WWU
MÜNSTER

Lokale Intrusion Prevention über eine simulations-basierte Validierung von Befehlen im Kontext von Smart Grids

Bachelorarbeit

Vorgelegt von:	Daniel Krug Marievengasse 9 48143 Münster
Matrikelnummer:	461791
Studiengang:	B.Sc. Informatik
Erstprüfer:	Prof. Anne Remke
Zweitprüfer:	Dr. Dietmar Lammers
Mit betreut durch:	Verena Menzel
Abgabetermin:	06.03.2023

wissen.leben

I Inhaltsverzeichnis

II Abbildungsverzeichnis

III Quelltextverzeichnis

1 Einführung

1.1. Motivation

Wir alle nutzen jeden Tag Strom. Unser Herd wird zumeist mit Strom betrieben. In Krankenhäusern werden Menschen durch Strom am Leben erhalten. Aber es gibt auch weniger offensichtliche Beispiele. Klär- und Wasserwerke werden mit Strom betrieben, diese sind für unsere Wasserversorgung und somit für die Toilettenspülung oder auch für die Dusche zuständig. Auch Gasanbieter benötigen Strom, um Haushalte mit Gas zu versorgen. Aus diesen Gründen zählt unser Stromnetz zur kritischen Infrastruktur. Sollte es also zu einem großflächigen Stromausfall kommen, kann weder geheizt oder gekocht werden noch ist Leitungswasser verfügbar. Damit es nicht zu genau diesen Notlagen kommt, werden Simulationen von Stromnetzen entwickelt. Mit den Simulationen können physische Stromnetze virtuell abgebildet werden, um die Reaktion des Netzes dann auf verschiedene Situationen zu testen. Tests sind an realen Netzen nicht umsetzbar, da es einerseits aus genannten Gründen zu gefährlich oder andererseits eine physische Nachbildung oft zu teuer und aufwendig wäre. Ein weiterer Aspekt, in dem Simulationen hilfreich sein können, ist die Cyber Security. Diese ist von Relevanz, da Stromnetze auf Grund verschiedener Faktoren über digitale Wege steuerbar sein müssen und sobald etwas digital erreichbar ist, kann böswillig in das System eingedrungen werden. Um diese Angriffe zu verhindern, gibt es verschiedene Ansätze. Einer ist die Intrusion Detection. Die Arbeitsgruppe von Professorin Remke hat bereits eine Stromnetzsimulation mit einem Intrusion Detection System (IDS) entwickelt. Der Aufbau dessen wird im Kapitel 3 genauer erläutert. Das System ermöglicht allerdings bisher nur eine Überprüfung auf physikalische Regeln und vertraut darauf, dass die Verbindung zwischen den Überwachungsmonitoren und der Steuereinheit (Remote Terminal Unit kurz RTU) des Stromnetzes sicher ist. Das Vertrauen in diese Verbindung kann aber nicht garantiert werden, da Befehle unter Verwendung des gleichen Protokolls wie in realen Stromnetzen unverschlüsselt und ohne Authentifizierung gesendet werden. Genauer ist dies in Kapitel 2.2 beschrieben. Das ist ein historisch gewachsenes Problem

und die Umstellung dieser Verbindung wäre an zu vielen Stellen notwendig, als dass eine Umstellung kurzfristig möglich wäre. Außerdem kann selbst bei einer Umstellung des Protokolls nie ein umfassender Schutz vor Angriffen gewährleistet werden. Hier setzt diese Arbeit an, mit dem Ziel, die unsichere Verbindung als gegeben anzunehmen und Angriffe, welche diese Unsicherheit ausnutzen, abzufangen und größere Schaden zu verhindern. Wichtig ist im ersten Schritt die Verbindung der RTU zu einer Validierungseinheit. Im zweiten Schritt sollte somit eine Validierung erfolgen. Welche Art der Validierung sinnvoll sein kann, wird im Konzept erarbeitet.

1.2. Forschungsfragen

Aus der vorher geschilderten Situation ergeben sich folgende Forschungsfragen, welche in dieser Arbeit thematisiert werden:

1. Wie kann eine Validierung von Befehlen an eine RTU vor ihrer Ausführung in das bestehende IDS von Menzel et al. integriert werden?[1]

2. Welche Angriffe auf ein Stromnetz können mit dieser Art der Validierung verhindert werden?

3. Ist es möglich die Validierung auf ein echtes Netz zu übertragen und wie erfolgversprechend wäre eine Validierung durch eine Simulation?

1.3. Quellcode

Der Quellcode ist in einem GitHub-Repository mit der folgenden URL zu finden:
`https://github.com/Nero3ooo/intrusion-prevention-mosaik-power-grid`
Version des Quellcodes bei Abgabe der Bachelorarbeit entspricht dem Release $v1.0.0$ mit dem Tag $release\_for\_bachelor$.

1.4. Aufbau

Zuerst werde ich in dieser Arbeit auf die theoretischen Grundlagen (siehe Kapitel 2) eingehen. Hierzu werden Stromnetze, SCADA, Intrusion Prevention, Digital Twin und Simulationen eines Stromnetzes erläutert. Nachdem die wissenschaftliche Grundlage geschaffen wurde, beschreibe ich in Kapitel 3 den Kontext, in dem diese Arbeit entsteht. Hier wird kurz auf vorherige Arbeiten eingegangen, um im nächsten Kapitel 4 das Konzept für die Umsetzung aufzubauen. Die Implementierung (siehe Kapitel 5) beschreibt nun die Umsetzung der Arbeit, welche in der Evaluation in Kapitel 6 noch einmal reflektiert und ausgewertet wird. Es folgt das Fazit mit einer Zusammenfassung und der Beantwortung der Forschungsfragen. Den Abschluss bildet ein kurzer Ausblick für weitere Forschung.

2 Theoretische Grundlagen

Für das Verständnis dieser Arbeit sind die relevanten Grundlagen in den verschiedenen Sektionen dieses Kapitels zusammengefasst. Zuerst wird in den Abschnitten 2.1 und 2.2 darauf eingegangen, wie Stromnetze aufgebaut sind, aus welchen Komponenten sie bestehen und wie sie gesteuert werden. Dies gibt einen Einblick in den größeren Kontext, in dem die Arbeit entsteht. Danach werden mögliche Sicherheitsmaßnahmen und Analysemöglichkeiten in den Abschnitten 2.3, 2.4 und 2.5 vorgestellt, um einen Einblick in die aktuellen Maßnahmen zum Schutz vor Angriffen zu geben und ein Konzept zur Validierung von Befehlen an die RTU zu erarbeiten.

2.1. Stromnetze

Ein Stromnetz besteht aus verschiedenen Komponenten. Es besitzt Produzenten, Verbraucher, Busse, Leitungen, Switche, Sensoren, Transformatoren, Relais und Sicherungen (vgl. [2]). Produzenten sind beispielsweise Atomkraftwerke, Solar-, Photovoltaik- oder Windkraftanlagen (Nr. 1 in Abb. 2.1, folgende ebenso dort). Verschiedene Verbraucher wären Industrieparks oder Privathaushalte (2). Verbunden sind Produzenten und Verbraucher durch Leitungen. Da verschiedene Spannungen vorherrschen, werden Transformatoren (3) benötigt, um die Spannung umzuwandeln. Höchstspannung wird für große Strommengen über weite Distanzen verwendet, da der Verlust hierbei am geringsten ist. Mittelspannung wird eher für größere Industrie und Solar- oder Windparks und Niederspannunng für private Haushalte verwendet [3], wobei es mittlerweile auch in diesem Bereich Produzenten wie Photovoltaikanlagen gibt. Des Weiteren kann das Netz in Teilnetze aufgeteilt werden, die jeweils von einer RTU (4), welche mit dem Rechenzentrum (5) verbunden ist, gesteuert werden. Die Verbindung zwischen den verschiedenen Komponenten ist in Abbildung 2.1 zu sehen. Das Rechenzentrum verwendet mittlerweile häufig sogenannte Smart-Grids, um bei wechselnder Produktionsleistung der erneuerbaren Energien oder schwankendem Verbrauch schnell einzugreifen und das Netz zu optimieren [4]. Die Smart-Grids werden

durch SCADA-Systeme, welche für die Steuerung von Stromnetzen oder anderen großen industriellen Prozessen zum Einsatz kommen, ergänzt [5].

