 AutomationML

Für das Verständnis der hier vorgestellten Arbeiten im Bereich der Darstellung von sicherheitsrelevanten Netzwerkinformationen in AutomationML (*Automation Markup-Language*), wird die Sprache hier basierend auf dem Dokument *„AutomationML in a Nutshell“* [Lüd16] vorgestellt.
Engineering von Produktionssystemen ist sehr komplex und erfordert

[1] https://www.w3.org/RDF/, zuletzt zugegriffen: 10.04.2021.

verschiedene Engineering-Werkzeuge, die in verschiedenen Engineering-Phasen eingesetzt werden. Ein großes Problem hierbei ist, dass Engineering-Aktivitäten wiederholt werden müssen, weil kein einheitlicher Datenaustausch zwischen den Engineering-Werkzeugen existiert.
Hier setzt AutomationML an. Mit AutomationML wurde, basierend auf der *Extensible Markup Language*[1] (*XML*), eine werkzeugübergreifende, in IEC 62714 [Int18] standardisierte Repräsentation von Engineering-Daten entwickelt. Die grundlegende Struktur von AutomationML basiert auf *Computer Aided Engineering Exchange* (*CAEX*) [Int16] und wird in Abbildung 2.1 dargestellt. CAEX erlaubt die Festlegung bestimmter Semantiken zu Systemobjekten mittels *Rollenklassen* (engl. *Role Class* (*RC*)), die in Rollenklassenbibliotheken definiert werden. Schnittstellen zwischen Systemobjekten werden unter Nutzung von *Schnittstellenklassen* (engl. *Interface Class* (*IC*)) modelliert, die in Schnittstellenklassenbibliotheken definiert werden. Klassen der Systemobjekte selbst, werden als *Systemobjektklassen* (engl. *System Unit Class* (*SUC*)) in Systemobjektklassenbibliotheken definiert. Laut [Lüd16] dienen alle modellierten Klassen der semantisch eindeutigen Modellierung der eigentlichen Projektinformationen. Hier muss jedoch berichtigt werden, dass es sich weniger um semantische Eindeutigkeit, als um eine semantische Vorlage für Instanzen handelt, die dem Standard folgend weder validiert noch erzwungen wird. Diese Vorlagen werden dann in mindestens einer *Instanzhierarchie* (engl. *Instance Hierarchy* (*IH*)) als eine Hierarchie von *internen Elementen* (engl. *Internal Element* (*IE*)) instanziiert. Die spezifischen Eigenschaften der Internen Elemente werden über Attribute abgebildet. Zudem können die Internen Elemente Verknüpfungen untereinander oder mit extern modellierten Informationen enthalten. So unterstützt AutomationML Verknüpfungen mit *COLLADA*[2] und *PLCopen XML*[3] und ist so ausgelegt, dass weitere Formate hinzukommen können.

[1] https://www.w3.org/XML, zuletzt zugegriffen: 14.05.2021.
[2] Offenes Austauschformat für 3D-Informationen, https://www.khronos.org/collada/, zuletzt zugegriffen: 02.11.2020.
[3] Offenes XML-basiertes Austauschformat für Logik von Steuerungskomponenten, https://plcopen.org/technical-activities/xml-exchange, zuletzt zugegriffen: 02.11.2020

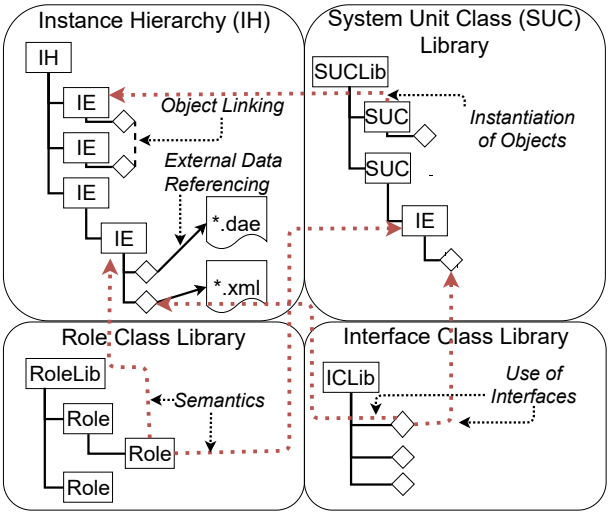


Abbildung 2.1: Überblick über die AutomationML-Struktur.

2.3 Open Platform Communications Unified Architecture (OPC UA)

OPC UA wurde 2008 vorgestellt und ist eine plattformunabhängige, serviceorientierte Architektur, die als mehrschichtiges Datenaustauschprotokoll mit eigenem Informationsmodell zusammengefasst werden kann. Wie in Abbildung 2.2 zu sehen, besteht die Informationsrepräsentation aus verschiedenen Modellen, einem Kernmodell, welches für OPC UA in der IEC-62541-Standardreihe standardisiert wird, spezifizierte Informationsmodelle, die in den sogenannten *Companion Specs*[1] definiert werden und den vom Hersteller oder Anwender eigens definierbaren Hersteller-Erweiterungen.

[1] Vgl. https://opcfoundation.org/developer-tools/specifications-opc-ua-information-models, zuletzt zugegriffen: 02.11.2020.

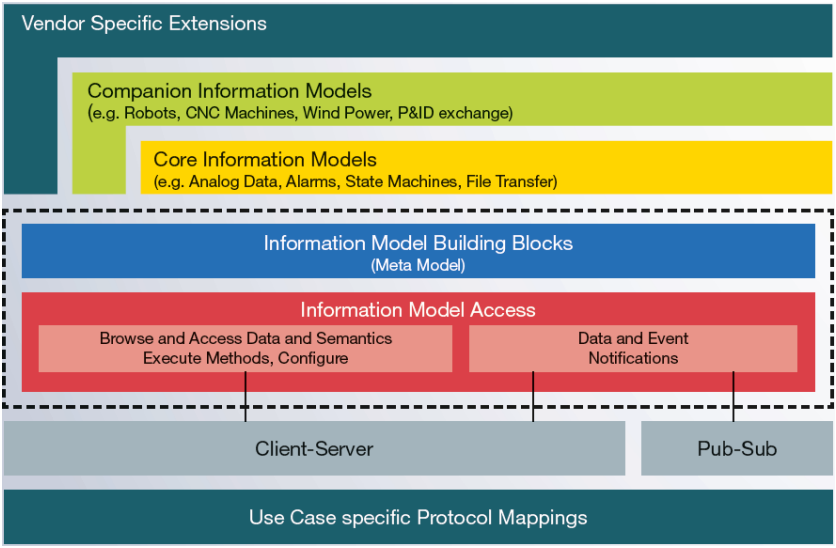


Abbildung 2.2: Überblick über die OPC-UA-Architektur mit Informationsfokus. *Quelle:* OPC Foundation, https://opcfoundation.org/about/opc-technologies/opc-ua/, zuletzt zugegriffen: 02.11.2020.

Die Informationsmodelle werden im sogenannten Adressraum als Knotenhierarchien instanziiert. Für mehr Informationen hierzu werden interessierte Lesende auf IEC 62541 Teil 3 [OPC17a] und Teil 5 [OPC17b] verwiesen. OPC UA kann zudem die authentifizierte, integre und vertrauliche Kommunikation zwischen Clients und Servern sicherstellen. Eine Untersuchung der Sicherheitseigenschaften von OPC UA schlussfolgerte, dass das Protokoll ein hohes Maß an Sicherheit bietet, sofern entsprechende Sicherheitsprofile (`Sign` oder `SignAndEncrypt`) genutzt werden [Asc16]. Standardmäßig werden Endgeräte jedoch oft mit aktiviertem Sicherheitsprofil NONE ausgeliefert, welches keine Sicherheitsmechanismen einsetzt.

2.4 Sicherheit für industrielle Systeme

IT-Systeme steuern Daten, wohingegen industrielle Systeme physische Abläufe steuern und damit andere Anforderungen als IT-Systeme haben. Dies wirkt sich auch auf die Sicherheitsanforderungen industrieller Systeme aus. Verfügbarkeit hat in industriellen Systemen eine besonders hohe Priorität, während die, in der in der Sicherheitsgemeinschaft viel behandelten Büro-IT so wichtige, Vertraulichkeit von Informationen oft nur sehr niedrig priorisiert wird [Com09].
Zudem führen Ressourcenbeschränktheit, Echtzeit- und Betriebssicherheitsanforderungen dazu, dass Sicherheitskonzepte aus der Büro-IT nicht ohne weiteres in industriellen Systemen einsetzbar sind. Ein Beispiel hierfür sind asymmetrische Verfahren, die durch ihren hohen Ressourcenverbrauch oft nicht in industriellen Komponenten eingesetzt werden können.
Des Weiteren sollten Scans, u.a. wegen der Gefährdung der Dienstverfügbarkeit und Echtzeitgarantien, nicht in industriellen Netzwerken eingesetzt werden.
Auch die angesprochenen Schwierigkeiten der Aktualisierung und Modernisierung industrieller Systeme (vgl. Abschnitt 2.1) hat besondere Auswirkungen auf Sicherheitsbetrachtungen. So können bekannt werdende Schwachstellen oft nicht sofort behoben werden. Daher sind vorgelagerte Sicherheitsmechanismen zu implementieren, die den verwundbaren Produktivsystemen zusätzlichen Schutz bieten. Ein Beispiel hierfür ist das Einschränken bestimmter Kommunikation über Firewalls, um das Ausnutzen einer bekannt gewordenen Schwachstelle explizit zu erschweren, ohne einen Stillstand des Systems zu benötigen. Dies macht auch noch einmal deutlich, wie relevant der sogenannte *Defense-in-Depth*-Ansatz in industriellen Umgebungen ist, der den vielschichtigen Einsatz von Gegenmaßnahmen fordert.

Generell unterscheiden sich bekannte Sicherheit-Best-Practices, Bedrohungsklassifizierungen und Sicherheitsmanagementprozesse zwischen Büro-IT und industriellen Systemen. Dabei sind in modernen industriellen Systemen sowohl die IT- als auch die OT-Sicherheit zu berücksichtigen, denn die fortschreitende Verbreitung kostengünstiger Ethernet- und IP-Geräte führt vermehrt

zur Ablöse proprietärer Technologien mit singulären Verwendungszwecken, was die Möglichkeit von Schwachstellen und Vorfällen in der Cybersicherheit erhöht [Nat11, Pae20]. Diese Dissertation berücksichtigt daher ebenfalls beide Sicherheitsdomänen (bspw. bei Netzwerksegmentierung/-zugriffskontrolle), fokussiert jedoch aufgrund der beschriebenen Forschungsfragen und Hypothesen industrielle Systeme.

2.5 Sicherheitsstandards und -Best-Practices

Um Sicherheit für Informationen oder Prozesse in einem Unternehmen erarbeiten zu können, bedarf es der Verankerung dieses Ziels in der Agenda der Unternehmensführung, sowie einer entsprechenden Rollen-, Zuständigkeits- und Aufgabenzuweisung. Dies sind Aspekte des Sicherheitsmanagements, die als *Sicherheitsmanagementsystem* (*SMS*) in einem Unternehmen umgesetzt werden sollten. Anforderungen für dessen Umsetzung, sowie Konzepte und Methoden, die als eine Art Vorlage dienen, werden in Standards wie ISO 27001 [Sta13] für Informationssysteme oder IEC 62443 [Com09] für Steuer- und Automatisierungssysteme spezifiziert. Die im industriellen Umfeld relevantere Standardreihe IEC 62443 beschreibt auch abstrakte Anforderungen an die Sicherheit von Systemen und Komponenten. Wie diese jedoch konkret umgesetzt werden, wird aufgrund des angestrebten weitläufigen Gültigkeitsbereichs des Standards offen gelassen.
Solche Umsetzungen werden durch Leitfäden konkretisiert, die spezifischere Best-Practices vorstellen. Hierzu gehören NIST's *„Guide to Industrial Control Systems (ICS) Security"* [Nat11] (von hieran abgekürzt als *NIST 800-82*), ENISA's *„Good Practices for Security of Internet of Things in the context of Smart Manufacturing"* [Eur18] (von hieran abgekürzt als *ENISA-GP*) und die Empfehlungen für industrielle Steuerungs- und Automatisierungssysteme des *Bundesamts für Sicherheit in der Informationstechnik* (*BSI*), zu denen das in Deutschland weitläufig bekannte *„ICS-Security-Kompendium"* [Bun13] gehört.
Mithilfe der Leitfäden und weiterer Quellen für Best-Practices werden im Rahmen eines SMS organisatorische und technische Richtlinien verfasst, die im Unternehmen umzusetzen sind. In dieser Dissertation werden nur die durch

die im folgenden Abschnitt beschriebenen Sicherheitsanalysen prüfbaren, technischen Richtlinien betrachtet. Diese befassen sich beispielsweise mit Passwortrichtlinien, Härtungsmaßnahmen oder Netzwerksegmentierungsstrategien.

2.6 Sicherheitsanalyse

Risikoanalysen sind die wohl am häufigsten von modellbasierten Sicherheitsanalysen adressierten Analysearten. Nichtsdestotrotz liegen diese nicht im Fokus der Arbeit, sondern sind ein möglicher Anwendungsfall der hier betrachteten Analysearten. Auch Teil 3-2 [Com20] des bereits erwähnten Sicherheitsstandards IEC 62443 befasst sich mit Risikoanalysen. Ein vereinfachtes, vom Standard abgeleitetes Ablaufdiagramm für eine Risikoanalyse ist beispielhaft in Abbildung 2.3 dargestellt. Dem Ablauf folgend sollen im ersten Schritt Bedrohungen identifiziert und im zweiten Schritt Schwachstellen ermittelt werden. Solche Bedrohungs- und Schwachstellenanalysen (vgl. Abschnitte 2.6.1 und 2.6.2) werden in dieser Arbeit adressiert.
Im Rahmen einer Risikoanalyse nach IEC 62443 werden zudem vorhandene Gegenmaßnahmen identifiziert und evaluiert, sowie fehlende Gegenmaßnahmen abgeleitet (vgl. Abbildung 2.3). Das Sicherstellen der korrekten Konfiguration dieser Gegenmaßnahmen ist für deren Effektivität unerlässlich. Entsprechende Analysen zum Aufdecken von Fehlkonfigurationen werden in dieser Arbeit unter dem Begriff Konfigurationsanalyse (vgl. Abschnitt 2.6.3) zusammengefasst, werden in der Literatur häufig jedoch auch unter *Security Configuration Management* (*SCM*) geführt.
Nach dem Durchführen der Risikoanalyse werden unter anderem technische und organisatorische Richtlinien im Rahmen eines Sicherheitsmanagementsystems (vgl. Abschnitt 2.5) festgelegt. Die technischen Richtlinien enthalten entsprechend Anforderungen technischer Natur, wie spezifische Anforderungen an die Netzwerksegmentierung. Ob diese Richtlinien im betrachteten System erfüllt werden und dies nach bewährten Praktiken erfolgt, wird mithilfe sogenannter Konformitätsanalysen (vgl. Abschnitt 2.6.4) geprüft.

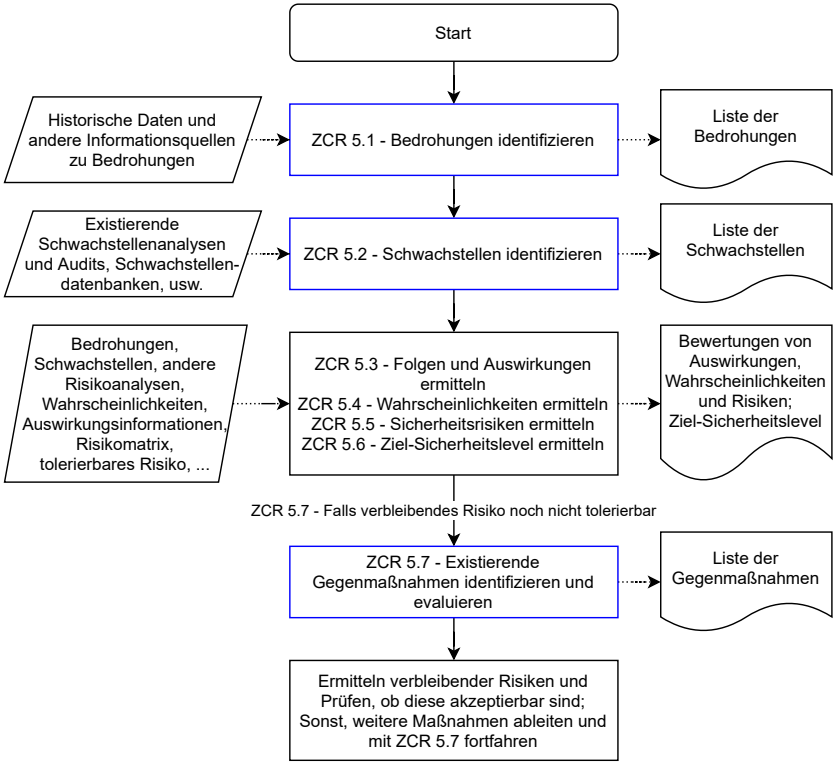


Abbildung 2.3: Vereinfachtes und übersetztes Ablaufdiagramm zur Risikoanalyse aus IEC 62443 Teil 3-2 [Com20]. Die hier referenzierten Aufgaben sind dabei gestrichelt eingerahmt.

Im laufenden Betrieb muss ein industrielles System kontinuierlich überwacht werden, um Angriffe frühzeitig zu erkennen. Hierbei ist ein bekanntes Problem der dafür zuständigen Überwachungs- und Detektionssysteme, dass auftretende Effekte zwar erkannt, aber oft nicht, oder erst sehr spät, zu Angriffen korreliert werden (vgl. Abschnitt 3.3.7). Angriffsmuster können verwendet werden, um solche Zusammenhänge zu erkennen. Solche Ansätze zur Angriffserkennung und -korrelation (vgl. Abschnitt 2.6.5) werden in dieser Arbeit

ebenfalls betrachtet.
Die genannten Analysearten werden in den folgenden Abschnitten näher erläutert.

2.6.1 Bedrohungsanalyse

Sind die eigenen Assets bekannt, wird in der Regel versucht festzustellen, welchen Bedrohungen diese Assets ausgesetzt sein könnten. Hierzu wird auf bekannte Angriffe und Angriffsstrategien zurückgegriffen. Im industriellen Bereich bietet das 2020 erschienene *MITRE ATT&CK®Framework for ICS* (im Weiteren *ICS ATT&CK* genannt) eine Wissensbasis für die Verknüpfung von Asset-Typen, allgemeinen Bedrohungen (bzw. Taktik, engl. *Tactic*), spezifischeren Bedrohungen (bzw. Technik, engl. *Technique*) und Gegenmaßnahmen (im engl. *Mitigations*) [Ale20]. In dieser Arbeit wird die automatisierte Bedrohungsanalyse mithilfe von ICS ATT&CK®betrachtet.

2.6.2 Schwachstellenanalyse

Schwachstellenanalyse im Kontext dieser Dissertation befasst sich mit dem automatisierten Auffinden von bekannten Schwachstellen von Hard- und Software. Dafür können öffentlich zugängige Datenbanken genutzt werden, welche Beschreibungen dieser Schwachstellen in standardisierten Formaten, wie *Common Vulnerabilities and Exposures* (*CVE*[1]), zusammen mit deren Bewertung zur Verfügung stellen. Die Schwachstellenmeldungen hierfür kommen aus verschiedenen Quellen, oft von *Computer Emergency Response Teams* (*CERTs*) oder Sicherheitsforschern. Das Ermitteln bekannter Schwachstellen im eigenen System dient, wie bereits erwähnt, der Risikoabschätzung und wird meist im Rahmen einer Bedrohungsanalyse durchgeführt, um den relevanten, bekannten Schwachstellen entsprechende Bedrohungen zuzuordnen und so eine bessere Auswahl der Gegenmaßnahmen vornehmen zu können (vgl. Abschnitt 3.3.4). Schwachstellenanalysen sind aber insbesondere auch zur Identifikation von

[1] https://cve.mitre.org/, zuletzt zugegriffen: 31.10.2020.

Handlungsbedarfen, wie Patchen oder Vorschalten von Zugriffskontrollen, einzusetzen und somit, besonders durch die fortlaufende Entdeckung neuer Schwachstellen, nicht nur für die initiale Risikoanalyse relevant, sondern als kontinuierliche Maßnahme zu etablieren.

2.6.3 Konfigurationsanalyse / Security Configuration Management

Der Terminus Security Configuration Management (SCM) wird für verschiedene Bereiche der Verwaltung von sicherheitsrelevanten Konfigurationen verwendet. In dieser Arbeit wird die automatisierte Suche nach Fehlkonfigurationen fokussiert, welche hier als Konfigurationsanalyse bezeichnet wird. Dafür werden die folgenden Klassen von Fehlkonfigurationseffekten definiert:

- *Intra-Konfiguration*: Hierzu zählen Effekte durch Konfigurationsfehler wie Redundanz, Widerspruch, Generalisierung usw., die im Rahmen einer zu konfigurierenden Entität (z.B. innerhalb einer Firewall oder eines DNS-Servers) auftreten können.
- *Inter-Konfiguration*: Diese Effektklasse deckt Effekte von Fehlern ab, welche Widersprüche, Verdeckungen und weitere Probleme zwischen mehreren zu konfigurierenden Entitäten beliebigen Typs hervorrufen.
- *Intra-Konfiguration-Konformität*: Hiermit werden Widersprüche klassifiziert, welche zwischen Konfigurationen einer Entität bestehen.
- *Inter-Konfiguration-Konformität*: Hiermit werden Widersprüche klassifiziert, welche zwischen der Konfigurationen von mindestens zwei Entitäten beliebigen Typs bestehen.

Aufgrund der unter Umständen kritischen Auswirkung von Fehlkonfigurationen, sollten Konfigurationsanalysen nach jeder Änderung einer Konfiguration eingesetzt werden. Auch wenn dies wünschenswert ist, wird dies aufgrund der fehlenden Zugänglichkeit von maschineninterpretierbaren Konfigurationen noch kaum eingesetzt.

2.6.4 Konformitätsanalyse

Eine Konformitätsanalyse (oder auch *Compliance-Analyse*) kann auf verschiedenen Abstraktionsebenen agieren und wird nicht verwendet, um Widersprüche zwischen Konfigurationen zu identifizieren, sondern um die Konformität von Konfigurationen gegenüber Vorgaben (vgl. Abschnitt 2.5) zu validieren. Der Abstraktionsgrad einer solchen Vorgabe kann auch als semantischer Abstand zwischen der Vorgabe und ihrer technischen Umsetzung gesehen werden. Auch Sicilia et al. schlussfolgerten in ihrer Veröffentlichung zu einer Untersuchung von Sicherheitsontologien [Sic15], dass der Begriff „*Policy*“ (also Vorgabe oder Richtlinie) auf verschiedenen Ebenen genutzt wird. Dabei muss bedacht werden, dass sich mit dem Abstraktionsgrad einer Vorgabe auch ihr Interpretationsspielraum vergrößert. Die im Teil 3-2 des IEC 62443 definierte Anforderung ZCR-3.2 – *„Separate business and control system assets“*, welche die Trennung von Büro- und Steuersystem-Assets verlangt – besitzt einen verhältnismäßig hohen Abstraktionsgrad und kann somit technologisch auf verschiedenstem Wege umgesetzt werden. Dem gegenüber steht die Vorgabe *„All outbound traffic from the control network to the corporate network should be source and destination-restricted by service and port.“* aus NIST 800-82, welche eine Dienst- und Port-basierte Beschränkung von Netzwerkverkehr zwischen Büro- und Steuersystem-Netzwerk verlangt und somit technisch weit weniger Interpretationsspielraum enthält. Auch zu erkennen ist, dass eine Umsetzung der NIST-Vorgabe konform zur IEC-62443-Vorgabe ist und als eine der Interpretationen dieser Vorgabe dienen kann. Diese Definitionen auf verschiedenen Ebenen sorgen dafür, dass eine Konformitätsanalyse eine gewisse Ungenauigkeit enthält, die als proportional zum Abstraktionsgrad von der technischen Umsetzung betrachtet werden kann. In dieser Dissertation werden mit dem Wort Richtlinie all diese Definitionsebenen adressiert. Allerdings wird keine Quantifizierung der Konformität bzw. ihrer Ungenauigkeit vorgenommen, sondern durch entsprechende Konzepte eine Festlegung der eigenen, unternehmensinternen Interpretationen unterstützt.

2.6.5 Angriffserkennung und -korrelation

Die Erläuterung des Verständnisses von Angriffserkennung und -korrelation in dieser Dissertation wird durch Abbildung 2.4 grafisch unterstützt. Der Erkennung von Angriffen liegt i.d.R. eine Menge von Vorfällen zugrunde. Monitoring-Anwendungen bzw. *Intrusion Detection Systemen*, also Anwendungen, die mögliche Vorfälle automatisiert detektieren, erzeugen bei Entdeckung solcher Vorfälle Meldungen. Diese Vorfallmeldungen werden manuell oder automatisiert korreliert, um Rückschlüsse auf einen Angriff zu ziehen, der diese ausgelöst hat. Diese Dissertation adressiert eine solche automatisierte Erkennung von Angriffen.
Jedoch besteht eine Bedrohung meist aus einer Folge von Angriffen. Bei der Ableitung dieser Zusammenhänge, die wiederum zur Identifikation des Gesamtangriffes (bzw. des Angriffsvektors) und damit der Bedrohung führt, wird in dieser Dissertation von einer Angriffskorrelation gesprochen. Diese Erkennung der Angriffe und Angriffsvektoren wird in der Literatur meist ebenfalls als Angriffserkennung bezeichnet.

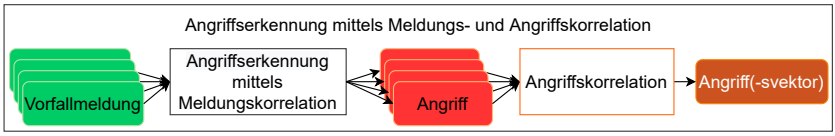


Abbildung 2.4: Zusammenhang zwischen Angriffserkennung und -korrelation.

2.7 Ontologien und Semantic Web

Für den aus der Philosophie stammenden Begriff *Ontologie* gibt es verschiedene Definitionen. Die wohl bekannteste im Bereich des Wissensaustauschs ist die Definition von Gruber, wonach eine Ontologie eine *explizite Spezifikation einer Konzeptualisierung* ist [Gru93]. Dies ist im Kontext dieser Arbeit insoweit zu präzisieren, dass eine Konzeptualisierung die explizite Formulierung der

semantischen Struktur und Restriktionen einer Domäne der Realität ist. Ontologien wurden für das Teilen von Wissen und die Wiederverwendung zwischen Entitäten innerhalb einer Domäne entworfen. Die semantische Struktur und eingebettete semantische Regeln unterscheiden eine Ontologie eindeutig von einer *Taxonomie*, welche lediglich als ein Klassifikationssystem gesehen werden kann [Und03]. Auch von *Thesauri* ist eine Ontologie abzugrenzen. Taxonomien und Thesauri werden zur Wissensorganisation (im Rahmen eines *Knowledge Oganization Systems* (*KOS*)) verwendet, wohingegen Ontologien für die Wissensrepräsentation eingesetzt werden [Ara05]. Wissensorganisation zeichnet sich durch geringere Komplexität als Wissensrepräsentation aus, besitzt aber dadurch auch eine geringere semantische Tiefe und weit weniger Möglichkeiten zur Definition von Restriktionen. Außerdem kann die Wissensorganisation, im Gegensatz zu auf formalen Sprachen basierenden Ontologien, nicht zur automatisierten Ableitung neuen Wissens, dem sogenannten Reasoning, eingesetzt werden. Im Gegenzug ist die Suche auf Taxonomien und Thesauri in der Regel robuster, da auch Verwandtschaft von Begriffen berücksichtigt werden kann. Es gibt zudem das standardisierte *Simple Knowledge Organization System* (*SKOS*)[1], welches eine Ontologie definiert, mit deren Hilfe KOS als Ontologien dargestellt werden können. Dadurch werden auch Verwandtschaften von Begriffen abgebildet und somit beide Vorteile kombiniert.

Anwendung finden Ontologien heutzutage vor allem in der Formalisierung von Wissen einer Domäne, für dessen Wiederverwendbarkeit in verschiedenen Programmen und der Ableitung neuen Wissens durch Reasoner. Prägend hierfür war die Erfindung des *Semantic Web*, dessen Ziele Hitzler et al. wie folgt zusammenfassen [Hit08]:

> *„Finde Wege und Methoden, Informationen so zu repräsentieren, dass Maschinen damit in einer Art und Weise umgehen können, die aus menschlicher Sicht nützlich und sinnvoll erscheint.“, [Hit08]*

Der Ansatz von Semantic Web basiert dabei auf der Entwicklung offener, erweiterbarer Standards zur Beschreibung von Informationen und dem Einsatz

[1] https://www.w3.org/TR/skos-primer, zuletzt zugegriffen: 10.04.2021.

von Methoden zur Herleitung neuer Informationen [Hit08]. Hierfür wurden unter anderem die Ontologiesprache *Web Ontology Language* (*OWL*), die Regelsprachen *Semantic Web Rule Language* (*SWRL*) [W3C04] und *Jena Rule Language*[1], sowie diverse Reasoner, wie *Pellet*[2] oder *Hermit*[3] entwickelt.
Diese Technologien werden von den meisten ontologiebasierten Sicherheitsanalyseansätzen verwendet. Da die Wiederverwendbarkeit, Austauschbarkeit und Automatisierbarkeit von Analysen und dem damit verbundenen Wissen (inkl. dessen Herleitung) im Fokus dieser Dissertation stehen, werden die genannten Semantic Web Technologien auch in den hier vorgestellten Konzepten und Methoden eingesetzt. Daher werden sie in den folgenden Unterabschnitten genauer beschrieben.

2.7.1 OWL 2 DL

OWL 2 [W3C12b] steht für die Web Ontology Language in Version 2. OWL 2 ist eine rückwärtskompatible Erweiterung von OWL. Da diese Erweiterungen in dieser Dissertation keine Rolle spielen, gelten die in den folgenden Paragraphen vorgestellten, und im restlichen Dokument verwendeten Konzepte sowohl für OWL 1 als auch für OWL 2. Daher wird die Versionsnummer im Folgenden, wo immer diese irrelevant ist, weggelassen. Für OWL 2 wurde die entscheidbare Untermenge *OWL 2 DL* [W3C12a] definiert, welche als eine *Beschreibungslogik* (engl. *Description Logic* (*DL*)) verwendet werden kann.
In diesem Abschnitt werden nur die für das Verständnis dieser Arbeit benötigten Grundlagen eingeführt. Für einen detaillierteren Einblick in OWL und OWL DL werden interessierte Lesende auf [Hit08] verwiesen.
Da die Lesbarkeit der OWL-Formeln im Vordergrund stehen soll, wird für OWL-Definitionen in dieser Dissertation die Syntax für Beschreibungslogiken verwendet.

OWL besteht aus den Grundbausteinen *Individuen*, *Klassen* und *Rollen*. Ein Individuum kann als Instanz einer Klasse betrachtet werden. Bei Rollen wird

[1] https://jena.apache.org/documentation/inference, zuletzt zugegriffen: 20.02.2021.
[2] https://www.w3.org/2001/sw/wiki/Pellet, zuletzt zugegriffen: 07.05.2021.
[3] http://www.hermit-reasoner.com, zuletzt zugegriffen: 07.05.2021.

in OWL zwischen abstrakten und konkreten Rollen unterschieden. Eine abstrakte Rolle drückt Beziehungen zwischen Individuen aus, während eine konkrete Rolle Beziehungen zwischen Individuen und Datenwerten ausdrückt. Diese Definitionen sollen anhand der Gleichung 2.1 erläutert werden. Dabei formalisiert Zeile 2.1a, dass `sps1` (Repräsentant einer bestimmten SPS) ein Individuum der Klasse Plc ist und Zeile 2.1b, dass das Individuum `int_192_168_10_11` (Repräsentant einer IPv4-Schnittstelle) zur Klasse IpV4Interface gehört. Zeile 2.1c zeigt, wie die abstrakte Rolle *interface* verwendet wird, um zu definieren, dass die IPv4-Schnittstelle `int_192_168_10_11` eine Schnittstelle von `sps1` ist. Zuletzt ist aus Zeile 2.1d zu entnehmen, dass der Schnittstelle `int_192_168_10_11`, mithilfe der konkreten Rolle *ip*, der Wert „192.168.10.11“ als Repräsentant der IP-Adresse zugewiesen wird.

$$\text{Plc}(\texttt{sps1}) \tag{2.1a}$$

$$\text{IpV4Interface}(\texttt{int\_192\_168\_10\_11}) \tag{2.1b}$$

$$\texttt{sps1}.\mathit{interface}(\texttt{int\_192\_168\_10\_11}) \tag{2.1c}$$

$$\texttt{int\_192\_168\_10\_11}.\mathit{ip}(\text{“}\mathit{192.168.10.11}\text{”}\hat{}\hat{}\texttt{xsd:string}) \tag{2.1d}$$

Wichtig ist an dieser Stelle auch der Datentyp der IP-Adresse. Erlaubt sind u.a. die XML-Datentypen, zu denen beispielsweise xsd:string und xsd:integer gehören, wobei `xsd:` ein Präfix ist, der den Namensraum der Datentypen substituiert. Der Namensraum und der entsprechende Suffix bilden zusammen einen *Internationalized Resource Identifier (IRI)*[1]. Die Gleichung definiert die IP-Adresse also als `xsd:string`. Wenn `xsd` das Präfix für `<http://www.w3.org/2001/XMLSchema#>` ist, bildet dies die IRI `http://www.w3.org/2001/XMLSchema#string` und identifiziert so eindeutig die Definition des Datentyps.
Genauer hat jede Klasse, Rolle und Ontologie, sowie jeder Datenwert, Datentyp und jedes Individuum in OWL einen Namensraum. Zur besseren Lesbarkeit wird in den Beispielen dieser Dissertation jedoch der Namensraum der Ontologie, die im jeweiligen Kontext gerade definiert wird, weggelassen. Dies ist auch generell üblich, da es sich dabei um den eindeutigen Basisnamensraum

[1] https://www.ietf.org/rfc/rfc3987.txt, zuletzt zugegriffen: 12.11.2020.

handelt, der für alle Aussagen und Definitionen im Rahmen der Ontologie einheitlich ist.

Um weitere Definitionen und ihre Auswirkungen aufzuzeigen, sei zudem noch angenommen, dass `bk1` ein Individuum der Klasse Buscoppler ist und sowohl Plc als auch Buscoppler Unterklassen von Asset sind:

$$\begin{aligned} &\text{Buscoppler}(\texttt{bk1}) \\ &\text{Plc} \sqsubseteq \text{Asset} \\ &\text{Buscoppler} \sqsubseteq \text{Asset} \end{aligned} \tag{2.2}$$

Somit würde ein Reasoner schlussfolgern, dass Folgendes gilt:

$$\begin{aligned} &\text{Asset}(\texttt{sps1}) \\ &\text{Asset}(\texttt{bk1}) \end{aligned} \tag{2.3}$$

Zudem könnte man definieren wollen, dass die Klassen Host und NetworkHost äquivalent sind:

$$\text{Host} \equiv \text{NetworkHost} \tag{2.4}$$

Somit ist jedes Individuum von Host auch ein Individuum der Klasse NetworkHost.

Auch Konjunktion (Zeile 2.5a), Disjunktion (Zeile 2.5b) und Negation (Zeile 2.5c) sind in OWL ausdrückbar:

$$\text{NetworkFirewall} \sqsubseteq \text{NetworkComponent} \sqcap \text{Firewall} \tag{2.5a}$$

$$\text{NetworkAsset} \sqsubseteq \text{NetworkComponent} \sqcup \text{Host} \tag{2.5b}$$

$$\text{NetworkComponent} \sqsubseteq \neg\text{Host} \tag{2.5c}$$

In dem Beispiel ist zu sehen, dass jedes Individuum von NetworkFirewall sowohl ein Individuum von NetworkComponent als auch von Firewall ist. Auch wird definiert, dass jedes Indiviuum von NetworkAsset auch entweder zur Klasse NetworkComponent oder zur Klasse Firewall gehört. Zudem wird definiert, dass Individuen der Klasse NetworkComponent nicht zur Klasse

Host gehören können.
Außerdem können komplexe Klassen über Rollen spezifiziert werden, sodass Folgendes gilt:

$$\text{Firewall} \sqsubseteq \forall \textit{config}.\text{FirewallConfig} \tag{2.6a}$$

$$\text{Firewall} \sqsubseteq \exists \textit{config}.\text{FirewallConfig} \tag{2.6b}$$

Zeile 2.6a impliziert, dass wenn ein Individuum der Klasse Firewall eine Beziehung zu einem weiteren Individuum über die Rolle *config* besitzt, dieses weitere Individuum von der Klasse FirewallConfig sein muss. Außerdem definiert Zeile 2.6b, dass ein Individuum der Klasse Firewall mindestens eine Beziehung zu einem zweiten Individuum der Klasse FirewallConfig über die Rolle *config* besitzen muss. Solche Einschränkungen werden im Rahmen dieser Arbeit als *Restriktionen* bezeichnet.
An dem Beispiel für die Restriktionen lässt sich jetzt eine Besonderheit von OWL erörtern, die *Offene-Welt-Annahme* (engl. *Open World Assumption* (*OWA*)). Nach der OWA wird davon ausgegangen, dass eine Wissensbasis, wie sie auch hier im Laufe der Beispiele erstellt wurde, immer unvollständig ist.
Für die bisher definierte Wissensbasis hat dies beispielsweise zur Folge, dass die Wissensbasis auch dann gültig ist, wenn ein Individuum der Klasse Firewall existiert, das keine Beziehung zu einem zweiten Individuum der Klasse FirewallConfig über die Rolle *config* besitzt. Denn die OWA besagt, dass eine solche Beziehung existieren könnte, nur bisher nicht explizit definiert wurde. Auch geht ein Reasoner nicht davon aus, dass zwei Individuen unterschiedlich sind. So könnte es sich bei `sps1` und `bk1` um dasselbe Individuum handeln. Erst, wenn man definieren würde, dass Folgendes gilt, würde ein Reasoner die Individuen als unterschiedlich betrachten:

$$\begin{aligned}&\text{Host}(\texttt{sps1})\\&\text{NetworkComponent}(\texttt{bk1})\end{aligned} \tag{2.7}$$

Der Grund hierfür ist die vorherige Definition, dass Host und NetworkComponent komplementär sind. Dass es sich bei den Individuen um unterschiedliche

Individuen handelt lässt sich auch direkt definieren:

$$\neg \texttt{sps1} = \texttt{bk1} \tag{2.8}$$

Generell kann man Beschreibungslogiken in eine *TBox* und eine *ABox* einteilen (also auch OWL DL). Dabei enthält die TBox terminologisches Schemawissen. Dazu gehören die Informationen, welche Klassen von Objekten es in der entsprechenden Domäne gibt und welche Eigenschaften diese haben. Die ABox hingegen, enthält das assertionale Instanzwissen, also Aussagen über Instanzen dieser Klassen. Gleichung 2.1 enthält ausschließlich solche ABox-Aussagen, wohingegen Gleichung 2.2 aus einer ABox-Aussage Buskoppler(bk1) und zwei TBox-Aussagen besteht. Letztere beziehen sich nur auf das terminologische Wissen der Domäne, indem sie definieren, dass sowohl Plc als auch Buscoppler Subklassen von Asset sind.
Wie auch in der Literatur üblich, ist in dieser Dissertation mit dem Terminus „Ontologie“ zunächst einmal nur die TBox gemeint, also nur die semantischen Konzepte einer Ontologie. Eine „instanziierte Ontologie“, oder „*Instanzontologie*“, beschreibt hingegen die Kombination aus TBox und ABox.

Ein besonderer Aspekt der Reduktion von OWL auf eine Beschreibungslogik (OWL DL) ist deren Entscheidbarkeit, d.h. es gibt einen Ableitungsalgorithmus, der zu jedem Ableitungsproblem (Inferenzproblem) einer in OWL DL definierten Wissensbasis immer terminiert. Eine *Wissensbasis* kann in diesem Dokument zudem immer auch als *Modell* gesehen werden. Die Umkehrrichtung gilt hierbei nur im Kontext von ontologiebasierten Modellen.

2.7.2 Regelsprachen für OWL 2 DL

Über Restriktionen und Definitionen komplexer Klassen lassen sich viele wünschenswerte Schlussfolgerungen nicht ableiten. Hierfür stehen Regelsprachen zur Verfügung, die auf Instanzontologien angewendet werden können. Diese sind wie Ontologien selbst wiederverwendbar. Im Folgenden wird auf die in dieser Arbeit meistverwendete Regelsprache Semantic Web Rule Language (SWRL) eingegangen. Eine ebenfalls häufig genutzte Alternative ist die Jena

Rule Language, welche hier jedoch nicht explizit eingeführt werden muss, da sie in diesem Dokument lediglich für ein Beispiele eingesetzt wird und dieses Beispiel an entsprechender Stelle erörtert wird.

2.7.2.1 Semantic Web Rule Language

Semantic Web Rule Language (SWRL) ist eine Sprache zur Definition von Hornklausel-artigen Regeln, welche wie OWL auf der OWA basiert und, ohne Erweiterungen, monoton ist. Der generelle Aufbau basiert auf einer Bedingung und einer Schlussfolgerung, die gefolgert wird, falls die Bedingung wahr ist. Monoton bedeutet hierbei, dass eine Bedingung, die als wahr ausgewertet wurde, nach der Schlussfolgerung immer noch wahr ist.
In dieser Dissertation wird die menschenlesbare Syntax von SWRL verwendet, die ebenfalls in der Spezifikation definiert ist. In dieser Syntax hat eine SWRL-Regel die folgende Form:

$$\textit{Bedingung} \rightarrow \textit{Konsequenz}$$

Dabei sind sowohl Bedingung als auch Konsequenz Konjunktionen der Form $a_1 \wedge ... \wedge a_n$. Variablen werden mit einem Fragezeichen beginnend geschrieben (also z.B. $?x$). Als Beispiel sei die folgende Regel gegeben:

$$\begin{aligned} &\text{Interface}(?i1) \wedge \textit{supportsProtocol}(?i1, \texttt{Opcua}) \wedge \text{Interface}(?i2) \\ &\wedge \textit{supportsProtocol}(?i2, \texttt{Opcua}) \rightarrow \textit{compatible}(?i1, ?i2) \end{aligned} \tag{2.9}$$

Diese Regel besagt, dass zwei Schnittstellen, welche jeweils OPC UA unterstützen, kompatibel zueinander sind, was mit einer Definition in OWL nicht möglich gewesen wäre. Variablen werden bei Auswertung der Regel je an ein Individuum oder einen Wert gebunden (je nachdem was die TBox definiert). Als Beispiel könnte das Individuum Opcua auch durch eine weitere Variable $?protocol$ ersetzt werden, falls die Aussage verallgemeinert werden sollte und somit jedes, von beiden Schnittstellen unterstützte Protokoll die Bedingung erfüllen soll. Die Variable $?protocol$ würde dann entweder einen Wert binden,

falls es sich bei *compatible* um eine konkrete Rolle handelt, oder ein Individuum, falls es sich um eine abstrakte Rolle handelt. In SWRL werden auch Erweiterungen definiert, deren Verwendung jedoch gegebenenfalls nicht mehr die Monotonie einer Regel sicherstellt. Ein Beispiel wäre die Regel:

$$\begin{aligned} &\mathit{hasAge}(\texttt{sps1}, ?\mathit{age}) \wedge \mathit{swrlb{:}add}(?\mathit{age}, ?\mathit{age}, 1) \\ &\wedge \mathit{swrlb{:}lessThan}(?\mathit{age}, 3) \rightarrow \mathit{hasAge}(\texttt{sps1}, ?\mathit{age})) \end{aligned} \tag{2.10}$$

Das Beispiel zeigt, dass die mit dem Präfix `swrlb:` versehenen Erweiterungen die Bedingung irgendwann ungültig werden lassen, was die Monotonie verletzt. Lässt man zudem noch *swrlb:lessThan*(?*age*, 3) weg, wird hier eine Endlosausführung kreiert.

In SWRL sind Ausdrücke, welche die *Closed World Assumption* (*CWA*), also das Gegenteil der OWA, annehmen, aus Gründen der Monotonie nicht erlaubt. Als Beispiel müsste die folgende **ungültige** Negation so interpretiert werden, dass falls für ?*a* keine Konfiguration bekannt ist, ?*a* als ConfigurationLessEntity deklariert wird:

$$\neg \mathit{config}(?a, ?c) \rightarrow \text{ConfigurationLessEntity}(?a) \tag{2.11}$$

Nun kann aber die Wissensbasis um eine Konfiguration für ?*a* erweitert werden, wodurch die Deklaration zurückgenommen werden müsste. Dies bricht die Monotonie. Einige Beschränkungen der Monotonie lassen sich durch Umgehungslösungen ausgleichen. So können beispielsweise nicht erlaubte Disjunktionen umgangen werden, indem die Regel in die disjunktive Normalform überführt und jede Konjunktion in eine eigene Regel ausgelagert wird. Beispielhaft wird so die **ungültige** Regel

$$\mathrm{A} \wedge (\mathrm{B} \vee \mathrm{C}) \rightarrow \mathrm{D} \tag{2.12}$$

über

$$(\mathrm{A} \wedge \mathrm{B}) \vee (\mathrm{A} \wedge \mathrm{C}) \rightarrow \mathrm{D} \tag{2.13}$$

zu

$$\begin{aligned} &A \wedge B \rightarrow D \\ &A \wedge C \rightarrow D \end{aligned} \tag{2.14}$$

wobei Gleichung 2.14 zwei gültige SWRL-Regeln sind.

2.7.3 Abfragesprachen für OWL 2 DL

Um Wissen aus der Wissensbasis abzufragen gibt es entsprechende Abfragesprachen. Darunter fällt auch eine Erweiterung von SWRL mit dem Namen *Semantic Query-enhanced Web Rule Language* (*SQWRL*) [OCo09], die es ermöglicht, SWRL für OWL-Abfragen einzusetzen. Als Beispiel wird die in dem vorherigen Abschnitt 2.7.2.1 definierte Regel in eine SQWRL Abfrage umgewandelt:

$$\begin{aligned} &\text{Interface}(?i1) \wedge \textit{supportsProtocol}(?i1, \texttt{Opcua}) \wedge \text{Interface}(?i2) \\ &\wedge \textit{supportsProtocol}(?i2, \texttt{Opcua}) \rightarrow \textit{sqwrl:select}(?i1, ?i2) \end{aligned} \tag{2.15}$$

Statt also die Kompatibilitätsaussage in die Wissensbasis einzufügen, wurden hier die kompatiblen Schnittstellen selektiert.

Alternativ gibt es die sehr mächtige, semantische Abfragesprache *SPARQL*[1], welche für die Modellierungssprache RDF entwickelt wurde, jedoch wegen der in OWL spezifizierten Abbildung zu RDF auch für OWL verwendet werden kann. SPARQL arbeitet mit der CWA und ermöglicht somit andere Abfragen als SQWRL. Für einen Großteil der Sicherheitsanalysen sind geschlossene Betrachtungen des Modells praktikabler, als die TBox so detailliert und restriktiv zu spezifizieren, dass für alle möglichen Negativ- und Existenzabfragen entsprechende Umgehungslösungen möglich sind (vgl. Abschnitt 2.7.2.1). Zum Beispiel, um herauszufinden, ob bestimmte Zustände erfüllt oder Konfigurationen und Assets vorhanden sind. Somit ist SPARQL zumindest für den letzten Auswertungs- und Abfrageschritt einer Analyse

[1] https://www.w3.org/TR/sparql11-query/, zuletzt zugegriffen: 04.11.2020.

sehr nützlich und wird auch in verwandten Arbeiten rege verwendet [Bou17, Moz18, Bou19, Eck20, Sar20]. In dieser Arbeit wird SPARQL in der Evaluation des Rahmenwerks zur Sicherheitsanalyse verwendet. Dafür wird hier eine kurze Einführung in die wesentliche Abfrage-Syntax von SPARQL gegeben. Ähnlich zu anderen Abfragesprachen können auch in SPARQL SELECT-Abfragen formuliert werden, welche bestimmtes Wissen aus dem Modell extrahieren.

```
1    SELECT ?notification WHERE {
2      ?notification a Notification.
3      FILTER EXISTS {
4        ?notification cef:hasNotificationData ?
             notification_type.
5        ?notification_type a :NotificationType.
6        ?notification_type :value "INCOMPLETE_TCP".
7      }
8    }
```

Listing 2.1: SPARQL-Beispielabfrage, welche die Meldungen zurückliefert, die den Meldungstyp INCOMPLETE_TCP besitzen.

Listing 2.1 zeigt beispielhaft eine Abfrage, welche alle Vorfallmeldungen zurückliefert, bei denen (ausgedrückt durch den Block nach dem Term WHERE) bestimmte Bedingungen erfüllt sind. Der Ausdruck `?notification` ist dabei eine Variable, was durch das führende Fragezeichen symbolisiert wird. Als Bedingung wurde in dem Beispiel angegeben, dass `?notification` semantisch zur Klasse `Notification` gehört (vgl. Zeile 2). Diese Zugehörigkeit wird mit dem Ausdruck `a` (als Kurzform von „is a“) angegeben. An Zeile 2 ist zu erkennen, dass SPARQL mit Tripeln aus Subjekt (z.B. `?notification`), Prätikat (z.B. `a`) und Objekt (z.B. `Notification`) arbeitet. Bedingungen, wie auch weitere Ausdrücke, werden mit Punkten voneinander getrennt. Listing 2.1 zeigt zudem, dass weiteres Filtern der Ergebnisse möglich ist. Konkret sorgt der „FILTER EXISTS“-Block (Zeilen 3-6) dafür, dass nur Meldungen des Typs

INCOMPLETE_TCP selektiert werden. Die Namen der Variablen können hierbei beliebig gewählt werden. Die SPARQL-Engine sucht bei Verarbeitung der Abfrage nach einer Lösung des Ausdrucks und bindet dafür verschiedene Individuen oder Werte an die Variablen. Wird eine Kombination von gebundenen Individuen und Werten gefunden, für die der Ausdruck wahr ist, werden die selektierten Variablen dieser Kombination zurückgegeben (im Beispiel jene für `?notification`). Wie in anderen Abfragesprachen ist es zudem möglich, Abfragen zu verschachteln und Ergebnisse über den Ausdruck GROUP BY zu gruppieren. SPARQL lässt noch eine Vielzahl weiterer Operationen und Ausdrücke zu. Für weitere Details werden interessierte Lesende auf [W3C13] verwiesen.

2.7.4 Ontologien erweitern und zusammenführen

Ontologien sind für die Wiederverwendung entworfen, die laut Pinto et al. [Pin99] in *Integration*, *Zusammenführung* (engl. *Merge*) und *Verwendung* unterteilt werden kann. Die Verwendung ist der direkte Einsatz einer Ontologie und bedarf daher keiner speziellen Erläuterung. Allerdings sind die Begriffe Integration und Zusammenführung in der Literatur mit verschiedener Bedeutung zu finden. Daher werden sie in den folgenden beiden Paragraphen, der Interpretation von [Pin99] folgend, erläutert. Für diesen Zweck sei eine Ontologie Θ als Menge von Konzepten $c \in \Theta$ definiert.

Integration

Bei der Integration der Ontologien $\Theta_1, \Theta_2, \Theta_3$ entsteht eine Ontologie Θ, für die gilt $\Theta_1 \subset \Theta, \Theta_2 \subset \Theta, \Theta_3 \subset \Theta$. Dabei ist die Ontologie Θ in der Regel die Ontologie, die nach der Integration weiterverwendet werden soll. Allerdings gilt ohne weitere Maßnahmen zunächst $\Theta = \Theta_1 \cup \Theta_2 \cup \Theta_3$. Die Domänen von $\Theta_1, \Theta_2, \Theta_3$ und Θ unterscheiden sich in der Regel alle untereinander, haben aber oft durchaus Überschneidungen. Das bedeutet, dass die integrierten Ontologien $\Theta_1, \Theta_2, \Theta_3$ zunächst nur in Θ inkludiert sind und Θ keine

weiteren Konzepte, Relationen oder Restriktionen definiert. Solche Erweiterungen vorzunehmen, kann ebenfalls Teil einer Integration sein, um die integrierten Ontologien anzupassen, zu spezialisieren oder zu erweitern [Pin99]. Dabei werden die integrierten Ontologien jedoch unverändert gelassen. So werden in dieser Dissertation *Abbildungsontologien* (engl. *Mapping Ontologies*) $\Theta_{M_{1,2,3}} \supset \Theta$ verwendet, die zusätzliche Beziehungen zwischen den integrierten Ontologien hinzufügen. Damit werden Überschneidungen explizit modelliert, indem beispielsweise Äquivalenzaussagen getroffen werden (z.B. $c' = c''(c' \in \Theta_1, c'' \in \Theta_2)$). Abbildungsbeziehungen können hierbei auch komplexerer Natur sein. Als Beispiel wird eine solche Abbildung im Folgenden in Form einer SWRL-Regel definiert (dabei sind o1: und o2: die Präfixe der Ontologien Θ_1 und Θ_2):

$$\begin{aligned} &\text{o1:NetworkComponent}(?a) \wedge \textit{o1:hasNacRule}(?a, ?r) \\ &\rightarrow \text{o2:Asset}(?a) \wedge \text{o2:Mitigation}(?a) \end{aligned} \tag{2.16}$$

Dies bedeutet, dass eine Netzwerkkomponente nach Θ_1, die eine NAC-Regel nach Θ_1 enthält, ein Asset und eine Gegenmaßnahme nach Θ_2 ist. Komplexe Abbildungen sind besonders dann einzusetzen, wenn Θ_2 abstrakter ist als Θ_1. Dies ist z.B. der Fall für Sicherheitsstandard-Ontologien und detailliertere Systemontologien ist.

Zusammenführung

Bei der Zusammenführung wird aus Ontologien $\Theta_1, \Theta_2, \Theta_3$ die Ontologie Θ erzeugt, sodass nicht mehr zwingend gilt $\Theta_1 \subset \Theta, \Theta_2 \subset \Theta, \Theta_3 \subset \Theta$. In der Regel bilden hier die zu vereinigenden und die resultierende Ontologie die selbe Domäne ab. Bei der Zusammenführung ist meist das Ziel, eine umfassendere Ontologie einer Domäne aus existierenden Ontologien dieser Domäne zu schaffen. Die Methode der Vereinigung kann zum Beispiel angewendet werden, um die TBox für die zu analysierende Systemontologie zu erzeugen. Denn Systemontologien bestehen in der Regel aus mehreren Wissensdomänen.

2.8 Pipelines, Workflows und Workflow-Management-Systeme

Zur Daten- und Informationsverarbeitung werden in der Regel *Pipelines* und *Workflows* eingesetzt. Auch wenn die Begriffe Pipeline und Workflow in der Literatur oft synonym verwendet werden, bezeichnen sie doch unterschiedliche Konzepte, die hier kurz eingegrenzt werden sollen. So ist eine Pipeline in der Informatik eigentlich eine sequenzielle Ausführung nach dem First-In-First-Out-Prinzip, wohingegen ein Workflow einen Ablauf definiert, der einen verbindlichen Start- und Endpunkt besitzt, allerdings auch zustandsabhängige Ausführungsverzweigungen enthalten kann. Es gibt beispielsweise im Sinne von Ad-hoc-Workflows noch flexiblere Varianten von Workflows, die hier jedoch nicht betrachtet werden. Somit kann man schlussfolgern, dass jede Pipeline ein Workflow ist, die Umkehrung jedoch im Allgemeinen nicht gilt.

2.8.1 Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN) stellt eine lesbare, leicht verständliche Notation bereit, die eine standardisierte Brücke zwischen dem Entwurf von Geschäftsprozessen und deren Implementierung bildet. Diese Einführung in BPMN basiert auf der Standardspezifikation von BPMN 2 [Obj13] und geht nur auf die in dieser Dissertation verwendeten Konzepte ein. Der Standard spezifiziert zur Definition von Prozessen verwendete, semantische Konzepte und verknüpft diese mit grafischen Modellierungselementen, Markierungen und Verbindungen. Zudem standardisiert er ein Austauschformat für BPMN, das besonders beim Austausch eines Modells zwischen Modellierungs- und Ausführungswerkzeugen Einsatz findet.
Im Folgenden werden die hier wichtigen Hauptbestandteile von BPMN eingeführt.

2.8.1.1 Prozesse

Generell beschreibt ein *Prozess* einen Fluss von Aktivitäten mit dem Ziel, eine bestimmte Arbeit zu verrichten. In BPMN wird ein Prozess als ein Diagramm von Fluss-Elementen dargestellt, die aus *Aktivitäten*, *Ereignissen*, *Gateways* und *Sequenzflüssen* bestehen, welche eine endliche Ausführungssemantik definieren. Ein Beispiel für einen stark vereinfachten, in BPMN modellierten Prozess zeigt Abbildung 2.5. Die dort zu findenden Elemente werden in den folgenden Paragraphen erklärt.

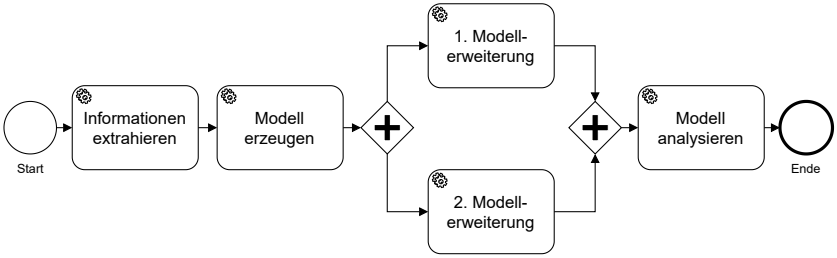


Abbildung 2.5: Einfacher, in BPMN modellierter Prozess.

2.8.1.2 Aktivitäten

Eine *Aktivität* (vgl. Abbildung 2.6) beschreibt eine auszuführende Arbeit in einem Prozess und kann atomar oder zusammengesetzt sein.

Abbildung 2.6: BPMN-Modellierungselemente für Aktivitäten.

Eine atomare Aktivität ist beispielsweise eine *Aufgabe*, die immer dann verwendet wird, wenn eine Aktivität nicht in weitere, feinere Arbeiten aufgeteilt wird. Aufgaben können beispielsweise von Diensten, wie Web-Dienste oder automatisierte Applikationen, ausgeführt werden. Hierfür wird in BPMN der Begriff *Dienstaufgabe* verwendet. Dienstaufgaben haben eine Eingabe- und eine Ausgabemenge.

2.8.1.3 Sequenzflüsse

Ein *Sequenzfluss*, wie er in Abbildung 2.7 zu sehen ist, verbindet zwei Flusselemente eines Prozesses, wie Aktivitäten oder die in den folgenden Paragraphen vorgestellten Ereignisse und Gateways. Der Sequenzfluss zwischen zwei Flusselementen gibt die Reihenfolge dieser Elemente im Prozess an.

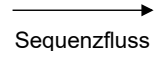


Abbildung 2.7: BPMN-Modellierungselement für einen Sequenzfluss.

2.8.1.4 Ereignisse

Ereignisse sind Vorkommnisse, welche die Ausführungsreihenfolge oder das Timing von Aktivitäten eines Prozesses beeinflussen.

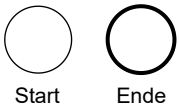


Abbildung 2.8: BPMN-Modellierungselemente für die Ereignisse Start und Ende.

Abbildung 2.8 stellt die einzigen im Rahmen der Forschungsarbeiten eingesetzten Ereignisse *Startereignis* und *Endereignis*, zum Starten und Beenden eines Prozesses, dar. Das Startereignis startet den Sequenzfluss des jeweiligen Prozesses und hat somit auch keinen eingehenden Sequenzfluss. Analog beendet das Endereignis den Sequenzfluss des jeweiligen Prozesses und hat somit auch keinen ausgehenden Sequenzfluss.

2.8.1.5 Gateways

Ein *Gateway* (vgl. Abbildung 2.9) wird entweder verwendet, um zu steuern, wie Sequenzflüsse interagieren, wenn sie innerhalb eines Prozesses konvergieren, oder wie diese divergieren. *Parallele Gateways*, wie sie in Abbildung 2.9 zu sehen sind, werden verwendet, um parallele Flüsse zu generieren oder parallele Flüsse zu synchronisieren. In Beispielabbildung 2.5 werden also die Informationsextraktion und Modellerzeugung sequenziell, die Modellerweiterungsaufgaben jedoch parallel ausgeführt und danach wieder für den darauffolgenden sequenziellen Fluss zur Modellanalyse synchronisiert.

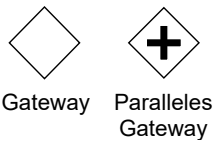


Abbildung 2.9: BPMN-Modellierungselemente für Gateways.

2.9 Software-Defined Networking (SDN)

Software-Defined Networking (SDN) ist ein Netzwerkparadigma, bei dem die Weiterleitungsaufgaben (auf der *Datenebene*, engl. *Data Plane*) von Steuerungsentscheidungen (auf der *Steuerungsebene*, engl. *Control Plane*) entkoppelt

sind [Nun14]. Bei SDN ist die Netzwerkintelligenz in Software-basierten Steuerungseinheiten, den *SDN-Controllern*, zentralisiert. Dahingegen werden die Netzwerkkomponenten zu einfachen Paketweiterleitungsgeräten, die über eine offene Schnittstelle programmiert werden können (z.B. mittels *OpenFlow* [McK08]). Bei der Programmierung installiert die Steuereinheit Weiterleitungsregeln (engl. *Flow*) auf den Netzwerkkomponenten, welche die Komponenten zur Weiterleitung befolgen. Zudem wird mittlerweile oft eine weitere Ebene hinzugenommen, die *Verwaltungsebene* (engl. *Management Plane*). Der Verwaltungsebene werden die Netzwerkapplikationen (engl. *Network Apps*) zugerechnet (wie Load-Balancing oder Firewalls) [Sha19]. Abbildung 2.10 zeigt eine Referenzarchitektur von SDN mit den eben beschriebenen Ebenen.

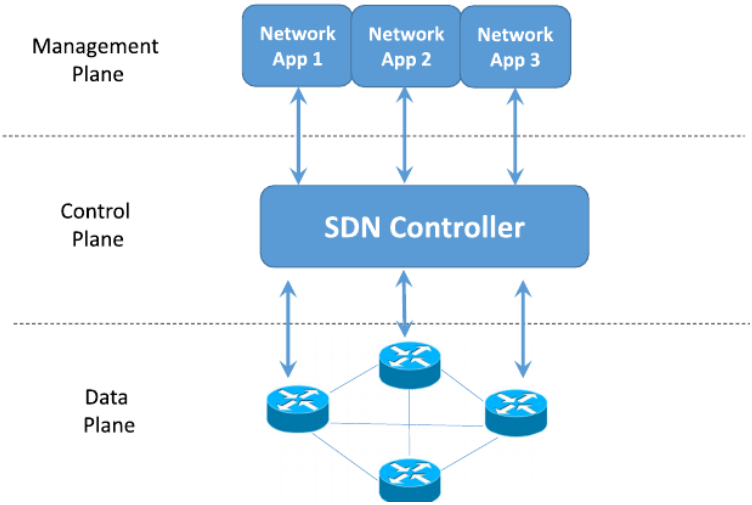


Abbildung 2.10: SDN-Referenzarchitectur mit den drei Ebenen Daten-, Steuerungs- und Verwaltungsebene. *Quelle:* [Sha19].

2.10 Automatisierte Vorfallreaktion und Angriffseindämmung

Automatisierte Vorfallreaktion (*Automated Incident Response (AIR)*), im Weiteren über das englische Akronym AIR abgekürzt, ist die Automatisierung von Schritten, die nach dem Erkennen eines Vorfalls unternommen werden müssen und können. Darunter fällt die Angriffseindämmung, das situative Überwachen und die Generierung von hilfreichen Meldungen für weitere manuelle Behandlung der Vorfälle. Sie fällt damit in den Bereich der Vorfallbehandlung (engl. *Incident Response* (*IR*)) und dient deren Unterstützung.

Durch Überschneidungen zwischen der automatisierten Vorfallbehandlung und Systemen zur Angriffseindämmung (*Intrusion Prevention Systems (IPS)*), können diese nicht klar voneinander getrennt werden. Diese Überschneidungen manifestieren sich insbesondere darin, dass beide Disziplinen schwerpunktmäßig versuchen, einen aktiven Angriff einzuschränken. Die Begriffe trennt eher der Fokus, der bei IPS vermehrt auf dynamische Filterfunktionalität und bei AIR auf der Unterstützung von Vorfallbehandlung liegt.
Unter der Annahme, dass der Terminus IPS keinen definierten Einschränkungen unterliegt, welche deren mögliche Funktionalitäten so einschränken, dass automatisierbare Prozesse der Vorfallbehandlung nicht von ihnen abgedeckt werden können, werden AIR und IPS in dieser Arbeit im Kontext der Vorfallbehandlung als Synonyme erachtet.

Zur Vorfallbehandlung gehören die folgenden zentralen Phasen [Kra11]:

- **Vorbereitung**: Im Vorfeld muss das IR-Team bereits für den Ernstfall vorbereitet werden.
- **Identifikation**: Untersuchung eines Ereignisses, in der Regel gemeldet durch ein Anomalie- oder Vorfalldetektionssystem (engl. *Intrusion Detection System* (*IDS*)) oder einen aufmerksamen Mitarbeiter, sowie Deklaration des Ereignisses als Vorfall und Analyse weiterer wichtiger Aspekte wie Auswirkung und Herkunft des Vorfalls.

- **Abschottung**: Isolation des Problems und Abschottung betroffener von nicht betroffenen Systemen.
- **Bereinigung**: Bereinigung der betroffenen Systeme (z.B. durch Wiederaufspielen eines Backups oder Neuinstallation, bzw. Firmware-Flashen).
- **Wiederherstellung**: Wiederherstellen des gewünschten Zustandes bei gleichzeitiger Sicherstellung, dass betroffene Systeme nicht wieder von dem Vorfall betroffen werden (z.B. durch Aufspielen von Patches und Härtung).
- **Gelernte Lektionen**: Ausführliche Dokumentation des Vorfalls, Nachbesprechung und Evaluation möglicher Verbesserungen.

Automatisierung kann hierbei besonders in den Bereichen Identifikation und Abschottung eingesetzt werden, wie in Abschnitt 4 gezeigt wird.

3 Automatisierte, minimalinvasive Sicherheitsanalyse

Dieses Kapitel befasst sich mit automatisierter, minimalinvasiver Sicherheitsanalyse industrieller Systeme. Dabei wird die modellbasierte Sicherheitsanalyse als Basis zugrunde gelegt.
Die konkreten Beispiele und Untersuchungen in diesem Kapitel beziehen sich auf zwei Beispielsysteme, die in Abschnitt 3.1 vorgestellt werden. Um den Lesenden auch für die modellbasierte Sicherheitsanalyse von Anfang an ein Beispiel zur Verfügung zu stellen, wird in Abschnitt 3.1.3 ein entsprechendes Anwendungsszenario, inklusive relevanter Akteure, präsentiert. Im Anschluss werden Anforderungen an eine adäquate Lösung für die angestrebte automatisierte, minimalinvasive Sicherheitsanalyse industrieller Systeme erfasst (vgl. Abschnitt 3.2). Dem folgend wird ein Überblick des Standes von Wissenschaft und Technik präsentiert (Abschnitt 3.3) und mit diesen Anforderungen verglichen (vgl. Abschnitt 3.3.11).
Im nächsten Schritt werden die in dieser Dissertation erarbeiteten Einzelkonzepte und Methoden zur Erfüllung der aufgestellten Anforderungen und Ziele beschrieben (vgl. Abschnitt 3.4). Diese enthalten Ergebnisse, welche einzelne Problemstellungen und Hypothesen aus Abschnitt 1.1.1 adressieren und in die Entwicklung des SyMP-Rahmenwerks einflossen. Die Einzelkonzepte und Methoden beziehen sich dabei auf die Informationsextraktion, Modellbildung, Modellintegration, Modellerweiterung und Sicherheitsanalyse.
Nach diesen Einzelbetrachtungen wird in Abschnitt 3.5 das angesprochene Rahmenwerk vorgestellt. Dass das Rahmenwerk praktisch umsetzbar ist, wurde mithilfe einer Implementierung gezeigt, welche in Abschnitt 3.6 präsentiert wird. Auf Basis der Implementierung wurden Evaluationen des Rahmenwerks

und spezifischer, bis dahin noch nicht evaluierter, Einzelkonzepte und Methoden durchgeführt, die in Abschnitt 3.7 beschrieben werden. In diesem Rahmen werden auch die Nachweise für die aufgestellten Hypothesen erbracht und die Erreichung der Ziele, sowie die Abdeckung der Anforderungen diskutiert.

3.1 Beispielumgebungen und Anwendungsszenario

In dieser Dissertation werden zwei Beispielumgebungen betrachtet. Eine Proof-of-Concept-Umgebung (kurz *PoC*, vgl. Abbildung 3.2) welche eine realistische Komplexität und eine voll funktionsfähige kleinformatige Produktionsanlage bietet, sowie eine frei konfigurierbare Laborumgebung (vgl. Abbildung 3.3). Beide Umgebungen werden in den folgenden Abschnitten kurz vorgestellt. Da die im Rest dieser Arbeit zu findenden Modellrepräsentationen der Umgebungen aus englischen Begriffen bestehen, wird hier über die Beschreibung der entsprechenden Architekturbilder eine Beziehung zwischen den deutschen und englischen Komponenten- und Netzwerknamen geschaffen.

3.1.1 BSI PoC

Im Rahmen des vom Bundesamt für Sicherheit in der Informationstechnik (BSI) in Auftrag gegebenen Projektes *Projekt 369: Sicherheits- und Funktionsanalysen/ Proof of Concepts für Industrie 4.0*[1], entwickelte das Fraunhofer IOSB eine komplexe Demonstrationsumgebung, die aus einer modernen IT-Architektur mit sechs Netzwerkarchitekturebenen und einer, durch echte Hardware angesteuerten, Fertigungsanlage in Modellgröße besteht. Die industriellen Netzwerkarchitekturebenen des Demonstrators sind Abbildung 3.1 zu entnehmen. Dabei wurde ein Teil der Komponenten virtualisiert und der Rest als Hardware

[1] https://www.evergabe-online.de/tenderdetails.html?1&id=219959, zuletzt zugegriffen: 26.10.2020.

integriert. So befindet sich im Prozessnetzwerk mit SPS-1 lediglich ein virtuelles Gerät. Diese abgebildeten Netzwerkebenen repräsentieren ein typisches industrielles System und enthalten Buskoppler (BK), Speicherprogrammierbare Steuerungen (SPS), eine Mensch-Maschinen-Schnittstelle (HMI), verschiedene Gateways (GW), ein Fertigungsmanagementsystem (MES) sowie weitere Komponenten, die Datenbanken und Protokoll-spezifische Dienste bereitstellen.

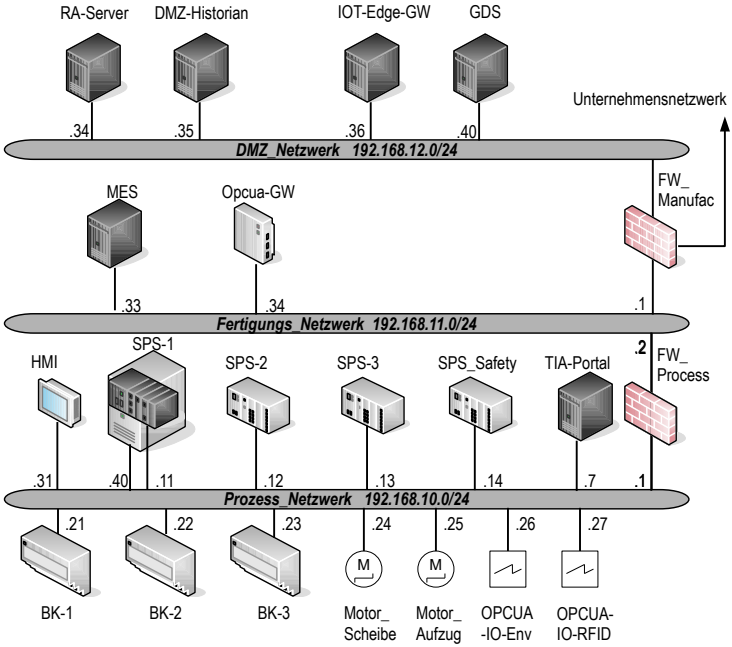


Abbildung 3.1: Netzplan der untersten drei Netzwerkarchitekturebenen des PoC-Demonstrators.

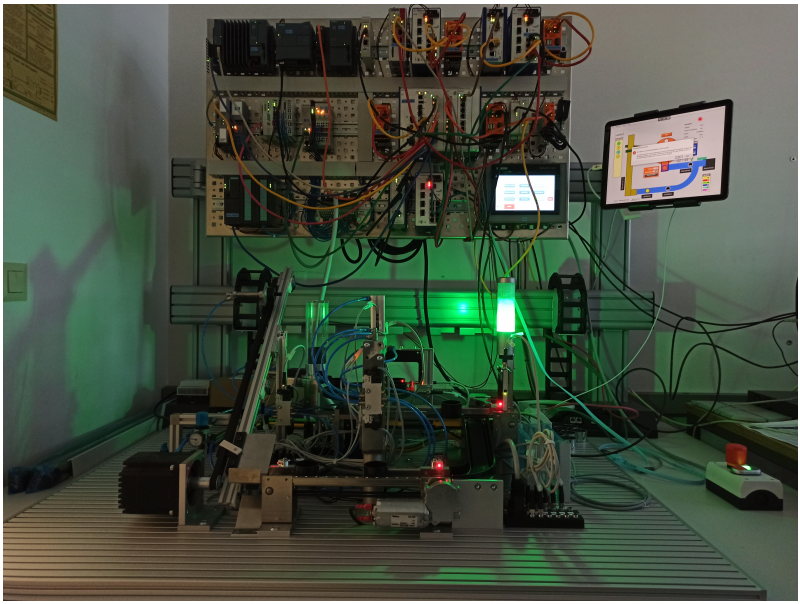


Abbildung 3.2: Bild der Hardwarekomponenten der PoC.

Wie der physische Teil des Demonstrators in der Realität aussieht, zeigt Abbildung 3.2. In der Abbildung ist zu erkennen, wie der Demonstrator eine Fertigungslinie darstellt, bei der Töpfchen stellvertretend für die zu fertigenden Teile durch die Anlage befördert werden. Dieser Töpfchen-transportierende Kreislauf besteht u.a. aus einem oberen und einem unteren Laufband, einer Rutsche (rechts im Bild.), einem Fahrstuhl (links im Bild), einer Rotationsscheibe, verschiedenen Sensoren (Infrarot, RFID, Temperatur, Feuchtigkeit, ...) und einem Roboterarm (mittig im Bild). Damit lassen sich verschiedene Prozessszenarien umsetzen, welche dem Prozess entsprechenden Netzwerkverkehr erzeugen und verschiedene Steuerungsabläufe möglich machen. Beispielsweise können Töpfchen anhand der aktuellen Temperatur- und Feuchtigkeitswerte aussortiert oder über ihre RFID-Tags durch die Anlage hinweg verfolgt werden, um auf MES-Ebene pro Prozessschritt individuelle Entscheidungen für

die Töpfchen treffen zu können. Die darüber liegenden Netzwerksegmente bestehen hingegen aus virtualisierten Komponenten und laufen in der Virtualisierungsumgebung Proxmox VE[1].

3.1.2 Laborumgebung

Da die PoC für eine Vielzahl von Projekten verwendet wurde, konnte diese nicht beliebig rekonfiguriert werden. Evaluationen, welche eine solche beliebige Rekonfiguration benötigten (z.B. für die Konfigurationsanalyse, welche beliebige Firewall-Konfigurationen voraussetzt), wurden auf einer dedizierten Laborumgebung durchgeführt. Diese Umgebung besteht, wie Abbildung 3.3 zeigt, aus vier Netzwerksegmenten, von denen eines das Internet ersetzt. In der Abbildung sind die Endgeräte jeweils mit einem Ubuntu- oder Windows-Logo versehen, damit die darauf laufenden Betriebssysteme erkenntlich sind. Zudem läuft auf den beiden Firewalls die FreeBSD-basierte Software pfSense[2] und auf dem RevolutionPi das Debian-basierte Raspbian[3]-Betriebssystem.
Der IEC-62443-Standardreihe folgend, wurden die Netzwerksegmente und deren Übergänge beispielhaft in vier *Zonen* und vier *Kanäle* (*Conduits*) unterteilt. Sowohl Zonen, als auch Kanäle sind nach der IEC-62443-Standardreihe logische Sicherheitsbereiche, die im Rahmen eines Sicherheitsmanagementprozesses definiert werden. Dadurch lassen sich bestimmte Sicherheitsanforderungen für Bereiche, statt nur einzelne Assets, festlegen und verwalten.
Der RevolutionPi im Feldnetzwerk (Field) ist hier die einzige Hardware-Komponente. Die restlichen Komponenten befinden sich in einer Proxmox-Virtualisierungsumgebung. In der Abbildung sind außer den bereits genannten Komponenten Arbeitsstationen (Workstations) im Unternehmensnetzwerk (Enterprise), eine Steuerstation (Control Station), ein Cloud-Server, sowie ein Web-, ein DNS- und ein Analyse-Server (Ontology & Analysis Platform) in der Demilitarized Zone (DMZ) zu sehen. Der Analyse-Server dient dabei als Senke für die Informationsextraktion und als Plattform für die

[1] https://www.proxmox.com/de/proxmox-ve, zuletzt zugegriffen: 21.12.2020.
[2] https://www.pfsense.org, zuletzt zugegriffen: 21.12.2020.
[3] https://www.raspbian.org, zuletzt zugegriffen: 21.12.2020

in Abschnitt 3.6 vorgestellte Implementierung der Modellverarbeitungs- und Analysekonzepte.

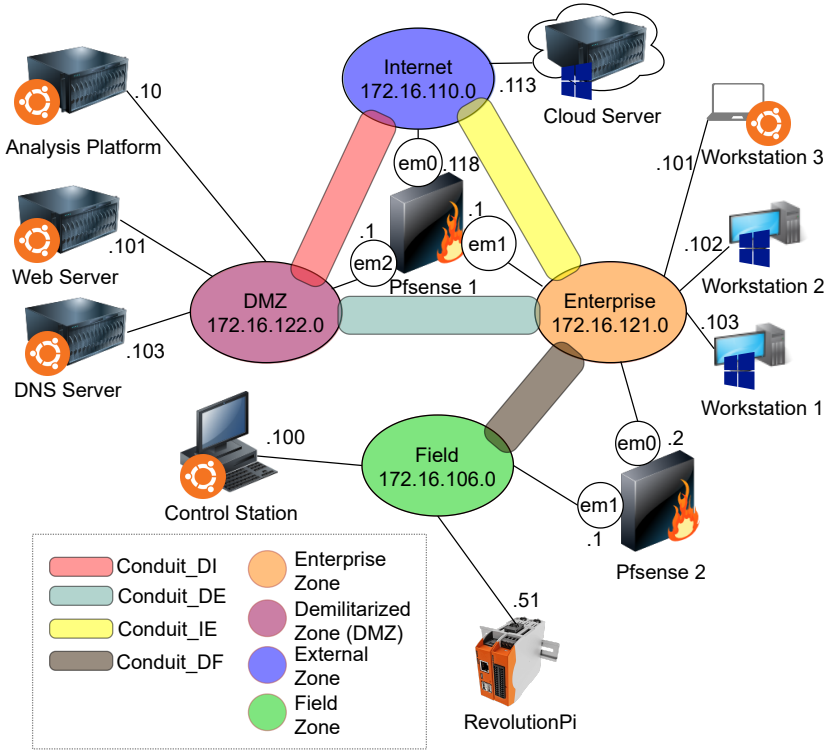


Abbildung 3.3: Architektur der flexiblen Laborumgebung.

3.1.3 Anwendungsszenario

Wie in der Einleitung bereits gezeigt, handelt es sich bei der modellbasierten Sicherheitsanalyse um einen Ablauf, der in Informationsextraktion, Modellbildung, Modellerweiterung/-verarbeitung und Analyse gegliedert werden

kann. Zudem wurde in der Einleitung bereits beschrieben, wie sich die hier behandelten Analysearten in einen Sicherheitslebenszyklus einordnen lassen. Hierzu soll nun ein anschauliches Anwendungsszenario konstruiert werden, welches die Nachvollziehbarkeit der folgenden Abschnitte zur Sicherheitsanalyse unterstützen soll.
Das Szenario ist in Abbildung 3.4 dargestellt und entspricht einem, auf die Sicherheitsanalyse fokussierten Sicherheitslebenszyklus, der aus den beiden bereits vorgestellten Abläufen von NIST 800-37 (vgl. Abschnitt 1) und IEC 62443-3-2 (vgl. Abschnitt 2.6) abgeleitet wurde. Wie auch bei den Standards, handelt es sich bei dem Ablauf aus Abbildung 3.4 um eine Vereinfachung. Es sind sowohl weitere Schritte, als auch andere Ablaufvarianten möglich. Für dieses Szenario wird die Laborumgebung als „System von Interesse“ betrachtet. Hauptakteur des Szenarios sei hier der Betreiber dieser Umgebung.
Um Analysen des Systems durchführen zu können, müssen zunächst dessen Assets erfasst werden. Hier sind dies Komponenten, Kanäle und Zonen. Die Erfassung sollte idealerweise fortlaufend erfolgen, um Änderungen so früh wie möglich aufzunehmen. Beteiligte Interessengruppen sind dabei, abhängig von der Automatisierung dieses Informationsextraktionsschrittes, u.a. Administratoren, Operatoren, Dienstleister und Hersteller.
In einem darauffolgenden Schritt müssen die Beziehungen zwischen den Assets und eventuell weiterer Kontext (z.B. Einsatzorte) erfasst werden. Durch die vorangegangene Modellierung der Assets, werden somit also einzelne Modelle (Komponentenmodelle) verknüpft und erweitert. Dabei kommt Systemdomänenwissen, insbesondere IT/OT-Wissen, zum tragen, das typischerweise Administratoren oder andere technologiefokussierte Rollen im Unternehmen besitzen. Beispielhaft wird in diesem Szenario davon ausgegangen, dass die IPv4-Netzwerkzugehörigkeit abgeleitet wird, wodurch explizit modelliert wird, welche Assets sich im selben IPv4-Netzwerk befinden. Nur so ist diese Information auch maschinenlesbar. Jegliche Erweiterung des Modells benötigt neben der Expertise aus der Systemdomäne auch Modellierungsexpertise. Entscheidend für die Ziele der Erweiterungen ist dabei jedoch das Wissen von Administratoren und weiteren Systemexperten.