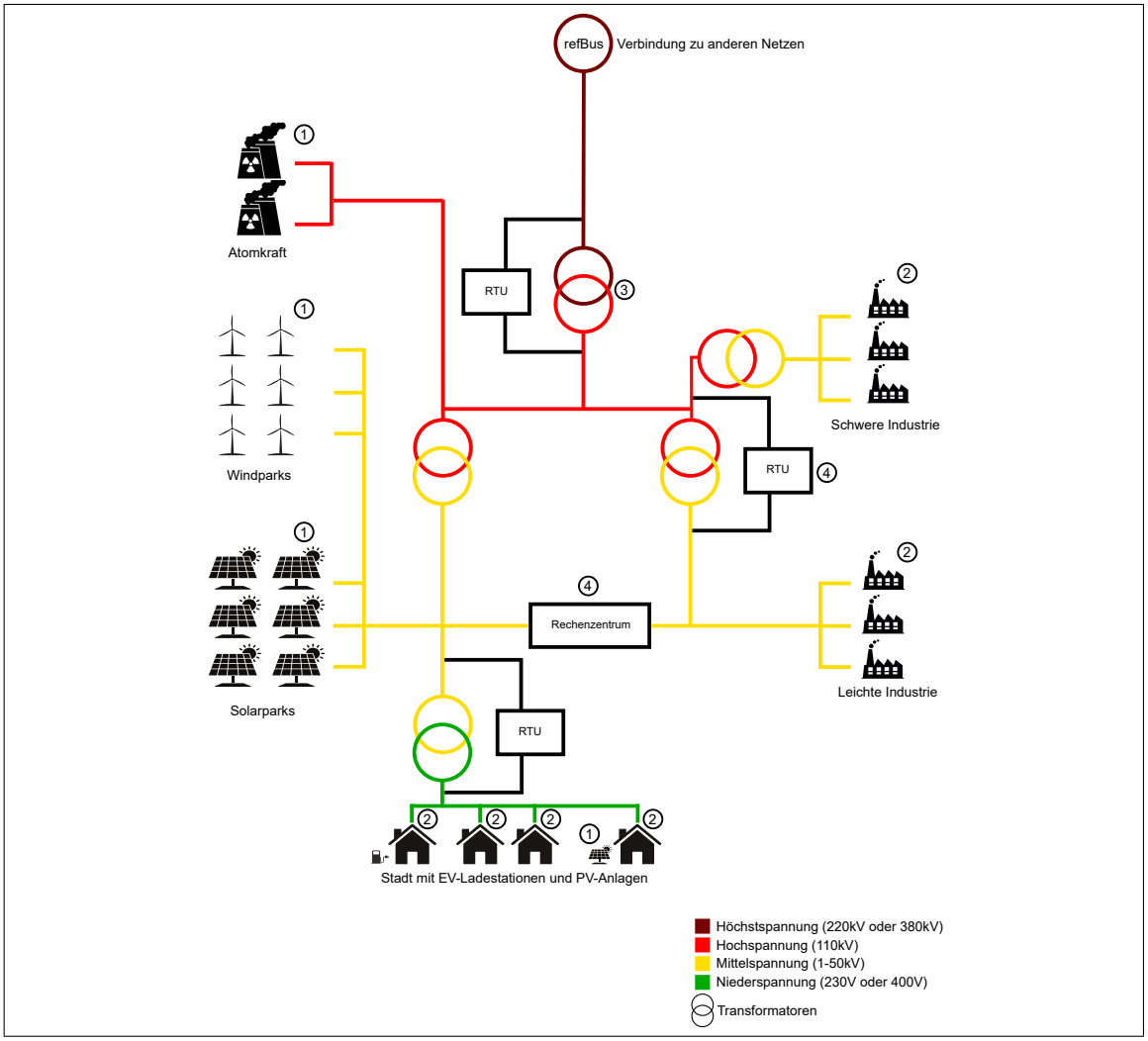

Abbildung 2.1.: Beispielhafte Abbildung eines Stromnetzes vergleichbar mit der europäischen Struktur [1]

Dies ist allerdings eine relativ neue Entwicklung, die unter anderem mit der Stromproduktion durch erneuerbare Energien zusammenhängt. Als es nur wenige große Produktionsanlagen gab, war das Stromnetz relativ einfach gehalten und hatte eine Top-Down Struktur, an der oben die Anlagen standen und unten die Verbraucher [6, 7]. Da es nur wenige

[1] Diese und alle folgenden Abbildungen wurden selbst erstellt.

große Produktionsanlagen gab und diese Einfluss auf die produzierte Strommenge hatten, wurde diese auf die Verbraucher angepasst. Mit den alternativen Energien wie Photovoltaik oder Windkraft sind mehrere kleine Produzenten hinzugekommen, die außerdem eine weniger flexible Produktion haben. Zum Beispiel kann eine Photovoltaik Anlage nachts oder bei starker Bewölkung nur weniger oder keinen Strom produzieren. Es ist entweder möglich die am Tag produzierte Energie zu speichern oder das Defizit durch Windkraft oder andere Produzenten abzufangen [8]. Mit den alternativen Energien gehen also mehr Daten, Erweiterung der SCADA-Systeme (Supervisory Control and Data Aquisition) und damit auch eine erhöhte Kommunikation einher. Das Stromnetz ist also flexibler gestaltet worden und muss es auch weiterhin [8]. Durch diese neue Flexibilität, kommt der technologische Fortschritt in Verbindung mit alten, bereits bestehenden Systemen zum Einsatz. Es werden Smart-Grids eingeführt[6], aber die SCADA-Protokolle Modbus und DNP3 beispielsweise sind von 1979 beziehungsweise 1993 [9, 10]. Sie sind in den alten Systemen verankert, auf welche die Smart Grids aufbauen. Auf eben diese Protokolle und das SCADA-System im Allgememeinen soll nun eingegangen werden.

2.2. SCADA

SCADA-Systeme erlauben es industrielle Prozesse lokal oder per Remote zu steuern und Daten aus dem Netzwerk zu sammeln und bereitzustellen. Hierzu hat ein SCADA System mindestens eine RTU (Remote Terminal Unit). Diese kommuniziert direkt mit den Sensoren, Switchen und anderen physischen Komponenten. Außerdem sind diese RTUs mit Master Station(en) verbunden [11]. Diese sorgen für die Darstellung der von der RTU bereitgestellten Werte und senden Befehle an die RTU. Die Befehle werden entweder durch die bereitgestellten Daten ermittelt oder manuell durch einen Menschen getätigt. Für diese Verbindung zwischen RTU und Station wurden Protokolle entwickelt, um auf einen Standard zurückgreifen zu können. Im Folgenden werden zwei Protokolle genauer erläutert.

2.2.1. SCADA-Protokolle

Modbus (MODBUS TCP/IP) und DNP3 sind SCADA-Protokolle, welche beide auf TCP basieren [12, 13]. Sie bauen einmalig eine bestehende Verbindung zwischen Server (RTU) und Client (Master Station) auf um Befehle und Informationen auszutauschen. Modbus definiert hierzu eine “Protocol Data Unit” (PDU), sie besteht aus dem Funktionscode und einem Datenblock. Der Funktionscode gibt an, was von der RTU gefordert wird, die verschiedenen Codes sind in der Tabelle 2.1 zu sehen. Es wird grundsätzlich zwischen diskreter Eingabe, Coils, Input- und Holding-Registern und außerdem zwischen Lese- und Schreiboperationen unterschieden. Während die diskrete Eingabe nur gelesen werden kann, ist bei den anderen drei Arten ein Lese- und Schreibzugriff erlaubt. Direkte Eingaben und Coils verwenden nur einen Bit, Register hingegen sind 16 Bit Wörter [14]. Im Stromnetz werden beispielsweise Switche durch Coils gesteuert, da sie nur zwei Status haben. Werte von Sensoren hingegen werden in Register geschrieben.

Operation	Funktionscode	Funktion	Wert
Lesen	01	Coil Status	Diskret
	02	Input Status	Diskret
	03	Holding Register	16 Bit
	04	Input Register	16 Bit
Schreiben	05	Single Coil	Diskret
	06	Single Register	16 Bit
	15	Multiple Coils	Diskret
	16	Multiple Registers	16 Bit

Tabelle 2.1.: Modbus Funktionscodes

In Abbildung 2.2 ist beispielhaft das Senden eines Befehls der Attack-Engine (siehe Abschnitt 3.3) zu der RTU dargestellt. Hierbei ist die Application Data Unit ganz am Ende zu sehen: $01\ 05\ 00\ 00\ ff\ 00$. 01 steht für die Objektadresse also den Switch. 05 ist die Schreiboperation für einen Single Coil. $00\ 00$ bezeichnen die Adressen der ersten Register Hi- bzw. Lo-Bytes. ff und 00 sind die Werte, welche in diese Register geschrieben werden sollen. Sie besagen, dass der Switch geschlossen ist. Die Werte 00 und 00 müssten

gesendet werden, um den Switch zu öffnen [14]. Um diese Befehle zu senden, werden aber auch Bibliotheken bereitgestellt.
Für Python existiert unter anderem “pymodbus3”, eine Bibliothek, die einen Modbus-Client mit Funktionen wie “read_register” oder “write_coil” zur Verfügung stellt. Diesen Funktionen werden Adresse und gegebenenfalls ein neuer Wert als Parameter mitgegeben[15], das Erstellen und Senden des konkreten Befehls erfolgt dann innerhalb der Bibliothek.

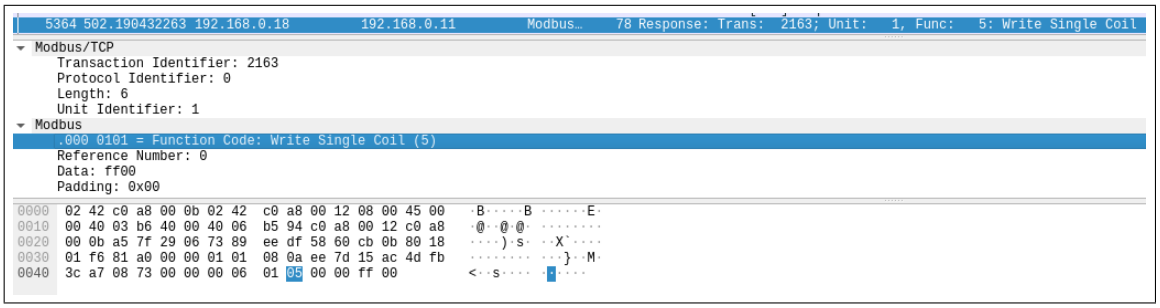

Abbildung 2.2.: Paket zum Senden eines Befehls von der Attack-Engine an die RTU dargestellt durch Wireshark [16]

DNP3 ist deutlich umfangreicher. Es verwendet 27 Funktionscodes für die Kommunikation zwischen der RTU und den anderen Komponenten. Innerhalb dieses Umfangs liegen auch Befehle, die eine Komponente auffordern einen kalten oder warmen Neustart zu machen oder Dateien zu löschen [17]. Es kann somit nicht nur für die Kommunikation zwischen RTU und Switch oder Sensor verwendet werden, sondern auch in einer hierarchischen feiner gegliederten Struktur, in der eine RTU beispielsweise mehrere Sub-RTUs besitzt die mit den Komponenten kommunizieren.

Beide Protokolle sind aus unterschiedlichen Gründen nicht auf Sicherheit ausgelegt. Zum einen wurde Modbus ursprünglich für abgeschlossene Systeme entwickelt [18], zum anderen wurde der Schutz vor Angriffen in digitaler Form bei der Entwicklung von Protokollen grundsätzlich nicht berücksichtigt. Entweder war eine Verschlüsselung vor dem Verwenden des Protokolls angedacht [19] oder in den Anfängen gar nicht erst nötig, da Technologie nicht verbreitet und Angriffe der Art nicht denkbar waren. Als Folge dieser Unsicherheit ist es heute durch Programme wie Wireshark sehr einfach, den Befehl darzustellen, wie

in Abbildung 2.2 zu sehen, und genauso wäre eine Mainpulation dieser Pakete denkbar. Eine Identifizierung der einzelnen Komponenten durch das geschlossene System war auch nicht notwendig. Zur Verschlüsselung innerhalb des Protokolls wurde für DNP3 nachträglich das Framework DNP3SA bereitgestellt. Für Modbus existieren auch Erweiterungen wie MODBUS/TCP Security, welches eine Identifizierung und Authentifizierung mittels Zertifikat mit sich bringt [20]. Eine Umstellung auf diese Protokolle wäre allerdings sehr aufwendig, da dies wahrscheinlich vor Ort und bei jeder RTU einzeln durchgeführt werden müsste. Außerdem erhöht dies zwar die Sicherheit durch das Protokoll, aber es ist denkbar, dass Verschlüsselungen und Authentifizierungen umgangen werden können. So kann ein Angreifer weiterhin das Protokoll nutzen, um Befehle mit nicht vorhersehbarem Schaden innerhalb des SCADA-Systems zu senden. Zum Schutz vor diesen Angriffen gibt es insbesondere zwei Strategien, die in der Wissenschaft verfolgt werden: Intrusion Prevention und Digital Twins. Letztere werden in der Fachliteratur nicht mit Intrusion Prevention verknüpft, da diese eine deutlich weitreichendere Funktion besitzen und sich die Intrusion Prevention auf die Netzwerkebene konzentriert.

2.3. Intrusion Prevention

Da die Steuerung des SCADA-Systems innerhalb eines Netzwerks und die Kommunikation häufig mittels TCP stattfindet, wird in der Intrusion Prevention meist das Netzwerk überwacht und abgesichert. Es soll Angreifern gar nicht erst ermöglicht werden, in das System einzudringen. Eine Firewall kann beispielsweise nur bestimmten Geräten den Zugriff auf das Netzwerk gewähren oder die Art der Kommunikation zwischen den Geräten einschränken. Außerdem wird auf den Datenverkehr geachtet, da die Kommunikation des SCADA-Netzes sehr regelmäßig und vorhersehbar ist [21]. So teilen die Sensoren der RTU beispielsweise in regelmäßigen Abständen die neuen Messwerte mit. Bei einer Manipulation der Sensorwerte kann es zu Unregelmäßigkeiten oder stark ansteigendem Traffic kommen. Zur genaueren Betrachtung der Datenpakete gibt eine Linux basierte Firewall, welche diese untersucht und feststellt, ob es sich um Pakete handelt, die das Modbus Protokoll nutzen [12, 22]. Für die Firewall können verschiedene Filter definiert werden,

diese betreffen unter anderem den Funktionscode, das Register oder die Transaktions-Id und sind beliebig miteinander kombinierbar. Eine ähnliche Variante existiert für DPN3, allerdings gibt es in diesem Bereich kein großes Spektrum [12, 23]. Die Intrusion Protection, bei der nach dem Eindringen versucht wird, den potenziellen Schaden zu minimieren, ist in der Fachliteratur nicht so verbreitet. Hierzu gibt es eher wenige Ansätze. Eine Intrusion Detection im allgemeinen kann wie bei der Intrusion Prevention auf Netzwerk-Ebene stattfinden oder auch das Stromnetz betrachten und untersuchen [24]. Bei der zweiten Variante wird versucht Angriffe an Stromnetzen zu erkennen, indem die Werte der Sensoren analysiert und im Zusammenhang mit dem produzierten und verbrauchten Strom betrachtet werden. Das ist beispielsweise mit einem Digital Twin umsetzbar.

2.4. Digital Twin

Eine weitere Form den Schutz vor Angriffen digitaler Art gegen Stromnetze zu erhöhen, bietet das Konzept des Digital Twins. Grundsätzlich ist ein Digital Twin eine virtuelle Abbildung des physisch existierenden Systems. Das System und die Abbildung sind miteinander verbunden und erhalten Informationen voneinander, so dass das physische System in Echtzeit virtuell dargestellt wird [25]. Dem Twin wird außerdem auf Grund der großen Datenmengen ermöglicht, Vorhersagen zu treffen und Schwachstellen zu erkennen. Es hilft aber auch dabei, ein wachsendes System zu planen, um es danach stabil auszubauen. Explizit bei dem Stromnetz ist es wichtig, und durch den Digital Twin möglich, komplexe Beziehungen in analytische, simulierte und berechenbare mathematische Beziehungen umzuwandeln [26]. Die Kombination aus dem physischen und virtuellen Stromnetz bringt eine Überwachung, Frühwarnung und Simulation mit sich. Des Weiteren findet eine Analyse, Ableitung von Befehlen und Steuerung des Netzes statt. Dies gelingt dem virtuellen System durch Modellabbildung des physischen Netzes, mit der verschiedene Zustände erkannt werden können. Es wird aus den vergangenen Daten gelernt und Entscheidungen, Weiterentwicklung des eigenen Systems und Steuerung des Netzes durch das Erteilen von Befehlen an das physische Netz können getroffen und umgesetzt werden [25, 26]. Wie auch in der Abbildung 2.3 zu sehen ist, entsteht ein Kreislauf zwischen den beiden

Komponenten. Das Virtuelle Netz lernt aus den Daten, die das physische Netz liefert, kann darauf basierend Entscheidungen treffen und Resultate dieser wiederum in zukünftigen Szenarien berücksichtigen. Da es sich bei Digital Twins um ganzheitliche Abbildungen handelt, welche sogar die Steuerungen übernehmen können, bemerken sie auch die Manipulation von Daten oder können das Ausführen gefährlicher Befehle verhindern [27]. Eine vollständige Implementation ist jedoch auf Grund der Größe und Komplexität nur schwer umsetzbar. Daher werden Digital Twins zumeist nicht mit dem vollen Umfang implementiert. Gerade der Bereich Intrusion Detection ist eher aktueller Forschungsstand und bisher selten umgesetzt. Eine weitere Möglichkeit zur Analyse und Entwicklung von Stromnetzen ist die Simulation dieser.

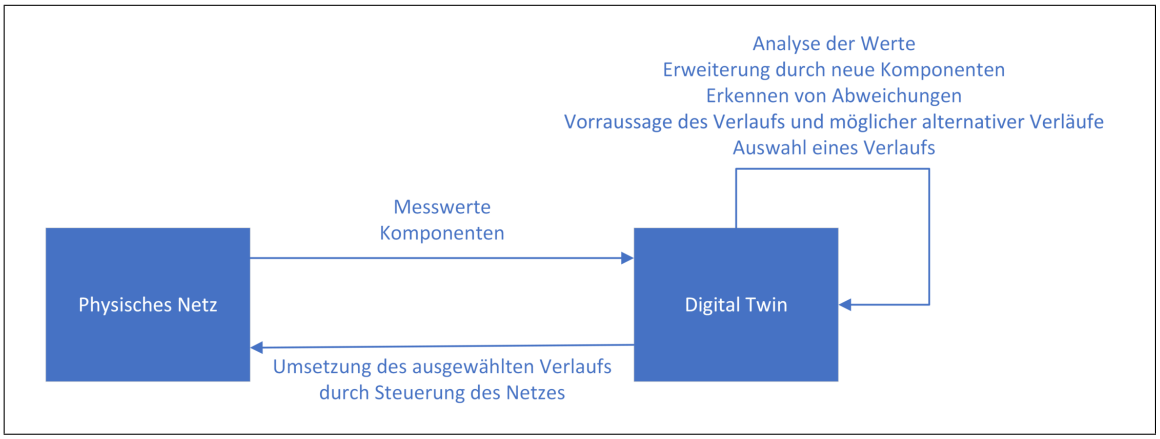

Abbildung 2.3.: Symbiose aus physischem und virtuellem Netz

2.5. Simulationen

Um Stromnetze zu untersuchen und zu entwickeln, können sie auch simuliert werden. Anders als bei den Digital Twins sind diese Simulationen eigenständig. Sie sind nicht an ein physisches Stromnetz gekoppelt, sondern sollen ein solches imitieren. Sie lassen sich einfacher untersuchen und bilden einen festen Rahmen. Außerdem dienen sie wie Digital Twins auch dazu, Netze auf Schwachstellen zu untersuchen und Erweiterungen zu planen, da Stromnetze im Allgemeinen zur kritischen Infrastruktur zählen und Änderungen oder Neuerungen getestet werden sollten, bevor sie an dem physischen Netz umgesetzt werden. Es gibt verschiedene Umgebungen, die sich dafür eignen, Stromnetze, das dazugehörige Netzwerk und ihre Komponenten zu simulieren. Je nach Umgebung wird ein Schwerpunkt der Betrachtung auf das Netzwerk oder auf unterschiedlich große Teile des Stromnetzes gelegt. Ein Framework, welches in der Arbeitsgruppe von Professor Remke verwendet wird, ist Mosaik und bezieht sich auf “kleinere” Netzstrukturen, wobei die Netzwerk-Ebene eher im Hintergrund steht und “kleiner” in diesem Kontext eher die Größe einer Stadt widerspiegelt.

2.5.1. Mosaik

Mosaik ist ein Co-Simulation-Framework, welches verschiedene Simulationen bereitstellt. Diese oder auch selbst entwickelte Simulationen werden von Mosaik miteinander verbunden. Das Framework versucht hierbei die Prozesse der Simulationen zu synchronisieren und verwaltet den Datenaustausch zwischen ihnen [28]. Für Austausch und Synchronisation stellt das Mosaik eine API für die Kommunikation bereit. Außerdem können Simulationsszenarien modelliert und Handler für unterschiedliche Arten von Simulationen implementiert werden. Beispielsweise bietet es die Möglichkeit, ein Smart-Grid zu simulieren, indem eine Simulation einer Photovoltaik-Anlage mit denen für Haushalte und einer für ein Überwachungstool miteinander verbunden sind. Eine solche Simulation mit verschiedenen Komponenten wurde bereits bei Chromik et al. [2] verwendet, um das Verhalten von Stromnetzen genauer zu untersuchen. In Abbildung 2.4 ist ein Beispiel eines Stromnetzes zu sehen. Es wurde von Menzel et al. in Verwendung von Mosaik entworfen.

Mosaik eignet sich auf Grund der Komplexität und Rechenleistung eher dazu, einen Teil des Stromnetzes genauer zu betrachten und zu untersuchen. Es können Szenarien mit mehreren RTUs, Verbrauchern, Produzenten und den anderen Komponenten erstellt und genauestens konfiguriert werden. So ist das Testbed, welches im Abschnitt Testbed beschrieben wird, mit zwei RTUs konfiguriert. Eine Betrachtung und Simulation eines größeren Netzes könnte schnell unübersichtlich werden und zu viel Rechenleistung benötigen. Es ist daher ersteinmal sinnvoll, das Netz etwas kleiner zu halten. Es verliert auf Grund des kleineren Designs nicht an Bedeutung im Gesamtkontext der Stromnetzsimulation, im Gegenteil ist eine stabile Simulation eines kleinen Teils die Grundlage für größere Netze, da diese aus vielen Teilnetzen zusammengesetzt sein können [29]. Außerdem kann eine Validierung der Befehle an eine RTU einfach auf andere RTUs in größeren Netzen übertragen werden.