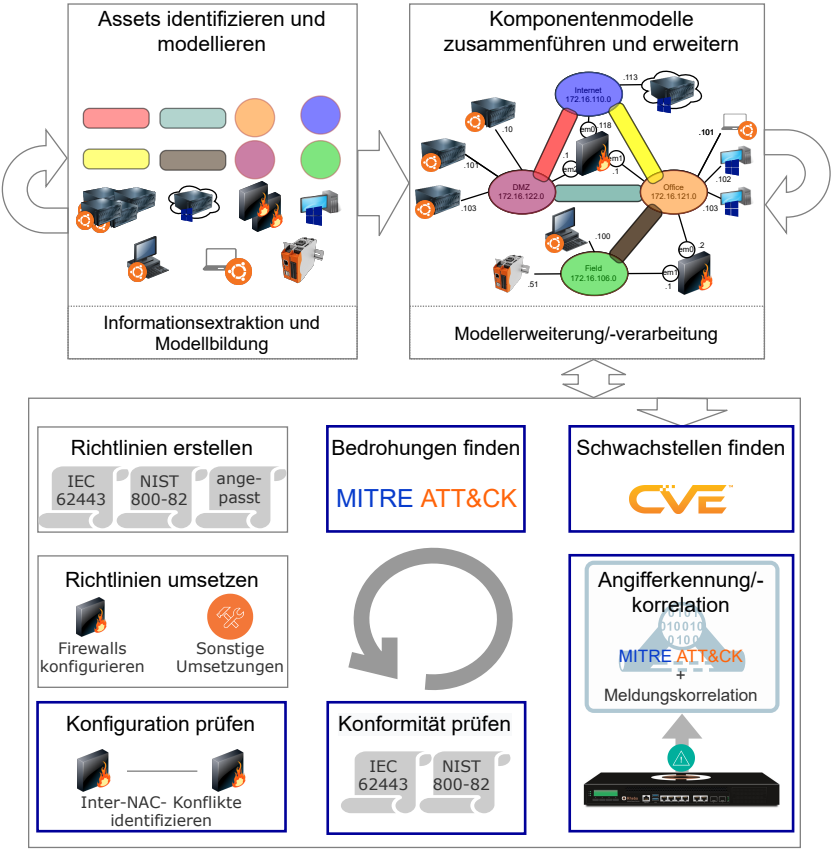


Abbildung 3.4: Anwendungsszenario mit Modellbasis (oben) und vereinfachtem Sicherheitslebenszyklus (unten) bei dem Analysen in blauen/dicken Rahmen kenntlich gemacht wurden.

Der in der unteren Hälfte von Abbildung 3.4 zu sehende, beispielhaft konkretisierte Sicherheitslebenszyklus kann bei dem Auffinden von Schwachstellen (Schwachstellenanalyse) begonnen werden. Beispielhaft wird in diesem Szenario davon ausgegangen, dass die im System befindlichen Softwarepakete mit einer CVE-Datenbank abgeglichen werden (vgl. Abschnitt 2.6.2).

Danach werden Bedrohungen für die Assets gesucht, was in diesem Szenario durch die Identifikation anwendbarer ICS-ATT&CK®-Techniken repräsentiert wird.
Sowohl für die Suche nach CVEs als auch nach den ICS-ATT&CK®-Techniken reichen die im Modell verfügbaren Informationen gegebenenfalls nicht aus, weshalb die Ausführungen der Analysen weitere Modellerweiterungen erfordern. Dies gilt auch für die restlichen Analysen, weshalb die in Abbildung 3.4 ersichtliche Beziehung zwischen den Analysen und der Modellerweiterung/-verarbeitung nicht nur Informationsflüsse, sondern auch Interaktion widerspiegelt. An diesem Punkt kommt also auch analysespezifisches Wissen zum Einsatz, welches die Einbindung von Sicherheitsexpertenwissen in die Modellerweiterungsvorgänge verlangt.
Entsprechend den Ergebnissen der vorherigen Analysen, die in der Regel im Rahmen einer Risikoanalyse durchgeführt werden (vgl. Abschnitt 2.6), werden Richtlinien zur Risikoreduzierung erstellt. Beispiele für technisch fokussierte Richtlinien seien in diesem Szenario aus den Standard-/Leitfadenauszügen IEC 62443-3-2 ZCR-3.2, NIST 800-82 5.5, NIST 800-82 5.3.1 und NIST 800-82 5.3 abgeleitet und beziehen sich damit auf NAC-Konfigurationen, die Netzwerkverkehr zwischen industriellen Netzabschnitten und Unternehmensnetzen, sowie dem Internet einschränken (in Abschnitt 3.4.3.2 werden die Auszüge detaillierter beschrieben, hier soll deren genereller Fokus vorerst genügen).
Die Richtlinien werden infolgedessen mittels verschiedenster Gegenmaßnahmen umgesetzt. Beispielhaft sei hier die Konfiguration der beiden Firewalls der Laborumgebung erwähnt. Dabei wird das System verändert, weshalb somit auch dessen Modellrepräsentation aktualisiert werden muss.
Ob die Umsetzung auch korrekt ist, hängt gerade bei komplexeren, Mehr-Geräte-Konfigurationen davon ab, ob diese konfliktfrei sind. In diesem Szenario wird im Rahmen der Konfigurationsanalyse geprüft, ob Konflikte zwischen den beiden Firewall-Konfigurationen bestehen. Dies ist zum Beispiel dann der Fall, wenn die Regelkette von PfSense1 Pakete eines IPv4-Absenderraums zu einem IPv4-Empfängerraum erlaubt, die von der Regelkette in PfSense2 geblockt werden. Die Unstimmigkeit zeugt dabei von einer inkonsistenten Konfiguration und deutet auf einen möglichen Konfigurationsfehler hin.

Weiter wird die Konformität der Systemkonfiguration zu den erstellten Richtlinien geprüft. Demnach besteht die Konformitätsanalyse in diesem Szenario aus der Prüfung der Systemkonfiguration bezüglich den zuvor genannten, aus den Standards/Leitfäden IEC 62443 und NIST 800-82 abgeleiteten Richtlinien. Zuletzt sollen im Rahmen der Angriffserkennung und -korrelation noch Meldungen als Indikatoren für bestimmte ICS-ATT&CK®-Techniken und darüber hinaus mehrere Meldungen unterschiedlicher Techniken als bestimmte Angriffsvektoren klassifiziert werden. Hierfür müssen die Meldungen, sowie das Wissen auf das sie sich beziehen, Teil des Systemmodells werden, da die Ergebnisse nur so automatisiert auf die Assets des Systems bezogen werden können. In Abbildung 3.4 wird der Kreislauf an dieser Stelle durch den Übergang auf die Schwachstellenanalyse geschlossen. Dies ist eine starke Vereinfachung, da sich nicht in jeder Situation alle Schritte vollumfänglich wiederholen und zudem nicht streng sequenziell abgearbeitet werden. Da das Szenario jedoch der besseren Nachvollziehbarkeit der Anwendung der Sicherheitsanalysen, deren Eingliederung in Sicherheitslebenszyklen und der Notwendigkeit des Einbeziehens verschiedener Interessengruppen in die Modellerweiterung dienen soll, wurde auf diese Komplexität bewusst verzichtet.
Das Szenario, wird nur durch den Einsatz einer Sicherheitsanalyselösung ermöglicht, wie sie hier angestrebt wird und bildet damit nicht den aktuellen Stand der Technik ab. Im Weiteren Teil dieser Dissertation wird zur besseren Einordnung von Teilproblemen und Überlegungen wiederholt auf das hier beschriebene Szenario verwiesen. Insbesondere wird dieses Szenario der Evaluation des bereits erwähnten Rahmenwerks zugrunde gelegt.

3.2 Anforderungen für automatisierte Sicherheitsanalyse

Wie das Anwendungsszenario aus Abschnitt 3.1.3 bereits erahnen lässt, unterliegt ein konsistentes Gesamtkonzept für die automatisierte Sicherheitsanalyse industrieller Systeme einer Vielzahl von Anforderungen, um einen adäquaten Erfüllungsgrad der gewünschten Flexibilität-, Wiederverwendbarkeits-,

Anpassbarkeits- und Abdeckungseigenschaften sicherzustellen. Diese nichttrivialen Anforderungen wurden im Rahmen dieser Dissertation identifiziert und werden in diesem Abschnitt erörtert. Nach der Einführung des Standes von Wissenschaft und Technik, wird dessen Erfüllung dieser Anforderungen in Abschnitt 3.3.11 untersucht. Ein Vergleich mit der Anforderungsabdeckung des später in diesem Kapitel vorgestellten Rahmenwerks, wird in Abschnitt 3.7.7 präsentiert.

3.2.1 Umgebungsbezogene Anforderungen

In den folgenden Absätzen werden Anforderungen vorgestellt, welche durch den gewünschten Einsatz in und für industrielle Umgebungen entstehen. Diese werden mit UA1-UA4 durchnummeriert, wobei UA für „**U**mgebungsbezogene **A**nforderung“ steht.

UA1 - Minimalinvasiv

Lösungen für den industriellen Bereich müssen so wenig invasiv wie möglich sein, um kritische Abläufe nicht zu stören und Garantien für beispielsweise Echtzeit, Verfügbarkeit und Betriebssicherheit einhalten zu können. Für Sicherheitsanalyseanwendungen bedeutet dies, ausschließlich nicht-gefährdende oder sogar passive Informationsextraktionsmethoden einzusetzen (vgl. Abschnitt 2.4).

UA2 - Ressourcenschonend

Es kann nicht davon ausgegangen werden, dass Komponentenhersteller Verfahren adaptieren und unterstützen, welche den Ressourcenverbrauch der auf industriellen Komponenten laufenden Software signifikant erhöhen [Nat11]. Daher müssen auf den Komponenten leichtgewichtige Verfahren eingesetzt werden oder es muss sogar auf generische Lösungen zurückgegriffen werden, welche für andere bzw. allgemeine Zwecke entworfen wurden.

UA3 - Quellheterogenität

Sicherheitsanalyse lebt von den zugrundeliegenden Informationen. Besonders SCM, Schwachstellen- und Konformitätsanalysen benötigen detaillierte sicherheitsrelevante Daten von verschiedensten Komponenten. Da in industriellen Systemen nicht davon auszugehen ist, dass diese Informationen alle durch den selben Quelltyp (z.B. einheitliche Selbstauskunft oder einheitlicher Engineering-Werkzeug-Export) zugänglich sind, muss eine adäquate Lösung verschiedene Quellen unterstützen und zukünftige Quelltypen adaptieren können.

UA4 - Abdeckung industrieller Assets

Geräte industrieller Umgebungen besitzen, wie bereits in Abschnitt 1.1.1 erörtert, nicht dieselben Schnittstellen und Informationsschemata zur Informationsextraktion wie Geräte der Büro-IT. Daher wird mit dieser Anforderung eine Unterstützung von Informationsextraktionsmöglichkeiten industrieller Systeme dediziert gefordert.

3.2.2 Anforderungen an Informationsextraktion und -repräsentation

Die Extraktion und Darstellung von sicherheitsrelevanten Informationen sind in industriellen Umgebungen besondere Herausforderungen, deren Anforderungen in diesem Abschnitt aufgeführt werden. Diese Anforderungen sind mit IA1-IA6 nummeriert, wobei IA für „**I**nformationsextraktion- und Informationsrepräsentation-**A**nforderung“ steht.

IA1 - Sichere Informationsextraktion

Bei der Informationsextraktion ist darauf zu achten, Vertraulichkeit, Integrität und Authentizität der Übertragung sicherzustellen. Besonders die Vertraulichkeit von Informationen ist in industriellen Systemen keine Selbstverständlichkeit, ist aber im Falle der Übertragung von sicherheitsrelevanten Informationen unabdingbar. Ein Dolev-Yao-Angreifer [Dol83] soll nicht die Möglichkeit haben, diese Informationen mitzulesen oder sogar zu manipulieren. Im Dolev-Yao-Modell wird ein Angreifer als (polynomiell-beschränkter) Teilnehmer des Netzwerks definiert, der die Fähigkeiten hat, Nachrichten im Netzwerk abzufangen, zu modifizieren, zu löschen, zu duplizieren und als jeder andere Netzteilnehmer an jeden beliebigen Netzteilnehmer zu senden.

IA2 - Reduktion der Abbildungskomplexität

Informationsquellen, wie Endgeräte, Netzwerkkomponenten oder Engineering-Software, besitzen eine eigene Informationsrepräsentation (bzw. ein eigenes Informationsmodell), welche durch eine bestimmte Serialisierung zugängig ist. Das Zielmodell und die Zielserialisierung, welche als Datenbasis für (Analyse-)Anwendungen eingesetzt werden, weicht in der Regel davon ab. Daher wird eine Abbildung oder Transformation von Quell- und Zielrepräsentation benötigt (vgl. Abbildung 3.5).

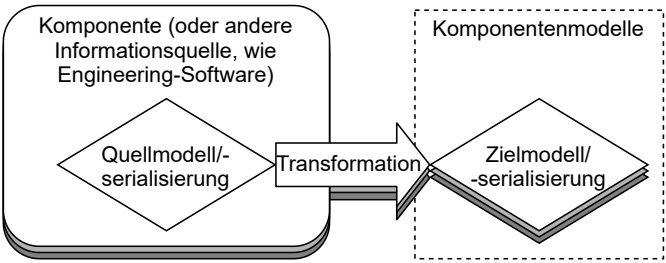


Abbildung 3.5: Visualisierung des Verhältnisses von Quell- zu Zielmodellen und -serialisierungen.

Verwandte Arbeiten lagern oft entweder die Abbildung vom Quell- auf das Zielmodell aus oder verlangen Individualabbildungen und widmen sich dieser Aufgabe daher nicht. Diese Anforderung soll zum Ausdruck bringen, dass die Abbildungen Teil der Lösung und skalierbar sein sollen.

IA3 - Unterstützung von Standards und konsolidierten Spezifikationen zu Informationsmodellen

Für die Sicherheitsanalyse werden verschiedene Informationsdomänen genutzt, die eigene, konsolidierte Quellinformationsmodelle mit z.B. spezifizierten Schemata besitzen (vgl. Umgebungsanforderung UA3). Im aktuellen Stand der Forschung wurde dies weitgehend ignoriert, indem Daten direkt in Schemata oder Ontologien einer Sicherheitsdomäne geparst wurden. Allerdings kollidiert dieses Vorgehen direkt mit UA2 und UA3 und schränkt zudem die Flexibilität bezüglich der, auf dem Modell arbeitenden, Anwendungen ein. Daher ist eine Unterstützung verschiedener Informationsdomänen auf der Ebene entsprechender konsolidierter Spezifikationen zu deren einheitlicher Modellierung nötig.

IA4 - Unterstützung verschiedener Serialisierungen

Diese Anforderung ist das, auf die Serialisierung bezogene, Pendant zu IA3. Sie ist erfüllt, wenn die Lösung Transformationen beliebiger Serialisierungen (und Syntax) zur Zielserialisierung (und -syntax) unterstützt.

IA5 - Unterstützung von Zielmodellen

Das direkte Analysieren der Quellmodelle würde IA2, IA3 und IA4 obsolet machen. Da somit jedoch unter anderem verhindert würde, dass Modellverarbeitungsschritte und Analysen Quelltyp-unabhängig definiert, wiederverwendet und ausgetauscht werden können, muss die Unterstützung von Zielmodellen gesichert sein, die bereits vor der Erzeugung des Systemmodells aus den Komponentenmodellen feststehen.

IA6 - Einfache Anpassung der Quellen

In vielen realen Systemen gibt es häufige Änderungen an der Zusammenstellung des Systems. Die Informationsextraktionslösung muss diese Änderungen unterstützen, um praxistauglich zu sein.

3.2.3 Anforderungen an Modellbildung und -verarbeitung

Wie in dem vorherigen Abschnitt beschrieben, wird von einer angemessenen Lösung eine Abbildung von Quellrepräsentationen auf eine Zielrepräsentation verlangt. In den folgenden Absätzen werden Anforderungen für die Bildung eines Modells aus den in der Zielrepräsentation dargestellten Informationen und für das Erweitern des Modells gelistet. Diese wurden mit den Nummerierungen MA1-MA9 versehen, wobei MA für „**M**odellbildung- und Modellverarbeitung-**A**nforderung“ steht.

MA1 - Einsatz von Ontologien zur Modellierung

Da die Verwendung reiner Taxonomien es erfordert, ein Datenmodell in der Software zu kodieren, ist mit jeder Änderung des Datenmodells auch eine Änderung der Software notwendig. Auch fehlen in Taxonomien nötige Konzepte, um Reasoning zu ermöglichen [Und03]. Ein weiterer wichtiger Vorteil von Ontologien ist deren Erweiterbarkeit, Mehrfachverwendung und Austauschbarkeit zwischen verschiedenen Systemen. Bereits 2001 wurde von Raskin et al. [Ras01] festgestellt, dass Ontologien alle sicherheitsrelevanten Aspekte eines Systems auf jedem Detailgrad organisieren, systematisieren und eine große Anzahl von heterogenen Instanzen auf eine kleine Anzahl von Eigenschaften reduzieren können. Die Autoren schlussfolgerten außerdem, dass Ontologien gerade deswegen besonders für die Darstellung von sicherheitsrelevanten Informationen und die Analyse auf diesen Informationen geeignet sind. In einer Umgebung, in der eine enorme Vielfalt von Konfigurationen analysiert werden muss, ist der Einsatz von Ontologien demnach eine angemessene Lösungsstrategie. Außerdem soll eine adäquate Lösung in der Lage sein, existierende

Wissensformalisierungen und Analyseverfahren zu unterstützen, die, wie eingangs motiviert, zum Großteil auf Ontologien setzen. Aus diesen Gründen wird der Einsatz von Ontologien hier als Anforderung formuliert.

MA2 - Einsatz monotoner Beschreibungslogiken zur Modellierung und Modellerweiterung

Monotone Beschreibungslogiken bieten die nötigen Garantien für Vollständigkeit, Korrektheit und Entscheidbarkeit, um automatisiert neues Wissen abzuleiten und dabei kein vorhandenes Wissen zu annullieren. Besonders bei wiederverwendbarer Logik sind diese Garantien von Vorteil, da der tatsächliche Einsatzort (das entsprechende instanziierte Modell) vorab nicht unbedingt bekannt ist und hierbei durch die Garantien Fehlfunktionen und nicht terminierende Anwendungen vermieden werden können.

MA3 - Unterstützung von nicht-DL Formalismen und Algorithmen

Aufgaben wie Kalkulationen oder das Hinzufügen von Individuen sind nicht monoton und können dadurch nicht mit den monotonen DL-Formalismen und -Algorithmen umgesetzt werden (vgl. Abschnitt 2.7.2.1). Zudem ist ein generisches Reasoning nicht so performant, wie ein auf ein spezifisches Problem optimiertes Verfahren (vgl. Abschnitt 3.7.2 Abbildung 3.43). Daher ist Verarbeitung ohne formale Logik, unter Einhaltung der in MA2 angesprochenen Garantien, ebenfalls zu unterstützen.

MA4 - Klare Trennung zwischen der Verwendung von Beschreibungslogiken und anderer Formalismen

Bei gleichzeitiger Unterstützung von DL- und nicht-DL-Verfahren müssen Verarbeitungsschritte mit Garantien durch die Beschreibungslogik klar von Schritten getrennt werden können, bei denen besondere Vorsicht in der Algorithmen- und Modellentwicklung geboten ist. So ist klar erkennbar, ob ein Erweiterungsmechanismus direkt verwendet werden kann oder ob der Verwendung

bestimmte Bedingungen zugrunde liegen, welche zunächst geprüft werden müssen.

MA5 - Separation-of-Concerns

Im Rahmen von modellbasierter Sicherheitsanalyse gibt es mehrere Akteure (vgl. Abschnitt 3.1.3). Genauer ist sogar jeder Experte einer Domäne (Netzwerke, Betriebssysteme, OT, ISO 27000, IEC 62443, ...) und jeder Anwender im Unternehmen ein relevanter Akteur, da sie über Wissen verfügen, das für die Systemmodellierung und/oder Sicherheitsanalyse relevant ist. Demnach gibt es unter diesen Experten unterschiedliches Wissen, unterschiedliche Fähigkeiten und unterschiedliche Rollen im Bezug auf die Analyselösung. Daher ist eine klare Trennung von Tätigkeiten, Domänen und Werkzeugen, sowie eine klare Definition der Schnittstellen zwischen den Akteuren erforderlich. Der Erfolg der Lösung bezüglich des Teilens und Austausches von Ontologien, Analysen, Richtlinien und Werkzeugen, sowie die Abdeckung von Komponenten ist von dieser Anforderung abhängig.

MA6 - Austauschbare Erweiterungslogik

Da Modellerweiterungen abhängig von Interpretationen, unternehmensinternen Gegebenheiten und Domänen sind und zudem oft optimiert werden können, soll die Möglichkeit gegeben werden, diese Logiken auszutauschen. Ein Beispiel hierfür ist das, im Anwendungsszenario (Abschnitt 3.1.3) erwähnte, Ableiten der Zugehörigkeit von Netzwerkteilnehmern zu einem IPv4-Netz, die anhand der IPv4-Adressen des Teilnehmers vorgenommen werden kann. Allerdings kann eine IPv4-Netzwerkadresse für mehrere Netzwerke verwendet werden, z.B. bei Einsatz von *Network Address Translation (NAT)*[1], oder die Ableitung muss zusätzlich von physischen Aspekten, wie der zugehörigen Maschinenzelle oder physischen Netzwerkverbindungen, abhängen. Solche Aspekte können nur durch austauschbare Erweiterungslogik abgedeckt

[1] NAT ist eine Methode zur Abbildung von einem IPv4-Adressraum auf einen anderen IPv4-Adressraum.

werden. Darüber hinaus ist die Austauschbarkeit eine Voraussetzung für die angestrebte Wiederverwendbarkeit der Erweiterungslogik.

MA7 - Analysespezifische Erweiterung

Verschiedene Analysen können verschiedene Modellerweiterungen voraussetzen. Diese können sich widersprechen und sollten somit nicht gleichzeitig angewandt werden, da das Modell infolgedessen inkonsistent wäre. Ein Beispiel ist die Ableitung von Netzwerkdienstnamen aus verschiedenen Quellen (bspw. /etc/services bei Debian-basierten Linux-Distributionen oder aus einer Applikationskonfiguration). Diese können sich für denselben TCP- oder UDP-Port unterscheiden. Soll die Abbildung verwendet werden, können unterschiedliche Analysen verschiedene Abbildungsquellen fordern oder erwarten. Zudem kann die Anzahl der Analysen, besonders durch die Unterstützung verschiedener Domänen, sehr groß werden und bestimmte Modellerweiterungen zeitintensiv sein. Daher skalieren Lösungen nicht, wenn sie alle Erweiterungen für jegliche Analysen bei der Modellverarbeitung ausführen.

MA8 - Abhängige Verarbeitungsschritte

In der Regel sind sowohl Analysen von Modellverarbeitungs- und -erweiterungs-schritten, als auch Modellverarbeitungsschritte voneinander abhängig. Solche Abhängigkeiten müssen auch technologieübergreifend (z.B. DL und nicht-DL) unterstützt werden. Dies gilt selbst für monotone Erweiterungen, beispielsweise durch (monotone) SWRL-Regeln. So kann eine SWRL-Regel von der Verfügbarkeit einer Ontologie im Modell abhängen, die erst in einem bestimmten Erweiterungsschritt hinzugenommen wird (bspw. weil es sich bei der Ontologie um die Repräsentation einer Applikationsdomäne handelt).

MA9 - Unterstützung von Nutzerinteraktion bei Modellbildung und -verarbeitung

Einige Informationen können dem Systemmodell nur manuell hinzugefügt werden. Hierzu gehören beispielsweise unternehmensinterne Klassifikationen von Netzsegmenten im Sinne der IEC-62443-Sicherheitszonen. Daher muss die Anbindung von Werkzeugen an den Modellbildungsprozess, oder zumindest eine Bearbeitung des Systemmodells mit einem Editor, unterstützt werden. Dies ist besonders für die automatisierte Ausführung von Analysen relevant.

3.2.4 Anforderungen an Sicherheitsanalysen

Besondere Anforderungen müssen auch an die Sicherheitsanalysen selbst gestellt werden. Diese sind im folgenden mit den Kürzeln AA1-AA6 gelistet, wobei AA für „**A**nalyse-**A**nforderung“ steht.

AA1 - Unterstützung der behandelten Analysearten

Die Lösung soll Schwachstellen-, Konfigurations-, Konformitäts- und Bedrohungsanalysen, sowie Angriffserkennung und -korrelation unterstützen. Diese Analysen sind sowohl für die Risikobewertung, als auch für die Evaluation des Sicherheitsstatus des aktuellen Systems relevant und werden für die Wahl von Gegenmaßnahmen, Systemhärtung und Verbesserung von Gegenmaßnahmen und deren Konfiguration benötigt.

AA2 - Technische Evidenz

Die gängige Methode zur Konformitäts- und Risikoermittlung ist die abstrakte Modellierung der Systeme ohne konkrete Konfigurationsinformationen (vgl. Abschnitt 3.3). Dies gilt insbesondere für den Einsatz in industriellen Systemen. Gleichzeitig ist aber die Bestimmung, ob eine Maßnahme konform zu bestimmten Richtlinien ist oder ob ein Risikowert gemindert oder erhöht werden soll, meist abhängig von dieser konkreten Konfiguration. Daher wird

mit dieser Anforderung verlangt, für die Ableitung solcher Schlussfolgerungen den Bezug zur tatsächlichen Konfiguration zu erhalten, um aussagekräftige Schlussfolgerungen zu ermöglichen.

AA3 - Flexible Richtlinien und Regeln

In der Regel gibt es eine Definitionslücke zwischen den abstrakten Richtlinien (von Standards und Best-Practices) und deren Interpretationen. Da die Interpretationen unternehmensspezifisch sein können und zudem unternehmensinterne Richtlinien prüfbar sein müssen, muss die Lösung austauschbare, selbst definierbare Richtlinien und Regeln unterstützen.

AA4 - Austauschbare/Wiederverwendbare Analysen

Für das Teilen von Analysen (bzw. Expertenwissen) und das Anpassen oder Ersetzen durchzuführender Analysen, z.B. um den jeweiligen Unternehmenskontext zu unterstützen, muss deren Austauschbarkeit und Wiederverwendbarkeit gewährleistet werden.

AA5 - Anpassungsfähigkeit

Die Analyselösung muss eine adäquate, praxistaugliche Strategie zur Anpassung an geänderte Systemzusammenstellung und Richtlinien, sowie neue Schwachstellen, Bedrohungen und Angriffsmuster aufweisen.

AA6 - Sicherheitsdomänen

Verschiedene Analysen verwenden nicht nur unterschiedliche analysespezifische Konzepte, sondern auch verschiedene Interpretationen der jeweiligen Sicherheitsdomänen. Selbst für eine Analyseart gibt es verschiedenste Interpretationen von Konzepten wie Richtlinien oder Sicherheitszielen. Außerdem definieren verschiedene Sicherheitsstandards, Leitfäden und Applikationsdomänen ebenfalls unterschiedliche Interpretationen von Sicherheitsbegriffen

und -konzepten. Somit wird mit dieser Anforderung auch die Austauschbarkeit von Sicherheitsdomänen gefordert.

3.2.5 Generelle Anforderungen

Auch gibt es Anforderungen, die genereller Natur sind und den vorherigen Anforderungstypen nicht zugewiesen werden können. Diese werden hier aufgeführt und sind mit GA1-GA5 durchnummeriert, wobei GA für „**G**enerelle **A**nforderung“ steht.

GA1 - Automatisierung

Wie bereits in Abschnitt 1.1.1 motiviert ist die Maximierung der Automatisierung von Informationsextraktion, Modellbildung, Modellverarbeitung und Analysen erstrebenswert und für eine hohe Abdeckung von Komponenten und Analysen notwendig. Daher erfüllen diese Anforderung nur Lösungen, welche die genannten Aufgaben automatisieren.

GA2 - Skalierbarkeit

Industrielle Systeme können aus vielen hundert oder tausend Komponenten bestehen. Um Praxistauglichkeit erreichen zu können, muss eine Lösung hinreichend skalierbar sein. Daher werden Skalierungsmaßnahmen benötigt, die beispielsweise Parallelisierung, verteilte Verarbeitung, Aufgabenzusammenschluss zur Ausführung in einem Prozess und Vermeidung von Mehrfachausführung umfassen.

GA3 - Nutzbarkeit

Eine relevante Anforderung für die Praxistauglichkeit der Lösung ist ihre Nutzbarkeit. Dabei muss darauf geachtet werden, verschiedenen Akteuren (bzw. Stakeholdern) die Arbeit mit der Lösung zu ermöglichen. Unabhängig von der Ausprägung der Lösung muss eine übersichtliche Verwaltung von

Informationsquellen, Modellverarbeitungsschritten und Analysen ermöglicht werden. Lösungen, die ein direktes Editieren eines Modells durch den Endnutzer verlangen, können nur von Modellexperten administriert und angepasst werden. Dieses offensichtlich weit verbreitete Vorgehen (vgl. Abschnitt 3.3) ist mit einer Firewall-Lösung vergleichbar, für die man bei jeder Änderung der Regeln einen Experten des Firewall-Herstellers benötigt und ist somit unflexibel und unwirtschaftlich.

GA4 - Offenheit

Eine angemessene Gesamtlösung sollte Forschern zugänglich sein, um diesen zu ermöglichen, Best-Practices und kollaborative, konsolidierte Extraktions-, Modellbildungs-, Modellerweiterungs- und Analyselösungen zu entwickeln. Aus diesem Grund wird die Offenheit der Lösung gefordert. Hierfür ist es hinreichend, dass die Konzepte und Methoden detailliert zugänglich sind und Forscher die entsprechenden Werkzeuge somit selbst entwickeln könnten.

GA5 - Lebenszyklusunterstützung

Der Systemlebenszyklus sollte über den Sicherheitslebenszyklus (vgl. Abschnitt 3.1.3) unterstützt werden. Das bedeutet, dass eine Analyse nicht nur in einer Phase des Systems einsetzbar sein sollte, sondern in verschiedenen Phasen. So sollten Bedrohungs-, Schwachstellen-, Konfigurations- und Konformitätsanalysen beim Entwurf des Systems, beim Entwurf von Änderungen und während der Produktivphase eines Systems durchgeführt werden können und dabei immer die aktuelle Datenbasis, im Sinne des Digitalen Zwillings, berücksichtigen.

3.3 Stand von Wissenschaft und Technik

In den folgenden Abschnitten wird der zum Zeitpunkt des Verfassens dieser Arbeit bekannte Stand von Wissenschaft und Technik in relevanten und verwandten Bereichen zusammengefasst.
Zunächst wird hier auf Arbeiten im Bereich von Sicherheitsontologien eingegangen, welche die Grundlage für ontologiebasierte Verfahren bilden.
Daraufhin werden Arbeiten verschiedener Bereiche beschrieben, auf denen die Konzepte und Methoden dieser Dissertation aufbauen. Sie sind dabei keine Alternativen zu den Ergebnissen dieser Dissertation, teilen sich mit ihnen jedoch einige der Ziele. Dennoch geht diese Dissertation mit ihren Zielsetzungen (vgl. Abschnitt 1.2.1) und Problemstellungen (vgl. Abschnitt 1.1.1) einen deutlichen Schritt weiter, besonders bezüglich Abdeckung, Automatisierung, Flexibilität, geteiltem Wissen, Optimierungsmöglichkeiten und gleichzeitiger Unterstützung der in den verwandten Arbeiten behandelten Analysearten.
Direkt nach den Ontologien werden Ansätze auf abstrakten Modellen, sowie proprietäre Lösungen vorgestellt. Die meisten existierenden Ansätze lassen die Untersuchung eines Modells für mehrere Analyseziele zu. Die in diesem Abschnitt vorgestellten Arbeiten befassen sich mit der Risiko-, Bedrohungs-, Schwachstellen-, Konfigurations- und Konformitätsanalyse, sowie der Angriffserkennung und -korrelation. Wie eingangs erwähnt, ist die Risikoanalyse, und somit der Forschungsschwerpunkt „Risikoquantifizierung“, jedoch nicht Bestandteil dieser Dissertation. Relevante Arbeiten der übrigen Bereiche werden, den proprietären Lösungen nachgestellt, erläutert.
In nahezu allen Bereichen werden auch Ansätze basierend auf Richtlinien verwendet. Dabei handelt es sich im Endeffekt um explizit modelliertes Sicherheitswissen, welches sich beispielsweise als formale Beschreibung von Konformitätsregeln oder Angriffsmustern eignet und in der Regel leicht in eine maschineninterpretierbare Form transformiert werden kann. Dieser Thematik wird daher ebenfalls ein Abschnitt gewidmet.
Auch existieren bereits Arbeiten, welche den Vorteil von modellbasierten Verfahren nutzen, die Modelle auch manuell manipulieren zu können, um zu testen, wie sich diese Änderungen auf die Analyseergebnisse auswirken. Solche Ansätze werden in Abschnitt 3.3.9 adressiert.

Daraufhin werden existierende Ansätze zu Gesamtlösungen für die Unterstützung verschiedener Analysearten diskutiert. Diese sind aufgrund ihres Ansatzes, eine Gesamtlösung zu bieten, direkt mit dieser Dissertation verwandt, weshalb in Abschnitt 3.3.11 vorwiegend Arbeiten aus dieser Sektion bezüglich ihrer Abdeckung der Anforderungen aus Abschnitt 3.2 verglichen werden.
Zuletzt werden die auf die Sicherheitsanalyse bezogenen Problemstellungen dieser Dissertation in Abschnitt 3.3.12 über Detailwissen aus verwandten Arbeiten konkretisiert und darauffolgend der erläuterte Stand von Wissenschaft und Technik zusammengefasst.

3.3.1 Sicherheitsontologien

Im Folgenden wird ein Überblick verschiedener, veröffentlichter Arbeiten zu Ontologien verschiedener Sicherheitsdomänen gegeben.

Blanco et al. [Bla11] untersuchten bereits 2011 die damals veröffentlichten Arbeiten zu Sicherheitsontologien und identifizierten sinngemäß die folgenden Zieldomänen: generische Sicherheitsontologien, Richtlinien-basiertes Netzwerkmanagement, Mobile Applikationen, Quality-of-Service-Bedingungen, Zugriffskontrolle, Voice-over-IP Dienste, Unterhaltungsrichtlinien, Audits, Netzwerksicherheit, Web-Dienste, IDS, soziale Aspekte, menschliche Sicherheitsprobleme, Sicherheitsrekonfiguration, Sicherheitsstandards, Schwachstellen und Ausfallsicherheit.
Für Veröffentlichungen von 2014-2015 führten Sicilia et al. [Sic15] eine ähnliche Untersuchung durch und identifizierten die folgenden Zieldomänen: Sicherheitsanforderungsanalyse (Sicherheitsstandards), Referenzontologien (Allgemeine Ontologien), Spezifikation und Abgleich (Zugriffskontrolle und Web-Dienste), Angriffsdetektion, Informationsextraktion und Vorbereitung für maschinelles Lernen.
Die Autoren schlussfolgern, dass die Vergleichbarkeit der Ontologien durch die verschiedenen Auslegungen von Sicherheitsbegriffen, wie Richtlinie oder Sicherheitsziel, schwierig ist. Zudem stellen die Autoren fest, dass die meisten Ontologien ohne eine komplette Instanziierung nutzlos sind. Dieses Problem

versuchen verschiedene Ansätze durch die Integration von speziellen Datenbanken zu lösen. So wurden beispielsweise Ontologien entworfen, welche Abbildungen der Konzepte mehrerer Sicherheitsinformationsstandards, wie *CPE*[1], *CVE*[2], *CAPEC*[3], *STIX*[4], bieten [Sye16, Kot18, Kie19], womit sich mehrere Quellen für bekannte Schwachstellen und Bedrohungen anbinden lassen. Solche Ontologien lassen sich besonders gut in der Schwachstellenanalyse einsetzen, wie die Verwendung von CVE-Repräsentationen aus dem Anwendungsszenario zeigt (vgl. Abschnitt 3.1.3).

Wie in Abschnitt 1.1.1 beschrieben, präsentierten de Franco Rosa und Jino 2017 eine Untersuchung zu Ontologien und Taxonomien mit dem Fokus auf Sicherheitsanalysen [Fra17b, Fra17a]. Ein Überblick über die Ergebnisse gibt Abbildung 3.6. Dort ist zu erkennen, dass die Wissensformalisierung sowie die Generalisierung der Informationssicherheit den Hauptfokus in der Forschung ausmacht. Ontologien zur Wissensrepräsentation von Sicherheitsanalysen gibt es den Autoren zufolge nicht. Stattdessen wurden außer den allgemeinen Informationssicherheitsontologien, die zuvor bereits erwähnten Einsatzbereiche adressiert. Aus dieser Untersuchung ging die *Security Assessment Ontology (SecAOnto)* hervor [Fra18], welche erstmals versucht, das Feld der Sicherheitsanalyse abzudecken und entsprechend einen wichtigen Schritt in Richtung des Ausnutzens von Synergien zwischen Analysearten geht. Ein Ausschnitt dieser Ontologie ist in Abbildung 3.7 zu sehen. Dieser enthält eine Klassenhierarchie und zeigt weitere, nicht näher beschriebene, Beziehungen zwischen den Klassen. Die Autoren argumentieren, dass eine weitere Detaillierung der Ontologie erforderlich ist. Dieses Vorhaben ist jedoch mit Vorsicht zu betrachten, da die Detaillierung von SecAOnto, die von den Autoren selbst als mittelschichtige Ontologie (engl. Intermediate Ontology) betitelt wird, nicht in

[1] Common Platform Enumeration (CPE), https://cpe.mitre.org/, zuletzt zugegriffen:16.01.2021.
[2] Common Vulnerabilities and Exposures (CVE), https://cve.mitre.org/, zuletzt zugegriffen:16.01.2021.
[3] Common Attack Pattern Enumeration and Classification (CAPEC), https://capec.mitre.org/, zuletzt zugegriffen:16.01.2021
[4] Structured Threat Information Expression (STIX), https://oasis-open.github.io/cti-documentation/stix/intro, zuletzt zugegriffen:16.01.2021

Spezialbereiche eindringen sollte, welche besser von austauschbaren, spezialisierten Ontologien adressiert werden sollten, um das entsprechende Wissen adäquat abzudecken.

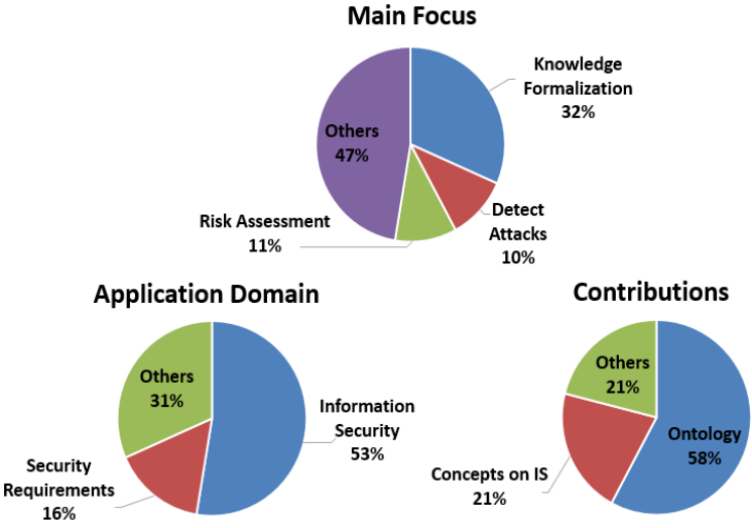


Abbildung 3.6: Zusammenfassende Darstellung existierender Sicherheitsanalyseontologien und -taxonomien. *Quelle:* [Fra18].

Es gibt Sicherheitsontologien, die eine besondere Popularität für den Einsatz bei Sicherheitsanalysen genießen oder genossen haben. Die allgemeinen Sicherheitsontologien von Herzog et al. [Her07], Fenz und Ekelhart [Fen09a], sowie von Kiesling et al. [Kie19], die eine der wenigen aktuell gehaltenen und online verfügbaren Sicherheitsontologien ist, sind hier besonders hervorzuheben.

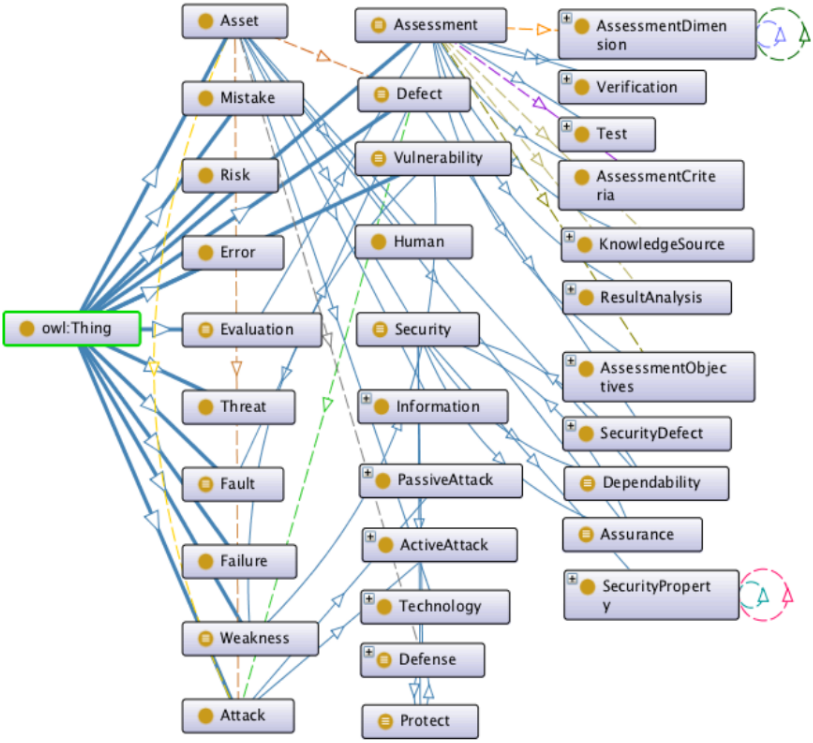


Abbildung 3.7: Ausschnitt der Security Assessment Ontologie (SecAOnto). *Quelle:* [Fra18].

Teil dieser Dissertation ist es weder eine ideale Sicherheitsontologie zu präsentieren, noch sich auf eine bestimmte Sicherheitsontologie festzulegen. Vielmehr haben die bereits existierenden Ontologien durchaus ihre Berechtigungen, da sie verschiedene Sichten auf Sicherheit und verschiedene Sicherheitsdomänen repräsentieren. Eine adäquate Sicherheitsanalyselösung mit dem Anspruch verschiedene Analysearten zu ermöglichen, sollte daher die Flexibilität bieten, für bestimmte Aufgaben, bestimmte Ontologien einsetzen und diese ggf. kombinieren zu können. Wie die weiteren Abschnitte zeigen, wird eine solche Flexibilität jedoch bisher von keiner Lösung geboten.

3.3.2 Analyse abstrakter Modelle

In diesem Abschnitt werden Analyselösungen vorgestellt, welche auf abstrakten Modellen arbeiten. Hierbei bezieht sich „abstrakt“ darauf, dass keine detaillierten Konfigurationen der Systeme modelliert werden, sondern lediglich Aussagen und Beobachtungen mit ähnlichem, meist aber sogar geringerem, Detailgrad. Ein Beispiel dafür sind Scan-basierte Ansätze, bei denen der Detailgrad von Konfigurationsinformationen in der Regel nicht über Netzwerkteilnehmer, Betriebssysteme, Adressen und Dienste hinaus geht.

Unter diese Kategorie fällt die *Cyber Security Modelling Language (CySeMoL)* [Som12b, Hol13, Som13, Hol14, Bus14, Mar14, Hol15b, Hol15a, Val15], ein Werkzeug mit eigener Modellierungssprache, welches dafür entwickelt wurde, Entscheidungsträger beim Nachvollziehen der Auswirkungen von Schwachstellen in einer Systemarchitektur auf andere Schwachstellen der Architektur zu unterstützen [Hol13]. Hierfür wurde ein auf *PRM (Probabilistic Relation Models)* basierender Ansatz gewählt, wobei ein PRM beschreibt, wie ein *Bayessches Netz* [Ben08] aus einem Modell erzeugt werden soll, welches ein Klassendiagramm abbildet (als Beispiel nennen die Autoren mit *Unified Modelling Language (UML)* [Obj17] erzeugte Klassendiagramme). Laut Sommestad et al. [Som13] ermöglicht dies das Berechnen von Wahrscheinlichkeiten von Objekteigenschaften in verschiedenen Systemarchitekturinstanzen. So werden in CySeMoL Wahrscheinlichkeiten von Erreichbarkeiten im Rahmen von Angriffspfaden abgeleitet.
Ein solcher Ansatz wurde von Holm et al. später in [Hol15b] als P^2CySeMoL vorgestellt. Das Besondere an P^2CySeMoL ist die enge Ausrichtung an echten Penetrationstests, also Kampagnen, bei denen Schwachstellen eines Systems gesucht und zu Testzwecken ausgenutzt werden, sowie eine zeitliche Abhängigkeit der Erfolgswahrscheinlichkeiten möglicher Angriffe. Der semantische Teil von CySeMoL und P^2CySeMoL enthält nur äußerst abstrakte Konzepte und wird nicht für die Darstellung von konkreten Systemkonfigurationen eingesetzt.
Der Grund für den hohen Abstraktionsgrad ist in der initialen Ausrichtung von CySeMoL zu finden, strukturelle Analysen basierend auf Enterprise Architecture [Win06] Modellen zu ermöglichen [Bus14]. Dies sind manuell erstellte

Modelle, die in der Regel beim Entwurf von u.a. IT-Architekturen eines Unternehmens entstehen. Hierzu gehören beispielsweise *SysML*[1]-Modelle [Mor11], für die es auch viele weitere Sicherheitsanalyseansätze gibt, die jedoch alle die Analyse von manuell erstellten Modellen im Rahmen der Planungs- und Entwurfsphase von Architektur- oder Softwareengineering adressieren [Ouc13, Lem14].
Wie bereits in Abschnitt 1.1.1 erwähnt, ist der Einsatz von Scans in industriellen Umgebungen kritisch. Dieser Fakt wurde ursprünglich auch von den Autoren selbst als Abgrenzung von verwandten Lösungen herangezogen und eigentlich für CySeMoL ausgeschlossen [Hol13]. In der Kommerzialisierung von CySeMoL taucht der Ansatz dennoch wieder auf (vgl. Abschnitt 3.3.3).

Der Vorteil von Analysen auf abstrakten Modellen ist die Möglichkeit des Arbeitens mit verhältnismäßig wenigen semantischen Konzepten und somit die Erstellung simpler Analysen. Daher ist diese Variante für beschränkte Informationsquellen, wie der manuellen Modellierung und der Anwendung statistischer Wahrscheinlichkeitsmodelle geeignet. Für Analysen auf abstrakter Ebene, beispielsweise das Ableiten möglicher Schwachstellen auf Basis des Betriebssystems, für die nur ein geeigneter Betriebssystem-Identifikator benötigt wird, sind abstrakte Modelle hinreichend.
Durch die meist manuelle Erstellung der Modelle skalieren diese Ansätze jedoch schlecht mit steigender Heterogenität und zunehmendem Detailgrad der Informationen. Im Gegenzug können abstrakte Modelle aus Quellen wie Enterprise Architecture durchaus sehr nützlich sein, da man mit ihnen Struktur- und organisatorische Informationen erhält, die automatisiert oft nicht zu ermitteln und ohnehin bereits für andere Verwendungszwecke manuell in maschinenlesbare Form gebracht wurden. Die Analyse abstrakter Modelle ist also in der Phase der Systemplanung, in welcher die reale Systemkonfiguration noch nicht feststeht, besonders hilfreich. Jedoch sind sie zu unspezifisch, um Aussagen über die reale Systemkonfiguration ableiten zu können. Hierfür bedarf es Ansätze, die detaillierte Modelle erzeugen und analysieren.

[1] SysML ist ein Profil von UML, siehe https://sysml.org, zuletzt zugegriffen: 06.11.2020.

3.3.3 Proprietäre Lösungen

Im Jahr 2005 stellten Shepard et al. mit CycSecure ein proprietäres Netzwerk-Risikobewertungs- und Analysewerkzeug vor [Bla05]. Später wurden weitere, ähnliche proprietäre Lösungen entwickelt. Dabei handelt es sich um die, zum Zeitpunkt des Verfassens dieser Dissertation, am weitesten entwickelte Lösung SecuriCAD [Eks15], die als Kommerzialisierung von CySeMoL entwickelt wurde. Sie ermöglicht dem Nutzer auf Basis von händisch über die Nutzeroberfläche zu erzeugenden Systemmodellen, mögliche, vorkonfigurierte Angreiferpfade und bekannte Schwachstellen zu identifizieren. Auch den Import von Systeminformationen über Wege wie, vom Komponentenhersteller zu programmierende, Agenten auf Basis einer lösungsspezifischen Import-API und die Informationsgewinnung aus Schwachstellen-Scannern ermöglicht die Anwendung (auch wenn dieser Einsatz von den Entwicklern selbst als kritisch betrachtet wird [Hol13]).
Generell lässt jedoch die Repräsentation sicherheitsrelevanter Informationen in eigenen, proprietären Modellierungssprachen, die alle proprietären Lösungen gemein haben, keine direkte Verwendung des internen Modells zu und unterstützt somit weder Wiederverwendbarkeit, noch Optimierung und Austauschbarkeit von Modellverarbeitungs- und Analyseschritten. Zudem ist eine Untersuchung der eingesetzten Modellverarbeitungs- und Analyseverfahren dadurch nicht möglich.

3.3.4 Schwachstellen- und Bedrohungsanalysen

Die meisten modellbasierten Lösungen zur Sicherheitsanalyse erlauben die Identifikation bekannter Schwachstellen im Systemmodell, solange diese die entsprechenden Details beinhalten [Ou05b, Ou05a, Wan10, Gam11, Gao13, Sim14, Kot18, Roh18]. Hierfür werden Datenbanken abgefragt, in denen Schwachstellenmeldungen, beispielsweise im CVE-Format, vorliegen. Durch die mit den Schwachstellenmeldungen verknüpften Angaben zu betroffenen Geräten oder betroffener Software, kann das Systemmodell nach diesen Schwachstellen durchsucht werden. Dazu eignet sich beispielsweise das CPE-Format.

Da Schwachstellenanalysen auch ein Teil der detaillierter Bedrohungsanalyse sein können, kombinieren viele der Ansätze das Wissen über Schwachstellen mit bekannten Bedrohungen, die entweder über Angriffsvektoren (bzw. Angriffsmustern) oder Angriffstechniken in die Modelle eingebracht und mit den Schwachstellen verbunden werden [Lae09, Wan10, Gam11, Gao13, Sim14, Kot18, Roh18]. Somit lassen sich aus einer gefundenen potenziellen Schwachstelle direkt Bedrohungen ableiten, welche die Schwachstellen ausnutzen können. Die Bedrohungen werden hierfür in der Regel händisch modelliert. Eine solche Lösung wird auch von dem in dieser Arbeit vorgestellten Gesamtkonzept unterstützt, wie Abschnitt 3.7.4 belegt.

3.3.5 Konformitätsanalyse

Den ersten und bedeutendsten ontologiebasierten Ansatz für Konformitätsanalyse stellten Fenz et al. vor. In [Fen09b, Fen09a] beschreiben Fenz und seine Koautoren Abbildungen des damaligen IT-Grundschutzkatalogs des deutschen Bundesamts für Sicherheit in der Informationstechnik (BSI) und der Sicherheitsmanagement-Standardreihe ISO 27000 auf eine generische Sicherheitsontologie. Hierbei wurden die jeweiligen Konzepte und Individuen für Gegenmaßnahmen, Assets, Bedrohungen und Schwachstellen verlinkt. Dieser sehr aufwändige, manuelle Prozess muss ebenfalls für jede Änderung durchgeführt werden, um das entstehende Modell aktuell zu halten. Im Kontrast zu diesem Ansatz, setzen die in dieser Dissertation beschriebenen Modellverarbeitungs- und Analysekonzepte auf austauschbare Sicherheitsdomänen (vgl. Abschnitt 3.4.1.3.2).
Fenz et al. stellten ihre Konformitätsanalyse in [Fen10] vor und beschrieben eine Konformitätsmetrik, welche der prozentualen Abdeckung von Gegenmaßnahmen entspricht, die dem ISO 27001 entnommen wurden. Dabei werden für jede Gegenmaßnahme alle (manuell modellierten) Assets selektiert, die von ihr geschützt werden sollten. Weiter wird geprüft, ob die (manuell modellierten) Anforderungen der Gegenmaßnahme mit den Assets verbunden sind. Fehlende Verbindungen verringern den Abdeckungsgrad einer Gegenmaßnahme. Die Autoren nennen die manuelle Erstellung des Systemmodells als den größten

Nachteil ihres Ansatzes und erwähnen dieses Problem weiter in [Fen15] mithilfe von Konfigurations-, Inventarisierungs- und Statusschnittstellen, gehen dort aber nicht auf entsprechende Implementierungen ein.
Bezüglich der Unterstützung verschiedener Sicherheitskontexte und Systemdomänen ist die Lösung von Fenz et al. unflexibel. Weder ist angedacht, den Sicherheitskontext (z.B. der eingesetzte Sicherheitsstandard) für verschiedene Analysen zu verwenden und auswechseln zu können, noch ist die Systemdomäne austauschbar. Modellerweiterungen sind ebenfalls nicht austauschbar und können nicht Analysen-spezifisch konfiguriert werden. Somit skaliert dieser monolithische Ansatz schlecht und ist weder für den Austausch von Erweiterungslogik und -modulen, noch für den Austausch von Analysen geeignet.
Wie in [Fen09b] beschrieben wurde, benötigen Standards und Best-Practices zusätzliche manuelle Interpretation der Konditionen, die ein Asset erfüllen muss, um zu einer Gegenmaßnahmenrichtlinie oder -definition konform zu sein. Im Gegensatz zu dem in dieser Dissertation beschriebenen Ansatz ermöglicht es der Ansatz von Fenz et al. nicht, solche Interpretationen durch beispielsweise eigene Richtlinien auszutauschen.

Einen auf AutomationML und OWL basierenden Ansatz zur Konformitätsanalyse stellten Eckhart et al. 2020 vor [Eck20]. Bei diesem Ansatz werden sicherheitsrelevante Informationen wie die Sicherheitszone eines Gerätes bereits (manuell) in das AutomationML-Modell eingebracht. Hierfür werden bestimmte Konzepte verwendet, welche die Autoren unter dem Begriff AMLsec zusammengefasst haben. Ein so erstelltes Modell wird dann in OWL transformiert. Leider geht aus [Eck20] nicht hervor, wie genau diese Transformation stattfindet. Die Autoren verweisen hierfür lediglich auf die Arbeiten von Hua und Hein [Hua18, Hua19], in denen mehrere Varianten vorgestellt wurden. Deren praktische Anwendbarkeit für AMLsec ist jedoch fraglich. Denn AMLsec definiert Konzepte, welche später in den Analysen verwendet werden sollen. Allerdings behalten die auf maschinellem Lernen basierenden Arbeiten von Hua und Hein diese Konzepte bei der Transformation gegebenenfalls nicht bei, was die entsprechenden Analysen verhindert. Zudem wird spezifisches, explizit zu modellierendes Hintergrundwissen verwendet, um Modellelemente

von einem Konzept in AutomationML zu einem Konzept in OWL zu transformieren. Möchte man also ein Konzept in AutomationML direkt auf ein Konzept in OWL transformieren, sind direkte Abbildungen hinzuzufügen, die den Einsatz des maschinellen Lernens obsolet machen.
Ist nun eine, wie auch immer geartete, Transformation von AutomationML nach OWL erfolgt, werden bei dem Ansatz von Ekhart et al. Konformitäts- oder Schwachstellenanalysen (evaluiert an den Beispielen IEC 62443 und *SEPSES Cybersecurity Knowledge Graph* [Kie19]) durch *SHACL-SPARQL*[1] Abfragen ausgeführt. Modellintegration und -erweiterung, austauschbare System- oder Sicherheitskonzepte, Synergieausnutzung zwischen Analysen oder die Unterstützung verschieden gearteter Module zur Optimierung, Abfrage, Interpretation oder zum Reasoning des Modells werden von dem Ansatz nicht adressiert.

3.3.6 Konfigurationsanalyse/SCM

Eine der relevantesten Lösungen für automatisierte Sicherheitsanalyse, die auch mehr als eine Analyseart unterstützt, ist das hauptsächlich von Ximming Ou entwickelte *Multihost, multistage Vulnerability Analysis (MulVAL)* Rahmenwerk [Ou05b, Ou05a]. MulVAL, dessen grundlegende Architektur in Abbildung 3.8 dargestellt wird, ist eine Lösung, die Informationen über Komponenten, durch auf diesen ausgeführte *OVAL*[2]-Scanner, sammelt. Hierbei steht OVAL für die *Open Vulnerability and Assessment Language* und stellt im wesentlichen ein Schema zur Auflistung von Konfigurationsdetails und eine Sprache zu deren Abfrage bereit. Zusätzlich werden von MulVAL Netzwerkinformationen aus Routern und Firewalls extrahiert. All diese Informationen werden als *Datalog* an eine Hauptanwendung gesendet. Datalog ist eine Teilmenge der logischen Programmiersprache *Prolog* [Ste99] und kann direkt in Prolog-Fakten umgewandelt werden [Zhu97]. Die Hauptanwendung erhält eine ebenfalls in Datalog geschriebene Liste von Regeln, welche Fakten zu

[1] https://www.w3.org/2014/data-shapes/wiki/Shacl-sparql, zuletzt zugegriffen: 27.10.2020.
[2] Open Vulnerability and Assessment Language, https://oval.cisecurity.org/, zuletzt zugegriffen: 23.08.2020.

verschiedenen Arten von Exploits, Schwachstellen (aus einer CVE-Datenbank) und Zugriffsrichtlinien definieren. Auf dieser Datenbasis wird Prolog verwendet, um aus den in Datalog beschriebenen Fakten Prädikatausdrücke abzuleiten, die auf entsprechende Probleme hinweisen.

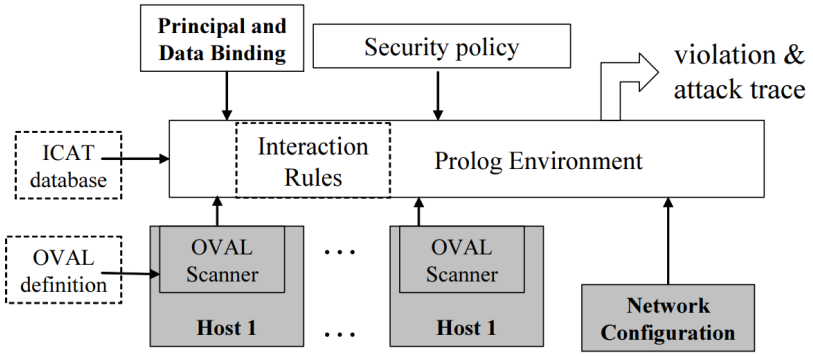


Abbildung 3.8: Überblick über das MulVAL Rahmenwerk. *Quelle:* [Ou05b].

MulVAL hat allerdings die folgenden Probleme: es ist nicht offen zugänglich, die Analysen sind nur angriffsorientiert, es unterstützt nur wenig explizite, wiederverwendbare Semantik und ermöglicht keine flexible Modellerweiterung oder Domänenrepräsentation.

Kong et al. stellten 2015 *Lightbulb* vor, ein Werkzeugset zur Konfigurationsanalyse im Rahmen von SCM [Kon15]. Dieses entspricht im Wesentlichen MulVAL, mit dem Unterschied, dass Lightbulb mit moderneren Programmiersprachen implementiert und zur Datenrepräsentation eine Erweiterung von Prolog, statt Datalog, verwendet wurde. Zudem unterstützt Lightbulb die Konfigurationsanalyse (insbesondere von Zugriffskontrollkonfigurationen) moderner

Datenbanklösungen wie *Apache Accumulo*[1] und anderer Technologien, wie *Security Enhanced Linux (SELinux)*[2] oder die Netzwerksoftware *Cisco IOS*[3].

Ein wesentlicher Aspekt beider Lösungen ist, dass für jede Verwendung und Verarbeitung der Informationen entweder die Konzepte der internen Repräsentation verwendet werden müssen oder eine programmatische Übersetzung zu und von anderen Konzepten zur internen Repräsentation notwendig ist. Zudem ist bei den beschriebenen Ansätzen eine Verwendung von Systemmodellen aus anderen Domänen, wie das eines Digitalen Zwillings, nicht vorgesehen und eine entsprechende Erweiterung wird durch die fehlende Option semantischer Abbildungen erschwert.

Neben MulVAL und Lightbulb beschreiben Fitzgerald und Foley einen ontologiebasierten Ansatz zur Identifikation von Konfigurationsproblemen in der Netzwerkzugriffskontrolle [Fit07, Fol08, Fit08]. Dabei stellten die Autoren in [Fit07] und [Fol08] eine OWL-Repräsentation von Netfilter-Firewall-Konfigurationen vor und erklärten, wie diese Darstellung verwendet wird, um Intra-Konfigurationseffekte mithilfe von SWRL und Reasoning zu finden. Später erweiterten die Autoren ihren Ansatz, indem sie Risiken und Schwellenwerte hinzufügten, um die Angemessenheit der NAC-Konfigurationen zu analysieren, was, wie die Autoren vorschlagen, für die Analyse von Inter-Konfigurationseffekten verwendet werden kann [Fit08]. Insbesondere soll dieser Ansatz laut der Autoren auch für die Analyse von Inter-Konfigurationskonformität verwendet werden können. Allerdings zeigen die Autoren nicht, wie dies skalierbar sein soll. Diese Frage stellt sich jedoch, da mit der Methode pro Richtlinie, zu der die Konformität geprüft werden soll, eine Regel pro möglicher NAC-Kombinationen modelliert werden muss. Eine mathematische Abschätzung des Problems ist schwierig, da sich die Anzahl nötiger Regeln reduziert, wenn mehrere NAC-Instanzen dieselbe Konfigurationssprache nutzen oder die Anzahl der in einer Richtlinie zu betrachtenden NAC-Instanzen vorgegeben werden kann. Gleichzeitig sind

[1] https://accumulo.apache.org/, zuletzt zugegriffen 07.11.2020.
[2] http://www.selinuxproject.org, zuletzt zugegriffen 07.11.2020.
[3] https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-software-releases-listing.html, zuletzt zugegriffen 07.11.2020.

diese Komplexitätsreduktionen jedoch bisher nicht untersucht. Ein Alternativansatz, der dieses Skalierungsproblem löst, wird in dieser Dissertation vorgestellt (vgl. Abschnitt 3.4.2.1).

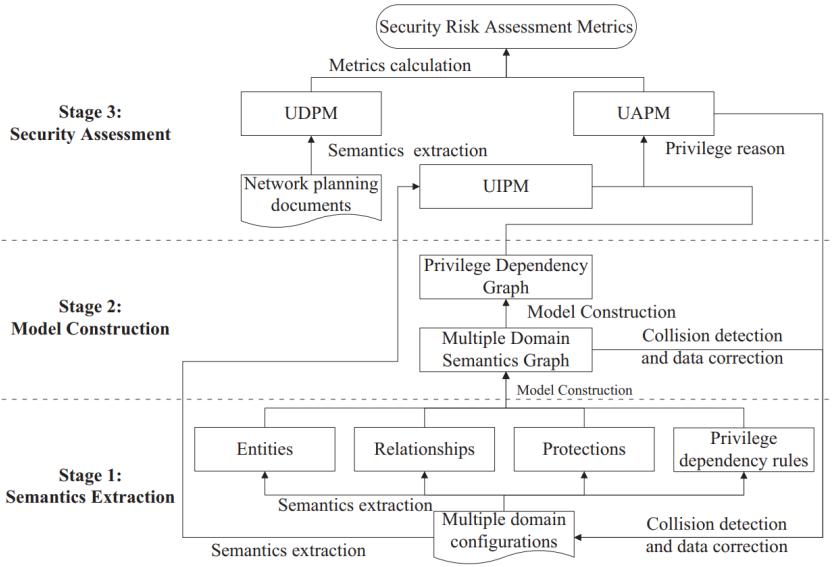


Abbildung 3.9: Architektur des MDC-Checker Rahmenwerks. *Quelle:* [Bai19].

Bai et al. stellten 2019 ein Rahmenwerk für die Nutzerberechtigung-zentrierte Analyse von NAC-Konfigurationen mit dem Namen *MDC-Checker* (*Multiple Domain Configurations Checker*) vor [Bai19]. Die Architektur dieses Rahmenwerks ist in Abbildung 3.9 zu sehen. MDC-Checker ist in die drei Phasen Semantikextraktion (Semantics Extraction), Modellkonstruktion (Model Construction) und Sicherheit-Assessment (Security Assessment) unterteilt. Diese zeigen einige wichtige Verarbeitungsschritte, welche für den speziellen Anwendungsfall von MDC-Checker notwendig sind. Besonders interessant ist die Betrachtung der Zusammenführung von Konfigurationen verschiedener

Domänen (im Multiple Domain Semantics Graph) und die entsprechende, automatisierte Konstruktion des zu analysierenden Modells (Priviledge Dependency Graph). Die Autoren zeigen dabei die Schwierigkeit auf, dass die einheitliche Repräsentation von NAC-Konfigurationen nicht mithilfe von Abstraktion zu bewältigen ist und sehen somit von der Repräsentation der Konfigurationsdetails ab.
Die in Abschnitt 3.4.2.1 beschriebene Alternativlösung kommt hingegen ohne diesen Informationsverlust aus. Auch zeigt MDC-Checker, wie schlecht eine monolithische Lösung skaliert, bei der alle Verarbeitungsschritte durchgeführt werden müssen, welche für die möglichen ausführbaren Analysen notwendig sind. Das in Abschnitt 3.5 präsentierte Rahmenwerk mindert auch dieses Skalierungsproblem.

3.3.7 Angriffserkennung und -korrelation

Die Angriffserkennung mit Hilfe von Modellen und Expertensystemen ist ein verhältnismäßig altes Forschungsfeld. Einer der ersten Ansätze wurde bereits 1989 von Bauer et al. [Bau89] zur Detektion von Eindringversuchen in UNIX-Systeme vorgestellt. Dieser wurde allerdings nur sehr abstrakt beschrieben und die technische Umsetzbarkeit wurde weitestgehend offengelassen.
Anfang des einundzwanzigsten Jahrhunderts wurden bereits Ansätze vorgestellt, die auf Ontologien setzten. Ein erstes ontologiebasiertes *Host-based Intrusion Detection System* (*HIDS*) wurden von Undercoffer et al. [Und03, Jef04] präsentiert. Undercoffer et al. konstruierten eine Ontologie für die Repräsentation von Angriffen mit der DARPA Agent Markup Language + Ontology Inference Layer (DAML+OIL[1]), einem Vorgänger von OWL, und stellten auch die zu untersuchenden Systemzustände mit DAML dar. Reasoning wurde daraufhin verwendet, um Anomalien, welche Aspekte der Angriffe aufweisen, entsprechend zu klassifizieren. Zusätzlich wurden mithilfe der in der Angriffsontologie beschriebenen Zusammenhänge, Schlussfolgerungen, wie die Klassifikation eines Syn-Flooding-Angriffs als Denial-of-Service-Angriff, abgeleitet. Weiter wurden auch Abfragen für mehrphasige Angriffe formuliert,

[1] https://www.w3.org/TR/daml+oil-reference/, zuletzt zugegriffen: 28.08.2020.

welche die Einzelangriffe korrelierten.
Dieser Ansatz wurde für verschiedene IDS weiterverfolgt, um die Vorteile von Ontologien, wie das Teilen von Wissen und die einfache Erweiterbarkeit, für die Angriffserkennung zu nutzen. So wurde der ontologiebasierte Ansatz von Undercoffer et al. (bzw. ähnliche Ansätze), für die Verbesserung von HIDS [He04], die Ermöglichung von IDS mit verteilten Sensoren oder verteilten IDS-Anwendungen [Man05a, Man05b, Ye08, Abd09, Bou19], die Verwendung in Mehr-Agenten-Systemen [Ye08, Abd09] und als semantische Grundlage für probabilistische [Ana05, Raz09] und andere lernende Ansätze [Col12] eingesetzt.

Ähnlich zu den Ansätzen für verteilte IDS-Anwendungen gibt es Arbeiten, die sich konkret mit Meldungskorrelation beschäftigen [Li10, Mor12, Fry12, Sad14, Si14, Nar18]. Li und Tian [Li10] stellten einen solchen Ansatz auf Basis einer Erweiterung von SWRL namens *XSWRL* vor, die nicht monoton ist und somit weder die Open World Assumption (OWA) noch Entscheidbarkeit unterstützt. Dabei werden Meldungen aus verschiedenen Quellen verarbeitet, indem Duplikate erkannt und entweder vereint oder gefiltert werden und die so entstehenden Meldungen als Evidenz für bestimmte Angriffe herangezogen werden. Die Angriffe werden wiederum von den XSWRL-Regeln abgeleitet. Da hierbei keine probabilistischen Methoden eingesetzt wurden, können nur Angriffe erkannt werden, für die alle in der jeweiligen XSWRL-Regel definierten Voraussetzungen erfüllt sind. Dasselbe gilt für die anderen aufgezählten Ansätze, bei denen jedoch andere Sprachen zur Individuenerzeugung eingesetzt wurden (bspw. SPARQL).
Weitere Ansätze, wie der von Al Balushi et al. [Al 16a, Al 16b], decken sowohl die Klassifikation des Netzwerkverkehrs, als auch die weitere Korrelation von gefundenen Angriffen ab.

Auch das Einbeziehen von zusätzlichen Informationen für die Angriffserkennung wurde bereits erforscht. ONTIDS [Sad14] ist ein ontologiebasiertes IDS-Rahmenwerk, welches laut den Autoren beliebige Quellen für Zusatzinformationen zulässt. Leider beschreibt die präsentierte Ontologie nur den Zusammenhang zwischen den Zusatzinformationen und weiteren Aspekten, wie Angriffen, Schwachstellen und Meldungen. Wie genau die, als Beispiel

genannten, Informationen von Konfigurationsmanagementsystemen in der Lösung integriert und insbesondere verwaltet werden sollen, wurde jedoch nicht beschrieben.
Etwas spezifischer sind hier die Ansätze von More et al. [Mor12] und Narayanan et al. [Nar18]. Diese extrahieren Informationen über Angriffe aus verschiedenen, in natürlicher Sprache vorliegenden Quellen unter der Verwendung von NLP.

Besonders interessant ist die Arbeit von Sarnovsky und Paralic [Sar20], in der beschrieben wird, dass Machine-Learning-Ansätze für die Klassifikation von generischen Angriffsklassen wie DoS eine hohe Genauigkeit aufweisen, jedoch bei der Klassifikation der konkreten Angriffe wie neptune[1] oder Smurf[2] schlechte Ergebnisse liefern. Um dem entgegenzuwirken, beschreiben die Autoren, wie sie ML-Verfahren für die Ermittlung der generischen Angriffsklassen einsetzen und eine Wissensbasis nutzen, um diese Angriffsklassifikation zu verfeinern.

Alle hier vorgestellten ontologiebasierten Verfahren lassen sich grundsätzlich mit dem in dieser Dissertation vorgestellten Rahmenwerk umsetzen (vgl. Abschnitt 3.7.6). Durch den Fokus der Ansätze auf IDS sind sie jedoch nicht für andere Aspekte der Sicherheitsanalyse geeignet und können so lediglich als Anwendungsfälle für SyMP betrachtet werden.

3.3.8 Richtlinien

Richtlinien-basiertes Netzwerkmanagement (engl. *Policy-Based Network Management* (*PBNM*)) dient nicht, dem Überprüfen ob Konfigurationen einer Richtlinie entsprechen, sondern der Generierung von Konfigurationen aus Richtlinien. Dieser Ansatz kann für bestimmte Sicherheitsmaßnahmen wie

[1] https://pen-testing.sans.org/resources/papers/gcih/neptunec-birth-syn-flood-attacks-102303, zuletzt zugegriffen: 04.09.2020.
[2] https://resources.sei.cmu.edu/asset_files/WhitePaper/1998_019_001_496180.pdf, zuletzt zugegriffen: 04.09.2020.

Firewalls sinnvoll sein, um eine formale, weniger fehleranfällige Nutzerschnittstelle bereitzustellen. Hierfür werden die Richtlinien u.a. in Ontologiesprachen formuliert, um sie für automatisierte Übersetzung maschineninterpretierbar zu repräsentieren. Beispiele für solche Ansätze sind [Xia06] und [Bas09].

López de Vergara et al. [Lóp09] stellten zudem einen Ansatz vor, mit dem Meldungen eines IDS in neue Sicherheitsregeln umgewandelt werden. Dieser Ansatz zeigt beispielhaft, welche Vorteile es haben kann, Konformitäts- und Konfigurationsanalyse mit Angriffserkennung zu verbinden. So kann mit diesem Ansatz beispielsweise ein Optimierungsprozess gebildet werden, durch den beobachtetes unerwünschtes Verhalten in Regeln überführt werden kann, welche wiederum befolgt werden können, um dieses Verhalten zu verhindern. Wird dann die Konfiguration angepasst, um die Regeln umzusetzen, kann sowohl eine Konformitätsanalyse als auch eine Konfigurationsanalyse auf dem Modell mit dieser Anpassung ausgeführt werden. Dadurch lässt sich die Wirksamkeit und Konsistenz der Umsetzung prüfen und ausschließen, dass diese Anpassung selbst Probleme und Widersprüche erzeugt hat. Dieser letzte Punkt liegt auch dem Konzept des fortlaufenden Security-by-Design zugrunde, welches im nächsten Abschnitt thematisiert wird.

3.3.9 Fortlaufendes Security-by-Design

Die gründliche Betrachtung von Sicherheitsaspekten eines Systems bei dessen Entwurf (engl. Security-by-Design) bekommt im Zusammenhang mit Digitalen Zwillingen eine neue Dimension. Denn Systeme müssen in der Regel mehrmals während ihrer Produktivzeit geändert werden. Auch bei der Planung dieser Änderungen, muss die Sicherheit des entstehenden angepassten Systems im Vorfeld analysiert werden [Nat11]. Boualem et al. adressierten dieses Thema für abstrakte Modelle von Systemen [Bou17] und entwickelten einen Ansatz, der Zusammenhänge zwischen generellen Auswirkungen von Wartungsmaßnahmen auf Bedrohungen, Gegenmaßnahmen und Schwachstellen eines Systems abbildet.
Digitale Zwillinge erlauben nun diese Änderungen in einem aktuell gehaltenen,

detaillierteren digitalen Abbild des Realsystems vorzunehmen und so Sicherheitsanalysen auf einer ebenso detaillierten Zukunftsversion des Realsystems durchzuführen. So können auch Probleme, wie Konfigurationskonflikte, identifiziert werden, bevor sie in der Realität entstehen. Dies ist trivial, da es nur bedeutet, dass bisher schon verwendete Ansätze auf aktuellere und detailliertere Modelle angewandt werden können und dies nicht nur initial, sondern bei jeder geplanten Änderung erfolgt. Ein solcher Ansatz wird auch prinzipiell in SyMP, durch die Möglichkeit der Nutzerinteraktion (vgl. Abschnitt 3.4.2.2) während der Modellerweiterung, unterstützt.

3.3.10 Rahmenwerke für verschiedene Analysearten

Diese Kategorie deckt Rahmenwerke ab, die für mehr als eine Analyseart gedacht sind. Hierzu gehören unter anderem auch MulVal und Lightbulb. Außerdem präsentierten Mozzaquatro et al. in [Moz18] ein Rahmenwerk für Sicherheits-Engineering und -Analyse von IoT-Systemen, welches im Forschungsprojekt C2NET[1] entwickelt wurde. Wie in Abbildung 3.10 zu sehen ist, befasst sich das Rahmenwerk mit fünf Phasen, die wiederum in Entwurfsphase (engl. Design Time) und Laufzeit (engl. Run Time) eingeteilt sind. Besonders interessant ist die Laufzeit, in der verschiedene Informationen aus Monitoring, Security-Scannern, IDS und Firewall-Konfigurationen extrahiert und über die Speicherung in relationalen Datenbanken, mithilfe der Ontop[2] Software in einen SPARQL-Endpunkt transformiert werden. Leider endet die Definition des Rahmenwerks mit dem Bereitstellen dieses Wissens (engl. Knowledge Provisioning) und die eigentlichen Analysen, sowie die nötigen Vorverarbeitungsschritte sind nicht mehr Teil davon, auch wenn zumindest die Verwendung von SPARQL und SWRL hierfür vorgesehen ist und in den Beispielimplementierungen eingesetzt wird.
Das in dieser Dissertation vorgestellte SyMP-Rahmenwerk setzt, abgesehen von den ebenfalls abgedeckten Informationsextraktions- und Modellbildungsschritten, dort an, wo die Definition des Rahmenwerks des C2NET-Projektes

[1] http://c2net-project.eu/, zuletzt zugegriffen: 07.11.2020.
[2] https://ontop-vkg.org/guide/, zuletzt zugegriffen: 07.11.2020.

aufhört. Die im C2NET-Rahmenwerk fehlenden Definitionen zur Modellverarbeitung, Analyse und Automatisierung werden somit in dieser Dissertation eingehend betrachtet.

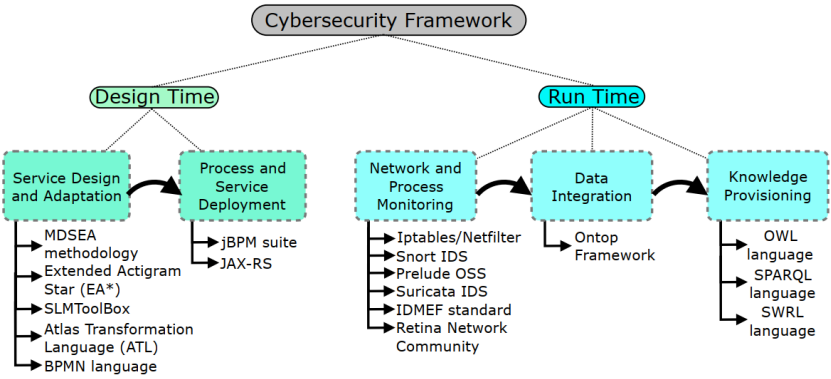


Abbildung 3.10: Implementierungssicht des IoT-Sicherheitsrahmenwerks aus dem EU-Projekt C2NET.
Quelle: [Moz18].

3.3.11 Anforderungsabdeckung in aktueller Forschung und Technik

Die Abdeckung der Anforderungen aus Abschnitt 3.2 durch die wichtigsten verwandten Arbeiten, wird in diesem Abschnitt bewertet. Tabelle 3.1 bietet einen Überblick, über die Anforderungskürzel, da diese in der Abdeckung verwendet werden.

Tabelle 3.1: Anforderungskürzel und deren Bedeutungen.

Kürzel	Anforderung
UA1	Minimalinvasiv
UA2	Ressourcenschonend
UA3	Quellheterogenität

weiter auf der nächsten Seite

Kürzel	Anforderung
UA4	Abdeckung industrieller Assets
IA1	Sichere Informationsextraktion
IA2	Reduktion der Abbildungskomplexität
IA3	Unterstützung von Standards und konsolidierten Spezifikationen zu Informationsmodellen
IA4	Unterstützung verschiedener Serialisierungen
IA5	Unterstützung von Zielmodellen
IA6	Einfache Anpassung der Quellen
MA1	Einsatz von Ontologien zur Modellierung
MA2	Einsatz monotoner Beschreibungslogiken zur Modellierung und Modellerweiterung
MA3	Unterstützung von nicht-DL Formalismen und Algorithmen
MA4	Klare Trennung zwischen der Verwendung von Beschreibungslogiken und anderer Formalismen
MA5	Separation-of-Concerns
MA6	Austauschbare Erweiterungslogik
MA7	Analysespezifische Erweiterung
MA8	Abhängige Verarbeitungsschritte
MA9	Unterstützung von Nutzerinteraktion bei Modellbildung und -verarbeitung
AA1	Unterstützung der behandelten Analysearten
AA2	Technische Evidenz
AA3	Flexible Richtlinien und Regeln
AA4	Austauschbare/Wiederverwendbare Analysen
AA5	Anpassungsfähigkeit
AA6	Sicherheitsdomänen
GA1	Automatisierung
GA2	Skalierbarkeit
GA3	Nutzbarkeit
GA4	Offenheit
GA5	Lebenszyklusunterstützung

Erfüllt eine Lösung eine Anforderung, wird dies mit ● angezeigt. Wird eine Anforderung nicht erfüllt, steht in dem entsprechenden Feld ein ○. Nun gibt es Anforderungen, die für den jeweiligen Kontext nicht zufriedenstellend erfüllt, jedoch auch nicht gänzlich unerfüllt sind. Diese werden mit einem ◐ markiert. Als Beispiel hierfür kann Tabelle 3.2 herangezogen werden, deren Anforderungen mit Bezug auf Informationsextraktion von manuell erzeugten Systemmodellen erfüllt werden. Denn manuell erzeugte Modelle erreichen, unter der Annahme unbeschränkter Ressourcen, theoretisch 100% Abdeckung. Da aber der hier wesentliche Aspekt der Reduktion manueller Aufwände von diesen Lösungen nicht unterstützt wird, wird ihnen hier kein gänzliches Erfüllen der Anforderungen zugestanden.

Wie die vorangegangenen Abschnitte zeigen, wurden in der Vergangenheit keine Gesamtlösungen für Sicherheitsanalysen vorgestellt, wie sie hier in dieser Arbeit angestrebt wird. Die Mehrheit verwandter Arbeiten geht nur auf Einzelaspekte von bestimmten Sicherheitsanalysen ein und bietet keine Verwendung für weitere Analysearten, keine Werkzeuge, keine Flexibilität für Modellierungsaspekte und mehr. In den folgenden Tabellen werden daher nur die Lösungen betrachtet, die einen adäquaten Teil der Anforderungen abdecken und somit eine Vergleichbarkeit mit der in dieser Arbeit vorgestellten Lösung ermöglichen. Die einzelnen Anforderungsergebnisse werden zudem kurz kommentiert, um die entsprechende Intension hinter der jeweiligen Bewertung hervorzuheben. Dabei erhält jeder Kommentar eine Kennzeichnung, die aussagt, auf welche Anforderung er sich bezieht.

Tabelle 3.2 zeigt die Abdeckung der umgebungsbezogenen Anforderungen. Mit Ausnahme des Ansatzes von Fenz et al. definieren alle Ansätze minimalinvasive Möglichkeiten zur Informationsextraktion und greifen danach nicht weiter in das System ein [UA1].
Außerdem definieren CySeMoL und C2Net passive Extraktionsansätze ohne Agenten und sind damit zwar ressourcenschonend, bieten diese Ansätze jedoch nur als Quellen für Modelle und Überwachungssysteme [UA2].

Tabelle 3.2: Abdeckung umgebungsbezogener Anforderungen durch den Stand von Forschung und Technik.