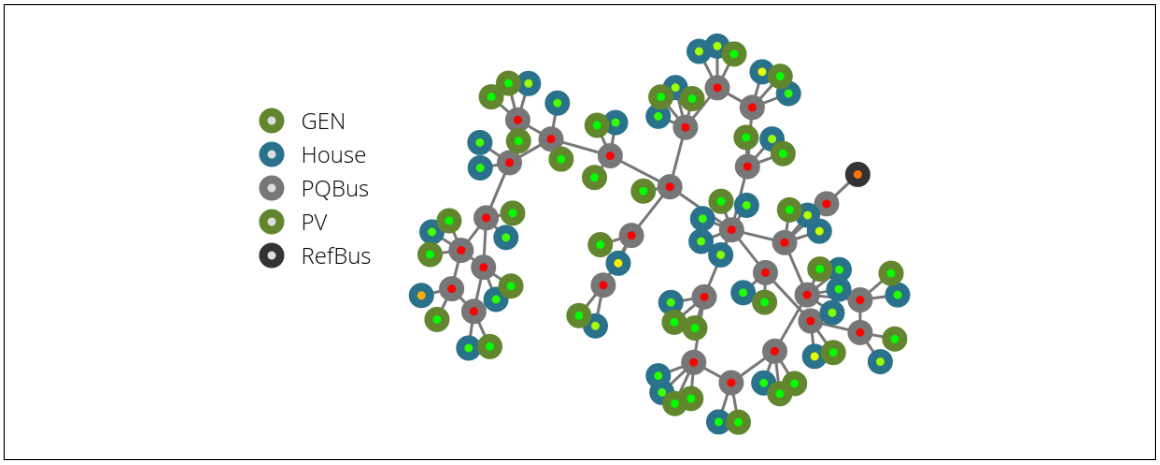

Abbildung 2.4.: Darstellung eines Stromnetzes durch Mosaik [2]

[2]Screenshot der Websimulation von Mosaik des von Menzel et al. entworfenen Netzes

3 Projektbeschreibung und vorherige Arbeiten

Der Aufbau der Testumgebung ist in Abbildung 3.1 dargestellt. Mosaik sorgt für eine Simulation des physischen Stromnetzes und übermittelt die Daten der Sensoren, Switche, Transformatoren und gegebenenfalls weiterer Komponenten an das Modbus RTU device. Das Modbus RTU device besitzt einen Modbus Server, dieser kommuniziert mit einem Localmonitor und gegebenenfalls weiteren Neighborhoodmonitoren, welche für die Steuerung und Überwachung des Stromnetzes zuständig sind. Die Attack-Engine kann sich auch mit dem Modbus Server verbinden und die Sensor Daten, welche durch den Controller geliefert werden, überschreiben. Des Weiteren ist es durch die Attack-Engine möglich, das RTU device mittels Modbus Befehlen zu steuern, und somit nicht nur den Status der Switche und Transformatoren zu überschreiben, sondern diese aktiv zu beeinflussen. Das heißt beispielsweise für einen Switch, dass es nicht nur im System steht, dass der Switch nun geschlossen sei, er ist auch tatsächlich umgelegt worden und somit geschlossen.

3.1. Testbed

Das Testbed, welches von J. J. Chromik entwickelt und von V. Menzel erweitert wurde [1], umfasst ein Testszenario in dem die verschiedenen Simulationen konfiguriert werden. Es gibt PyPower, HouseholdSim, PV, zwei RTUs und WebVis. PyPower ist ein Framework, welches zur Bestimmung des optimalen Stromflusses in einem Netz dient und kann mittels Mosaik-PyPower in Mosaik eingebunden werden. Die HouseholdSim dient zur Simulation der Haushalte als Verbraucher, indem sie auf eine Datei mit vordefinierten Verbrauchen für die jeweiligen Häuser zugreift. Die PV simuliert Photovoltaik-Anlagen in der gleichen Art wie die HouseholdSim, nur dass Strom zum Netz hinzugegeben und nicht verbraucht wird. Die beiden RTU-Simulationen sorgen für eine Steuermöglichkeit des Netzes und eine Bereitstellung der Daten. Die Simulation WebVis ist für die Veranschaulichung des

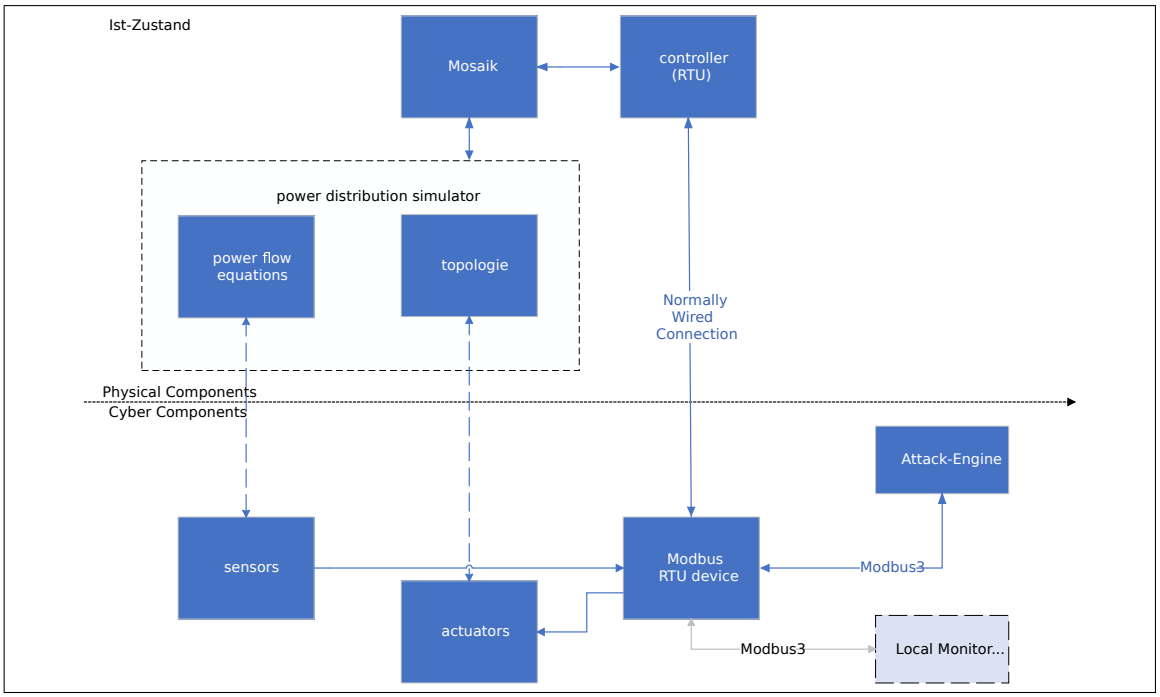

Abbildung 3.1.: Der Zustand des Testbettes nach dem Projektseminar IT-Sicherheit für Stromnetze an der WWU Münster [1]

gesamten Netzes zuständig und ist bereits in Abbildung 2.4 zu sehen gewesen. Das Stromnetz wird in der $demo\_mv\_grid.json$ definiert. Die $new\_rtu\_0.xml$ und $new\_rtu\_1.xml$ konfigurieren aufbauend auf dem definierten Stromnetz und zusammen mit der $config.cfg$ die RTUs. Der Verbrauch der Haushalte ist in $profiles\_mv.data$ und die Produkton der Photvoltaik-Anlagen in der $pv\_10kw.csv$ definiert. Für die Validierung ist dies insofern von Relevanz, als dass sowohl Produktion als auch Verbrauch nach vordefiniertem Schema ablaufen und dies in einem echten Stromnetz nicht der Fall ist. Dort kann nur durch das Sammeln von Daten eine potenzielle Zukunft geschaffen werden. Die $new\_rtu\_0.xml$, $new\_rtu\_1.xml$ und $config.cfg$ sind die Dateien, welche vermutlich bei einer Validierung durch die RTUs angepasst und verändert werden müssten, da eine Kopie der RTU für die Validierung in Frage kommen könnte. Die RTUs sind bereits an Monitore zur Überwachung des Testbeds angeschlossen, dies wird im Folgenden weiter erläutert.

[1] Die Abbildung 3.1 orientiert sich an Fig. 3 aus “An integrated testbed for locally monitoring SCADA systems in smart grids” von J.J. Chromik, A. Remke und B.R. Haverkort aus dem Jahr 2018[30]

3.2. Überwachung durch Monitore

Für die Überwachung der Local-Monitore werden die über das Modbus-Protokoll gesendeten Daten mittels sechs Regeln überprüft. Hier kommt es ausschließlich zu einer Überprüfung der physikalischen Eigenschaften der Komponenten. Die Neighborhood-Monitore überprüfen nur die Regeln Req3 und Req4. Außerdem existiert eine Webansicht, welche diese Regelverstöße sichtbar macht. Sie kann mit dem Überwachungspanel der Master Station eines SCADA-Systems gleichgesetzt werden.

- **Req1:** Die Summe des einkommenden Stroms aller Leitungen eines Busses muss der Summe des ausgehenden Stroms aller Leitungen eines Busses entsprechen (Knotenpunktsatz).
- **Req2:** Jede Stromspannung einer Leitung an einem Bus ist gleich.
- **Req3:** Es ist kein Strom auf einer Leitung mit einem offenen Switch.
- **Req4:** Stromstärke und Spannung sind an jedem gemessenen Punkt der Leitung gleich.
- **Req7:** Die gemessene Stromstärke ist geringer als der für den Bus maximale Strom.
- **Req8:** Die gemessene Spannung ist geringer als die für den Bus maximale Spannung.

Da die Daten jedoch über das alte Modbus-Protokoll von 1979 gesendet werden, besteht die Gefahr, dass die Werte überschrieben werden. Die Überprüfung würde also auf Annahme falscher Werte stattfinden. Zur Ausführung eines Angriffs der Art wurde bereits die Attack-Engine entwickelt.

3.3. Manipulation durch die Attack-Engine

Wie bereits erwähnt, kann die RTU mittels der Attack-Engine gesteuert werden. Sie sendet hierzu wie im Abschnitt SCADA-Protokolle beschrieben mittels Modbus Befehle an die RTU. Diese werden in dem aktuellen Aufbau nicht validiert. Ein Angriff oder menschlicher Fehler kann somit nicht frühzeitig erkannt oder verhindert werden. Ein denkbarer Angriff an dieser Stelle wäre eine Art Man-in-the-Middle Angriff wie er in “Process-aware SCADA Traffic Monitoring: A Local Approach” im Unterkapitel 5.4.1 von J. J. Chromik aus dem Jahr 2019[30] beschrieben wird. Die Verbindung zwischen SCADA-Server und RTU ist nicht sicher und es können sowohl Befehle an die RTU gesendet als auch die Werte von Sensoren manipuliert werden. Dies ist aufgrund der Unsicherheit des Modbus Protokolls möglich, welches keine Verschlüsselung oder Autorisierung vorsieht. Die Attack-Engine ist außerdem die einzige Komponente im Testbed, die eine Steuerung des Netzes ermöglicht, sie ist in der Abbildung 3.2 zu sehen.