	UA1	UA2	UA3	UA4
CySeMoL[a]	●	◐	◐	◐
SecuriCAD[b]	●	○	●	◐
Fenz et al.[c]	◐	○	○	○
MulVAL[d]	●	○	●	○
Lightbulb[e]	●	○	●	○
C2NET[f]	●	◐	●	◐

[a] Analyse abstrakter Modelle aus Abschnitt 3.3.2.
[b] Proprietärer Ansatz aus Abschnitt 3.3.3.
[c] Der ontologiebasierte Ansatz zur ISO 27000-Konformitätsanalyse aus Abschnitt 3.3.5.
[d] Prolog-basierter Ansatz aus Abschnitt 3.3.6.
[e] Moderner, MulVAL-ähnlicher Ansatz aus Abschnitt 3.3.6.
[f] Das IoT-Sicherheitsrahmenwerk aus Abschnitt 3.3.10.

Generell definieren jedoch nur Fenz et al. keine Lösungen für verschiedene Quelltypen [UA3].
UA4 wird hierbei von keiner Lösung zufriedenstellend erfüllt [UA4]. Die Lösungen mit ◐ haben sich mit der Abdeckung industrieller Assets nicht befasst, wohingegen die Lösungen mit ○ auf manuelle Modellierung, eigene APIs, Enterprise Architecture oder Überwachungssysteme setzen. Damit bleibt die Mehrheit von Geräten und Informationen verborgen [UA4].

Tabelle 3.3 zeigt die Abdeckung von Anforderungen an Informationsextraktion und -repräsentation. Mit Ausnahme von SecuriCAD wurde bei keiner der Lösungen die sichere Übertragung der relevanten Informationen beschrieben [IA1]. Gleichzeitig ist die sichere Übertragung der Informationen von den verwendeten Protokollen (und deren Infrastruktur) abhängig und es wurde bei keiner der Lösungen ein triftiger Grund gefunden, weshalb solche sicheren Protokolle nicht eingesetzt werden könnten [IA1].
SecuriCAD und Lightbulb gehen davon aus, dass Modelle im eigenen Schema

vorliegen, MulVal erwartet das OVAL-Schema und Fenz et al. gehen auf die Umwandlung von Konfigurationen in die eigene TBox nicht ein. Somit wird IA2 von diesen Lösungen nicht unterstützt [IA2]. CySeMoL und C2NET übernehmen zumindest selbst die Umwandlung und unterstützen existierende Informationsmodelle [IA2].

Tabelle 3.3: Abdeckung von Anforderungen an Informationsextraktion und -repräsentation durch den Stand von Forschung und Technik.

	IA1	IA2	IA3	IA4	IA5	IA6
CySeMoL	◐	●	●	◐	●	●
SecuriCAD	●	○	○	○	●	●
Fenz et al.	◐	○	○	○	●	○
MulVAL	◐	○	○	○	●	●
Lightbulb	◐	○	○	○	●	●
C2NET	◐	◐	◐	◐	○	●

SecuriCAD, Fenz et al. und Lightbulb unterstützen keine Quellstandards [IA3]. MulVal unterstützt OVAL, was nur als Zielschema und nicht als Quellstandard angesehen werden kann [IA3]. C2NET generiert auch aus Schemata, die Standards folgen, Wissensrepräsentationen, übernimmt aber direkte Abbildungen (weshalb es auch IA2 nicht gänzlich erfüllt) [IA3]. CySeMoL unterstützt UML-basierte Quellstandards und übersetzt sie in die eigene interne Repräsentation [IA3].
IA4 kann nur von Lösungen erfüllt werden, die IA3 erfüllen und sowohl bei CySeMoL als auch C2NET konnten keine signifikanten Hinweise zur Abdeckung oder Nicht-Abdeckung verschiedener Serialisierungen gefunden werden [IA4].
Bis auf C2NET definieren alle Lösungen ein Zielmodell [IA5].
Von den entsprechenden Veröffentlichungen kann abgeleitet werden, dass alle

Lösungen mit Quelländerungen umgehen können sollten [IA6]. Eine Ausnahme stellt der Ansatz von Fenz et al. dar, da es zur Informationsextraktion keine konkreten Konzepte bietet [IA6].

Tabelle 3.4: Abdeckung von Anforderungen an die Modellbildung und -verarbeitung durch den Stand von Forschung und Technik.

	MA1	MA2	MA3	MA4	MA5	MA6	MA7	MA8	MA9
CySeMoL	○	○	●	○	●	○	○	○	●
SecuriCAD	○	○	●	○	○	○	○	○	●
Fenz et al.	●	●	○	○	○	◐	○	○	○
MulVAL	○	●	○	○	○	◐	○	○	○
Lightbulb	○	●	○	○	○	◐	○	○	○
C2NET	●	●	◐	○	●	◐	◐	○	◐

Tabelle 3.4 zeigt die Abdeckung von Anforderungen an Modellbildung und -verarbeitung. Ontologien werden nur von Fenz et al. und C2NET eingesetzt [MA1].
Die gleichzeitige Unterstützung von DL- und Nicht-DL-Algorithmen wird nur von C2NET, wenn auch nicht detailliert, definiert [MA2, MA3].
Der koordinierte Einsatz von Verarbeitungsketten, die sowohl DL- als auch Nicht-DL-Algorithmen und -Definitionen zulassen, wird von keinem der Ansätze geboten [MA4]. Nur CySeMoL und das Rahmenwerk des C2NET-Projektes berücksichtigen mehrere Akteure [MA5].
Wirklich vorstellbar ist eine Unterstützung wiederverwendbarer und austauschbarer beliebiger Modellerweiterungs-Analyse-Kombinationen lediglich bei C2NET, da die anderen Anwendungen weniger flexible Konzepte verfolgen. Allerdings ist dies bei C2NET auch nur deswegen vorstellbar, weil C2NET kein Alternativkonzept definiert und somit theoretisch im Sinne von MA6 und MA7 erweitert werden könnte [MA6, MA7]. Abhängigkeiten zwischen Verarbeitungsschritten werden von keiner Lösung unterstützt [MA8].

Durch die manuelle Modellierung bei SecuriCAD und CySeMoL wird von diesen die entsprechende Nutzerinteraktion unterstützt [MA9]. Auch C2NET unterstützt zumindest bei der Erstellung der Geschäftsprozesse Nutzerinteraktion [MA9].

Tabelle 3.5: Abdeckung von analysebezogenen Anforderungen durch den Stand von Forschung und Technik.

	AA1	AA2	AA3	AA4	AA5	AA6
CySeMoL	○	◐	◐	◐	○	○
SecuriCAD	○	○	○	○	○	○
Fenz et al.	○	◐	●	●	○	○
MulVAL	○	●	●	●	◐	○
Lightbulb	○	●	●	●	◐	○
C2NET	◐	◐	●	◐	◐	◐

Tabelle 3.5 zeigt die Abdeckung von analysebezogenen Anforderungen. CySeMoL ist Angriffsvektor-orientiert und kann somit ohne signifikanten Mehraufwand keine der Analyseanforderungen für Analysen voll erfüllen [AA1-AA6]. SecuriCAD ist ebenso Angriffsvektor-orientiert, ist aber zudem auch noch proprietär und nicht beliebig erweiterbar. Daher kann es keine der Analyseanforderungen erfüllen [AA1-AA6].
Obwohl C2NET zumindest eine Unterstützung alle hier adressierten Analysearten anzustreben scheint, liefert es hierfür keine konkreten Konzepte [AA1]. Die anderen Lösungen, wurden nicht für die Unterstützung aller Analysearten entworfen und liefern daher ebenfalls keine entsprechenden Konzepte [AA1]. Das nötige Detailniveau für die technische Evidenz erfüllen nur MulVAL und Lightbulb in Gänze [AA2].
Außer CySeMoL und SecuriCAD sind die Lösungen erweiterbar, unterstützen Regeln und Richtlinien und setzen zudem auf Logikprogrammierung, was die Erfüllung von Wiederverwendbarkeitsanforderungen ermöglicht [AA3-AA5].

Unterschiedliche Sicherheitsdomänen werden von keiner der Lösung unterstützt [AA6]. Nur C2NET enthält keine Definitionen, welche die Erweiterung für diese Unterstützung erschweren [AA6].

Tabelle 3.6: Abdeckung von generellen Anforderungen durch den Stand von Forschung und Technik.

	GA1	GA2	GA3	GA4	GA5
CySeMoL	◐	◐	◐	◐	○
SecuriCAD	◐	●	◐	◐	○
Fenz et al.	◐	◐	◐	○	○
MulVAL	●	◐	◐	○	○
Lightbulb	●	◐	◐	○	○
C2NET	◐	◐	◐	○	●

Tabelle 3.6 zeigt die Abdeckung von generellen Anforderungen. Die Ansätze, welche nicht alle Schritte von der Informationsextraktion bis zur Analyse abdecken, können GA1 nicht in Gänze erfüllen [GA1].
Die Skalierbarkeit der Forschungsansätze ist nicht ausgeschlossen, durch fehlende Studien aber auch nicht konkret einschätzbar [GA2]. SecuriCAD ist zumindest durch den skalierbaren Cloud-Ansatz im Bereich der Angriffvektorbasierten Analyse skalierbar [GA2].
Keiner der Ansätze setzt auf durchgängiges Separation-of-Concerns und definiert gleichzeitig entsprechende Schnittstellen [GA3].
Keine der Lösungen ist zudem quelloffen, jedoch sind CySeMoL und SecuriCAD zumindest herunterlad- und ausführbar [GA4].
Nur C2NET adressiert den gesamten Sicherheitslebenszyklus [GA5].

3.3.12 Konkretisierung der Problemstellung

Dieser Abschnitt konkretisiert die, in Abschnitt 1.1.1 beschriebenen, Problembereiche und setzt sie in Beziehung zum beschriebenen Stand von Wissenschaft und Technik.

Automatisierung von Prozessen

Der Aufwand zur Informationsextraktion und Modellierung ist, wie zuvor angesprochen, ein entscheidender Nachteil modellbasierter, gegenüber nichtmodellbasierter Ansätze zur Sicherheitsanalyse. Gerade auf Basis von Inventar- oder Konfigurationsdaten ist dieser Aufwand aufgrund der Vielzahl und Komplexität der Informationen besonders hoch. Fenz und Neubauer schätzten in [Fen18], dass für Risikoanalysen durch das Sammeln und Modellieren von Inventarinformationen ein Zusatzaufwand von 30% gegenüber konventionellen, manuellen, nicht-modellbasierten Ansätzen entsteht. Der Stand von Wissenschaft und Technik weist, ausgenommen für die Angriffserkennung und -korrelation, klare Forschungsbedarfe zur Automatisierung der immer noch überwiegend manuellen Modellierung und zur flexiblen Modellerweiterung auf.
Die meisten vorgestellten Ansätze erfordern das Ausführen aller Regeln und Programmbausteine zum Ableiten weiteren Wissens (vgl. Abschnitt 2.7.2.1) bei Erzeugung und Aktualisierung des Systemmodells, egal ob dieses Wissen daraufhin genutzt wird oder nicht. Auch lagern alle vorgestellten Ansätze mit flexiblen Analysen die meisten nötigen Modellerweiterungs- und Analyseschritte an die Analyseanwendung aus und verhindern so deren Wiederverwendung. Beides schränkt die Skalierbarkeit dieser Ansätze unnötig ein, denn die Anzahl dieser Erweiterungen wird in der Realität schnell drei-, vier- oder sogar fünfstellig, ist somit schwer zu verwalten und der Ressourcen- und Zeitaufwand zur Anwendung der Regeln übersteigt schnell jegliche akzeptable Größe. Wie aufwendig selbst eine einzige, simple Regel sein kann wird in Abschnitt 3.7.2 noch einmal verdeutlicht.

Wiederverwendung von Wissen

Der aktuelle Stand von Wissenschaft und Technik enthält keine automatisierte Lösung, die eine Wiederverwendbarkeit verschiedenen Wissens umfassend unterstützt. So werden beispielsweise Modellerweiterungsstrategien eingesetzt, welche festen Annahmen bezüglich der Konfiguration von Komponenten unterliegen, die immer dann zum Einsatz kommen, wenn nicht explizit verfügbares Wissen abgeleitet werden soll. Diese Annahmen können jedoch von Umgebung zu Umgebung unterschiedlich sein, was die fehlerfreie Anwendung und Wiederverwendbarkeit der Modellerweiterungen und Analysen verhindert. Beispielsweise kann im Rahmen einer Netzwerkstrategie definiert werden, ob zwei IP-Netzwerke mit identischer Netzwerkadresse vorkommen dürfen oder nicht. Je nach Definition kann so von der Netzwerkadresse abgeleitet werden, ob sich zwei Geräte im selben Netzwerk befinden oder nicht. Im Rahmen von automatisierten Erweiterungen von Systemmodellen gibt es beliebig viele solcher Annahmen. So kann die Annahme getroffen werden, dass Dienst-Port-Abbildungen den durch die IANA[1] standardisierten Abbildungen entsprechen oder dass sich IP-Adressbereiche von statischen und dynamischen IP-Adressen nicht überschneiden (weil beide über DHCP vergeben werden). Können solche Annahmen nicht angepasst werden, sind sie entweder nicht verwendbar oder disqualifizieren Analysen, welche die dadurch abgeleiteten Informationen benötigen, für jede Umgebung in der die Annahmen abweichen. Existierende Ansätze unterstützen zudem entweder nur statische Verarbeitungsketten oder überlassen komplexere Berechnungen den Anwendungen (hier Analysen). Im ersten Fall, wird die Zahl möglicher Analysen stark beschränkt. Im zweiten Fall, wird die Wiederverwendbarkeit der Verarbeitungsschritte nicht gewährleistet.
Wie zuvor beschrieben, erreicht keiner der Ansätze eine gleichzeitige Unterstützung der Konzepte verschiedener Sicherheitsstandards, Systemdomänen und Modellerweiterungsstrategien sowie der Wiederverwendbarkeit und des

[1] Internet Assigned Numbers Authority (IANA), https://www.iana.org, zuletzt zugegriffen: 25.10.2020.

lösungsübergreifenden Austausches von Modellerweiterungs- und Analyseschritten.
Das bereits in Abschnitt 1.1.1 angeschnittene Problem der fehlenden Expertisetrennung ist bei den hier beschriebenen verwandten Ansätzen gut zu erkennen. Denn entweder handelt es sich um eine sehr offene, flexible (meist monolithische) Lösung, die gleichzeitig aber verlangt, dass ein Nutzer sowohl Modellierungs- also auch Systemdomänen- und Sicherheitsexperte ist, oder um eine geschlossene Lösung, die lediglich vorkonfigurierte Analysen oder stark beschränkte Analysesprachen bietet, der es wiederum an der nötigen semantischen und programmatischen Flexibilität fehlt.
Wie die Beschreibung des Stands von Wissenschaft und Technik zeigt, wurde im Bereich der Konformitätsanalysen, nach bestem Wissen des Autors, bisher keine Lösung präsentiert, welche die semantischen Konzepte verschiedener Sicherheitssichten (weiterhin auch als *Sicherheitsdomänen* bezeichnet), wie verschiedene Sicherheitsstandards und Best-Practices, unterstützt und eine Unabhängigkeit von den Konzepten des zu analysierenden Systemmodells sicherstellt. Für die Wiederverwendung und Austauschbarkeit von Analysen ist dies jedoch eine grundlegende Voraussetzung.

Nicht-Gefährdung der zu analysierenden Systeme

Wie bereits beschrieben ist das, oftmals auch in den zuvor beschriebenen Arbeiten eingesetzte, Scannen von Netzwerkteilnehmern eine potenzielle Gefahr für das entsprechende Gerät. Während des Scannens werden verschiedenste Netzwerkpakete erzeugt und an die Netzwerkteilnehmer gesendet. Die darauffolgenden Antworten der Teilnehmer nutzt der Sender dann als Informationsgewinn und kann so beispielsweise herausfinden, welche IP-Adressen genutzt werden, auf welchen Ports die Teilnehmer lauschen, welche Dienste sie anbieten oder welche Betriebssysteme auf den Teilnehmern laufen. Die dadurch gewonnenen Informationen können bereits für einige Netzwerk-fokussierte Analysen ausreichen. Allerdings tendieren industrielle Komponenten bei solchen Scans zur Fehlfunktion, bis hin zum Totalausfall [Pfr17, Pfr18]. Dies gilt selbst dann, wenn als vermeintlich sicher geltende

Verfahren wie *Ping Sweeps*[1] eingesetzt werden, weshalb diese Methode auf Produktivsystemen nicht angewendet werden sollte [Dug05].

Erweiterung der Analyseabdeckung

Ein wichtiger Bestandteil der Analyseabdeckung ist die Abdeckung möglichst vieler für die Analysen relevanter Komponenten. Gleichzeitig sollen die Komponenteninformationen, wie zuvor beschrieben, automatisiert aber minimalinvasiv gewonnen werden. Als automatisierte, minimalinvasive Alternativen kommen das Auslesen von Konfigurationsdatenbanken, Konfigurationsschnittstellen und Statusschnittstellen, sowie die Geräte-Selbstauskunft und das passive Überwachen des Netzwerkverkehrs in Betracht.
Ersteres bedient sich der zahlreichen Schnittstellen, die in der Büro-IT zur Verfügung stehen. Einen guten Überblick bietet [Fen15], wobei hier darauf hingewiesen werden soll, dass seit der Entstehung des Artikels weitere Möglichkeiten, wie NETCONF[2] oder Lösungen auf Basis des Common Information Models[3] (CMI), wie das von Microsoft stammende WMIC[4] Werkzeug, entwickelt und verbreitet wurden. Das Auslesen von Konfigurations- und Statusschnittstellen birgt jedoch Schwierigkeiten, welche sie – zumindest als Einzellösung – ausschließen. Solche Schnittstellen sind schon in der Büro-IT weitgehend heterogen, in industriellen Systemen sind diese jedoch noch weit diverser, da die Vielfalt von Firmwares und Applikationen im industriellen Bereich besonders groß ist. In der Regel besitzen industrielle Komponenten nur SSH- oder HTTP-Schnittstellen, deren Informationsrepräsentation keinem Standard oder spezifizierten Schema folgen. Erschwerend kommt hinzu, dass diese Konfigurations- und Statusschnittstellen oft nur wenige sicherheitsrelevante Informationen preisgeben.

[1] Mit dem Ping-Befehl durchgeführte Ermittlung erreichbarer Endgeräte.
[2] https://tools.ietf.org/html/rfc6241, zuletzt zugegriffen: 10.09.2020.
[3] https://www.dmtf.org/standards/cim/, zuletzt zugegriffen: 10.09.2020.
[4] https://docs.microsoft.com/de-de/windows-server/administration/windows-commands/wmic, zuletzt zugegriffen: 10.09.2020.

Konfigurationsdatenbanken hingegen, besitzen meist sowohl einheitliche Schemata als auch viele sicherheitsrelevante Informationen. Für industrielle Komponenten werden sie jedoch nicht verwendet und können somit nur zum Auslesen von Informationen zu Workstations oder ähnlich zu klassifizierenden Endgeräten verwendet werden. Dass diese Datenbanken nicht für industrielle Komponenten verwendet werden hat ähnliche Gründe wie die problematische Verwendbarkeit der Konfigurations- und Statusschnittstellen als Informationsquellen. Zur Versorgung der Konfigurationsdatenbanken mit Konfigurationsdaten werden Selbstauskunft-Agenten wie WMI[1] eingesetzt. Hierdurch können einzelne Programme auf den Komponenten von Interesse installiert werden, welche deren relevante Informationen (Konfigurationen und Status) auslesen und an bestimmte Senken, wie eine Konfigurationsdatenbank oder Analyseplattform, senden. Industrielle Komponenten sind allerdings in der Regel ressourcenschwach und bieten durch hoch spezialisierte Soft- und Hardware wenig Spielraum für Erweiterungen [Nat11]. Somit werden aktuell auf solchen Komponenten keine Selbstauskunft-Programme eingesetzt. Die Entwicklung und Verwaltung von Selbstauskunft-Programmen für industrielle Komponenten skaliert durch deren hohen Individualisierungsgrad der Firmware zudem sehr schlecht. Die Adaption solcher Lösungen wird zusätzlich eingeschränkt, wenn diese Programme nur für Sicherheitszwecke entwickelt werden und dadurch keinen direkten ökonomischen Mehrwert für Komponentenhersteller, Integratoren und Betreiber bieten.
Zuletzt gibt es die Möglichkeit, Informationen mittels Überwachung des Netzwerkverkehrs zu sammeln. Diese sichere Variante der Informationsextraktion ist auch im industriellen Umfeld sehr verbreitet, beschränkt sich jedoch auf Netzwerkkommunikation und reduziert somit, bei alleinigem Einsatz, die Analysemöglichkeiten.

[1] Windows Management Instrumentation (WMI) https://docs.microsoft.com/en-us/sql/relational-databases/wmi-provider-configuration/working-with-the-wmi-provider-for-configuration-management, zuletzt zugegriffen: 18.10.2020.

3.3.13 Zusammenfassung

Der Stand von Wissenschaft und Technik weist, ausgenommen für die Angriffserkennung und -korrelation, klare Forschungsbedarfe zur Automatisierung der immer noch überwiegend manuellen Modellierung und zur flexiblen Modellerweiterung auf. Alle Lösungen scheitern entweder an der Betrachtung verschiedener Analysearten und somit der Ausnutzung ihrer Synergien oder vernachlässigen die Informationsextraktion für industrielle Systeme und somit die Systemabdeckung der Analysen. Dieser Aspekt führt dazu, dass die Analysen auf Modellen arbeiten, welche stark vom realen System abstrahieren und dadurch ungenau und eingeschränkt aussagekräftig sind.
Trotz des Einsatzes von Ontologien und Beschreibungslogiken decken verfügbare Ansätze nicht den Austausch von relevantem Wissen und Algorithmen zwischen mehreren Parteien ab. Somit verhindert der hohe individuelle Aufwand, sowohl eine Analyselösung für ein bestimmtes Unternehmen anzupassen, als auch eine hohe Abdeckung für die folgenden Aspekte zu erreichen: System- bzw. Konfigurationsinformationen, Modellerweiterungen, Standards, Best-Practices und Analysen. Dies liegt hauptsächlich an fehlenden klaren Konzepten und Strategien zur Informationsextraktion, Modellbildung, Modellerweiterung und Analyse, die ein für die Modularisierung und Austauschbarkeit von Wissen notwendiges Separation-of-Concerns umsetzen.

Des Weiteren gibt es im Bereich der ontologiebasierten Sicherheitsanalyse Forschungsbedarf bezüglich der Best-Practices zum Aufbau und zur Durchführung solcher Analysen. Dies ist besonders darauf zurückzuführen, dass es bisher keine adaptiven Konzepte und Methoden gibt, die eine Umgebung bilden, in der sich diese Best-Practices entwickeln und untersuchen lassen.

3.4 Lösungsansätze für einzelne Phasen

Im Rahmen dieser Dissertation wurden Einzelkonzepte und -methoden entwickelt, welche einige der Hypothesen aus Abschnitt 1.1.1, der Ziele aus Abschnitt 1.2.1 und der Anforderungen aus Abschnitt 3.2 adressieren. Sie bieten

dabei Lösungen und Best-Practices der Phasen Informationsextraktion, Modellbildung, Modellerweiterung/-vorverarbeitung (inkl. Modellintegration) und Sicherheitsanalyse. Somit bilden sie eine Grundlage für das in Abschnitt 3.5 beschriebene Rahmenwerk, das ebenfalls all diese Phasen abdeckt.

3.4.1 Informationsextraktion und Modellbildung

Die Quellen für sicherheitsrelevante Informationen (z.B. Konfigurationen) des Systems wurden in Abschnitten 1.1.1 und 3.3.12 diskutiert. Dort wurde bereits festgestellt, dass bisher verwendete Informationsquellen in industriellen Systemen nur sehr eingeschränkt nutzbar sind. Solange es an Alternativen mangelt, sollen diese Quellen trotzdem unterstützt werden, um die bestmögliche Informationsabdeckung zu erzielen. Diesbezüglich wurde im Rahmen dieser Arbeit beispielsweise eine Monitoring-Lösung eingesetzt (vgl. Abschnitt 3.7).
Ebenfalls dem *Brownfield*-Paradigma[1] folgend, wurde im Rahmen der Dissertation untersucht, ob und wie sich Netzwerkinformationen mit AutomationML darstellen lassen *[Pat17]*. Diese Forschungsarbeit wird im folgenden Abschnitt vorgestellt und adressiert direkt Hypothese 4, die sich mit der Extraktion von sicherheitsrelevanten Informationen mittels AutomationML befasst.
Zudem wurde in *[Pat19d]* untersucht, wie die Verwaltungsschale als Informationsquelle und -extraktionsmethode, sowie zur Modellbildung eingesetzt werden kann. Diese Untersuchung befasst sich demnach mit Forschungsfrage 5 und wird in Abschnitt 3.4.1.2 präsentiert.
Bei der eben bereits angesprochenen Modellbildung, genauer dem Erzeugen von ontologiebasierten Modellen aus den extrahierten Informationen, muss geklärt werden, wie die Informationen zu repräsentieren sind, um die angestrebten Modellverarbeitungs- und Analysemöglichkeiten zu unterstützen. Auch hierzu wurden im Rahmen der Forschungsarbeiten Untersuchungen durchgeführt, deren Erkenntnisse in Abschnitt 3.4.1.3 diskutiert werden.

[1] Entwicklungsansätze können in Brown- und Greenfield unterteilt werden, wobei Brownfield den Einsatz der zu entwickelnden Lösung in aktuellen Umgebungen und mit Greenfield der Einsatz in einer Altlasten-unabhängigen Umgebung adressiert [Som12a].

3.4.1.1 Netzwerkinformationen in Engineering-Werkzeug-Exporten

Wie in Abschnitt 2.2 erläutert, wurde AutomationML für den Informationsaustausch zwischen Engineering-Werkzeugen entwickelt. Ist die Extraktion von Netzwerkinformationen mittels AutomationML möglich, können Exports aus Engineering-Werkzeugen für die Sicherheitsanalyse eingesetzt werden. Dies ist, wie ebenfalls bereits erwähnt, besonders für die Abdeckung von Komponenten relevant, aus denen die Informationen anderweitig nicht zu extrahieren sind (z.B. industrielle Steuerungen).

Das generelle Problem von AutomationML ist die fehlende, explizite Definition von Restriktionen, die in anderen Sprachen einen Großteil der maschineninterpretierbaren Semantik ausmacht. Diese Restriktionen können nur den von der Community definierten Modellierungsmethoden entnommen werden, die durch Whitepaper, Normen und Best-Practice-Dokumente spezifiziert werden müssen. Zudem werden für diese Spezifikationen Vorlagen in Form von Rollenklassen- und Schnittstellenklassen-Bibliotheken veröffentlicht, um die in den Spezifikationen behandelten Konzepte bereitzustellen und dem Modellierer dabei zu helfen die Methoden umzusetzen. Diese Vorlagen sind, neben Referenzen auf Terminologien, die einzigen durch AutomationML explizit darstellbaren Semantiken. Um homogene und konsistente Modelle in AutomationML zu ermöglichen, werden die genannten Spezifikationen verwendet und resultierende Modelle über eigens programmierte Konformitäts- und Konsistenzprüfungen, oft als Teil der Import-Funktionalität einer Anwendung, validiert.
Vor der Veröffentlichung von *[Pat17]* gab es bereits eine Spezifikation zur Modellierung von Netzwerkinformationen in AutomationML [Aut14]. In *[Pat17]* konnte jedoch gezeigt werden, dass die dort beschriebene Modellierungsmethode selbst für das einfache Beispielnetzwerk aus Abbildung 3.11 nicht zu einer konsistenten Modellierung der Informationen führen und somit auch nicht von Konformitäts- und Konsistenzprüfungen abgedeckt werden kann. Dies wird in folgenden Paragraphen beschrieben.

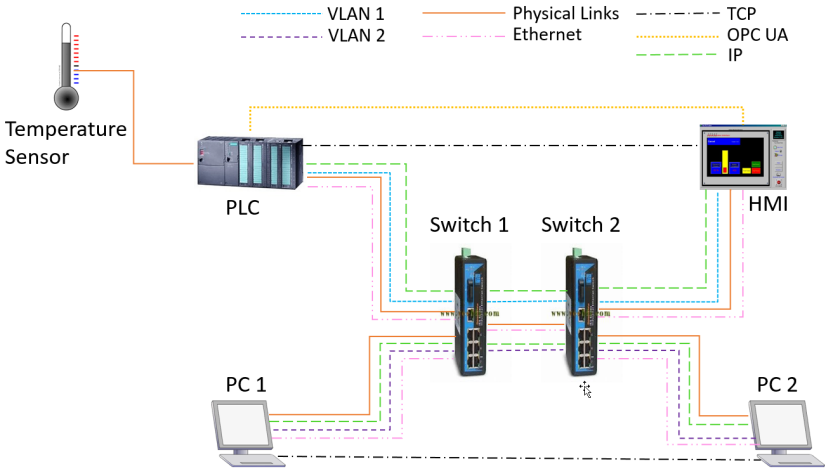


Abbildung 3.11: Beispielnetzwerk, welches in AutomationML modelliert werden soll.

3.4.1.1.1 Untersuchung der existierenden Modellierungsmethode

Abbildung 3.11 zeigt das angesprochene Minimalbeispiel für ein industrielles Netzwerk (wie es z.B. in einer Maschinenzelle zu finden ist). Dieses besteht aus einem Temperatursensor (engl. Temperature Sensor), zwei PCs, einer industriellen Steuerung (engl. PLC), einem HMI und zwei Switches. Dabei ist dieses Netzwerk in zwei virtuelle Netzwerke (*Virtual Local Area Network* (*VLAN*)) aufgeteilt, die Komponenten kommunizieren über Ethernet, IP, TCP und OPC UA und sind dabei über physische Verbindungen (engl. Physical Links), wie Kabel, verbunden. Zunächst wurde versucht die entsprechenden Rollen- und Schnittstellenklassenbibliotheken aus der Spezifikation zu verwenden, die in den Abbildungen 3.12 und 3.13 zu sehen sind, um das Beispielnetzwerk zu modellieren. Die beiden Bibliotheken definieren dabei die Rollen- und Schnittstellen-Klassen `PhysicalX` und `LogicalX`, wobei `X` als Platzhalter für

je eines der folgenden Konzepte dient: `Device`, `Endpoint`[1], `Connection` oder `Network`.

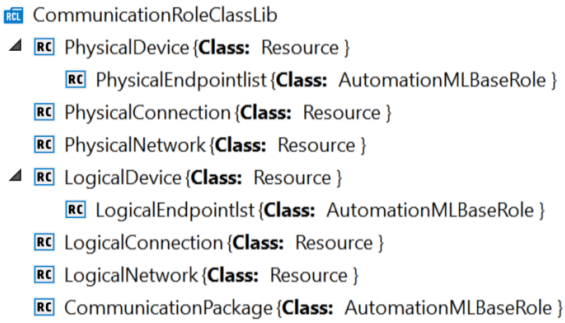


Abbildung 3.12: Rollenklassen-Bibliothek, wie sie von der Spezifikation in [Aut14] vorgegeben wird.

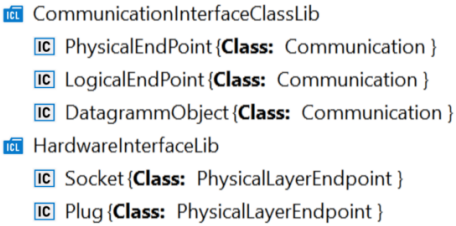


Abbildung 3.13: Schnittstellenklassen-Bibliothek, wie sie von der Spezifikation in [Aut14] vorgegeben wird.

[1] In den Originalbibliotheken gibt es ebenfalls die Schreibweise „EndPoint“, welche sich semantisch jedoch nicht von „Endpoint“ unterscheidet.

Abbildung 3.14 visualisiert die entsprechende Methode zu deren Verwendung. Ein physisches Gerät (`PhysicalDevice`, z.B. ein Switch oder ein PLC) kann dabei mehrere logische Geräte (`LogicalDevice`) enthalten, beispielsweise Anwendungen mit eigenen Endpunkten von der Klasse `LogicalEndpoint`. Außerdem wird eine physische Verbindung (`PhysicalConnection`) als Relation zwischen zwei externen Schnittstellen (engl. *External Interfaces*) von Geräten definiert, indem die Verbindung zwei `PhysicalEndpoints` der Schnittstellenklasse `Plug` enthält, die mittels interner Verknüpfung (engl. *Internal Link*) mit je einem `PhysicalEndpoint` der Klasse `Socket` verbunden sind, welche wiederum Teil der zu verbindenden Geräte sind. Analog verhält es sich mit logischen Verbindungen, Geräten und Endpunkten.

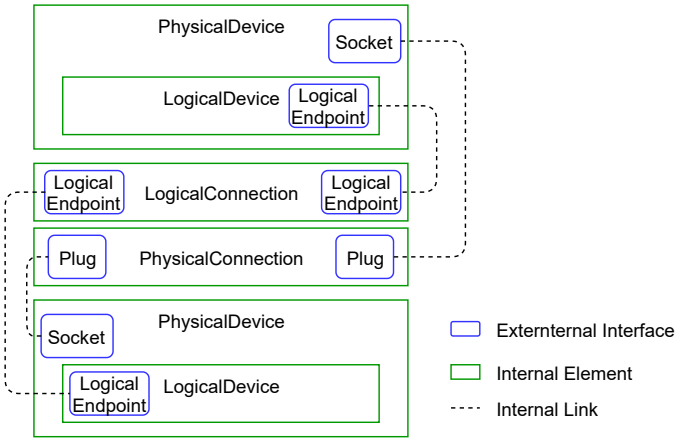


Abbildung 3.14: Konzepte für physische und logische Verbindungen, Geräte und Schnittstellen wie sie von der Spezifikation in [Aut14] vorgegeben werden.

Für die Modellierung sollen aussagekräftigere Klassen für die internen Elemente (IEs) der Verbindungen und Geräte erzeugt werden. [Aut14] nennt als Beispiele `CommunicationXYPhysicalPlug`, `CommunicationXYPhysicalSocket`

und `ApplicationXYLogicalEndpoint`. Gewissermaßen als Container für die Verbindungen gibt es zudem noch die beiden Klassen `PhysicalNetwork` und `LogicalNetwork`. Die Spezifikation orientiert sich an dem klassischen ISO/IEC *OSI-Modell* [ISO94], welches ein Referenzmodell für Kommunikationsnetzwerke nach dem Schichtenprinzip definiert. Es definiert die folgenden Schichten, denen hier beispielhaft die Protokolle des Minimalbeispiels zugeordnet sind:

1. Bitübertragungsschicht (z.B. 1000BaseT, Ethernet)
2. Sicherungsschicht (z.B. Ethernet, VLAN)
3. Vermittlungsschicht (z.B. IP)
4. Transportschicht (z.B. TCP)
5. Sitzungsschicht (z.B. OPC UA)
6. Darstellungsschicht (z.B. OPC UA)
7. Anwendungsschicht (z.B. OPC UA)

Die Spezifikation ordnet die `PhysicalX`-Klassen den Schichten 1-2 und die `LogicalX`-Klassen den Schichten 3-7 zu.
Damit lassen sich sowohl physische Geräte, als auch physische Verbindungen zufriedenstellend modellieren. Allerdings findet die Modellierung von VLANs nach der Spezifikation mittels physischer Verbindungen und Schnittstellen statt, da VLANs auf Schicht 2 des OSI-Modells arbeiten. VLANs agieren semantisch gesehen jedoch auf einer logischen Ebene und verwenden eine beliebige Anzahl physischer Verbindungen. Der Spezifikation folgend können VLANs demnach nicht sinnig modelliert werden.
Dies führt direkt zu einem weiteren Problem, denn es ist nicht spezifiziert, wie die Zusammenhänge unterschiedlicher Schichten und entsprechend die Zusammenhänge von Protokollen modelliert werden. Dadurch ist weder eine einheitliche Modellierung von Protokollstapeln noch von Beziehungen unterschiedlicher Verbindungen möglich.
Durch die fehlende Methode, diese Grundlagen der Netzwerktechnik abzubilden, ist es zudem nicht nur unklar, wie die Modellierung auszusehen hat, es ist ebenso unklar, wie ein entsprechendes Modell interpretiert werden soll, was dessen Verwendung in weiteren Werkzeugen verhindert. Wie in *[Pat17]*

beschrieben wurde die Methode aus [Aut14] daher abgeändert und um explizite Modellierung von Netzwerkgrundlagen erweitert. Dies wird im folgenden Paragraphen genauer beschrieben.

3.4.1.1.2 Neue Methode mit expliziter Modellierung von Netzwerkgrundlagen

Die Grundlage für die neue Methode bot die Masterarbeit von Sarkar *[Sar17]*. Die darin beschriebene Methode wurde im Nachgang zur Masterarbeit jedoch weiterentwickelt und in *[Pat17]* veröffentlicht. Diese weiterentwickelte Methode wird in den folgenden Absätzen zusammenfassend beschrieben. Zur besseren Nachvollziehbarkeit ist das Modell, welches mit dieser Methode zur Repräsentation des Beispielnetzwerks aus 3.11 erzeugt wurde, auf GitHub verfügbar[1].

Zunächst wird statt der Schichten 1 und 2 nur die Schicht 1 über die `PhysicalX`-Klassen abgebildet. Schichten 2-7 werden über die `LogicalX`-Klassen repräsentiert. Diese Zuordnung wird durch das Einführen von schichtenspezifischen `Endpoint`-Schnittstellenklassen in einer neuen OSI-Modell-Bibliothek (`IsoOsiLib`) verfestigt (vgl. Abbildung 3.15). Diese neuen `Endpoint`-Klassen werden dann für die Erzeugung von Protokollschnittstellen (vgl. `ProtocolLib` aus Abbildung 3.15) verwendet. Um die Unterstützung einer Technologie durch ein Gerät abzubilden, wird für die Technologie eine Rollenklasse angelegt. In Abbildung 3.16 ist in einem Ausschnitt der Instanzhierarchie zu sehen, wie Switch1 des Beispiels aus Abbildung 3.11 die Rollenklasse `VlanDevice` unterstützt (definiert über die `SupportedRoleClass`-Beziehung).
Passend zu den Protokollschnittstellen wurden zudem `Connection`-Rollenklassen für jedes Protokoll angelegt (siehe Abbildung 3.17), die als Vorlage bereits mit den entsprechenden Schichten-spezifischen Protokollschnittstellen ausgestattet wurden. Durch die Nutzung dieses Vorlagenprinzips können Fehler beim Einsatz der neuen Methode verhindert werden.

[1] https://github.com/FlorianPatzer/aml_network_modelling.

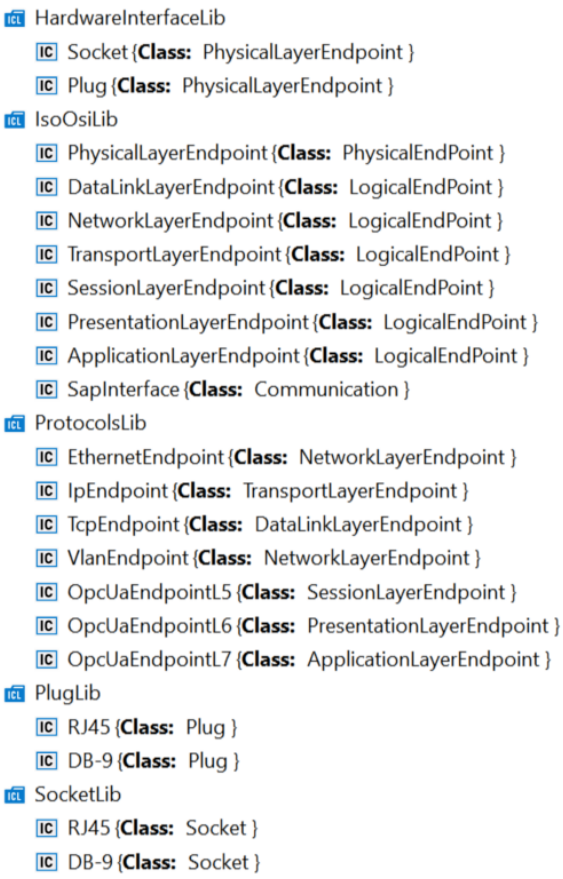


Abbildung 3.15: Schnittstellenklassen der neuen Methode zur Modellierung von Netzwerkinformationen in AutomationML.

Teil der neuen Methode ist zudem die Einschränkung, dass nur logische Endpunkte der selben Klasse miteinander verknüpft werden dürfen. Nur Schnittstellen desselben Typs miteinander zu verlinken ist eigentlich auch eine Prämisse des Basiskonzepts der ursprünglichen AutomationML-Definition [Int14], wurde aber in den existierenden Bibliotheken oft nicht

umgesetzt (siehe z.B. `Socket` und `Plug`, die zwar den selben Elterntyp besitzen, selbst aber unterschiedlich sind und trotzdem, deren Spezifikation folgend, verbunden werden sollen). Des Weiteren werden Netzwerke in Hierarchien modelliert und enthalten zwar, wie in der ursprünglichen Spezifikation, Verbindungen, diese sind aber nun Protokoll-spezifisch und dürfen nur noch über ihre OSI-Schichtschnittstellen verbunden werden.

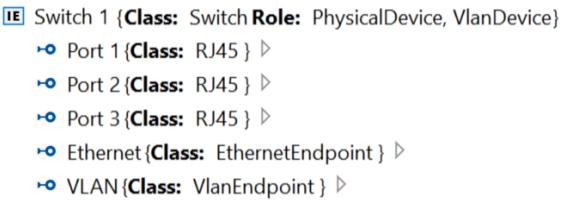


Abbildung 3.16: Instanzhierarchie von Switch1.

Ein weiterer wichtiger Aspekt ist die Beziehung zwischen den einzelnen Schichten. Hierfür wurde das Konzept der *Service Access Points* (*SAP*), also festgelegte Übergangspunkte zwischen den Protokollen, mithilfe der `SapInterface`-Schnittstelle modelliert. Die neue Methode verlangt, dass jede Verbindung nur je eine `SapInterface`-Schnittstelle für die darunter- und darüberliegende OSI-Schicht, als auch die eigene OSI-Schicht besitzt, und dass diese Schnittstellen ausschließlich über die Rollenklassen der `IsoOsiServiceAccessPointLib` miteinander verbunden werden dürfen. Da diese, wie in Abbildung 3.18 zu sehen, nur Schnittstellen zweier aufeinanderfolgender Schichten enthalten, schränkt dies die Modellierung der OSI-Schichten-Beziehungen ein (auch hier nur via Vorlage und Definition der Methode, da in der Instanzhierarchie beliebige Schnittstellen hinzugefügt werden könnten). Als Hinweis ist hier noch hinzuzufügen, dass die Reihenfolge der Schichten in der Benennung von Elementen irrelevant ist. Beispielsweise hätte statt `Layer1Layer2Sap` auch `Layer2Layer1Sap` gewählt werden können. In der Methode wird jedoch als Konvention die niedrigere

Ebene zuerst genannt.

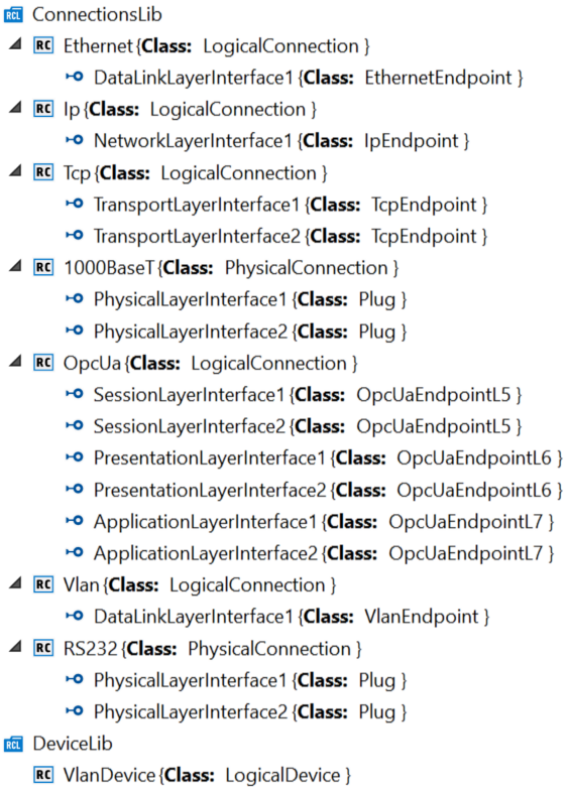


Abbildung 3.17: Rollenklassen der neuen Methode zur Modellierung von Netzwerkinformationen in AutomationML.

Wie das für die IP- und TCP-Netzwerke des Minimalbeispiels aus Abbildung 3.11 aussieht, ist in Abbildung 3.19 zu sehen. Darin werden die Instanziierungen der Schichtenverknüpfungen durch die IEs `Sap23` und `Sap34`, also

Verbindungen der Schichten 2 und 3 (für Ethernet und IP) sowie 3 und 4 (für IP und TCP), dargestellt.

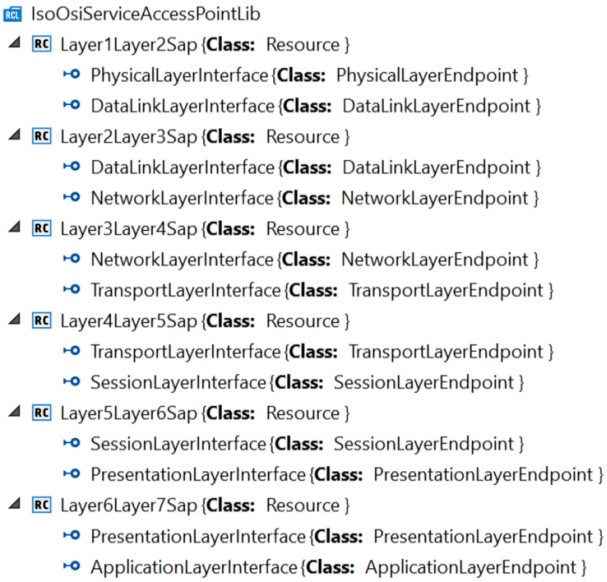


Abbildung 3.18: Service-Access-Point-Rollenklassen der neuen Methode zur Modellierung von Netzwerkinformationen in AutomationML.

Durch die Modellierung des Minimalbeispiels, konnte in *[Pat17]* gezeigt werden, dass die neue Methode eine einheitliche Modellierung von Netzwerkverbindungen und Protokollen, sowie ihrer Relationen untereinander ermöglicht. Dies stellt die Grundlage der Exportfunktionalität von Netzwerkinformationen durch Engineering-Werkzeuge dar. Denn fehlen die nötigen Modellierungs- und somit Interpretationsvorgaben für eine Informationsdomäne, werden die Hersteller von verschiedenen Engineering-Werkzeugen diese Domäne im Rahmen des AutomationML-Exports nicht abdecken, da sie nicht erwarten können, dass die zugehörigen Informationen beim Import

korrekt interpretiert werden.

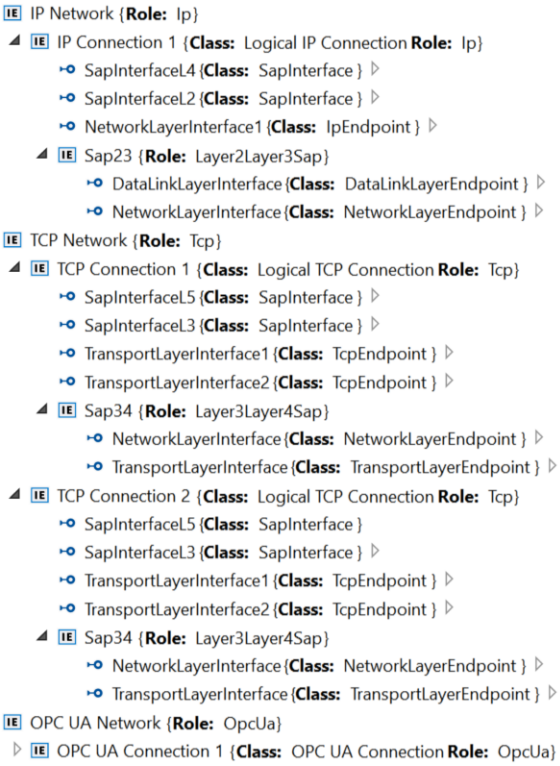


Abbildung 3.19: Instanzhierarchie-Ausschnitt des modellierten Beispielnetzwerkes mit Darstellung der OSI-Schichtenverknüpfungen.

Nichtsdestotrotz bleibt das Problem bestehen, dass diese Methode aufgrund der sprachlichen Beschränkungen von AutomationML nicht explizit im Modell verankert werden kann und dadurch immer eine gewisse Interpretationsfreiheit bei der Modellierung bleibt.

Eine Lösung für dieses Problem wäre die Darstellung der Restriktionen aus der Methode in einer Ontologie. In einem Greenfield-Ansatz könnte man AutomationML mit der mächtigeren Web Ontology Language ersetzen, die hinreichend ist, die gesamte Struktur von AutomationML abzudecken [Gla16]. So lassen sich Restriktionen der Methode in OWL definieren, wie dies für die ontologiebasierte Modellierung von Systemen für den Einsatz von Sicherheitsanalysen üblich ist. Für einen Brownfield-Ansatz, bei dem AutomationML weiterhin verwendet werden soll, lässt sich ein AutomationML-Modell entweder direkt in OWL umwandeln (wie von Glawe und Fay vorgestellt [Gla16]), oder, um die sichere Übertragung des Modells zu garantieren, auf standardisiertem Wege in ein OPC-UA-Informationsmodell transformieren [AUT16], welches wiederum nach OWL abgebildet werden kann. Diese Abbildung von OPC UA auf OWL wurde im Rahmen dieser Dissertation erstmals untersucht und wird im folgenden Abschnitt vorgestellt.

3.4.1.2 Verwaltungsschale als einheitliche Informationsschnittstelle

Ein zukunftsorientiertes und alternatives Konzept zur Extraktion von Informationen über Komponenten, ist die bereits beschriebene Verwaltungsschale. Die Verwaltungsschale bietet per Definition ein digitales Abbild eines Assets und eine Schnittstelle, über die eine digitale Repräsentation beliebiger Aspekte eines Assets verwaltet werden kann.
Wie diese Repräsentation aussieht, wird in der Definition weitgehend offen gelassen. Das von der Plattform Industrie 4.0 definierte Metamodell der Verwaltungsschale zeigt, ähnlich wie AutomationML, eine abstrakte Struktur, die zur Erstellung konkreter Modelle genutzt werden soll und einige Basiskonzepte enthält (vgl. [Pla19]). Spezifischere Modelle sollen in sogenannten Teilmodellen definiert werden. Des Weiteren schlägt [Pla19] unter anderem AutomationML und OPC UA als Umsetzung der Verwaltungsschale vor und zeigt eine Modellierung des Metamodells in AutomationML und eine weitere in OPC UA. Die technologische Umsetzung der Verwaltungsschale bleibt weitgehend undefiniert.
Im Rahmen der Forschungsarbeiten zu dieser Dissertation wurden mögliche

Umsetzungen der Verwaltungsschale zur Informationsextraktion und Modellbildung untersucht *[Pat19d]*, da diese die Chance für eine interoperable Informationsquelle für beliebige Komponenten eines industriellen Systems bietet. Dabei wurde das in [Pla19] präsentierte abstrakte Modell zur Verwaltungsschale noch nicht verwendet, da dieses zum Zeitpunkt der Untersuchung noch keinen stabilen Stand erreicht hatte und für die Informationsextraktion für Sicherheitsanalysen keinen ausschlaggebenden Mehrwert bot. Vielmehr wurde untersucht, wie sich generell eine Implementierung der Verwaltungsschale (mit beliebigem abstraktem Modell) mithilfe von OPC UA umsetzen lässt. Da OPC UA genau wie AutomationML keine explizite Definition von Restriktionen zulässt, wurden Abbildungsstrategien von OPC UA zu OWL 2 DL erarbeitet und untersucht.

Die Repräsentation von Informationen, die aus mehreren Quellen mit unterschiedlichen semantischen Darstellungen dieser Informationen stammen, benötigt die Abbildung dieser Semantiken auf eine gemeinsame Systemontologie, die dann beispielsweise für die Sicherheitsanalysen verwendet werden kann. Diese bildet eine Systemdomäne ab und kann wiederum aus mehreren Domänen und somit mehreren Ontologien zusammengesetzt werden.
Es wird also eine Strategie der Informationsintegration benötigt, welche dieses Ziel unterstützt. Dabei gibt es generell zwei Ansätze. Die erste Alternative verfolgt die Strategie, die Schemata und Sprachen der jeweiligen Quellen in einer Ontologie darzustellen, um die Daten direkt dorthin zu kopieren und über weitere Verarbeitungsschritte, bei denen dann auch Reasoning eingesetzt werden kann, in eine gemeinsame Domäne zu überführen.
Die zweite Alternative ist die Interpretation der Daten bei deren Extraktion, um diese direkt in eine gemeinsame Systemontologie zu überführen. Beide Varianten wurden in *[Pat19d]* untersucht, um festzustellen, welche sich besser für die Verwendung von Verwaltungsschalen und ähnlichen Konzepten eignen. Diese Analyse wird hier zusammenfassend beschrieben.

Da OPC UA bereits von der Plattform Industrie 4.0 als eine der Realisierungstechnologien für die Verwaltungsschale festgelegt wurde, wurde OPC UA auch für die hier beschriebene Untersuchung als Datenübertragung ausgewählt. Wie zuvor erwähnt, existiert auch eine Spezifikation zur Übertragung von

AutomationML-Modellen via OPC UA. Somit konnten durch die Wahl von OPC UA bereits zwei wichtige Darstellungsformen (OPC UA und AutomationML) inklusive der sicheren Übertragung der Informationen von industriellen Komponenten zu einer Senke unterstützt werden.
Untersucht wurde zunächst die erste Alternative, für welche die Abbildungszuordnung aus Tabelle 3.7 verwendet wurde, um sowohl TBox als auch ABox einer OWL-Ontologie zu erzeugen. Die Namen der Klassen, Individuen und Rollen wurden dabei aus den Namen der Objekttypen (engl. *Object Types*), Objekte (engl. *Objects*) und Variablen (engl. *Variables*) erzeugt. Die Zuordnung deckt nicht alle Konzepte von OPC UA ab, sondern nur jene, die im Rahmen des OPC UA Schemas des BMBF-Projektes FlexSi-Pro[1] eingesetzt wurden, in dem diese Zuordnungsstrategie unter anderem für die Analyse von Systemen bezüglich bekannter Software-Schwachstellen eingesetzt wurde (vgl. Abschnitt 3.7.4). Aufgrund dieses Einsatzszenarios eignete sich besonders die Extraktion und Repräsentation eines Software-Inventars, um die verschiedenen Ansätze zu vergleichen. Ein Minimalbeispiel zur Darstellung eines solchen Inventars in OPC UA kann Abbildung 3.20 entnommen werden.

Tabelle 3.7: Abbildung von OPC UA auf Semantic Web Konzepte.

OPC UA	OWL
Object Type	Class (Klasse)
Object	Individual (Individuum)
hasTypeDefinition	Subclass Relation (Subklassenbeziehung)
hasComponent von Object auf Object	Object Property (abstrakte Rolle)
hasComponent von Object auf Variable	Data Property (konkrete Rolle)

Die entsprechende durch die generische Generierung erzeugte Ontologie ist in Abbildung 3.21 zu sehen. Wie in *[Pat19d]* beschrieben, ist diese Variante ein,

[1] https://www.ip45g.de/projekte/flexsi-pro, zuletzt zugegriffen: 31.01.2021.

mit wenig Aufwand verbundener Weg, die Semantic-Web-Werkzeuglandschaft nutzen zu können. Sich mit *[Pat19d]* zeitlich überschneidend, haben Bakakeu et al. [Bak19] aus eben diesem Grund einen sehr ähnlichen Ansatz gewählt und präsentiert, der mit ähnlichen Abbildungsregeln ebenfalls eine solche generische Erzeugung von Ontologien aus OPC-UA-Informationsmodellen bietet.

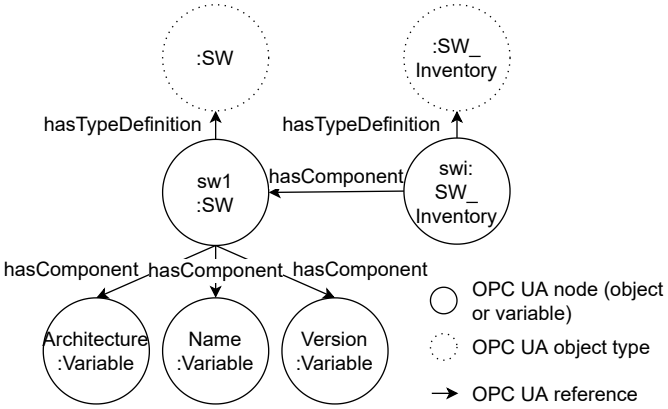


Abbildung 3.20: Beispiel OPC-UA-Nodesets zur Darstellung eines Software-Inventars.

Das Hauptproblem mit dem generischen Ansatz ist jedoch die Weiterverarbeitung der Informationen. Abfragen oder Regeln können nicht im Vorfeld formuliert werden und Restriktionen können ebenfalls nicht auf Ebene der zu transformierenden Informationen definiert werden. Der Grund dafür ist die fehlende Kenntnis über die in der Zukunft erzeugte Ontologie. Dies ist auch in [Bak19] zu erkennen, da die Weiterverarbeitung mithilfe von SPARQL-Abfragen Informationen nutzt, die im Vorfeld nicht bekannt sind. Somit kann dieser Ansatz lediglich verwendet werden, um individuelle Untersuchungen im Nachgang der Transformation durchzuführen und ist für das hier beschriebene und angestrebte Sicherheitsanalysekonzept nicht einsetzbar.
Die zweite Alternative wurde in zwei Varianten entwickelt. Eine komplexe

Abbildung, die es möglich macht, Ontologien einer beliebigen, existierenden TBox zu instanziieren und eine Mischform, die eine strukturerhaltende Abbildung umsetzt, indem sie die Instanziierung einer, für ein bestimmtes Schema entwickelten, TBox ermöglicht. Die entsprechenden Ergebnisse bei Durchführung der jeweiligen Transformation sind in den Abbildungen 3.22 und 3.23 zu sehen.

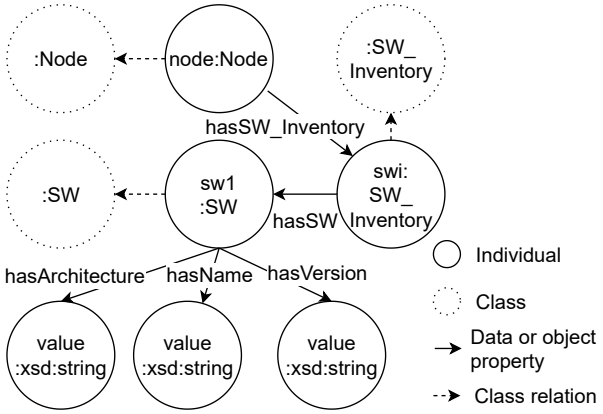


Abbildung 3.21: Mittels generischer Abbildung erzeugte Software-Inventar-Ontologie.

Beide Varianten bieten Vor- und Nachteile, die keine eindeutige Schlussfolgerung zulassen, welche der beiden Varianten verwendet werden sollte. Die strukturerhaltende Abbildung ist simpel und mit einem generischen Programm leicht umsetzbar. Zudem bleiben Entwurfsentscheidungen der ursprünglichen Informationsdarstellung erhalten und die TBox wird vor der Datentransformation festgelegt (vgl. Software, Software-Inventory, sowie *architecture*, *name*, usw. aus Abbildung 3.22), wodurch die Weiterverarbeitungsschritte ebenfalls bereits im Vorfeld entworfen und konfiguriert werden können. Die TBox ist dabei nicht nur bekannt, sondern bleibt auch bei Updates der Daten stabil.

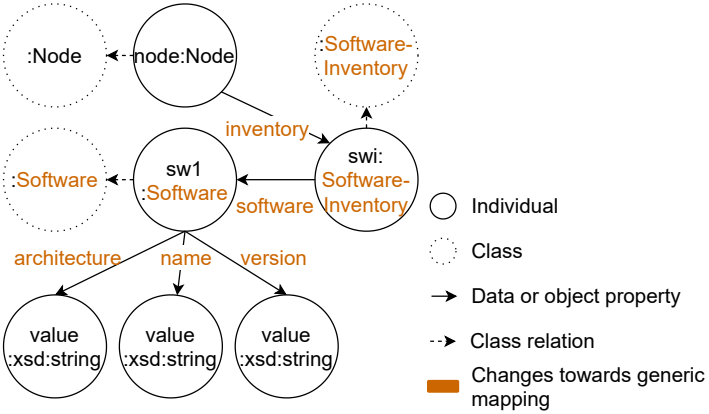


Abbildung 3.22: Mittels strukturerhaltender Abbildung erzeugte Software-Inventar-Ontologie.

Allerdings muss die entsprechende TBox für jedes Schema manuell erzeugt und auf eine einheitliche Systemontologie[1] abgebildet werden. Je öfter dieses Schema zum Einsatz kommt, desto geringer ist der relative Aufwand dieser manuellen Erzeugung und Abbildung, da die TBox und die entsprechende Abbildung beliebig wiederverwendet werden kann. Diese einfache Wiederverwendung ist der Tatsache geschuldet, dass die Abbildung mithilfe einer Abbildungsontologie (vgl. Abschnitt 3.4.1.3.2) umgesetzt und somit geteilt werden kann, ohne eine Software anpassen zu müssen. Für die komplexe Abbildung wurde das Python-Paket *X2Owl*[2] entwickelt, welches diese Abbildung auf eine einheitliche Systemontologie, direkt über ein YAML-Konfigurationsschema ermöglicht. Listing A.1 zeigt einen Ausschnitt der Konfiguration zur Extraktion von Software-Inventar-Informationen als einen Einblick in das Schema. Wie in dem Listing zu erkennen ist, orientiert sich das Konfigurationsschema daran, wie Kommandozeilenausgaben geparst werden können und bietet dem Nutzer

[1] Diese einheitliche Systemontologie kann in dem später vorgestellten Rahmenwerk auch ausgetauscht werden, an dieser Stelle geht es jedoch zunächst darum, dass alle Komponentenontologien auf eine gemeinsame Systemontologie abgebildet werden.

[2] https://github.com/FlorianPatzer/x2owl.

die Möglichkeit, entweder ein solches Parsing oder einen OPC-UA-Server als Datenquelle zu verwenden.

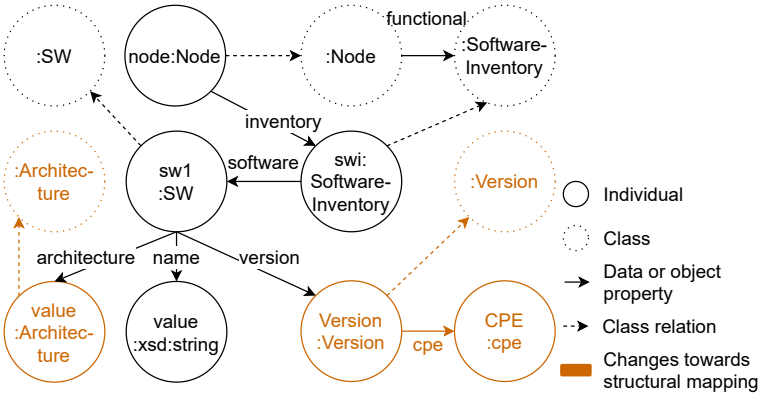


Abbildung 3.23: Mittels komplexer Abbildung erzeugte Software-Inventar-Ontologie.

Das Programm und sein zugehöriges Konfigurationsschema sind so gestaltet, dass auch Vorverarbeitungsroutinen konfiguriert werden können. So werden beispielsweise Werte von OPC-UA-Variablen vorverarbeitet und in verschiedene Informationen aufgeteilt (vgl. Architecture, Version und CPE aus Abbildung 3.23). Ein Beispiel für die Konfiguration und Anwendung solcher Routinen ist in Listing A.2 zu sehen.
Zusammengefasst ist die komplexe Abbildung weit mächtiger als die strukturerhaltende Abbildung und deckt diese zumindest programmatisch ab (X2Owl kann unverändert eingesetzt werden, um auch die strukturerhaltende Abbildung umzusetzen). Ein Nachteil der komplexen Abbildung ist jedoch, dass sie keine schemaabbildende Ontologie erzeugt, die entsprechend von konsolidierten Schemata profitiert. Für Digitale Zwillinge ganzer Systeme, die aus den Informationen mehrerer Verwaltungsschalen (o.ä. Quellen) zusammengesetzt werden, gibt es allerdings eine Vielzahl konsolidierter Schemata, deren Domänen sich überschneiden. Daher bringt es einen größeren Vorteil,

Vorberechnungen und ggf. Schemaanpassungen vorzunehmen und dafür eine höhere Einheitlichkeit der Implementierungen für die Weiterverarbeitung und Analyse der Modelle zu ermöglichen, als an Schemata, wie OPC-UA-Companion-Specs, festzuhalten.
Somit resultiert aus der Untersuchung, dass die komplexe Abbildung, unter Verwendung einer entsprechenden Werkzeugunterstützung, wie X2Owl, für die Informationsextraktion und -repräsentation insbesondere für Sicherheitsanalysen bevorzugt werden kann. Dass diese Schlussfolgerung valide ist, wurde durch den mehrfachen Einsatz der komplexen Abbildung mittels X2Owl in den Sicherheitsanalysen, im Rahmen der Forschungsarbeiten zu dieser Dissertation gezeigt. Konkrete Beispiele hierfür werden in Abschnitten 3.7.2, 3.7.3 und 3.7.4 beschrieben.

3.4.1.3 Wissensrepräsentation

Abgesehen von der Abdeckung der Informationsquellen und den damit zur Verfügung stehenden Daten hängen die möglichen Erkenntnisse aus Sicherheitsanalysen entscheidend von der Repräsentation des extrahierten Wissens ab. In diesem Abschnitt werden wichtige herausgearbeitete Aspekte der Wissensrepräsentation diskutiert. Dabei wird zunächst darauf eingegangen, welche Wissensdomänen bei der Sicherheitsanalyse unterschieden werden (vgl. Abschnitt 3.4.1.3.1) und wie diese im Sinne von Separation-of-Concerns voneinander getrennt und im richtigen Kontext vereint werden können (vgl. Abschnitt 3.4.1.3.2).
Auch wurden bei den Untersuchungen zu dieser Arbeit Probleme der bisherigen in der Literatur vorzufindenden Ansätze identifiziert, zu deren Vermeidung Abschnitt 3.4.1.3.3 entsprechende Best-Practices diskutiert. Zudem wird die Modellierung des bisher meist außen vor gelassenen, aus Sicht der Wissensrepräsentation aber wesentlichen, Grundlagenwissens in Abschnitt 3.4.1.3.4 adressiert.

3.4.1.3.1 Einsatz von Ontologien

Der Einsatz von Ontologien hat sich in der modellbasierten Sicherheitsanalyse aufgrund der Möglichkeit der maschinellen Interpretation und des Ableitens neuen Wissens über Reasoner bewährt (vgl. Abschnitt 3.3). Eine Ontologie ist zwar eine explizite, hier auch formale, Repräsentation einer Domäne, jedoch ist die Entscheidung, was zu einer bestimmten Domäne gehört, subjektiv. Auf dem Weg von der Informationsextraktion bis hin zur letztendlichen Analyse lassen sich mindestens vier Domänenklassen festlegen:

Komponentendomäne

Dies ist die Domänenklasse der Eingangsdatendomänen, die sich in der Regel durch Datenschemata und zusätzliche Dokumentation festlegen lässt. Im Fall von OPC UA wäre dies zum Beispiel die Zusammensetzung aus den Companion Specs und eigenen Teilmodellen, die für die Erstellung des Informationsmodells einer Komponente eingesetzt wird. Wie an dem Beispiel schon zu erkennen ist, bestehen auch Komponentendomänen bereits aus weiteren Domänen, eine pro Companion Spec und eine für jedes weitere Teilmodell. Um Missverständnissen vorzubeugen, gilt es hier noch darauf hinzuweisen, dass Informationen der Komponentendomäne i.d.R. nicht mit den bereits mehrfach genannten Komponentenmodellen, sondern mithilfe der Quellmodelle (vgl. Abschnitt 3.2.2) dargestellt werden.

Systemdomäne

Dies ist die Domänenklasse des Systemwissens, die aus weiteren Domänen für Semantik zu Netzwerken, Diensten, Komponenten und mehr besteht. Sie stellt somit die Basis für Analysen und das generelle Arbeiten auf Systeminformationen dar. Informationen dieser Domäne werden u.a. in Komponentenmodellen und Systemmodellen dargestellt.

Sicherheitsdomäne

Die Sicherheitsdomänen sind unterschiedliche Sichten auf das Thema Sicherheit. Beispielsweise werden diese durch Standards zu Sicherheitsmanagementsystemen, wie den IEC 62443 oder den ISO 27000, beschrieben. Andere Sichten sind die operative Administration oder sicherheitsbezogene Applikationsdomänen, wie das im Bereich der Vorfallbehandlung etablierte *Security Information and Event Management* (*SIEM*). All diese Bereiche definieren ihre eigenen semantischen Konzepte, welche von entsprechenden Applikationen, wie Sicherheitsanalyseanwendungen, verwendet werden.

Analysedomäne

Analysedomänen definieren sich über die zusätzlichen, in einer Menge von Analysen verwendeten Konzepte, welche nicht von einer der anderen Domänen abgedeckt werden. Beispielsweise können Analysen, wie in Abschnitt 3.3.5 beschrieben, über die automatisierte Erweiterung von Modellen Wissen ableiten und in der Ergebnisauswertung abfragen. Die für die Modellierung dieses abgeleiteten Wissens eingesetzten Konzepte sind nicht unbedingt den zuvor genannten Domänen zuzuordnen, da sie ggf. nur für die interne Verarbeitung der Analyse verwendet werden. In diesem Fall sind sie Teil der Analysedomäne.

Solches Domänenwissen austauschen und kombinieren zu können, und das oft ohne verarbeitende Programme anpassen zu müssen, ist eine der Stärken von Ontologien (vgl. Abschnitt 2.7.4). Diese Domänen überschneiden sich und müssen für eine vollständige Analyse-Pipeline, die von der Informationsextraktion bis zur fertigen Analyse reicht, aufeinander abgebildet werden. Auch dafür sind Ontologien besonders gut geeignet, wie der nächste Abschnitt verdeutlicht.

3.4.1.3.2 Domänentrennung und -abbildung

Wie in dieser Arbeit mehrfach motiviert wurde, ist Separation-of-Concerns eine Kernstrategie für die Optimierung, Austauschbarkeit und Wiederverwendbarkeit von Informationsextraktions-, Modellierungs-, Modellerweiterungs- und Analyseschritten, sowie der Expertisetrennung bei deren Erstellung. Separation-of-Concerns lässt sich in diesem Kontext durch die getrennte Betrachtung von Domänen umsetzen. Beispiele für Klassen solcher Domänen wurden bereits im vorangegangenen Abschnitt vorgestellt. Spezifische Domänen lassen sich durch Ontologien maschineninterpretierbar darstellen und wiederverwenden. Dabei können Expertisebereiche von Anfang an getrennt werden.
Um Separation-of-Concerns auch umsetzen zu können, müssen diese Domänen aber nicht nur getrennt voneinander entwickelt und verwaltet werden, sondern auch zusammen eingesetzt werden können. Im Kontext dieser Arbeit werden dafür Abbildungsontologien verwendet (vgl. Abschnitt 2.7.4). Eine beispielhafte Abbildungshierarchie für die hier beschriebenen Konzepte ist in Abbildung 3.24 zu finden.

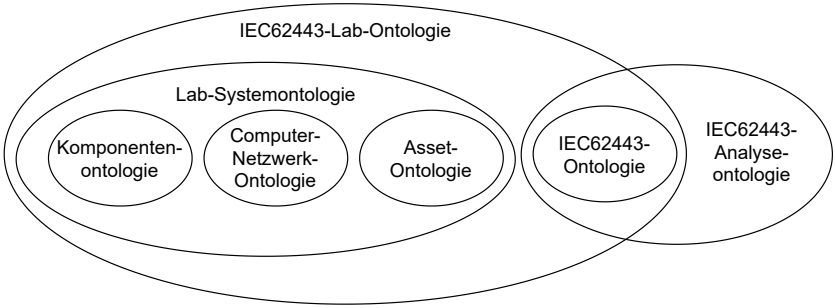


Abbildung 3.24: Beispiel für eine Abbildungshierarchie von für die Sicherheitsanalyse eingesetzten Ontologien.

Dabei sind die Lab-Systemontologie (Ontologie, welche die zur Modellierung der Laborumgebung passenden Konzepte enthält), die IEC-62443-Lab-Ontologie und die IEC-62443-Analyseontologie Abbildungsontologien. Diese Abbildungshierarchie zeigt beispielhaft wie Domänentrennung und -abbildung im Sinne von Separation-of-Concerns für eine IEC-62443-Sicherheitsanalyse im Anwendungsszenario (vgl. Abschnitt 3.1.3) aussehen kann. Es ist wichtig zu erwähnen, dass sowohl für jede der integrierten Ontologien, als auch für jede der Abbildungsontologien Applikationen existieren können, die diese instanziieren, erweitern, analysieren und verarbeiten können oder für administrative Vorgänge editier- und überwachbar machen.

3.4.1.3.3 Probleme existierender Ontologien und Vorschläge von Best-Practices

In den folgenden Paragraphen werden Probleme bisheriger Ansätze identifiziert, die bei der automatisierten Modellierung von Wissen für Sicherheitsanalysen berücksichtigt werden sollten:

Beibehalten des Detailgrades

Unter den genannten verwandten Ansätzen aus Abschnitt 3.3 konnten nur Ontologien identifiziert werden, bei deren Erzeugung offensichtlich bereits bekannt war, welche konkreten Analysen später darauf ausgeführt werden. Ein Indiz hierfür ist das Anlegen von Individuen auf genau dem Abstraktionsgrad, der für die in den Veröffentlichungen präsentierten Beispielanalysen geeignet ist. Das Instanziieren einer Ontologie für beliebige Analysen, muss jedoch das Ziel verfolgen, den Detailgrad der extrahierten Informationen beizubehalten. Bei einigen dieser Ansätze, wie dem von Fenz et al. (vgl. Abschnitt 3.3.5), ist dies durchaus der Fall, da sie nicht für beliebige Analysen entworfen wurden. Andere Ansätze unterstützen allerdings beliebige SPARQL- oder SQWRL-Abfragen, bei denen durch solche Abstraktionen eine Reihe von Abfragemöglichkeiten verhindert werden. Diese können zwar reguläre Ausdrücke oder mathematische Operationen auf Werten mit entsprechenden Datentypen ausführen,

jedoch nicht auf IRIs. Demzufolge sollten IRIs zur Identifikation von zu verarbeitenden Informationen, aber nicht zu deren Repräsentation genutzt werden. Als Beispiel sei hier eine IP-Adresse genannt. Diese kann zwar als IRI definiert werden, weil man eventuell davon ausgeht, dass es im System nur eine Instanz dieser IP-Adresse geben kann, allerdings können die beispielhaft genannten Operationen nur auf Werten ausgeführt werden, was eine Repräsentation von IP-Adressen, z.B. als Strings oder Unsigned Integer, erfordert.

Ununterscheidbare Daten

Zudem wurden bei existierenden Ansätzen zur Repräsentation von IT/OT-Wissen keine Konzepte zum Umgang mit redundanten, zur Instanziierung verwendeten Daten betrachtet. So wurden keine Namensräume definiert und Attribute wie Namen von Schnittstellen, Diensten oder Geräten wurden nicht als Zeichenketten, sondern als IRIs definiert, wodurch Entitäten mit gleichem Namen keine unterschiedlichen IRIs garantiert wurden. Auch entsprechende Zusammenführungen, die anhand unterschiedlicher Strategien vorgenommen werden können, wurden nicht thematisiert. Dabei ist es beispielsweise völlig normal in einem System eine Vielzahl von Netzwerkschnittstellen *eth0* zu haben. Als IRI *eth0* zu verwenden ergibt nur dann Sinn, wenn all diese Schnittstellen wirklich zu einer Instanz vereinheitlicht werden sollen. Dies ist jedoch ein ungewünschter Informationsverlust, da jede *eth0*-Schnittstelle eine eigenständige Entität mit eigenen Relationen und Werten darstellt, die dann verloren gehen würden. Welche Informationen zu einer einheitlichen Repräsentation zusammengeführt oder anderweitig verknüpft werden, sollte daher im Nachgang zur Modellbildung und Integration im Rahmen von austauschbaren Strategien entschieden werden. Dadurch bleiben die Originalinformationen erhalten.

3.4.1.3.4 Explizite Modellierung von Grundlagenwissen

Gerade im Bereich von Netzwerkinformationen fällt auf, dass in der Vergangenheit zwar grundlegende Begriffe der Netzwerktechnik in Systemontologien

eingesetzt wurden, jedoch kaum Grundlagenwissen, wie die Abhängigkeit zwischen IP-Adressen und -Netzen oder Netzwerkstapeln. Solches Wissen wurde den Analysen überlassen. Dabei wurde nicht zwischen den verschiedenen Domänen unterschieden. Allerdings ist es geschickter, die explizite Modellierung von Grundlagenwissen zu wählen und deren Einsatz für Reasoning und andere Modellerweiterungsschritte so früh wie möglich in einem Verarbeitungsablauf anzusiedeln. Also dort, wo diese von zukünftigen Schritten genutzt und vorausgesetzt werden können. Dieses Vorgehen wird in Abschnitt 3.4.2 zusammen mit anderen Erweiterungsstrategien genauer erläutert.

3.4.2 Modellintegration und -erweiterung

Die Informationsextraktion und Modellbildung führt zunächst zu Modellen einzelner Komponenten und zusätzlichem Wissen, wie von Überwachungssystemen mitgeschnittene Kommunikationsbeziehungen. Diese Modellarten werden hier analog zu Abschnitt 3.2.2 unter dem Begriff des Komponentenmodells zusammengefasst. Die Komponentenmodelle müssen in ein *Systemmodell*, bzw. die (instanziierte) Systemontologie, überführt werden und erzeugen zunächst das in Abschnitt 3.4.1.3.3 angesprochene Problem der redundanten Daten. An dieser Stelle wird davon ausgegangen, dass die Komponentenmodelle bereits die TBox der angestrebten Systemontologie verwenden und dass innerhalb eines Komponentenmodells keine doppelten IRIs existieren.
Ein informeller Überblick des Integrationsvorgangs ist in Abbildung 3.25 zu finden. Dabei werden die Namensräume der verschiedenen Komponentenmodelle für deren ABoxen geändert, sodass diese ohne Redundanzkonflikte in die Systemontologie eingefügt werden können. So werden z.B. aus zwei Individuen mit der IRI `https://iosb.fraunhofer.de/ICS-Security#indiv1` ein Individuum `https://iosb.fraunhofer.de/ICS-Security/compA#indiv1` und ein Individuum `https://iosb.fraunhofer.de/ICS-Security/compB#indiv1`. Die Systemontologie existiert entweder bereits oder muss erst noch erzeugt werden. Im letzteren Fall besitzt sie noch keine Instanzen. In jedem Fall werden als nächstes jedoch die ABoxen zur Systemontologie

hinzugefügt. Nun, da alle Komponentenmodelle in einer Ontologie zusammengeführt wurden, können Redundanzen aufgelöst werden.

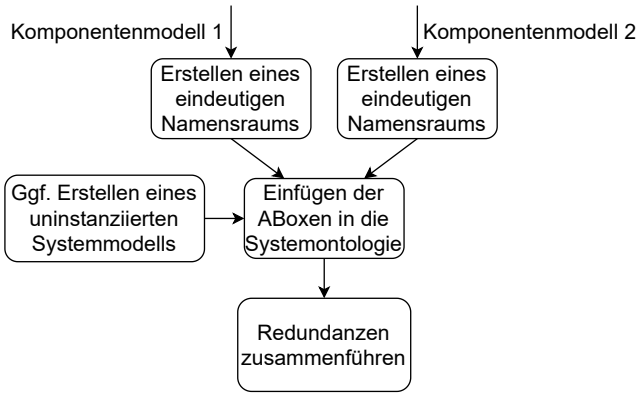


Abbildung 3.25: Integrationsvorgang zur Generierung eines Systemmodells aus Komponentenmodellen.

Gerade bei der Extraktion von Konfigurationsdaten müssen Individuen für Entitäten angelegt werden, die von Konfigurationsrepräsentationen mehrerer Komponenten referenziert werden. Beispiele dafür sind Dienste (z.B. DNS, DHCP, NTP oder SMB), Server für diese Dienste und Netzwerke. Diese werden in den Komponentenmodellen definiert und sind somit ggf. nach dem Einfügen in die Systemontologie mehrfach vorhanden (wenn auch unter verschiedenen IRIs zu finden, dank der eindeutigen Namensräume).
Hierbei macht eine kurze Überlegung jedoch deutlich, dass bereits hier die Austauschbarkeit der Integrationsschritte nötig ist: Wie bereits in Abschnitt 3.2.3 MA6 argumentiert, sind in manchen Unternehmen IPv4-Netzwerke eindeutig (auch lokale Adressbereiche), in anderen wird aber beispielsweise NAT eingesetzt, das verschiedene Netzwerke mit denselben IPv4-Adressen unterstützt. Auch werden beispielsweise Standardabbildungen zwischen Netzwerkdiensten

und Ports in manchen Unternehmen verwendet, in anderen jedoch nicht. Beide Beispiele zeigen, dass eine feste Fusionsstrategie zu Problemen führen kann. Abbildung 3.26 fasst das bisher verfolgte Vorgehen im oberen Teil zusammen und zeigt im unteren Teil das neue Vorgehen inklusive der relevanten Anpassungen (blau markiert).

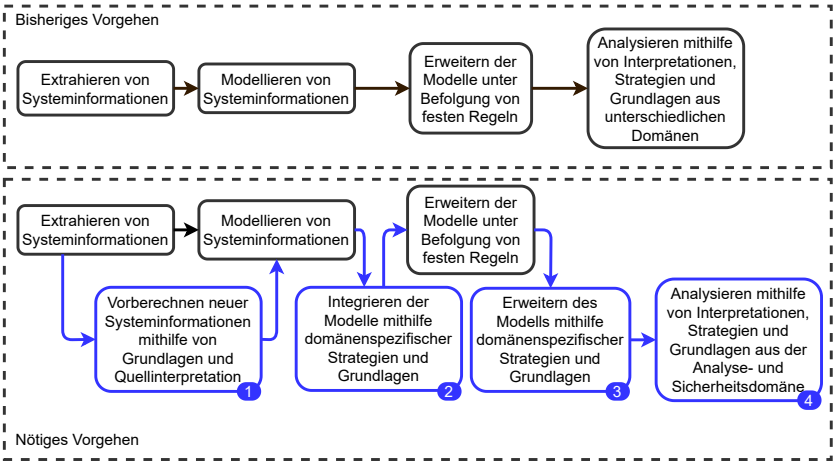


Abbildung 3.26: Herkömmlicher Einsatz von Grundlagenwissen, Strategien und Interpretationen (oben) und Neuverteilung durch Separation-of-Concerns (unten). Dabei sind Änderungen und Neuerungen blau hervorgehoben.

In diesem Abschnitt wurde bisher „Änderung 2“ motiviert. Die weiteren Änderungen werden in den folgenden Abschnitten erläutert.