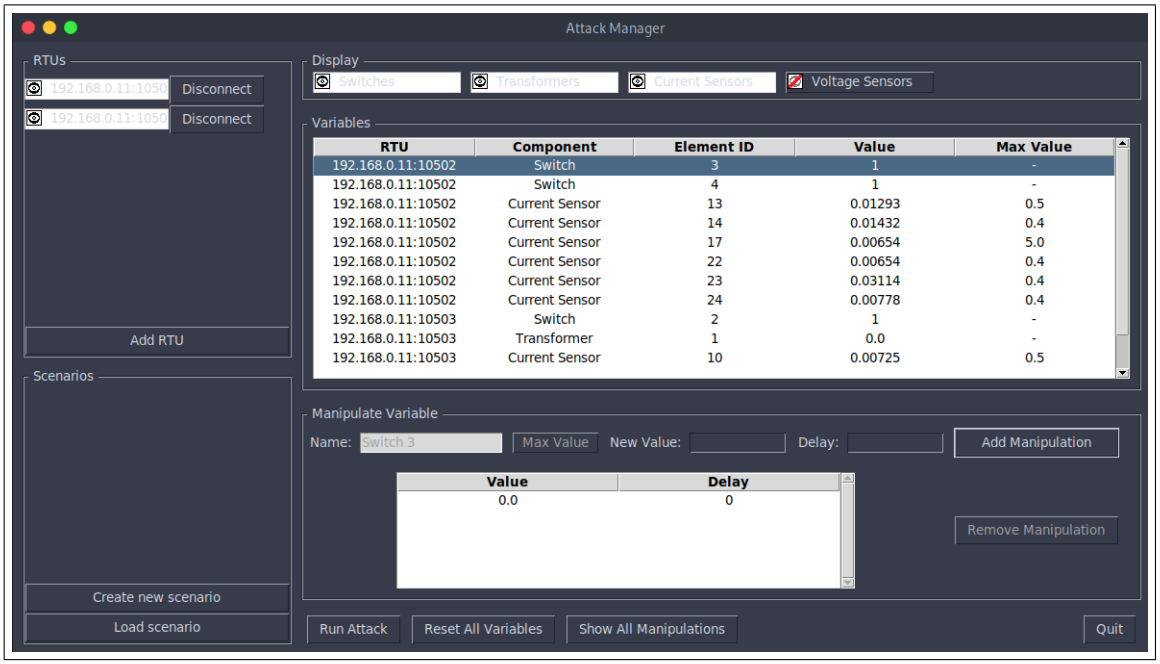

Abbildung 3.2.: Abbildung der Attack-Engine

Die Engine kann nicht nur einen einzelnen Befehl an die RTU senden, sondern auch Szenarien planen, in denen mehrere Befehle für die Steuerung der Switche gleichzeitig oder versetzt mit der Manipulation von den Sensorwerten kombiniert werden. Bei der Manipulation findet eine Annäherung an die Werte statt, so dass es keine großen Sprünge gibt, das ist weniger auffällig und die Manipulation bleibt länger unentdeckt. Die Sensorwerte, welche die RTU bereitstellt, werden auch immer neu überschrieben, so dass der Server diese nur durch Zufall in der gleichen Sekunde mitbekommen kann. Befehle, die auf Grundlage der veränderten Werte getroffen werden, sorgen gegebenenfalls für Schäden an dem Stromnetz oder führen zu einem Blackout.

Um diese Art von Angriffen zu verhindern, könnten die Protokolle aktualisiert werden und somit eine Authentifizierung notwendig sein. Zusätzlich wäre es denkbar, den Netzwerktraffic zu überwachen, wie es in der Intrusion Prevention beschrieben wurde. Das Überschreiben der Sensoren und das Senden der Befehle geschieht erstens von einem Gerät im Netzwerk, welches wahrscheinlich nicht im Netz sein dürfte. Zweitens werden diese Pakete sehr häufig gesendet, um die Überschreibung der Werte, die die RTU liefert, zu garantieren und drittens dürfte eine Komponente nicht gleichzeitig Sensorwerte aktualisieren und Befehle an die RTU senden. Die dritte Möglichkeit, sich vor Angriffen dieser Art zu schützen, besteht darin, die Befehle, welche die RTU erhält, vorher zu validieren und so sowohl bösartige Befehle von der Attack-Engine als auch Befehle, die auf Grund von falschen Sensorwerten getroffen wurden, zu erkennen und zu verhindern, falls das Netz beschädigt werden würde. Ein Konzept zu dieser Möglichkeit wird im folgenden erarbeitet und umgesetzt (siehe Kapitel 4 und 5).

4 Konzept

Wie bereits im Projektbeschreibung und vorherige Arbeiten zu sehen war und in der Einführung angesprochen wurde, fehlt in dem gegebenen Testbed eine Validierung der Befehle, welche für die Steuerung der RTU zuständig sind. Hierzu bietet es sich an eine Validierungskomponente einzubinden. Da je nach Verfahren der Validierung ein entstehender Rechenaufwand sehr groß werden kann, ist es sinnvoll, diese nicht direkt mit der RTU zu verbinden, sondern die RTU als Client eine Verbindung zu einem Validierungsserver aufbauen zu lassen, auf dem die Validierungskomponente implementiert wird. Die Intrusion Prevention auf Netzwerkebene eignet sich nicht für eine Validierung, da eine Netzwerküberwachung den Verlauf des Stromnetzes nicht vorhersagen kann und hiervon größtenteils unabhängig ist. Die Validierungskomponente sollte auch eingreifen, wenn das Intrusion Prevention System auf Netzwerkebene versagt. Ein Digital Twin hingegen wird für die Planung und Optimierung der Strukturen eingesetzt und weniger für die Überwachung eines Stromnetzes. Dennoch könnte es sinnvoll sein, das Konzept der Simulation auf die Validierung der Befehle zu übertragen, insbesondere da es bereits eine Simulation des Netzes in Form des Testbeds gibt. Das Testbed baut auf Docker auf und somit können die relevanten Dateien einfach in den Dockercontainer der Validierung eingebunden werden, was ein Starten der Simulation innerhalb des Validierungsservers ermöglichen sollte. Um die Simulation für eine Validierung zu verwenden, müssen Kriterien definiert werden, nach denen die Simulation beurteilt, ob ein Befehl von der RTU durchgeführt werden darf oder nicht. Einerseits sollte eine Überprüfung auf Stromausfälle und ihre Reichweite stattfinden. Andererseits muss auf physikalische Grenzen wie bei den lokalen Monitoren geprüft werden, da gerade eine Überlastung des Busses und zu starke Spannungsschwankungen dem Bus und den Endgeräten schaden könnten. Für diese Überprüfung können die Sensorwerte, welche die RTU zur Verfügung hat, betrachtet werden. In der Arbeit werden nur die Veränderungen durch Switche berücksichtigt, daher würden sich die Regeln REQ3, REQ7 und REQ8 besonders eignen (siehe Regeln der Monitore). Zusätzlich müssten die

Sensoren, bei denen die Messwerte auf Null fallen, gezählt und ausgewertet werden. Der Einfachheit halber werden Verstöße gegen die physikalischen Gegebenheiten und eine geringe Anzahl an Null-Sensoren als Warnungen interpretiert und nur eine hohe Anzahl an Null-Sensoren als Error, da es außerhalb des Rahmens der Bachelorarbeit fällt, eine Gewichtung dieser Verstöße zu analysieren und festzulegen.

Im Folgenden wird zur besseren Unterscheidung die RTU, welche für die Validierung zuständig ist, mit RTU(IPS) benannt. Es wird sich um die gleiche Pythondatei handeln, wobei es während der Arbeit zu einer Erweiterung der Datei und Unterscheidung zwischen RTU des Intrusion Detection Systems und RTU(IPS) des Intrusion Protection Systems gekommen ist.

Die Kommunikation zwischen den verschiedenen Komponenten ist in Abbildung 4.1 dargestellt und sollte so ablaufen, dass ein Angreifer die Attack-Engine verwendet und ein Angriffsszenario erstellt. Wenn er dieses ausführt, werden Befehle von der Attack-Engine an die RTU gesendet. Die RTU ruft daraufhin eine Methode auf dem Validierungsserver auf und übergibt dieser die aktuellen Werte der RTU und einen Zeitstempel, an welchem Punkt sich die Simulation befindet. Die Validierungsmethode erstellt ein neues Testbed mit den übergebenen Werten und dem passenden Startzeitpunkt. Die mit dieser neuen Simulation erstellte RTU(IPS) sollte eine vorgegebene Anzahl an Simulationsschritten durchlaufen um dann die vorher definierten Warnungen und Errors an die Validierungskomponente zurückzugeben. Diese wiederum leitet das Ergebnis weiter an die RTU, welche den erhaltenen Befehl je nach Ergebnis ausführen oder verweigern sollte und dies dem Nutzer anschließend mitteilen sollte.

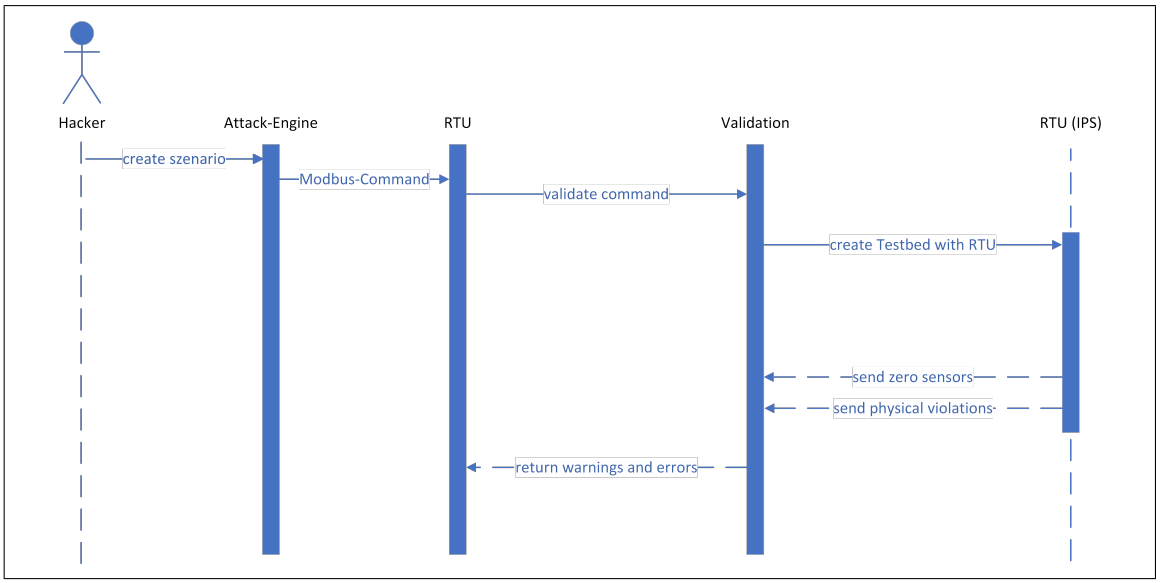

Abbildung 4.1.: Sequenzdiagramm zur Validierung eines Befehls von der Attack-Engine