3.4.2.1 Generalisierung/Abstraktion von Konfigurationssprachen

In bisherigen Arbeiten wurde die Interpretation von Konfigurationen meist ebenfalls in den eigentlichen Analysen vorgenommen. Im Folgenden wird anhand des Beispiels von Konfigurationen für die Netzwerkzugriffskontrolle motiviert, warum dies ein Problem darstellt und wie das Verschieben dieser

Interpretationsphase und die explizite Modellierung von Netzwerkinformationen beiträgt, die Probleme zu lösen. Diese Strategie ist in „Änderung 1“ aus Abbildung 3.26 einzuordnen.
Konfigurationssprachen für NAC sind i.d.R. herstellerspezifisch und unterscheiden sich oft selbst zwischen verschiedenen Lösungen eines Herstellers. Dennoch müssen diese Konfigurationen zur Identifikation und Untersuchung von Inter-Konfigurationseffekten (vgl. Abschnitt 2.6.3) in eine gemeinsame Form überführt werden, da sie sonst in verschiedenen Versionen vorliegen müssen und somit nicht skalieren. Dies wird weiter hinten in diesem Abschnitt genauer erläutert. Zudem müssen Modellerweiterungen und Analysen selbst zur Analyse einer einzigen NAC-Instanz für jede einzelne NAC-Konfigurationssprache erstellt und gepflegt werden und sind somit kaum wiederverwendbar.
Dies gilt natürlich nur, falls sie nicht NAC-Konfigurationssprachen-agnostisch sind, also z.B. keine Auswertung der NAC-Regeln erfordern. Bai et al., die in [Bai19] ein Rahmenwerk für die nutzerberechtigungszentrierte Analyse von NAC-Konfigurationen vorstellten, beschrieben dieses Problem des Vereinens von NAC-Konfigurationsinformationen wie folgt:

> *“It is a challenging task to represent multiple domain information in a single model. An extreme low-level abstraction will make the representation complicated and unusable, while an extreme highlevel abstraction will make the abstraction fail to grasp the influences of configuration changes fully and accurately. So, we focus on the device functions and interrelationships rather than the configuration details.”*, [Bai19]

In *[Pat21]* konnte jedoch gezeigt werden, dass die beiden von Bai et al. beschriebenen Probleme über die Ableitung der *Effektiven Konfiguration* für Flusskontrolle im Rahmen von NAC zu lösen sind. Diese Lösung wird hier zusammenfassend beschrieben.

Der grundlegende Ansatz der Effektiven Konfiguration nutzt aus, dass Analysen von NAC-Konfigurationen und deren Auswirkungen meist den, durch die NAC-Instanzen zugelassenen oder verbotenen, Netzwerkverkehr adressieren

und nicht die konkreten konfigurierten NAC-Regeln. Sie adressiert so also Fragestellungen wie

Kann ein beliebiges Asset aus der Feldebene eine TCP-Verbindung mit einem beliebigen Asset aus dem Büronetzwerk aufbauen?

nicht aber Fragestellungen wie

Besitzt die Regelkette von Firewall 1 eine überdeckte, somit nicht verwendete, Regel?

Eine NAC-Regel besteht grundsätzlich aus einem Muster und einer Aktion. Für jedes Netzwerkpaket, welches bei der NAC-Instanz ankommt, wird geprüft, welches Muster auf das Paket zutrifft. Treffen mehrere Muster auf das Paket zu (dies ist immer dann der Fall, wenn nicht nur die Standardregel zutrifft), wird eine festgelegte Auswertungsstrategie eingesetzt, um jene Regel aus den zutreffenden Regeln zu wählen, deren Aktion auf das Paket angewendet werden soll. Eine solche Auswertungsstrategie ist die „Last-Match-Wins"-Strategie. Diese besagt, dass die Aktion der letzten Regel gewählt wird, deren Muster bei Abarbeitung der Regelkette auf das Netzwerkpaket zutrifft. Mehrere solcher Strategien können innerhalb einer Regelkette verwendet werden, indem Schlüsselwörter als Teil einer Regel angeben, welche Strategie für diese Regel gilt.

Das in *[Pat21]* beschriebene Vorgehen zur Erzeugung des Modells der Effektiven Konfiguration kann wie folgt zusammengefasst werden (der genaue Ablauf kann Anhang A.3 und *[Pat21]* entnommen werden):

1. Lege einen Erlauben-Behälter und einen Ablehnen-Behälter für die entsprechenden Aktionen „Erlauben" und „Ablehnen" an.
2. Bereite die NAC-Regelkette vor, u.a. durch Ersetzen von Dienstnamen durch ihre Ports.
3. Sortiere die NAC-Regelkette so um, dass die gesamte Kette der „Last-Match-Wins"-Strategie entspricht.
4. Gehe die NAC-Regelkette durch und leite für jede Regel alle Netzwerkprotokoll-spezifischen Identifikatormengen (z.B.

IPv4-Adressbereiche und Ports) ab, die als Muster der Regel verwendet werden.

5. Prüfe, ob das Muster der Regel sich mit einem bereits im Behälter liegenden Muster überschneidet[1]. Dabei wird für jede Überschneidung das alte Muster angepasst, ggf. in mehrere Teile zerlegt und den entsprechenden Behältern zugewiesen. Dabei bleiben die Beziehungen zwischen den Identifikatoren erhalten.
6. An diesem Punkt enthalten die Behälter alle auf deren Aktion zutreffenden Identifikatormengen. Dies entspricht bereits der Effektiven Konfiguration, die jetzt nur noch in OWL übersetzt wird. Abbildung 3.27 zeigt einen Ausschnitt dieser Repräsentation für einen Erlauben-Behälter (vgl. AllowedFlow). Dabei ist zu erkennen, dass die jeweiligen Protokollinformationen der Muster miteinander verbunden blieben (vgl. *usesFlow*). Der Ablehnen-Behälter erhält mit *DisallowedFlows* eine analoge Repräsentation.

Abbildung 3.27 zeigt außerdem die Verknüpfung mit einer beispielhaften OWL-Repräsentation der einzelnen Regeln, für den Fall, dass diese bereits Teil des Modells waren. In einem solchen Fall wird der Algorithmus auf der OWL-Repräsentation der Regeln ausgeführt und fügt den Mustern jeweils die Regeln, aus denen sie resultieren, hinzu, bevor sie den Behältern zugewiesen werden. Für den Algorithmus ist, mit Ausnahme des eben genannten Schrittes der Beibehaltung der Regel-Verknüpfung, nicht relevant, auf welcher Repräsentation der Regeln er angewendet wird. Das Parsen der Regeln ist eine Voraussetzung für den Algorithmus, jedoch kein Teil davon. Der entscheidende Vorteil ist jedoch, dass durch ein einmaliges Ableiten der Effektiven Konfiguration beliebige, NAC-Konfigurationssprachen-agnostische Modellerweiterungen und Analysen auf diesen Informationen angewendet werden können.

[1] Muster *A* und *B* überschneiden sich nur dann, wenn für jede, in beiden Protokollen vorkommende ISO/OSI-Schicht, dasselbe Protokoll definiert ist und sich die jeweiligen Protokoll-spezifischen Identifikatormengen überschneiden.

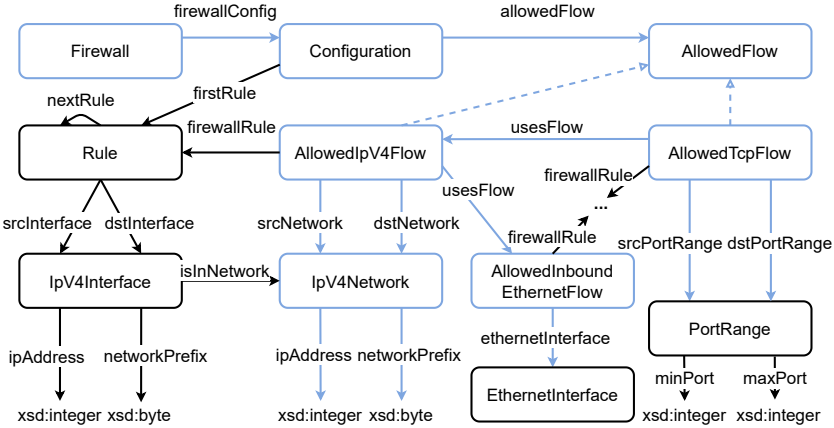


Abbildung 3.27: Ausschnitt einer Repräsentation für einen Erlauben-Behälter als Teil der Effektiven Konfiguration einer Firewall. Blau markierte Konzepte gehören dabei zur Effektiven Konfiguration, schwarz dargestellte Konzepte zur Repräsentation der Firewall-Konfiguration.

Nimmt man an, dass ein Modellerweiterungs- oder Analyseschritt die Informationen von $n \in \mathbb{N}$ verschiedenen NAC-Instanzen verwendet und insgesamt $k \in \mathbb{N}$ mögliche NAC-Konfigurationssprachen abgedeckt werden müssen, werden im schlimmsten Fall k^n Versionen dieses Schrittes benötigt. Im Vergleich dazu, muss ein auf der Effektiven Konfiguration arbeitender Schritt lediglich in einer Version vorliegen und benötigt die k unterschiedlichen Ableitungen der Effektiven Konfiguration. Unter der vereinfachten Annahme, dass n für alle Schritte gilt und der Aufwand eines Ansatzes mit der Gesamtanzahl seiner Schritte abgeschätzt werden kann, so liegt der Aufwand einer Verarbeitungspipeline mit $s \in \mathbb{N}$ Schritten im schlimmsten Fall in $O(s * k^n)$ ohne und $O(k + s * n)$ mit der Effektiven Konfiguration.
Dabei ist darauf zu achten, dass durchaus Schritte existieren können, die zwar Informationen von n unterschiedlichen NAC-Instanzen verwenden, aber die Art der Informationen sich nicht unterscheidet. In einem solchen besten Fall würden auch ohne Effektive Konfiguration lediglich k Vorberechnungen dieser Informationsart und eine Version des Schrittes benötigt. So lässt sich also für

$i \in \mathbb{N}$ Informationsarten der konstante Aufwand $O(k)$ für die Erzeugung der Effektiven Konfiguration, dem polynomiellen Aufwand $O(k * i)$ ohne Effektive Konfiguration gegenüberstellen.
Allerdings ist ebenfalls zu beachten, dass dies nicht nur einem Ressourcenaufwand entspricht, sondern einem manuellen Programmieraufwand. Dies verdeutlicht, wie viel besser die Effektive Konfiguration gegenüber der Regelkettenauswertung in Modellverarbeitungs- und Analyseschritten selbst im besten Fall skaliert.

In *[Pat21]* wurde der Ansatz der Effektiven Konfiguration nicht nur mit der Regelkettenauswertung in Modellverarbeitungs- und Analyseschritten verglichen, sondern mit den folgenden Klassen verwandter Ansätze:

- *Ansatz 1* - Modellierung von Richtlinien in einer abstrakten Konfigurationssprache, die zur Auswertung der Regeln und zur automatischen Generierung von NAC-spezifischen Regeln verwendet werden kann [Mar07, Dav13].
- *Ansatz 2* - Automatisches Übersetzen von NAC-spezifischen Regeln in eine abstrakte Konfigurationssprache, die zur Auswertung der Regeln verwendet werden kann [Hu11, Ton15].
- *Ansatz 3* - Modellierung jeder NAC-Konfiguration mit sprachabhängigen Repräsentationen und Suche nach Problemen auf der Regelketten-Ebene [Fit07, Fol08, Cor18].
- *Ansatz 4* - Manuelles Modellieren der Beziehungen zwischen den Regeln und Suche nach Problemen auf der so erstellten abstrakten Beziehung [Fit08, Fit09].

Eine Untersuchung der aufgeführten Veröffentlichungen zu diesen Ansätzen führte zu der in Tabelle 3.8 zu sehenden Bewertung. Dabei wurden die durch die Ansätze verfolgten Ziele in sechs Aspekte gegliedert und auf einer Skala von 0-4 bewertet, wobei 4 die höchste und 0 die geringste Abdeckung des Aspektes beschreibt. Zusätzlich zu den beschriebenen Ansätzen, wurde auch das Konzept der Effektiven Konfiguration bewertet, um einen Vergleich zu ermöglichen.

Tabelle 3.8: Vergleich verwandter Ansätze mit dem Einsatz der Effektiven Konfiguration.

	Ans. 1	Ans. 2	Ans. 3	Ans. 4	Eff. Konf.
Auffinden von Konfigurationsproblemen in Regelketten, wie überdeckte, redundante oder generalisierte Regeln	1	3	4	0	(4)
Auffinden von NAC-übergreifenden Problemen, wie Inkonsistenzen bezüglich bestimmter Richtlinien	3	2	0	1	4
Finden von netzwerkzentrierten Problemen, die nicht auf NAC-Regeln fokussiert sind	2	2	0	3	4
Bereitstellen einer herstellerunabhängigen NAC-Konfigurationsdarstellung	4	3	0	0	2
Bereitstellen einer Darstellung, die alle NAC-Konfigurationssprachen abdeckt	2	2	0	0	3
Bereitstellen einer vollautomatischen Modellerstellungsstrategie	2	2	2	0	3

Dabei ist deutlich zu erkennen, dass die Effektive Konfiguration die aufgeführten Aspekte von allen Alternativen am besten abdeckt. Jedoch bedarf die Bewertung der Effektiven Konfiguration für den ersten Aspekt einer kurzen Erläuterung. Diese ist in Klammern gesetzt und entspricht in ihrer Höhe Ansatz 3, also dem Modellieren von NAC-Konfigurationen als Konfigurationssprachenspezifische Repräsentation der Regelketten. Dies kommt daher, dass die Effektive Konfiguration dies nicht explizit adressiert, jedoch, durch die Verknüpfung mit einer Konfigurationssprachen-spezifischen Repräsentation, Ansatz 3

vollumfänglich unterstützt. Die Effektive Konfiguration bringt bei Intra-NAC-Konfigurationsanalysen also keinen Vorteil, behindert sie jedoch auch nicht. Für weitere Diskussion dieser Ergebnisse wird auf *[Pat21]* verwiesen.

3.4.2.2 Manuelle Erweiterung

Wie durch Anforderung MA9 „Unterstützung von Nutzerinteraktion bei Modellbildung und -verarbeitung“ aus Abschnitt 3.2.3 motiviert, gibt es Wissen, das zunächst einmal manuell in maschinenlesbare Form gebracht werden muss. Als Beispiel soll hier das Zonen- und Kanäle-Konzept des IEC 62443 dienen, dessen Anwendung in der Laborumgebung bereits beispielhaft gezeigt wurde (vgl. Abschnitt 3.1.2). Die Architektur der Laborumgebung ist in Abbildung 3.28 nochmalig dargestellt. Dort ist zu erkennen, wie diese Umgebung in Zonen und Kanäle eingeteilt werden kann. Diese beschreiben Bereiche – und damit Asset-Mengen – für die dediziert Sicherheitsniveaus festgelegt werden. Das Beispiel ist sehr einfach gehalten, sodass die Netzwerksegmente je einer Zone und jeder Übergang zwischen zwei Segmenten einem Kanal zugeordnet sind. Diese Zuordnung ist Teil der Sicherheitsstrategie des jeweiligen Unternehmens und liegt für gewöhnlich nicht maschinenlesbar vor.
Nun ist es möglich, bestimmte Regeln zu definieren, die Bedingungen formulieren, welche gelten müssen, damit ein Asset einer bestimmten Zone oder einem bestimmten Kanal zugeordnet wird. Diese könnten vorab definiert und als Modellerweiterungsschritt eingebracht werden. Im Beispiel könnte man die IPv4-Adressen als Bedingungen verwenden. Diese Bedingung wird jedoch schnell zum Problem, wenn die Adressen, wie im Beispiel von Abschnitt 3.4.2 mehrfach und insbesondere in unterschiedlichen Zonen verwendet werden. Man findet also leicht Beispiele, in denen Informationen einem Modell manuell, wenn auch durch Werkzeugunterstützung, hinzugefügt werden müssen.
Ein solches Werkzeug wurde beispielsweise im Rahmen der Masterarbeit von Roehr konzipiert *[Roe21]*. Es handelt sich um eine Applikation zur Unterstützung für die Integration manueller Modellerweiterung für beliebige Anwendungsdomänen, wie IEC 62443 und ISO 27000. Ein Überblick über dieses Werkzeugkonzept wird in Abbildung 3.29 gegeben.

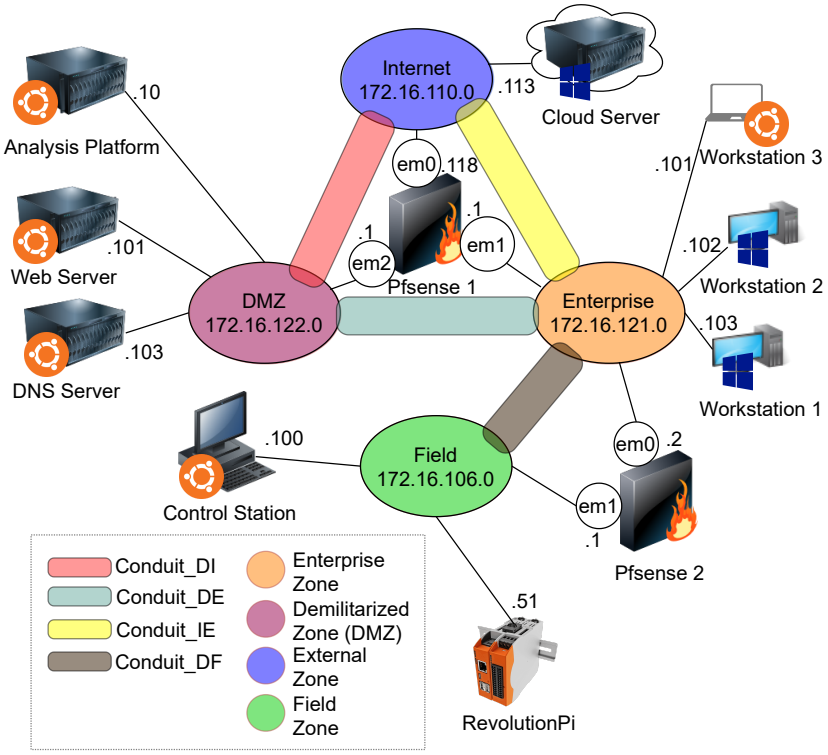


Abbildung 3.28: Netzwerkarchitektur des Laborbeispiels mit Zonen und Kanälen nach IEC 62443.

Je Anwendungsdomäne wird dabei eine Perspektive des Werkzeugs unterstützt, welche ihre Modellierungskomponenten an eine Ontologie koppelt, die als Werkzeug-Ontologie gesehen werden kann. Durch Laden einer Abbildungsontologie können so beliebige Systemontologien unterstützt werden, wodurch das Werkzeug die hier angestrebten Flexibilitätsanforderungen erfüllen kann.

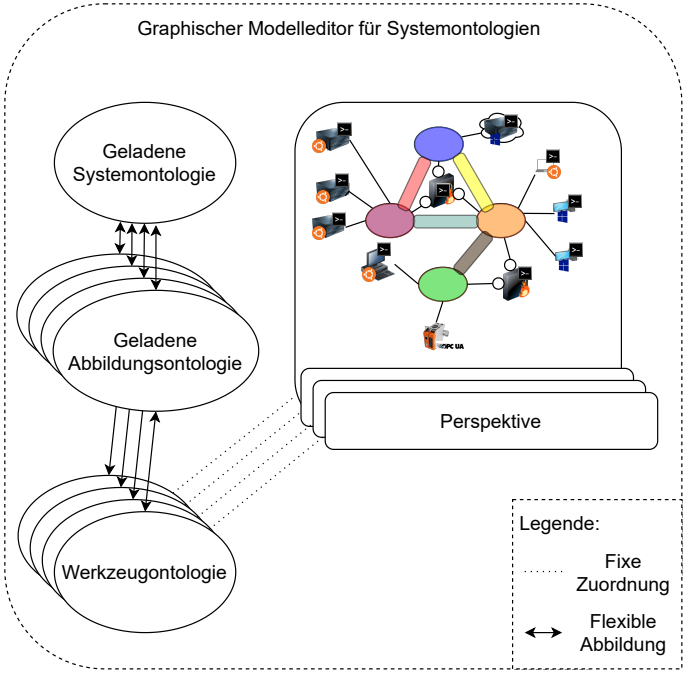


Abbildung 3.29: Überblick über die Flexibilitätsaspekte des Grafischen Modelleditors für Systemontologien.

3.4.3 Sicherheitsanalyse

Wurde ein Systemmodell erzeugt und entsprechend vorverarbeitet, können darauf Sicherheitsanalysen ausgeführt werden. Um dabei die gewünschten Synergien zwischen den Analysen und gleichzeitig die angestrebte Flexibilität und Wiederverwendbarkeit von Analysen und Analyseschritten bei gleichzeitiger Maximierung der Automatisierung zu gewährleisten, wurde im Rahmen dieser Dissertation ein Ansatz zur generischen Sicherheitsanalyse entwickelt. Dieser wird in den beiden folgenden Abschnitten vorgestellt.

3.4.3.1 Regelbasierte Analyse als Konvergenz der Analysearten

Durch die Untersuchung der verschiedenen Ansätze zu den hier behandelten Analyseverfahren konnte festgestellt werden, dass diese Verfahren mithilfe eines generischen, regelbasierten Analysekonzepts abgedeckt werden können. Das entsprechende Konzept wird in den folgenden Paragraphen vorgestellt.
Abbildung 3.30 zeigt einen Überblick des generischen, regelbasierten Analysekonzepts. Dieses ist in die Sichten von Sicherheits- und Modellierungsexperten unterteilt. Die verschiedenen und für die Analysen zugrundeliegenden sicherheitsrelevanten Formulierungen werden dabei als *technische Richtlinien* interpretiert und in *Analyseregeln* umgewandelt. Analyseregeln bestehen hierbei aus *Vorbedingungen*, die an die Modellgenerierung und -erweiterung gestellt werden, sowie *Abfragen*, mit denen die entsprechenden Analyseergebnisse extrahiert werden.
Technische Richtlinien werden meist textuell in Dokumenten festgelegt, können aber auch als *formale Regeln* mittels formalisierter Regelsprachen wie OrBAC [Kal03], Ponder [Dam01] oder Rei [Kag02] definiert werden. Da diese technischen Richtlinien unterschiedliche Abstraktionsgrade besitzen können, enthält die manuelle Transformation zu formalen Regeln oder Analyseregeln meist einen gewissen Interpretationsspielraum. Generell gibt es zwei Ausprägungsextrema von technischen Richtlinien, die hier mit *offene Richtlinien* und *geschlossene Richtlinien* bezeichnet werden sollen.
Offene Richtlinien weisen eine Definitionslücke zwischen der Richtlinie und der entsprechenden technischen Umsetzung auf. Ein Beispiel hierfür sind IEC62443-Richtlinien, deren konkrete Umsetzungen nicht vorgegeben sind und nur durch Best-Practices, individuelle Strategien und eventuelle, weitere Interpretationszwischenstufen (wie Gegenmaßnahmen aus dem ICS-ATT&CK®-Rahmenwerk, vgl. Abschnitt 3.7.3) definiert werden können. Offene Richtlinien sind also beliebig abstrakt und erfordern für deren Überprüfung entsprechend viel explizit modelliertes Wissen und intelligente Strategien zur Abstraktion des Wissens und/oder zur Detaillierung der Richtlinien verschiedener Abstraktionsebenen.

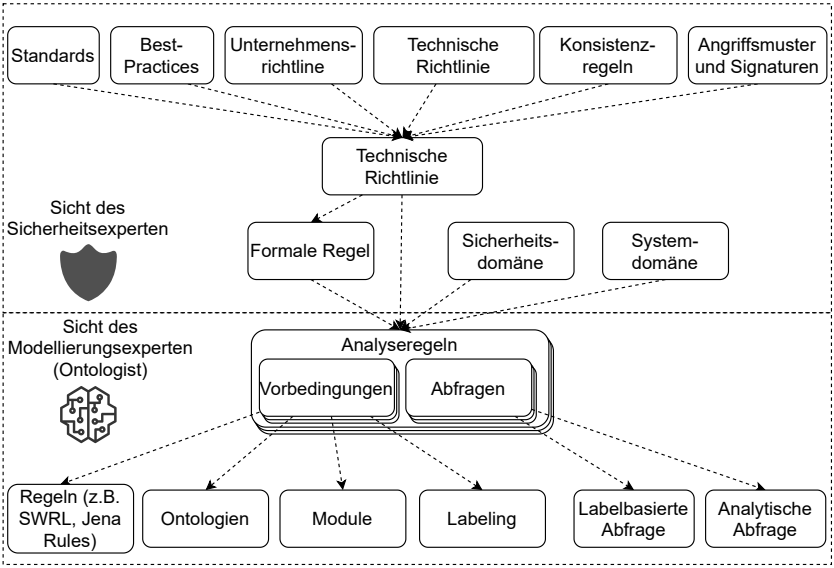


Abbildung 3.30: Abbildung des generischen, regelbasierten Analysekonzepts.

Geschlossene Richtlinien definieren direkt prüfbare Vorgaben, wie die Anforderung auf allen Plattformen, die dies unterstützen, ein Host-IDS zu installieren oder das Verlangen der Definition bestimmter Zugriffskontrollregeln.

Tabelle 3.9: Beispielrichtlinien aus NIST 800-82, ICS ATT&CK®, IEC 62443, sowie aus Konfigurations- und Schwachstellenanalyse.

Identifikator	Kernaussage (sinngemäß)
IEC 62443-3-3 SR 3.1	Das ICS soll in der Lage sein, die Integrität übertragener Informationen zu schützen
IEC 62443-3-2 ZCR-3.2	Trenne Unternehmens- und ICS-Assets
ICS ATT&CK®M1030	Nutze physische oder logische Segmentierung, um Zugriff auf kritische Systeme, Funktionen und Ressourcen einzuschränken

weiter auf der nächsten Seite

Identifikator	Kernaussage (sinngemäß)
ICS ATT&CK®M1037	Filtere eingehenden und ausgehenden Netzwerkverkehr des ICS-Netzwerks
NIST 800-82 5.3.4	Implementiere eine Firewall mit DMZ zwischen Unternehmens- und ICS-Netzwerk
NIST 800-82 5.3.1	Kein Gerät, außer eine Firewall, darf sowohl im Unternehmens- als auch im Steuernetzwerk hängen
NIST 800-82 5.5 1	Default-Firewallregeln sollten alles verbieten, nichts erlauben
NIST 800-82 5.5 7	Ausgehender ICS-Verkehr zum Unternehmensnetzwerk sollte Quell- und Ziel-beschränkt sein, inkl. Dienst und Port
Inter-NAC-Konflikt 1	Eine NAC-Regel darf einer anderen nicht widersprechen, falls diese zu zwei unterschiedlichen NAC-Instanzen gehören und die Treffermenge identisch ist
Inter-NAC-Konflikt 2	NAC-Regel 1 darf NAC-Regel 2 nicht generalisieren[1], falls beide Regeln eingehenden ICS-Verkehr adressieren und der Verkehr zuerst NAC 2 und dann NAC 1 passiert[2]
CPE-CVE-Suche	Eine Entität mit einer bestimmten CPE[3] assoziiert mit einer CVE-Schwachstellen-beschreibung[4] soll im System nicht existieren

weiter auf der nächsten Seite

[1] Regel 1 generalisiert Regel 2, wenn die Treffermenge von Regel 1 größer ist als die von Regel 2 und beide Regeln die selbe Aktion (z.B. zulassen oder blockieren) definieren.

[2] Eine solche Konfiguration bedeutet beispielsweise, dass eine Netzwerk-basierte Firewall bestimmten eingehenden Verkehr in ein ICS-Netzwerksegment verbietet, eine Host-Firewall eines Assets in diesem Netz den selben Verkehr erlaubt. In der Realität tritt diese Konfigurationssituation häufig ein, zeugt jedoch davon, dass in dem Beispiel die Host-Firewall nicht nur den Verkehr zulässt, der wirklich benötigt wird.

[3] Zur Erinnerung: CPE ist eine Spezifikation für Identifikatoren von Soft- und Hardware.

[4] Zur Erinnerung: CVE ist ein spezifiziertes Beschreibungsformat von Schwachstellen, mit denen betroffene Entitäten unter anderem über CPE-Identifikatoren assoziiert werden.

Identifikator	Kernaussage (sinngemäß)
Man-in-the-Middle	Die Beobachtung einer neuen Ethernetadresse zu einer bekannten IP-Adresse und darauffolgende Kommunikation zwischen der IP-Adresse und jener Ethernetadresse welcher die IP-Adresse zuvor zugeordnet war, sollte nicht erlaubt sein

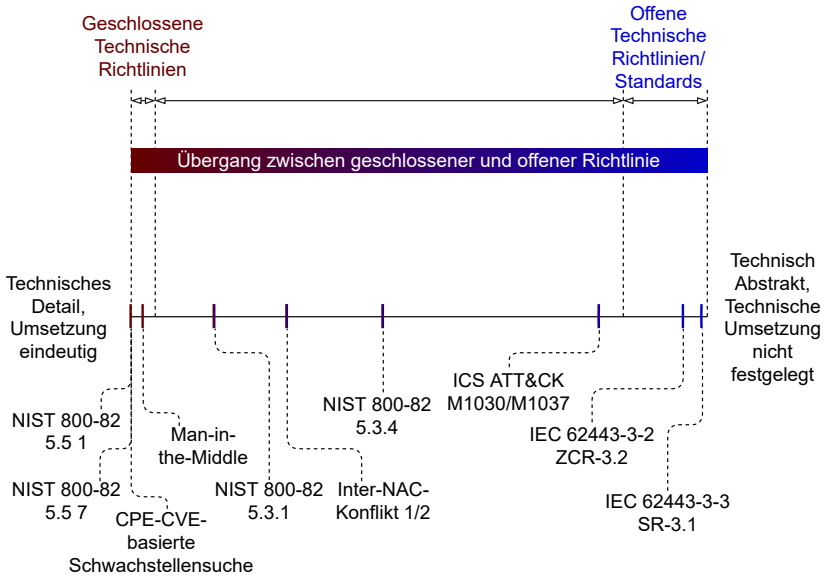


Abbildung 3.31: Einordnung der Beispielrichtlinien aus Tabelle 3.9 in die Dimension zwischen geschlossener und offener Richtlinie.

In Tabelle 3.9 werden weitere Beispiele für Richtlinien aus den in dieser Arbeit behandelten Analysebereichen gegeben, welche in Abbildung 3.31 in die Dimension eingeordnet wurden, welche durch den Übergang zwischen geschlossenen und offenen Richtlinien aufgespannt wird. Industrielle Systeme

werden in diesem Beispiel zugunsten der Übersichtlichkeit durch die populäre Abkürzung ICS vertreten.

Werden die technischen Richtlinien zunächst in Regelsprachen transformiert, dienen die entstehenden formalen Regeln als maschinenlesbare Zwischenrepräsentation. Wie der folgende Abschnitt genauer erläutert, kann wiederum eine Transformation pro Regelsprache entwickelt werden, welche die Zwischenrepräsentationen in Analyseregeln übersetzt.

3.4.3.2 Das Konzept der Analyseregeln

Der Aufbau von Analyseregeln, wie er in Abbildung 3.30 bereits angedeutet ist, wurde in einer OWL-Ontologie formalisiert. Ein Ausschnitt dieser OWL-Ontologie ist in Definition 3.1 zu sehen. Eine Analyseregel (vgl. Policy) besteht aus Vorbedingungen (vgl. Requirement), die an das Modell gestellt werden und Abfragen (vgl. Query), die zur Extraktion von abgeleitetem Wissen dienen (vgl. Zeile 3.1a). Wie Zeile 3.1b definiert, besitzen Abfragen entweder eine primitive (vgl. LabelQuery) oder analytische Ausprägung (vgl. AnalyticQuery).

$$\text{Policy} \sqsubseteq \exists \mathit{requires}.\text{Requirement} \sqcup \exists \mathit{analysedBy}.\text{Query} \tag{3.1a}$$

$$\text{AnalyticQuery} \sqcup \text{LabelQuery} \sqsubseteq \text{Query} \tag{3.1b}$$

$$\text{Implementation} \sqcup \text{OntologyDependency} \sqsubseteq \exists \mathit{meets}.\text{Requirement} \tag{3.1c}$$

$$\top \sqsubseteq \forall \mathit{dependsOn}.\{\text{Query} \sqcup \text{Implementation} \sqcup \text{OntologyDependency}\} \tag{3.1d}$$

$$\text{Query} \sqcup \text{Implementation} \sqcup \text{OntologyDependency} \sqsubseteq \exists \mathit{dependsOn}.\top \tag{3.1e}$$

$$\text{Rule} \sqcup \text{ProcessingModule} \sqsubseteq \text{Implementation} \tag{3.1f}$$

$$\text{JenaRule} \sqcup \text{SwrlRule} \sqsubseteq \text{Rule} \tag{3.1g}$$

Abfragen primitiver Ausprägung sind solche, die gelabelte Informationen extrahieren und somit, im Gegensatz zu analytischen Abfragen, keine eigene Analyselogik implementieren. Analytische Abfragen werden beispielsweise eingesetzt, um Konfigurationen während der Analyse zu interpretieren, oder

wenn die Analyse keine Änderung des Modells vornehmen soll.
Wie in den vorangegangenen Abschnitten dieses Kapitels bereits verdeutlicht wurde, beziehen sich Analysen auf verschiedene, ggf. voneinander abhängige Domänen, die durch Ontologien (vgl. OntologyDependency) repräsentiert werden. Weitere Vorbedingungen für die Analysen sind die untereinander abhängigen Modellverarbeitungsschritte (vgl. Implementation). Verarbeitungsschritte und im Systemmodell verwendete Ontologien erfüllen (vgl. *meets*) somit die Anforderungen einer Analyse (vgl. Zeile 3.1c). All diese Abhängigkeiten werden zwischen den Individuen der Klassen Query, Implementation und OntologyDependency definiert (vgl. Zeilen 3.1d und 3.1e). So entstehen zwei Deklarations- und Abhängigkeitsebenen, die in Abbildung 3.32 zu sehen sind. Dabei können Analyseregelformulierung und -umsetzung unterschieden werden. Dies ermöglicht es, verschiedene Umsetzungen für eine Analyseregel zu definieren (z.B. abhängig von Domänen und Strategien), Umsetzungsbausteine für die Zusammensetzung verschiedener Umsetzungen beliebig zusammenzusetzen und diese entsprechend wiederzuverwenden. Zudem wird die Klasse Implementation noch in Rule und ProcessingModule unterschieden (vgl. Zeile 3.1f), wobei ersteres aus Regelklassen besteht (hier SwrlRule und JenaRule, vgl. Zeile 3.1g) und letzteres die Individualprogramme klassifiziert. Dieser Mechanismus führt auch zur klaren Trennbarkeit von monotonen und nicht-monotonen Umsetzungen.

Der Grund für diese Formalisierung ist die Verwendbarkeit für verschiedene maschinengestützte Verfahren, sowohl zum Erstellen der Analyseregel als auch zu deren Bereitstellung und Verwendung in einem Analysesystem. Diese Intension wird in Abschnitt 3.5.4 klar, wenn die automatisierte Bereitstellung von in einer Community-Lösung erzeugten Regeln vorgestellt wird. Zudem können durch die Verwendung der Formalisierung Vorschlagsysteme eingesetzt werden, welche beispielsweise die Verwaltung der Beziehungen zwischen Formulierungen und ihren Umsetzungen unterstützen. Außerdem ist die Formalisierung leicht erweiterbar, ohne Konsistenz einzubüßen. So können beispielsweise weitere Regeltypen hinzugefügt oder Individualprogramme weiter unterteilt werden. Dies ermöglicht eine leichte Anpassung für beliebige Anwendungen.

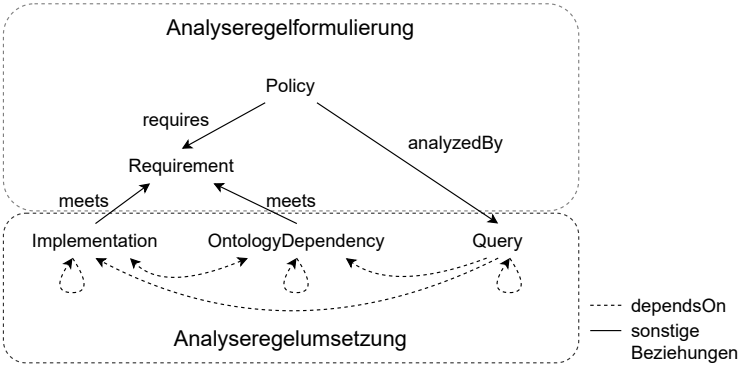


Abbildung 3.32: Deklarationsebenen der Ontologie für Analyseregeln.

Darüber hinaus hat die Formalisierung auch weitere Vorteile. So zeigten Lóde Vergara et al. bereits 2008 eine Transformation der formalen Regelsprache OrBAC nach OWL und SWRL [Ver08]. Hierfür entwickelten sie eine OrBAC-Ontologie und eine automatisierte Übersetzung von OrBAC-Regeln in SWRL-Regeln. Diese automatisierte Übersetzung kann in dem in Abbildung 3.30 gezeigten Konzept für die Abbildung von formalen Regeln zu Implementierungen der Analyseregeln eingesetzt werden, indem die OrBAC Ontologie als Ontologie-Vorbedingung und die SWRL-Regeln als Regel-Vorbedingungen definiert werden. Lediglich eine Abfrage der jeweiligen Analyseergebnisse muss dann noch erfolgen. Eine formale Regelsprache (in diesem Fall OrBAC) kann damit also, wie im vorherigen Abschnitt erwähnt, in eine Analyseregeldefinition transformiert werden.

3.5 Rahmenwerk zur ontologiebasierten Systemanalyse

Die Umsetzung der beschriebenen Einzelkonzepte und Schlussfolgerungen aus Abschnitt 3.4 erfordert ein konsistentes Gesamtkonzept, mit entsprechender

Architektur und Methodik. Dieses wird in dem im Folgenden vorgestellten *System Model Processing* (*SyMP*) Rahmenwerk definiert.

SyMP soll Szenarien, wie das in Abschnitt 3.1.3 vorgestellte Anwendungsszenario, methodisch und mit einem konsistenten Gesamtkonzept ermöglichen. Abbildung 3.33 zeigt das Kernkonzept von SyMP, welches die Modellverarbeitungsumgebung zur Umsetzung von Informationsextraktion, Modellbildung, Modellverarbeitung und Analyse definiert. Der Grundgedanke ist dabei die klare Trennung sowie sinnvolle Wiederverwendbarkeit und Austauschbarkeit von Expertise und Arbeitsschritten, wodurch auch verschiedene Akteure explizit unterstützt werden.
Wie in der Abbildung zu sehen, wird die Umgebung in system- (vgl. *System Domain*), anwendungs- (vgl. *Application Domain*) und analysespezifische (vgl. *Analysis*) Bereiche unterteilt. Der systemspezifische Bereich umfasst alle Konzepte zur Informationsextraktion, Modellbildung und Modellverarbeitung, wobei letzteres auf Schritte zur Integration der Komponentenmodelle (vgl. Abschnitt 3.4.2), Bereinigung des resultierenden Modells und dessen Erweiterung beschränkt ist. Der anwendungsspezifische Bereich enthält alle Konzepte zur Abbildung auf die Domäne, für die eine Analyse durchgeführt wird, sowie zur anwendungsdomänenspezifischen Erweiterung des Modells. Er adressiert damit also insbesondere die in Abschnitt 3.4.1.3.2 vorgestellt Sicherheitsdomäne. Zuletzt enthält der analysespezifische Bereich nur Konzepte, welche für die eigentlichen Analysen relevant sind, also die Analysedomäne.

Wie in Abbildung 3.33 zu sehen ist, wird die Modellverarbeitungsumgebung in SyMP in sechs Phasen unterteilt, von denen jede Phase die von der Vorgängerphase erzeugten Modelle als Eingabe verwendet (mit Ausnahme der ersten Phase). Die Phasen werden im Folgenden beschrieben.

Phase 1 - Knowledge Collection *(Wissenserfassung)*

In dieser Phase werden Informationen aus einer beliebigen Anzahl von Quellen extrahiert. Diese Informationen können bereits unter Verwendung einer Zielsyntax serialisiert sein und die System-TBox verwenden. Wenn dies jedoch nicht der Fall ist, sind die erforderlichen Transformationsaufgaben (vgl.

Abschnitt 3.4.1.2) auch Teil dieser Phase. Das Ergebnis dieser Phase sind *Komponentenmodelle* (engl. *Component Models*).

Phase 2* - Knowledge Fusion *(Wissenszusammenführung)

In dieser Phase wird die Komposition von Komponentenmodellen zu einem einzigen Systemmodell durchgeführt. Beispiele für Aufgaben, die in dieser Phase durchgeführt werden, sind die Umbenennung von Individuen, die Zusammenführung von äquivalenten Netzwerkdarstellungen und die Integration in ein Modell (vgl. Abschnitt 3.4.2). Das Ergebnis dieser Phase ist ein *Systemmodell* (engl. *System Model*).

Phase 3* - Model Cleaning *(Modellbereinigung)

Die automatische Verschmelzung von Komponentenmodellen führt in der Regel zu einem Modell, das Aspekte wie Redundanzen enthält (z.B. redundante Darstellungen von Netzwerkdiensten oder Softwarepaketen, vgl. Abschnitt 3.4.2). Aufgaben, die diesen Problemen entgegenwirken, sind Teil dieser Phase. Das Ergebnis dieser Phase ist das *bereinigte Systemmodell* (engl. *Cleaned System Model*).

Phase 4* - Static Knowledge Extension *(Statische Wissenserweiterung)

In dieser Phase wird das bereinigte Systemmodell mit Wissen über eine Anwendungsdomäne erweitert. Häufige Aufgaben für diese Phase sind das Reasoning zur Vervollständigung des Wissens über eine Anwendungsdomäne, die Benutzerinteraktion zur Erweiterung um organisationsspezifisches Wissen (z.B. Abbildung von Sicherheitszonen auf Netzwerksegmente) und die Interpretation der Konfiguration (z.B. Firewall-Regelketten). Änderungen am Systemmodell, die in dieser Phase durchgeführt werden, zielen auf alle Anwendungen innerhalb dieser Domäne ab und müssen daher von keiner Anwendung (bzw. Analysemaschine) selbst durchgeführt werden. Das Ergebnis dieser Phase ist

das *anwendungsdomänenspezifische Systemmodell* (engl. *Application-Specific System Model*).

Phase 5 - Dynamic Knowledge Extension *(Dynamische Wissenserweiterung)*

Diese Phase besteht aus Aufgaben, die für eine spezifische Analyse durchgeführt werden, wodurch das anwendungsdomänenspezifische Systemmodell für die eigentliche Analyselogik vorbereitet wird. Sie kann als die Phase interpretiert werden, in der spezielle Analyseanforderungen an das Modell erfüllt werden. Ein Beispiel wäre die Ableitung von transitiven *tcpConnected*-Rollenzuweisungen, die darauf hinweisen, dass ein Gerät über TCP mit einem bestimmten zweiten Gerät verbunden werden kann, was bei einer Analyse, die nach potenziellen TCP-Verbindungen vom Feld zum Unternehmensnetzwerk sucht, erforderlich sein könnte. Das Ergebnis dieser Phase ist das *analysespezifische Systemmodell* (engl. *Analysis-Specific System Model*).

Phase 6 - Analysis *(Analyse)*

In dieser Phase wird die Analyselogik auf dem analysespezifischen Systemmodell ausgeführt, welche nicht Teil einer Analyseabfrage ist. Eine übliche Technik in der ontologiebasierten Sicherheitsanalyse ist die Kennzeichnung von Entitäten im Modell, um sie in einem späteren Schritt abzufragen (vgl. z.B. [Fen18]). Beispielsweise könnten Firewalls als Whitelist- und Blacklist-Firewalls gekennzeichnet werden, was aus deren Standardregel (alles verbieten oder alles erlauben) ableitbar ist. Dies führt zu einem Modell, in welchem abgefragt werden kann, ob unerwünschte Blacklist-Firewalls im System existieren. Die eigentliche Analyselogik wird somit also von dieser Phase übernommen und das Modell muss nur noch von primitiven Abfragen ausgewertet werden. Wie in Abschnitt 3.4.3.2 beschrieben, ist die Alternative hierzu, zusätzliche Logik in die Abfrage selbst zu codieren. Auch diese komplexen Abfragen können einen Teil der Logik in dieser Phase vorberechnen lassen. Das Ergebnis dieser Phase ist ein *vorberechnetes, analysespezifisches Systemmodell.*

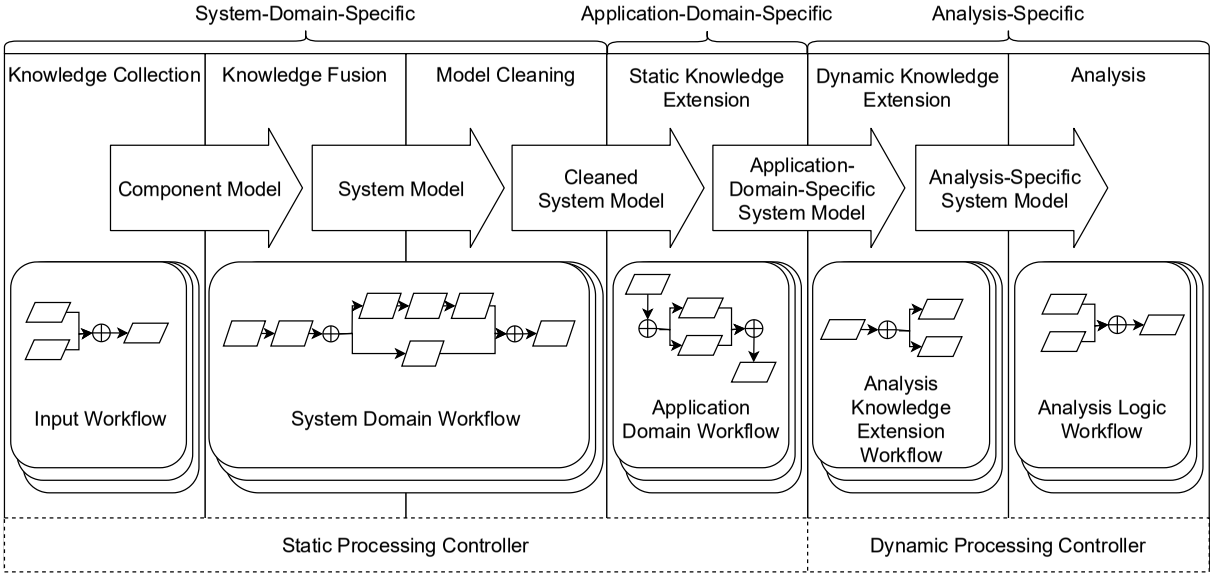


Abbildung 3.33: Konzept der Modellverarbeitungsumgebung aus SyMP.

In Abbildung 3.33 ist zudem eine Steuerung für statische Verarbeitung (vgl. *Static Processing Controller* (*SPC*)) und eine Steuerung für dynamische Verarbeitung (vgl. *Dynamic Processing Controller* (*DPC*)) zu sehen. Die ersten vier Phasen erzeugen Modellinstanzen, die von mehreren (Analyse-)Anwendungen verwendet werden können und daher im aktuellen Systemzustand „stabil" sind. Daher wird diese Art der Verarbeitung hier als „statisch" bezeichnet und durch den SPC gesteuert. Analog dazu erzeugt die dynamische Verarbeitung (gesteuert vom DPC) anwendungsspezifische Modelle. Denn diese ändern sich mit dem Zustand der Anwendung und nicht mit dem Zustand des Systems, was zu ihrer Klassifizierung als dynamisch führt. Aus Sicht der Analyse umfasst die statische Verarbeitung die Vorverarbeitungsaufgaben, die für die meisten Analysen als relevant angesehen werden. Diese Aufgaben sind daher von keiner Analysevoraussetzung abhängig. Im Gegensatz dazu wird die dynamische Verarbeitung nur dann durchgeführt, wenn sie von einer oder für eine bestimmte Analyse angefordert wird. Mehr zu diesen Steuerungskomponenten wird in Abschnitt 3.5.2 vorgestellt.

Bis zu dieser Stelle der Rahmenwerksdefinition können die unterschiedlichen, modellverändernden Aufgaben, Phasen zugeordnet werden, die wiederum definieren, welche Ergebnisse der Verarbeitung aus jeder Phase erwartet werden können, sowie in welcher Reihenfolge die Ausführung stattfindet. In den folgenden Abschnitten werden die Konzepte erläutert, welche die Planung, die automatisierte Ausführung, die Austauschbarkeit der Verarbeitungsstrategien und mehr unterstützen.

3.5.1 Workflows

Innerhalb des SyMP-Rahmenwerks werden die Verarbeitungspipelines durch Workflows (vgl. Abschnitt 2.8) dargestellt. Manche Phasen besitzen eigene Workflows, andere teilen sich einen Workflow (vgl. Abbildung 3.33). Workflows können als gerichtete Graphen modelliert werden, wobei die Aufgaben Knoten sind und die Ausführungsreihenfolge durch die Kanten dargestellt wird. Darüber hinaus kann, ähnlich wie bei einem UML-Aktivitätsdiagramm, die

Parallelisierung mit Gabelungs- und Synchronisationsknoten modelliert werden. Jeder der Knoten repräsentiert einen Modellverarbeitungsschritt. SyMP definiert folgende Basis-Verarbeitungsschritte: Ontologien laden, Ontologien zusammenführen, prozedurales Programm (vgl. Processing Module aus Formel 3.1) ausführen, Reasoning, Regelmaschine ausführen und Benutzerinteraktion durchführen (vgl. Abschnitt 3.4.2.2).

Durch die Einstufung der Phasen als systemdomänen-, anwendungs- und analysespezifisch wird definiert, welche Art von Domänen, und somit Ontologien, in welcher Phase angewendet wird. Aufgaben innerhalb einer Domäne arbeiten auf den gleichen Konzepten. Folglich werden die Workflows auf Domänenbasis modelliert und angewendet. Eine Ausnahme von dieser Verallgemeinerung bilden jedoch die Eingabe-Workflows (engl. *Input Workflows*), da diese ebenfalls von der Daten- bzw. Informationsquelle abhängen. Die Beziehungen zwischen Workflows und Domänen sind durch das Diagramm in Abbildung 3.34 dargestellt.

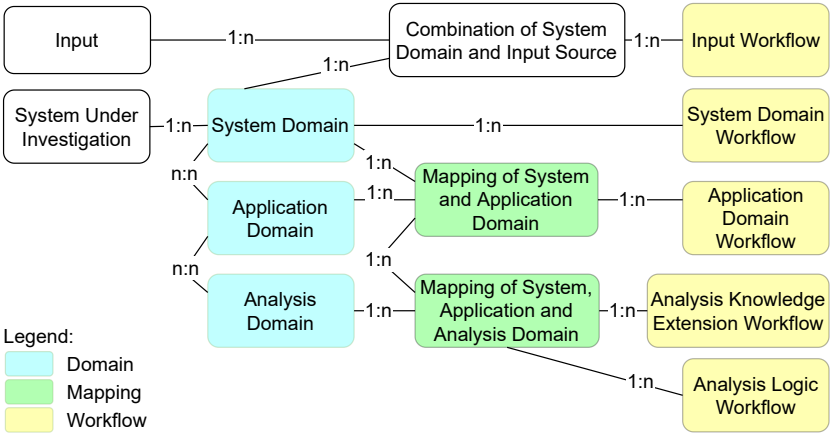


Abbildung 3.34: Beziehung zwischen Workflows, Domänen und Abbildungen.

Das Diagramm zeigt unter anderem, dass die Domänen voneinander unabhängig und für die Phasen 4-6 Abbildungen erforderlich sind. Eine logische Konsequenz ist die Zugehörigkeit jedes Workflows in den Phasen 4-6 zu genau einer Abbildung. Diese Abbildungen werden mittels Abbildungsontologien realisiert, da die Domänen durch Ontologien repräsentiert werden. Für die Abbildungsontologien gilt wiederum, dass sie aus weiteren Abbildungsontologien bestehen können (vgl. Abschnitt 3.4.1.3.1). Darüber hinaus zeigt das Diagramm, dass jeder Workflow mit genau einer Systemdomäne verbunden ist und dass jeder Workflow Alternativen haben kann.

3.5.2 Steuerung der Phasen

Zur Gewährleistung der Konsistenz von SyMP werden noch weitere Konzepte und Anforderungen für die Steuerkomponenten SPC und DPC definiert.
Zunächst müssen SPC und DPC die aktive Systemdomäne kennen, z.B. durch manuelle Konfiguration.
Zudem kann die Phase der Wissenserfassung entweder durch die Quelle (Push-Modus, z.B. weil sich die Konfiguration der Quelle geändert hat) oder die Verarbeitungsumgebung (Pull-Modus, z.B. weil sie manuell oder nach einem Zeitplan aufgerufen wurde) eingeleitet werden. Daher muss der SPC eine Schnittstelle zur Quelle anbieten, die im Falle des Push-Modus den richtigen Workflow für die Verbindung von Quelle und Änderungsart auswählen muss. Die Änderungsart ist dabei eine Klassifizierung wie „Erzeugen“, „Einfügen“ oder „Aktualisieren“, die in der Regel unterschiedliche Aufgaben zum Aufbau des jeweiligen Komponentenmodells erfordert. Darüber hinaus muss die Schnittstelle im Falle des Pull-Modus eine Änderungsart-spezifische Erfassungsroutine unterstützen, welche Daten von den Quellen anfordert.
Weitere Aufgaben von SPC und DPC sind die Überwachung eines laufenden und der Aufruf des nächsten Workflows. Die aktiven Domänen sind Teil der Workflow-Konfiguration und müssen den Steuerkomponenten daher nicht dediziert mitgeteilt werden.

3.5.3 Akteure

Durch die zuvor in diesem Kapitel beschriebenen Konzepte, werden unter anderem Separation-of-Concerns und Wiederverwendbarkeit gefördert und umgesetzt. Dies führt dazu, dass mehrere Akteure der Analyselösung mit separaten Interaktionsmöglichkeiten und getrennten Aufgabenbereichen unterstützt werden. Diese sind in der folgenden Liste aufgeführt[1]:

- *Komponentenhersteller* werden durch die verschiedenen Abbildungsstrategien unterstützt, durch welche keine zusätzliche Software auf den Komponenten benötigt wird. Zudem ist durch die Unterstützung zukünftiger, allgemeiner Schnittstellen, wie die Verwaltungsschale, deren Ziel die Gesamtabdeckung aller Assets ist, eine zusätzliche Entlastung der Komponentenhersteller zu erreichen, sollten sich diese Konzepte durchsetzen.
- *Modellierungsexperten* sind die Personen, welche in der Lage sind, TBoxen für Komponenten-, System- und Sicherheitsontologien sowie zugehörige Abbildungsontologien zu erstellen.
- *Endnutzer* der Analyselösung sind Sicherheitsanalysten, Sicherheitsbeauftragte oder Administratoren, die keine Berührungspunkte mit den modellbezogenen Funktionalitäten, Darstellungen und, im Allgemeinen, Implementierungen haben müssen. Endnutzer möchten Analysen konfigurieren und sich hierfür des wiederverwendbaren Wissens der Sicherheitsgemeinschaft bedienen. Zudem möchten sie Analysen ohne konkrete Kenntnisse des Modells ausführen und auswerten.
- Endnutzer sind nicht die einzigen Sicherheitsexperten. Insbesondere müssen sie sich nicht mit allen Standards, Richtlinien, Best-Practices, möglichen Fehlerquellen usw. auskennen. Genau genommen wäre es unrealistisch zu behaupten, dass diese Anforderungen irgendein Sicherheitsexperte erfüllen kann. Aus diesem Grund ist das Wissen

[1] Die Liste ist nicht erschöpfend, führt aber die Akteure auf, die für ein klares Verständnis des Rahmenwerks und seines Separation-of-Concerns-Ansatzes hilfreich sind.

der gesamten Sicherheitsgemeinschaft besonders wertvoll. In diesem Rahmenwerk wird ein Weg definiert, dieses Wissen nutzbar zu machen (vgl. insbes. Abschnitt 3.5.4). Dabei bilden sich aus der Sicherheitsgemeinschaft zwei Akteure:

1. *Policy-Experten*: Die Sicherheitsexperten der Sicherheitsgemeinschaft, welche in bestimmten Teilbereichen der hier behandelten Disziplinen (Sicherheitsstandards, Best-Practices, Bedrohungen, Schwachstellen, usw.) in der Lage sind, Regeln in natürlicher Sprache zu formulieren, die in diesen Bereichen gelten und prüfbar sind, sowie diese mit entsprechenden weiteren Informationsquellen verknüpfen können (falls vorhanden). Diese natürlichsprachlichen Richtlinen entsprechen den Analyseregelformulierungen aus Abschnitt 3.4.3.2.
2. *SyMP-Power-User*: Modellierungsexperten, die in der Lage sind, Analyseregelumsetzungen für die Analyseregelformulierungen bereitzustellen. Sie müssen keine SyMP-Experten (siehe nächster Punkt) sein, müssen jedoch gewisse Kenntnisse der Modellierungssprache, der entsprechenden relevanten Ontologien und der Abfragesprache besitzen und gleichzeitig über genug technisches Verständnis verfügen, um die Analyseregelformulierungen zu verstehen.

- *SyMP-Experten* übernehmen die Rolle der Weiterentwicklung und Aktualisierung einer SyMP-Implementierung.

Abbildung 3.35 bietet einen Überblick über die Aufgaben, welche die Akteure in SyMP manuell erfüllen müssen und visualisiert deren Zusammenhänge und Abhängigkeiten über Informationsflüsse. Eine mögliche Reihenfolge der Schritte wird über deren Nummerierung vorgegeben. Die Akteure sind als Rollen zu sehen, sodass auch mehr als eine Rolle von einer Person übernommen werden kann.

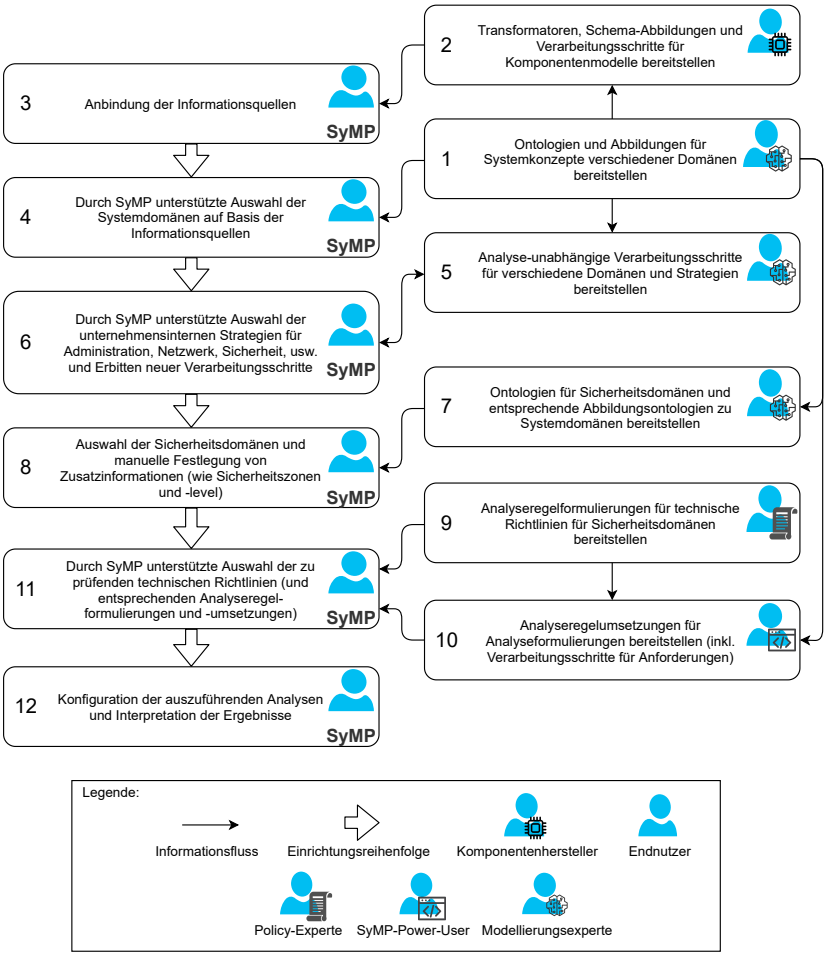


Abbildung 3.35: Überblick über die manuell zu verrichtenden Schritte mit Zuordnung zu Akteuren.

Außerdem ist eine Analyse eine Menge von Analyseregeln (bzw. Paaren von Formulierungen und Umsetzungen). Denn in der Regel möchte ein Endnutzer nicht nur genau eine technische Richtlinie prüfen. Mehr dazu erörtert der nächste Abschnitt, denn bis hierhin fehlt noch das konkrete Konzept für die Analysesicht, welches das Analyseregelkonzept aus Abschnitten 3.4.3.1 und 3.4.3.2 in das restliche Rahmenwerk einbindet. Dies wird im folgenden Abschnitt nachgeholt.

3.5.4 Kollaborationsansatz und Automatisierung der Analysen

Ein bedeutender Aspekt von SyMP ist die Unterstützung der Sicherheitsgemeinschaft, indem die Zusammenarbeit von Policy-Experten, Modellierungsexperten (damit natürlich auch SyMP-Power-Usern) und Endnutzern technisch ermöglicht wird. Dies erfolgt unter Berücksichtigung von Separation-of-Concerns durch ein *SyMP-Analyse-Hub* (kurz *SyMP-Hub*) und ein Automatisierungskonzept für die Ableitung von Workflows für die Phasen 4-6. Ein Überblick dieses Konzeptes ist in Abbildung 3.36 zu sehen und wird in den folgenden Abschnitten näher erörtert.

3.5.4.1 SyMP-Hub

Dieser Teil des SyMP-Rahmenwerks wurde zum Großteil in einer Masterarbeit entwickelt *[Klo21]*. Wesentlich hierbei ist das Analyseregelkonzept aus den Abschnitten 3.4.3.1 und 3.4.3.2.
Dem Konzept folgend entwickeln Policy-Experten Analyseregelformulierungen in natürlicher Sprache. Dabei können diese Beschreibungen um Anforderungsbeschreibungen ergänzt werden, womit die Regeln um Aussagen zum Informationsbedarf erweitert werden (vgl. Abbildung 3.36). Diese wiederum erleichtern eine Umsetzungsentwicklung der Analyseregeln. Denn diese Umsetzung wird nicht von Policy-, sondern von SyMP-/Modellierungsexperten durchgeführt. Dabei kann und sollte es, wie in Abschnitt 3.4.3.2 beschrieben, mehrere Umsetzungen einer Analyseregel geben, die zwar alle mit der

Formulierung verknüpft werden, jedoch unterschiedliche semantische und technologische Aspekte adressieren. So ist es normalfalls nötig, pro Systemontologie eine Analyseregelumsetzung bereitzustellen, da die entsprechende Richtlinie die Verwendung von systemspezifischen Konzepte nötig macht.

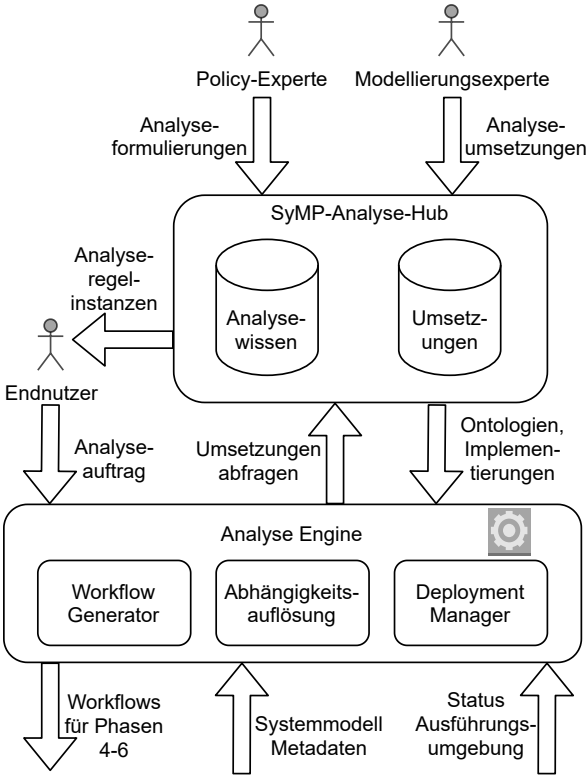


Abbildung 3.36: Überblick über das Konzept von SyMP-Hub und -Engine.

Das Hub ist die Plattform auf der die Formulierungen und Umsetzungen verwaltet werden, die gemeinsam das Analysewissen bilden und dient gleichzeitig als zentrale Registrierungsumgebung für die in den Umsetzungsdefinitionen referenzierten Ontologien und Programmmodule. Die Auswahl der jeweiligen Analyseregeln und deren Umsetzungen (die resultierende Kombination wird hier *Analyseregelinstanz* genannt) obliegt hierbei dem Endnutzer, wobei die Ontologie-basierte Repräsentation der Analyseregeln maschinelle Unterstützung bei dieser Aufgabe ermöglicht, sodass eine Implementierung des Hubs dem Endnutzer zum Beispiel zu seinem System passende Umsetzungen vorschlagen kann.

Zwar werden weitere Aufgaben des Hubs von SyMP vorgeschrieben, jedoch muss das Hub im Sinne der Automatisierung ein weiteres Teilkonzept unterstützen, die Analyse-Engine.

3.5.5 Analyse-Engine

Die *Analyse-Engine* (vgl. Abbildung 3.36) hat mindestens die folgenden Aufgaben:

- Entgegennehmen eines *Analyseauftrags* (also eine Menge von Analyseregelinstanzen), die logisch zu einer vom Endnutzer definierten Analyse zusammengefasst wurden.
- Abrufen der Ontologien, Implementierungen und sonstigen Informationen, die zu den Analyseregelinstanzen gehören, jedoch nicht vom Endnutzer geliefert werden.
- Auflösen von Abhängigkeiten. Dazu gehört herzuleiten, welche Implementierungen und Ontologieabhängigkeiten bereits durch die Phasen 1-3 in der Erzeugung des bereinigten Systemmodells eingesetzt wurden. Diese müssen nicht erneut eingesetzt werden. Analog wird geprüft, welche Workflows von Phasen 4-6 überhaupt erzeugt werden müssen, da die entsprechenden Umsetzungen eventuell bereits durch existierende Workflows abgedeckt sind.

- Generieren von maximal drei Workflows pro Analyseauftrag (einen für jede der letzten drei Phasen) unter Verwendung des Wissens der Abhängigkeitsauflösung.
- Prüfen, ob alle benötigten Arbeiter bereits eingerichtet und gestartet sind und Nachholen der Einrichtung und des Startens, falls dem nicht so ist.
- Übernahme der DPC-Funktionalität.

Dem Endnutzer sollte in einer geeigneten Implementierung von SyMP ein entsprechendes Werkzeug zur Verfügung stehen, mit dem er die Analyseaufträge und Analyseausführungen bzw. -ergebnisse, verwalten kann. Die Ausprägung des Werkzeugs wird hier jedoch nicht definiert.
Im folgenden Abschnitt wird noch genauer auf die Workflow-Generierung eingegangen.

3.5.5.1 Workflow-Generierung

Der *Workflow-Generator* ist eine Analyse-Engine-Komponente, welche die vom Endnutzer in Auftrag gegebenen Analysen in Workflows umwandelt. Zur Generierung der Workflows besitzt jeder Modellverarbeitungsschritt in den Analyseregelumsetzungen eine Zuordnung zu einer der letzten drei SyMP-Phasen. Diese Zuordnung ist bereits bei der Zusammensetzung der Analyseregelumsetzung relevant, da diese die Modellverarbeitungsschritte modularisieren und somit leicht wiederverwendbar machen. Die Modellverarbeitungsschritte und Ontologieabhängigkeiten können als Knoten eines Graphen und deren Abhängigkeiten voneinander als gerichtete Kanten dieses Graphen interpretiert werden.
Allerdings besitzen die Phasenzuordnungen, wie bereits angedeutet, bei der Generierung der Workflows eine geringere Priorität, als die Prüfung der bereits eingesetzten Modellverarbeitungsschritte und Ontologieabhängigkeiten und sorgen so nur für eine Zuordnung, falls die Modellverarbeitungsschritte und Ontologieabhängigkeiten nicht bereits abgedeckt sind.

Umgekehrt enthalten die Analyseregelumsetzungen auch Modellverarbeitungsschritte und Ontologieabhängigkeiten, welche von den ersten drei Phasen erwartet werden, um Anforderungen an das bereinigte Systemmodell abzubilden. Wurden diese nicht in den ersten drei Phasen eingesetzt, bilden sie den ersten Teil des Static-Knowledge-Extension Workflows. Durch diese beiden Sonderfälle wird sichergestellt, dass Separation-of-Concerns auch für die Überschneidung der System- und Sicherheitsdomänen eingesetzt werden kann, ohne die Redundanz von Verarbeitungsschritten zu fördern.

Ein konkreter Algorithmus für die Erstellung der Workflows wird von SyMP nicht vorgegeben, wurde jedoch in *[Klo21]* entwickelt, implementiert und evaluiert.

3.6 SyMP-Implementierung

Zur Evaluation des zuvor beschriebenen SyMP-Rahmenwerks und für dessen Einsatz in der modellbasierten Sicherheitsanalyse im BMBF-Projekt AICAS[1], wurde eine entsprechende prototypische Implementierung entworfen, umgesetzt und über mehrere Monate weiterentwickelt. Der Stand dieser Implementierung, der zur Evaluation (vgl. Abschnitt 3.7) eingesetzt wurde, wird im Folgenden vorgestellt.

Abbildung 3.37 zeigt die grundlegende Architektur der SyMP-Implementierung, inklusive des entsprechenden Ökosystems, unter dem beispielsweise der *SyMP-Analyse-Client* (kurz *SyMP-Client*) einzuordnen ist. Diese Architektur wird im Rest dieses Abschnitts näher erläutert. Eine zentrale Rolle in der Implementierung spielt die Automatisierungslösung *Camunda*[2], von der die *Camunda-Plattform*[3] und der *Camunda-Modeler*[4] verwendet wurden. Die Camunda-Plattform implementiert eine Workflow-Engine für den

[1] https://www.forschung-it-sicherheit-kommunikationssysteme.de/projekte/aicas, zuletzt zugegriffen: 08.05.2021.

[2] https://camunda.com/de/products/camunda-bpm, zuletzt zugegriffen: 27.11.2020.

[3] https://camunda.com/de/products/camunda-platform, zuletzt zugegriffen: 18.04.2021.

[4] https://camunda.com/de/products/camunda-platform/modeler, zuletzt zugegriffen: 18.04.2021.

Workflow-Modellierungsstandard BPMN und bietet entsprechende Administrationswerkzeuge, wie das Aufgabenmanagement-Werkzeug *Tasklist* oder die Überwachungsoberfläche *Cockpit*.

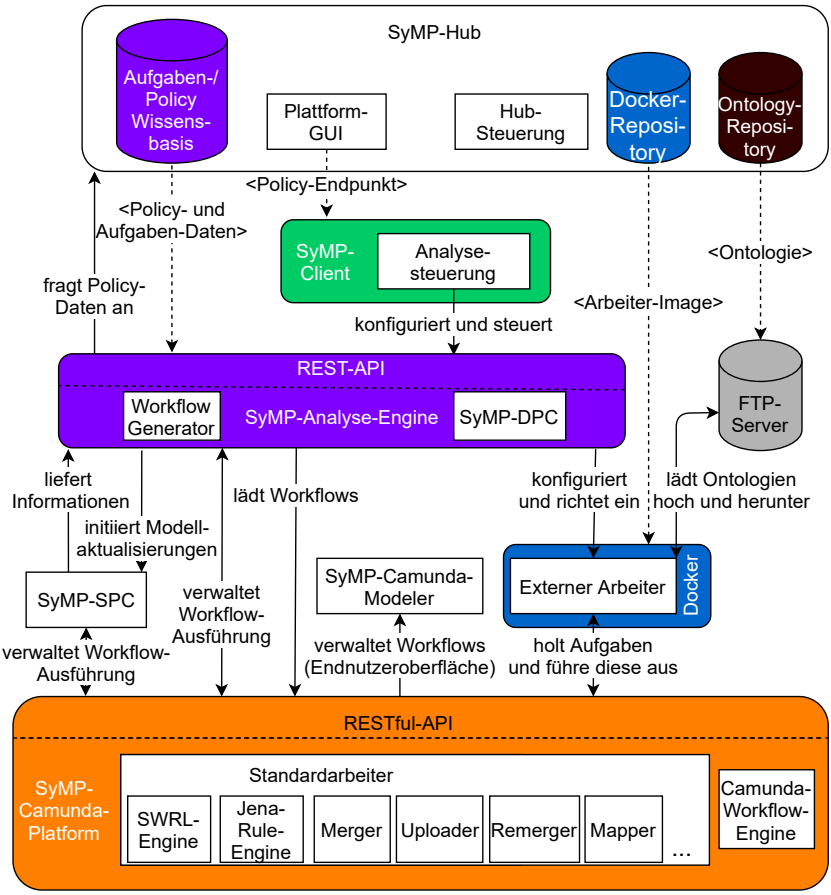


Abbildung 3.37: Implementierte Gesamtarchitektur für die minimalinvasive Sicherheitsanalyse mit SyMP.

Die Plattform bietet außerdem eine *REST API*[1], über welche diese und weitere beliebige Werkzeuge angebunden werden. Zu diesen Werkzeugen zählt beispielsweise der Camunda-Modeler, welcher zur manuellen Modellierung von Workflows in BPMN verwendet wird.
Wird ein BPMN-Workflow in der Camunda-Plattform installiert und über die API (z.B. durch die Tasklist) gestartet, werden die jeweils aktuellen Aufgaben des Workflows, von denen jede ein Thema (engl. *Topic*) besitzt, in einen Aufgabenpool gelegt. Die Camunda-Plattform unterstützt damit das *Publish-Subscribe-Muster*. Subscriber[2] der Themen sind dabei beliebige interne oder externe Arbeiter, welche die Bearbeitung und Fertigstellung eines Themas melden und somit Mehrfachausführung von Aufgaben vermeiden. Zusätzlich können Variablen im Kontext einer Aufgabe oder im Kontext der gesamten Workflow-Ausführung zwischen Arbeiter und Plattform (und somit auch zwischen mehreren Arbeitern) ausgetauscht werden. Damit stellt Camunda eine ideale Grundlage für die Implementierung des Workflow-basierten SyMP Rahmenwerks dar.

Für die Ontologieverwaltung der SyMP-Implementierung wurde, aufgrund der eingeschränkten Kompatibilität von semantischen Datenbanken mit mehr als einer Semantic-Web-API, ein Datei-basiertes Verfahren eingesetzt. Die Dateiablage wird von einem FTP-Server übernommen. Arbeiter wurden als in Docker[3]-Containern laufende Subscriber implementiert. Dabei lädt jeder Arbeiter nach Ausführung der Aufgabe seine Ergebnisontologie unter dem eindeutigen Identifikator der Aufgabe auf den FTP-Server[4] und hängt die URI der Datei an eine Liste an, die von allen Tasks eines Workflows genutzt und als Workflow-interne Variable verwaltet wird. So weiß der nächste Arbeiter, welche Ontologie er laden muss.

[1] https://docs.camunda.org/manual/latest/reference/rest/overview, zuletzt zugegriffen: 27.11.2020.
[2] Hier entsprechend das Programm, welches bei einer neuen Arbeit, die dem Thema zugewiesen ist, „benachrichtigt werden möchte“.
[3] https://docs.docker.com/engine, zuletzt zugegriffen: 04.12.2020.
[4] In Abbildung 3.37 ist die Kommunikation mit dem FTP-Server aus Übersichtlichkeitsgründen nur für die Arbeiter außerhalb der Plattform visualisiert. Dies gilt jedoch auch für die Plattform-internen Standardarbeiter.

Für die Plattform wurden verschiedene Standardarbeiter erstellt, die wegen Ihres regen Einsatzes direkt als Teil der Plattform umgesetzt wurden:

- *Ontology Upload*: Der Arbeiter lädt eine oder mehrere Ontologien zu deren Verarbeitung. Die zu ladenden Ontologiedateien werden als Aufgabenparameter referenziert.
- *Ontology Merge*: Der Arbeiter integriert die geladenen Ontologien in eine gemeinsame Ontologie und verwendet dabei die TBox der ersten, geladenen Ontologie.
- *Ontology Remerge*: Der Arbeiter führt Ergebnisontologien $\Theta_1, .., \Theta_i$ im Rahmen der Synchronisation paralleler Sequenzflüsse zusammen, indem alle Aussagen von Θ_x ($x \in [2, i] \subset \mathbb{N}$) in Θ_1 kopiert werden, die nicht bereits in Θ_1 existieren.
- *Ontology Mapping*: Der Arbeiter fügt eine als Aufgabenparameter übergebene Abbildungsontologie dem letzten letzte Ergebnismodell hinzu. Abbildungsausdrücke und importierte Ontologien werden dabei übernommen. Das Resultat ist ein Modell, in dem all diese Ontologien integriert sind.
- *SWRL Engine*: Der Arbeiter führt einen Pellet-Reasoner und dessen SWRL-Regel-Engine auf der Ontologie aus. Die SWRL-Regeln werden dabei als Aufgabenparameter übergeben und verbleiben in der Ontologie.
- *Jena Rule Engine*: Der Arbeiter führt die *Jena-Rule-Engine*[1] für OWL aus. Die Jena-Regeln werden als Aufgabenparameter übergeben und verbleiben nicht in der Ontologie.

Diese Arbeiter und die entsprechenden Aufgaben bilden die grundlegende Ontologieverwaltung und -verarbeitung von SyMP ab. Alle weiteren Arbeiter werden als externe Arbeiter angebunden.

[1] https://jena.apache.org/documentation/inference, zuletzt zugegriffen: 07.03.2021.

Für die Workflow-Modellierung, welche in der hier vorgestellten Implementierung für die ersten drei Phasen noch manuell stattfindet, wurde der Camunda-Modeler über ein eigenes Plugin erweitert. Dieses Plugin ermöglicht sowohl eine visuelle Unterstützung des Nutzers, um die entsprechenden Standard-Aufgaben leicht unterscheiden zu können, als auch einen Linter, der bei der korrekten Verwendung dieser Aktivitäten hilft. Durch den Linter werden Meldungen generiert, wenn ein Nutzer bestimmte fehlerhafte Definitionen vornimmt. Beispielsweise dürfte in einer ersten Version der SyMP Camunda-Plattform, die im Rahmen einer Bachelorarbeit entstand *[Bet20]*, nach einer Upload- nur eine Merge- oder eine weitere Upload-Aufgabe folgen. Dies ist in Abbildung 3.38 zu erkennen.

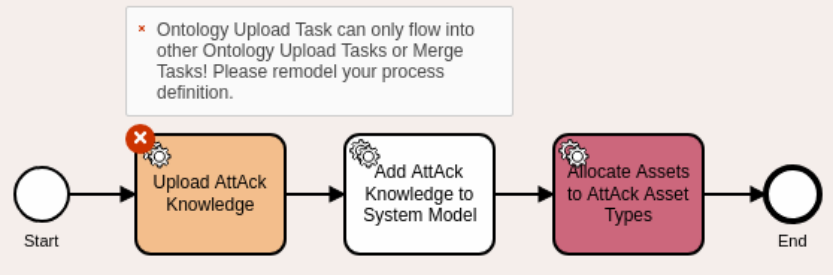


Abbildung 3.38: Durch den Linter gefundener Fehler im mithilfe des Camunda-Modelers entworfenen Workflow.

Zudem hilft der Linter bei der Konzeptionierung des Workflow-Flusses, indem Verbesserungsmöglichkeiten identifiziert und durch Vorschläge visualisiert werden. So findet der Linter parallelisierbare oder zusammenführbare Aufgaben. Ein Beispiel hierfür ist in Abbildung 3.39 zu sehen, in der zwei parallele SWRL-Regel-Aufgaben vom selben SWRL-Engine-Arbeiter verarbeitet und somit zur Effizienzsteigerung in eine Aufgabe zusammengefasst werden können. Der SPC wurde mithilfe einer Spring-Boot-Anwendung[1] umgesetzt, welche die

[1] https://spring.io/projects/spring-boot, zuletzt zugegriffen: 07.03.2021.

Verwaltung bereits installierter, systemdomänenspezifischer Workflows über die Camunda REST-API regelt und über die eigene REST-API Informationen zu Workflows, Arbeitern und Modellergebnissen der ersten drei SyMP-Phasen bereitstellt. Der DPC wurde, als Teil der SyMP-Analyse-Engine, ebenfalls mit einer Spring-Boot-Anwendung umgesetzt.

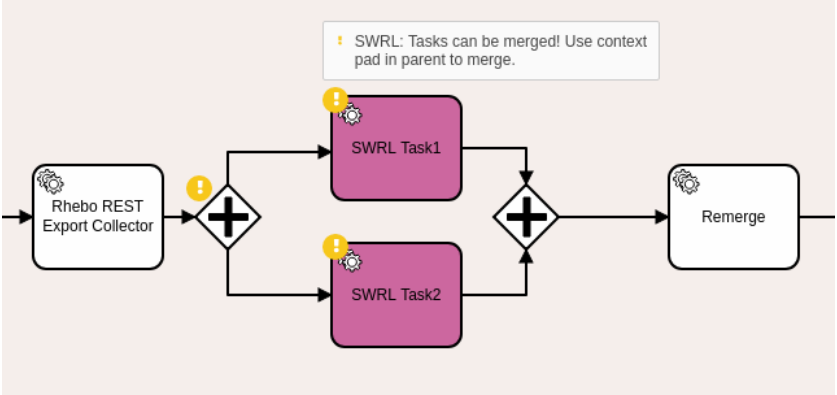


Abbildung 3.39: Durch den Linter gefundene, mögliche Verbesserung im mithilfe des Camunda-Modelers entworfenen Workflow.

Wie in Abschnitt 3.5.5 beschrieben, ist die Analyse-Engine die Schnittstelle zwischen SPC, Camunda-Plattform, einer Docker-Umgebung, dem SyMP-Hub und dem Endnutzer. Für den Endnutzer wurde in dieser Implementierung jedoch der schon erwähnte SyMP-Client entwickelt, der das Verwalten der Analysen durch eine zusätzliche Automatisierungsschicht erleichtert und damit auch SyMP-konforme Schnittstellen bereitstellt.
Bevor hier weiter auf die Analysis-Engine eingegangen wird, müssen die Umsetzungen der drei zuletzt genannten Komponenten erläutert werden. Das SyMP-Hub wurde ebenfalls mit einer Spring-Boot-Anwendung umgesetzt.

Diese bietet zum Zeitpunkt des Schreibens dieser Dissertation die Funktionalität, sowohl die natürlichsprachlichen Analyseregelformulierungen, als auch die Analyseregelumsetzungen einzutragen, zu teilen und zu verwalten.

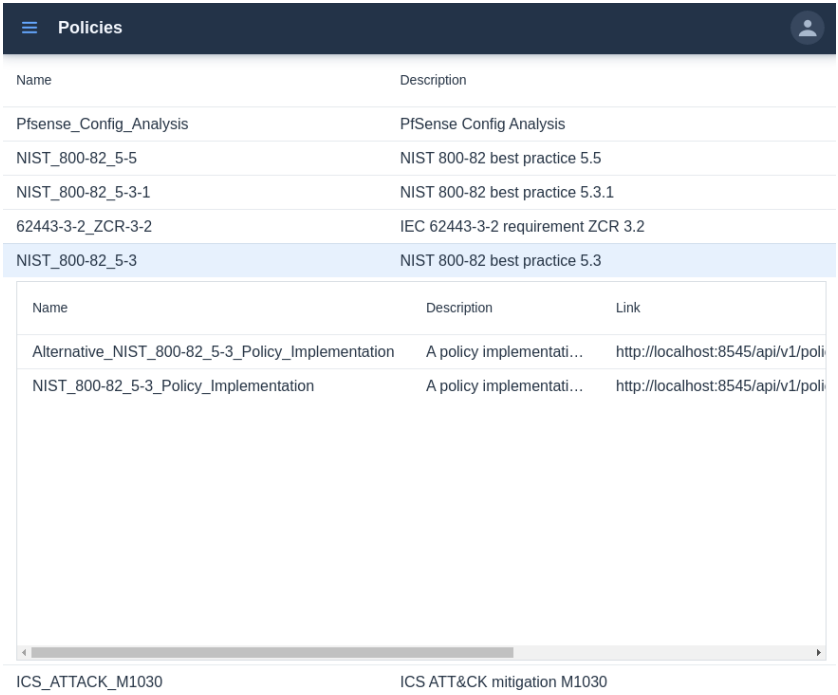


Abbildung 3.40: Auswahl Analyseregeln und Umsetzungen über die Benutzeroberfläche des SyMP-Hub.

Dafür wurde die, in Abschnitt 3.4.3.2 beschriebene, Analyseregelontologie als Policy-Wissensbasis mithilfe der Graphdatenbank neo4j[1] mit der semantischen Erweiterung[2] umgesetzt. Damit sind auch semantische Abfragen auf den Analyseregel-Repräsentationen durchführbar, die unter anderem die Extraktion beliebiger Beziehungen zwischen den Regeln ermöglichen.
Da das SyMP-Hub als Web-Plattform geplant wurde, verfügt es über eine Web-Oberfläche (Plattform-GUI) zur Verwaltung der Analyseregelumsetzungen und ihrer Implementierungen (vgl. Abbildung 3.40). Der Endnutzer kann darüber eine beliebige Analyseregelinstanz auswählen und deren URI-Endpunkt kopieren (z.B. *NIST_800-82_5-3* mit *NIST_800-82_5-3_Policy_Implementation* aus Abbildung 3.40). Dieser Endpunkt wird dann bei der Erstellung einer Analyse im SyMP-Client zum Hinzufügen der Analyseregelinstanz (bzw. aus Analysesicht „Policy“) angegeben, was in Abbildung 3.41 zu sehen ist[3]. Hier ist es möglich mehrere Analyseregelinstanzen anzugeben, wobei der Client für jede von ihnen die entsprechenden Metainformationen herunterlädt. Daher bekommt der Nutzer auch eine direkte Rückmeldung, ob die Instanz gefunden werden konnte oder nicht (vgl. *Policy found* in Abbildung 3.41). In Abbildung 3.41 ist außerdem zu sehen, dass für die dargestellte Analyse das Systemmodell *Lab* gewählt wurde. Beim Start einer Analyse über den SyMP-Client wird ein entsprechender Analyseauftrag an die Analyse-Engine gesendet, der sowohl die Referenzen zu den Analyseregelumsetzungen, als auch die Information über das Zielsystem (im Beispiel *Lab*) enthält. Ersteres löst die Analyse-Engine über die REST-Schnittstelle des SyMP-Hub auf, letzteres dient der Extraktion der in den ersten drei Phasen verwendeten Modellverarbeitungsschritte und Ontologien über eine REST-Schnittstelle des SPC. So bekommt der Workflow-Generator die Informationen, die für die Abhängigkeitsprüfung der Analyseregelumsetzungen im Rahmen der Workflow-Generierung benötigt werden (vgl. Abschnitte 3.5.5 und 3.5.5.1). Wie bereits erwähnt sollte der Algorithmus

[1] https://neo4j.com/, zuletzt zugegriffen: 07.03.2021.

[2] https://neo4j.com/labs/neosemantics/, zuletzt zugegriffen: 07.03.2021

[3] In den beiden Abbildungen der URI ist der Host unterschiedlich. Dies liegt daran, dass das Hub sich nur als *localhost* kennt, wobei der Client das Hub über den Namen des Docker-Containers als *ah* (Analysis Hub) ansprechen muss. Diese Änderung ist bei einem, in der Cloud betriebenen Hub natürlich nicht notwendig und beschränkt sich daher auf die Testumgebung.

des Workflow-Generators verschiedenste Optimierungsregeln unterstützen. Darunter fällt in dieser Implementierung das Zusammenführen von SWRL-Regeln und Jena-Regeln (wie dies auch der Linter des Camunda-Modelers vorschlägt), das Parallelisieren unabhängiger Aufgaben und die Priorisierung der Parallelisierung vor der Zusammenführung von Regeln. Für weitere Details werden interessierte Lesende auf *[Klo21]* verwiesen.

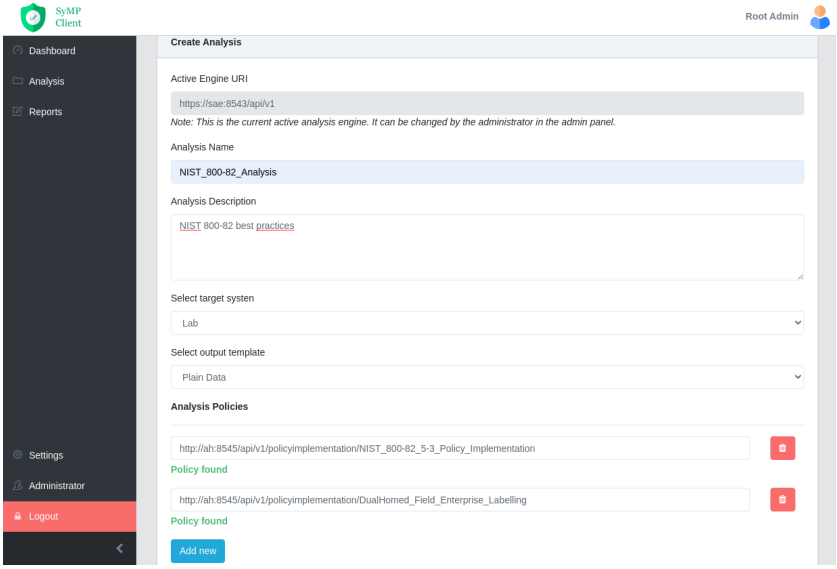


Abbildung 3.41: Erzeugen einer Analyse über den SyMP-Client.

Generell führt die Analyse-Engine nach dem Initiieren der Ausführung einer Analyse (das über ihre REST-API erfolgt) die folgenden Schritte durch:

1. Anfragen der eben beschriebenen Informationen über die Systemmodellgenerierung.

2. Starten der Systemmodellgenerierung über die REST-API des SPC. Sind keine Änderungen vorhanden, teilt der SPC sofort mit, dass das Modell aktuell ist.
3. Ausführen des Algorithmus zur Workflow-Generierung. Die Workflows werden dabei direkt als BPMN-Workflows erzeugt.
4. Ermittlung der noch fehlenden Arbeiter-Implementierungen (-Container) und deren Herunterladen vom Docker-Repository.
5. Installieren und Starten der Arbeiter-Container[1].
6. Installieren, Starten und Steuern der generierten Workflows und Speichern wichtiger Workflow-Metainformationen, die für die Generierung neuer Workflows berücksichtigt werden (vgl. Abschnitt 3.5.5.1).

Eine Ausnahme bildet der Fall, in dem sich das Systemmodell nicht verändert hat (d.h. der Zeitstempel der Modell-Datei nicht verändert wurde) und die Analyse bereits auf der aktuellen Version des Systemmodells ausgeführt wurde. In jenem Fall gibt die Analyse direkt das Ergebnis zurück.
Ist der letzte Workflow erfolgreich durchgelaufen, stellt die Analyse-Engine alle im Analyseauftrag angegebenen Abfragen an das vorberechnete analysespezifische Systemmodell und generiert damit die entsprechenden Berichte, die dem SyMP-Client zur Verfügung gestellt werden (ein Einblick in die entsprechenden Oberfläche wir in der Evaluation (Abbildung 3.53) gegeben). Beides erfolgt über einen Plugin-Mechanismus. Aktuell existiert nur ein Abfrage- und Bericht-Plugin, welches die Abfrage über SPARQL definiert und den Bericht als JSON-Repräsentation der Abfrageergebnisse zurückgibt. Alle Berichte einer Analyse werden gesammelt und über eine REST-Rückruffunktion an den SyMP-Client übergeben, wo sie gemeinsam in eine auswählbare Vorlage (vgl. Abbildung 3.52) geparst und angezeigt werden.

[1] Alle Komponenten der Prototypimplementierung laufen in Docker-Containern und teilen sich ein virtuelles Netzwerk. So sind die Hauptkomponenten wie die Camunda-Plattform, der SPC, die Analyse-Engine und der FTP-Server über statische Docker-Service-Namen erreichbar. Dadurch müssen Arbeiter-Container nicht dediziert konfiguriert werden.

3.7 Evaluation

Die Strategie zur Evaluation von SyMP basiert auf der Überprüfung, ob die in Abschnitt 1.1.1 definierten Hypothesen nachgewiesen werden konnten und somit die Forschungsfragen aus Abschnitt 1.1.1 beantwortet wurden.
Hierfür wurden, für alle in dieser Arbeit behandelten Analysearten, verschiedene Analysen unter Verwendung der SyMP-Implementierung durchgeführt (vgl. Abschnitte 3.7.2-3.7.6). Die beschriebenen Analysen bestehen dabei insgesamt aus 11 Richtlinien, 12 Ontologien, über 70 unterschiedlichen Modellverarbeitungsschritten und wurden auf die beiden in Abschnitt 3.1 beschriebenen Systeme aufgeteilt. Dabei gilt es noch einmal festzuhalten, dass lediglich die Camunda-Plattform, also die Workflow-Managementumgebung, und der Camunda-Modeler sowie die zugehörigen REST-Schnittstellen Fremdleistungen waren. Alle Regeln, Ontologien, Arbeiter-Programme, Controller, Clients, Registries und weitere Software sind Eigenentwicklungen zur Umsetzung von SyMP (und den damit verbundenen Konzepten und Methoden), die im Rahmen der hier beschriebenen Evaluation folglich auch untersucht werden.
Anhand dieser Umsetzungen konnte untersucht werden, ob und wie weit die Anforderungen aus Abschnitt 3.2 erfüllt werden konnten. Damit wurde auch die Möglichkeit geschaffen, SyMP mit dem Stand der Technik zu vergleichen, obwohl ein direkter technologischer Vergleich mangels verfügbarer, offener Lösungen und dem zu großen Aufwand von Nachbildungen nicht realisiert werden kann.
Der Aufbau dieses Kapitels beginnt mit einer Einführung in die erstellten und eingesetzten Ontologien. Dem folgend, werden die durchgeführten Analysen in den jeweiligen Unterabschnitten erläutert. Dabei werden die Phasen des SyMP-Rahmenwerks wie folgt abgekürzt: *KC* (Knowledge Collection), *KF* (Knowledge Fusion), *MC* (Model Cleaning), *SKE* (Static Knowledge Extension), *DKE* (Dynamic Knowledge Extension) und *A* (Analysis). Danach wird geprüft, ob die in Abschnitt 1.2 definierten Ziele erreicht wurden. Im vorletzten Schritt wird der Nachweis der Hypothesen diskutiert, bevor in Abschnitt 3.7.10 eine Beschreibung identifizierter Probleme und möglicher Lösungsansätze erfolgt. Über das gesamte Evaluationskapitel hinweg werden die Ergebnisse

an den entsprechenden Stellen diskutiert. Daher entfällt ein generelles Diskussionskapitel. Allerdings werden die Evaluationsergebnisse in Abschnitt 3.7.11 noch einmal zusammengefasst.

Alle Evaluationen wurden auf einem Ubuntu 18.04.05 LTS Betriebssystem ausgeführt, welches auf einer virtuellen Maschine mit 64 GB Arbeitsspeicher und 8 „Intel(R) Xeon(R) CPU E5-2650 v4"-Kernen mit 2,2 GHz Taktung in einer „Proxmox 6.3-2"-Serverumgebung lief, wobei keine der in der Evaluation eingesetzten Anwendungen Mehrkernnutzung unterstützte.
Als zugrundeliegende, industrielle Systeme wurden die in Abschnitt 3.1 beschriebenen Beispielumgebungen eingesetzt. Dabei wurde mithilfe der Laborumgebung die automatisierte, selbstauskunftbasierte Informationsextraktion (vgl. Abschnitt 3.4.1.2) und die Effektive Konfiguration (vgl. Abschnitt 3.4.2.1) evaluiert. Die Laborumgebung wurde außerdem aufgrund ihrer beliebigen Konfigurierbarkeit als System für Evaluationen zur Konfigurations-, Schwachstellen- und Kompatibilitätsanalyse verwendet.
Hingegen wurde die PoC-Umgebung zum Nachweis der Anwendbarkeit des SyMP-Rahmenwerks für realistische industrielle Systeme und dessen Evaluation für die Analysearten Bedrohungsanalyse und Angriffserkennung/-korrelation eingesetzt, welche den Einsatz von SyMP für reale Angriffe, realen und industriellen Netzwerkverkehr und die Unterstützung einer aktuellen Monitoring-Lösung für industrielle Systeme zeigen.

3.7.1 Ontologien

In diesem Abschnitt werden die in den Analyse-Evaluationen verwendeten Ontologien eingeführt. Die Namensräume wurden den Ontologien in Form von Präfixen zugeordnet. Damit wird vermieden, vor jeder SWRL-Regel oder SPARQL-Abfrage Präfixe zu definieren und trotzdem die Eindeutigkeit der Konzepte sichergestellt. Einige der Ontologien sind zudem öffentlich zugängig und können mit gängigen RDF-Editoren wie Protegé[1] angesehen und editiert werden.

[1] https://protege.stanford.edu, zuletzt zugegriffen: 25.04.2021.

ICS-ATT&CK®-Ontologie (Präfix ics-attAck)

Mithilfe eines Python-Programms wurde die, bereits in Abschnitt 2.6.1 vorgestellte, ICS-ATT&CK®-Wissensbasis[1] geparst und in eine Ontologie überführt. Dabei wurden sowohl TBox als auch ABox automatisch generiert. Die entsprechende Ontologie ist auf GitHub[2] verfügbar. Sie enthält die wesentlichen Elemente des Rahmenwerks und ihre Zusammenhänge. Zu den Elementen gehören unter anderem Taktiken (engl. *Tactics*), Techniken (engl. *Techniques*), Gegenmaßnahmen (engl. *Mitigations*), aber auch Referenzen zu Standards wie IEC 62443 und NIST SP 800-53.

IEC-62443-Ontologie (Präfix iec-62443)

Diese Ontologie wurde manuell aus den IEC-62443 Standards erzeugt, mit dem konkreten Ziel der Sicherheitsanalyse. Demnach enthält die Ontologie auch nur die für die Analyse relevanten Konzepte der Standards und soll durch die Sicherheitsgemeinschaft gepflegt und erweitert werden. Sie ist ebenfalls auf GitHub[3] verfügbar.

ICS-ATT&CK®-IEC-62443-Abbildungsontologie (Präfix attAck-62443)

Wie bereits erwähnt, enthält die ICS-ATT&CK®-Ontologie Referenzen zu IEC-62443-Anforderungen. Die ICS-ATT&CK®-IEC-62443-Abbildungsontologie (kurz *Att&CK-62443-Ontologie*) nutzt diese Referenzen zur Verknüpfung von Gegenmaßnahmen- und IEC-62443-Anforderungen. Diese Ontologie wird ebenfalls manuell gepflegt und ist auf GitHub[4] verfügbar.

[1] https://collaborate.mitre.org/attackics/, zuletzt zugegriffen: 02.12.2020.
[2] https://github.com/FlorianPatzer/AttACK_knowledge.
[3] https://github.com/FlorianPatzer/iec62443_knowledge.
[4] https://github.com/FlorianPatzer/mapping_ontologies.

Rhebo-Systemontologie (Präfix rhebo-system)

Einige der Evaluationen basieren auf den aus Netzwerkverkehr extrahierten Informationen über Hosts und deren Kommunikation in der PoC. Diese Informationen wurden mithilfe der Monitoring-Plattform *Rhebo Industrial Protector*[1] gesammelt. Von der Plattform können diese Informationen mittels REST-API geladen werden. Um nun die geladenen Informationen in eine Ontologie umzuwandeln, wurde eine spezifische TBox (die Rhebo-Systemontologie) erstellt.

Rhebo-CEF-Ontologie (Präfix rhebo-cef)

Der Rhebo Industrial Protector bietet ebenfalls eine Anomalieerkennung, deren Anomaliemeldungen im CEF-Format [Mic17] exportiert werden können. Diese Meldungen werden von einem Python-Programm geparst und in eine Rhebo-CEF-Ontologie überführt. Dabei wird ein Teil der TBox dieser Ontologie manuell gewartet, ein weiterer Teil wird automatisch generiert, da dieser abhängig von der jeweiligen Version des Industrial Protectors ist.

PoC-Systemontologie (Präfix poc)

Diese Ontologie enthält nur je ein Individuum pro Asset der PoC. Die Individuen gehören dabei alle zu der ebenfalls dort enthaltenen Klasse `Asset`. Diese Ontologie ist für statische Systeminformationen gedacht, die nicht dynamisch abgeleitet werden können oder sollen.

Lab-Systemontologie (Präfix lab)

Diese Ontologie enthält die TBox zur digitalen Repräsentation der Laborumgebung. Sie wurde manuell als unabhängige Testontologie erstellt, um verschiedene Aspekte der Vorarbeiten zu dem hier evaluierten Rahmenwerk

[1] https://rhebo.com/de/produkte/rhebo-industrial-protector/, zuletzt zugegriffen: 03.12.2020.

zu testen und enthält somit unter anderem die in Abschnitten 3.4.1.2 (Verwaltungsschale) und 3.4.2.1 (Effektive Konfiguration) verwendeten Klassen, Rollen und Restriktionen. Sie wurde in der Abschlussarbeit von Volz *[Vol18]* entwickelt und in der Abschlussarbeit von Kahles *[Kah19]* für den Einsatz zur NAC-Konfigurationsanalyse erweitert. Seitdem befindet sich die Ontologie durch ihren regen Gebrauch in verschiedenen Evaluations- und Testszenarien in ständiger Weiterentwicklung.

Weitere Ontologien

Es gibt weitere Ontologien, die in den Workflows der nächsten Abschnitte eingesetzt wurden. Diese besitzen jedoch nur wenig Informationsgehalt und werden für Einzelaufgaben und -erweiterungen eingesetzt.
So besteht die *CVE-Ontologie* (Präfix *cve*) nur aus der Klasse CVE und der abstrakten Rolle *cve*, um CVEs hinzuzufügen und mit den betroffenen Individuen zu verknüpfen.
Die in Abschnitt 3.7.6 vorkommende *General-Policy-Knowledge-Ontologie* (*GPK-Ontologie*, Präfix *gpk*) enthält im Vergleich nur Klassen und Rollen zum Ausdrücken von erlaubten Events und Zuständen, die für bisherige Evaluationen nützlich waren.
Ebenfalls in Abschnitt 3.7.6 vorkommend, enthält die *IntrusionDetection-Ontologie* (Präfix *id*) Klassen und Rollen zur Auszeichnung von mit potenziellen Angriffen in Verbindung stehenden Ereignissen.
Auch werden in den Abschnitten weitere Abbildungsontologien genutzt, die jedoch äußerst leichtgewichtig sind, da sie nur vereinzelte Konzepte aufeinander abbilden (z.B. lab:Asset auf iec-62443:Asset). Diese werden auch nicht extra in Überblick-Boxen der Analyse-Abschnitte gelistet.

3.7.2 Konfigurationsanalyse

Zuerst wird hier eine Konfigurationsanalyse beschrieben, welche Aufgaben definiert, die auch in den Untersuchungen zur Schwachstellen- und Konformitätsanalyse (vgl. Abschnitte 3.7.3) und 3.7.4 wiederzufinden sind. Ziel der hier

beschriebenen Konfigurationsanalyse ist die Identifikation von widersprüchlichen Konfigurationen zweier Firewalls. Sie entspricht im Wesentlichen der in *[Pat21]* erläuterten Analyse, die auf der Effektiven Konfiguration (vgl. Abschnitt 3.4.2.1) basiert, wurde hier jedoch mit dem SyMP-Rahmenwerk durchgeführt. Die Evaluation untersucht demnach den Einsatz des Rahmenwerks für komplexe Konfigurationsanalysen. Einen Überblick des Evaluationsaufbaus zeigt Info-Box 3.1.

Phasenbasierte Zusammenfassung der automatisierten Schritte

Phase	Inhalt
KC	Transformation von CLI-Datei-Export mittels X2Owl
KF/MC	Netzwerkzusammenführung und Modellerweiterungen wie Netzwerkhierarchien bilden; Netzwerkdienste zusammenführen
SKE	Effektive Konfiguration ableiten und weitere Modellerweiterung durchführen
DKE	Überschneidungen zwischen Flows ableiten
A	Markieren von Widersprüchen

Eingesetzte Ontologien, Regeln, Arbeiter und Aufgaben

Tabelle 3.10: test

Ontologien/Wissensdomänen	Lab-Systemontologie
Aufgaben	23
SWRL-Regeln	9
Python-Arbeiter	2
Java-Arbeiter	6

Info-Box 3.1: Überblick des Evaluationsaufbaus zur Konfigurationsanalyse.

In der Box ist in Kurzform festgehalten, welche Modellverarbeitungen und -erweiterungen in den einzelnen SyMP-Phasen vorgenommen wurden. Zudem enthält die Box Hinweise auf die eingesetzten Ontologien und Informationen

darüber, wie viele der jeweiligen Schritte von welcher Art von Arbeitern übernommen wurden. Auch alle weiteren Analyse-Evaluationsabschnitte enthalten eine solche Übersicht mit identischer Struktur.
In der **KC-Phase** werden zunächst die Gerätekonfigurationen als Kommandozeilenausgaben extrahiert und in endgerätespezifischen Input-Workflows zu Komponentenontologien konvertiert. Hierfür wurde die X2Owl-Bibliothek eingesetzt, wodurch jeder dieser Workflows nur aus einer Aufgabe besteht.
Daraufhin wurden im Workflow der **KF-** und **MC-Phase**, verschiedene Zusammenführungs- und Erweiterungsaufgaben definiert und abgearbeitet. Dieser Workflow ist in Abbildung 3.42 dargestellt. Da dies der erste hier präsentierte Evaluation-Workflow ist, wird dieser hier schrittweise erklärt. Später aufgeführte Workflows werden entsprechend kürzer erläutert. Wie in Abbildung 3.42 zu sehen, besteht die erste Aufgabe aus dem Hochladen der Komponentenontologien, das bei der Workflow-Ausführung vom Upload-Arbeiter übernommen wurde. Die nächste Aufgabe definiert das Zusammenführen der Komponentenontologien mithilfe der in Abschnitt 3.4.2 vorgestellten Integrationsstrategie. Ist dies erledigt, wird die nun zusammenhängende Systemontologie durch zwei parallel zu erfüllende Aufgaben weiterverarbeitet.
Die erste parallele Aufgabe ist die Zusammenführung von Netzwerkdienst-Individuen. Netzwerkdienst-Individuen liegen in Redundanz vor, da jede Komponente alle ihr bekannten Dienste meldet, wodurch eine bis zu 100%-ige Überschneidung zustande kommen kann. Diese Aufgabe wurde von einem Java-Modul übernommen, da die Zusammenführung mittels SWRL und Jena Rules nicht skalierbar war. Abbildung 3.43 zeigt, wie unterschiedlich die jeweiligen Implementierungen skalieren. Alle drei bekamen 16GB Arbeitsspeicher zugewiesen (25% des Gesamtarbeitsspeichers). Das Java-Modul war dabei bereits nach wenigen Sekunden erfolgreich und benötigte dabei weit weniger Speicher als die anderen Arbeiter. Gleichzeitig waren die beiden anderen Arbeiter selbst nach längerer Zeit und deutlich höherem Ressourcenverbrauch nicht erfolgreich.

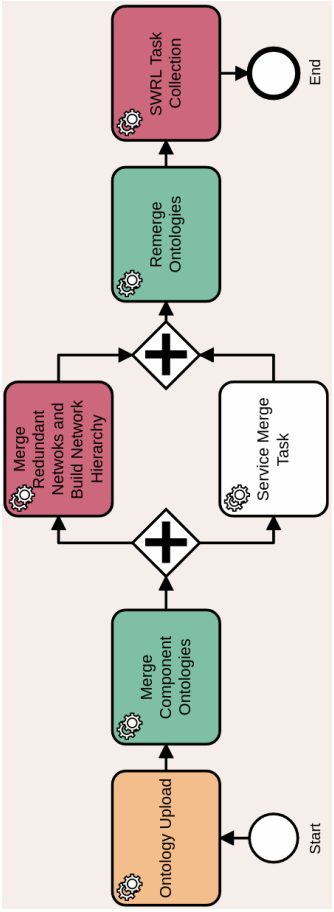


Abbildung 3.42: KF/MC-Workflow für die Laborumgebung.

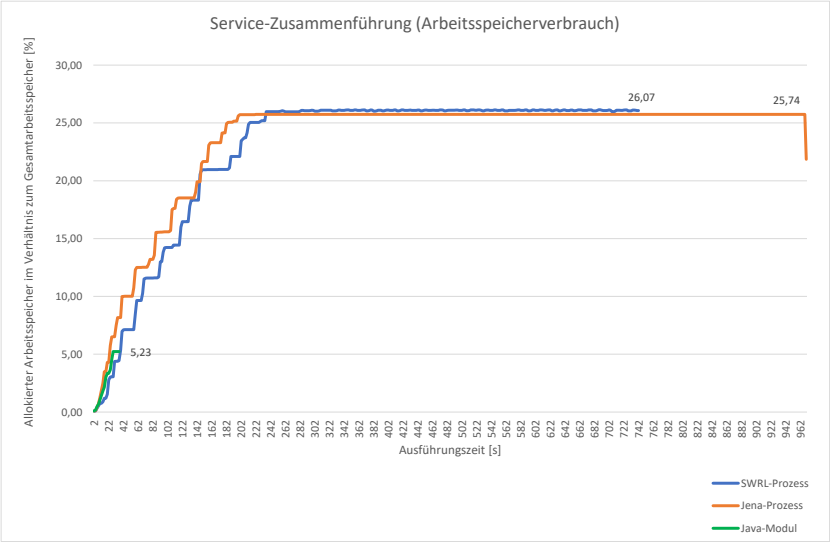


Abbildung 3.43: Messung des Arbeitsspeicherverbrauchs und der Laufzeit bei der Service-Zusammenführung durch SWRL-, Jena- und Java-Modul-Arbeiter.

Der mit *OWLAPI*[1]-Bibliothek erstellte SWRL- und der Jena-Arbeiter liefen wesentlich länger und konnten nicht erfolgreich terminieren. Sie hätten weiteren Speicher benötigt und warfen je einen Out-of-Memory-Fehler. Der Vergleich wurde auf der instanziierten Lab-Systemontologie ausgeführt, die vom erwähnten Java-Arbeiter auch im Rahmen des Workflows geladen wird. Diese enthält 95001 Axiome und 9011 lab:Service-Individuen und ist somit verhältnismäßig klein.
Im Rahmen der zweiten parallelen Aufgabe führte der SWRL-Arbeiter, welcher ebenfalls auf der OWLAPI-Bibliothek basiert, mithilfe des Pellet-Reasoners die in Gleichungen 3.2 und 3.3 definierten Regeln aus (vgl. Gleichung 3.2). Die erste der beiden Regeln führt gleiche IPv4-Netzwerke zusammen. Da in der Laborumgebung davon ausgegangen werden kann, dass eine IPv4-Netzwerkadresse

[1] http://owlcs.github.io/owlapi/, zuletzt zugegriffen: 20.02.2021.

nicht für verschiedene Netzwerke verwendet wird, wurde hier die Netzwerkadresse als Indikator für das Identifizieren von redundanten Netzen verwendet.

$$\begin{aligned}&\text{lab:Network}(?n1) \wedge \text{lab:Network}(?n2) \wedge \mathit{lab{:}prefixBits}(?n1,?pb1)\\&\wedge \mathit{lab{:}prefixBits}(?n2,?pb2) \wedge \mathit{swrlb{:}equal}(?pb1,?pb2)\\&\wedge \mathit{lab{:}ipV4Address}(?n1,?a1) \wedge \mathit{lab{:}ipV4Address}(?n2,?a2)\\&\wedge \mathit{swrlb{:}equal}(?a1,?a2) \rightarrow \mathit{owl{:}sameAs}(?n1,?n2)\end{aligned} \tag{3.2}$$

Wäre die Annahme nicht korrekt, könnte man diese Regel problemlos austauschen, ohne weitere Anpassungen am Modellverarbeitungsprozess vorzunehmen.
Die zweite Regel errechnet die IPv4-Netzwerkhierarchie und definiert diese über die *subnet*-Rolle (vgl. Gleichung 3.3).

$$\begin{aligned}&\text{lab:Network}(?n1) \wedge \text{lab:Network}(?n2) \wedge \mathit{lab{:}ipV4Address}(?n1,?na1)\\&\wedge \mathit{lab{:}ipV4Address}(?n2,?na2) \wedge \mathit{lab{:}prefixBits}(?n1,?np1)\\&\wedge \mathit{lab{:}prefixBits}(?n2,?np2) \wedge \mathit{swrlb{:}lessThan}(?np2,?np1)\\&\wedge \mathit{swrlb{:}subtract}(?diff,?np1,?np2) \wedge \mathit{swrlb{:}pow}(?divisor,2,?diff)\\&\wedge \mathit{swrlb{:}divide}(?res1,?na1,?divisor) \wedge \mathit{swrlb{:}equal}(?na2,?res1)\\&\rightarrow \mathit{lab{:}subnet}(?n2,?n1)\end{aligned} \tag{3.3}$$

Durch die darauffolgende Gateway-Definition werden die beiden parallelen Aufgaben synchronisiert. Um die erzeugten Systemmodellversionen wieder zusammenzuführen, wurde eine Remerge-Aufgabe nach dem Gateway eingefügt. Dieser Aufgabe nimmt sich ein entsprechender Camunda-interner, ebenfalls auf der OWLAPI-Bibliothek basierender, Java-Arbeiter an, der die beiden Modelle zusammenführt. Dafür kopiert der Arbeiter lediglich alle Aussagen eines der Modelle in das andere Modell, die dort nicht bereits vorliegen. Weil die beiden Modelle dieselben TBoxen nutzen und die parallel ausgeführten Aufgaben keine Abhängigkeiten voneinander haben, kann das Modell dadurch nicht inkonsistent werden.

Der letzten Aufgabe des Workflows folgend, wurden noch zwei weitere, netzwerkbezogene SWRL-Regeln ausgeführt, die in den Gleichungen 3.4 und 3.5 zu finden sind. Die erste Regel definiert die Zuweisung von IPv4-Schnittstellen zu übergeordneten IPv4-Netzwerken (vgl. Gleichung 3.4).

$$\text{lab:IpV4Interface}(?i) \wedge \textit{lab:isInNetwork}(?i, ?n) \wedge \textit{lab:parentNetwork}(?n,?pn) \rightarrow \textit{lab:isInNetwork}(?i, ?pn) \tag{3.4}$$

Um spätere Abfragen zusätzlich zu vereinfachen, werden durch die zweite Regel auch Assets den IPv4-Netzwerken direkt zugewiesen (vgl. Gleichung 3.5).

$$\text{lab:Asset}(?a) \wedge \text{lab:Network}(?n) \wedge \text{lab:IpV4Interface}(?i) \wedge \textit{lab:isInNetwork}(?i, ?n) \wedge \textit{lab:interface}(?a,?i) \rightarrow \textit{lab:isInNetwork}(?a, ?n) \tag{3.5}$$

Dafür verwendet die Regel aus Gleichung 3.5 das Ergebnis der vorherigen Regel (aus Gleichung 3.4). Dies wird vom Pellet-Reasoner automatisch unterstützt. Die beiden Regeln hätten auch bereits in der parallelen SWRL-Aufgabe definiert werden können. Hier sollten jedoch sowohl die parallele Verarbeitung, als auch das Weiterverarbeiten von zusammengeführten Systemontologieversionen evaluiert werden. Zudem sind die Regeln aus Gleichungen 3.4 und 3.5 anders zu klassifizieren als die vorherigen Regeln, da sie Beziehungen ableiten, welche implizit schon modelliert sind (sozusagen eine Art kognitive Abkürzung). Dadurch werden zukünftige Ableitungsschritte vereinfacht.
In der **SKE-Phase** (vgl. Abbildung 3.44) findet das Erzeugen der Effektiven Konfiguration statt. Diese kann entweder bereits bei der Informationsextraktion von der Originalrepräsentation der NAC-Regeln oder von einer bereits in eine Ontologie transformierten Repräsentation der NAC-Regeln abgeleitet werden. Da die Regelketten der pfSense-Firewalls jedoch bereits Teil der Firewall-Komponentenmodelle sind, wurde in dieser Evaluation letztere

Strategie gewählt.

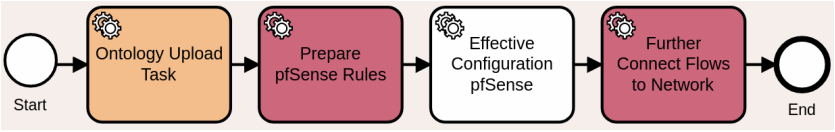


Abbildung 3.44: SKE-Workflow für die Laborumgebung zur NAC-Konfigurationsanalyse.

Zunächst wurde die Regelrepräsentation durch eine SWRL-Regel erweitert (vgl. Gleichung 3.6). Diese fügt eine direkte Beziehung zwischen dem lab:FirewallConfig-Individuum, welches die gesamte Regelkettenkonfiguration repräsentiert, und allen enthaltenen Regeln hinzu.

$$\begin{aligned} &\text{lab:PfConfiguration}(?\mathit{config}) \wedge \mathit{lab{:}containsRule}(?\mathit{config}, ?\mathit{firstRule}) \\ &\wedge \mathit{lab{:}pfNextRule}(?\mathit{firstRule}, ?\mathit{secondRule}) \\ &\rightarrow \mathit{lab{:}containsRule}(?\mathit{config}, ?\mathit{secondRule}) \end{aligned} \tag{3.6}$$

Nach diesem Vorbereiten der Regeln folgte die Aufgabe zur Erzeugung der Effektiven Konfiguration. Der Arbeiter (hier ein Python-Arbeiter), der diese Aufgabe abarbeitet, könnte auch selbst die in Gleichung 3.6 gezeigte Vorbereitung der NAC-Regeln übernehmen. Das ist deswegen relevant, weil nur die Vorbereitung und Erzeugung der Effektiven Konfiguration NAC-Konfigurationssprachen-spezifisch sind und diese Spezialisierung somit weiter auf eine Aufgabe reduziert werden könnte. Hätte man also mehr als nur eine NAC-Art, wäre dies der einzige Punkt an dem die Aufgaben nicht Konfigurationssprachen-agnostisch sind und man würde die entsprechenden anderen Aufgaben zur Ableitung der Effektiven Konfiguration parallel zu den beiden pfSense-Aufgaben definieren.

Zur einfacheren Definition von weiteren Analysen wurden zudem alle IPv4-Flows durch eine SWRL-Regel pro IpV4Flow-Art (erlaubte und abgelehnte Flows) und je Quell- und Zielnetz mit den zugehörigen Subnetzen verknüpft (vgl. Gleichung 3.7 für die Quellnetzwerke der lab:AllowedIpV4Flow-Individuen).

$$\begin{aligned} &\text{lab:AllowedIpV4Flow}(?aif) \wedge lab{:}srcNetwork(?aif, ?srcNet) \\ &\wedge lab{:}parentNetwork(?subNet, ?srcNet) \rightarrow lab{:}srcNetwork(?aif, ?subNet) \end{aligned} \tag{3.7}$$

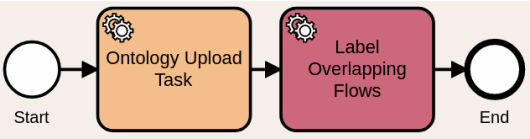


Abbildung 3.45: DKE-Workflow für die Laborumgebung zur NAC-Konfigurationsanalyse.

In der **DKE-Phase** wurden mithilfe eines SWRL-Arbeiters, sich bzgl. ihrer Portspannen überschneidende, lab:AllowedTcpFlow- und lab:DisallowedTcpFlow-Individuen markiert, um spätere Regeln und Abfragen zum Finden von Widersprüchen deutlich zu vereinfachen (vgl. Abbildung 3.45). Gleichungen 3.8 und 3.9 zeigen entsprechende Regeln, die prüfen, ob sich die Quellportspannen eines lab:AllowedTcpFlow-Individuums und eines lab:DisallowedTcpFlow-Individuums überschneiden.
Wobei *lab:overlapsWith* eine symmetrische, abstrakte Rolle ist. D.h., wenn gilt *lab:overlapsWith*(?*aSrcPR*, ?*dSrcPR*), gilt auch *lab:overlapsWith*(?*dSrcPR*, ?*aSrcPR*). Bei den Ableitungen aus Gleichungen 3.8 und 3.9 ist wichtig zu erkennen, dass es sich nicht nur um Flows einer Firewall handelt. Hier findet also theoretisch sowohl eine NAC-Instanz-interne als auch -übergreifende Modellerweiterung statt. Praktisch gesehen ist es jedoch ausschließlich eine NAC-Instanz-übergreifende Modellerweiterung, da die Erzeugung der Effektiven Konfiguration bereits für überschneidungsfreie Flow-Deklarationen sorgt und somit keine NAC-Instanz-internen Überlappungen über die Effektive

Konfiguration gefunden werden.

$$\begin{aligned}
&\text{lab:AllowedTcpFlow}(?atf) \wedge \text{lab:DisallowedTcpFlow}(?dtf) \\
&\wedge lab{:}srcPortRange(?atf, ?aSrcPR) \wedge lab{:}srcPortRange(?dtf, ?dSrcPR) \\
&\wedge lab{:}maxPort(?aSrcPR, ?aSrcPMax) \wedge lab{:}minPort(?dSrcPR, ?dSrcPMin) \\
&\wedge lab{:}minPort(?aSrcPR, ?aSrcPMin) \wedge lab{:}maxPort(?dSrcPR, ?dSrcPMin) \\
&\wedge swrlb{:}greaterThanOrEqual(?aSrcPMax, ?dSrcPMin) \\
&\wedge swrlb{:}lessThanOrEqual(?aSrcPMin, ?dSrcPMax) \\
&\rightarrow lab{:}overlapsWith(?aSrcPR, ?dSrcPR)
\end{aligned} \tag{3.8}$$

$$\begin{aligned}
&\text{lab:AllowedTcpFlow}(?atf) \wedge \text{lab:DisallowedTcpFlow}(?dtf) \\
&\wedge lab{:}dstPortRange(?atf, ?aDstPR) \wedge lab{:}dstPortRange(?dtf, ?dDstPR) \\
&\wedge lab{:}maxPort(?aDstPR, ?aDstPMax) \wedge lab{:}minPort(?dDstPR, ?dDstPMin) \\
&\wedge lab{:}minPort(?aDstPR, ?aDstPMin) \wedge lab{:}maxPort(?dDstPR, ?dDstPMax) \\
&\wedge swrlb{:}greaterThanOrEqual(?aDstPMax, ?dDstPMin) \\
&\wedge swrlb{:}lessThanOrEqual(?aDstPMin, ?dDstPMax) \\
&\rightarrow lab{:}overlapsWith(?aDstPR, ?dDstPR)
\end{aligned} \tag{3.9}$$

Die **Analysis-Phase** enthält bereits ein SWRL-basiertes Markieren von trivialen Widersprüchen bzw. Inkonsistenzen (vgl. Abbildung 3.46). Beispielhaft wird im Folgenden eine einfache Regel aufgezeigt, die einen Erlauben- und einen Ablehnen-TCP-Flow als Konflikt kennzeichnet, falls sie zum selben IPv4-Quell- und -Zielnetz gehören und die erlaubte TCP-Port-Spanne eine Teilmenge der abgelehnten ist (vgl. Gleichung 3.10).

$$
\begin{aligned}
&\text{lab:AllowedTcpFlow}(?atf) \wedge \text{lab:DisallowedTcpFlow}(?dtf) \\
&\wedge \text{lab:AllowedIpV4Flow}(?aif) \wedge \text{lab:DisallowedIpV4Flow}(?dif) \\
&\wedge \mathit{lab{:}usesFlow}(?atf, ?aif) \wedge \mathit{lab{:}usesFlow}(?dtf, ?dif) \\
&\wedge \mathit{lab{:}srcNetwork}(?aif, ?srcNet) \wedge \mathit{lab{:}srcNetwork}(?dif, ?srcNet) \\
&\wedge \mathit{lab{:}dstNetwork}(?aif, ?dstNet) \wedge \mathit{lab{:}dstNetwork}(?dif, ?dstNet) \\
&\wedge \mathit{lab{:}portRange}(?atf, ?aSrcPR) \wedge \mathit{lab{:}portRange}(?dtf, ?dSrcPR) \\
&\wedge \mathit{lab{:}overlapsWith}(?aSrcPR, ?dSrcPR) \rightarrow \mathit{lab{:}inConflictWith}(?dtf, ?atf)
\end{aligned}
\tag{3.10}
$$

Die abstrakte Rolle *lab:inConflictWith* ist ebenfalls symmetrisch.

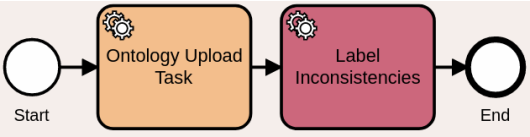


Abbildung 3.46: Analysis-Workflow für die Laborumgebung zur NAC-Konfigurationsanalyse.

Im Rahmen einer **analytischen Abfrage** wurden die entsprechenden Konflikte daraufhin mit Hilfe von SPARQL extrahiert (vgl. Listing 3.1).
Auf diese Weise wurden verschiedene, sich widersprechende NAC-Regeln der beiden pfSense-Firewalls aus der Laborumgebung zuverlässig, automatisiert identifiziert. Beispielsweise wurde ein Konflikt zwischen, der in Listing 3.2 zu sehenden Regel aus Pfsense 1 (Firewall zwischen Enterprise-Netz, DMZ und Internet) und der in Listing 3.3 zu sehenden Regel aus Pfsense 2 (Firewall zwischen Feld- und Enterprise-Netz) gefunden.

```
SELECT ?firewall1 ?firewall2 ?f1 ?f2 WHERE {
  ?firewall1 lab:firewallConfig ?config1.
  ?config1 lab:flow ?f1.
  ?firewall2 lab:firewallConfig ?config2.
  ?config2 lab:flow ?f2.
  ?tcpFlow1 :usesFlow ?f1.
  ?tcpFlow2 :usesFlow ?f2.
  ?tcpFlow1 :inConflictWith ?tcpFlow2.
}
```

Listing 3.1: SPARQL-Abfrage, welche die in Konflikt stehenden Firewalls und Flows, zurückliefert.

```
pass in quick on em1 inet proto tcp from 172.16.106.0/24 to
    172.16.110.0/24 port 80,443,8080 flags S/SA keep state
    label "Allow HTTP(S) from field to internet"
```

Listing 3.2: Beispielregel aus Firewall Pfsense 1.

```
block drop in quick on em1 inet proto tcp from
    172.16.106.0/24 to 172.16.110.0/24 port = 80 flags S/SA
    keep state label "Deny HTTP from field network to
    internet"
```

Listing 3.3: Beispielregel aus Firewall Pfsense 2.

Das entsprechende Ergebnis der Abfrage aus Listing 3.1 ist in Abbildung 3.47 dargestellt.
An dieser Stelle soll noch einmal betont werden, dass alle Schritte nach Erzeugung der Effektiven Konfiguration Hersteller-, Modell- und vor allem Konfigurationssprachen-unabhängig sind. Dies kommt insbesondere auch der Konformitätsanalyse zugute. Darauf wird im nächsten Abschnitt eingegangen.

	firewall1	firewall2	f1	f2
1	http://iosb.fraunhofer.de/ICS-Security/pfsense1#fw-ids-office_silab-ip_iosb_fraunhofer_de	http://iosb.fraunhofer.de/ICS-Security/pfsense2#fw-ids-festo_silab-ip_iosb_fraunhofer_de	:allowedIpV4Flow2	:disallowedIpV4Flow12
2	http://iosb.fraunhofer.de/ICS-Security/pfsense2#fw-ids-festo_silab-ip_iosb_fraunhofer_de	http://iosb.fraunhofer.de/ICS-Security/pfsense1#fw-ids-office_silab-ip_iosb_fraunhofer_de	:disallowedIpV4Flow12	:allowedIpV4Flow2

Abbildung 3.47: Ergebnisse der Abfrage von Konflikten im Rahmen der NAC-Konfigurationsanalyse.

3.7.3 Konformitätsanalyse

In diesem Abschnitt werden einige Konformitätsanalysen gezeigt, die sich sowohl mit der Konfigurationsanalyse als auch untereinander Modellerweiterungen teilen. Einen Überblick des Evaluationsaufbaus zeigt Info-Box 3.2. Anders als bei den anderen Analyse-Evaluationen wurde diese Evaluation auf Basis der automatisierten Analyse unter Einsatz des SyMP-Hubs, des Analyse-Clients und der erweiterten Analyse-Engine mit automatisierter Workflow-Generierung durchgeführt.
Die Konfiguration im SyMP-Client kann Abbildung 3.48 entnommen werden. Diese enthält Analyseregeln, die eine regelmäßig im industriellen Umfeld eingesetzte offene Richtlinie prüfen, nämlich die Trennung von ICS- und Unternehmensnetzwerk. Unter den Regeln ist die Anforderung IEC 62443-3-2 ZCR-3.2 (vgl. Abschnitt 3.4.3.1), die besagt, dass Assets des Unternehmens- und ICS-Bereichs getrennt werden müssen. Darüber soll vermieden werden, dass verschiedene Bedrohungen aus den Unternehmensnetzwerken zu Bedrohungen der ICS-Netzwerke werden.

Phasenbasierte Zusammenfassung der automatisierten Schritte

Phase	Inhalt
KC	Transformation von CLI-Datei-Export mittels X2Owl
KF/MC	Netzwerkzusammenführung und Modellerweiterungen wie Netzwerkhierarchien bilden; Netzwerkdienste zusammenführen
SKE	Redundante Netzwerkzusammenführung und Modellerweiterungen (Redundanz ist gewünscht, siehe Beschreibung); Firewalls klassifizieren
DKE	-
A	Dual-Homed-Geräten und Whitelist-Firewalls klassifizieren

Eingesetzte Ontologien, Regeln, Arbeiter und Aufgaben

Ontologien/Wissensdomänen	Lab-Systemontologie, ICS-ATT&CK®-Ontologie, IEC-62443-Ontologie
Aufgaben	17 (redundante nicht mitgezählt)
SWRL-Regeln	6 (zzgl. einer Jena-Regel)
Python-Arbeiter	1
Java-Arbeiter	6

Info-Box 3.2: Überblick des Evaluationsaufbaus zur Konformitätsanalyse

Für diese Evaluation gilt die Annahme, dass die Anforderung IEC 62443-3-2 ZCR-3.2 genau dann als erfüllt gilt, wenn

- außer Firewalls keine weiteren Komponenten die Feld- und die Enterprise-Zone verbinden.
- alle relevanten Firewalls Whitelist-Firewalls sind.

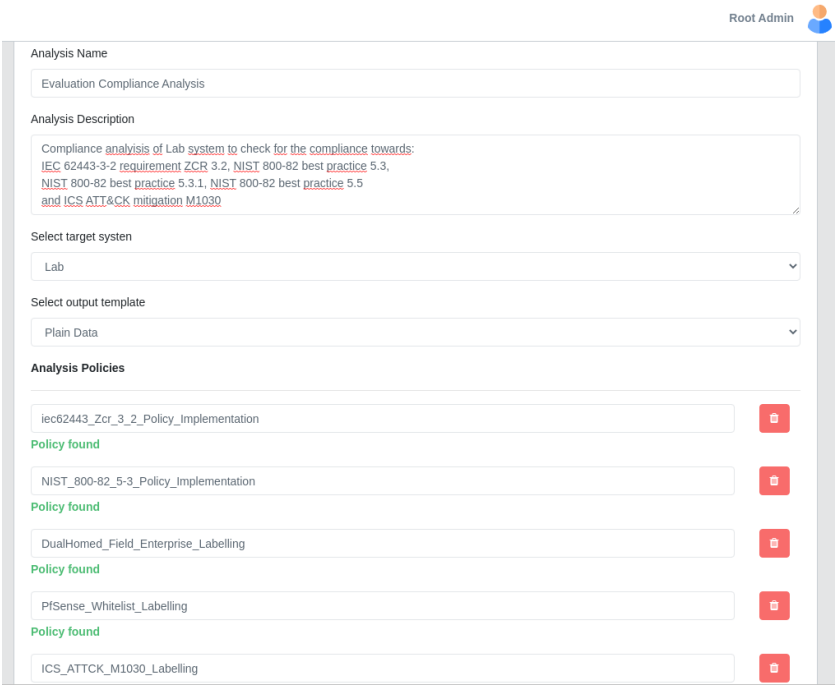


Abbildung 3.48: Konfiguration des SyMP-Clients zur Konformitätsanalyse-Evaluation.

Dies bedeutet auch, dass aus der Konformität zu den folgenden, ebenfalls im Rahmen der Analyse abgefragten Richtlinien, die Konformität zu IEC 62443-3-2 ZCR-3 folgt:

- **NIST 800-82 5.5**, wonach die Grundeinstellung der Firewall-Regelkette auf „alles ablehnen“ gesetzt ist und die Firewall somit eine Whitelist-Firewall darstellt.
- **NIST 800-82 5.3.1**, die verlangt, keine Geräte (außer NAC-Geräte) zu erlauben, die sowohl im industriellen als auch im Unternehmensnetz sind.

- **NIST 800-82 5.3** und **ICS ATT&CK® M1030**, welche beide eine Netzwerkseparierung zwischen dem industriellen und dem Unternehmensnetz fordern.

Diese Voraussetzungen lassen sich zwar beliebig verschärfen, sollen an dieser Stelle jedoch ausreichen, um die Einsetzbarkeit der Konformitätsanalyse mittels SyMP und die Ausnutzung von Synergien zu evaluieren.

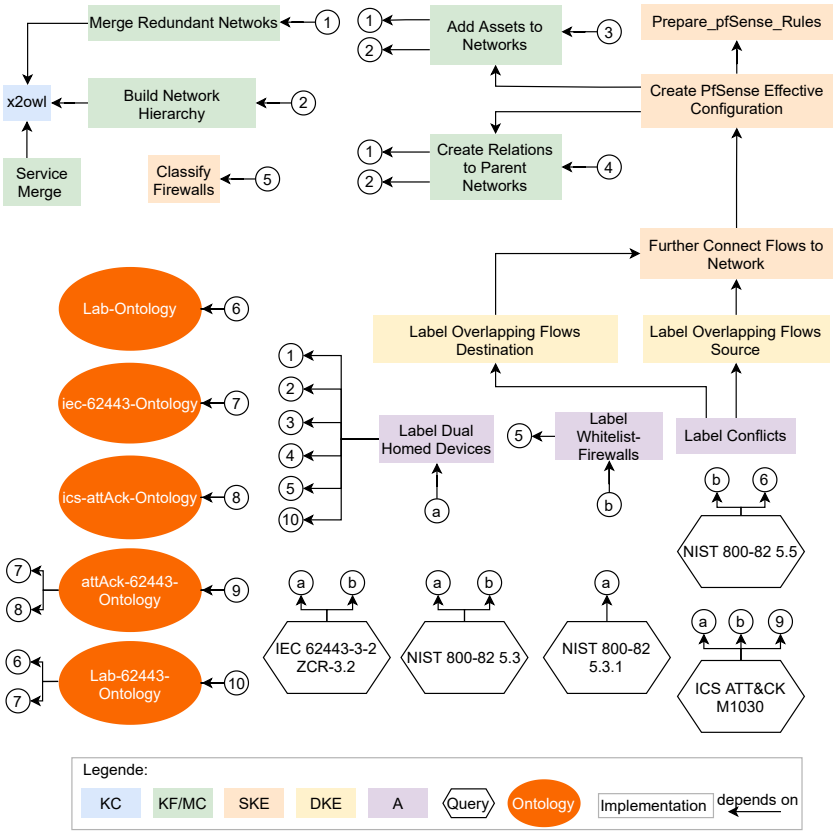


Abbildung 3.49: Konfiguration zur Konformitätsanalyse-Evaluation von Analyseregelumsetzungen und ihren Abhängigkeiten.

Um die Workflow-Generierung angemessen zu evaluieren, wurde die Information darüber, welche Modellerweiterungsschritte bereits in der Systemmodell-Erzeugung eingesetzt wurden (vgl. Abschnitt 3.6), aus der entsprechenden Datenbank des SPC gelöscht, damit die Analyse-Engine gezwungen ist, diese in der SKE-Phase einzubinden. So kam eine deutlich höhere Abhängigkeitskomplexität zustande, welche die Workflow-Generierung bewältigen musste. Die Konfiguration der Modellerweiterungsschritte bezüglich ihrer Abhängigkeiten ist in Abbildung 3.49 dargestellt. Diese Abbildung zeigt damit, welche Analyseregelumsetzungen zur Evaluation auf dem Hub angelegt wurden. Die eingekreisten Zahlen und Buchstaben in der Abbildung dienen der Reduktion von sich kreuzenden Strichen, die durch die Abhängigkeitspfeile entstehen würden. In der Abbildung sind also sowohl verschiedene Modellverarbeitungsschritte (Java-Module, SWRL- oder Jena-Regeln), Ontologieabhängigkeiten und Abfragen, als auch deren Abhängigkeiten zueinander dargestellt. Zusätzlich sind die Modellverarbeitungsschritte entsprechend der konfigurierten Zuordnung zu den SyMP-Phase eingefärbt. Zu sehen sind in der Abbildung sowohl die aus der Konfigurationsanalyse (vgl. Abschnitt 3.7.2) bereits bekannten als auch die für die Konformitätsanalyse hinzugefügten Umsetzungen. Letztere werden im Folgenden beschrieben.
Da alle pfSense-Firewalls Whitelist-Firewalls sind, können diese mithilfe der Regel in Gleichung 3.11 direkt als solche klassifiziert werden.

$$\begin{aligned} &\text{lab:Firewall}(?f) \wedge \text{lab:PfConfiguration}(?c) \wedge \textit{lab:firewallConfig}(?f,?c) \\ &\rightarrow \text{lab:WhitelistFirewall}(?f) \end{aligned} \tag{3.11}$$

Eine Abfrage, welche alle Firewalls des Modells liefert, die keine Whitelist-Firewalls sind (vgl. Listing 3.5 Zeilen 1, 5, 6 und 9), gibt demnach Widersprüche zu NIST 800-82 5.5 zurück und prüft somit die Konformität des Modells zu dieser Richtlinie.
Des Weiteren wurden alle Nicht-Firewall-Assets, die sowohl in der Enterprise- als auch in der Field-Zone liegen, als iec-62443:DualHomedDevice klassifiziert (vgl. Listing 3.4).
Diese Regel ist durch den *noValue*-Ausdruck nicht monoton, weshalb hier eine Jena-Regel statt einer SWRL-Regel gewählt wurde. Die Syntax kann dabei,

ähnlich zu der von SPARQL, als eine Konjunktion von Tripeln interpretiert werden, wobei *noValue* einer Negation gleichkommt. Eine Abfrage nach solchen Assets liefert Widersprüche zu NIST 800-82 5.3.1 (vgl. Listing 3.5 Zeilen 1, 2 und 9) und kann so die Konformität des Modells zu dieser Richtlinie prüfen. Führt sowohl die Abfrage nach Blacklist-Firewalls als auch nach Dual-Homed-Geräten zu keinen Ergebnissen (vgl. Listing 3.5 ohne Zeile 8), kann zudem geschlussfolgert werden, dass das System sowohl zu NIST 800-82 5.3 als auch zu IEC 62443-3-2 ZCR-3 konform ist.

```
(?a a lab:Asset), noValue(?a a lab:Firewall),
(?a lab:isInNetwork ?n1), (?a lab:isInNetwork ?n2),
(?n1 iec-62443:inZone iec-62443:FieldZone),
(?n2 iec-62443:inZone iec-62443:EnterpriseZone),
noValue(?n1 lab:parentNetwork ?n2),
noValue(?n2 lab:parentNetwork ?n2)
->(?a a iec-62443:DualHomedDevice)
```

Listing 3.4: Jena-Regel zum Klassifizieren von Dual-Homed-Geräten zwischen Field- und Enterprise-Zone.

```
1  SELECT ?asset WHERE {
2    {?asset a iec-62443:DualHomedDevice.}
3    UNION
4    {
5      ?asset a lab:Firewall.
6      MINUS{?asset a lab:WhitelistFirewall.}
7    }
8    MINUS{?asset ics-attAck:isA ics-attAck:M1030.}
9  }
```

Listing 3.5: SPARQL-Abfrage, welche für die ICS-ATT&CK®-Konformitätsanalyse eingesetzt wurde. Sie enthält alle Abfrageelemente der anderen Konformitätsabfragen, daher werden diese nicht extra aufgeführt.

Auch könnte man sagen, dass damit ebenfalls ICS ATT&CK® M1030 abgehandelt ist. Um aber mehr Komplexität durch die Hinzunahme der ics-attAck-Ontologieabhängigkeit zu erzeugen, wurde für M1030 noch die zusätzliche Restriktion eingefügt, dass ein widersprüchliches Asset nicht selektiert werden soll, wenn es anderweitig als M1030-Mitigation deklariert wurde. Diese Abfrage ist in Listing 3.5 zu sehen.
Vor Durchführung der Analyse, wurden von der Analyse-Engine Workflows der **SKE-** und **Analysis-Phase** generiert und auf der Camunda-Plattform installiert. Abbildungen 3.50 und 3.51 zeigen das Ergebnis, wie es im Cockpit der Camunda-Plattform eingesehen werden kann.
Für die **DKE-Phase** wurde im Rahmen der aufgeführten Konformitätsanalysen kein entsprechender Schritt generiert, da, wie in der Konfiguration aus Abbildung 3.49 zu sehen ist, für diese Phase keine Schritte konfiguriert wurden. Hingegen wurden Schritte wie *Merge Redundant Networks* in den SKE-Workflow eingebunden, da der SPC der Analyse-Engine wie erwartet nicht mitteilte, dass die Abhängigkeiten bereits erfüllt wurden. Dies und die zu sehenden Optimierungen, wie Parallelisierung, Redundanzvermeidung und Zusammenführung von SWRL-Regeln, entsprechen den Erwartungen und zeigen die Umsetzbarkeit des Konzeptes.

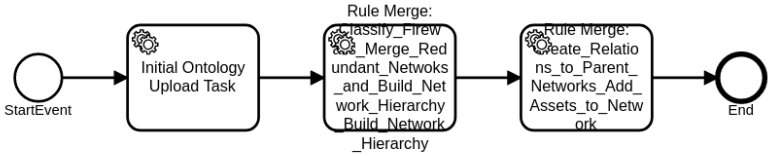


Abbildung 3.50: Von der Analyse-Engine generierter und in Camunda installierter SKE-Workflow aus der Konformitätsanalyse (ohne optische Anpassung, da nur für die Ausführungsumgebung relevant).

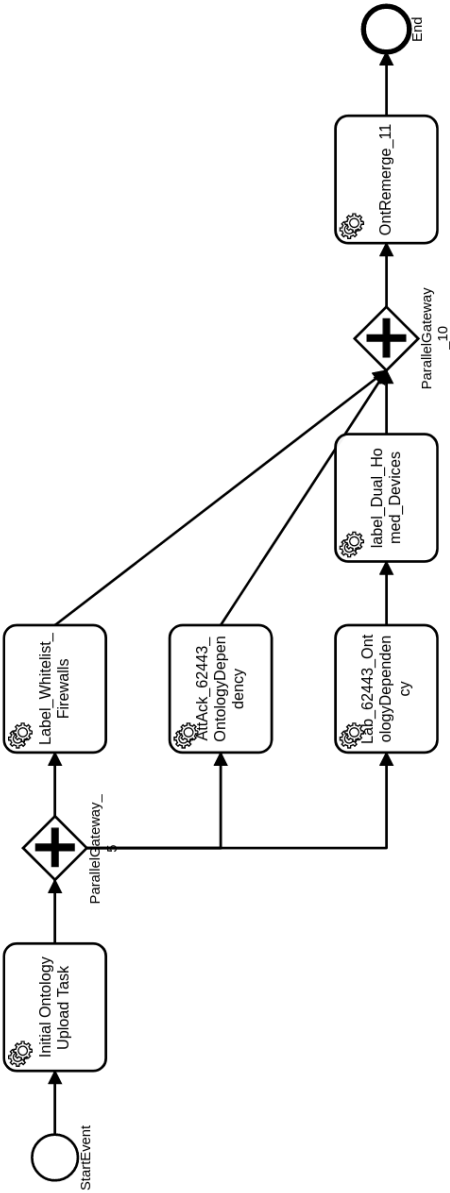


Abbildung 3.51: Von der Analyse-Engine automatisch generierter und in Camunda installierter Analysis-Workflow aus der Konformitätsanalyse (ohne optische Anpassung, da nur für die Ausführungsumgebung relevant).

Die Ausführung der Workflows auf der Plattform wurde von der Analyse-Engine automatisch verwaltet. Außerdem führte die Engine vollautomatisch die konfigurierten Abfragen auf dem vorberechneten, analysespezifischen Systemmodell aus und meldete, dass die Resultate zur Verfügung stehen an den SyMP-Client, der diese dann als Ergebnisbericht anzeigte.

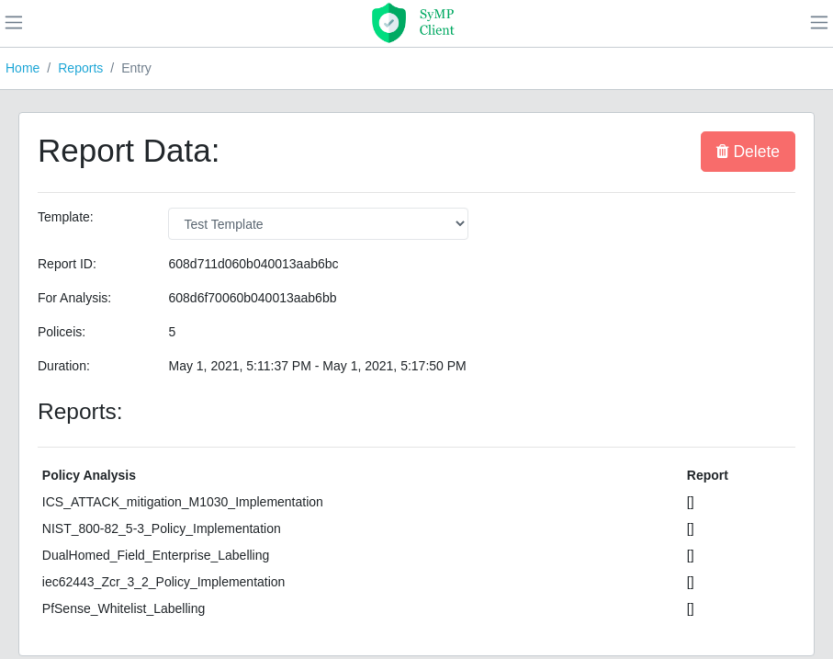


Abbildung 3.52: Anzeige des Ergebnisberichts im SyMP-Client mit Analyseergebnissen ohne Widersprüche.

In Abbildung 3.52 ist der Ergebnisbericht der hier vorgestellten Konformitätsanalyse zu sehen. Da die Laborumgebung zu den aufgeführten Richtlinien konform ist, enthält das Ergebnis auch keine gefundenen Widersprüche. Dies ist für die Evaluation nicht allzu aussagekräftig.

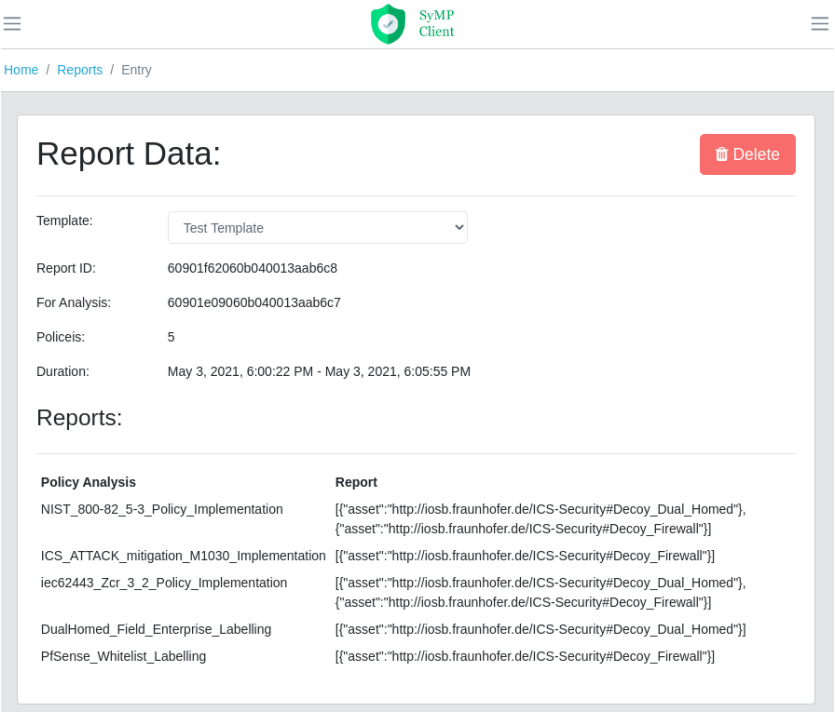


Abbildung 3.53: Anzeige des Ergebnisberichts im SyMP-Client mit als nicht konform identifizierten Individuen.

Darum wurde das Systemmodell in einem zweiten Durchlauf um zwei Individuen erweitert. Das erste Individuum `lab:Decoy_Dual_Homed` wurde als ics-attAck:M1030 und lab:HardwareDevice (Subklasse von lab:Asset) deklariert und mittels *lab:isInNetwork* sowohl zu einem Netzwerkindividuum der Zone `iec-62443:FieldZone` als auch einem der Zone `iec-62443:EnterpriseZone` hinzugefügt. Das zweite Individuum `lab:Decoy_Firewall` wurde der Klasse lab:Firewall zugewiesen, erhielt jedoch keine pfSense-Konfiguration.
Nun, da nicht konforme Gereätepräsentationen eingefügt wurden, fand die Analyse diese als Widersprüche zu den Richtlinien (vgl. Abbildung 3.53). Wie an

den Ergebnissen zu erkennen ist, funktioniert die SyMP-Werkzeugumgebung wie erwartet. Außer den hier beschriebenen, waren keine weiteren manuellen Vorgänge nötig.

3.7.4 Schwachstellenanalyse

Ziel dieser Analyse ist die Identifikation von verwundbaren Software-Paketen, inklusive der Verwundbarkeitsidentifikatoren als CVE-IDs. Einen Überblick des Evaluationsaufbaus zeigt Info-Box 3.3, bei der zwei alternative Schwachstellenanalysen unterschieden werden.
Im Rahmen des in Abschnitt 4.6.3 beschriebenen Sicherheitskonzeptes, wird ein stufenbasiertes Sicherheitsmanagement von Geräten verwendet, das in jeder Stufe beliebige Überprüfungen ermöglicht und bei Erfolg oder Misserfolg entsprechende automatisierte Vorfallreaktionen (resp. Stufenübergänge) durchführt. Eine dieser beliebigen Überprüfungen ist die in der Phasenübersicht zu sehende Alternative 1. Sie übernimmt die Suche nach bekannten Schwachstellen in der Software eines Gerätes und zeigt somit auch ein mögliches Zusammenspiel der in dieser Dissertation beschriebenen Konzepte. Das Auslesen der relevanten Geräteinformationen funktioniert dabei über OPC UA mithilfe von X2Owl. Daraufhin werden die Informationen von einer Java-Anwendung direkt mit einer Offline-Datenbank abgeglichen. Das Ergebnis wird verwendet, um dem Netzwerkmanagement mitzuteilen, ob der Gerätestatus positiv angepasst werden kann, also keine Schwachstellen gefunden wurden, oder nur durch den Administrator geändert werden kann, weil Schwachstellen vorliegen. Wie in der Phasenübersicht zu erkennen ist, macht Alternative 1 keinen Gebrauch von den Phasen KF, MC, SKE und DKE, da nur auf den Informationen des entsprechenden Komponentenmodells gearbeitet wird.
Der komplexere Ablauf ist daher Alternative 2, bei der die gesamte Systemrepräsentation auf bekannte Softwareschwachstellen untersucht wird. Dabei wird die bereinigte Systemontologie aus der Konfigurationsanalyse (vgl. Abschnitt 3.7.2) eingesetzt, bzw. die entsprechenden Workflows aus den Phasen KC, KF und MC. Die Abbildungslogik von Alternative 1 entspricht der von Alternative 2 (wenn auch mit einer alternativen Implementierung). Folglich wird in diesem Abschnitt nur die umfassendere Alternative 2 näher erläutert.

Phasenbasierte Zusammenfassung der automatisierten Schritte

Phase	Inhalt
KC	*Alternative 1:* Transformation von OPC UA Export mittels X2Owl *Alternative 2:* Transformation von CLI-Datei-Export mittels X2Owl
KF/MC	*Alternative 1:* - *Alternative 2:* Netzwerkzusammenführung und Modellerweiterungen wie Netzwerkhierarchien bilden; Netzwerkdienste zusammenführen
SKE	*Alternative 1:* - *Alternative 2:* CVE-Ontologie-Mapping
DKE	-
A	*Alternative 1:* CVE-Abgleich mit *SDN-AIR*[a] *Alternative 2:* CVE-Labelling

[a] SDN-basierte Incident-Response-Lösung, die in Kapitel 4 vorgestellt wird.

Eingesetzte Ontologien, Regeln, Arbeiter und Aufgaben

Ontologien/Wissensdomänen	Lab-Systemontologie, CVE-Ontologie
Aufgaben	19
SWRL-Regeln	4
Python-Arbeiter	3
Java-Arbeiter	6

Info-Box 3.3: Überblick des Evaluationsaufbaus zur Schwachstellenanalyse

Zur Identifikation bekannter Schwachstellen wurde in der **SKE-Phase** zunächst eine CVE-Ontologie integriert (vgl. Abbildung 3.54). Danach wurde in der **Analysis-Phase** ein Python-Arbeiter eingesetzt, welcher eine lokale Kopie der öffentlich verfügbaren National Vulnerability Database[2] (NVD)

[2] https://nvd.nist.gov/vuln/data-feeds, zuletzt zugegriffen: 06.12.2020.

der NIST im JSON-Format pflegt und die Software-Pakete des Modells aus der SKE-Phase mit den in dieser Datenbank enthaltenen CPEs vergleicht.

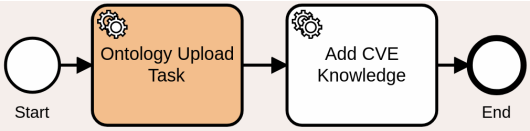


Abbildung 3.54: SKE-Workflow für das Integrieren von CVE-Wissen.

Der entsprechende Workflow ist in Abbildung 3.55 zu sehen. Da die CPEs den CVEs zugeordnet sind, leitet der Arbeiter somit ab, welche CVEs ein Paket betreffen und modelliert dies in der Ontologie. Ein Auszug der JSON-CVE-Repräsentation mit entsprechenden CPEs ist im angehängten Listing A.6 zu sehen. Der Arbeiter führt also ein analysespezifisches Labelling durch.

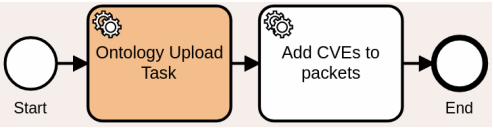


Abbildung 3.55: Analysis-Workflow für das Abgleichen von CPE-Informationen mit einer synchronisierten, lokalen CVE-Datenbank.

Das Ergebnis des Labellings wird dann lediglich noch von einer **Label-Abfrage** abgerufen (vgl. SPARQL-Abfrage aus Listing 3.6). Nach dem Ausführen der Evaluation der Konfigurationsanalyse (vgl. Abschnitt 3.7.2) mussten die ersten drei Phasen für Alternative 2 nicht erneut durchlaufen werden, da die Analysen auf dem selben bereinigten Systemmodell aufsetzen. Diese Synergie zwischen den Analysearten konnte somit trotz der im weiteren Verlauf unterschiedlichen Modellverarbeitung problemlos ausgenutzt werden. Mehr noch, die beiden Analysen lassen sich parallel ausführen. Die ersten drei Phasen werden dabei trotzdem nur einmalig durchlaufen.

```
SELECT ?asset ?package ?cve WHERE {
  ?asset a lab:Asset.
  ?package a Software.
  ?cve a cve:CVE.
  ?asset lab:softwareInventory ?swi.
  ?swi lab:software ?package.
  ?package cve:cve ?cve.
}
```

Listing 3.6: SPARQL-Abfrage, welche die verwundbaren Assets und Software-Pakete, sowie die entsprechenden CVEs zurückliefert.

Außerdem kann eine Analyse mit einem anderen Ziel, bspw. der Identifikation von Assets mit einer bestimmten Verwundbarkeit, auf dem vorberechneten, analysespezifischen Systemmodell durchgeführt werden, ohne eine Änderung am Modell vornehmen zu müssen. Hierfür ist lediglich eine andere SPARQL-Abfrage nötig. Kein Workflow muss hierfür erneut ausgeführt werden.

3.7.5 Bedrohungsanalyse

Auch eine Bedrohungsanalyse wurde mit dem SyMP-Rahmenwerk durchgeführt, welche Bedrohungen für die Assets des PoC-Systems identifizierte. Einen Überblick des Evaluationsaufbaus zeigt Info-Box 3.4.
Zunächst wurden im Rahmen der **KC-Phase** Informationen zu Hosts und Kommunikationsbeziehungen der PoC extrahiert. Diese Informationen wurden dabei initial von der Monitoring-Komponente Rhebo Industrial Protector aus dem mitgeschnittenen Netzwerkverkehr abgeleitet, wie in der ersten Aufgabe des KC-Workflows (vgl. Abbildung 3.56) zu sehen ist. Diese Aufgabe übernimmt ein Python-Arbeiter, der über die REST-Schnittstelle des Industrial Protectors besagte Informationen abfragt und damit die Rhebo-Systemontologie instanziiert. Die eingesetzten SWRL-Aufgaben klassifizieren daraufhin mittels ihrer Regeln interne Hosts durch deren IP-Adressen (vgl. Gleichung 3.12) und führen parallel Host-Individuen über ihre MAC-Adressen zusammen (vgl. Gleichung 3.13). Beides ist für die Vorverarbeitung angemessen, da das spätere Systemmodell sowohl zwischen den, im mitgeschnittenen Netzwerkverkehr

vorkommenden, lokalen und externen Hosts unterscheiden, als auch möglichst redundanzfrei sein soll.

Phasenbasierte Zusammenfassung der automatisierten Schritte

Phase	Inhalt
KC	Rhebo-REST-Export
KF/MC	PoC-Abbildung
SKE	ICS-ATT&CK®- und IEC-62443-Abbildung
DKE	-
A	-

Eingesetzte Ontologien, Regeln, Arbeiter und Aufgaben

Ontologien/Wissensdomänen	Rhebo-Systemontologie, PoC-Systemontologie, ATT&CK®-62443-Ontologie
Aufgaben	9
SWRL-Regeln	40
Python-Arbeiter	1
Java-Arbeiter	4

Info-Box 3.4: Überblick des Evaluationsaufbaus zur Bedrohungsanalyse

$$\begin{aligned} &\text{rhebo-system:Host}(?h) \wedge \textit{rhebo-system:containsInterface}(?h,?i) \\ &\wedge \text{rhebo-system:IpV4Interface}(?i) \wedge \textit{rhebo-system:longIp}(?i, ?lip) \\ &\wedge \textit{swrlb:lessThan}(?lip, 3232301055) \\ &\wedge \textit{swrlb:greaterThan}(?lip, 3232235520) \\ &\rightarrow \text{lab:InternalHost}(?h) \end{aligned} \tag{3.12}$$

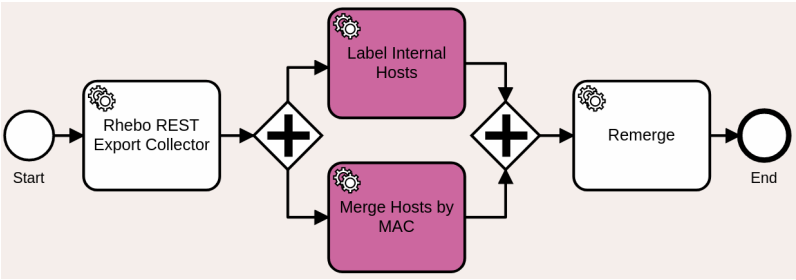


Abbildung 3.56: KC-Workflow zur Extraktion von Informationen aus dem PoC-Netzwerkverkehr.

$$\begin{aligned} &\textit{rhebo-system:containsInterface}(?h1,?i1) \wedge \text{rhebo-system:Host}(?h1) \\ &\wedge \text{rhebo-system:EthernetInterface}(?i1) \wedge \textit{lab:containsInterface}(?h2,?i2) \\ &\wedge \text{lab:Host}(?h2) \wedge \text{lab:EthernetInterface}(?i2) \\ &\wedge \textit{lab:mac}(?i1, ?mac1) \wedge \textit{lab:mac}(?i2, ?mac2) \\ &\wedge \textit{swrlb:equal}(?mac1, ?mac2) \rightarrow \textit{owl:sameAs}(?h1, ?h2) \end{aligned} \tag{3.13}$$

Im Workflow der **KF- und MC-Phasen** (vgl. Abbildung 3.57) wurde zudem statisches Systemwissen hinzugefügt, welches dazu führt, dass die wesentlichen Komponenten der PoC mittels MAC-Adresse auf manuell definierte Individuen abgebildet werden. Die Zuordnung besteht hierbei aus einer Reihe von SWRL-Regeln und ist dadurch intuitiv und einfach erweiterbar.

Abbildung 3.57: KF/MC-Workflow zur Weiterverarbeitung von Informationen aus dem PoC-Netzwerkverkehr.

Gleichung 3.14 zeigt diese Zuordnung für das Beispiel der Motorscheibe aus der PoC, welche über das Individuum `Motor-Scheibe` repräsentiert wird.

$$\begin{aligned} &\text{rhebo-system:InternalHost}(?h) \wedge \textit{rhebo-system:containsInterface}(?h,?i) \\ &\wedge \text{rhebo-system:EthernetInterface}(?i) \wedge \textit{rhebo-system:mac}(?i, ?mac) \\ &\wedge \textit{swrlb:equal}(?mac, \text{“}\texttt{54:e3:b0:00:d1:ec}\text{”}\hat{}\hat{}\texttt{xsd:string}) \\ &\rightarrow \textit{sameAs}(?h, \texttt{poc:Motor-Scheibe}) \end{aligned} \tag{3.14}$$

Diese nun bekannten Komponenten der PoC werden in der **SKE-Phase** (vgl. Abbildung 3.58) ebenfalls über SWRL mit Assets aus dem ICS-ATT&CK®-Rahmenwerk verknüpft, wofür zuvor die PoC-Systemontologie mit der ATT&CK®-62443-Ontologie zusammengeführt wird.

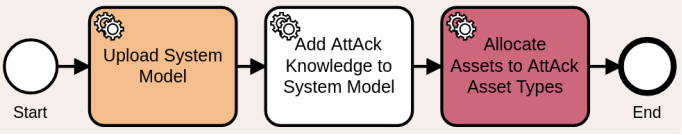


Abbildung 3.58: SKE-Workflow zur Weiterverarbeitung von Informationen aus dem PoC-Netzwerkverkehr.

Für die Bedrohungsanalyse reicht dies schon aus, um mithilfe der in Listing 3.7 zu sehenden analytischen SPARQL-Abfrage alle Bedrohungen (bzw. Techniken) zu erhalten, welche laut dem ICS-ATT&CK®-Rahmenwerk auf das System angewendet werden können. Praktischerweise enthält die ICS-ATT&CK®-Ontologie ebenfalls alle mit den Techniken verknüpften Gegenmaßnahmen, wodurch einem Analysten vorgeschlagen werden kann, welche Gegenmaßnahmen laut des ICS-ATT&CK®-Rahmenwerks für das System umzusetzen sind. Durch die in Abschnitt 3.7.3 eingeführte Abbildung von ICS-ATT&CK®-Mitigations zu IEC-62443-Anforderungen kann so auch automatisiert abgeleitet werden, welche der Anforderungen dadurch adressiert würden und durch die Umsetzung welcher Anforderungen bestimmte Angriffstechniken verhindert werden können.

```
SELECT ?threat ?assetType ?asset WHERE {
  ?threat a ics-attAck:Technique.
  ?threat ics-attAck:applicableTo ?assetType.
  ?asset ics-attAck:isA ?assetType.
}
```

Listing 3.7: SPARQL-Abfrage, welche die ICS-ATT&CK®-Techniken für die, im Systemmodell enthaltenen, Assets zurückliefert.

3.7.6 Angriffserkennung und -korrelation

In diesem Abschnitt wird die durchgeführte Evaluation zur Angriffserkennung und -korrelation beschrieben. Einen Überblick des Evaluationsaufbaus zeigt Info-Box 3.5. Diese Analyseart basiert immer auf Netzwerkmitschnitten eines Monitoring-Systems. Statt einer direkten Auswertung des Netzwerkverkehrs wurden in der hier vorgestellten Evaluation, wie bei gängigen Ansätzen üblich, vorverarbeitete Daten verwendet. Diese Daten kommen auch hier wieder vom Rhebo Industrial Protector, welcher neben den bereits in der Bedrohungsanalyse eingesetzten Informationen zu Hosts und Kommunikationsbeziehungen auch Zugriff auf Anomalie-Meldungen (vgl. Abschnitt 3.7.1) bietet. Die Anomalien werden dabei ermittelt, indem jedes Event als Anomalie klassifiziert wird, welches nicht schon einmal manuell über die Nutzerschnittstelle als *normal* gekennzeichnet wurde. Demnach ist diese Art der Anomalieerkennung ein einfaches Filtersystem für Netzwerkereignisse.
Die im Folgenden vorgestellte Evaluation der Bedrohungsanalyse, wurde im Rahmen des BMBF-Projektes AICAS durchgeführt. Dabei wurden die beiden erwähnten Quellen für folgende Ziele eingesetzt:

1. Verbessern der Anomalieerkennung durch Definition maschinen-interpretierbarer Regeln zum Ausdruck erlaubter Kommunikation, um die Anzahl der Anomalien automatisiert zu reduzieren und so die Wahrscheinlichkeit zu erhöhen, dass es sich bei den verbleibenden Meldungen um Hinweise zu Angriffen handelt.