Konkret muss für die Umsetzung der Validierung und Kommunikation Folgendes implementiert werden:

1. Anlegen eines neuen Docker-Container im Netzwerk des Testbeds
2. Aufsetzen eines Validierungsservers
3. Implementation einer Methode zur Validierung von Befehlen
4. Konfiguration der neuen Instanz des Testbeds
5. Die Möglichkeit zum Starten einer Instanz des Testbeds in dem neuen Container
6. Die Rückgabe des Ergebnisses der Simulation an die Validierungskomponente
    a) Für die auf Null fallenden Sensoren
    b) Für die physikalischen Verstöße
7. Die Rückgabe des Validierungsergebnisses an das Testbed
8. Die Ausgabe an den Benutzer und die Umsetzung des Ergebnisses

Für die Evaluation dieser Arbeit werden fünf verschiedene Szenarien aufgestellt und getestet. In Szenario 1 sollte sichergestellt werden, dass die Simulation weiterhin stabil und ohne Einfluss der Validierungskomponente läuft, wenn keine Befehle an die RTU gesendet werden. Des Weiteren sollte in Szenario 2 und 3 der Umgang mit Stromausfällen betrachtet werden, in dem durch das Senden von Befehlen 1. eine geringe Zahl an Sensoren und 2. eine größere Zahl an Sensoren den Stromwert 0.0 messen. Eine geringe Menge sollte tolerierbar sein, da es, beispielsweise bedingt durch Baustellen oder Reparaturen am Netz, möglich sein sollte, Teile des Netzes vom Strom zu nehmen. Befehle, bei denen größere Bereiche betroffen sind, sollten verhindert werden. Das Szenario 4, in dem physikalische Verstöße provoziert werden, sollte auch durchlaufen werden, um diese Art des Verstoßes zu testen. In Szenario 5 sollte getestet werden, was passiert, wenn die Validierungskomponente nicht erreichbar ist. Somit könnte ein DDOS Angriff gegen den Validierungsserver simuliert werden. Die verschiedenen Szenarien sind in Kapitel 6 zu finden.

5 Implementierung

Die Implementation der Validierung für die Befehle der RTU setzt sich, wie auch im Konzept zu sehen, aus den drei größeren Teilbereichen Vorbereitung der Serverumgebung, Verbindung via OPC und Validierung zusammen. Im Folgenden werden diese Bereiche in drei Kapiteln genauer erläutert.

5.1. Vorbereitung der Serverumgebung

Das Testbed, die Local- und Neighborhood-Monitore und die Attack-Engine haben jeweils eigene Docker-Container und es existiert bereits eine Docker-Compose Datei, in der die einzelnen Container definiert werden. Die Validierung sollte somit auch innerhalb eines eigenen Docker-Containers geschehen. So ist die Komponente nicht physisch abhängig von dem Testbed und kann in der Theorie als eigener Server betrachtet werden. Um das Testbed mit allen notwendigen Dateien auch auf dem Validierungsserver zur Verfügung zu haben, bietet Docker die Möglichkeit, Volumen zu teilen. Diese Funktion dient eigentlich auch zum Austausch von Dateien und Übertragen von Änderungen. So könnten beispielsweise die veränderten RTU-Daten auch auf diesem Weg an die Validierungskomponente übergeben werden. Allerdings setzt dies einen gemeinsamen Rechner voraus, so dass es einer Umsetzung auf Unabhängigen Servern im Wege steht. Sollte man anstatt Docker zu verwenden einen Server aufsetzen wollen, müssten die Ordner des Testbeds anders zugänglich gemacht werden. Der vollständige Quelltext für den Docker-Container des Validierungsservers ist auf der nächsten Seite zu sehen (siehe Quelltext 5.1). Die relevanten Ordner des Testbeds sind unter dem Punkt $volumes$ von Zeile 9 bis 16 zu finden. Die vorausgesetzten Bibliotheken sind die gleichen wie bei dem Testbed, da zur Validierung keine weiteren Bibliotheken benötigt werden. Allerdings sind sowohl beim Testbed als auch bei der Validierung die Bibliotheken “asyncio” und “asyncua” hinzugekommen. Diese werden zur Verbindung zwischen den beiden Komponenten benötigt, welche im nächsten Kapitel genauer erläutert wird.

```
# docker container for validation
validation:
  image: itsis-blackout/validation:latest
  build:
    context: ../validation
  container_name: mosaik_ids_validation
  env_file: config/environment/val.env
  command: python validation_setup.py
  volumes:
      # different ressources from testbed
      - ./testbed/:/app
      - ./testbed/mosaikpypower/mosaik_pypower/:/ids/mosaik_pypower
      - ./testbed/mosaikrtu/:/ids/mosaikrtu
      - ./testbed/mosaik-web/:/ids/mosaik-web
      - ./testbed/outputs/:/ids/outputs
      - ./config/ certificates :/config/ certificates
  networks:
    local_network_dev:
      ipv4_address: 192.168.0.19
```

Quelltext 5.1: Docker-Container für die Validierungskomponente

5.2. Verbindung via OPC

Für Verbindung zwischen RTU und Validierungskomponente wird die Python-Bibliothek $FreeOpcUa/opcua\text{-}asyncio$ verwendet. Einerseits dient sie bereits der Kommunikation zwischen Local- und Neighborhood-Monitoren und hat sich hierbei als zuverlässig und sicher herausgestellt. Andererseits ist sie einfach implementierbar und flexibel für die Zwecke innerhalb dieser Arbeit einsetzbar. Es handelt sich hierbei um eine Client-Server-Architektur, wobei die Validierungskomponente, wie im Konzept beschrieben, als Server und das Testbed als Client dient.

5.2.1. Server

Für die Vorkonfiguration des Servers wird die Datei $validation\_setup.py$ erstellt. Die $opc\_validation.py$ soll der eigentliche OPC-Server inklusive Validierungsmethoden werden. In der Main Methode der $opc\_validation.py$ wird der kommende Server um die Datenstrukturen “SwitchData”, “OtherData” und “RTUData” für die Übergabe aller RTU-Daten

an den Validierungserver, “PhysicalViolationSensor”, “PhysicalViolation” und “PhysicalViolations” für die Übergabe der Verstöße gegen die physikalischen Voraussetzungen von der RTU(IPS) an den Validierungsserver und außerdem “ValidationResult”, um das komplette Validierungsergebnis zusammenzufassen und an die RTU zurückzugeben, erweitert. Die drei Methoden “validate”, “return_zeros” und “return_physical_violations” werden als RemoteProcedures registriert und können so von dem Client aufgerufen werden. Der Inhalt der Methoden wird in Validierung genauer beschrieben. Zur Übersicht ist die $opc\_validation.py$ noch einmal in der Abbildung 5.1 dargestellt.

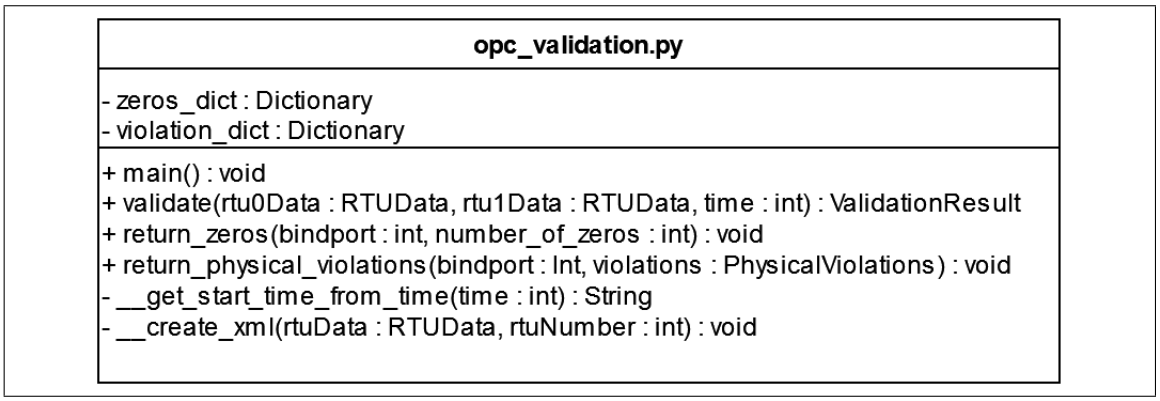

Abbildung 5.1.: Klassendiagramm der $opc\_validation.py$

5.2.2. Client

Die RTU sollte sowohl zur Validierung eines einkommenden Befehls als auch zur Übergabe eines Validierungsergebnisses als OPC-Client eine TCP Verbindung zu dem Validierungsserver aufbauen können. Hierzu wird in der $rtu.py$ jeweils ein OPC-Client erstellt, der sich mit dem Server verbindet. Um dies zu realisieren wurden die Methoden “__validate_commands” und “__return_result_to_validation” hinzugefügt. Beide Methoden erstellen zuerst einen Client und fragen beim Server die Datentypdefinitionen an. Die genaue Funktionsweise der Methoden wird in dem folgenden Kapitel genauer beschrieben.

5.3. Validierung

Zur Validierung einkommender Befehle an die RTU wird in dem darauf folgenden Simulationsschritt die Methode “__validate_commands” aufgerufen. Diese erstellt auf Grund der aktuellen RTU-Daten zwei “RTUData” Objekte. Danach wird die Methode “validate” mit den Objekten und der aktuellen Zeit als Parameter auf dem Validierungsserver aufgerufen. Ein Pseudocode der Methode ist am Ende des Abschnitts dargestellt (siehe Quelltext 5.2). Die Methode versieht die neu zu erstellenden Instanzen des Testbeds mit einer Id, welche für den Port und die spätere Zuordnung der Validierungsergebnisse dient. Nun wird zuerst für jede RTU(IPS) eine XML-Datei zur Konfiguration der Startwerte der jeweiligen RTU(IPS) inklusive Port erstellt (Zeilen 6 und 11). Dann wird in Zeile 16 eine neue Instanz des Testbeds mit den Parametern “in_val” und “start_time” erstellt. Der erste Parameter dient zur Unterscheidung zwischen der Validierungssimulation und anderer Simulation. Der zweite Parameter definiert den Startzeitpunkt an dem die Simulation beginnen soll. Das Testbed konfiguriert, wie in der Simulation eines physischen Netzes auch, die verschiedenen Komponenten unter anderem durch das Laden der vorher geschriebenen Konfigurationsdatei. Während der Simulation werden in der RTU(IPS) in jedem Schritt die physikalischen Voraussetzungen geprüft und die Sensoren mit dem Wert 0.0 gezählt. Für die Überprüfung wurden die Methoden “_check_req_3” und “_check_req_7_and_8” und die globalen Variablen “physical_violations” und “global_zero_counter” in der RTU(IPS) implementiert. Die Methoden prüfen die aktuellen Werte, die der RTU(IPS) vorliegen auf die entsprechenden Verstöße und speichern diese in dem Dictionary “physical_violations”. Der global_zero_counter repräsentiert immer die maximale Anzahl an Null-Sensoren unter allen Schritten.