2. Verbinden von Meldungen möglicher Angriffe zur Ableitung zusammengehöriger Angriffe, um die Wahrscheinlichkeit der Aufdeckung bösartiger Aktionen weiter zu erhöhen.

Damit adressiert die Untersuchung sowohl die Angriffserkennung als auch die Angriffskorrelation.

Phasenbasierte Zusammenfassung der automatisierten Schritte

Phase	Inhalt
KC	Rhebo-REST-Export
KF/MC	PoC-Abbildung
SKE	Deklaration erlaubter Verbindungen
DKE	Ableiten von Angriffsindikatoren
A	Labeln von mehrstufigen Angriffen

Eingesetzte Ontologien, Regeln, Arbeiter und Aufgaben

Ontologien/Wissensdomänen	Rhebo-Systemontologie, PoC-Systemontologie, ATT&CK®-62443-Ontologie
Aufgaben	9
SWRL-Regeln	40
Python-Arbeiter	1
Java-Arbeiter	4

Info-Box 3.5: Überblick des Evaluationsaufbaus zur Angriffserkennung und -korrelation

Für die Umsetzung wurde neben dem KC-Workflow aus Abschnitt 3.7.5 ein weiterer Workflow zur **KC-Phase** hinzugefügt (vgl. Abbildung 3.59), welcher die Aufgabe zum Hochladen einer instanziierten Rhebo-CEF-Ontologie definiert. Das Instanziieren der Rhebo-CEF-Ontologie und das dafür notwendige Parsen des Anomaliemeldung-Exports übernimmt indes ein dediziertes

Python-Programm, das Aufgrund seiner langen Laufzeit nicht über das Rahmenwerk gesteuert wurde.

Abbildung 3.59: KC-Workflow zum Hochladen der instanziierten Rhebo-CEF-Ontologie.

In Abbildung 3.60 ist zu erkennen, wie der Workflow der **KF-** und **MC-Phasen** im Vergleich zur Bedrohungsanalyse angepasst wurde. Wie dort zu sehen ist, musste lediglich eine weitere Aufgabe „Add CEF Knowledge“ hinzugefügt werden, welche das Integrieren der hochgeladenen instanziierten Rhebo-CEF-Ontologie verlangt. Auf dem aus diesem Workflow resultierenden Systemmodell kann auch die beschriebene Bedrohungsanalyse 3.7.5 unverändert ausgeführt werden, da es sich bei der beschriebenen Hinzunahme der Meldungsinformationen um eine Integration und nicht um ein Zusammenführen der Ontologien handelt.

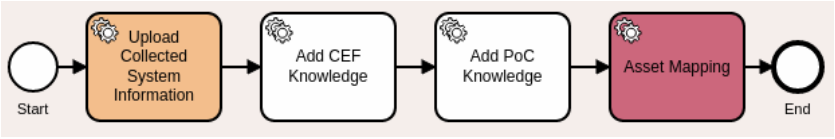


Abbildung 3.60: KF/MC-Workflow zur Erzeugung des PoC-Systemmodells aus CEF- und Inventory-Informationen.

Basierend auf dem so resultierenden Systemmodell wurden im Workflow der **SKE-Phase** (vgl. Abbildung 3.61) einfache SWRL-Regeln für die Deklaration erlaubter Verbindungen definiert.

Abbildung 3.61: SKE-Workflow zum Hinzufügen von ICS-ATT&CK®- und GPK-Ontologie, sowie zur Deklaration erlaubter Verbindungen.

Dafür wurde zunächst die GPK-Ontologie hinzugefügt, welche u.a. die abstrakte, nicht symmetrische Rolle *mayConnectTo* enthält. Eine Beispielregel basierend auf dieser Rolle ist in Gleichung 3.15 zu sehen.

$$
\begin{aligned}
&\text{poc:Asset}(\text{poc}:\texttt{SPS-1}) \\
&\rightarrow \mathit{gpk{:}mayConnectTo}(\text{poc}:\texttt{SPS-1}, \text{poc}:\texttt{SPS-2}) \\
&\wedge \mathit{gpk{:}mayConnectTo}(\text{poc}:\texttt{SPS-1}, \text{poc}:\texttt{SPS-3}) \\
&\wedge \mathit{gpk{:}mayConnectTo}(\text{poc}:\texttt{SPS-2}, \text{poc}:\texttt{SPS-1}) \\
&\wedge \mathit{gpk{:}mayConnectTo}(\text{poc}:\texttt{SPS-2}, \text{poc}:\texttt{SPS-3}) \\
&\wedge \mathit{gpk{:}mayConnectTo}(\text{poc}:\texttt{SPS-3}, \text{poc}:\texttt{SPS-1}) \\
&\wedge \mathit{gpk{:}mayConnectTo}(\text{poc}:\texttt{SPS-3}, \text{poc}:\texttt{SPS-2})
\end{aligned}
\tag{3.15}
$$

Die Regel beschreibt, dass die SPS-1, SPS-2 und SPS-3 untereinander verbunden sein dürfen. Da nur die Implikation der Regel benötigt wird, ist die Bedingung der Regel so gewählt, dass diese immer wahr ist (SPS-1 ist immer ein poc:Asset). Die *gpk:mayConnectTo*-Beziehungen für erlaubte, gerichtete Verbindungen werden später verwendet, um die False-Positives[1] zu reduzieren. Außerdem wurde im Rahmen der Aufgabe, welche die GPK-Ontologie-Integration initiiert, auch das Hinzufügen der ICS-ATT&CK®- und der IntrusionDetection-Ontologie verlangt.

[1] False-Positives sind in diesem Kontext „fälschlicherweise als Anomalien gekennzeichnete Ereignisse“.

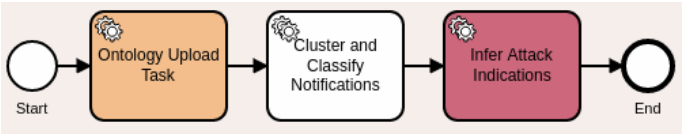


Abbildung 3.62: DKE-Workflow zum Ableiten von Angriffsindikatoren.

In der **DKE-Phase** erfolgte eine Markierung von Angriffen, welche als Grundlage für die Angriffskorrelation dient, aber auch einzeln abgefragt werden kann (vgl. DKE-Workflow in Abbildung 3.62). Konkret wurden hier Hosts markiert, welche unter Verdacht stehen andere Hosts zu scannen. Da der Industrial Protector bei den in den Angriffen vorkommenden SYN-Scans keine verlässlichen Informationen lieferte, wurden mithilfe eines Python-Arbeiters Individuen der Klasse id:PortScanNotificationCluster aus der IntrusionDetection-Ontologie erzeugt, für die Folgendes zutrifft:

- Ein id:PortScanNotificationCluster-Individuum verweist mittels der abstrakten *id:containsNotification*-Rolle auf alle *INCOMPLETE_TCP*-Meldungen[1], die sich Quell- und Zielhost teilen.
- Innerhalb eines Zeitfensters von 10 Sekunden wurden mehr als 20 *INCOMPLETE_TCP*-Meldungen mit dem Quell- und dem Zielhost erzeugt.
- Die Abstände zwischen den Meldungen dürfen innerhalb eines Clusters die Zeit von 20 Sekunden nicht überschreiten.

Diese Ableitung ist ein Beispiel, bei dem SWRL nicht verwendet werden kann, da die Ableitung auf Basis der Anzahl von bestimmten Individuen zu einem bestimmten Moment ist und bei Veränderung dieser Anzahl ungültig werden kann. Dies würde der Monotonie von SWRL widersprechen.
Die Zeitfenster-abhängige Heuristik ist nötig, um weitestgehend auszuschließen, dass es sich um reguläre Verbindungsversuche handelt und bietet so eine gewisse Sicherheit, dass die Meldungen auf einen Scan-Vorgang hinweisen.

[1] Die Meldung wird vom Industrial Protector ausgegeben, wenn eine TCP-Verbindung nicht vollständig aufgebaut wurde oder niemals Nutzdaten übertragen hat.

Auch wenn dabei die Absicht des Scans nicht identifiziert werden kann, reicht dies doch, um ein Indiz für Ereignisse im Rahmen der *Discovery*-Bedrohung[1] nach dem ICS-ATT&CK®-Rahmenwerk darzustellen. Hergeleitet wird diese Klassifizierung als Discovery-Bedrohung wieder von einer SWRL-Regel (vgl. Gleichung 3.16), welche den erwähnten Cluster-Individuen die Technik *Network Service Scanning*[2] (mit dem Identifikator T0841) über die abstrakte Rolle *id:indicates* zuweist.

$$\text{id:PortScanNotificationCluster}(?c) \rightarrow \textit{id:indicates}(?c, \texttt{ics-attack:T0841}) \tag{3.16}$$

Analog wurde eine SWRL-Regel hinzugefügt, welche die *Persistence*-Bedrohung[3] durch eine *id:indicates*-Verknüpfung zwischen S7-Download-Meldungen (mehr dazu im nächsten Paragraphen) und der Technik *Program Download*[4] (mit dem Identifikator T0843) klassifiziert (vgl. Gleichung 3.17).

$$\begin{aligned} &\text{rhebo-cef:Notification}(?n) \wedge \textit{rhebo-cef:hasNotificationData}(?n, ?d) \\ &\wedge \text{rhebo-cef:Cs1}(?d) \wedge \textit{rhebo-cef:value}(?d, ?v) \\ &\wedge \textit{swrlb:equal}(?v, \textit{“S7\_DOWNLOAD”}\texttt{\^\^xsd:string}) \\ &\rightarrow \textit{id:indicates}(?n, \texttt{ics-attack:T0843}) \end{aligned} \tag{3.17}$$

Basierend auf dem so erzeugten Modell konnten verschiedene **analytische Abfragen** realisiert werden. Die simpelste Abfrage besteht aus dem Abfragen von Meldungen und zugehörigen Techniken, die auf eine Discovery-Bedrohung hinweisen (vgl. Listing 3.8).
Weitere Abfragen identifizieren typische Angriffsvektoren im Rahmen der Angriffskorrelation.

[1] https://collaborate.mitre.org/attackics/index.php/Discovery, zuletzt zugegriffen: 29.01.2021.
[2] https://collaborate.mitre.org/attackics/index.php/Technique/T0841, zuletzt zugegriffen: 29.01.2021.
[3] https://collaborate.mitre.org/attackics/index.php/Persistence, zuletzt zugegriffen: 29.01.2021.
[4] https://collaborate.mitre.org/attackics/index.php/Technique/T0843, zuletzt zugegriffen: 29.01.2021.

```
SELECT ?notification ?technique WHERE {
  ?notification a rhebo-cef:Notification.
  ?cluster a id:PortScanNotificationCluster.
  ?cluster id:indicates ?technique.
  ics-attack:Discovery ics-attack:associatedWith ?technique.
  ?cluster id:containsNotification ?notification.
}
```

Listing 3.8: SPARQL-Abfrage, welche die Meldungen zurückliefert, die auf den Einsatz der ICS-ATT&CK®-Taktik Discovery beziehen.

Dabei wurde beispielsweise abgefragt, ob ein mehrstufiger Angriff aufgezeichnet wurde, der zuerst die Discovery-, dann die Persistence-Taktik enthält (vgl. Listing 3.9). Dies ist beispielsweise dann der Fall, wenn nach einem Scan auf eine Komponente auch versucht wurde, diese mit Schadcode zu belegen. Bei SPSen, welche in der PoC zum Herunterladen von Funktionsblöcken das *S7-Protokoll*[1] verwenden, weist darauf eine vom Rhebo Industrial Protector generierte *S7-Download*-Meldung hin.
Ein nicht-leeres Ergebnis dieser Abfrage bedeutet zum einen, dass dieser Angriffsvektor mit hoher Wahrscheinlichkeit stattgefunden hat, zum anderen deutet es darauf hin, dass die im Ergebnis enthaltenen Meldungen wirklich bösartige Ereignisse identifizieren.
Auch wenn es sich dabei nur um eine Wahrscheinlichkeit handelt, wurden durch die Abfrage weitere gutartige Szenarien ausgeschlossen. Ein Beispiel dafür ist das Prüfen der korrekten Port-Konfiguration durch einen Systemadministrator, ohne dass dieser danach eine Projektierung durchführt. Wie eingangs erwähnt ist eine weitere Version dieser Abfragen sinnvoll. Dabei wird jeder Abfrage der Filter aus Listing 3.10 nachgestellt. Dieser Filter ermöglicht die Reduktion der Meldungen auf jene, die nicht explizit erlaubte Verbindungen betreffen und reduziert somit die zuvor erwähnten False-Positives.

[1] https://www.ipcomm.de/protocol/S7ISOTCP/de/sheet.html, zuletzt zugegriffen: 17.05.2021.

```
SELECT ?cluster ?technique1 ?notification ?technique2 WHERE {
  {
    SELECT ?cluster ?technique1 (max(?time) as ?
        maxTimeFromCluster) WHERE {
      ?cluster id:indicates ?technique1.
      ?technique1 ics-attack:associatedWith ics-attack:
          Discovery.
      ?cNotification a rhebo-cef:Notification.
      ?cluster id:containsNotification ?Notification.
      ?cNotification rhebo-cef:hasNotificationData ?rt.
      ?rt a rhebo-cef:Rt.
      ?rt rhebo-cef:value ?time.
    } group by ?cluster ?technique1
  }
  ?notification a rhebo-cef:Notification.
  ?notification id:indicates ?technique2.
  ?technique2 ics-attack:associatedWith ics-attack:
      Persistence.
  FILTER (?maxTimeFromCluster < ?timeFromNotification)
}
```

Listing 3.9: SPARQL-Abfrage, welche die Meldungen zurückliefert, die potenziell zu dem Angriffsvektor Discovery-Persistence gehören.

```
filter not exists{
    ?notification poc:hasSrcHost ?src.
    ?notification poc:hasDstHost ?dst.
    ?src gpk:mayConnectTo ?dst.
}
```

Listing 3.10: SPARQL-Filterausdruck, der zur weiteren Reduzierung von False-Positives genutzt werden kann.

Für die Evaluation wurden im Rahmen einer Masterarbeit *[Kah21]* 247 Angriffsschritte in der PoC durchgeführt und mithilfe des Industrial Protectors aufgezeichnet. Dieser hatte zuvor bereits mehrere Monate eine Baseline gelernt. D.h. er hat durch Nutzerfeedback ein Modell des regulären Netzwerkverkehrs erstellt, um Abweichungen als Anomalien identifizieren zu können. In dem Zeitraum der Ausführung der Angriffe (ca. 48 Stunden) wurden vom Industrial Protector 90.000 Anomaliemeldungen erzeugt.
Durch die einfachen *gpk:mayConnectTo*-Regeln konnten diese Meldungen um 139 False-Positives auf 89.861 reduziert werden. Da die Regeln recht sparsam eingesetzt wurden und der Industrial Protector eigentlich bereits gelernt hatte, welches Verhalten normal ist, war diese Reduktion höher als vor Durchführung der Evaluation erwartet. Weitere Regeln könnten voraussichtlich noch weitere False-Positives herausfiltern, da sich unter den Meldungen auch eine nicht genau bestimmte Anzahl erlaubter Verbindungen befindet, die jedoch durch beispielsweise Verbindungsabbrüche oder unvollständige TCP-Handshakes als verdächtig eingestuft wurden. Dabei ist allerdings Vorsicht geboten. Selbst die *gpk:mayConnectTo*-Regeln sind eigentlich schon zu generisch. Denn durch das pauschale Ausfiltern (vgl. Listing 3.10) und die Regeldefinitionen aus Gleichung 3.15 wird verhindert, dass möglicher bösartiger Verkehr zwischen den Steuerungen beachtet wird. Regeln und regelabhängige Bedingungen sollten also mit Bedacht gewählt und so feingranular wie möglich eingesetzt werden.
Eine manuelle Identifikation von Techniken, Taktiken und Angriffskorrelationen hätte aufgrund der verbleibenden Menge von 89.861 Meldungen kaum Aussicht auf Erfolg. Durch die eben beschriebenen Workflows war es möglich, automatisiert ein Modell zu erzeugen, welches bereits die gesuchten Informationen enthält, die der Analyst nur noch mit einer semantischen Abfragesprache extrahieren musste. Dabei wurden 8.114 Meldungen als Hinweise auf Port-Scans klassifiziert und in insgesamt 634 Scan-Cluster zusammengefasst. Durch Adjustieren der zuvor gelisteten Entscheidungsparameter zur Cluster-Zuweisung, kann die Anzahl der Cluster entsprechend reduziert oder erhöht werden. Je nach Einstellung werden somit mehr oder weniger False-Positives bezüglich der Scan-Erkennung generiert.
Gleichzeitig wurden 12 Meldungen als T0843 (Program Download) klassifiziert. Hier hat sich das Filtern der Meldungen ebenfalls als nützlich herausgestellt.

Da der entsprechende S7-basierte Download (vgl. Gleichung 3.17) in der Praxis meist nur von wenigen Geräten durchgeführt wird (oft nur eine einstellige Anzahl von Projektierungs-Workstations), können genau dafür auch unkomplizierte Erlauben-Regeln festgelegt werden. Im Falle der PoC führte dies dazu, dass die 12 klassifizierten Meldungen alle tatsächliche Angriffe waren.
Die Angriffserkennung konnte also gegenüber der reinen Anomalieerkennung des Industrial Protectors deutlich vereinfacht und somit verbessert werden.
Wie bereits zuvor erwähnt hat sich auch die semantische Angriffskorrelation als nützlich erwiesen. Durch die Ableitung des ICS-ATT&CK®-Wissens und die Zeitstempel der Meldungen konnten zusammenhängende Angriffsschritte mit semantischen Abfragen festgestellt werden. Vor jedem der 12 Programm-Download-Meldungen erfolgte in den Angriffsszenarien ein Scan. Dieser Zusammenhang konnte durch die semantische Abfrage aus Listing 3.9 extrahiert werden. Auf dem ICS-ATT&CK®-Rahmenwerk basierende Angriffsvektoren lassen sich so folglich mit den weitläufig bekannten und eingesetzten Semantic-Web-Technologien abfragen und wie in Abschnitt 3.6 beschrieben automatisiert erkennen.

3.7.7 Erfüllung der Anforderungen

In den vorangegangenen Abschnitten wurde beschrieben, wie im Rahmen dieser Evaluation jede in dieser Dissertation behandelte Analyseart mit der Implementierung des SyMP-Rahmenwerks getestet wurde. Dadurch und durch die Untersuchungen in Abschnitt 3.4.1 ist die Anforderungsabdeckung der entwickelten Konzepte und Methoden (bzgl. der Anforderungen in Abschnitt 3.2) diskutierbar. Diese Diskussion wird in den folgenden Paragraphen geführt.
Als Überblick zeigen Tabellen 3.12, 3.13, 3.14, 3.15 und 3.16 hierzu die Anforderungsabdeckungseinschätzungen aus Abschnitt 3.3.11, welche um die Abdeckung der Anforderungen durch SyMP erweitert wurden. Entsprechend zugeordnete Nachweise werden im Nachgang der jeweiligen Tabelle präsentiert. Ein handlicher Überblick über die Anforderungskürzel und deren zugehörige Titel wird zudem in Tabelle 3.11 bereitgestellt. Auch sei noch einmal erwähnt, dass ● die Erfüllung einer Anforderung, ○ die Nichterfüllung einer Anforderung und ◐ die partielle, nicht hinreichende Erfüllung einer Anforderung ausdrückt.

Tabelle 3.11: Anforderungskürzel und deren Bedeutungen.

Kürzel	Anforderung
UA1	Minimalinvasiv
UA2	Ressourcenschonend
UA3	Quellheterogenität
UA4	Abdeckung industrieller Assets
IA1	Sichere Informationsextraktion
IA2	Reduktion der Abbildungskomplexität
IA3	Unterstützung von Standards und konsolidierten Spezifikationen zu Informationsmodellen
IA4	Unterstützung verschiedener Serialisierungen
IA5	Unterstützung von Zielmodellen
IA6	Einfache Anpassung der Quellen
MA1	Einsatz von Ontologien zur Modellierung
MA2	Einsatz monotoner Beschreibungslogiken zur Modellierung und Modellerweiterung
MA3	Unterstützung von nicht-DL Formalismen und Algorithmen
MA4	Klare Trennung zwischen der Verwendung von Beschreibungslogiken und anderer Formalismen
MA5	Separation-of-Concerns
MA6	Austauschbare Erweiterungslogik
MA7	Analysespezifische Erweiterung
MA8	Abhängige Verarbeitungsschritte
MA9	Unterstützung von Nutzerinteraktion bei Modellbildung und -verarbeitung
AA1	Unterstützung der behandelten Analysearten
AA2	Technische Evidenz
AA3	Flexible Richtlinien und Regeln
AA4	Austauschbare/Wiederverwendbare Analysen
AA5	Anpassungsfähigkeit
AA6	Sicherheitsdomänen
GA1	Automatisierung
GA2	Skalierbarkeit

weiter auf der nächsten Seite

Kürzel	Anforderung
GA3	Nutzbarkeit
GA4	Offenheit
GA5	Lebenszyklusunterstützung

Tabelle 3.12: Abdeckung umgebungsbezogener Anforderungen durch den Stand von Wissenschaft und Technik und SyMP.

	UA1	UA2	UA3	UA4
CySeMoL	●	◐	◐	◐
SecuriCAD	●	○	●	◐
Fenz et al.	◐	○	○	○
MulVAL	●	○	●	○
Lightbulb	●	○	●	○
C2NET	●	◐	●	◐
SyMP	●	●	●	◐

UA1: SyMP unterstützt die passive Informationsextraktion in der KC-Phase. Gezeigt wurde dies in Abschnitten 3.7.2-3.7.6.
UA2: Die Anforderung wird durch die KC-Phase von SyMP erfüllt (vgl. Abschnitte 3.7.2-3.7.6) und wurde anhand der Quellen OPC UA, AutomationML-Export und Rhebo Industrial Protector evaluiert (vgl. entspr. Abschnitte 3.4.1.1, 3.7.4, 3.7.5 und 3.7.6).
UA3: Dies wird durch die Unterstützung von verschiedenen Quellen und Abbildungsstrategien in der KC-Phase von SyMP garantiert. Evaluiert wurde dies in den Konfigurationen der KC-Phase von Abschnitten 3.7.2-3.7.6 und mithilfe der erweiterbaren Python-Bibliothek X2Owl (vgl. Abschnitt 3.4.1.2).

UA4: Durch die folgenden Beiträge wurde die Abdeckung industrieller Assets über den Stand von Wissenschaft und Technik hinaus ermöglicht:

- Unterstützung beliebiger Quellen und Abbildungsstrategien in der KC-Phase von SyMP
- AutomationML-basierte Methode zur Modellierung und Extraktion von Netzwerkinformationen aus Engineering-Werkzeugen
- Untersuchung von Abbildungsmethoden zur Generierung Digitaler Zwillinge, inkl. X2Owl-Python-Bibliothek

UA4 wird auch von SyMP nicht vollständig erfüllt. Zwar wurden mit den Fortschritten in der Modellierung mit AutomationML, X2Owl und der Untersuchung von Verwaltungsschalen-Abbildungsstrategien wichtige Schritte in Richtung einer hohen Abdeckung gemacht, die tatsächliche Abdeckung hängt aber unter anderem vom Erfolg von Konzepten wie der Verwaltungsschale und AutomationML in der Praxis ab. Dies verdeutlicht, dass die tatsächliche Abdeckung der Informationsextraktion von Komponenten- und Softwareherstellern abhängt. Die Wissenschaft kann hier nur Methoden und Konzepte bereitstellen, welche bspw. zur Erhöhung der Schnittstellenhomogenität oder Vereinheitlichung von Syntax und Semantik führen, sodass in der Praxis eine höhere Abdeckung erreicht werden kann.

Tabelle 3.13: Abdeckung von Anforderungen an Informationsextraktion und -repräsentation durch den Stand von Wissenschaft und Technik und SyMP.

	IA1	IA2	IA3	IA4	IA5	IA6
CySeMoL	◐	●	●	◐	●	●
SecuriCAD	●	○	○	○	●	●
Fenz et al.	◐	○	○	○	●	○
MulVAL	◐	○	○	○	●	●
Lightbulb	◐	○	○	○	●	●
C2NET	◐	◐	◐	◐	○	●
SyMP	●	●	●	●	●	●

IA1: Wird in der Implementierung aktuell durch den Einsatz von OPC UA mit dem SignAndEncrypt-Profil, sowie durch die Verwendung von TLS für die REST-Schnittstellen von Geräten zu den *Knowledge-Collection*-Arbeitern der KC-Phase und zwischen SyMP-Komponenten erreicht. Dies ist allerdings implementierungsabhängig und nicht Teil des Rahmenwerks.
IA2: Systemontologie-TBox kann bei Extraktion verwendet werden (vgl. CLI File Export mittels X2Owl unter Verwendung der Lab-Systemontologie). Beliebige Abbildungen sind in KC-Phase möglich und werden durch Abbildungs- und Fusionsstrategien unterstützt (vgl. Abschnitte 3.4.1.3.2 und 3.4.2). Nachgewiesen u.a. durch Verknüpfung der Rhebo- mit der PoC-Systemontologie.
IA3: Beliebige Abbildungen werden vom Workflow-Generator unterstützt und somit automatisiert in das Modell integriert. Die Analysen können dabei entsprechend von beliebigen Domänenontologien abhängen. Nachgewiesen in der jeweiligen SKE-Phase der zuvor beschriebenen Workflows.
IA4: SyMP lässt beliebige Arbeiter zu und kann somit für beliebige Serialisierungen erweitert werden.
IA5: Durch die automatisierte Abhängigkeitsauflösung und Abbildungsstrategien von SyMP werden beliebige Systemmodelle unterstützt und die Konsistenz und Funktionalität von Analysen und Modellerweiterungen wird gewahrt.
IA6: Durch die Automatisierung mittels Workflows können Modellaktualisierungen vollautomatisch durchgeführt werden. Ein besonderer Vorteil ist dabei die Phaseneinteilung von SyMP, die dafür sorgt, dass eine Änderung nicht die Ausführung aller Workflows zur Folge hat. Bspw. führt eine Änderung einer Komponente nicht zum erneuten Ausführen der KC-Workflows anderer Quellen. Auch führen Änderungen in einer SKE-Phase nicht zu Änderungen des bereinigten Systemmodells oder zur erneuten Ausführung anderer SKE-Workflows oder den davon abhängigen späteren Workflows.

Tabelle 3.14: Abdeckung von Anforderungen an die Modellbildung und -verarbeitung durch den Stand von Wissenschaft und Technik und SyMP.

	MA1	MA2	MA3	MA4	MA5	MA6	MA7	MA8	MA9
CySeMoL	○	○	●	○	●	○	○	○	●
SecuriCAD	○	○	●	○	○	○	○	○	●
Fenz et al.	●	●	○	○	○	◐	○	○	○
MulVAL	○	●	○	○	○	◐	○	○	○
Lightbulb	○	●	○	○	○	◐	○	○	○
C2NET	●	●	◐	○	●	◐	◐	○	◐
SyMP	●	●	●	●	●	●	●	●	●

MA1: Die Modellverwaltung in SyMP ist Ontologie-basiert.
MA2: Die zuvor beschriebenen Workflows setzen SWRL ein. SyMP ermöglicht den Einsatz beliebiger Arbeiter und Reasoner.
MA3: Die zuvor beschriebenen Workflows setzen Python- und Java-Arbeiter ein. SyMP ermöglicht den Einsatz beliebiger Arbeiter und Reasoner.
MA4: In SyMP werden Regeltypen, wie SWRL-Regeln und Jena-Regeln, voneinander und von sonstigen Implementierungen mit Hilfe separater Klassifikationen getrennt, die bereits vor dem Erzeugen der Workflows maschineninterpretierbar zur Verfügung stehen (vgl. Abschnitt 3.4.3.2).
MA5: In SyMP werden die Akteure durch voneinander getrennte Werkzeuge, Schnittstellen und Verarbeitungsbereiche (vgl. SyMP-Phasen) unterstützt. Eine Umsetzung davon wurde in Abschnitt 3.6 beschrieben und für die Evaluation erfolgreich eingesetzt.
MA6: In SyMP sind Modellerweiterungen der ersten drei Phasen beliebig editierbar. Bei automatisierter Generierung der Workflows, können die Modellverarbeitungsschritte durch neue Analyseregelumsetzungen angepasst werden und sind somit sogar von weiteren Nutzern einsetzbar.
MA7: Durch die Abhängigkeiten und deren evaluierte Auflösung bei der Workflow-Generierung werden Modellerweiterungen der letzten drei Phasen

genau für die Analysen eingesetzt, welche auf dem Modell ausgeführt werden sollen.
MA8: Die Abhängigkeitsverwaltung und -auflösung mittels Workflow-Generator wurde im Rahmen der automatischen Erstellung von Workflows für diese Evaluation und in *[Klo21]* ausgiebig getestet.
MA9: Die Anbindung von externen Werkzeugen wurde durch den Einsatz von Camunda umgesetzt und von Werkzeugseite in *[Roe21]* getestet und evaluiert.

Tabelle 3.15: Abdeckung von analysebezogenen Anforderungen durch den Stand von Wissenschaft und Technik und SyMP.

	AA1	AA2	AA3	AA4	AA5	AA6
CySeMoL	○	◐	◐	◐	○	○
SecuriCAD	○	○	○	○	○	○
Fenz et al.	○	◐	●	●	○	○
MulVAL	○	●	●	●	◐	○
Lightbulb	○	●	●	●	◐	○
C2NET	◐	◐	●	◐	◐	◐
SyMP	●	●	●	●	●	●

AA1: Wie in Abschnitten 3.7.2-3.7.6 gezeigt, werden die hier behandelten Analysearten von SyMP unterstützt.
AA2: Wie in Abschnitten 3.7.2 und 3.7.3 zu sehen ist, wird der Bezug eines Fundes zur tatsächlichen technischen Konfiguration bewahrt.
AA3: Besonders Abschnitt 3.7.3 demonstriert, dass SyMP flexible, austauschbare Richtlinien unterstützt.
AA4: Die Austauschbarkeit und Wiederverwendbarkeit von Analysen wurde mithilfe des SyMP-Hubs und seiner Wissensbasis, des Analyse-Clients und des Zusammenspiels dieser Komponenten mit der Analyse-Engine demonstriert.
AA5: Die gezeigte Austauschbarkeit und Wiederverwendung von Analysen und Modellen erfüllt diese Anforderung.

AA6: Die Austauschbarkeit von Sicherheitsdomänen wurde in den vorgestellten Analysen gezeigt und durch die ICS-ATT&CK®- und IEC-62443-Ontologien repräsentiert.

Tabelle 3.16: Abdeckung von generellen Anforderungen durch den Stand von Wissenschaft und Technik und SyMP.

	GA1	GA2	GA3	GA4	GA5
CySeMoL	◐	◐	◐	◐	○
SecuriCAD	◐	●	◐	◐	○
Fenz et al.	◐	◐	◐	○	○
MulVAL	●	◐	◐	○	○
Lightbulb	●	◐	◐	○	○
C2NET	◐	◐	◐	○	●
SyMP	●	●	●	◐	●

GA1: Die zuvor beschriebenen Analysen konnten nach initialer Konfiguration vollautomatisch durchgeführt werden.
GA2: Abschnitte 3.7.2 und 3.7.5 zeigen parallelisierte Aufgaben. Diese werden von externen Arbeitern verteilt bearbeitet. Zudem unterstützt der Workflow-Generator das Zusammenführen von Regel-Aufgaben zu einer einzelnen Aufgabe mit einer Regelliste als Eingabe (vgl. Abschnitt 3.6).
GA3: SyMP basiert auf Separation-of-Concerns. Dies wurde in Abschnitt 3.5 erläutert und mithilfe der Implementierung (vgl. Abschnitt 3.6) umgesetzt. Weiter konnten das SyMP-Hub *[Klo21]* und der SyMP-Client, in Verbindung mit der Sicherheitsanalyse-Engine, als Akteur-Schnittstellen erfolgreich für die mehrschichtige Analysedefinition, -konfiguration und -ausführung eingesetzt werden.
GA4: Die Implementierung von SyMP basiert auf Open-Source-Software. Wesentliche Ontologien, Bibliotheken und Arbeiter-Programme wurden bereits

veröffentlicht (vgl. Abschnitte 3.4.1.2 und 3.7.1). Weitere Komponenten wurden noch nicht veröffentlicht. Dies ist jedoch zum Zeitpunkt des Verfassens dieser Dissertation bereits in Arbeit und ist bisher lediglich aufgrund der Qualitätssicherung noch nicht geschehen.
GA5: Der Einsatz von SyMP-Implementierungen ist durch die flexible Konfiguration unabhängig von der Phase des Systems. Zu erkennen ist dies auch an den durchgeführten Analysen, die bereits alle wesentlichen Analysen eines Sicherheitslebenszyklus abdecken.

Zusammenfassend ist festzuhalten, dass die gelisteten Anforderungen von SyMP fast vollständig abgedeckt werden konnten. Lediglich die Abdeckung industrieller Assets und die freie Verfügbarkeit der SyMP-Software weisen hier Defizite auf. Dabei deckt SyMP zwar die Erweiterbarkeit bezüglich Informationsextraktionsmethoden ab und diese Dissertation stellt zudem bereits weitere Methoden zur Abdeckung bereit, eine vollständige Abdeckung industrieller Assets bleibt jedoch ein wirtschaftspolitisches Problem und ist nicht allein durch Technologien zu lösen.

3.7.8 Zielerreichung

Im Folgenden werden die zur Evaluation der Zielerreichung aus Abschnitt 1.2 verwendeten Fragestellungen gelistet. Tabelle 3.17 gibt die zugehörigen Antworten und entsprechenden Begründungen. Zuletzt wird ein Resümee gezogen.
Fragestellungen zur Erreichung der Zielsetzung aus Abschnitt 1.2:

1. Können mit den erarbeiteten Lösungen mehr Schwachstellen, Fehlkonfigurationen, Inkonsistenzen, Bedrohungen, Konformitätsprobleme und Angriffe identifiziert werden, als durch den Stand der Technik (vgl. Abschnitt 3.3)?
    a) Wurde die Abdeckung der Geräte im Rahmen der Informationsextraktion erhöht?
        i. Werden bisherige Ansätze zur Informationsextraktion unterstützt?
        ii. Werden Gerätetypen unterstützt, die bisher nicht durch die Informationsextraktion abgedeckt waren?

   b) Zum Austausch von Netzwerkstrategien:
      i. Erhöht die Anpassungsmöglichkeit von Netzwerkstrategien die Abdeckung nach Zielsetzung 1?
      ii. Können Netzwerkstrategien in der Lösung ausgetauscht werden?
   c) Können Analysemodellerweiterungen ausgetauscht und wiederverwendet werden?
      i. Erhöht die Austauschbarkeit und Wiederverwendbarkeit der Modellerweiterungen die Abdeckung nach Zielsetzung 1?
      ii. Erhöht die Austauschbarkeit und Wiederverwendbarkeit der Modellerweiterungen die Effizienz von Analyselösungen?
   d) Können Analysen ausgetauscht und wiederverwendet werden?
      i. Erhöht die Austauschbarkeit und Wiederverwendbarkeit der Analysen die Abdeckung nach Zielsetzung 1?
      ii. Erhöht die Austauschbarkeit und Wiederverwendbarkeit der Analysen die Effizienz von Analyselösungen?
2. Wurde die Effizienz von Analyselösungen gesteigert?
   a) Wurde der Automatisierungsgrad des Stands der Technik für die Modellbildung erreicht?
   b) Wurde der Automatisierungsgrad des Stands der Technik für die Modellbildung übertroffen?
   c) Wurde der Automatisierungsgrad des Stands der Technik für die Modellerweiterung erreicht?
   d) Wurde der Automatisierungsgrad des Stands der Technik für die Modellerweiterung übertroffen?
3. Zur Konzeptionierung des Rahmenwerks:

a) Wurde ein wissensbasiertes Rahmenwerk mit hohem Automatisierungsgrad entwickelt?

b) Ist das Rahmenwerk flexibel?

c) Werden automatisierte Angriffserkennung und -korrelation, sowie automatisierte Schwachstellen-, Konfigurations, Konformitäts- und Bedrohungsanalysen unterstützt?

d) Ist der Einsatz der Analysen im Rahmen des Rahmenwerks für industrielle Systeme geeignet?

Tabelle 3.17: Antworten auf die Fragen zur Zielerreichung.

Frage	Antwort
1.a)	Ja, vgl. Antworten zu 1.a)i. und 1.a)ii.
1.a)i.	Ja, da die KC-Phase beliebige Schnittstellen und Quellsemantiken unterstützt.
1.a)ii.	Ja, da mit der entwickelten Methode für AutomationML auch Engineering-Werkzeuge zur Informationsextraktion genutzt werden können (vgl. Abschnitt 3.4.1.1).
1.b)i.	Ja, dies zeigt das einfache Austauschbeispiel von NAT vs. eindeutige IP-Adressen das mit Erfüllung von Anforderung MA6 ermöglicht wird (vgl. Abschnitt 3.7.7).
1.b)ii.	Ja, Netzwerkstrategien entsprechen Aufgaben von Workflows und können somit ausgetauscht werden.
1.c)	Ja, diese sind, wie auch die Netzwerkstrategien, Aufgaben von Workflows und können somit beliebig definiert und ausgetauscht werden. Ggf. sind jedoch auch die entsprechenden Arbeiter-Implementierungen notwendig.

weiter auf der nächsten Seite

Frage	Antwort
1.c)i.	Ja, da mehrere parallele Versionen eines Modells möglich sind, können auch sich widersprechende Erweiterungen genutzt werden. Außerdem sind beliebig viele Erweiterungen, mit beliebigen Domänen und beliebigen Umsetzungen (Arbeiter-Implementierungen) möglich. Diese Flexibilität wird von keiner bisherigen Lösung unterstützt. Die Wiederverwendbarkeit wird zudem durch den Hub-Ansatz aktiv unterstützt.
1.c)ii.	Ja, da Aufgaben von beliebigen, insbesondere optimierten, Arbeiter-Implementierungen ausgeführt und selbst durch effizientere Aufgabendefinitionen mit der selben Ausgabe ersetzt werden können. Beispielsweise kann eine SWRL-Aufgabe durch eine performantere Individualprogramm-Aufgabe ersetzt werden.
1.d)	Ja, da die Definition der Analyseregeln unabhängig von der Auswahl und Konfiguration der eingesetzten Analyseregeln ist.
1.d)i.	Ja, da die bisher übliche Abdeckung von einzelnen Analysearten auf die hier beschriebenen gleichzeitige Abdeckung verschiedener Analysearten erweitert wurde und der Hub-Ansatz mehrere Analyseregel-Implementierungen für eine Richtlinie zulässt.
1.d)ii.	Mit der in Abschnitten 3.7.2 und 3.7.3 nachgewiesenen Ausnutzung von Synergien, wird die Effizienz bei Einsatz verschiedener Analyseregeln erhöht. So wird durch SyMP konsequent Mehrfachausführung von Modellverarbeitungsschritten und Ausführung ungenutzter Schritte vermieden.
2.	Ja, da die Lösung auch Eigenschaften unterstützt, die bei keiner bisherigen Automatisierungslösung unterstützt werden (siehe Abschnitt 3.7.7).
2.a)	Ja, da die Vollautomatisierung nach initialer Konfiguration möglich ist.
2.b)	Ja, da die Vollautomatisierung nach initialer Konfiguration auch für flexible Quellenschnittstellen, Semantiken und Abbildungsstrategien möglich ist.

weiter auf der nächsten Seite

Frage	Antwort
2.c)	Ja, da die Vollautomatisierung nach initialer Konfiguration möglich ist.
2.d)	Ja, da für die in der Implementierung flexible, austauschbare und wiederverwendbare Modellerweiterungen und Analysen bis zur Erstellung dieser Dissertation keine Automatisierungslösung existierte. Insbesondere nicht für die Unterstützung verschiedener Analysearten und Wissensdomänen.
3.a)	Ja.
3.b)	Ja, denn Analysen lassen sich in diesem Rahmenwerk flexibel definieren, konfigurieren, kombinieren und ausführen. Dabei werden komplexe Detailkonfigurationen (Workflows) computergestützt erstellt, sodass der Nutzer diese nicht selbst erzeugen muss. Dadurch werden trotz der hohen Flexibilität Abhängigkeiten und Konsistenz computergestützt sichergestellt.
3.c)	Ja, wie Abschnitten 3.7.2-3.7.6 zu entnehmen ist.
3.d)	Ja, vgl. Abschnitt 3.7.7.

Wie die, aus der Evaluation resultierenden, Antworten auf die Fragen zur Zielerreichung zeigen, konnten die festgelegten Ziele alle erreicht werden.

3.7.8.1 Auswirkungen auf die Bedrohungslandschaft

In diesem Abschnitt werden durch die hier vorgestellte Arbeit erreichte Auswirkungen auf die Bedrohungslandschaft diskutiert.
Wie zuvor ausgiebig diskutiert erweitert das Rahmenwerk die Abdeckung sicherheitsrelevanter Informationen und erhöht zusätzlich die Genauigkeit von Analysen durch optimierbare, austauschbare Modellverarbeitungs- und Analyseschritte. Die führt trivialer Weise dazu, dass mehr Bedrohungen, Schwachstellen usw. gefunden werden können. Daneben konnte aber auch gezeigt werden, welche positiven Auswirkungen die hier vorgestellten Konzepte und Methoden auf die detektierbaren Effekte und Angriffe haben kann. Dies wird in den folgenden zwei Paragraphen dediziert diskutiert.

3.7.8.1.1 Detektierbare Effekte

Zur Nachvollziehbarkeit der Auswirkungen der Lösung auf die Bedrohungslandschaft ist zunächst zu prüfen, welche Effekte (vgl. Abschnitt 2.6.3) mit der Lösung entdeckt werden können. Intra-Konfigurationseffekte werden in dieser Thesis nicht dediziert betrachtet, da sich hierfür Prüfalgorithmen, die in die jeweilige Konfigurationsschnittstelle eingebettet sind, besser eignen als externe Werkzeuge. Am Fall der effektiven Konfiguration von Firewalls (vgl. Abschnitt 3.4.2.1) ist jedoch nachvollziehbar, dass die hier präsentierte Lösung eine Detektion dieser Effekte trotzdem unterstützt.
Das Auffinden von Inter-Konfigurationseffekten ist immer abhängig von den konfigurationsübergreifenden Regeln, welche die Widersprüche formalisieren. Solche Regeln können als Konfigurations- oder Konformitätsregeln definiert werden. Beides wurde in der Evaluation gezeigt und somit auch die Abdeckung von Inter-Konfigurationseffekten nachgewiesen. Im Fall von NAC-Instanzübergreifenden Regeln ist die Effektive Konfiguration die erste skalierbare Lösung im Rahmen ontologiebasierter Sicherheitsanalyse.

3.7.8.1.2 Angriffserkennung

Existierende, signaturbasierte Lösungen mit und ohne probabilistischen Modellen können mithilfe des SyMP-Rahmenwerks und der vorgestellten Analysestrategie ebenfalls eingesetzt werden.
Zusätzlich können mit SyMP Filtermechanismen integriert werden, welche die Menge von Anomaliemeldungen einschränken und somit die Erkennung von unbekannten, als auch bekannten Angriffen verbessern (vgl. Abschnitt 3.7.6). Zudem können selbst unbekannte Angriffe durch das Clustern von Anomalien und die Klassifikation von Meldungen effektiver entdeckt werden. Dies wurde in Abschnitt 3.7.6 anhand der Verknüpfung mit Wissen aus dem ICS-ATT&CK®-Rahmenwerk gezeigt. Auf diesen Weg kann flexibel öffentlich verfügbares Wissen in das Modell integriert werden und somit ein echter Mehrwert für die Angriffserkennung geschaffen werden.
Zudem wurde in Abschnitt 3.7.6 gezeigt, wie dieses Wissen eingesetzt werden kann, um Angriffskorrelation zu betreiben und so die Grundlage für eine bessere Vorfallbehandlung zu schaffen.

3.7.9 Belege für die aufgestellten Hypothesen

Im Folgenden soll diskutiert werden, ob die Hypothesen aus Abschnitt 1.1.1 belegt werden konnten. Die Hypothesen sind zur besseren Übersichtlichkeit noch einmal in Tabelle 3.18 gelistet.

Tabelle 3.18: Überblick der, für diese Evaluation relevanten, Hypothesen.

Referenz	Hypothese
Hyp. 1	Die Identifikation und Trennung typischer Phasen der Modellerweiterung und -analyse führt sowohl zur Wiederverwendbarkeit abgeleiteter Modellerweiterungen, als auch zur Reduktion nötiger, durchzuführender Modellerweiterungen je Analyse.
Hyp. 2	Separation-of-Concerns und der Einsatz von Ontologien kann automatisierte, austauschbare und wiederverwendbare Modellbildung, -erweiterung und -analyse ermöglichen, welche unterschiedliche Domänen für System- und Sicherheitswissen unterstützen.
Hyp. 3	Durch die automatisierte Vorinterpretation von NAC-Konfigurationen und die Modellierung der so abgeleiteten, effektiven Konfiguration kann ein Netzwerkmodell erzeugt werden, welches sowohl die Netzwerkendpunkt- und NAC-spezifischen Konfigurationen, als auch deren Beziehungen zueinander enthält und dabei die Wiederverwendbarkeit und Effizienz von Sicherheitsanalysen und Modellerweiterungen erhöht.
Hyp. 4	Die offene, werkzeugübergreifende Sprache AutomationML kann genutzt werden, um die automatisierte Extraktion sicherheitsrelevanter Informationen für industrielle Komponenten zu ermöglichen, die weder Selbstauskunft noch standardisierte Konfigurationsschnittstellen bieten.
Hyp. 5	Durch Separation-of-Concerns, den Einsatz von Ontologien und einer geschickten logikbasierten Umsetzung von Analysen, kann ein Analysesystem für mehrere gängige Analysearten geschaffen werden, welches die Ausnutzung von Synergien unterstützt.

SyMP trennt sechs Phasen von der Modellbildung bis zum Labeln von Analyseergebnissen. Abschnitte 3.7.2-3.7.6 haben gezeigt, wie dabei Phasenergebnisse für unterschiedliche Analysen wiederverwendet werden konnten und gleichzeitig analysespezifische Modellerweiterungen möglich waren, ohne Erweiterungen für weitere Analysen durchführen zu müssen. Dies kann als Nachweis der Hypothese 1 und als positive Antwort auf Forschungsfrage 1 interpretiert werden.
SyMP basiert auf dem Separation-of-Concerns-Paradigma (vgl. Abschnitt 3.7.7 MA1, MA5) und dem durchdringenden Einsatz von Ontologien (vgl. Abschnitt 3.7.7 MA1), welche sowohl die Basis für die eingesetzten Modelle, als auch für die Wissensrepräsentation rund um die Analysen bilden. Abschnitt 3.7.7 zeigt sowohl, dass die damit ermöglichte Modellbildung, -erweiterung und -analyse, automatisiert (GA1, IA3, IA5, IA6), austauschbar und wiederverwendbar (AA4, IA2, MA6) ist, als auch, dass dabei unterschiedliche Domänen für System- und Sicherheitswissen unterstützt werden (AA3, AA6, MA5, MA7). Mit diesem Wissen kann geschlussfolgert werden, dass SyMP die Hypothese 2 belegt und Forschungsfrage 2 bejaht.
Hypothese 3 wurde mithilfe der, in Abschnitt 3.4.2.1 beschriebenen, Methode zur Ableitung und Modellierung der Effektiven Konfiguration belegt. Somit bietet die Effektive Konfiguration eine positive Antwort auf Forschungsfrage 3.
Hypothese 4 konnte, wie in Abschnitt 3.4.1.1 detailliert erläutert, nachgewiesen werden. Dabei verbleibt eine Interpretationsungenauigkeit der Methode, die ohne Software-gestützte Validierung zu Inkompatibilität führen kann. Dies ist ein generelles Problem von AutomationML. Eine weitere Einschränkung des AutomationML-basierten Ansatzes ist die fehlende Nachweisbarkeit der vollständigen Abdeckung aller, in Engineering-Werkzeugen darstellbaren, Netzwerkinformationen mithilfe der vorgestellten Methode. Das Ziel, die automatisierte Extraktion sicherheitsrelevanter Informationen für industrielle Komponenten zu ermöglichen, die weder Selbstauskunft noch standardisierte Konfigurationsschnittstellen bieten, konnte durch die Extraktion dieser Informationen aus Engineering-Werkzeugen jedoch erreicht und, wie in Abschnitt 3.4.1.1 erörtert, exemplarisch nachgewiesen werden. Zusammen mit der automatisierten Informationsextraktions- und Modellbildungsmöglichkeit durch SyMP, ist die Methode also eine Antwort auf Forschungsfrage 4.

Zur offen formulierten Forschungsfrage 5 liegt keine Hypothese zu deren Beantwortung vor. Stattdessen wurde diese mithilfe der Untersuchung aus *[Pat19d]* adressiert und in Abschnitt 3.4.1.2 zusammengefasst. Das Ergebnis ist weder eindeutig noch allgemeingültig. Allerdings hat sich der Einsatz von HTTPS und OPC UA sowie die in der Untersuchung evaluierte, komplexe Abbildungsstrategie (umgesetzt durch X2Owl) in der in diesem Kapitel beschriebenen Evaluation als flexible und geeignete Umsetzung einer Verwaltungsschale für die Sicherheitsanalyse erwiesen. Somit kann diese Umsetzung als eine Antwort auf Forschungsfrage 5 betrachtet werden.
Dass wertvolle Synergien zwischen Analysearten durch das Konzept der Regelbasierten Analyse ausgenutzt werden können, wurde in den Abschnitten 3.7.2-3.7.6 deutlich gemacht. Besonders die Evaluation der Konformitätsanalyse (vgl. Abschnitt 3.7.3) hat gezeigt, dass auch innerhalb einer Analyseart Synergien der Analysen ausgenutzt werden können. Zudem wurde durch die generische Definition von Analyseregeln als deren Ausführungsbestandteile die Möglichkeit eröffnet, verschiedene Analysen unterschiedlicher Analysearten mit der selben Syntax und Semantik zu beschreiben und durch die selben Werkzeuge zu verwalten. Damit konnte auch Hypothese 5 nachgewiesen und somit Forschungsfrage 6 bejaht werden.

3.7.10 Diskussion technologischer Aspekte

Die Evaluation zeigt, dass mithilfe von SyMP die Entwicklung einer Lösung möglich ist, welche die in diesem Kapitel vorgestellten Ansätze konsistent und effizient unterstützt.
Im Rahmen der Evaluation konnten verschiedene Ressourcen-aufwändige Schritte identifiziert werden. So stellte sich die automatisierte Umwandlung von Syslog-Daten in OWL als besonders rechenintensiv heraus. Bei 389456 Syslog-Zeilen benötigte die Umwandlung mit ca. 75% CPU-Auslastung durch den Python-Prozess ca. 26 Stunden.
Durch die Unabhängigkeit der Zeilen könnte diese Umwandlung auch mittels Devide-and-Conquer-Verfahren parallelisiert werden. Für kontinuierliche Umwandlung, d.h. das direkte Umwandeln einer Log-Meldung sobald diese eintrifft, besteht im Evaluationsszenario kein Handlungsbedarf, da die Logs im Schnitt alle 20 Sekunden eintreffen und die Umwandlung einer einzelnen

Zeile weniger als eine Sekunde benötigt (inkl. öffnen, auslesen und schließen der Log-Datei, exklusive der Übertragung der Datei). Dabei ist darauf zu achten, dass die Analysen dann eventuell nicht bei jedem Eintreffen einer neuen Meldung durchgeführt werden können, da die Ausführung der SyMP-Phasen und die darauffolgende Abfrage, je nach Analysekomplexität, länger als die angegebenen 20 Sekunden benötigt. Entsprechende Pufferung der Meldungen bis die Analyse durchgelaufen ist, wäre jedoch eine entsprechende Lösung für diese Problematik.
Eine weitere Problematik stellt die beobachtete, deutlich unterschiedliche Laufzeit der Aufgabenabarbeitung dar, die bei der Generierung von Workflows aktuell nicht einbezogen wird und somit zu unbeschäftigten Prozessen im Rahmen von parallelisierten Aufgaben führt. Da das zu analysierende Modell zum Zeitpunkt der Workflow-Generierung nicht bekannt ist, gleichzeitig aber die tatsächliche Laufzeit der Aufgabenabarbeitung von der Anzahl der zu bearbeitenden Aussagen im Modell abhängt, kann die tatsächliche Laufzeit der Aufgabenabarbeitung auch nicht bei der initialen Workflow-Generierung berücksichtigt werden.
Die Workflow-Generierung würde demnach von alternativen Metriken und vor allem Heuristiken profitieren, welche eine verbesserte Ressourcennutzung mithilfe optimiert parallelisierter Workflows ermöglichen würde.
Ein ebenfalls wahrscheinlich erfolgversprechender Ansatz wäre das Überwachen eines Workflows während seiner Ausführung und die Optimierung des selbigen nach dem Durchlauf anhand der zeitabhängigen Gewichtung der Schritte. Das SyMP-Konzept lässt eine häufige Analyseausführung zu, die beispielsweise fester Bestandteil eines Änderungsmanagementprozesses sein kann. In einem solchen Szenario könnte die genannte Optimierung bereits ab der zweiten Ausführung einen deutlichen Performance-Gewinn erreichen, da sich im Rahmen einer Änderung selten das gesamte System und somit nur einzelne Bereiche des Modells ändern und die Ausführungszeiten der Schritte daher wenig variieren.
Mit der Laufzeit der Arbeiter hängt auch ein weiteres Problem zusammen. Wider Erwarten haben sich die verschiedenen, für die Ontologiebearbeitung

eingesetzten, Bibliotheken Jena, OWLAPI und *owlready2*[1] als äußerst schlecht skalierbar und unvollständig herausgestellt. Besonders Reasoning und Suche auf und in den geladenen Modellen war mit diesen Bibliotheken so unperformant, dass diese Operationen für einige Arbeiter selbst bei Ontologien unter 100.000 Axiomen durch Individualprogramme ersetzt werden mussten.
Auch stellte sich heraus, dass die Präfixe der verschiedenen Ontologien von owlready2 beim Speichervorgang durch eigene, automatisch aus den Namensräumen abgeleitete Präfixe ersetzt wurden. Dieser Zustand verlangte ein zusätzliches Anpassen der Modelldateien nach Ausführung der entsprechenden Arbeiter, welches durch ein zusätzliches Bearbeiten mit einem XML-Parser nötig machte, um die Anpassung zu automatisieren. Dadurch wurde die Laufzeit der Arbeiter im Schnitt verdoppelt.
Dies führte dazu, dass die Implementierung des SyMP-Rahmenwerks zum Zeitpunkt des Verfassens dieser Dissertation bereits von der dateibasierten Variante auf die Graph-Datenbank Neo4j[2] mit RDF4J[3] umgestellt wurde. Die Umstellung hat zur Folge, dass die Verwendung der Bibliotheken bestehen bleibt, da sich weiterhin Ontologien von der Datenbank abfragen lassen. Gleichzeitig können jedoch die Performance-Vorteile der Graph-Datenbank bezüglich Reasoning und semantischer Suche genutzt werden. Der Nachteil ist jedoch die zusätzlich zu implementierende Versionierung der Modelle in der Datenbank und die zusätzliche Verwendung von Datenbankmanagementsystem-abhängigen Wrappern in den Arbeiter-Implementierungen.
Eine vollständige Abdeckung von für beliebige Sicherheitsanalysen relevanten Informationen ist kein realistisches Ziel. SyMP löst das Problem auf semantischer Ebene, indem über die Abhängigkeiten zwischen Analyseregeln und Ontologien sichergestellt wird, dass die für eine Analyse nötigen Konzepte ab einem bestimmten Erweiterungsschritt im Modell verfügbar sind. Dies stellt jedoch weder sicher, dass die Konzepte auf entsprechende, verfügbare Aussagen einer anderen Domäne abgebildet werden, noch dass alle dem realen

[1] https://pypi.org/project/Owlready2/, zuletzt zugegriffen: 20.02.2021.
[2] https://neo4j.com/, zuletzt zugegriffen: 20.02.2021.
[3] https://rdf4j.org/, zuletzt zugegriffen: 20.02.2021.

System entsprechenden Aussagen Teil des Modells sind. Vereinfacht ausgedrückt, stellt SyMP nicht sicher, dass einer Analyse alle Informationen zur Verfügung stehen, die im realen System verfügbar sind, sondern nur, dass eine konfigurierte Analyse auch ausgeführt werden kann.
SyMP erreicht einen deutlich höheren Automatisierungs- und Wiederverwendbarkeitsgrad als verwandte Lösungen. Dies gilt sowohl für analysespezifische Modellverarbeitung und Auswertung als auch für die Erstellung und Konfiguration von Analysen. Die Konfiguration der ersten drei SyMP-Phasen, also jene, die sich mit der Erstellung des initialen Systemmodells befassen, sind aktuell noch auf manuelle Erstellung der Workflows und Abbildungen angewiesen. Implementierungen von SyMP sollten für diese Bereiche jedoch eine ähnliche Automatisierungs- und Wiederverwendbarkeitsunterstützung zur Verfügung stellen wie für den Analysebereich.
Außerdem bekommen Endnutzer aktuell noch keine Unterstützung bei der Kombination von Analyseregelumsetzungen und -formulierungen auf dem SyMP-Hub, die aber erstrebenswert ist, sobald mehr als eine Umsetzung für eine Definition zur Verfügung steht. Hier sollen in Zukunft Metadaten der ersten drei SyMP-Phasen genutzt werden, die der Endnutzer in seiner SyMP-Instanz (und seinem System) einsetzen kann, um automatisiert abzuleiten, welche Umsetzungen für den Endnutzer überhaupt anwendbar und welche davon zu präferieren sind. Das SyMP-Analyseregelkonzept unterstützt solche Vorschlagsysteme (Recommender-Systeme) bereits, denn neben der Implementierungs-Agnostik und einfachen Erweiterbarkeit wurde die Ontologierepräsentation der Analyseregeln genau für diese Anwendungen gewählt.
Neben der Identifikation von Erweiterungsmöglichkeiten und Implementierungsproblemen konnte die Evaluation die Umsetzbarkeit und Sinnhaftigkeit des SyMP-Rahmenwerks vollständig bestätigen. So wurde sichergestellt, dass das Rahmenwerk einen wertvollen Fortschritt in Richtung Praxistauglichkeit von ontologiebasierten Sicherheitsanalysen – und somit der minimalinvasiven Sicherheitsanalyse industrieller Systeme – bietet.

3.7.11 Zusammenfassung

Wie in Abschnitt 3.7.7 erörtert wurde, konnten die in Abschnitt 3.2 festgelegten Anforderungen für eine wissensbasierte, minimalinvasive Sicherheitsanalyselösung für industrielle Systeme erfüllt werden. Außerdem konnten die in Abschnitt 1.2 gesteckten technischen Ziele erreicht werden (vgl. Abschnitt 3.7.8). Die Schlussfolgerungen zur Anforderungserfüllung und dem Erreichen der Ziele ermöglichte es, in Abschnitt 3.7.9 eine fundierte Argumentation zum Beleg der anfangs aufgestellten Hypothesen führen zu können. Die Hypothesen konnten dabei bestätigt werden. Daraus resultierend, konnten die Forschungsfragen aus der Problemstellung (vgl. Abschnitt 1.1.1) beantwortet werden. Dadurch konnte, wie in der Problemstellung motiviert, ein relevanter Beitrag zu aktuellen Forschungsbedarfen geleistet werden.

4 Automatisierte, minimalinvasive Vorfallreaktion

Vorfälle, wie sie von der Angriffserkennung und -korrelation aus Abschnitt 3.7.6 erkannt werden, erfordern eine zeitnahe und nicht-gefährdende Reaktion. Diesbezüglich werden in diesem Kapitel zunächst verwandte Arbeiten zur automatisierten Vorfallreaktion (AIR) adressiert (vgl. Abschnitt 4.1). Die für die Umsetzung einer geeigneten Lösung relevanten Anforderungen und Vorüberlegungen, werden daraufhin in den Abschnitten 4.2 und 4.3 vorgestellt. In Abschnitt 4.4 wird das Konzept (im Folgenden als *SDN-AIR* abgekürzt) zur automatisierten, minimalinvasiven Vorfallreaktion für SDN beschrieben. Dieses Konzept wurde prototypisch umgesetzt (vgl. Abschnitt 4.5) und mithilfe dieser Umsetzung evaluiert. Die Evaluationsergebnisse werden in Abschnitt 4.6 präsentiert und diskutiert.

4.1 Stand von Wissenschaft und Technik

In diesem Abschnitt werden die verwandten Arbeiten zur automatisierten Vorfallreaktion mit und ohne den Bezug auf industrielle Systeme adressiert. Wie bereits erwähnt, konzentrieren sich die in dieser Arbeit vorgestellten Konzepte auf Drehbuch-basierte Vorfallreaktion, d.h. Reaktionen, welche im Vorfeld geplant wurden. Der Grund hierfür ist die Einsetzbarkeit in kritischen Umgebungen, für die eine in das System eingreifende Maßnahme deterministisch und vorhersehbar sein muss, um Gefährdung durch physische Auswirkungen des Eingriffs im Vorfeld auszuschließen, oder zumindest abschätzen zu können. In der Forschung sind jedoch auch Arbeiten zu finden, die sich mit Reaktionsstrategien für die Reaktionsauswahl befassen [Has08, Oss15].

Zum Ermitteln der Grundlagen solcher Entscheidungen gibt es die Forschungsdisziplin *Threat Intelligence*, in der Informationen über Bedrohungen gesammelt und ausgetauscht werden, um daraus Präventions- und Gegenmaßnahmen abzuleiten. In der Regel stammt dieses Bedrohungswissen aus beobachteten Vorfällen und kann genutzt werden, um Ereignisse zu klassifizieren oder mögliche Bedrohungen für das Umfeld eines Unternehmens zu identifizieren. Letzteres wird durch Datenbanken über bekannte Schwachstellen unterstützt, die nützlich sind, um die Relevanz einer Bedrohung in einer bestimmten Umgebung zu beurteilen. So kann geprüft werden, ob eine bestimmte Klasse von Schwachstellen in der Umgebung vorliegt und, falls dies der Fall ist, geschlussfolgert werden, dass Bedrohungen, welche diese Klasse von Schwachstellen nutzen, aktuell besonders relevant sind. Somit können beispielsweise Gegenmaßnahmenumsetzungen priorisiert werden. Es existieren etablierte Plattformlösungen, wie MISP[1] und OPENCTI[2], die Threat Intelligence unterstützen.
Der Trend, die Informationen über konkrete Bedrohungen als Selektierwerkzeug für entsprechende Handlungsempfehlungen zu verknüpfen, ist seit einiger Zeit zu beobachten. Dies wird beispielsweise als besonders wichtige Funktion in den Produkten von Dragos Inc. [Ser18] und Darktrace [Dar20] kommuniziert. Letzteres automatisiert diesen Prozess nach eigenen Angaben so weit, dass selbst Gegenmaßnahmen automatisiert eingeleitet werden [Dar20]. Hierzu konnten allerdings keine Details und Veröffentlichungen gefunden werden.
Eine der ersten Arbeiten dieser Art wurde von Chung et al. präsentiert [Chu13] und ist besonders interessant, da in dem IPS-Ansatz unter anderem die Invasivität der Reaktion in den Entscheidungsprozess zur Reaktionsselektion miteinbezogen wird. Allerdings muss der Lösung eine Angriffsprädiktion vorliegen und für einen Angriffsschritt müssen genug Vergangenheitsdaten verfügbar sein, um die nötigen Wahrscheinlichkeitsverteilungen bilden zu können.
Die Lösung von Chung et al. gehört zu einer Entwicklung im IPS-Bereich, die sich dadurch manifestiert, dass IPS zu zentralisiert auf Virtualisierungen, wie Cloud-Computing, arbeitenden intelligenten Systemen transformiert

[1] https://www.misp-project.org/, zuletzt zugegriffen am 19.08.2020.
[2] https://www.opencti.io/en/, zuletzt zugegriffen am 19.08.2020.

wurden [Jin13, Xin14]. Zuvor wurden IPS lediglich als Zusatzfunktionen von Netzwerk-Sicherheitsmaßnahmen eingesetzt, die entweder auf Hosts oder Netzwerkkomponenten existierten [Bat04, Ras05, Gon07]. Der entscheidende Vorteil hierbei ist die Gesamtsicht auf das Netzwerk, die einige Reaktionsmöglichkeiten und Entscheidungsfindungen überhaupt erst möglich machen. In diesem Rahmen wurden IPS auch bereits auf SDN-basierten Netzwerken erforscht [Xin14, Amm16, Chi17], die in Virtualisierungsumgebungen gerne für Verwaltung der Netzwerke auf Basis der angesprochenen Gesamtsicht eingesetzt werden.
Es gibt jedoch auch Forschungsarbeiten zu SDN-basiertem AIR/IPS, die nicht auf Virtualisierungsumgebungen spezialisiert sind. Besonders relevant für diese Dissertation ist hierbei *Software-Defined Networking for Security (SDN4S)* [Kou17], ein SDN-basiertes AIR-System, das automatisierte Gegenmaßnahmen zur Minimierung der Reaktionszeit nutzt. Ein Architekturüberblick von SDN4S ist in Abbildung 4.1 zu sehen.

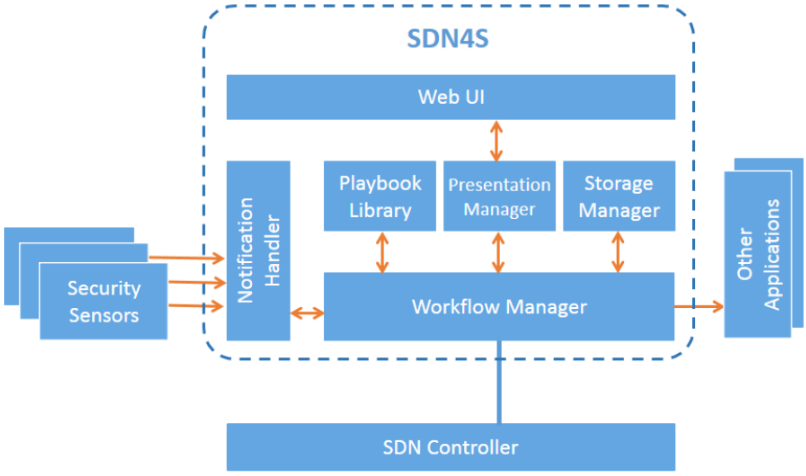


Abbildung 4.1: Architektur von SDN4S. *Quelle:* [Kou17].

Die Autoren nutzen das Konzept der Drehbücher (bzw. *Playbooks*), welches aus einer Reihe von Auslösern (entsprechend dem im Drehbuch behandelten Vorfall) und einer Reihe von ausführbaren Aktionen, also den Reaktionsmöglichkeiten, besteht. Drehbücher beschreiben also auf deterministische Weise die Entscheidungen, welche ein SDN-Controller bei der Ankunft eines Vorfallalarms treffen muss. SDN4S definiert mit dem *Notification Handler* die Schnittstelle zum Entgegennehmen von Vorfallmeldungen. Die zur Meldung passenden Reaktionsentscheidungen werden dann von einem *Workflow-Manager* aus der *Playbook-Library* selektiert und zu entsprechenden Handlungsabläufen zusammengebaut. Hierzu gehört das Erstellen der Flows, mithilfe des SDN-Controllers, die dann auf den SDN-Switches installiert werden. Die weiteren Komponenten dienen der Konfiguration, Verwaltung und Überwachung der Anwendung (vgl. *Web UI*), dem Extrahieren zusätzlicher Informationen (vgl. *Storage Manager*) und als Ausführungsumgebung für Aufgaben, bei denen keine Flows installiert werden (vgl. *Other Applications*). Da SDN4S nicht für ICS-Umgebungen ausgelegt ist, werden in die Entscheidungen keine spezifischen Anforderungen, wie Verfügbarkeit, Zuverlässigkeit, Betriebssicherheit, Zeitempfindlichkeit und Redundanz, mit einbezogen. Dennoch ist dieser Ansatz eine gute Grundlage für AIR in industriellen Netzwerken und wird in Abschnitt 4.4 entsprechend wiederaufgegriffen.

Auch für industrielle Systeme wurden bereits automatisierte Vorfallreaktionsstrategien untersucht, die über die klassischen Netzwerkverkehr-Filtermechanismen hinausgehen. Di Lallo et al. [Di 17] befassten sich mit den Echtzeitanforderungen für SDN-basierte ICS-Sicherheit. In ihrem Ansatz nutzen sie nicht ausgenutzte Bandbreite, um Replikate der gesendeten Pakete innerhalb des Netzwerks an ein Intrusion Detection Systems (IDS) zu übertragen. Dies zeigt, wie SDN-basiertes AIR in der Identifikationsphase der Reaktion auf Vorfälle helfen kann (vgl. Abschnitt 2.10), ohne die Dienstqualitätsgarantien des industriellen Netzwerks zu gefährden.

Piedrahita et al. zeigen in [Mur18a] und [Mur18b] eine auf SDN und *Network Function Virtualization* (*NFV*) basierte AIR-Lösung für industrielle Netzwerke. Die Autoren schlagen virtuelle Vorfallreaktionsfunktionen vor, die angegriffene industrielle Komponenten ersetzen. Wenn ein Angriff auf eine Komponente identifiziert wird, wird die virtuelle Funktion verwendet, um

einen sogenannten Honeypot zu erstellen. Dieser wird dem Angreifer statt der realen Komponente ausgeliefert, indem der Angriffsverkehr mittels SDN auf den Honeypot umgeleitet wird. Der Methode liegt allerdings die Annahme zugrunde, dass der gesamte Angriffsverkehr umgeleitet wird. In der Realität ist es jedoch oft sehr schwer zu erkennen, welcher Netzwerkverkehr tatsächlich zu einem Angreifer gehört. Als weitere Alternative schlagen die Autoren vor, die angegriffene Komponente mit einer Virtualisierung zu ersetzen. Die Praxistauglichkeit dieses Ansatzes wird nicht nachgewiesen.
Eine Lösung, welche die nun vielfach genannten, relevanten Eigenschaften industrieller Systeme berücksichtigt und gleichzeitig die Vorfallbehandlung durch automatisierte Vorfallreaktion unterstützt, wurde bis zum Beginn der Arbeiten zu dieser Dissertation nicht vorgestellt.

4.2 Anforderungen für automatisierte Vorfallreaktion

In diesem Kapitel werden Anforderungen für die automatisierte Vorfallreaktion vorgestellt. Zunächst wird ein Angreifermodell entwickelt, gegen das die Sicherheitslösung evaluiert werden kann. Danach werden umgebungsspezifische und generelle Anforderungen definiert.

4.2.1 Angreifermodell

Die Sicherheit einer Lösung bezieht sich immer auf Annahmen darüber, welche Fähigkeiten ein Angreifer besitzt. Im Folgenden wird ein Angreifermodell vorgestellt, mit dessen Hilfe die Sicherheit einer Lösung zur automatisierten Vorfallreaktion untersucht werden kann.
Abbildung 4.2 zeigt für das AIR-Szenario, auf welche Bereiche der Angreifer im Rahmen des Modells einen Einfluss hat. Dabei besitzt der Angreifer die Fähigkeiten eines Dolev-Yao-Angreifers [Dol83] mit zwei im Folgenden definierten Einschränkungen. Wie bereits erwähnt, ist ein Dolev-Yao-Angreifer ein (polynomiell-beschränkter) Teilnehmer des Netzwerks, der die Fähigkeit hat, Nachrichten im Netzwerk zu senden, zu empfangen und zu modifizieren.

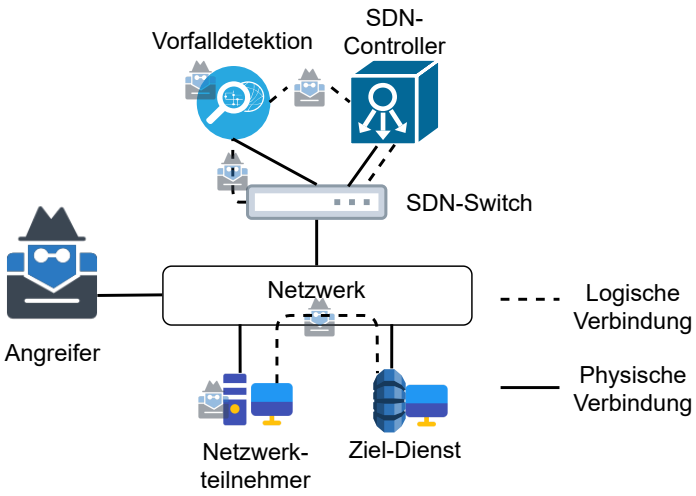


Abbildung 4.2: Abbildung des Angreifermodells für das AIR-Szenario, die zeigt, auf welche Bereiche der Angreifer einen Einfluss hat.

Als erste Einschränkung, ist der hier definierte Angreifer allerdings nicht in der Lage, SDN-Flows hinzuzufügen, zu entfernen oder zu modifizieren. Von seinem direkten Einfluss ausgeschlossen ist also der Netzwerkverkehr zwischen dem SDN-Controller und dem SDN-Switch sowie die beiden Komponenten selbst. Ohne diese Einschränkung wäre die Sicherheitsanalyse des Konzepts unnötig, da er das Netzwerk bereits direkt kontrollieren würde und das AIR-Konzept nicht ausnutzen müsste, um seine Ziele zu erreichen. Er kann also nur auf der Datenebene beliebig Netzwerkverkehr mitlesen, manipulieren und unterbinden.
Auch ist er in der Lage, Endgeräte zu kompromittieren, bekommt jedoch ein Angriffsziel in Form eines beliebigen aber festen Dienstes, welcher nicht direkt kompromittiert werden kann. Gäbe es diese zweite Einschränkung nicht, wäre eine Analyse unnötig, denn ein Angreifer, der jeden beliebigen Dienst (bzw. das Angriffsziel) direkt beeinflussen kann, muss das AIR-Konzept nicht ausnutzen, um seine Ziele zu erreichen.

4.2.2 Domänenanforderungen

Als wichtigste Anforderung für ein Konzept zur automatisierten Vorfallreaktion gilt im Kontext dieser Dissertation die Einsetzbarkeit in industriellen Systemen. Was dies bedeutet, soll hier nun erörtert werden.
Die in industriellen Systemen besonders wichtigen Anforderungen umfassen beispielsweise Betriebssicherheit, Ausfallsicherheit, Echtzeitanforderungen und Determinismus [Gal13]. Um dies in industriellen Netzen sicherstellen zu können, müssen Redundanz-, Echtzeit- und weitere Dienstgüteanforderungen sichergestellt werden können [Gal13, Nat11]. Somit ist eine in das Netzwerk eingreifende Lösung nur dann in industriellen Systemen einsetzbar, wenn diese Anforderungen für Netzwerkteilnehmer, Netzwerkkomponenten und Verbindungen erfüllt und gewahrt werden können. Diese Aspekte wurden bereits im Grundlagenabschnitt 2.4 eingehend erläutert.

4.3 Vorüberlegungen

Das hier vorgestellte SDN-AIR-Konzept definiert keine eigenen Vorfalltypen und -meldungen, sondern bietet die Möglichkeit diese beliebig zu konfigurieren. Dadurch ist es für beliebige Domänen von Software-definierten Netzwerken einsetzbar. Meldungen werden entweder manuell oder durch einen beliebigen Sicherheitsmechanismus, wie die automatisierte, minimalinvasive Sicherheitsanalyse mit SyMP oder Host- und Netzwerk-basierte IDS, ausgelöst. Als Arbeitsgrundlage werden hier die abstrakten Vorfälle „kompromittierter Host“, „kompromittierter Switch“ und „bösartiger Link“ definiert. *Hosts* sind dabei beliebige Netzwerkteilnehmer, *Switches* sind SDN-Switches und ein *Link* beschreibt eine identifizierbare Ende-zu-Ende-Verbindung zwischen zwei Hosts (diese Verbindung muss nicht aktiv sein). Darüber hinaus wird in diesem Kapitel der Begriff *Knoten* sowohl zur Bezeichnung eines Hosts, als auch eines Switches verwendet.

4.3.1 Asset-Klassifikation

SDN-AIR basiert unter anderem auf der Klassifizierung von Assets. Hierfür werden in diesem Abschnitt Basisklassen vorgestellt. In verschiedenen Umgebungen können weitere Klassen zielführend sein, weshalb bei der Konzipierung von SDN-AIR die Erweiterbarkeit auf zusätzliche Klassen berücksichtigt wurde.
Bestimmte Hosts und Links sind von entscheidender Bedeutung, z.B. für die Steuerung einer Produktionsanlage oder für die Ausführung von Betriebssicherheitsfunktionen. Typischerweise darf die Datenübertragung zwischen solchen Hosts oder über solche Verbindungen nicht unterbrochen werden. Daher wird für solche Hosts und Links die Klasse *functionally-critical* (funktionskritisch) eingeführt. Auch die Zeitkritikalität ist ein wichtiger Aspekt innerhalb des industriellen Systems. Bestimmte Links (und Knoten) müssen bestimmte Zeitanforderungen erfüllen und einhalten. Diese Assets werden daher als *time-critical* (zeitkritisch) klassifiziert.
Zuletzt wird noch die Klasse *redundant* eingeführt. Redundante Netzwerkpfade werden in industriellen Systemen oft eingebaut, um die Wahrscheinlichkeit zu erhöhen, dass im Falle eines Angriffs oder einer Störung ein Dienst funktionsfähig bleibt, da bei Ausfall eines Pfades ein Alternativpfad verwendet werden kann. Bei solchen redundanten Pfaden ist es ggf. wichtig, dass sie einen disjunkten Satz von physikalischen Übertragungsknoten enthalten. Daher kann die Umleitung oder Unterbrechung redundanter Verbindungen unerwünscht sein. Folglich können anstelle von Pfaden auch Links als redundant deklariert werden. Dann gilt, dass einem als redundant klassifizierten Link mehr als ein Pfad zur Verfügung stehen muss und die entsprechenden Pfade physisch disjunkt sein müssen (bis auf den ersten und letzten Netzwerkknoten). Zudem ist es möglich redundante Knoten einzusetzen.
Zuletzt bleibt zu erwähnen, dass Links und Knoten zu mehr als einer Klasse gehören können. So sind zum Beispiel Verbindungen im Rahmen der Betriebsicherheit üblich, die sowohl funktions- als auch zeitkritisch sind. Folglich bezieht sich die Klassifizierung eines Assets auf die Menge der Klassen, denen es zugeordnet ist.

4.3.2 Reaktionsmöglichkeiten

Bevor auf das eigentliche Konzept und seine Architektur eingegangen wird, sollen hier noch mögliche automatisierte Reaktionen vorgestellt werden. Die Liste der Reaktionsmöglichkeiten ist nicht abgeschlossen, enthält aber die generellen, auf industrielle Systeme anwendbaren Reaktionen. Durch den bereits erwähnten Drehbuchansatz von SDN-AIR können beliebige weitere Reaktionsmöglichkeiten hinzukommen.

Isolation von Hosts

Man nehme an, dass eine Warnung darauf hindeutet, dass ein Host kompromittiert ist. Dann könnte eine angemessene Reaktion darin bestehen, diesen Host vom Rest des Netzwerks zu isolieren. Zu diesem Zweck kann der SDN-Controller Flow-Einträge mit hoher Priorität auf dem SDN-Switch installieren, mit dem der Host direkt verbunden ist. Diese Flow-Einträge würden dann vom Switch verlangen, alle Pakete von und zu diesem Host zu verwerfen.
Eine weitere Strategie ist die Isolierung des Hosts und bestimmter Kommunikationspartner vom Rest des Netzwerks, aber nicht voneinander. Dies ist zum Beispiel dann relevant, wenn der Host mit diesen Kommunikationspartnern funktionskritische Verbindungen pflegt. So kann ein Angriff in seiner Ausbreitung behindert werden und es werden die Flows aufrechterhalten, welche die kritische Verbindung ermöglichen. Da Flow-Einträge für spezifische Verbindungen definiert werden können (beispielsweise für bestimmte TCP/IP-Verbindungen[1]), kann diese partielle Isolation durch den Einsatz von zwei verschieden Prioritätsklassen von Flows erreicht werden: Hochpriore Flows, welche die kritische Verbindung ermöglichen und Flows niedrigerer Priorität, die alle Pakete von und zu dem betreffenden Host verwerfen.

Isolation von Switches

Wie Hosts können auch SDN-Switches vom restlichen Netzwerk isoliert werden, wenn sie kompromittiert zu sein scheinen. Eine solche Vorfallreaktion

[1] Hier sind nicht aktive, sondern mögliche Verbindungen gemeint, die entsprechend über IP-Port-Kombinationen identifiziert werden.

hat jedoch wesentlich mehr Auswirkungen auf das restliche Netzwerk als die Isolierung eines Hosts. Die Isolierung von Switches kann dazu führen, dass auch andere Knoten isoliert werden. Wenn daher eine kritische Verbindung nicht über Pfade umgeleitet werden kann, die den kompromittierten Switch nicht enthalten, wird die Verfügbarkeit von Diensten in unakzeptabler Weise beeinträchtigt.

Blockieren einzelner Links

Das Blockieren bestimmter Links ist ebenfalls eine mögliche Reaktion, insbesondere auf Denial-of-Service-Angriffe, bei denen es darum geht, die Verfügbarkeit eines Dienstes mittels einer hohen Anzahl von Anfragen, oder spezifischen, für den Dienst besonders fordernden Anfragen zu beeinträchtigen. Falls beispielsweise ein Knoten innerhalb des Netzwerks andere durch einen Denial-of-Service-Angriff beeinträchtigt, können die für diesen Angriff verwendeten Links oft sehr schnell identifiziert und dann blockiert werden. Dadurch kann erreicht werden, dass nicht gleich der gesamte Knoten isoliert wird, was für die Funktionalität und Betriebssicherheit des Systems entscheidend sein kann. Dieses Beispiel zeigt, dass die Blockierung von Verbindungen als schwächere Reaktionsalternative zur Isolierung ganzer Knoten gewählt werden kann.

Überwachung

Eine der wohl praktikabelsten Reaktionen ist die Replikation von Paketen zur Überwachung. So kann beispielsweise die Überwachung an bestimmten Stellen im Netzwerk eingeschaltet werden. Während in traditionellen Switches mit Mirroring-Funktionalität (Kopieren von Netzwerkverkehr und Senden der Kopien an eine Senke) oder Netzwerk-TAPs (dedizierte Hardware zur Überwachung von Netzwerkverkehr) mit der Fähigkeit zur Paketreplikation benötigt wurden, kann das SDN die Spiegelung auf jedem SDN-Switch durchführen, indem entsprechende Flow-Einträge eingesetzt werden. Diese Reaktion erzeugt zusätzlichen Verkehr, beeinträchtigt das Netzwerk darüber hinaus aber nicht. Wie bereits erwähnt, schlugen Di Lallo et al. [Di 17] einen Ansatz vor, um diesen negativen Effekt durch das Ausnutzen nicht genutzter Bandbreite

zu minimieren.
Wenn die Replikation von Paketen zur Überwachung eines Hosts durchgeführt wird, kann jeder SDN-Switch, mit dem der Host direkt verbunden ist, als Replikationspunkt ausgewählt werden.
Ist die Überwachung eines Switches erwünscht, besteht das Ziel im Allgemeinen in einer Maximierung der Anzahl der überwachten Links, die über diesen Switch geleitet werden. Bei Verbindungen, die den jeweiligen Switch als einzigen Netzwerkknoten verwenden, ist eine vertrauenswürdige Überwachung nur dann möglich, wenn der Switch nicht kompromittierungsverdächtig ist und daher die Pakete selbst replizieren kann. Die Pakete anderer Links können von anderen SDN-Switches entlang ihrer Pfade repliziert werden.

Virtualisierung

Laut Piedrahita et al. ist es möglich, ein angegriffenes System mit mehreren Virtualisierungsstrategien zu unterstützen und einen vom Angriff beeinträchtigten Host dynamisch durch seine virtuelle Repräsentation zu ersetzen [Mur18a, Mur18b]. Darüber hinaus kann dies sogar dahingehend erweitert werden, dass physikalische Prozesse durch entsprechende Simulationen ersetzt werden. Auch wenn die Praxistauglichkeit dieses Ansatzes nicht nachgewiesen wurde, ist dies technisch möglich und könnte daher seine Daseinsberechtigung haben. Von den Autoren wird als Beispielkomponente ein Sensor genannt. Gerade das Ersetzen eines Sensors durch eine von weiterhin verfügbaren Werten abhängige Simulation, könnte sicherer sein, als ein Ausbleiben der Sensorwerte. Der Virtualisierungsansatz kann also potenziell tatsächlich dazu genutzt werden, die Ausfallsicherheit eines Systems zu verbessern.

Benachrichtigung

Die SDN-AIR-Lösung kann Benachrichtigungen erstellen, welche die ursprüngliche Vorfallmeldung um zusätzliche Informationen ergänzen. Die Benachrichtigungsaktion kann zu mehreren Zeitpunkten in der automatisierten Vorfallreaktion eingesetzt werden. Sie kann direkt nach Erhalt einer Benachrichtigung gesendet werden, um den Administrator zu informieren und erste Reaktionsvorschläge zu machen, die nach Erteilung der Genehmigung automatisch

ausgeführt werden können. Darüber hinaus kann die Benachrichtigung mit wichtigem Wissen über das aktuelle Netzwerk-Layout (oder die Topologie) angereichert werden, um die Analyse zu unterstützen und so die Gesamtreaktionszeit zu verkürzen. Auch kann die Benachrichtigung einen Bericht über die während einer automatisierten Vorfallreaktion ergriffenen Maßnahmen liefern.

4.4 Konzept der kontextsensitiven, automatisierten Vorfallreaktion

In diesem Abschnitt wird nun das Kernkonzept der kontextsensitiven, automatisierten Vorfallreaktion beschrieben. Dieses wurde in einer Masterarbeit *[Heß18]* entworfen und später im Rahmen des BMBF-Projektes FlexSi-Pro weiterentwickelt und evaluiert. Dabei wurde die Implementierung stabilisiert, die Klassifikation von Assets und Restriktionen generischer gestaltet *[Pat19c]*, SDN-AIR im Anwendungskontext von Sicherheitsstatus-basiertem Netzwerkmanagement eingesetzt und die Sicherheit des Konzeptes selbst evaluiert *[Pat20]*. Die Ergebnisse dieser Arbeiten werden sowohl in diesem Kapitel, als auch in den Abschnitten 4.5 und 4.6 präsentiert und diskutiert. Den Kern des Konzepts bilden die in Abschnitt 4.3.1 vorgestellte Klassifikation von Assets und die Einschränkungen der Reaktionsmöglichkeiten, die hierfür definiert werden. Letzteres wird über restriktive Regeln umgesetzt, die im nächsten Abschnitt beispielhaft erklärt werden.

4.4.1 Restriktive Regeln

Die restriktiven Regeln dienen der Einschränkung von Reaktionen (vgl. Abschnitt 4.3.2) die für Vorfälle gewählt werden können, welche sich wiederum auf bestimmte Assets bestimmter Klassen beziehen. Dieses Konzept ist ausführlich in *[Pat19c]* und *[Pat20]* beschrieben. In diesem Abschnitt wird das Konzept jedoch noch einmal anhand eines Beispiels einer solchen Regel vorgestellt. Diese Regel wurde mit Hilfe von Prädikatenlogik definiert und enthält die folgenden Variablen und Prädikate:

- I ist die Menge der Identifikatoren, von denen jeder einen Host eindeutig identifiziert.
- $l_{x,y}$ ist eine Variable für einen Link zwischen den über $x \in I$ und $y \in I$ identifizierten Hosts ($x \neq y$).
- h_x ist eine Variable für einen Host, der über $x \in I$ identifiziert wird.
- funct_crit ist ein Prädikat, welches als funct_crit(a) = „a ist funktionskritisch“ definiert ist, wobei a ein beliebiges Asset sein kann (z.B. ein Host oder Link, vgl. Abschnitt 4.3.1).
- X ist die Menge bekannter Links.

Die folgende, beispielhafte, restriktive Regel definiert die Voraussetzung für das Erlauben der Isolation von Host h_i:

$$\neg[\text{funct\_crit}(h_i) \vee \exists l_{p,q} \in X : \text{funct\_crit}(l_{p,q})](p \neq q, i \in \{p,q\} \subset I) \quad (4.1)$$

Die Regel besagt, dass h_i nur isoliert werden darf, falls weder h_i noch ein beliebiger Link zwischen h_i und einem weiteren Knoten funktionskritisch sind. So können Restriktionen von Vorfallreaktionen formuliert werden.

4.4.2 Architektur und Methode

In den vorherigen Abschnitten wurden die relevanten Bausteine zur Definition des Kontextes und der Berücksichtigung des Kontextes bei der Reaktionsauswahl vorgestellt. Die in diesem Abschnitt vorgestellte Architektur und Anwendungsmethode definieren die noch für eine Umsetzung des SDN-AIR Konzeptes fehlenden Aspekte. Hierfür wird an das, bereits im Stand von Wissenschaft und Technik vorgestellte, SDN4S-Konzept von Koulouris et al. [Kou17] angeknüpft. SDN4S wird dabei um die Kontextdefinition und -berücksichtigung erweitert. Abbildung 4.3 zeigt einen Überblick über die Architektur von SDN-AIR. SDN-AIR fügt dem SDN4S-Ansatz im wesentlichen zusätzliche Bibliotheken (vgl. *Library*) und Datenbanken (vgl. *DB*) hinzu und ändert die Methode der Reaktionsermittlung, wofür die zentrale Komponente

von SDN4S, der Workflow-Manager, durch eine neue Komponente *Security Decision Engine* (*SDE*) ersetzt wird.

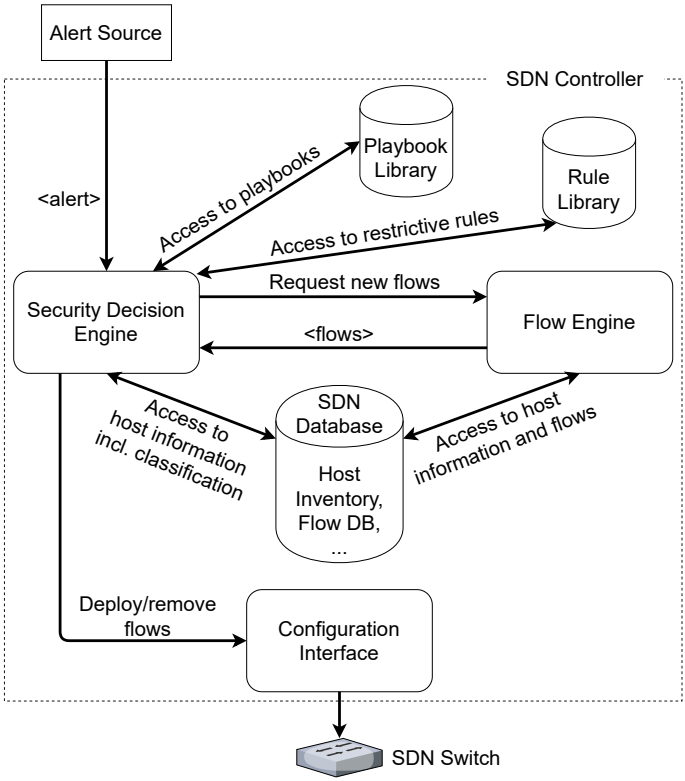


Abbildung 4.3: Architektur des SDN-AIR-Konzepts.

Die SDE erhält Vorfallmeldungen, z.B. von einem IDS. Bei Erhalt einer Meldung leitet die SDE den passenden Drehbucheintrag aus der Drehbuch-Bibliothek (vgl. *Playbook Library*) ab, die aus Einträgen mit den Zuordnungen von Meldung, Asset und Reaktion besteht. Da die hier beschriebene SDE auf deterministischen Entscheidungen beruht, geht das Konzept davon aus, dass es für

jede empfangene Meldung einen passenden Drehbucheintrag gibt.
Mithilfe der für die Meldung-Asset-Kombination vorgeschlagenen Reaktionen und des Asset-Typs (d.h. Host, Switch oder Link) holt die SDE die entsprechenden einschränkenden Regeln (vgl. Abschnitt 4.4.1) aus der Regelbibliothek (vgl. *Rule Library*). Aus den Regeln leitet die SDE ab, welche Prüfungen für welche Assets durchgeführt werden müssen, um zu entscheiden, ob eine Aktion ausgeführt werden soll oder nicht. Dazu muss die Klassifizierung der Assets jedoch erst aus den entsprechenden Inventaren, nämlich dem Host-Inventar (vgl. *Host Inventory*) oder dem Link-Inventar (vgl. *Link Inventory*), abgerufen werden. Kommt die SDE zu dem Schluss die Reaktionen durchzuführen, erzeugt sie diese entweder selbst oder mithilfe der Flow-Engine des SDN-Controllers.
Greift die SDE auf die vorhandene Flow-Engine zurück, kann sie die optimierten Algorithmen der Flow-Engine nutzen, um die besten Flows für die aktuelle Topologie zu finden, z.B. wenn Links umgeleitet werden müssen. Alternativ kann die SDE auch direkt Flow-Einträge erstellen, z.B. wenn eine bestimmte Verbindung blockiert werden muss und die Flow-Engine für diese Aufgabe nicht benötigt wird (z.B. weil keine Links umgeleitet werden müssen).
In jedem Fall werden die berechneten Flow-Einträge über die Konfigurationsschnittstelle (vgl. *Configuration Interface*, z.B. NETCONF [Enn11], OpenFlow [McK08] oder *SNMP* [Cas90]) auf die Switches angewendet oder von diesen entfernt.
Wie in *[Pat19c]* und *[Pat20]* beschrieben, wird eine Strategie der Flow-Generierung und -Bereitstellung genutzt, die das Prioritätsattribut der Flow-Einträge verwendet, um ein einfaches Zurücksetzen nach der Durchführung von Aktionen durch die SDE zu ermöglichen. Dies wird dadurch erreicht, dass die aktuellen Flow-Einträge nicht entfernt werden müssen, da sie einfach durch Sicherheits-Flow-Einträge höherer Priorität überschrieben werden. Entfernt man diese später wieder, werden die zuvor verwendeten Flow-Einträge sofort wieder aktiviert. Bei den meisten Aktionen könnte diese Strategie dem Entfernen von Flow-Einträgen vorzuziehen sein.
Darüber hinaus kann eine Implementierung des SDE-Konzepts auch bestehende Sicherheitskontrollen wie NFV-Firewalls (also Firewalls der Netzwerkmanagement-Ebene, vgl. Abschnitt 2.9) nutzen. Für Aktionen, die

durch solche Kontrollen durchgeführt werden können, ist dies ein konsistenterer Ansatz, als die direkte Anwendung von Flow-Einträgen.

4.5 Implementierung des Konzepts

Die prototypische Implementierung von SDN-AIR wurde bereits in *[Pat19c]* und *[Pat20]* vorgestellt und wird hier entsprechend zusammengefasst.
Das SDN-AIR-Konzept wurde für dessen Evaluation auf Basis von *OpenDaylight*[1] (*ODL*), einer quelloffenen SDN-Plattform, umgesetzt. Der damit implementierte Demonstrator verwendet das OpenFlow-Protokoll zur Konfiguration der SDN-Switches und enthält diverse Erweiterungen von ODL. So wurden für SDN-AIR eine SDE und die in Abschnitt 4.4 beschriebenen Inventare und Bibliotheken als Datenbanken implementiert und in ODL integriert.
Einen Überblick über die Architektur des Demonstrators bietet Abbildung 4.4. Dabei stellt das OpenFlow-Plugin die Schnittstelle zu den SDN-Switches dar (in ODL *Southbound Interface* genannt) und eine *RESTCONF*-API [Bie17] setzt die Schnittstelle zu Anwendungen, wie den hier besonders relevanten Quellen für Vorfallmeldungen, um (in ODL *Northbound Interface* genannt). Die Umgebung für die Datenbankverwaltung und die Kommunikation zwischen den Modulen von ODL wird als *Model-driven Service Abstraction Layer*[2] (*MD-SAL*) bezeichnet. Darüber wurden die Schnittstellen für die interne (Modul zu Modul) und externe Kommunikation (hier RESTCONF) definiert. Zudem wurde in MD-SAL das Datenbankschemata der Bibliotheken und der Vorfallmeldungen festgelegt. Die Netzwerkmanagementlogik auf ISO/OSI-Layer 2, welche bei SDN zentralisiert im SDN-Controller angesiedelt ist, übernimmt in ODL das L2Switch-Modul, welches somit der Flow Engine des SDN-AIR-Konzepts entspricht. Der L2Switch verwendet die von ODL aus mitgeschnittenen Netzwerkverkehr generierte Netzwerktopologie, welche in der Topologie-Datenbank (engl. *Topology Inventory*) persistiert wird, um die Ableitung geeigneter Flows umzusetzen. Die Flows werden nicht nur über OpenFlow installiert, sondern auch in der Flow-Datenbank (vgl. Flow DB aus Abbildung 4.4) gespeichert,

[1] https://www.opendaylight.org/, zuletzt zugegriffen: 19.11.2020.
[2] https://docs.opendaylight.org/projects/mdsal/en/latest/, zuletzt zugegriffen: 19.11.2020.

wodurch die aktuelle Konfiguration für das Netzwerkmanagement zugreifbar bleibt. ODL bietet zwar eine Switch-Datenbank (engl. *Switch Inventory*), Datenbanken für Hosts (engl. *Host Inventory*) und Links (engl. *Link Inventory*) wurden jedoch zusätzlich angelegt.

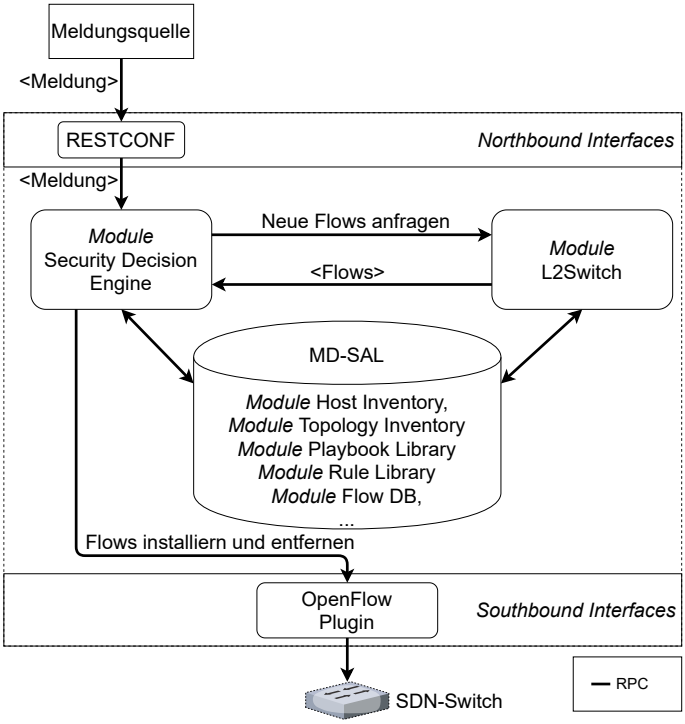


Abbildung 4.4: Implementierte Architektur des auf OpenDaylight basierenden SDN-AIR-Demonstrators.

4.6 Evaluation

Eine Lösung zur automatisierten Vorfallreaktion kann als einsetzbar in industriellen Systemen angesehen werden, wenn sie die in Abschnitt 4.2.2 definierten Anforderungen erfüllt und die Sicherheit des Systems nicht negativ beeinträchtigt. Um also einen Nachweis für die Erreichung von Ziel 4 aus Abschnitt 1.2 zu erbringen, gilt es einen Nachweis für die Erfüllung der Anforderungen aus Abschnitt 4.2.2 zu erbringen und die Sicherheit der Lösung zu untersuchen. Dies wird in den folgenden Abschnitten präsentiert.
Zudem wurde die Auswirkung der, durch das Konzept eingeführten, zusätzlichen Funktionalität untersucht, um möglichen Nachbesserungsbedarf bezüglich der Konzeptumsetzung zu identifizieren. Bevor die bereits durchgeführten Evaluationen vorgestellt und diskutiert werden, wird im folgenden Abschnitt ein Anwendungsbeispiel präsentiert, für dessen Umsetzung das Konzept und die neuste Ausbaustufe des entsprechenden Demonstrators im Rahmen des Forschungsprojektes FlexSi-Pro evaluiert wurde.

4.6.1 Anwendungsfall - Sicherheitsstatus-basiertes Netzwerkmanagement

Als Anwendungsfall für SDN-AIR dient ein im Rahmen des Forschungsprojektes FlexSi-Pro entwickeltes Sicherheitsstatus-basiertes Netzwerkmanagement. Dieses wurde bereits in *[Pat20]* beschrieben.
Das Sicherheitsstatus-basierte Netzwerkmanagement verfolgt das Ziel, als Reaktion auf Vorfallmeldungen den aktuellen Sicherheitsstatus von Hosts, Switches, Links oder ganzer Netzwerksegmente neu zu bewerten. Eine Neubewertung bewirkt eine Änderung des Verhaltens des Netzwerkmanagements und anderer Dienste, die sich auf diesen Status beziehen, gegenüber der neu bewerteten Entität. Daher ist die Anpassung des Sicherheitsstatus keine typische Reaktionsmaßnahme. Er ist vielmehr ein Kontrollmechanismus den eine SDN-AIR Lösung unterstützen sollte, um für eine solche Sicherheitslösung anwendbar zu sein.
Abbildung 4.5 zeigt ein Beispiel für diese Statusabhängigkeit. Darin ist eine SDN-Plattform dargestellt, die ein SDN-Sicherheitsmodul enthält, welches den Sicherheitsstatus eines Authentifizierungsservers, eines Robotermoduls

und einer Arbeitsstation verwaltet. Das Beispiel zeigt die Abfolge, welche ein nicht vertrauenswürdiges Gerät (z.B. eine neue Arbeitsstation) durchlaufen muss, bis es erfolgreich authentifiziert ist. Wie aus den roten Pfeilen in der Abbildung abgeleitet werden kann, darf das Gerät innerhalb der SDN-Domäne nicht frei kommunizieren und ist erst dann vertrauenswürdig, wenn es die Authentifizierung erfolgreich durchgeführt hat. Im ersten Schritt lernt das SDN-Sicherheitsmodul die Arbeitsstation kennen und setzt ihren Sicherheitsstatus von *Unbekannt* auf *Bekannt, aber nicht authentifiziert.*

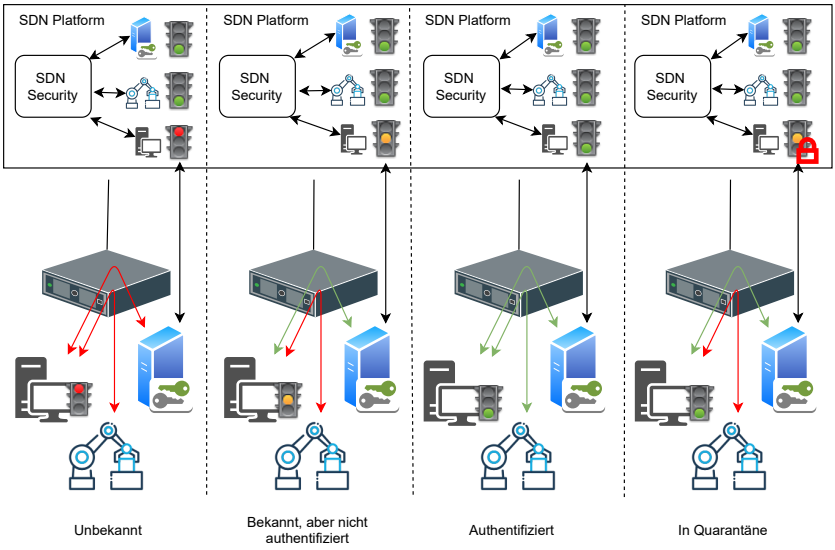


Abbildung 4.5: Vier beispielhafte Sicherheitsstatus eines Netzwerkteilnehmers (hier eine Arbeitsstation) in einem SDN-Netzwerk und deren Auswirkungen auf die erlaubte Kommunikation.

In einem zweiten Schritt führt der Authentifizierungsserver die Authentifizierung durch, nachdem er sichergestellt hat, dass der Sicherheitsstatus der

Arbeitsstation diese Aktion nicht verbietet. Wenn die Authentifizierung erfolgreich war, schlägt der Server dem SDN-Sicherheitsmodul vor, den Sicherheitsstatus der Arbeitsstation zu erhöhen. Folglich aktualisiert das SDN-Sicherheitsmodul den Status auf *Authentifiziert*. Ab diesem Zeitpunkt lassen die SDN-Switches die Arbeitsstation und den Roboter miteinander kommunizieren, da beide den Status *Authentifiziert* haben. Schließlich versetzt das SDN-Sicherheitsmodul die Arbeitsstation in Quarantäne, wenn ein Vorfall auftritt, der sich auf die Beteiligung oder sogar Kompromittierung der Arbeitsstation zurückführen lässt und aktualisiert ihren Status auf *In Quarantäne*. Dienste, wie der Authentifizierungsserver, können dann den aktuellen Status der Arbeitsstation erfragen und sicherstellen, dass sie entsprechend handeln, z.B. indem sie eine erneute Authentifizierung des Geräts anfordern oder verweigern.

4.6.2 Erfüllung von Anforderungen

SDN-AIR ist so entworfen, dass die zu erfüllenden umgebungsbezogenen Anforderungen zunächst maschinenlesbar dargestellt werden. Dabei wird folgender, von einem IT/OT-Administrator zu durchlaufender Prozess unterstützt:

1. Von SDN-AIR zu unterstützende Vorfälle werden explizit konfiguriert (Vorfallsklassen werden definiert).
2. Alle Assets werden identifiziert.
3. Alle Assets werden klassifiziert.
4. Von SDN-AIR zu unterstützende Reaktionen auf die definierten Vorfälle werden konfiguriert (Drehbücher).
5. Zu jeder Kombination aus Assettyp, Meldung (bzw. Meldungskategorie) und Reaktion wird eine Restriktionsregel definiert (es sind auch leere Restriktionsregeln erlaubt; wichtig ist jedoch, dass keine Kombination aus Assettyp, Meldung und Reaktion unbeachtet bleibt).

Dieser Prozess wird im Folgenden angenommen, um Konfigurationslücken auszuschließen.

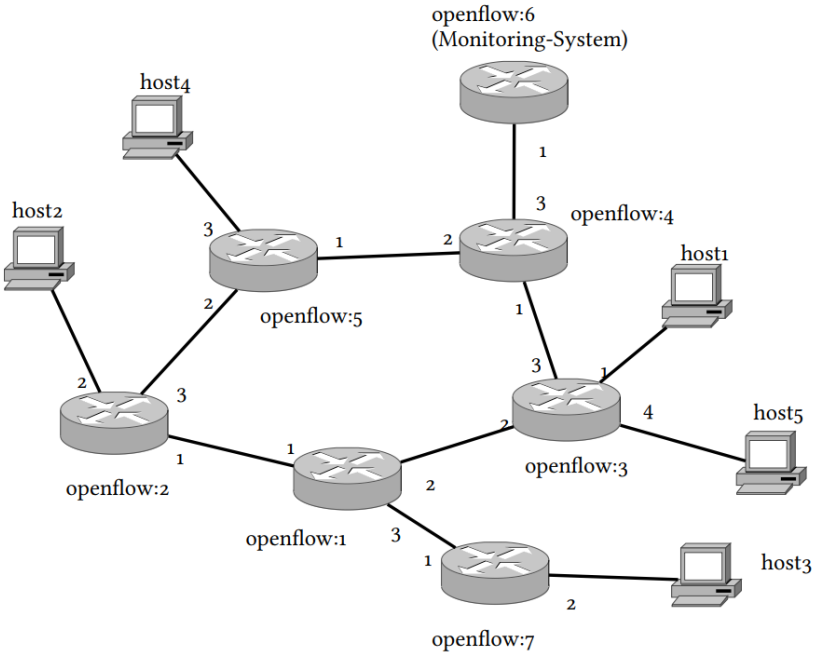


Abbildung 4.6: Beispieltopologie aus SDN-AIR Evaluation mit 6 OpenFlow-unterstützenden Switches, 5 Hosts und einem, als Switch emulierten Monitoring-System.

Der Prozess führt zu einer Transformation der umgebungsbezogenen Anforderungen in eine Konfiguration und Verarbeitungsvorlage für die SDN-AIR-Instanz. Unter der Annahme, dass dieser Prozess befolgt wird, stellt sich also die Frage, ob eine deterministische SDN-AIR-Instanz die konzipierte Restriktionsstrategie umsetzen kann. Denn dies würde unter der weiteren Annahme, dass die Lösung keine zusätzlichen Sicherheitsprobleme einführt, bedeuten, dass somit SDN-basierte, automatisierte Vorfallreaktion um die Konfigurierbarkeit genau der Aspekte erweitert werden kann, die im Widerspruch zur Einsetzbarkeit in industriellen Systemen stehen. Demnach würde der Nachweis der Umsetzbarkeit von SDN-AIR (insbesondere mit existierenden SDN-Lösungen) die Hypothese 6 belegen. Solche Nachweise der Umsetzbarkeit wurden bereits

in mehreren Arbeiten für verschiedene Versionen des SDN-AIR-Demonstrators erbracht. Sie werden in diesem Abschnitt diskutiert. Die Untersuchung zu den, durch das Konzept eingebrachten, Sicherheitsproblemen wird in Abschnitt 4.6.4 durchgeführt.
Die ersten Umsetzbarkeitsnachweise wurden bereits in *[Heß18]* erbracht. Um eine hinreichende Komplexität der SDN-Topologie und eine flexible Laborumgebung zu erzeugen, in der die Auswirkungen der Vorfallreaktion messbar sind, wurden hierfür verschiedene Netzwerktopologien mit dem Netzwerkemulator Mininet[1] aufgebaut, auf denen die Umsetzung getestet und evaluiert wurde. Eine dieser Topologien, ist in Abbildung 4.6 dargestellt. Die Topologie besteht aus 7 OpenFlow-unterstützenden, virtuellen SDN-Switches (Open vSwitches[2]) und 5 virtuellen Mininet-Hosts, wobei einer der Switches ein Monitoring-System emuliert.
Diese Topologie wurde in Mininet 2.2.0 geladen und mit der SDN-AIR-Instanz verknüpft. Mit ihr wurden die folgenden Vorfallreaktionen untersucht:

- **Isolation eines Hosts**, indem der mit dem Host direkt verbundene Switch (identifizierbar über das Topologie-Inventar) mit zwei Flow-Einträgen konfiguriert wird. Soll bspw. `host2` isoliert werden, wird `openflow:2` mit zwei Flows belegt, die eine Übereinstimmungsregel für die Ethernet-Adresse von `host2` besitzen (einmal als Quelle und einmal als Ziel) und als Instruktion eine sogenannte `drop-action` definieren. Um einmal einen solchen Flow zu zeigen, wird in Listing 4.1 ein Ausschnitt des entsprechenden Isolations-Flows dargestellt, der ausgehenden Netzwerkverkehr von `host2` verwirft.
- **Isolation eines Switches**, indem alle benachbarten Switches entsprechende Umleitungs-Flows erhalten, sofern dies möglich ist. Dies wird mithilfe eines Tricks umgesetzt. Dafür wird dem L2Switch eine Version der aktuellen Topologie ohne den betreffenden Switch übergeben, welcher daraufhin die zu implementierenden Flows berechnet.

[1] http://mininet.org/, zuletzt zugegriffen: 13.10.2020.
[2] https://www.openvswitch.org/, zuletzt zugegriffen: 13.10.2020.

- **Überwachen eines Endgeräts**, indem die Flows so angepasst werden, dass die Pakete zum Switch, an dem das Monitoring System angeschlossen ist (hier `openflow:2`), umgeleitet und von dort zum eigentlichen Empfänger weitergeleitet werden. Diese Strategie führte allerdings zu unnötigen Einschränkungen der Reaktionsumsetzbarkeit durch die potenziell ungünstige Umleitung, da SDN in der Lage ist, Pakete mithilfe der Definition von zwei `output-action`-Anweisungen innerhalb eines Flows zu duplizieren und so keine Umleitungen eingerichtet werden müssen, die potenziell negative Auswirkungen auf Qualitätsgarantien haben. Die Replikate werden jedoch in jedem Fall in VLANs zu dem Monitoring-System geleitet, um sie vom Originalverkehr unterscheiden zu können und Schleifen zu vermeiden *[Heß18]*.
- **Überwachen von Switches**, was in der Ausführung im Wesentlichen der Überwachung von Endpunkten entspricht, nur dass die Replikation der Pakete in allen direkt benachbarten Switches stattfindet.

Die Beschränkung dieser Reaktionen wurde anhand verschiedener Netzwerktopologien, Asset-Klassifikationen und Vorfallreaktionen getestet. Ein Teil dieser Tests wurde auch als Unit-Tests und Testskripte (für die Konfiguration über die RESTCONF-Schnittstelle) in das Software-Projekt übernommen. Einen Ausschnitt der Tests ist in Listing A.3 zu finden. Geprüft wird dabei, ob die Kernkomponente der Restriktionseinschränkung (der *FeasibilityAnalyser*) die Reaktion erlaubt oder verbietet. Zugrunde gelegt wird die Topologie aus Abbildung 4.6, wobei für die Testreihe anfangs spezifische Flows konfiguriert werden.
Außer der reinen Umsetzbarkeit wurden auch Auswirkungen auf das Netzwerk gemessen. Alle Messungen wurden auf einer virtuellen Maschine mit Ubuntu 14 LTS mit zwei Prozessorkernen und 1024 MB Arbeitsspeicher durchgeführt. Die Maschine wurde mit VirtualBox 5.2 ohne Gasterweiterung virtualisiert. Als Hostsystem wurde ein Macbook Pro mit einem 2.0 GHz getakteten Intel i7-4750HQ-Prozessor, einem 8 GB Arbeitsspeicher und einer 256GB großen SSD verwendet. Als Betriebssystem wurde macOS 10.13.6 verwendet. Zudem wurde

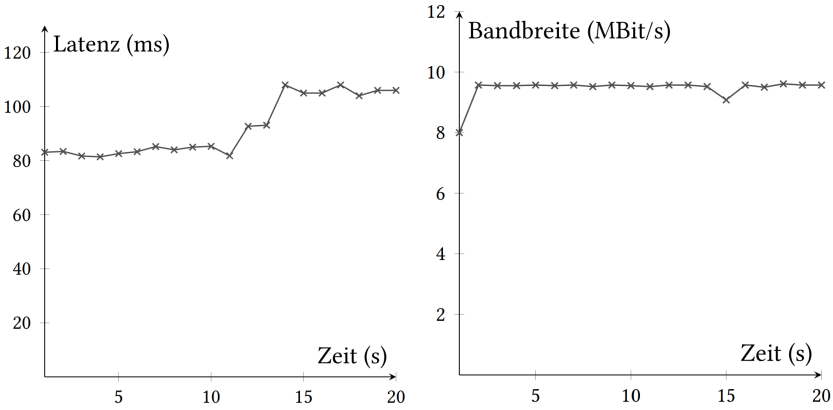
für die Messungen die emulierten Netzwerkabschnitte mit einer maximalen Bandbreite von 10MBit/s und einer festen Verzögerung von 10ms konfiguriert. Damit wurden die im Folgenden zusammengefassten Untersuchungen durchgeführt.

```
id: sde-drop-out-host2,
priority: 100,
match: {
  ethernet-match: { ethernet-source: { address
      :00:00:00:00:00:02 }}
},
instructions: { instruction: [{ order:0, apply-actions: {
    action: [{
        order: 0,
        drop-action
      }]}}]}
```

Listing 4.1: Ausschnitt aus dem Isolations-Flow, der ausgehenden Netzwerkverkehr von `host2` verwirft. Alle Flows werden mit der höchsten Flow-Priorität versehen (hier 100), sodass alte Flows nicht überschrieben werden müssen und gleichzeitig die SDE-Flows alle anderen überdecken.

Die zuvor beschriebenen Reaktionen wurden ausgelöst und, da die Auswirkungen auf das Netzwerk getestet werden sollten, nicht durch Restriktionen verboten. Eine Ausnahme stellt die Isolation von Endgeräten dar, die logischerweise zum Abbruch der Kommunikation zwischen zwei Hosts führt und somit keine sinnvollen Messergebnisse liefern kann. Während der Reaktionsausführungen wurden Kommandos durchgeführt, deren Ausgaben auf die Latenz und die aktuell zur Verfügung stehende Bandbreite schließen lassen. Für ersteres wurde ein Ping von einem Host zu einem weiteren ausgeführt, deren Pfad zueinander von der jeweiligen Reaktion betroffen war. Wegen der festen Verzögerung von 10ms ist aus der benötigten Zeit eines Pings zu ermitteln, wie viele Verbindungsabschnitte dieser zurückgelegt hat. Dieser Wert lässt sich plotten und eignet sich, auch wenn die Verzögerung der Abschnitte konstant ist, gut zur Betrachtung der Auswirkung der Reaktionen auf die Latenz.

Für die aktuelle Bandbreite wurde das von Mininet unterstützte Messwerkzeug *iPerf*[1] eingesetzt. Dieses Werkzeug versucht so viele Daten wie möglich von einem iPerf-Client zu einem iPerf-Server zu senden, die von Mininet auf den zwei betroffenen Hosts installiert wurden. Die Ausgabe von iPerf zeigt nicht nur die Gesamtgröße der erfolgreich übermittelten Daten, sondern auch die ausgenutzte Bandbreite. Diese lässt sich ebenfalls plotten, um die Auswirkungen der Reaktion auf die nutzbare Bandbreite zwischen den Hosts zu visualisieren.



(a) Latenz vor, während und nach der Isolation von `openflow:1`. *Quelle: [Heß18].*

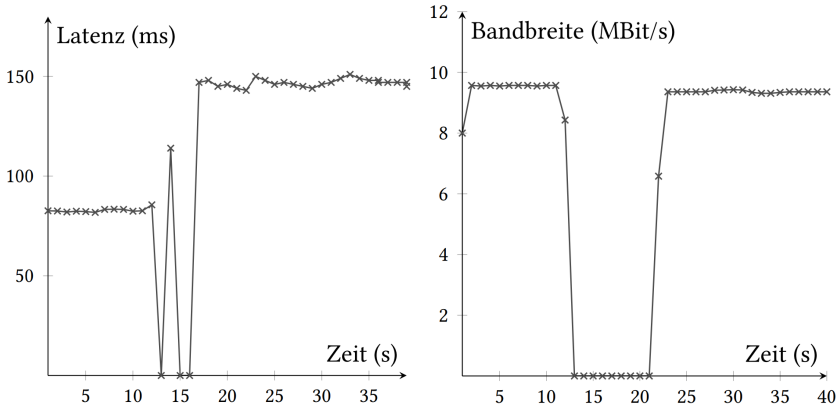
(b) Bandbreite vor, während und nach der Isolation von `openflow:1`. *Quelle: [Heß18].*

Abbildung 4.7: Messungen vor, während und nach der Isolation von `openflow:1`.

Die Auswirkungen auf die Kommunikation zwischen `host1` und `host2` durch die Isolation von `openflow:1`, welche nach etwa 12 Sekunden gestartet wurde, sind in den Abbildungen 4.7 a und b dargestellt. Sie zeigen, dass die, durch die Isolation entstehenden, schlechteren Pfade eine höhere Latenz bewirken. Durch die Umstellung der Flows wird zudem kurzzeitig die Bandbreite in

[1] https://iperf.fr, zuletzt zugegriffen: 01.02.2021.

Sekunde 15 beeinträchtigt, stabilisiert sich jedoch innerhalb von Sekunde 16 wieder. Zudem lässt sich daraus ableiten, dass bei der Verwendung von SDN-AIR von der Auslösung der Switch-Isolation bis zur Stabilisierung des Netzes in etwa 3 Sekunden vergehen, wobei dies von den zu überprüfenden Restriktionen (hier vernachlässigbar wenige) und den eingesetzten Ressourcen abhängt. Zur Überwachung von `host1` wurde der Datenverkehr zwischen `host1` und `host2` über das Monitoring-System umgeleitet.



(a) Latenz vor, während und nach der Umstellung auf die Überwachung von `host1`. *Quelle: [Heß18].*

(b) Bandbreite vor, während und nach der Umstellung auf die Überwachung von `host1`. *Quelle: [Heß18].*

Abbildung 4.8: Messungen vor, während und nach der Umstellung auf die Überwachung von `host1`.

Abbildungen 4.8 a und b zeigen die entsprechenden Latenz- und Bandbreitenmessungen vor, während und nach der Umstellung auf die Überwachung, welche erneut etwa in Sekunde 12 ausgelöst wurde. Beide Messungen zeigen einen wesentlich deutlicheren Einbruch als bei der Isolation von `openflow:1`. Der Einbruch der Latenz ist hierbei durch einen vorübergehenden Paketverlust zu erklären und der Berechnung aus der Ping-Ausgabe geschuldet. Zudem ist eine deutlich längere Dauer bis zur Erholung der Kommunikation zwischen

host1 und host2 erkennbar. Die längeren Erholungszeiten sind vermutlich darauf zurückzuführen, dass bei der Reaktion nicht, wie zuvor, neue Flows hinzukommen, welche die Pakete nur einen anderen Weg nehmen lassen, sondern Flows, welche ein Paket mit einem VLAN-Tag versehen. Weiß ein anderer Switch zum Zeitpunkt des ankommenden getaggten Paketes noch nicht, wie dieses zu handhaben ist, weil die entsprechenden Flows noch nicht installiert sind, wird das Paket verworfen und die Verbindung bricht ein. Bei der Umstellung auf die Überwachung von openflow:1 wurde außer des iPerf-Verkehrs von host1 zu host2 weiterer iPerf-Verkehr von host4 zu host5 hinzugefügt. Die Verbindungen teilen sich keine Netzwerkabschnitte.

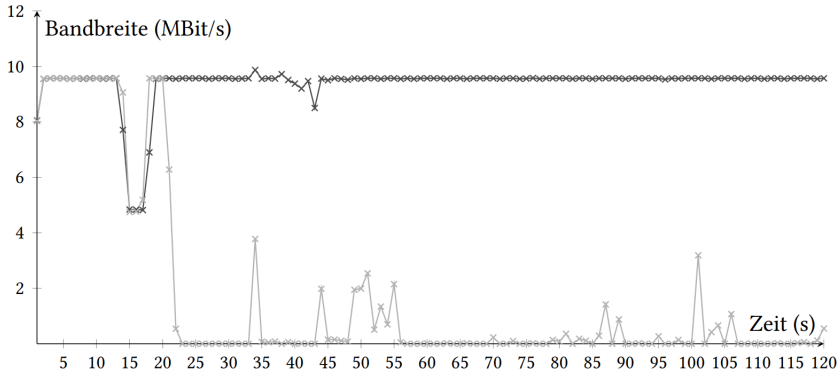


Abbildung 4.9: Vergleich der Bandbreite von host1-host2- (schwarz) und host4-host5-Kommunikation (grau) vor, während und nach der Umstellung auf die Überwachung von openflow:1.
Quelle: [Heß18].

Abbildung 4.9 zeigt den Vergleich der zur Verfügung stehenden Bandbreiten dieser Verbindungen. Dabei ist zu erkennen, dass diese anfangs beide die volle Bandbreite nutzen, jedoch nach der Umstellung in Sekunde 12 nur noch die Verbindung von host1 zu host2 die maximale Bandbreite nutzt. Dass beide Verbindungen nach der Umstellung konkurrieren, ist durch die

nun geteilten Netzwerkabschnitte zu begründen. Warum die `host1`-zu-`host2`-Kommunikation priorisiert wird, ist jedoch nicht klar. Ein möglicher Grund könnte sein, dass die Open vSwitches generell VLAN-Verkehr bevorzugen. Da diesbezüglich jedoch keine öffentlich verfügbaren Informationen zur Verfügung standen und das Nachvollziehen der Implementierung Open vSwitch nicht möglich war, konnte die Annahme nicht verifiziert werden. Die Messungen zeigen, warum die Konfiguration der Restriktionen so wichtig ist. Wo Einbrüche, wie die durch die Überwachungsreaktion ausgelösten, nicht akzeptiert werden können, müssen Restriktionen diese verhindern. Zudem zeigen die Messungen, dass die Überwachung von Geräten zwar keine dauerhaften negativen Auswirkungen haben, solange die Netzwerkkapazitäten hinreichend dimensioniert sind, jedoch durchaus zu kurzweiligen Netzwerkproblemen führen. Daher besitzt diese Reaktion einen negativen Aspekt, der bei der Konzeptionierung noch nicht bekannt war. Solche Effekte müssen weiter untersucht werden, um passende Restriktionskonfigurationen zu finden (vgl. Abschnitt 5.2).
Diese Evaluation zeigte zwar bereits die Umsetzbarkeit von SDN-AIR, enthielt jedoch noch keinen Bezug zum Sicherheitsmanagement moderner industrieller Systeme. Daher wird im Folgenden die Umsetzung des in Abschnitt 4.6.1 beschriebenen Anwendungsfalls präsentiert, welche im Rahmen des Forschungsprojektes FlexSi-Pro durchgeführt und als Teil der Abschlussevaluation von Projektträger und Partnern abgenommen wurde.

4.6.3 Sicherheitsstatusmanagement

Die Idee, den Anwendungsfall Sicherheitsstatus-basierten Netzwerkmanagements mithilfe von SDN-AIR umzusetzen, liegt nahe, da hierfür sowohl ein zentralisiertes Netzwerkmanagement in Form von SDN-Controller und -Plattform als auch ein automatisierter Sicherheitsmechanismus benötigt wird, welcher das Netzwerk rekonfigurieren kann. Die Umsetzung der auf SDN-AIR basierenden Netzwerkmanagement-Strategie ist der Kern der FlexSi-Pro Sicherheitsarchitektur, die in dem an den Abschlussbericht *[Pat19a]* angehängten *„Rahmenwerk für flexible, sichere und zeitsensitive Produktionsanlagen“* beschrieben wurde, welches vom gesamten FlexSi-Pro-Konsortium erarbeitet wurde.

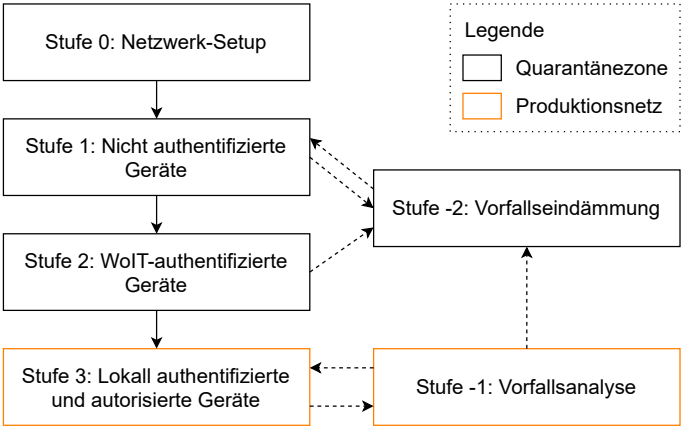


Abbildung 4.10: Stufenkonzept des Sicherheitsstatus-basierten Netzwerkmanagements. *Quelle: [Pat19a].*

Die in Zusammenarbeit mit der WIBU SYSTEMS AG[1] ausgearbeitete Architektur basiert auf einem Sicherheitsstatus-abhängigen Stufensystem, für das ein Beispiel in Abbildung 4.10 zu sehen ist. Die Stufen decken die technisch steuerbaren Sicherheitsaufgaben der im Rahmenwerk definierten Netzwerk-Lebenszyklus-Phasen *Vorbereitung*, *Registrierung*, *Laufzeit* und *Entfernen* ab. Die Stufen spiegeln den Grad des Vertrauens in die Netzwerkteilnehmer wider. Dabei ist die Anzahl der Stufen variabel. Für dieses Rahmenwerk wurde eine vierstufige Lösung mit zwei Rückstufungen getestet. Um Vertrauen zu erhalten, sprich in eine höhere Stufe aufzusteigen oder die aktuelle Stufe zu halten, müssen sich die Netzwerkteilnehmer in den Stufen 1-3 Sicherheitsüberprüfungen unterziehen. Die Pfeile in Abbildung 4.10 zeigen die konfigurierten Stufenübergänge, wobei die durchgezogenen Pfeile den Prozess des Vertrauensaufbaus symbolisieren. Beide Prozesse werden hierbei von der SDN-AIR-Instanz umgesetzt. Das erhaltene Vertrauen kann allerdings jederzeit wieder reduziert werden, falls eine Überprüfung fehlschlägt. Dies wird durch die gestrichelten

[1] https://www.wibu.com, zuletzt zugegriffen: 23.11.2020.

Pfeile dargestellt, welche somit den Prozess der Vorfallreaktion bilden. Der Vorgang des Vertrauensentzugs wird dabei als Rückstufung bezeichnet. Je höher die Stufe ist, in der sich ein Netzwerkteilnehmer befindet, desto größer ist das ihm entgegengebrachte Vertrauen und damit steigt auch dessen Berechtigung im Netzwerk. Demnach hat beispielsweise ein Netzwerkteilnehmer in Stufe 0 nur die nötigsten Berechtigungen, um die Netzwerkeinrichtung durchführen zu können. Dahingegen hat ein Netzwerkteilnehmer in Stufe 3 alle nötigen Berechtigungen, um seine Rolle im Netzwerk erfüllen zu können. Somit befinden sich Netzwerkteilnehmer, die sich in Stufe 3 befinden, im regulären, nicht von der SDN-AIR-Instanz eingeschränkten Produktionsnetz.

Request URL

```
http://localhost:8181/restconf/operational/hostinventory:hosts
```

Response Body


```
{
  "hosts": {
    "host": [
      {
        "id": "host3",
        "securitystateid": 0,
        "ip": "10.0.0.3",
        "mac": "00:00:00:00:00:03",
        "node-connector": "openflow:4:3"
      },
      {
        "id": "host2",
        "securitystateid": 0,
        "ip": "10.0.0.2",
        "mac": "00:00:00:00:00:02",
        "node-connector": "openflow:2:2"
      },
      {
        "id": "host1",
```


Abbildung 4.11: Hosts mit initialem Status 0, der die Stufe 0 suggeriert.

Eine ausführliche Beschreibung des Konzeptes und dessen prototypische Implementierung wird in dem Rahmenwerk-Dokument *[Pat19a]* präsentiert.

Hier wird diese kurz zusammengefasst, um einen besseren Einblick in den Einsatz von SDN-AIR im Anwendungsfall zu bieten. Für die praktische Umsetzung wurde der SDN-AIR-Demonstrator aus Abschnitt 4.5 um den Eintrag der Sicherheitsstufe als Host-Variable (vgl. Abbildung 4.11) und die nötige Konfigurationsmöglichkeit der Stufen erweitert.
Wie in Code-Ausschnitt von Listing 4.2 zu sehen ist, wurde Stufe 1 für den Aufruf einer *WoIT*-Instanz konfiguriert (vgl. HTTP-Aufruf von `woit:authenticate_device`). WoIT steht hierbei für *Web of Industrial Trust* und ist ein, ebenfalls in FlexSi-Pro erarbeiteter, Sicherheitsmechanismus zur Authentifizierung von Endgeräten gegenüber einer Authentifizierungseinheit, die WoIT-Instanz genannt wird. Für weitere Details werden interessierte Lesende auf *[Pat19a]* verwiesen. Wichtig für die Evaluation des Konzeptes ist hierbei, dass die WoIT-Instanz von der SDN-AIR-Instanz aufgerufen wird. Nach durchgeführter Authentifizierung des Endgeräts sendet die WoIT-Instanz der SDN-AIR-Instanz entweder ein Erfolgs- oder ein Misserfolgssignal. Bei Erfolg erhöht die SDN-AIR-Instanz den Sicherheitsstatus des Gerätes, da keine weiteren Mechanismen registriert sind und überführt es somit in die nächste Stufe. Bei Misserfolg wird der Sicherheitsstatus auf die ID für „Incident-Containment“ gesetzt, von der das Gerät erst durch manuelles Überführen in Stufe 1 (im Beispiel durch Setzen des Status auf 1) zurückgelangt. Zudem zeigt der vollständige Konfigurations-Code in Listing A.4 die Konfiguration der Stufe 2, in der die CVE-basierte Sicherheitsanalyse aus Abschnitt 3.7.4 für das Gerät durchgeführt wird. Dafür wird nicht direkt die SyMP-Instanz aufgerufen, sondern ein dedizierter Agent, der den Aufruf in Einzelaufrufe der Analyseschritte aus Abschnitt 3.7.4 umwandelt. Dieser war zu diesem Zeitpunkt noch notwendig, da SyMP noch nicht weit genug entwickelt war und unter anderem die Analyse-Engine und auch das Workflow-Konzept fehlten (wie in Abschnitt 3.7.4 zu sehen ist, kann hierfür nun SyMP verwendet werden). Liefert die CVE-Analyse gefundene Schwachstellen, wird das Gerät auch hier wieder in die „Incident-Containment“-Stufe gesetzt und kann von dort wieder nur manuell in Stufe 1 zurückgesetzt werden.
Durch das REST-Konzept kann sich die SDN-AIR-Instanz auch selbst bei Stufenübergang aufrufen. So kann eine beliebige Vorfallmeldung bei Betreten der Stufen -1 und -2 (in der Konfiguration 4 und 5) gewählt werden.

```
curl -H 'Content-Type: application/json' -X POST -d '
{
  "securityproperties:input": {
    "securityproperties:name": "Unauthenticated",
    "securityproperties:successor-states": [
    "2", "5"
    ],
    "securityproperties:RESTcontentType": "IP",
    "securityproperties:RESTcredentials": "admin:admin",
    "securityproperties:RESTdestination": "http://localhost
        :6213/restconf/operations/woit:authenticate_device"
  }
}' -u $CREDENTIALS 'http://localhost:8181/restconf/operations
    /securityproperties:add-state'

curl -H 'Content-Type: application/json' -X POST -d '
{
  "securityproperties:input": {
    "securityproperties:name": "Authenticated",
    "securityproperties:successor-states": [
    "3", "5"
    ],
    "securityproperties:RESTcontentType": "IP",
    "securityproperties:RESTcredentials": "admin:admin",
    "securityproperties:RESTdestination": "http://localhost
        :8889/cve"
  }
}' -u $CREDENTIALS 'http://localhost:8181/restconf/operations
    /securityproperties:add-state'
```

Listing 4.2: Konfiguration der 1. und 2. Sicherheitsstufe für den FlexSi-Pro Demonstrator (vollständige Konfiguration befindet sich in Anhang A.4).

Für Stufe -1 wird also eine niederpriore Host-Kompromittierung gemeldet und für Stufe -2 eine hochpriore. Damit wird in Stufe -1 die Überwachung und

in Stufe -2 die Isolierung des Hosts erreicht. Wichtig ist jedoch, dass hierbei immer die Restriktionen des Hosts beachtet werden und es somit passiert, dass der Host bei Übergang von Stufe 1 zu Stufe -2 nicht isoliert wird. Dies ist ein Implementierungsnachteil, der für die Einfachheit des Demonstrators in Kauf genommen wurde. In Produktivimplementierungen müsste an dieser Stelle eine Fallunterscheidung stattfinden und die Konfiguration entsprechend erweitert werden.
Mit der Umsetzung des FlexSi-Pro-Demonstrators konnte der Einsatz von SDN-AIR in einem modernen Gesamtkonzept für flexible und sichere Produktionsanlagen gezeigt werden.

4.6.4 Sicherheitsuntersuchung

Basierend auf dem Angreifermodell aus Abschnitt 4.2.1, wurde bereits in *[Pat20]* eine Sicherheitsuntersuchung von SDN-AIR im hier beschriebenen Anwendungsfall durchgeführt. Diese wird im Folgenden zusammengefasst.
Die Untersuchung bezieht sich auf die durch SDN-AIR eingeführten Aspekte, nicht aber auf die allgemeine Sicherheit von SDN. In *[Pat20]* wurden diesbezüglich die folgenden Angriffsvektoren untersucht:

1. Ändern oder Unterdrücken von an die SDE gesendeten Meldungen.
2. Missbrauch des automatisierten Flow-Deployments.
3. Ändern von Flow-Anforderungen oder Flow-Engine-Antworten.
4. Manipulieren von Benachrichtigungen und Sicherheitsstatusinformationen.
5. Kompromittieren von Drehbuch- und Regelbibliotheken.

Unter Verwendung des in Abschnitt 4.2.1 vorgestellten Angreifermodells wurden dabei folgende zugehörige Schlussfolgerungen abgeleitet:

1. Die Manipulation von Alarmen muss durch Nachrichtenauthentizität und -integrität verhindert werden. Zudem muss sichergestellt werden, dass unterdrückte Nachrichten bei der SDE erkannt werden. Ein Beispielmechanismus ist ein regelmäßiges, Authentizität- und

Integrität-geschütztes Reporting von Sendestatistiken durch die Meldung-generierenden Instanzen an die SDE.

2. Ein Angreifer, der bösartigen Netzwerkverkehr verursacht, um die SDE Änderungen der Netzwerktopologie vornehmen zu lassen, erreicht damit nicht mehr, als bei einem direkt an den Zieldienst gerichteten Angriff. Insbesondere, weil eine korrekte Konfiguration der SDN-AIR-Lösung keine kritischen Änderungen auslöst. Die Alternative, eine Vorfalldetektionsinstanz zu kompromittieren, unterläge denselben Beschränkungen im Bezug auf die Ausnutzung von SDN-AIR.
3. Sollten SDE und Flow-Engine über eine Netzwerkschnittstelle kommunizieren und nicht, wie im hier vorgestellten Demonstrator, in einer gemeinsamen Komponente laufen, müsste auch dieser Kommunikationskanal Nachrichtenauthentizität und -integrität sicherstellen.
4. Auch die Schnittstellen zur Statusverwaltung und Kommunikation mit Incident-Respondern müssen zur Garantie der Nachrichtenauthentizität und -integrität entsprechende Maßnahmen implementieren.
5. Besonderen Schutz müssen auch die Drehbuch- und Regelbibliothek genießen. Dies beinhaltet sichere Nutzerauthentifikation und -autorisierung, sowie die Sicherstellung von Nachrichtenauthentizität, -integrität und -vertraulichkeit.

Durch die Analyse in *[Pat20]* konnte abgeleitet werden, dass SDN-AIR die Angriffsfläche auf das System nicht vergrößert. Insbesondere schützt es konstruktionsbedingt bereits kritische Dienste, Kommunikationswege und Endgeräte, wodurch deren negative Beeinträchtigung durch SDN-AIR bei korrekter Konfiguration ausgeschlossen wird.

4.6.5 Bewertung

Die Evaluation zeigt, dass SDN-AIR ein sicheres Konzept zur automatisierten Vorfallreaktion bietet, welches für verschiedene Sicherheitskonzepte eingesetzt

werden kann und dabei konstruktionsbedingt kritische Dienste, Kommunikationswege und Endgeräte schützt. SDN-AIR lässt damit also zu, Betriebssicherheit, Ausfallsicherheit, Echtzeitanforderungen und Determinismus für Netzwerkteilnehmer, Netzwerkkomponenten und Verbindungen zu wahren. Demnach kann abgeleitet werden, dass SDN-AIR den Nachweis für Hypothese 6 erbringt und Ziel 4 aus Abschnitt 1.2 erreicht wurde. Diese Aussagen stützen sich auf die korrekte Konfiguration von SDN-AIR. Um eine solche Konfiguration erreichen zu können, ist eine klare Identifikation und Festlegung von schützenswerten Diensten, Verbindungen und weiteren Assets notwendig, die in der Praxis oft mangelhaft ist. Auch wird SDN in der Praxis zum Zeitpunkt des Verfassens dieser Dissertation noch kaum für industrielle Systeme eingesetzt, wobei Entwicklungen mit Bezug auf Asset-Management und Netzwerkzentralisierung, bzw. -virtualisierung bereits fortschreitende Verbesserung und Modernisierung beider Aspekte bewirken.

5 Ausblick und Zusammenfassung

In diesem Kapitel wird zunächst die Dissertation zusammengefasst. Abschließend wird erörtert, welche weiterführenden Forschungsarbeiten die zuvor vorgestellten Lösungen ermöglichen und welche Lücken durch weitere Arbeiten gefüllt werden können.

5.1 Zusammenfassung

In dieser Dissertation wurden Forschungsbedarfe der automatisierten minimalinvasiven Sicherheitsanalyse und Vorfallreaktion adressiert. Durch den Fokus auf industrielle Systeme standen dabei u.a. die problematische Heterogenität und Komplexität der Endgeräte und Netzwerke, sowie die Wahrung von Garantien, wie Verfügbarkeit, Echtzeitverarbeitung und Redundanz im Vordergrund. In diesem Rahmen präsentierte die Arbeit Beiträge zur Erweiterung der Abdeckung sicherheitsrelevanter Informationen, der flexiblen und wiederverwendbaren Modellverarbeitung und -analyse, sowie der deterministischen, garantieerhaltenden SDN-basierten Vorfallreaktion.
Für die Sicherheitsanalyse wurden verschiedene eigene Forschungsergebnissen und ein daraus abgeleitetes, umfassendes Rahmenwerk präsentiert, das eine neuartige Vereinbarkeit von Automatisierung, Flexibilität und Wiederverwendbarkeit von modellbasierten Analyseverfahren ermöglicht. Außerdem unterstützt das Rahmenwerk als erste Lösung die gleichzeitige Anwendbarkeit der wichtigsten Analysearten eines typischen Sicherheitslebenszyklus industrieller Systeme, die Ausnutzung von Synergien mehrerer Analysen und verschiedene Akteure. Die Umsetzbarkeit dieses Rahmenwerks wurde nachgewiesen und umfassend evaluiert. Diese Evaluation zeigte, dass mit der Konzeptionierung des Rahmenwerks ein wichtiger Schritt in Richtung praxistauglicher minimalinvasiver Sicherheitsanalyse für industrielle Systeme gemacht werden

konnte. Damit und durch einen Vergleich mit existierenden Lösungen konnte der Beitrag zum Fortschritt von Wissenschaft und Technik bestätigt werden. Das im Rahmen der Dissertation entwickelte Konzept zur SDN-basierten, minimalinvasiven Vorfallreaktion zeigt die Einsetzbarkeit von automatisierter Vorfallreaktion für industrielle Systeme und unterstützt selbst umfangreiche Eingriffe in das entsprechende Netzwerk, sofern hinreichend maschinenlesbares Wissen über dessen Architektur und die sicherheitsbezogene Klassifikation von Assets verfügbar ist. Für das Konzept wurde ein deterministischer, Drehbuch-basierter Reaktionsansatz gewählt, der durch ebenfalls deterministische, restriktive Regeln die Einhaltung von Garantien unterstützt. Die Evaluation dieses Ansatzes zeigte seine Umsetzbarkeit und, durch den Einsatz in eine Sicherheitsmanagementlösung für industrielle Netzwerke, die Anwendbarkeit für industrielle Systeme. Die Evaluation machte dabei auch den Bedarf deutlich, die Wahl angemessener Drehbucheinträge und geeigneter restriktiver Regeln weiter zu erforschen.
Folglich ist festzuhalten, dass sowohl die Beiträge zur Sicherheitsanalyse als auch zur Vorfallreaktion den Stand von Wissenschaft und Technik erweitern und gleichzeitig eine Grundlage für weitere, relevante Forschungsarbeiten auf dem Gebiet der industriellen Sicherheit bieten.

5.2 Ausblick

Basierend auf den hier vorgestellten Konzepten können weitere Verbesserungen ihrer Umsetzungen vorgenommen werden, die in den Abschnitten 3.7.10 und 4.6.5 beschrieben wurden. Darüber hinaus wurden durch die Konzepte weitere Entwicklungsschritte ermöglicht, die hier nun als Ausblick für weiterführende Forschung vorgestellt werden.
SyMP schafft erstmals eine Grundlage für die Trennung von Expertise in der automatisierten, ontologiebasierten Sicherheitsanalyse. Diese Trennung kann weitergeführt werden, indem Vorschlagsysteme (bzw. Recommender-Systeme) eingesetzt werden, um die manuell zu treffenden Entscheidungen zu automatisieren. Durch das bereits vorhandene formalisierte Wissen kann so beispielsweise eine Lösung entwickelt werden, bei der ein Endnutzer lediglich angeben muss, welchen Standards nach sein System konform sein

muss. Die zugehörigen Policies und Umsetzungen können dabei intelligent, unter Berücksichtigung der zugrundeliegenden Komponenteninformationen, gewählt werden. Hierbei können bereits weitere Optimierungsschritte, wie die optimale Synergie zwischen den Umsetzungen (wie sie in Abschnitt 3.7 evaluiert wurde), einbezogen werden.
Auch die (teil-)automatische Erzeugung von Umsetzungen kann ermöglicht werden, sodass Modellierungsexperten (bzw. SyMP-Power-User) weniger Security- und System-Domänenwissen besitzen müssen. Ein möglicher Ansatz ist dabei die automatisierte Umwandlung von natürlichsprachlichen technischen Richtlinien in eine formale Zwischensprache (z.B. ähnlich zu OrBAC), für die eine erweiterbare Abbildung auf die Analyseregeln der SyMP-Hub-Wissensbasis existiert (vgl. Abschnitt 3.4.3.1). Ein erster Schritt in diese Richtung wurde bereits durch eine Masterarbeit *[Har20]* umgesetzt, indem eine Modularisierung von SPARQL-Abfragen entwickelt wurde, die aufgrund ihrer Granularität für solche Zwischensprachen geeignet ist.
Beide Punkte sollten weiter untersucht werden und werden nach den Ergebnissen dieser Dissertation vom Autor als vielversprechende Erweiterungen von SyMP erachtet, die moderne Geschäftskonzepte ermöglichen können. So könnten beispielsweise Modellierungs- und Policy-Experten als Freelancer unabhängig Umsetzungen und technische Richtlinien erarbeiten oder optimieren, während Endnutzer diese für ihre Systeme hinzukaufen können.
Wie in Abschnitt 3.7.10 beschrieben stellt SyMP nicht sicher, dass einer Analyse alle Informationen zur Verfügung stehen, die im realen System verfügbar sind, sondern nur, dass eine konfigurierte Analyse auch ausgeführt werden kann. Ähnliche Probleme werden in der Forschung mithilfe der Hinzunahme von Unsicherheitsbewertungen bei der Ableitung von Aussagen adressiert (z.B. beim *Reasoning with Uncertainty* [Ert11]). Dabei können fehlende Informationen mit Wahrscheinlichkeitsverteilungen ersetzt werden. Um weitere, wenn auch mit Unsicherheiten behaftete, Analyseergebnisse zu erzielen, sollten zukünftig entsprechende Ableitungsmethoden untersucht werden. Eine Herausforderung dabei wird die Datengrundlage zur Ermittlung der Wahrscheinlichkeitsverteilungen darstellen, da keine Statistiken für beispielsweise Konfigurationsinformationen existieren.

Die Parallelisierung von Aufgaben ist eine der Stärken des Einsatzes von Workflows und verteilten Arbeiter-Programmen. Wie in Abschnitt 3.7.10 beschrieben, bereitet die stark heterogene, zur Laufzeit der Workflow-Generierung nicht bekannte Ausführungsdauer der Aufgabenabarbeitung Probleme, diese Stärke optimal zu nutzen. Um dem entgegenzuwirken, können in zukünftiger Forschung Ansätze untersucht werden, welche die tatsächliche Laufzeit bestmöglich vorhersagen und so den Workflow-Generator bei der Parallelisierung von Aufgaben unterstützen, um sich dem jeweils optimalen Workflow besser anzunähern.
Außerdem soll in zukünftiger Forschung die in Abschnitt 3.7.10 automatisierte Konfiguration und Wiederverwendbarkeitsunterstützung für die ersten drei SyMP-Phasen, also die Systemmodell-erzeugenden Phasen, adressiert werden, um insbesondere den manuellen Aufwand bei der initialen Konfiguration dieser Phasen zu verringern. Der Grundstein hierfür ist mit der Wissensbasis für Aufgaben, zu denen auch jene der ersten drei Phasen gehören, bereits gelegt. Jedoch soll eine ähnliche, Akteur-fokussierte Anwendung für Endnutzer wie Administratoren und weitere Systemexperten umgesetzt werden, wie sie für Sicherheitsexperten bereits existiert.
Das beschriebene SDN-AIR-Konzept wurde unabhängig von den Arbeiten zur automatisierten, minimalinvasiven Sicherheitsanalyse erarbeitet. Allerdings überschneiden sich die zur Ausführung notwendigen Informationen beider Lösungen bei gleichzeitigem Einsatz in einem System. In zukünftiger Forschung soll die Ausnutzung dieses Zusammenhangs gefördert werden. Beispielsweise kann das Systemmodell durch Netzwerkkonfigurationswissen aus dem SDN-Topologieinventar erweitert werden. Außerdem kann SDN-AIR mit Analyseergebnissen versorgt und somit die Sicherheitsanalyse nicht nur, wie bereits möglich, als eine Meldungsquelle für SDN-AIR eingesetzt werden, sondern auch um eine fortlaufende Sicherheitsprüfung des aktuellen SDN-basierten Netzwerks bereitzustellen.
Zuletzt sind bei der Umsetzung von Reaktionen durch SDN-AIR verschiedene Problematiken aufgefallen, wie das Priorisieren der umgeleiteten Verbindung bei der Überwachung von `openflow:1` (vgl. Abschnitt 4.6.5). Obwohl die Einsetzbarkeit SDN-basierter Vorfallreaktion in industriellen Systemen durch SDN-AIR ermöglicht wird, sind diese Probleme weiter zu untersuchen. So

sollte abgeleitet werden, welche Restriktionen für welche Reaktionen zu konfigurieren sind. Dadurch kann eventuell eine Menge von Reaktionen identifiziert werden, die solche Effekte gar nicht besitzen.
Dieser Abschnitt machte noch einmal deutlich, dass die in dieser Dissertation vorgestellten Ergebnisse interessante neue Forschungsthemen auf den Gebieten der automatisierten, modellbasierten Sicherheitsanalyse und SDN-basierten Vorfallreaktion für industrielle Systeme aufzeigen.

Literatur

[Abd09] Abdoli, F. and Kahani, M.: „Ontology-based Distributed Intrusion Detection System“. In: *14th International CSI Computer Conference*. Hrsg. von Staff, IEEE. Tehran, Iran: IEEE, 2009, S. 65–70. doi: 10.1109/CSICC.2009.5349372 (siehe S. 94).

[Al 16a] Al Balushi, Abdullah; McLaughlin, Kieran and Sezer, Sakir: „Contextual Intrusion Alerts for Scada Networks - An Ontology based Approach for Intrusion Alerts Post Processing“. In: *Proceedings of the 2nd International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and and Technology Publications, Feb. 2016, S. 457–464. doi: 10.5220/0005745504570464 (siehe S. 94).

[Al 16b] Al Balushi, Abdullah; McLaughlin, Kieran and Sezer, Sakir: „OSCIDS: An Ontology based SCADA Intrusion Detection Framework“. In: *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications*. SCITEPRESS - Science and Technology Publications, Juli 2016, S. 327–335. doi: 10.5220/0005969803270335 (siehe S. 94).

[Ale20] Alexander, Otis; Belisle, Misha and Steele, Jacob: MITRE ATT&CK® for Industrial Control Systems: Design and Philosophy. online, 2020. url: https://collaborate.mitre.org/attackics/img_auth.php/3/37/ATT%26CK_for_ICS_-_Philosophy_Paper.pdf (besucht am 22. 04. 2021) (siehe S. 33).

[Amm16] Ammar, Moustafa; Rizk, Mohamed; Abdel-Hamid, Ayman and Aboul-Seoud, Ahmed K.: „A Framework for Security Enhancement in SDN-Based Datacenters“. In: *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*.

IEEE, Nov. 2016, S. 1–4. DOI: 10.1109/NTMS.2016.7792427 (siehe S. 245).

[Ana05] ANAGNOSTOPOULOS, T.; ANAGNOSTOPOULOS, C. and HADJIEFTHYMIADES, S.: „Enabling attack behavior prediction in ubiquitous environments". In: *Proceedings International Conference on Pervasive Services, ICPS '05*. Piscataway, NJ: IEEE Service Center, 2005, S. 425–428. DOI: 10.1109/PERSER.2005.1506559 (siehe S. 94).

[App19] APPLIED RISK BV: The State of Industrial Cyber Security 2019. Hrsg. von APPLIED RISK BV. 2019. URL: https://applied-risk.com/resources/the-state-of-industrial-cyber-security-2019 (besucht am 22. 04. 2021) (siehe S. 2).

[Ara05] ARANO, Silvia: Thesauruses and ontologies. Hrsg. von HIPERTEXT.NET. 2005. URL: http://eprints.rclis.org/8972/2/12.pdf (besucht am 22. 04. 2021) (siehe S. 37).

[Asc16] ASCOLAB AND SIGNON AND TÜV SÜD: Sicherheitsanalyse OPC UA. Hrsg. von BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK (BSI). Bonn, Apr. 2016. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/OPCUA/OPCUA.pdf?__blob=publicationFile&v=3 (besucht am 22. 04. 2021) (siehe S. 17, 28).

[Aut14] AUTOMATIONML CONSORTIUM, Hrsg.: AutomationML - The Glue for Seamless Automation Engineering: AutomationML Whitepaper Communication. 2014. URL: https://www.automationml.org/o.red/uploads/dateien/1459418220-AutomationML%20Whitepaper%20-%20AutomationML%20Communication%20v1_Sept2014.pdf (besucht am 22. 04. 2021) (siehe S. 113, 115, 116, 118).

[AUT16] AUTOMATIONML E.V. / OPC FOUNDATION: OPC UA for AutomationML: Release 1.00. Hrsg. von OPC FOUNDATION. 2016. URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture/opc-unified-architecture-for-automationml/ (besucht am 22. 04. 2021) (siehe S. 124).

[Bai19] Bai, Wei; Pan, Zhisong; Guo, Shize; Chen, Zhe and Xia, Shiming: „MDC-Checker: A novel network risk assessment framework for multiple domain configurations“. In: *Computers & Security* 86 (2019), S. 388–401. doi: 10.1016/j.cose.2019.06.016 (siehe S. 92, 140).

[Bak19] Bakakeu, J.; Brossog, M.; Zeitler, J.; Franke, J.; Tolksdorf, S.; Klos, H. and Peschke, J.: „Automated Reasoning and Knowledge Inference on OPC UA Information Models“. In: *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*. IEEE, 2019, S. 53–60. doi: 10.1109/ICPHYS.2019.8780114 (siehe S. 127).

[Bas09] Basile, Cataldo; Lioy, Antonio; Scozzi, Salvatore and Vallini, Marco: „Ontology-Based Policy Translation“. In: *Computational Intelligence in Security for Information Systems*. Hrsg. von Kacprzyk, Janusz; Herrero, Álvaro; Gastaldo, Paolo; Zunino, Rodolfo and Corchado, Emilio. Bd. 63. Advances in Intelligent and Soft Computing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 117–126. doi: 10.1007/978-3-642-04091-7_15 (siehe S. 96).

[Bat04] Battistoni, Roberto; Gabrielli, Emanuele and Mancini, Luigi V.: „A Host Intrusion Prevention System for Windows Operating Systems“. In: *Computer Security – ESORICS 2004*. Bd. 3193. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, S. 352–368. doi: 10.1007/978-3-540-30108-0_22 (siehe S. 245).

[Bau89] Bauer, D. S.; Eichelman, F. R.; Herrera, R. M. and Irgon, A. E.: „Intrusion Detection: An Application of Expert Systems to Computer Security“. In: *Proceedings. International Carnahan Conference on Security Technology*. IEEE, Okt. 1989, S. 97–100. doi: 10.1109/CCST.1989.751961 (siehe S. 93).

[Ben08] Ben-Gal, Irad: „Bayesian Networks“. In: *Encyclopedia of Statistics in Quality and Reliability*. Hrsg. von Ruggeri, Fabrizio; Kenett, Ron S. and Faltin, Frederick W. Bd. 10. Chichester, UK:

John Wiley & Sons, Ltd, 2008, S. 1. DOI: 10.1002/9780470061572.eqr089 (siehe S. 84).

[Bet20] Betten, Niklas: „Rule Management for Complex Ontology Manipulation“. Bachelorarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2020 (siehe S. 174).

[Bie17] Bierman, Andy; Bjorklund, Martin and Watsen, Kent: RESTCONF Protocol. RFC 8040. RFC Editor, Jan. 2017, S. 1–136. URL: https://tools.ietf.org/html/rfc8040 (besucht am 22. 04. 2021) (siehe S. 258).

[Bla05] Blake Shepard u. a.: „A Knowledge-Based Approach to Network Security: Applying Cyc in the Domain of Network Risk Assessment“. In: *AAAI*. 2005 (siehe S. 86).

[Bla11] Blanco, Carlos; Lasheras, Joaquín; Fernández-Medina, Eduardo; Valencia-García, Rafael and Toval, Ambrosio: „Basis for an integrated security ontology according to a systematic review of existing proposals“. In: *Computer Standards & Interfaces* 33.4 (2011), S. 372–388. DOI: 10.1016/j.csi.2010.12.002 (siehe S. 4, 80).

[Bos20] Boss, Birgit; Malakuti, Somayeh; Lin, Shi-Wan; Usländer, Thomas; Clauer, Erich; Hoffmeister, Michael and Stojanovic, Ljiljana: Digital Twin and Asset Administration Shell Concepts and Application in the Industrial Internet and Industrie 4.0: An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper. Hrsg. von Industrial Internet Consortium, Plattform Industrie 4.0. online, 2020. URL: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Digital-Twin-and-Asset-Administration-Shell-Concepts.pdf?__blob=publicationFile&v=11 (besucht am 22. 04. 2021) (siehe S. 25).

[Bou17] Boualem, Si Ahmed; Meryem, Berrani and Fatima, Nibouche: „Maintenance & information security ontology“. In: *2017 4th International Conference on Control, Decision and Information*

Technologies (CoDIT). Piscataway, NJ: IEEE, 2017, S. 0312–0317. DOI: 10.1109/CoDIT.2017.8102610 (siehe S. 46, 96).

[Bou19] Bourekkache, Samir; Kazar, Okba and Aloui, Ahmed: „Computer and Network Security: Ontological and Multi-agent System for Intrusion Detection". In: *Journal of Digital Information Management* 17.3 (2019), S. 133. DOI: 10.6025/jdim/2019/17/3/133-144 (siehe S. 46, 94).

[Bro17] Bromiley, Matt: „The Show Must Go On! The 2017 SANS Incident Response Survey". In: (Juni 2017). URL: https://www.sans.org/reading-room/whitepapers/incident/paper/37815 (besucht am 22. 04. 2021) (siehe S. 5).

[Bun13] Bundesamt für Sicherheit in der Informationstechnik (BSI): ICS-Security-Kompendium. 2013. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/ICS/ICS-Security_kompendium_pdf.pdf?__blob=publicationFile&v=2 (besucht am 22. 04. 2021) (siehe S. 30).

[Bus14] Buschle, Markus; Holm, Hannes; Sommestad, Teodor; Ekstedt, Mathias and Shahzad, Khurram: „A Tool for Automatic Enterprise Architecture Modeling". In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Hrsg. von Bayro-Corrochano, Eduardo and Hancock, Edwin. Bd. 8827. Lecture Notes in Computer Science. Cham: Springer International Publishing, Imprint und Springer, 2014, S. 1–15. DOI: 10.1007/978-3-642-29749-6_1 (siehe S. 84).

[Cas90] Case, Jeffrey D.; Davin, James R.; Fedor, Mark and Schoffstall, Martin Lee: A Simple Network Management Protocol (SNMP). RFC 1098. RFC Editor, Mai 1990, S. 1–35. URL: https://tools.ietf.org/rfc/rfc1157.txt (besucht am 22. 04. 2021) (siehe S. 257).

[Chi10] Chien, Eric; Falliere, Nicolas and O Murchu, Liam: W32.Stuxnet Dossier. Hrsg. von Symantec Corporation. Nov. 2010. URL: https://www.wired.com/images_blogs/threatlevel/

2010/11/w32_stuxnet_dossier.pdf (besucht am 22. 04. 2021) (siehe S. 1).

[Chi17] Chi, Yaping; Jiang, Tingting; Li, Xiao and Gao, Cong: „Design and implementation of cloud platform intrusion prevention system based on SDN". In: *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*. IEEE, März 2017, S. 847–852. doi: 10.1109/ICBDA.2017.8078757 (siehe S. 245).

[Chu13] Chung, Chun-Jen; Khatkar, Pankaj; Xing, Tianyi; Lee, Jeongkeun and Huang, Dijiang: „NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems". In: *IEEE Transactions on Dependable and Secure Computing* 10.4 (2013), S. 198–211. doi: 10.1109/TDSC.2013.8 (siehe S. 244).

[Col12] Colace, Francesco; Santo, Massimo de and Ferrandino, Salvatore: „A Slow Intelligent Approach for the Improvement of Intrusion Detection and Prevention System". In: *Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. Piscataway, NJ: IEEE, 2012, S. 130–137. doi: 10.1109/IMIS.2012.128 (siehe S. 94).

[Com09] Commission, International Electrotechnical: IEC TS 62443-1-1:2009: Industrial communication networks - Network and system security - Part 1-1: Terminology, concepts and models. Techn. Ber. Geneva, Schweiz: International Electrotechnical Commission, 2009. url: https://webstore.iec.ch/publication/7029 (besucht am 22. 04. 2021) (siehe S. 29, 30).

[Com20] Commission, International Electrotechnical: IEC 62443-3-2:2020: Security for industrial automation and control systems - Part 3-2: Security risk assessment for system design. Techn. Ber. Geneva, Schweiz: International Electrotechnical Commission, 2020. url: https://webstore.iec.ch/publication/30727 (besucht am 22. 04. 2021) (siehe S. 31, 32).

[Cor18] Cordova, Ruben F.; Marcovich, Armando L. and Santivanez, Cesar A.: „An Efficient Method for Ontology-Based Multi-Vendor Firewall Misconfiguration Detection: A Real-Case

Study“. In: *2018 IEEE ANDESCON*. Hrsg. von Cely Callejas, José David. [Piscataway, New Jersey]: IEEE, 2018, S. 1–3. doi: 10.1109/ANDESCON.2018.8564655 (siehe S. 144).

[Dam01] Damianou, Nicodemos; Dulay, Naranker; Lupu, Emil and Sloman, Morris: „The Ponder Policy Specification Language“. In: *Policies for Distributed Systems and Networks*. Hrsg. von Goos, Gerhard; Hartmanis, Juris; van Leeuwen, Jan; Sloman, Morris; Lupu, Emil C. and Lobo, Jorge. Bd. 1995. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Nature, 2001, S. 18–38. doi: 10.1007/3-540-44569-2_2 (siehe S. 149).

[Dar20] Darktrace: Darktrace Cyber AI Analyst: Augmenting Your Security Team with AI-Driven Investigations. 2020. url: https://em360tech.com/sites/default/files/2021-01/Darktrace%20Cyber%20AI%20Analyst.pdf (besucht am 09. 05. 2021) (siehe S. 244).

[Dav13] Davy, Steven; Barron, Jason; Shi, Lei; Butler, Bernard; Jennings, Brendan; Griffin, Keith and Collins, Kevin: „A language driven approach to multi-system access control“. In: *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)* (2013), S. 1004–1008 (siehe S. 144).

[Di 17] Di Lallo, Roberto; Griscioli, Federico; Lospoto, Gabriele; Mostafaei, Habib; Pizzonia, Maurizio and Rimondini, Massimo: „Leveraging SDN to monitor critical infrastructure networks in a smarter way“. In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, Mai 2017, S. 608–611. doi: 10.23919/INM.2017.7987341 (siehe S. 246, 252).

[Dol83] Dolev, D. and Yao, A.: „On the Security of Public Key Protocols“. In: *IEEE Transactions on Information Theory* 29.2 (1983), S. 198–208. doi: 10.1109/TIT.1983.1056650 (siehe S. 69, 247).

[Dug05] Duggan, David P.; Berg, Michael; Dillinger, John and Stamp, Jason: Penetration Testing of Industrial Control Systems: SANDIA REPORT SAND2005-2846P. Hrsg. von Sandia National Laboratories. online, 2005. url: https://energy.sandia.gov/wp-

content/gallery/uploads/sand_2005_2846p.pdf (besucht am 22. 04. 2021) (siehe S. 11, 109).

[Eck20] Eckhart, Matthias; Ekelhart, Andreas and Weippl, Edgar R.: „Automated Security Risk Identification Using AutomationML-based Engineering Data". In: *IEEE Transactions on Dependable and Secure Computing* (2020), S. 1. doi: 10.1109/TDSC.2020.3033150 (siehe S. 46, 88).

[Eks15] Ekstedt, Mathias; Johnson, Pontus; Lagerstrom, Robert; Gorton, Dan; Nydren, Joakim and Shahzad, Khurram: „SecuriCAD by Foreseeti: A CAD Tool for Enterprise Cyber Security Management". In: *19th IEEE International Enterprise Distributed Object Computing Conference workshops, EDOCW 2015*. Hrsg. von Hallé, Sylvain and Mayer, Wolfgang. Piscataway, NJ: IEEE, 2015, S. 152–155. doi: 10.1109/EDOCW.2015.40 (siehe S. 86).

[Enn11] Enns, Rob; Bjorklund, Martin; Schoenwaelder, Juergen and Bierman, Andy: Network Configuration Protocol (NETCONF). RFC 6241. RFC Editor, Juni 2011, S. 1–112. url: https://tools.ietf.org/rfc/rfc6241.txt (besucht am 22. 04. 2021) (siehe S. 257).