Die Übergabe der gesammelten Ergebnisse erfolgt nach der Simulation durch die zuvor erwähnten Methoden “return_zeros” und “return_physical_violations”. Hierzu wird das Dictionary erst in den Datentyp “PhysicalViolations” umgewandelt, da dieser erst nach erfolgreichem Verbindungsaufbau zu dem Server verfügbar ist und OPC-UA nicht vorsieht, Dictionaries zu übergeben. Danach werden das erstellte Objekt und der Zähler zusammen mit dem Port der RTU(IPS) an die Validierungskomponente geleitet. Die Methoden

“return_zeros” und “return_physical_violations” speichern die übergebenen Daten in Dictionaries, wobei der Port als Schlüssel dient. Nachdem die Simulation durchlaufen und die Daten übergeben wurden, wird in der Methode “validate” auf die zuvor befüllten Dictionaries mittels dem zugewiesenen Port zugegriffen. Diese werden umstrukturiert, für die Rückgabe an die RTU aufbereitet und in ein neues “ValidationResult” Objekt geschrieben. Dies ist in den Zeilen 22 bis 32 zu sehen. Danach wird das Ergebnis an die RTU zurückgegeben (Zeile 35), welche das Ergebnis der Validierung entgegennimmt und anhand der Null-Sensoren entscheidet, ob der Befehl ausgeführt werden sollte. Außerdem werden sämtliche Warnungen durch physikalische Verstöße und Stromausfälle ausgegeben, so dass Benutzer des Testbeds sie sehen und entsprechend handeln können.

```
1    def validate(parent, rtu0Data, rtu1Data, time):
2        global port, zeros_dict, violation_dict
3
4        if (rtu0Data is not empty):
5            bindport1 = port
6            __create_xml(rtu0Data, 0, "192.168.0.19", bindport1)
7            port += 1
8
9        if (rtu1Data is not empty):
10           bindport2 = port
11           __create_xml(rtu1Data, 1, "192.168.0.19", bindport2)
12           port += 1
13
14       while ports are in bind or allready tried 5 times:
15           # run simulation in IPS mode
16           test_scenario.main(True, __get_start_time_from_time(time))
17           break
18       if (after 5 times trying port is bind):
19           throw error
20
21       # after running simulation map the result and send it to the RTU
22       result = ua.ValidationResult()
23       if (rtu0Data is not empty):
24           result.physical_violations = violation_dict[bindport1]
25           result.zero_sensors = zeros_dict[bindport1]
26
27       if (rtu1Data is not empty):
28           result.physical_violations = violation_dict[bindport2]
29           result.zero_sensors = zeros_dict[bindport2]
30
31       delete violation_dict[bindport1, bindport2]
32       delete zero_dict [bindport1, bindport2]
33
34       # successfull validation return result
35       return result
```

Quelltext 5.2: Pseudocode der Methode “validate” der Validierungskomponente

5.4. Anpassung des Stromnetzes

Um eine bessere Untersuchung des Netz zu ermöglichen, wurde das vorher verwendete Stromnetz angepasst. Es wurden einige Sensoren und Busse entfernt und ein neuer Switch hinzugefügt, welcher für eine erhöhte Steuerbarkeit im Netz sorgt. Hierzu mussten die verschiedenen Konfigurationsdateien, welche im Testbed beschrieben wurden, angepasst werden. Eine Anpassung der Haushalte wäre zu aufwendig gewesen, so dass unverhältnismäßig viele Haushalte mit einem Branch verbunden sind. Dies verändert allerdings nicht die Aussagekraft der Szenarien, weil es zwar eine Auswirkung darauf hat, wann es beispielsweise zu Überlastungen kommt. Es verändert aber nicht die Bedeutung, dass eine Validierung unter vorgegebenen Regeln möglich ist. Der für die Arbeit relevante Teil des angepassten Netzes ist in Abbildung 5.2 zu sehen.

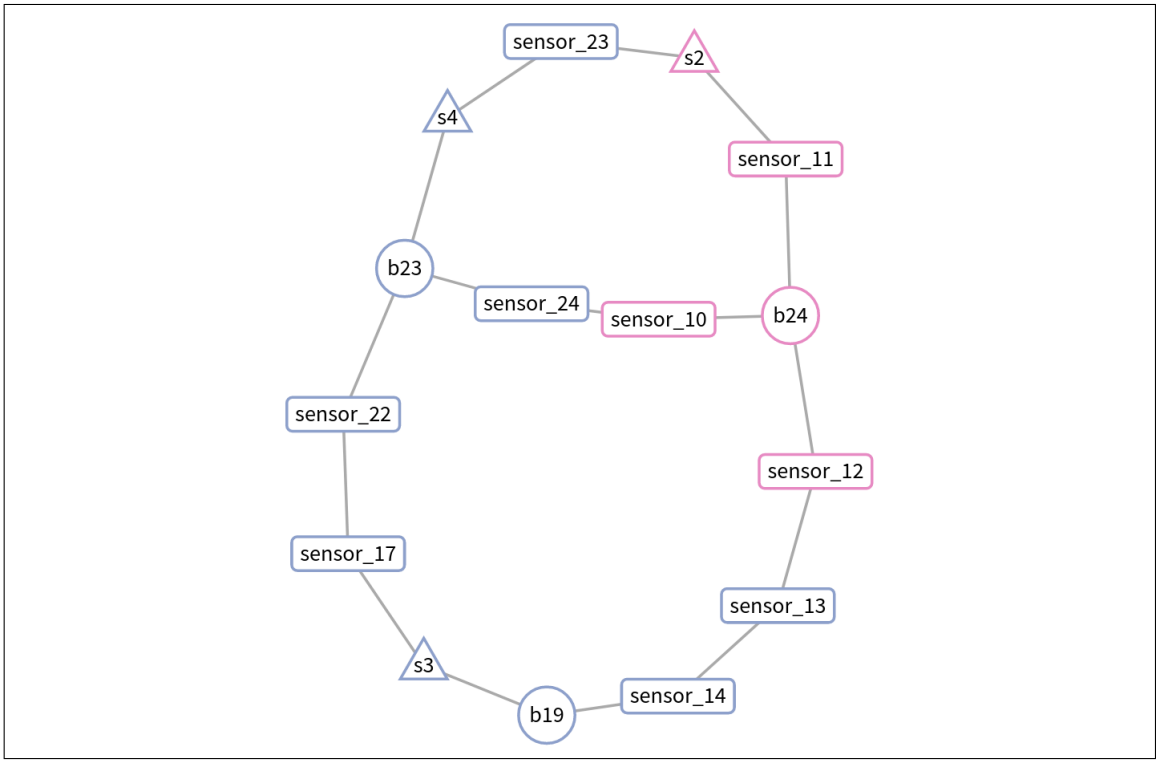

Abbildung 5.2.: Aufbau des von dem Intrusion Detection System überwachten und angepassten Netzes, dargestellt durch die Visualization aus dem Projektseminar ($RTU0$ in blau und $RTU1$ in rot dargestellt)

6 Evaluation

Um die Implementierung zu evaluieren wurden verschiedene Testszenarien entwickelt, welche im Folgenden angesprochen werden. Außerdem kam es zu verschiedenen Problemen, die es auszuwerten gilt.

6.1. Testszenarien

Die Durchführung der im Konzept aufgestellten Szenarien wird in den einzelnen Unterkapiteln beschrieben.

6.1.1. Normaler Simulationsdurchlauf

In dem ersten Szenario wird die Simulation normal gestartet und ohne Veränderung bis zum Ende ausgeführt. Bei einem normalen Simulationsdurchlauf wird nur beim Starten des Testbeds eine Verbindung zu der Validierungskomponente aufgebaut. Ihr werden die Startwerte übermittelt und die Simulation zur Validierung wird einmal durchlaufen. Die Rückgabe hat keinen Einfluss auf das Testbed. Dieses läuft ohne Einschränkungen bis zum Ende der Simulation.

6.1.2. Stromausfall in einem kleinen Bereich

Bei dem zweiten Szenario wird ein Befehl zum Umlegen eines Switches von der Attack-Engine an die RTU gesendet. Dies hat nur Auswirkungen auf eine kleine konstante Anzahl an Sensoren, in diesem Fall zwei Sensoren auf einer Leitung. Die RTU baut daraufhin eine Verbindung zu dem Validierungsserver auf und übermittelt die Daten der RTU. Die Validierungskomponente speichert diese Daten in die XML-Datei, welche im Kapitel 3.1 zu dem Thema Testbed und Konfigurationsdateien bereits erwähnt wurde, und startet eine Instanz des Testbeds. Nach der vorkonfigurierten Anzahl an Simulationsschritten, in allen Szenarien waren es 20 Schritte, baut diese Instanz eine Verbindung zu der Validierungskomponente auf und übermittelt, dass es zwei Sensoren ohne Strom im Netz gibt. Die übergibt der RTU somit, dass es zwei Sensoren ohne Strom geben wird, wenn

der Befehl ausgeführt wird. Dies wird dem Nutzer über die Konsole mitgeteilt, wie in Abbildung 6.1 zu sehen. Der Switch wird im nächsten Schritt von der RTU umgelegt und die Simulation wird bis zum Ende fortgesetzt.

```
mosaik_testbed          |
mosaik_testbed          | -----WARNING: HERE IS A LOCAL BLACKOUT-----
mosaik_testbed          |
```