[Ert11] Ertel, Wolfgang, Hrsg.: Introduction to Artificial Intelligence. Undergraduate Topics in Computer Science. London: Springer London, 2011. doi: 10.1007/978-0-85729-299-5 (siehe S. 281).

[Eur18] European Union Agency for Network and Information Security (ENISA): Good Practices for Security of Internet of Things in the context of Smart Manufacturing. Hrsg. von European Union Agency for Network and Information Security. 2018. url: https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot (besucht am 22. 04. 2021) (siehe S. 30).

[Fen09a] Fenz, Stefan and Ekelhart, Andreas: „Formalizing Information Security Knowledge". In: *Proceedings of the 4th International Symposium on Information, Computer, and Communications*

Security - ASIACCS '09. Hrsg. von Safavi-Naini, Rei; Varadharajan, Vijay; Li, Wanqing; Susilo, Willy and Tupakula, Udaya. New York, New York, USA: ACM Press, 2009, S. 183. doi: 10.1145/1533057.1533084 (siehe S. 82, 87).

[Fen09b] Fenz, Stefan; Pruckner, Thomas and Manutscheri, Arman: „Ontological Mapping of Information Security Best-Practice Guidelines". In: *Business Information Systems*. Hrsg. von Abramowicz, Witold. Bd. 21. Lecture Notes in Business Information Processing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 49–60 (siehe S. 87, 88).

[Fen10] Fenz, Stefan: „Ontology-based Generation of IT-Security Metrics". In: *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*. Hrsg. von Shin, Sung Y.; Ossowski, Sascha; Schumacher, Michael; Palakal, Mathew J. and Hung, Chih-Cheng. New York, New York, USA: ACM Press, 2010, S. 1833. doi: 10.1145/1774088.1774478 (siehe S. 87).

[Fen15] Fenz, Stefan; Heurix, Johannes and Neubauer, Thomas: How to Increase the Inventory Efficiency in Information Security Risk and Compliance Management. 2015. doi: 10.18151/7217311 (siehe S. 88, 109).

[Fen18] Fenz, Stefan and Neubauer, Thomas: „Ontology-based information security compliance determination and control selection on the example of ISO 27002". In: *Information and Computer Security* 26.5 (2018), S. 551–567. doi: 10.1108/ICS-02-2018-0020 (siehe S. 106, 158).

[Fit07] Fitzgerald, William M.; Foley, Simon N. and O'Foghlu, M.: „Confident firewall policy configuration management using description logic". In: *12th Nordic Workshop on Secure IT Systems*. 2007, S. 11–12 (siehe S. xviii, 91, 144).

[Fit08] Fitzgerald, William M.; Foley, Simon N. and Foghlú, M\'\icheál Ó.: „Network Access Control Interoperation using Semantic Web Techniques". In: *12th Nordic Workshop on Secure IT Systems*. 2008, S. 26–37 (siehe S. 91, 144).

[Fit09] Fitzgerald, William M.; Foley, Simon N.; Foghlú, Micheál Ó. u. a.: „Network access control configuration management using semantic web techniques“. In: *Journal of Research and Practice in Information Technology* 41.2 (2009), S. 99 (siehe S. 144).

[Fol08] Foley, Simon N. and Fitzgerald, William M.: „Semantic Web and Firewall Alignment“. In: *2008 IEEE 24th International Conference on Data Engineering Workshop*. 2008, S. 447–453 (siehe S. 91, 144).

[Fra17a] Franco Rosa, Ferrucio de; Bonacin, Rodrigo and Jino, Mario: The Security Assessment Domain: A Survey of Taxonomies and Ontologies. 2017. doi: 10.13140/RG.2.2.12437.73441 (siehe S. 6, 81).

[Fra17b] Franco Rosa, Ferrucio de and Jino, Mario: „A Survey of Security Assessment Ontologies“. In: *Recent Advances in Information Systems and Technologies*. Hrsg. von Rocha, Álvaro; Correia, Ana Maria; Adeli, Hojjat; Reis, Luís Paulo and Costanzo, Sandra. Bd. 569. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2017, S. 166–173. doi: 10.1007/978-3-319-56535-4_17 (siehe S. 4, 6, 81).

[Fra18] Franco Rosa, Ferrucio de; Jino, Mario and Bonacin, Rodrigo: „Towards an Ontology of Security Assessment: A Core Model Proposal“. In: *Information Technology- New Generations*. Hrsg. von Latifi, Shahram. Bd. 738. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2018, S. 75–80. doi: 10.1007/978-3-319-77028-4_12 (siehe S. 81–83).

[Fry12] Frye, Lisa; Cheng, Liang and Heflin, Jeff: „An ontology-based system to identify complex network attacks“. In: *IEEE International Conference on Communications (ICC)*. IEEE, 2012, S. 6683–6688. doi: 10.1109/ICC.2012.6364689 (siehe S. 94).

[Gal13] Galloway, Brendan and Hancke, Gerhard P.: „Introduction to Industrial Control Networks“. In: *IEEE Communications Surveys & Tutorials* 15.2 (2013), S. 860–880. doi: 10.1109/SURV.2012.071812.00124 (siehe S. 249).

[Gam11] Gamal, Maher Mohamed; Hasan, Bahaa and Hegazy, Abdel Fatah: „A Security Analysis Framework Powered by an Expert System“. In: *International Journal of Computer Science and Security (IJCSS)* 4.6 (2011), S. 505 (siehe S. 86, 87).

[Gao13] Gao, Jian-bo; Zhang, Bao-wen; Chen, Xiao-hua and Luo, Zheng: „Ontology-Based Model of Network and Computer Attacks for Security Assessment“. In: *Journal of Shanghai Jiaotong University (Science)* 18.5 (2013), S. 554–562. doi: 10.1007/s12204-013-1439-5 (siehe S. 86, 87).

[Gla16] Glawe, Matthias and Fay, Alexander: „Wissensbasiertes Engineering automatisierter Anlagen unter Verwendung von AutomationML und OWL“. In: *at - Automatisierungstechnik* 64.3 (2016). doi: 10.1515/auto-2015-0077 (siehe S. 124).

[Gon07] Gonzalez, Jose M.; Paxson, Vern and Weaver, Nicholas: „Shunting: a hardware/software architecture for flexible, high-performance network intrusion prevention“. In: *Proceedings of the 14th ACM conference on Computer and communications security - CCS '07*. Hrsg. von Ning, Peng; Di Capitani Vimercati, Sabrina de and Syverson, Paul. New York, New York, USA: ACM Press, 2007, S. 139. doi: 10.1145/1315245.1315264 (siehe S. 245).

[Gri15] Grieves, Michael: Digital Twin: Manufacturing Excellence through Virtual Factory Replication. Hrsg. von Dassault Systèmes. online, 2015. url: https://www.researchgate.net/publication/275211047_Digital_Twin_Manufacturing_Excellence_through_Virtual_Factory_Replication (besucht am 22. 04. 2021) (siehe S. 25).

[Gru93] Gruber, Thomas R.: „A translation approach to portable ontology specifications“. In: *Knowledge Acquisition* 5.2 (1993), S. 199–220. doi: 10.1006/knac.1993.1008 (siehe S. 4, 36).

[Har20] Hardt, Henrike: „Optimierung und Generalisierung Ontologiebasierter ICS Security Analyse“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2020 (siehe S. 281).

[Has08] Haslum, Kjetil; Moe, Marie E. G. and Knapskog, Svein J.: „Real-time intrusion prevention and security analysis of networks using HMMs“. In: *2008 33rd IEEE Conference on Local Computer Networks (LCN)*. IEEE, Okt. 2008, S. 927–934. doi: 10.1109/LCN.2008.4664305 (siehe S. 243).

[He04] He, Yanxiang; Chen, Wei; Yang, Min and Peng, Wenling: „Ontology Based Cooperative Intrusion Detection System“. In: *Network and Parallel Computing. NPC 2004*. Hrsg. von Jin, Hai. Bd. 3222. Lecture Notes in Computer Science. Berlin und New York: Springer, 2004, S. 419–426. doi: 10.1007/978-3-540-30141-7_59 (siehe S. 94).

[Her07] Herzog, Almut; Shahmehri, Nahid and Duma, Claudiu: „An Ontology of Information Security“. In: *International Journal of Information Security and Privacy (IJISP)* 1.4 (2007), S. 1–23. doi: 10.4018/jisp.2007100101 (siehe S. 82).

[Heß18] Hess, Maximilian: „Regelbasiertes Incident Handling für Software-defined Networks in industriellen Umgebungen“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2018 (siehe S. 254, 264, 265, 267–269, 344).

[Hit08] Hitzler, Pascal; Krötzsch, Markus; Rudolph, Sebastian and Sure, York: Semantic Web. Springer-Verlag Berlin Heidelberg, 2008 (siehe S. xvii, 37, 38).

[Hol13] Holm, Hannes; Sommestad, Teodor; Ekstedt, Mathias and Nordström, Lars: „CySeMoL: A tool for cyber security analysis of enterprises“. In: *22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013)*. Piscataway, N.J.: IEEE, 2013, S. 1109. doi: 10.1049/cp.2013.1077 (siehe S. 84–86).

[Hol14] Holm, Hannes: „A Framework and Calculation Engine for Modeling and Predicting the Cyber Security of Enterprise Architectures“. Diss. Stockholm, Schweden: KTH, Royal Institute of Technology, 2014 (siehe S. 84).

[Hol15a] Holm, Hannes; Korman, Matus and Ekstedt, Mathias: „A Bayesian network model for likelihood estimations of acquirement of critical software vulnerabilities and exploits“. In: *Information and Software Technology* 58 (2015), S. 304–318. doi: 10.1016/j.infsof.2014.07.001 (siehe S. 84).

[Hol15b] Holm, Hannes; Shahzad, Khurram; Buschle, Markus and Ekstedt, Mathias: „P2CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language“. In: *IEEE Transactions on Dependable and Secure Computing* 12.6 (2015), S. 626–639. doi: 10.1109/TDSC.2014.2382574 (siehe S. 84).

[Hu11] Hu, Hongxin; Ahn, Gail-Joon and Kulkarni, Ketan: „Ontology-based Policy Anomaly Management for Autonomic Computing“. In: *Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*. IEEE, Okt. 2011. doi: 10.4108/icst.collaboratecom.2011.247119 (siehe S. 144).

[Hua18] Hua, Yingbing and Hein, Bjorn: „Concept Learning in AutomationML with Formal Semantics and Inductive Logic Programming“. In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. [Piscataway, New Jersey]: IEEE, 2018, S. 1542–1547. doi: 10.1109/COASE.2018.8560541 (siehe S. 88).

[Hua19] Hua, Yingbing and Hein, Bjorn: „Interpreting OWL Complex Classes in AutomationML based on Bidirectional Translation“. In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sep. 2019, S. 79–86. doi: 10.1109/ETFA.2019.8869456 (siehe S. 88).

[Int14] International Electrotechnical Commission: IEC 62714-1:2014: Engineering data exchange format for use in industrial automation systems engineering - Automation markup language - Part 1: Architecture and general requirements. 2014. url: https://webstore.iec.ch/publication/7388 (besucht am 22. 04. 2021) (siehe S. 119).

[Int16] International Electrotechnical Commission: IEC 62424:2016 Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools. Hrsg. von International Electrotechnical Commission. 2016. URL: https://www.vde-verlag.de/iec-normen/223713/iec-62424-2016.html (besucht am 14. 05. 2021) (siehe S. 26).

[Int18] International Electrotechnical Commission: IEC 62714-1:2018 Engineering data exchange format for use in industrial automation systems engineering – Automation markup language – Part 1: Architecture and general requirements. Hrsg. von International Electrotechnical Commission. 2018. URL: https://www.vde-verlag.de/iec-normen/225580/iec-62714-1-2018.html (besucht am 22. 04. 2021) (siehe S. 26).

[ISO94] ISO/IEC: Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model. 1994. URL: https://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip (besucht am 22. 04. 2021) (siehe S. 117).

[Jef04] Jeffrey Undercoffer , John Pinkston, Anupam Joshi, Timothy Finin: „A Target-Centric Ontology for Intrusion Detection“. In: *PROCEEDING OF THE IJCAI-03 WORKSHOP ON ONTOLOGIES AND DISTRIBUTED SYSTEMS*. Hrsg. von Morgan Kaufmann Pub. 2004, S. 47–58 (siehe S. 93).

[Jin13] Jin, Hai; Xiang, Guofu; Zou, Deqing; Wu, Song; Zhao, Feng; Li, Min and Zheng, Weide: „A VMM-based intrusion prevention system in cloud computing environment“. In: *The Journal of Supercomputing* 66.3 (2013), S. 1133–1151. DOI: 10.1007/s11227-011-0608-2 (siehe S. 245).

[Kag02] Kagal, Lalana: Rei: A Policy Language for the Me-Centric Project. 2002. DOI: 10.13016/M2MG5B-HRA9 (siehe S. 149).

[Kah19] Kahles, David: „Automatisierte Analyse der Umsetzung von Security-Best-Practices und -Standards in industriellen Systemen“. Bachelorarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2019 (siehe S. 184).

[Kah21] Kahles, David: „Threat Classification in Industrial Networks Based on Heterogeneous Information“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2021 (siehe S. 221).

[Kal03] Kalam, A.A.E.; Baida, R. E.; Balbiani, P.; Benferhat, S.; Cuppens, F.; Deswarte, Y.; Miege, A.; Saurel, C. and Trouessin, G.: „Organization based access control“. In: *Proceedings*. Los Alamitos, Ca.: IEEE Computer Society, 2003. doi: 10.1109/policy.2003.1206966 (siehe S. 149).

[Kie19] Kiesling, Elmar; Ekelhart, Andreas; Kurniawan, Kabul and Ekaputra, Fajar: „The SEPSES Knowledge Graph: An Integrated Resource for Cybersecurity“. In: *The Semantic Web - ISWC 2019*. Hrsg. von Ghidini, Chiara; Hartig, Olaf; Maleshkova, Maria; Svátek, Vojtěch; Cruz, Isabel; Hogan, Aidan; Song, Jie; Lefrançois, Maxime and Gandon, Fabien. Information Systems and Applications, incl. Internet/Web, and HCI. Cham: Springer International Publishing und Springer, 2019, S. 198–214 (siehe S. 81, 82, 89).

[Klo21] Kloppenburg, Manuel: „Flexible, wissensbasierte Analyseplattform für die Identifikation von Sicherheitsproblemen industrieller Systeme“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2021 (siehe S. 166, 170, 178, 228, 229).

[Kon15] Kong, Derrick; Mandelberg, David; Lapets, Andrei; Watro, Ronald J.; Smith, Daniel and Runkle, Matthew: „Lightbulb: A Toolkit for Analysis of Security Policy Interactions“. In: *The Fourteenth International Conference on Networks (INC 2015)*. International Academy, Research, und Industry Association (IARIA), 2015, S. 151–156 (siehe S. 90).

[Kot18] Kotenko, Igor; Fedorchenko, Andrey; Doynikova, Elena and Chechulin, Andrey: „An Ontology-based Storage of Security Information“. In: *Information Technology And Control* 47.4 (2018). doi: 10.5755/j01.itc.47.4.20007 (siehe S. 81, 86, 87).

[Kou17] Koulouris, Theofrastos; Casassa Mont, Marco and Arnell, Simon: SDN4S: Software Defined Networking for Security. 2017 (siehe S. 245, 255).

[Kra11] Kral, Patrick: Incident Handler's Handbook. Hrsg. von SANS Institute. 2011. url: https://www.sans.org/reading-room/whitepapers/incident/incident-handlers-handbook-33901 (besucht am 22. 04. 2021) (siehe S. 54).

[Lae09] Laender, Alberto H. F.; Castano, Silvana; Dayal, Umeshwar; Casati, Fabio; de Oliveira, José Palazzo M.; Elahi, Golnaz; Yu, Eric and Zannone, Nicola, Hrsg.: A Modeling Ontology for Integrating Vulnerabilities into Security Requirements Conceptual Foundations: Conceptual Modeling - ER 2009. Springer Berlin Heidelberg, 2009 (siehe S. 87).

[Lem14] Lemaire, Laurens; Lapon, Jorn; Decker, Bart de and Naessens, Vincent: „A SysML Extension for Security Analysis of Industrial Control Systems“. In: *ICS-CSR 2014: Proceedings of the 2nd International Symposium on ICS & SCADA Cyber Security Research 2014*. Electronic Workshops in Computing. BCS Learning & Development, 2014. doi: 10.14236/ewic/ICSCSR2014.1 (siehe S. 85).

[Li10] Li, Wan and Tian, Shengfeng: „An ontology-based intrusion alerts correlation system“. In: *Expert Systems with Applications* 37.10 (2010), S. 7138–7146. doi: 10.1016/j.eswa.2010.03.068 (siehe S. 94).

[Lóp09] López de Vergara, Jorge E.; Vázquez, Enrique; Martin, Antony; Dubus, Samuel and Lepareux, Marie-Noëlle: „Use of Ontologies for the Definition of Alerts and Policies in a Network Security Platform“. In: *Journal of Networks* 4.8 (2009). doi: 10.4304/jnw.4.8.720-733 (siehe S. 96).

[Lüd16] Lüder, Arndt and Schmidt, Nicole: „AutomationML in a Nutshell“. In: *Handbuch Industrie 4.0*. Hrsg. von Vogel-Heuser, Birgit; Bauernhansl, Thomas and Hompel, Michael ten. Bd. 9. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, S. 1–46. doi: 10.1007/978-3-662-45537-1_61-1 (siehe S. 25, 26).

[Man05a] Mandujano, S.; Galvan, A. and Nolazco, J. A.: „An Ontology-based Multiagent Architecture for Outbound Intrusion Detection“. In: *ACS/IEEE International Conference on Computer Systems and Applications*. Pisctaway, N.J.: IEEE, 2005, S. 494–501. doi: 10.1109/AICCSA.2005.1387085 (siehe S. 94).

[Man05b] Mandujano, Salvador: „An Ontology-supported Outbound Intrusion Detection System“. In: 2005. url: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.9249&rep=rep1&type=pdf (besucht am 22. 04. 2021) (siehe S. 94).

[Mar07] Martínez Pérez, Gregorio; García Clemente, Félix J. and Gómez Skarmeta, Antonio F.: „Managing semantic–aware policies in a distributed firewall scenario“. In: *Internet Research* 17.4 (2007), S. 362–377. doi: 10.1108/10662240710828049 (siehe S. 144).

[Mar14] Markus Buschle: „Tool Support for Enterprise Architecture Analysis: with application in cyber security“. Diss. Stockholm, Sweden: KTH, Royal Institute of Technology, 2014 (siehe S. 84).

[McK08] McKeown, Nick; Anderson, Tom; Balakrishnan, Hari; Parulkar, Guru; Peterson, Larry; Rexford, Jennifer; Shenker, Scott and Turner, Jonathan: „OpenFlow: Enabling innovation in campus networks“. In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), S. 69–74. doi: 10.1145/1355734.1355746 (siehe S. 53, 257).

[Mic17] Micro Focus: Micro Focus Security ArcSight Common Event Format: Implementing ArcSight Common Event Format (CEF). 2017. url: https://community.microfocus.com/t5/ArcSight-Connectors/ArcSight-Common-Event-Format-CEF-

Implementation-Standard/ta-p/1645557?attachment-id=68077 (besucht am 22. 04. 2021) (siehe S. 183).

[Mon11] Montesino, Raydel and Fenz, Stefan: „Information Security Automation: How Far Can We Go?“ In: *2011 Sixth International Conference on Availability, Reliability and Security*. IEEE, Aug. 2011, S. 280–285. doi: 10.1109/ARES.2011.48 (siehe S. 4).

[Mor11] Morkevičius, Aurelijus and Gudas, Saulius: „Enterprise Knowledge Based Software Requirements Elicitation“. In: *Information Technology And Control* 40.3 (2011). doi: 10.5755/j01.itc.40.3.626 (siehe S. 85).

[Mor12] More, Sumit; Matthews, Mary; Joshi, Anupam and Finin, Tim: „A Knowledge-Based Approach to Intrusion Detection Modeling“. In: *IEEE Symposium on Security and Privacy workshops (SPW)*. Piscataway, NJ: IEEE, 2012, S. 75–81. doi: 10.1109/SPW.2012.26 (siehe S. 94, 95).

[Moz18] Mozzaquatro, Bruno Augusti; Agostinho, Carlos; Goncalves, Diogo; Martins, João and Jardim-Goncalves, Ricardo: „An Ontology-Based Cybersecurity Framework for the Internet of Things“. In: *Sensors* 18.9 (2018), S. 3053. doi: 10.3390/s18093053 (siehe S. 46, 97, 98).

[Mur18a] Murillo Piedrahita, Andres F.; Gaur, Vikram; Giraldo, Jairo; Cardenas, Alvaro A. and Rueda, Sandra Julieta: „Leveraging Software-Defined Networking for Incident Response in Industrial Control Systems“. In: *IEEE Software* 35.1 (2018), S. 44–50. doi: 10.1109/MS.2017.4541054 (siehe S. 246, 253).

[Mur18b] Murillo Piedrahita, Andrés F.; Gaur, Vikram; Giraldo, Jairo; Cardenas, Alvaro A. and Rueda, Sandra Julieta: „Virtual incident response functions in control systems“. In: *Computer Networks* 135 (2018), S. 147–159. doi: 10.1016/j.comnet.2018.01.040 (siehe S. 246, 253).

[Nar18] Narayanan, Sandeep; Ganesan, Ashwinkumar; Joshi, Karuna; Oates, Tim; Joshi, Anupam and Finin, Tim: Cognitive Techniques for Early Detection of Cybersecurity Events. 2018. url: https://arxiv.org/pdf/1808.00116 (besucht am 22. 04. 2021) (siehe S. 94, 95).

[Nat11] National Institute of Standards and Technology (NIST): Guide to Industrial Control Systems (ICS) Security: NIST Special Publication 800-82. Hrsg. von Stouffer, Keith; Falco, Joe and Scarfone, Karen. Gaithersburg, MD, USA, 2011 (siehe S. 13, 30, 67, 96, 110, 249).

[Nat18] National Institute of Standards and Technology (NIST): Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy. Gaithersburg, MD: National Institute of Standards and Technology (NIST), 2018. doi: 10.6028/NIST.SP.800-37r2 (siehe S. 2, 3).

[Nun14] Nunes, Bruno Astuto A.; Mendonca, Marc; Nguyen, Xuan-Nam; Obraczka, Katia and Turletti, Thierry: „A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks“. In: *IEEE Communications Surveys & Tutorials* 16.3 (2014), S. 1617–1634. doi: 10.1109/SURV.2014.012214.00180 (siehe S. 53).

[Obj13] Object Management Group, Inc.: Business Process Model and Notation (BPMN). Hrsg. von Object Management Group, Inc. (OMG). Needham, MA, USA, 2013. url: https://www.omg.org/spec/BPMN/2.0.2 (besucht am 22. 04. 2021) (siehe S. 49).

[Obj17] Object Management Group, Inc.: OMG® Unified Modeling Language® (OMG UML®). Hrsg. von Object Management Group, Inc. 2017. url: https://www.omg.org/spec/UML/2.5.1/PDF (besucht am 22. 04. 2021) (siehe S. 84).

[OCo09] O'Connor, Martin and Das, Amar: „SQWRL: A Query Language for OWL“. In: *Proceedings of the 6th International Conference*

on OWL: Experiences and Directions - Volume 529. OWLED'09. Aachen, DEU: CEUR-WS.org, 2009, S. 208–215 (siehe S. 45).

[OPC17a] OPC Foundation: OPC Unified Architecture Specification Part 3: Address Space Model. Hrsg. von OPC Foundation. Scottsdale, AZ, USA, 2017. URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-3-address-space-model (besucht am 22. 04. 2021) (siehe S. 28).

[OPC17b] OPC Foundation: OPC Unified Architecture Specification Part 5: Information Model. Hrsg. von OPC Foundation. Scottsdale, AZ, USA, 2017. URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-5-information-model (besucht am 22. 04. 2021) (siehe S. 28).

[Oss15] Ossenbuhl, Sven; Steinberger, Jessica and Baier, Harald: „Towards Automated Incident Handling: How to Select an Appropriate Response against a Network-Based Attack?" In: *2015 Ninth International Conference on IT Security Incident Management & IT Forensics*. IEEE, Mai 2015, S. 51–67. DOI: 10.1109/IMF.2015.13 (siehe S. 243).

[Ou05a] Ou, Xinming: „A Logic-programming Approach to Network Security Analysis". Diss. Princeton, NJ, USA, Nov. 2005 (siehe S. 86, 89).

[Ou05b] MulVAL: A Logic-based Network Security Analyzer. Bd. 14. Proceedings of the 14th Conference on USENIX Security Symposium. 2005. DOI: 10.1002/0471741833.ch1 (siehe S. 86, 89, 90).

[Ouc13] Ouchani, Samir; Mohamed, Otmane Ait and Debbabi, Mourad: „A Security Risk Assessment Framework for SysML Activity Diagrams". In: *2013 IEEE 7th International Conference on Software Security and Reliability*. IEEE, Juni 2013, S. 227–236. DOI: 10.1109/SERE.2013.11 (siehe S. 85).

[Pae20] Paes, Richard; Mazur, David C.; Venne, Bruce K. and Ostrzenski, Jack: „A Guide to Securing Industrial Control Networks: Integrating IT and OT Systems“. In: *IEEE Industry Applications Magazine* 26.2 (2020), S. 47–53. doi: 10.1109/MIAS.2019.2943630 (siehe S. 30).

[Pat17] Patzer, Florian; Sarkar, Aranya; Birnstill, Pascal; Schleipen, Miriam and Beyerer, Jürgen: „Towards the Modelling of Complex Communication Networks in AutomationML“. In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sep. 2017, S. 1–8. doi: 10.1109/ETFA.2017.8247571 (siehe S. 16, 112, 113, 117, 118, 122).

[Pat19a] Patzer, Florian; Kobzan, Thomas and Haas, Christian: FlexSi-Pro : Schlussbericht FlexSi-Pro: Berichtszeitraum: 1.1.2017-31.12.2019. Hrsg. von Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung. online, 2019. url: https://www.tib.eu/de/suchen/id/TIBKAT:1728069513/FlexSi-Pro-Schlussbericht-FlexSi-Pro-Berichtszeitraum (besucht am 23. 04. 2021) (siehe S. 18, 270–273).

[Pat19b] Patzer, Florian; Meshram, Ankush; Birnstill, Pascal; Haas, Christian and Beyerer, Jürgen: „Towards Computer-Aided Security Life Cycle Management for Critical Industrial Control Systems“. In: *Critical Information Infrastructures Security*. Hrsg. von Luiijf, Eric; Žutautaitė, Inga and Hämmerli, Bernhard M. Bd. 11260. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, S. 45–56. doi: 10.1007/978-3-030-05849-4_4 (siehe S. 18).

[Pat19c] Patzer, Florian; Meshram, Ankush and Hess, Maximilian: „Automated Incident Response for Industrial Control Systems Leveraging Software-defined Networking“. In: *Proceedings of the 5th International Conference on Information Systems Security*

and Privacy. SCITEPRESS - Science and Technology Publications, 2019, S. 319–327. DOI: 10.5220/0007359503190327 (siehe S. 18, 254, 257, 258).

[Pat19d] PATZER, Florian; VOLZ, Friedrich; USLÄNDER, Thomas; BLOCHER, Immanuel and BEYERER, Jürgen: „The Industrie 4.0 Asset Administration Shell as Information Source for Security Analysis". In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sep. 2019, S. 420–427. DOI: 10.1109/ETFA.2019.8869059 (siehe S. 16, 112, 125–127, 238).

[Pat20] PATZER, Florian; LÜDTKE, Philipp; MESHRAM, Ankush and BEYERER, Jürgen: „Context-Aware Software-Defined Networking for Automated Incident Response in Industrial Networks". In: *Information Systems Security and Privacy*. Hrsg. von MORI, Paolo; FURNELL, Steven and CAMP, Olivier. Bd. 1221. Communications in Computer and Information Science. Cham: Springer International Publishing, 2020, S. 137–161. DOI: 10.1007/978-3-030-49443-8_7 (siehe S. 18, 254, 257, 258, 260, 275, 276).

[Pat21] PATZER, Florian and BEYERER, Jürgen: „Efficient Semantic Representation of Network Access Control Configuration for Ontology-Based Security Analysis". In: *Proceedings of the 7th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*. SciTePress, 2021, S. 550–557 (siehe S. xviii, 17, 140, 141, 144, 146, 185, 352).

[Per18] PERLROTH, Nicole and KRAUSS, Clifford: „A Cyberattack in Saudi Arabia Had a Deadly Goal. Experts Fear Another Try." In: *The New York Times* (März 2018). URL: https://www.nytimes.com/2018/03/15/technology/saudi-arabia-hacks-cyberattacks.html (besucht am 22. 04. 2021) (siehe S. 1).

[Pfr17] PFRANG, Steffen; MEIER, David and KAUTZ, Valentin: „Towards a Modular Security Testing Framework for Industrial Automation and Control Systems: ISuTest". In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory*

Automation (ETFA). IEEE, Sep. 2017, S. 1–5. DOI: 10.1109/ETFA.2017.8247727 (siehe S. 108).

[Pfr18] Pfrang, Steffen; Meier, David; Friedrich, Michael and Beyerer, Jürgen: „Advancing Protocol Fuzzing for Industrial Automation and Control Systems". In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, Jan. 2018, S. 570–580. DOI: 10.5220/0006755305700580 (siehe S. 108).

[Pin99] Pinto, H. Sofia; Gómez-Pérez, Asunción. and Martins, Joao P.: „Some Issues on Ontology Integration". In: *Proceedings of IJCAI99's Workshop on Ontologies and Problem Solving Methods: Lessons Learned and Future Trends* 18 (1999). URL: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-18/7-pinto.pdf (besucht am 22. 04. 2021) (siehe S. 47, 48).

[Pla15] Plattform Industrie 4.0: Umsetzungsstrategie Industrie 4.0: Ergebnisbericht der Plattform Industrie 4.0. Hrsg. von BITKOM e.V, VDMA e.V, ZVEI e.V. online, 2015. URL: https://www.bitkom.org/sites/default/files/file/import/150410-Umsetzungsstrategie-0.pdf (besucht am 22. 04. 2021) (siehe S. 24).

[Pla18] Plattform Industrie 4.0: Diskussionspapier: Sicherer Bezug von CAE-Daten. Hrsg. von Bundesministerium für Wirtschaft und Energie. 2018. URL: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/sicherer-bezug-von-cae-daten.pdf?__blob=publicationFile&v=8 (besucht am 22. 04. 2021) (siehe S. 16).

[Pla19] Plattform Industrie 4.0: Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 1.0). Hrsg. von Federal Ministry for Economic Affairs and Energy. online, 2019. URL: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V1.pdf?__blob=

publicationFile&v=10 (besucht am 22. 04. 2021) (siehe S. 24, 25, 124, 125).

[Pla20] Plattform Industrie 4.0: Was ist Industrie 4.0? 2020. url: https://www.plattform-i40.de/PI40/Navigation/DE/Industrie40/WasIndustrie40/was-ist-industrie-40.html (besucht am 22. 04. 2021) (siehe S. 23).

[Ras01] Raskin, Victor; Hempelmann, Christian F.; Triezenberg, Katrina E. and Nirenburg, Sergei: „Ontology in information security: a useful theoretical foundation and methodological tool". In: *Proceedings of the 2001 workshop on New security paradigms - NSPW '01*. Hrsg. von Raskin, Victor; Greenwald, Steven J.; Timmerman, Brenda and Kienzle, Darrell. New York, New York, USA: ACM Press, 2001, S. 53. doi: 10.1145/508171.508180 (siehe S. 71).

[Ras05] Rash, Michael and Henmi, Anne: Intrusion prevention and active response: Deploying network and host IPS. Rockland, Mass.: Syngress Pub, 2005 (siehe S. 245).

[Raz09] Razzaq, Abdul; Ahmed, Hafiz Farooq; Hur, Ali and Haider, Nasir: „Ontology based Application Level Intrusion Detection System by using Bayesian Filter". In: *2nd International Conference on Computer, Control & Communication*. Piscataway, N.J.: IEEE, 2009, S. 1–6. doi: 10.1109/IC4.2009.4909223 (siehe S. 94).

[Roe21] Roehr, Marco: „Erweiterung von Systemontologien um Wissen der Informationssicherheit: Konzeption und Implementierung eines Werkzeuges am Beispiel der Normenreihe IEC 62443". Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2021 (siehe S. 146, 228).

[Roh18] Rohith Yanambaka Venkata: „An Ontology-Driven Framework for Security and Resiliency in Cyber Physical Systems". In: 2018. url: https://csrl.cse.unt.edu/kavi/Research/ICSEA-2018.pdf (besucht am 22. 04. 2021) (siehe S. 86, 87).

[Sad14] Sadighian, Alireza; Fernandez, José M.; Lemay, Antoine and Zargar, Saman T.: „ONTIDS: A Highly Flexible Context-Aware and Ontology-Based Alert Correlation Framework“. In: *Foundations and Practice of Security. FPS 2013. Lecture Notes in Computer Science*. Hrsg. von Danger, Jean-Luc; Debbabi, Mourad; Marion, Jean-Yves; Garcia-Alfaro, Joaquin and Zincir-Heywood, Nur. Bd. 8352. LNCS sublibrary. SL 4, Security and cryptology. Cham: Springer, 2014, S. 161–177 (siehe S. 94).

[Sar17] Sarkar, Aranya: „Standardized Device Description in AutomationML with OPC UA Configurations and IT Security Mechanisms“. Masterarbeit. Karlsruhe: Otto von Güricke Universität, 2017 (siehe S. 118).

[Sar20] Sarnovsky, Martin and Paralic, Jan: „Hierarchical Intrusion Detection Using Machine Learning and Knowledge Model“. In: *Symmetry* 12.2 (2020), S. 203. doi: 10.3390/sym12020203 (siehe S. 46, 95).

[Ser18] Sergio Caltagirone: Industrial Control Threat Intelligence. Hrsg. von Dragos Inc. online, 2018. url: https://www.dragos.com/wp-content/uploads/Industrial-Control-Threat-Intelligence-Whitepaper.pdf (besucht am 22. 04. 2021) (siehe S. 244).

[Sha19] Shah, Nadir; Giaccone, Paolo; Rawat, Danda B.; Rayes, Ammar and Zhao, Nan: „Solutions for adopting software defined network in practice“. In: *International Journal of Communication Systems* 32.17 (2019), e3990. doi: 10.1002/dac.3990 (siehe S. 53).

[Si14] Si, Cheng; Zhang, Hongqi; Wang, Yongwei and Liu, Jiang: „Network Security Situation Elements Fusion Method Based on Ontology“. In: *2014 Seventh International Symposium on Computational Intelligence and Design*. IEEE, Dez. 2014, S. 272–275. doi: 10.1109/ISCID.2014.132 (siehe S. 94).

[Sic15] Sicilia, Miguel-Angel; García-Barriocanal, Elena; Bermejo-Higuera, Javier and Sánchez-Alonso, Salvador: „What are Information Security Ontologies Useful for?“ In: *Metadata and Semantics Research*. Hrsg. von Garoufallou, Emmanouel; Hartley, Richard J. and Gaitanou, Panorea. Bd. 544. Communications in Computer and Information Science. Cham: Springer, 2015, S. 51–61. doi: 10.1007/978-3-319-24129-6_5 (siehe S. 4, 35, 80).

[Sim14] Simmons, Chris B.; Shiva, Sajjan G. and Simmons, Lakisha L.: „A Qualitative Analysis of An Ontology Based Issue Resolution System for Cyber Attack Management“. In: *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*. IEEE, Juni 2014, S. 323–329. doi: 10.1109/CYBER.2014.6917483 (siehe S. 86, 87).

[Som12a] Sommerville, Ian: Software Engineering. 9., aktualisierte Auflage. Always Learning. München: Pearson, 2012 (siehe S. 112).

[Som12b] Sommestad, Teodor: „A framework and theory for cyber security assessments“. Diss. Stockholm, Sweden: KTH, Royal Institute of Technology, 2012 (siehe S. 84).

[Som13] Sommestad, Teodor; Ekstedt, Mathias and Holm, Hannes: „The Cyber Security Modeling Language: A Tool for Assessing the Vulnerability of Enterprise System Architectures“. In: *IEEE Systems Journal* 7.3 (2013), S. 363–373. doi: 10.1109/JSYST.2012.2221853 (siehe S. 84).

[Sta13] Standardization, International Organization for: Information technology — Security techniques — Information security management systems — Requirements. Standard. Geneva, Schweiz: International Organization for Standardization, Okt. 2013. url: https://www.iso.org/standard/54534.html (besucht am 22. 04. 2021) (siehe S. 30).

[Sta20] Standardization, International Organization for: Automation systems and integration — Digital Twin framework for manufacturing — Part 1: Overview and general principles (Draft).

Standard. Geneva, Schweiz: International Organization for Standardization, 2020. (Besucht am 13. 05. 2021) (siehe S. 25).

[Ste99] Sterling, Leon S. and Shapiro, Ehud Y.: The art of Prolog: Advanced programming techniques. 2nd ed., 3rd printing. Logic programming. Cambridge, Mass: MIT Press, 1999 (siehe S. 89).

[Sye16] Syed, Zareen; Padia, Ankur; Finin, Tim; Mathews, Lisa and Joshi, Anupam: „UCO: A Unified Cybersecurity Ontology". In: *Proceedings of the AAAI Workshop on Artificial Intelligence for Cyber Security* (2016) (siehe S. 81).

[Ton15] Tong, Wencan; Liang, Xiaoyan; Li, Xiaojian; Zhao, Jiejie and Liang, Xuemei: „An analysis method of NAC configuration conflict based on ontology". In: *Proc. of the 3rd International Conference on Digital Enterprise and Information Systems (DEIS2015)*. 2015, S. 46 (siehe S. 144).

[Und03] Undercoffer, Jeffrey; Joshi, Anupam and Pinkston, John: „Modeling Computer Attacks: An Ontology for Intrusion Detection". In: *Recent Advances in Intrusion Detection*. Hrsg. von Goos, Gerhard; Hartmanis, Juris; van Leeuwen, Jan; Vigna, Giovanni; Kruegel, Christopher and Jonsson, Erland. Bd. 2820. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Nature, 2003, S. 113–135. doi: 10.1007/978-3-540-45248-5_7 (siehe S. 37, 71, 93).

[Val15] Valja, Margus; Korman, Matus; Shahzad, Khurram and Johnson, Pontus: „Integrated Metamodel for Security Analysis". In: *48th Hawaii International Conference on System Sciences (HICSS)*. Hrsg. von Bui, Tung X. and Sprague, Ralph H. Piscataway, NJ: IEEE, 2015, S. 5192–5200. doi: 10.1109/HICSS.2015.613 (siehe S. 84).

[Ver08] Vergara, Jorge E. López de; Vázquez, Enrique and Guerra, Javier: „SECURITY POLICY INSTANTIATION TO REACT TO NETWORK ATTACKS - An Ontology-based Approach using

OWL and SWRL". In: *Proceedings of the International Conference on Security and Cryptography - Volume 1: SECRYPT, (ICETE 2008)*. INSTICC. SciTePress, 2008, S. 78–83. DOI: 10.5220/0001929300780083 (siehe S. 155).

[Vol18] Volz, Friedrich: „Knowledge Base für ICS-Security-Lifecycle-Management". Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2018 (siehe S. 184).

[W3C04] W3C: SWRL: A Semantic Web Rule Language: Combining OWL and RuleML. Hrsg. von Horrocks, Ian; Patel-Schneider, Peter F.; Boley, Harold; Tabet, Said; Grosof, Benjamin and Dean, Mike. 2004. URL: https://www.w3.org/Submission/SWRL/ (besucht am 22. 04. 2021) (siehe S. 38).

[W3C12a] W3C: OWL 2 Web Ontology Language Direct Semantics. Hrsg. von Motik, Boris; Patel-Schneider, Peter F. and Cuenca Grau, Bernardo. 2012. URL: https://www.w3.org/TR/owl2-direct-semantics/ (besucht am 22. 04. 2021) (siehe S. 38).

[W3C12b] W3C: OWL 2 Web Ontology Language Primer. Hrsg. von Hitzler, Pascal; Krötzsch, Markus; Parsia, Bijan; Patel-Schneider, Peter F. and Rudolph, Sebastian. 2012. URL: https://www.w3.org/TR/owl2-primer/ (besucht am 22. 04. 2021) (siehe S. 38).

[W3C13] (W3C), World Wide Web Consortium: SPARQL 1.1 Query Language. 2013. URL: https://www.w3.org/TR/sparql11-query/ (besucht am 22. 04. 2021) (siehe S. 47).

[Wan10] Wang, Ju; Guo, Michael M. and Camargo, Jairo: „An Ontological Approach to Computer System Security". In: *Information Security Journal: A Global Perspective* 19.2 (2010), S. 61–73. DOI: 10.1080/19393550903404902 (siehe S. 86, 87).

[Win06] Winter, Robert and Fischer, Ronny: „Essential Layers, Artifacts, and Dependencies of Enterprise Architecture". In: *2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)*. IEEE, Okt. 2006, S. 30. DOI: 10.1109/EDOCW.2006.33 (siehe S. 84).

[Xia06] Xiao, Debao and Xu, Hui: „An Integration of Ontology-based and Policy-based Network Management for Automation“. In: *2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06)*. IEEE, Dez. 2006, S. 27. doi: 10.1109/CIMCA.2006.43 (siehe S. 96).

[Xin14] Xing, Tianyi; Xiong, Zhengyang; Huang, Dijiang and Medhi, Deep: „SDNIPS: Enabling Software-Defined Networking based intrusion prevention system in clouds“. In: *10th International Conference on Network and Service Management (CNSM) and Workshop*. IEEE, Nov. 2014, S. 308–311. doi: 10.1109/CNSM.2014.7014181 (siehe S. 245).

[Ye08] Ye, Dayong; Bai, Quan and Zhang, Minjie: „Ontology-Based Knowledge Representation for a P2P Multi-agent Distributed Intrusion Detection System“. In: *International Conference on Network and Parallel Computing (IFIP 2008)*. Hrsg. von Cao, Jian. Piscataway, NJ: IEEE, 2008, S. 111–118. doi: 10.1109/NPC.2008.8 (siehe S. 94).

[Zhu97] Zhu, Yangyong; Guo, Depei and Shi, Baile: „Techniques of Integrating Datalog with PROLOG“. In: *Journal of Computer Science and Technology* 12.6 (1997), S. 520–531. doi: 10.1007/BF02947204 (siehe S. 89).

Eigene Veröffentlichungen

Dieser Abschnitt enthält ein vollständiges Verzeichnis der eigenen Veröffentlichungen. Im Themenblock Sicherheitsanalyse befassen sich die Publikationen [3], [4], [8] und [9] mit der Informationsextraktion und Modellbildung, wobei [4] und [9] die Sicherheit des Informationstransfers adressieren und damit nicht im Fokus dieser Dissertation stehen. Weiter befasst sich [6] mit der generellen Anwendung von modellbasierter Sicherheitsanalyse auf Konfigurationsdaten industrieller Systeme und [12] mit der Effektiven Konfiguration für NAC-Informationen. Zum Zeitpunkt der Fertigstellung dieser Dissertation wurde ebenfalls an der Fertigstellung eines Artikels zum SyMP-Rahmenwerk gearbeitet.
Der Themenblock Vorfallreaktion wird von den Publikationen [5], [7] und [11] adressiert, wobei es sich bei [5] um den Abschlussbericht des Projektes handelt, in dem die AIR-Lösung entwickelt und im Rahmen einer Gesamtlösung für sichere, flexible Produktionsanlagen evaluiert wurde.
Die Publikationen [1], [2] und [10] befassen sich mit anderen Aspekten der Sicherheitsforschung und können somit nur im weiteren Sinne als relevant für die vorliegende Arbeit angesehen werden.

[1] Patzer, Florian; Jakoby, Andreas; Kresken, Thomas and Müller, Wilmuth: „Protecting sensitive data in a distributed and mobile environment“. In: *International Conference on Cyber Warfare and Security (ICCWS)*. Krüger Nationalpark, Südafrika: Academic Conferences und Publishing International Limited, März 2015, S. 519–524. URL: http://publica.fraunhofer.de/eprints/urn_nbn_de_0011-n-3748315.pdf (besucht am 22. 04. 2021).

[2] Patzer, Florian; Jakoby, Andreas; Kresken, Thomas and Müller, Wilmuth: „Security Overlay for Distributed Encrypted Containers“. In: *International Conference on Security and Management (SAM'15)*. Las Vegas, Nevada, USA, Juli 2015, S. 130–136. URL: http://worldcomp-proceedings.com/proc/p2015/SAM_contents.html (besucht am 22. 04. 2021).

[3] Patzer, Florian; Sarkar, Aranya; Birnstill, Pascal; Schleipen, Miriam and Beyerer, Jürgen: „Towards the Modelling of Complex Communication Networks in AutomationML“. In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sep. 2017, S. 1–8. DOI: 10.1109/ETFA.2017.8247571.

[4] Plattform Industrie 4.0: Diskussionspapier: Sicherer Bezug von CAE-Daten. Hrsg. von Bundesministerium für Wirtschaft und Energie. 2018. URL: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/sicherer-bezug-von-cae-daten.pdf?__blob=publicationFile&v=8 (besucht am 22. 04. 2021).

[5] Patzer, Florian; Kobzan, Thomas and Haas, Christian: FlexSi-Pro : Schlussbericht FlexSi-Pro: Berichtszeitraum: 1.1.2017-31.12.2019. Hrsg. von Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung. online, 2019. URL: https://www.tib.eu/de/suchen/id/TIBKAT:1728069513/FlexSi-Pro-Schlussbericht-FlexSi-Pro-Berichtszeitraum (besucht am 23. 04. 2021).

[6] Patzer, Florian; Meshram, Ankush; Birnstill, Pascal; Haas, Christian and Beyerer, Jürgen: „Towards Computer-Aided Security Life Cycle Management for Critical Industrial Control Systems“. In: *Critical Information Infrastructures Security*. Hrsg. von Luiijf, Eric; Žutautaitė, Inga and Hämmerli, Bernhard M. Bd. 11260. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, S. 45–56. DOI: 10.1007/978-3-030-05849-4_4.

[7] Patzer, Florian; Meshram, Ankush and Hess, Maximilian: „Automated Incident Response for Industrial Control Systems Leveraging

Software-defined Networking“. In: *Proceedings of the 5th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, 2019, S. 319–327. DOI: 10.5220/0007359503190327.

[8] Patzer, Florian; Volz, Friedrich; Usländer, Thomas; Blocher, Immanuel and Beyerer, Jürgen: „The Industrie 4.0 Asset Administration Shell as Information Source for Security Analysis“. In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sep. 2019, S. 420–427. DOI: 10.1109/ETFA.2019.8869059.

[9] Plattform Industrie 4.0: Diskussionspapier: Sichere unternehmensübergreifende Kommunikation mit OPC UA. Hrsg. von Bundesministerium für Wirtschaft und Energie. Berlin, 2019. URL: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/sichere-kommunikation-opc-ua.pdf?__blob=publicationFile&v=13 (besucht am 22. 04. 2021).

[10] Meier, David; Patzer, Florian; Drexler, Matthias and Beyerer, Jürgen: „Portable Trust Anchor for OPC UA Using Auto-Configuration“. In: *Proceedings 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Hrsg. von IEEE. 2020, S. 270–277.

[11] Patzer, Florian; Lüdtke, Philipp; Meshram, Ankush and Beyerer, Jürgen: „Context-Aware Software-Defined Networking for Automated Incident Response in Industrial Networks“. In: *Information Systems Security and Privacy*. Hrsg. von Mori, Paolo; Furnell, Steven and Camp, Olivier. Bd. 1221. Communications in Computer and Information Science. Cham: Springer International Publishing, 2020, S. 137–161. DOI: 10.1007/978-3-030-49443-8_7.

[12] Patzer, Florian and Beyerer, Jürgen: „Efficient Semantic Representation of Network Access Control Configuration for Ontology-Based Security Analysis“. In: *Proceedings of the 7th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*. SciTePress, 2021, S. 550–557.

Betreute studentische Arbeiten

Dieser Abschnitt enthält die während der Dissertation betreuten studentischen Arbeiten. Davon befassten sich die Arbeiten [3], [5], [6], [7], [8], [9], [11], [12] und [13] mit Konzepten und Methoden, die im Fokus dieser Dissertation liegen. Weiter adressieren [2] und [4] die Sicherheit für OPC UA und damit auch für die Informationsextraktion und die Arbeiten [1] und [10] sind nur aufgrund ihres Fokus auf die Sicherheit industrieller Systeme im weiteren Sinne relevant.

[1] Dillmann, Jörg: „Usage Control im Industrial Data Space“. Bachelorarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2017 (siehe S. 319).

[2] Goerke, Niklas: „Security for Industrial Control Systems Based on Trust Anchors“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2017 (siehe S. 319).

[3] Sarkar, Aranya: „Standardized Device Description in AutomationML with OPC UA Configurations and IT Security Mechanisms“. Masterarbeit. Karlsruhe: Otto von Güricke Universität, 2017 (siehe S. 319).

[4] Drexler, Matthias: „OPC UA-Vertrauensaufbau mittels mobilem Vertrauensanker“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2018 (siehe S. 319).

[5] Hess, Maximilian: „Regelbasiertes Incident Handling für Software-defined Networks in industriellen Umgebungen“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2018 (siehe S. 319).

[6] Volz, Friedrich: „Knowledge Base für ICS-Security-Lifecycle-Management“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2018 (siehe S. 319).

[7] Kahles, David: „Automatisierte Analyse der Umsetzung von Security-Best-Practices und -Standards in industriellen Systemen“. Bachelorarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2019 (siehe S. 319).

[8] Betten, Niklas: „Rule Management for Complex Ontology Manipulation“. Bachelorarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2020 (siehe S. 319).

[9] Hardt, Henrike: „Optimierung und Generalisierung Ontologiebasierter ICS Security Analyse“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2020 (siehe S. 319).

[10] Kellner, Kevin: „Security Analysis of the Industry 4.0 middleware BaSys“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2020 (siehe S. 319).

[11] Kahles, David: „Threat Classification in Industrial Networks Based on Heterogeneous Information“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2021 (siehe S. 319).

[12] Kloppenburg, Manuel: „Flexible, wissensbasierte Analyseplattform für die Identifikation von Sicherheitsproblemen industrieller Systeme“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2021 (siehe S. 319).

[13] Roehr, Marco: „Erweiterung von Systemontologien um Wissen der Informationssicherheit: Konzeption und Implementierung eines Werkzeuges am Beispiel der Normenreihe IEC 62443“. Masterarbeit. Karlsruhe: Karlsruher Institut für Technologie, 2021 (siehe S. 319).

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

AIR	Automated Incident Response
BMBF	Bundesministerium für Bildung und Forschung
BMWi	Bundesministerium für Wirtschaft und Energie
BPMN	Business Process Model and Notation
BSI	Bundesamt für Sicherheit in der Informationstechnik
CAEX	Computer Aided Engineering Exchange
CAPEC	Common Attack Pattern Enumeration and Classification
CLI	Command Line Interface
CPE	Common Platform Enumeration
CPS	Cyber-physisches System
CVE	Common Vulnerabilities and Exposures
CWA	Closed World Assumption
CySeMoL	Cyber Security Modelling Language

DHCP	Dynamic Host Configuration Protocol
DL	Description Logic
DNS	Domain Name System
ENISA	European Union Agency for Network and Information Security
ERP	Enterprise Resource Planning
HMI	Human Machine Interface
HTTP	Hypertext Transfere Protocol
HTTPS	Hypertext Transfere Protocol Secure
IC	Interface Class
ICS	Industrial Control Systems
IDS	Intrusion Detection Systems
IE	Internal Element
IED	Intelligent Electronic Device
IH	Instance Hierarchy
IIoT	Industrial Internet of Things
IOSB	Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung
IoT	Internet of Things

IPS	Intrusion Prevention Systems
IRI	Internationalized Resource Identifier
IT	Information Technology
KIT	Karlsruher Institut für Technologie
KOS	Knowledge Organization System
MD-SAL	Model-driven Service Abstraction Layer
MulVAL	Multihost, multistage Vulnerability Analysis
NAC	Network Access Control
NAT	Network Address Translation
NFV	Network Function Virtualization
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
NTP	Network Time Protocol
ODL	OpenDaylight
OPC UA	Open Platform Communications Unified Architecture
OT	Operational Technology
OWA	Open World Assumption
OWL	Web Ontology Language

PBNM	Policy-Based Network Management
PLC	Programmable Logic Controller
PoC	Proof-of-Concept-Umgebung
PRM	Probabilistic Relation Model
RC	Role Class
RDF	Resource Description Framework
SAP	Service Access Point
SCM	Security Configuration Management
SDE	Security Decision Engine
SDN	Software-Defined Networking
SDN4S	Software-Defined Networking for Security
SELinux	Security Enhanced Linux
SIEM	Security Information and Event Management
SKOS	Simple Knowledge Organization System
SMB	Server Message Block
SMS	Sicherheitsmanagementsystem
SPARQL	SPARQL Protocol and RDF Query Language
SPS	Speicherprogrammierbare Steuerung

SQWRL	Semantic Query-enhanced Web Rule Language
SUC	System Unit Class
SWRL	Semantic Web Rule Language
SyMP	System Model Processing
SysML	Systems Modelling Language
UML	Unified Modelling Language
VLAN	Virtual Local Area Network
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

Glossar

ABox Hier: Der Teil einer Ontologie, welcher das assertionale Instanzwissen, also Aussagen über Instanzen.

Analyseregelformulierung Natürlichsprachliche Formulierung einer technischen Richtlinie.

Analyseregelinstanz Eine Kombination aus einer Analyseregelformulierung und einer Analyseregelumsetzung.

Analyseregelumsetzung Kombination aus verschiedenen voneinander abhängigen Modellerweiterungsschritten und Ontologieanforderungen, die zur automatisierten Ableitung von Wissen dienen, das abgefragt werden kann, um eine Analyseregel zu prüfen.

Asset Hardware, Software, Daten o.ä. die für ein Unternehmen von Wert sind.

Axiom Eine Grundsatzaussage, die als Grundlage logischer Ableitung in einem Kalkül dient.

Closed World Assumption Eine Annahme, welche besagt, dass eine Aussage falsch ist, wenn sie nicht Teil der Wissensbasis ist oder aus ihr abgeleitet werden kann.

Companion Specs Von Gremien definierte Spezifikationen von OPC-UA-Informationsmodellen bestimmter Domänen.

Conduit In IEC 62443 spezifizierte, spezielle Sicherheitszonen, die hauptsächlich zur Kategorisierung von Zonenübergängen eingesetzt werden.

Cyber-physische Systeme Verbund von informationstechnologischen mit mechanischen oder elektrischen Komponenten.

Description Logic Englischer, geläufiger Ausdruck für Beschreibungslogik.

Entscheidbarkeit Eine beschreibungslogische Sprache ist entscheidbar, wenn es einen Ableitungsalgorithmus gibt, der zu jedem Ableitungsproblem (Inferenzproblem) immer terminiert.

Flow Eine Weiterleitungsregel im Rahmen von SDN.

Hypertext Transfere Protocol Ein Protokol für verteilte, verknüpfte, multimediale Informationssysteme (vgl. RFC 2616[1]).

Hypertext Transfere Protocol Secure Die Anwendung von Hypertext Transfere Protocol (HTTP) über eine authentifizierte und verschlüsselte Verbindung.

ICS ATT&CK Kurzform für das Mitre ATT&CK®-Framework for ICS, welches eine unter Experten konsolidierte, öffentliche Wissensbasis für bekannte Angriffstaktiken und -techniken darstellt.

[1] https://tools.ietf.org/html/rfc2616, zuletzt zugegriffen: 02.11.2020.

Industrial Internet of Things Die Anwendung von IoT-Technologie in industriellen Umgebungen.

Industrie 4.0 Die vierte industrielle Revolution, die sich insbesondere auf die zunehmende Vernetzung von Maschinen bezieht.

Internet of Things Technologien die zur Vernetzung von nahezu beliebiger Gegenstände eingesetzt werden, sodass diese miteinander kommunizieren und Funktionen bereitstellen bzw. ausführen können.

NIST 800-82 Der weitläufig bekannte Best-Practice-Leitfaden "'Guide to Industrial Control Systems (ICS) Security"' des US-amerikanischen National Institute of Standards and Technology (NIST).

Ontologie Explizite Spezifikation einer Konzeptualisierung einer Wissensdomäne.

Open World Assumption Eine Annahme, welche besagt, dass eine Aussage wahr sein kann, obwohl sie nicht Teil der Wissensbasis ist, solange sie nicht dem Wissen in der Wissensbasis widerspricht.

Patchen Beheben von Softwarefehlern.

Reasoner Software zum Ableiten semantischer Aussagen von Fakten oder Axiomen.

Security-by-Design Entwurfsparadigma bei dem Sicherheitsaspekte des angestrebten Ergebnisses (z.B. des Produktes) bereits in der Entwurfsphase berücksichtigt werden.

Selbstauskunft Strategie, bei der eine Software auf einem Gerät Auskunft über dessen Status, Funktionen, Konfigurationen o.ä. mithilfe mindestens einer dedizierten Schnittstelle preisgibt.

Separation-of-Concerns Entwurfsprinzip u.a. für Modelle und Programme, bei dem diese in separate, in sich abgeschlossene Bereiche unterteilt werden, die jeweils unterschiedliche Anliegen adressieren.

TBox Hier: Der Teil einer Ontologie, welcher das terminologische Schemawissen enthält, also Klassen und Beziehungen der Wissensdomäne.

Workflow Eine definierte, ganz oder teilweise programmatisch ausführbare Abfolge von Aufgaben, die von programmatischen oder menschlichen Arbeitern ausgeführt werden sollen.

Workflow-Engine Programm zum Steuern, Verwalten und Ausführen von Workflows.

Zone In IEC 62443 spezifizierter logischer Sicherheitsbereich, für den im Rahmen eines Sicherheismanagementsystems u.a. Sicherheitsziele definiert werden, die sich dann auf alle enthaltenen Assets beziehen.

A Listings

In diesem Abschnitt werden Listings zu den X2Owl- und SDN-AIR-Implementierungen aufgeführt. Außerdem enhält der Abschnitt den Pseudo-Code für die Erzeugung der Effektiven Konfiguration (vgl. Abschnitt 3.4.2.1) und dessen Erörterung, sowie einen Ausschnitt der CVE-Repräsentation, die der CVE-Labelling-Arbeiter nutzt, welcher in Abschnitt 3.7.4 beschrieben wurde.

A.1 X2Owl

Für die komplexe Abbildung von OPC-UA-Informationsmodellen oder CLI-Ausgaben auf OWL wurde das Python-Paket *X2Owl*[1] entwickelt. Die Abbildung wird dabei über ein YAML-Konfigurationsschema definiert. Listing A.1 zeigt die Verwendung des Schemas zur Konfiguration einer OPC-UA-OWL-Abbildung. Konfigurationszeilen mit dem `ua`-Präfix beziehen sich auf das OPC-UA-Informationsmodell, Zeilen mit dem `owl`-Präfix auf die Ontologiegenerierung. Andere Zeilen dienen der Erzeugung des programminternen Objektbaums, der auf die entsprechenden OWL-Konzepte abgebildet wird, und dem Parsen der OPC-UA-Knotenwerte.

```
ua_endpoint: 'opc.tcp://localhost:4840/'
ua_namespaces:
    - 'https://www.flexsi-pro.de/UA/'
    - 'urn:unconfigured:application'
    - 'http://opcfoundation.org/UA/'
```

[1] https://github.com/FlorianPatzer/x2owl.

```
ua_target_namespace: 'urn:unconfigured:application'
ua_root_object:
    ns: 2
    i: '85'

commands_from_file: false
commands:
    -
        command: 'dpkg-query -l'
        object: 'software_inventory'
        regex: '^nis_pattern'
        default_value: 'inventory'
        ua_object_type:
            ns: 'https://www.flexsi-pro.de/UA/'
            i: '10001'
        iterate_subtree_mapping: true
        objects:
            -
                object: 'software_package'
                owl_class: 'Software'
                owl_objectProperty: 'installedSoftware'
                owl_objectProperty_domain_object: '
                    software_inventory'
                owl_class_parent: 'Configuration'
                regex: '\nii +(.*?) +'
                regex_rewind: true
                ua_object_type:
                    ns: 'https://www.flexsi-pro.de/UA/'
                    i: '10002'
                objects:
                    -
                        object: 'name'
                        owl_dataProperty: 'hasName'
```

```
                owl_dataProperty_parent: 'owl:
                    topDataProperty'
                owl_dataProperty_domain_object: '
                    software_package'
                regex: '\nii +(.*?)\s+'
                ua_variable_qualified_name:
                    ns: 'https://www.flexsi-pro.de/UA/'
                    browse_name: 'Name'
            -
                object: 'version'
                owl_dataProperty: 'version'
                owl_dataProperty_parent: 'owl:
                    topDataProperty'
                owl_dataProperty_domain_object: '
                    software_package'
                regex: '\s*(.*?)\s'
                ua_variable_qualified_name:
                    ns: 'https://www.flexsi-pro.de/UA/'
                    browse_name: 'Version'
            -
                object: 'architecture'
                owl_dataProperty: 'architecture'
                owl_dataProperty_parent: 'owl:
                    topDataProperty'
                owl_dataProperty_domain_object: '
                    software_package'
                regex: '\s*(.*?)\s'
                ua_variable_qualified_name:
                    ns: 'https://www.flexsi-pro.de/UA/'
                    browse_name: 'Architecture'
```

Listing A.1: Konfigurationsbeispiel für X2Owl-Konfiguration zur Abbildung von Software-Inventaren.

Listing A.2 zeigt ein Beispiel für die Konfiguration und Anwendung von Vor- und Nachbearbeitungsroutinen, die in X2Owl eingesetzt werden können. Dort wird eine Nachbearbeitungsroutine (vgl. `owl_postprocessing_functions`) definiert, die nach der Erstellung der IPv4-Adresse und -Netzmaske die entsprechenden Netzwerke auf Basis dieser Werte erzeugt[1]. Zudem werden zwei Vorbearbeitungsroutinen (vgl. `owl_preprocessing_function`) definiert, die hauptsächlich der Trennung von IP-Adresse und Bitmaske dienen.

```
object: 'ip_interface'
owl_class: 'IpV4Interface'
owl_class_parent: 'Interface'
owl_objectProperty: 'interface'
owl_iri_suffix: 'IpV4Interface'
regex: '(\S+?):\s+flags'
regex_rewind: true
owl_postprocessing_functions:
   - 'addIpV4Networks'
objects:
   -
       object: 'name'
       owl_dataProperty: 'hasName'
       owl_dataProperty_parent: 'owl:topDataProperty'
       owl_dataProperty_domain_object: 'ip_interface'
       regex: '(\S+?):\s+flags'
   -
       object: 'ip'
       owl_dataProperty: 'ipV4Address'
       owl_dataProperty_parent: 'owl:topDataProperty'
       owl_dataProperty_domain_object: 'ip_interface'
       owl_preprocessing_function: 'ipV4ToInteger'
       regex: 'inet ([0-9]+?\.[0-9]+?\.[0-9]+?\.[0-9]+)'
```

[1] Dieser Schritt wird durch die Beschreibung zur Erzeugung der Effektiven Konfiguration in Abschnitt 3.4.2.1 weiter erläutert.

```
-
    object: 'netmask'
    owl_dataProperty: 'prefixBits'
    owl_dataProperty_parent: 'owl:topDataProperty'
    owl_dataProperty_domain_object: 'ip_interface'
    data_type: 'int'
    owl_preprocessing_function: 'ipV4NetmaskToInteger'
    regex: 'netmask ([0-9]+?\.[0-9]+?\.[0-9]+?\.[0-9]+)'
```

Listing A.2: Konfigurationsbeispiel für X2Owl-Konfiguration zur komplexen Abbildung von IP-Konfigurationen mit Vor- und Nachbearbeitungsroutinen.

A.2 SDN-AIR

In *[Heß18]* wurde die Beschränkung verschiedener Reaktionen anhand unterschiedlicher Netzwerktopologien, Asset-Klassifikationen und Vorfallreaktionen getestet. Ein Teil dieser Tests wurde auch als Unit-Tests und Testskripte in das Software-Projekt übernommen. Ein Ausschnitt der Tests ist in Listing A.3 zu finden. Der Ausschnitt zeigt drei verschiedene Reaktionstests, für jeweils mehrere Asset-Klassifikationen von Links, die als Quell-Ziel-Paare konfiguriert werden. Dabei wird geprüft, ob die Kernkomponente der Restriktionseinschränkung (der *Feasibility Analyser*) die Reaktion erlaubt oder verbietet. Den Tests wird die Topologie aus Abbildung 4.6 zugrunde gelegt, wobei für die Testreihe anfangs spezifische Flows konfiguriert werden.

```
private static Flow buildFlow(final MacAddress source,
        final MacAddress destination) {
    return new FlowBuilder().setMatch(
        new MatchBuilder()
            .setEthernetMatch(
                new EthernetMatchBuilder()
                    .setEthernetSource(
                        new EthernetSourceBuilder()
                            .setAddress(source).build()
                    )
                .setEthernetDestination(
                    new EthernetDestinationBuilder()
                        .setAddress(destination).build()
                ).build()
            ).build()
    ).build();
}

private static List<Flow> generateTestFlows() {
    final List<Flow> result = new ArrayList<Flow>();
```

```
    result.add(buildFlow(new MacAddress("00:00:00:00:00:01"),
        new MacAddress("00:00:00:00:00:02")));
    result.add(buildFlow(new MacAddress("00:00:00:00:00:02"),
        new MacAddress("00:00:00:00:00:03")));
    result.add(buildFlow(new MacAddress("00:00:00:00:00:03"),
        new MacAddress("00:00:00:00:00:04")));
    result.add(buildFlow(new MacAddress("00:00:00:00:00:04"),
        new MacAddress("00:00:00:00:00:05")));
    return result;
}

private static YangHelper mockYangHelper() {
    final YangHelper yangHelper = mock(YangHelper.class);
    when(yangHelper.getHost(new MacAddress("00:00:00:00:00:01")
        )).thenReturn(new HostBuilder().setId("host1").setMac(
        new MacAddress("00:00:00:00:00:01")).build());
    when(yangHelper.getHost(new MacAddress("00:00:00:00:00:02")
        )).thenReturn(new HostBuilder().setId("host2").setMac(
        new MacAddress("00:00:00:00:00:02")).build());
    when(yangHelper.getHost(new MacAddress("00:00:00:00:00:03")
        )).thenReturn(new HostBuilder().setId("host3").setMac(
        new MacAddress("00:00:00:00:00:03")).build());
    when(yangHelper.getHost(new MacAddress("00:00:00:00:00:04")
        )).thenReturn(new HostBuilder().setId("host4").setMac(
        new MacAddress("00:00:00:00:00:04")).build());
    when(yangHelper.getHost(new MacAddress("00:00:00:00:00:05")
        )).thenReturn(new HostBuilder().setId("host5").setMac(
        new MacAddress("00:00:00:00:00:05")).build());
    return yangHelper;
}

private static Constraint buildConstraint(final int id, final
    String sourceId, final String destinationId,
        final List<Limitation> limitations) {
```

```
    return new ConstraintBuilder()
            .setId(id)
            .setSourceId(sourceId)
            .setDestinationId(destinationId)
            .setLimitations(limitations)
            .build();
}

@Test
public void
    testPossibleToMonitorSwitchWithDoNotRedirectConstraint() {
    final List<Flow> flows = generateTestFlows();
    final List<Constraint> constraints = new ArrayList<
        Constraint>();
    final YangHelper yangHelper = mockYangHelper();

    constraints.add(new ConstraintBuilder().setSourceId("host1"
        ).setDestinationId("host2").setLimitations(Arrays.
        asList(Limitation.DONOTREMOVEPATH)).build());
    assertTrue(FeasibilityAnalyzer.possibleToMonitorSwitch(
        flows, constraints, yangHelper));

    constraints.add(new ConstraintBuilder().setSourceId("host5"
        ).setDestinationId("host2").setLimitations(Arrays.
        asList(Limitation.DONOTREDIRECTPATH)).build());
    assertTrue(FeasibilityAnalyzer.possibleToMonitorSwitch(
        flows, constraints, yangHelper));

    constraints.add(new ConstraintBuilder().setSourceId("host2"
        ).setDestinationId("host1").setLimitations(Arrays.
        asList(Limitation.DONOTREDIRECTPATH)).build());
    assertFalse(FeasibilityAnalyzer.possibleToMonitorSwitch(
        flows, constraints, yangHelper));
```

```
}

@Test
public void
    testPossibleToMonitorSwitchWithDoNotMonitorConstraint() {
    final List<Flow> flows = generateTestFlows();
    final List<Constraint> constraints = new ArrayList<
        Constraint>();
    final YangHelper yangHelper = mockYangHelper();

    constraints.add(new ConstraintBuilder().setSourceId("host1"
        ).setDestinationId("host2").setLimitations(Arrays.
        asList(Limitation.DONOTREMOVEPATH)).build());
    assertTrue(FeasibilityAnalyzer.possibleToMonitorSwitch(
        flows, constraints, yangHelper));

    constraints.add(new ConstraintBuilder().setSourceId("host5"
        ).setDestinationId("host2").setLimitations(Arrays.
        asList(Limitation.DONOTMONITORPATH)).build());
    assertTrue(FeasibilityAnalyzer.possibleToMonitorSwitch(
        flows, constraints, yangHelper));

    constraints.add(new ConstraintBuilder().setSourceId("host2"
        ).setDestinationId("host1").setLimitations(Arrays.
        asList(Limitation.DONOTMONITORPATH)).build());
    assertFalse(FeasibilityAnalyzer.possibleToMonitorSwitch(
        flows, constraints, yangHelper));
}

@Test
public void testPossibleToBlockSwitch() {
    final List<Flow> flows = generateTestFlows();
    final List<Constraint> constraints = new ArrayList<
        Constraint>();
```

```
    final YangHelper yangHelper = mockYangHelper();

    constraints.add(new ConstraintBuilder().setSourceId("host1"
        ).setDestinationId("host2").setLimitations(Arrays.
        asList(Limitation.DONOTMONITORPATH)).build());
    assertTrue(FeasibilityAnalyzer.possibleToBlockSwitch(flows,
         constraints, yangHelper));

    constraints.add(new ConstraintBuilder().setSourceId("host1"
        ).setDestinationId("host5").setLimitations(Arrays.
        asList(Limitation.DONOTREMOVEPATH)).build());
    assertTrue(FeasibilityAnalyzer.possibleToBlockSwitch(flows,
         constraints, yangHelper));

    constraints.add(new ConstraintBuilder().setSourceId("host3"
        ).setDestinationId("host4").setLimitations(Arrays.
        asList(Limitation.DONOTREMOVEPATH)).build());
    assertTrue(FeasibilityAnalyzer.possibleToBlockSwitch(flows,
         constraints, yangHelper));

    constraints.add(new ConstraintBuilder().setSourceId("host4"
        ).setDestinationId("host3").setLimitations(Arrays.
        asList(Limitation.DONOTREDIRECTPATH)).build());
    assertFalse(FeasibilityAnalyzer.possibleToBlockSwitch(flows
        , constraints, yangHelper));
}
```

Listing A.3: Ausschnitt aus den Java-Unit-Tests der ersten SDN-AIR-Implementierung.

Im Rahmen von FlexSi-Pro wurde mit SDN-AIR außerdem, das in Abschnitt 4.6.3 beschriebe stufenbasierte Sicherheitsmanagement umgesetzt. Die Konfiguration dieser Stufen ist in Listing A.4 zu sehen.

```
#!/bin/bash

CREDENTIALS="admin:admin"
curl -H 'Content-Type: application/json' -X POST -d '
{
  "securityproperties:input": {
    "securityproperties:name": "Unknown",
    "securityproperties:successor-states": ["1"],
    "securityproperties:RESTcontentType": "IP",
    "securityproperties:RESTcredentials": "admin:admin",
    "securityproperties:RESTdestination": ""
  }
}
' \
-u $CREDENTIALS 'http://localhost:8181/restconf/operations/
    securityproperties:add-state'

curl -H 'Content-Type: application/json' -X POST -d '
{
  "securityproperties:input": {
    "securityproperties:name": "Unauthenticated",
    "securityproperties:successor-states": ["2", "5"],
    "securityproperties:RESTcontentType": "IP",
    "securityproperties:RESTcredentials": "admin:admin",
    "securityproperties:RESTdestination": "http://localhost
        :6213/restconf/operations/woit:authenticate_device"
  }
}
' \
 -u $CREDENTIALS 'http://localhost:8181/restconf/operations/
     securityproperties:add-state'

curl -H 'Content-Type: application/json' -X POST -d '
```

```
{
  "securityproperties:input": {
    "securityproperties:name": "Authenticated",
    "securityproperties:successor-states": ["3", "5"],
    "securityproperties:RESTcontentType": "IP",
    "securityproperties:RESTcredentials": "admin:admin",
    "securityproperties:RESTdestination": "http://localhost
        :8889/cve"
  }
}
' \
 -u $CREDENTIALS 'http://localhost:8181/restconf/operations/
     securityproperties:add-state'

curl -H 'Content-Type: application/json' -X POST -d '
{
  "securityproperties:input": {
    "securityproperties:name": "Compliant",
    "securityproperties:successor-states": ["4"],
    "securityproperties:RESTcontentType": "IP",
    "securityproperties:RESTcredentials": "admin:admin",
    "securityproperties:RESTdestination": ""
  }
}
' \
 -u $CREDENTIALS 'http://localhost:8181/restconf/operations/
     securityproperties:add-state'

curl -H 'Content-Type: application/json' -X POST -d '
{
  "securityproperties:input": {
    "securityproperties:name": "Incident-Analysis",
    "securityproperties:successor-states": ["3", "5"],
    "securityproperties:RESTcontentType": "IP",
```

```
    "securityproperties:RESTcredentials": "admin:admin",
    "securityproperties:RESTdestination": "MELDUNGSPLATZHALTER"
  }
}
' \
 -u $CREDENTIALS 'http://localhost:8181/restconf/operations/
     securityproperties:add-state'

curl -H 'Content-Type: application/json' -X POST -d '
{
  "securityproperties:input": {
    "securityproperties:name": "Incident-Containment",
    "securityproperties:successor-states": ["1"],
    "securityproperties:RESTcontentType": "IP",
    "securityproperties:RESTcredentials": "admin:admin",
    "securityproperties:RESTdestination": "MELDUNGSPLATZHALTER"
  }
}
' \
 -u $CREDENTIALS 'http://localhost:8181/restconf/operations/
     securityproperties:add-state'
```

Listing A.4: Konfiguration der Sicherheitsstufen für den FlexSi-Pro Demonstrator.

A.3 Pseudocode zur Effektiven Konfiguration

Listing A.5 enthält den Pseudocode der Strategie zur Erzeugung und Modellierung der Effektiven Konfiguration. Hierfür gilt folgendes:

- Λ enthält die Regelkette, also die Hauptkonfiguration der NAC-Instanz. Dabei wird davon ausgegangen, dass eventuelle Substitutionen (oder ähnliche Abstraktionen der eigentlichen Konfiguration) bereits aufgelöst wurden oder als Teil der Routine *integrate_additional_config* in die Regelkette transformiert werden, sodass Λ spätestens nach dieser Funktion nur noch aus der vorbereiteten Regelkette besteht.
- ***transform_action_types*** nimmt die Transformation der Aktionen der Regeln zu „erlauben“ und „blockieren“ vor. Diese Reduktion ist sinnvoll, da später nicht mehr das genaue Verhalten der NAC-Instanz relevant ist, sondern nur ob ein bestimmter Netzwerkverkehr die Instanz passieren darf oder nicht. Als Beispiel nehme man zwei ICMP-blockierende Aktionen, wobei die eine beim Empfang des ICMP-Pakets eine Antwort sendet und die andere nicht. Diese würden beide durch die Aktion „blockieren“ ersetzt werden.
- Σ ist die Menge der Auswertungsstrategien zu Λ.
- $\Lambda_s \subset \Lambda$ ist die Teil-Regelkette einer Auswertungsstrategie $s \in \Sigma$.
- „**integrate** Λ'_s **into** Λ“ bedeutet, dass die veränderte Teil-Regelkette Λ'_s die originale Teil-Regelkette Λ_s in Λ so ersetzt, dass sich diese in die korrekte Gesamtreihenfolge eingliedert. Diese Umordnung ist wichtig, da sich die Auswertungsstrategie von Λ'_s gegenüber Λ_s geändert hat. Ein Beispiel ist in *[Pat21]* beschrieben.
- Π enthält sowohl den Erlauben-Behälter, als auch den Blockieren-Behälter (vgl. Abschnitt 3.4.2.1). Im Pseudocode werden diese nicht getrennt, da dies durch die zusätzlichen Fallunterscheidungen zu einer deutlichen Ausweitung des Algorithmus mit großer Redundanz führen würde. Stattdessen werden

die Muster-verarbeitenden Funktionen als Aktionstyp-bewusst definiert.

- ***get_overlapping_patterns*** gibt die Muster aus Π zurück, die sich mit dem übergebenen Muster überschneiden.
- ***split_expand_reduce*** erzeugt so viele Teilmuster, wie nötig sind, um die entsprechenden übergebenen Muster darzustellen, ohne Widersprüche bezüglich der zugehörigen Aktionen zu enthalten. Wenn sich ein Muster p_r einer Regel r mit einem Muster $p \in \Pi$ überschneidet, können die folgenden Fälle auftreten:
  1. p überschneidet sich teilweise mit p_r und die zu p gehörenden Regeln haben eine andere Aktion als r.
  2. p überschneidet sich teilweise mit p_r und die zu p gehörenden Regeln haben die gleiche Aktion wie r.

  Im ersten Fall wird p in die Teile zerlegt, die nicht von p_r abgedeckt werden und in die Teile, die von p_r abgedeckt werden. Die ersten Teile werden als Ersatz für p in Π eingefügt, während die zweiten Teile durch p_r ersetzt werden. Im zweiten Fall werden p und p_r zusammengeführt. Muster, welche sich mit anderen Mustern überschneiden, kommen in jeder NAC-Konfiguration vor, da sich jede Regel mit den Standardregeln, wie „alles blockieren“ oder „alles zulassen“, überschneidet.
- ***create_flow_concepts*** erzeugt die noch nicht im Modell existierenden Flow-Konzepte. Dies sind Klassen und Rollen, welche pro Kombination aus Aktion, Identifikator-Typ, ISO/OSI-Schicht und Protokoll verwendet werden können, um das jeweilige Teilmuster inklusive seiner Beziehungen zu anderen Teilmustern und den Ursprungsregeln (falls auch Teil des Modells) im Modell zu repräsentieren. Ein Beispiel hierfür zeigt Abbildung 3.27.
- Υ ist die geordnete Liste der Flow-Konzepttypen (vgl. AllowedIpV4Flow und AllowedTcpFlow in Abbildung 3.27). Die Reihenfolge der Elemente in der Liste ist aufsteigend mit Bezug auf die betroffene ISO/OSI-Schicht.

- ***add_to_ontology*** fügt die übergebenen Informationen des Musters unter Verwendung der Flow-Konzepte des entsprechenden Flow-Konzepttyps zum Modell hinzu. In diesem Schritt werden auch die bereits hinzugefügten Informationen desselben Musters mit den neu hinzugefügten Informationen verknüpft (vgl. *usesFlow* aus Abbildung 3.27). Für den Fall der erlaubten Aktionen spiegelt dies wider, dass jedes untergeordnete Konzept von AllowedFlow (z.B. AllowedEthernetFlow, AllowedIpV4Flow und AllowedTcpFlow) entweder ein in sich geschlossenes Muster oder eine Verfeinerung eines anderen AllowedFlow ist.

```
integrate_additional_config(Λ)
transform_action_types(Λ)
for each s ∈ Σ {
    Λ'_s = transform_to_last_match_wins(Λ_s)
    integrate Λ'_s into Λ
}
for each rule r ∈ Λ {
    p_r = get_pattern(r)
    P = get_overlapping_patterns(p_r)
    if P ≠ ∅ {
        P' = split_expand_reduce(P, p_r)
        remove P from Π
        add P' to Π
    } else {
        add p_r to Π
    }
}
ϒ = create_flow_concepts(Π)
for each p ∈ Π {
    for each t ∈ ϒ {
        if p contains t-specific information i_t {
            add_to_ontology(i_t, p)
        }
    }
}
```

Listing A.5: Pseudocode, der die Strategie zur Erzeugung und Modellierung der Effektiven Konfiguration beschreibt.

A.4 NVD CVE Repräsentation

Listing A.6 enthält einen Ausschnitt der CVE-Repräsentation, die der CVE-Labelling-Arbeiter nutzt, welcher in Abschnitt 3.7.4 beschrieben wurde. Die Repräsentation entspricht der Originalserialisierung der CVEs aus der National Vulnerability Database der NIST.

```
"cve" : {
      "data_type" : "CVE",
      "data_format" : "MITRE",
      "data_version" : "4.0",
      "CVE_data_meta" : {
        "ID" : "CVE-2020-0785",
        "ASSIGNER" : "cve@mitre.org"
      },
      "problemtype" : {
        "problemtype_data" : [ {
          "description" : [ {
            "lang" : "en",
            "value" : "CWE-269"
          } ]
        } ]
      },
      "references" : {
        "reference_data" : [ {
          "url" : "https://portal.msrc.microsoft.com/en-US/
              security-guidance/advisory/CVE-2020-0785",
          "name" : "https://portal.msrc.microsoft.com/en-US/
              security-guidance/advisory/CVE-2020-0785",
          "refsource" : "MISC",
          "tags" : [ "Patch", "Vendor Advisory" ]
        } ]
      },
      "description" : {
```

```
      "description_data" : [ {
        "lang" : "en",
        "value" : "An elevation of privilege vulnerability
            exists when the Windows User Profile Service (
            ProfSvc) improperly handles symlinks, aka '
            Windows User Profile Service Elevation of
            Privilege Vulnerability'."
      } ]
    }
  },
  "configurations" : {
    "CVE_data_version" : "4.0",
    "nodes" : [ {
      "operator" : "OR",
      "cpe_match" : [ {
        "vulnerable" : true,
        "cpe23Uri" : "cpe:2.3:o:microsoft:windows_10
            :-:*:*:*:*:*:*:*"
      }, {
        "vulnerable" : true,
        "cpe23Uri" : "cpe:2.3:o:microsoft:windows_10
            :1607:*:*:*:*:*:*:*"
      }, {
        "vulnerable" : true,
        "cpe23Uri" : "cpe:2.3:o:microsoft:windows_10
            :1709:*:*:*:*:*:*:*"
      },
      .
      .
      .
      {
        "vulnerable" : true,
        "cpe23Uri" : "cpe:2.3:o:microsoft:windows_server_2019
            :-:*:*:*:*:*:*:*"
```

```
        } ]
      } ]
    },
```

Listing A.6: Ausschnitt eines, im JSON-Format dargestellten, CVEs aus der NIST National Vulnerability Database (NVD).