Abbildung 6.1.: Warnung des Testbeds an den Nutzer

6.1.3. Blackout eines Subgrids

Das dritte Szenario beginnt wie das vorherige Szenario. Hier wird zuerst auch ein Befehl zum Umlegen eines Switches an die RTU gesendet, der nur Auswirkungen auf eine Leitung mit zwei Sensoren hat. Danach wird allerdings ein weiterer Befehl gesendet und die Warnung ignoriert, welche von der Validierung zurückgegeben wurde. Dieser Befehl soll das Umlegen eines zweiten Switches bewirken, so dass auf zwei Leitungen mit insgesamt drei Sensoren kein Strom fließen würde. Bei dem zweiten Befehl baut die RTU wieder eine Verbindung zu dem Validierungsserver auf und dieser startet erneut eine Instanz des Testbeds. Dieses Mal gibt die Instanz eine drei zurück, welche an die RTU weitergeleitet wird. Diese stellt fest, dass das vorgegebene Maximum, der Sensoren, die auf Null fallen dürfen, überschritten werden würde. Der Befehl wird dem entsprechend nicht ausgeführt und dem Nutzer zusätzlich mitgeteilt.

```
mosaik_testbed          |
mosaik_testbed          | -----ERROR: COMMAND NOT EXECUTED, BLACKOUT DETECTED-----
mosaik_testbed          |
```

Abbildung 6.2.: Error des Testbeds an den Nutzer

6.1.4. Physikalische Verstöße

In dieser Sektion wurden mehr Photovoltaik-Anlagen hinzugefügt, was eine höhere Strommenge zur Folge hat. Ein Szenario, in dem spezifisch ein Verstoß durch Umlegen eines Switches provoziert wird, konnte nicht erstellt werden, da dies einen größeren Eingriff in das vorhandene Netz erfordert hätte und somit nicht in dem Rahmen dieser Arbeit möglich ist. Die Abbildung 6.3 zeigt die physikalischen Verstöße, zu denen es in der Validierung

kam. Zu den Verstößen gegen REQ3 (in der Abbildung 6.3 Physical violation S3 in blau) kommt es, da die Simulation mit einem offenen Switch in der RTU gestartet wird, dieser aber noch nicht in der Simulation geöffnet wurde. Nach dem ersten Simulationsschritt ist der Switch auch in der Simulation offen, die Sensoren messen keinen Strom mehr und es kommt nicht zu weiteren Verstößen. Es ist also ein Problem der Simulation und kein realer Verstoß. Die Verstöße, welche durch die Photovoltaik-Anlagen provoziert wurden, sind in der Abbildung in rot dargestellt.

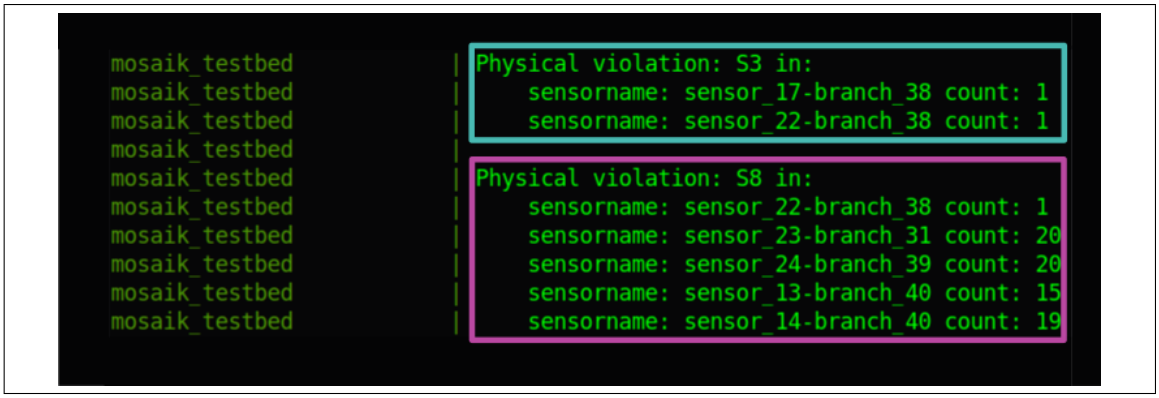

Abbildung 6.3.: Warnung des Testbeds an den Nutzer

6.1.5. Nicht-Erreichbarkeit der Validierungskomponente

Das fünfte und letzte Szenario dient zur Überprüfung der Ergebnisse, die bei Nicht-Erreichbarkeit der Validierungskomponente eintreten würden. Die Durchführung erfolgt, indem der Docker-Container mit der Validierungskomponente nicht gestartet wird. Es wird am Anfang und bei jedem Befehl, den die RTU erhält, bis zu fünf mal mit einem Abstand von fünf Sekunden versucht, eine Verbindung zum Validierungsserver aufzubauen. Nach fünf gescheiterten Versuchen wird der Befehl ausgeführt. Die 25 Sekunden wirken sich nur auf die Ausführung des Befehls und nicht auf die Simulationszeit aus, da Mosaik im Hintergrund weiter arbeitet und nicht auf Fertigstellung dieses RTU-Schrittes wartet.

6.1.6. Ergebnis der Implementation

Wie durch die Szenarien zu sehen ist, wurden die im Konzept entwickelten Punkte erfolgreich implementiert. Es wurde zuerst ein neuer Docker-Container, der als Validierungsserver dient, erstellt. Danach war es möglich, eine Methode vom Client aus auf dem Server aufzurufen, welche die Daten für das Testbed aktualisiert und eine Instanz des Testbeds startet. Diese hat nach dem Durchlauf das Ergebnis an die Validierungskomponente zurückgegeben, welche das Ergebnis danach aufbereitet hat. Diese Daten wurden dann dem Testbed übermittelt und dort dem Nutzer ausgegeben. Je nach Anzahl der Null-Sensoren wird der Befehl ausgeführt oder verweigert. Das vorher im Konzept aufgestellte Sequenzdiagramm (vgl. Abb. 4.1) beschreibt also auch nach der Implementation den Kommunikationsfluss zwischen den verschiedenen Komponenten nach dem Senden eines Befehls an die RTU.

6.2. Probleme

Während der Implementierung sind verschiedene Probleme aufgetreten, die weitere Handlungen erforderten. Im Folgenden werden die relevanteren Probleme und der Umgang mit ihnen erläutert.

6.2.1. Keine weitreichenden Vorhersagen

Es existiert nur ein gewisses Zeitfenster zwischen zwei Schritten einer Simulation, in dem eine Validierung der Befehle stattfinden kann. Durch die Beschleunigung von Mosaik mittels Wegnahme des Realtimefactors ist es möglich, in der Validierungskomponente Vorhersagen durch die Simulation zu treffen. Dies ist in Abbildung 6.4 am Beispiel eines Sensors zu sehen. Dennoch ist das Fenster sehr gering und von zwei verschiedenen Faktoren abhängig. Zum einen ist es relevant, ob der Befehl an die RTU zum Anfang oder gegen Ende des Zeitfensters zwischen zwei Schritten gesendet wird. Denn sobald der nächste Schritt beginnt, kann das Ergebnis der Validierung nicht mehr berücksichtigt werden. Der andere entscheidende Punkt ist, dass das Zeitfenster abhängig davon ist, wie gestresst das Netz ist. Je mehr Photovoltaik-Anlagen im Netz sind, desto häufiger werden

die Sensoren aktualisiert. Sobald es zu einer Aktualisierung kommt, kann das Ergebnis der Validierung nicht mehr berücksichtigt werden. Eine mögliche Lösung für dieses Problem wäre, nachdem ein Befehl gesendet wurde, diesen zu der Validierungskomponente zu leiten und zu verweigern. Danach kann regelmäßig bei dem Validierungsserver angefragt werden, ob es Validierungsergebnisse gibt. Der Server könnte nach der Validierung diese Ergebnisse zusammen mit dem Befehl bereitstellen und der Befehl kann im Nachhinein ausgeführt werden. So wäre man nicht auf Zeitfenster angewiesen. Allerdings wäre der Befehl zu einem anderen Zeitpunkt getestet worden, als er tatsächlich ausgeführt wird und es müsste auch berücksichtigt werden, was passiert, wenn die Validierungskomponente nicht antwortet, da die RTU per se alle Befehle ablehnen würde und das Netz nicht mehr steuerbar wäre. Es besteht auch die Möglichkeit, das Zeitfenster zu erhöhen, indem die Geschwindigkeit der Simulation mittels des Realtimefactors aus Mosaik reduziert wird. Dies würde eine langsamere Simulation, aber auch eine höhere Wahrscheinlichkeit der erfolgreichen Validierung zur Folge haben und könnte zur Veranschaulichung expliziter Fälle dienen. In der Umsetzung an einem echten Stromnetz käme diese Lösung nicht zum Einsatz, da das physische Netz nicht verlangsamt werden kann.

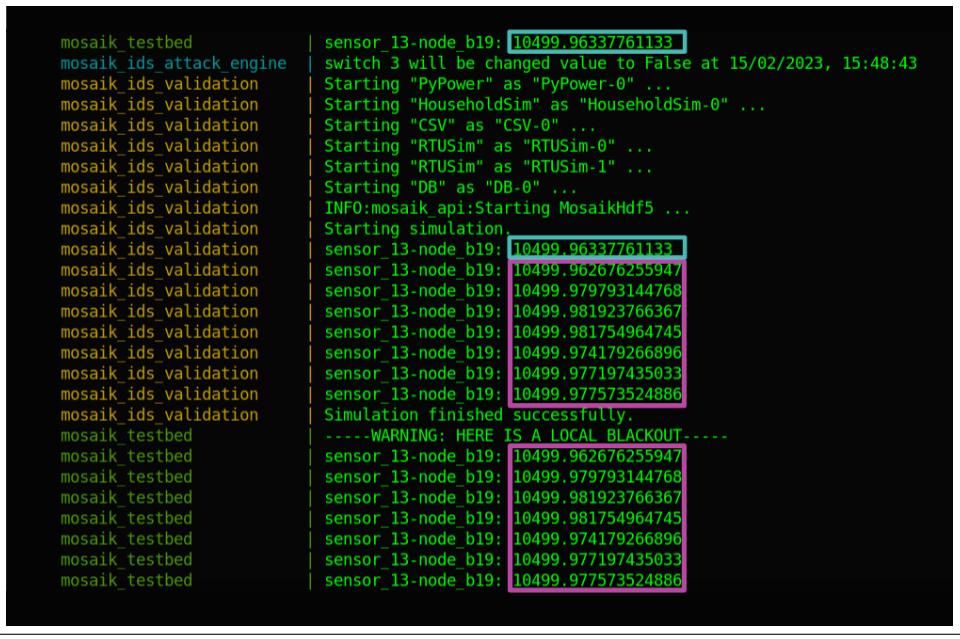

Abbildung 6.4.: Vorhersagen Beispielhaft an einem Sensor

6.2.2. Verbindung zwischen RTU und Validierung

Für die Verbindung zwischen der RTU und der Validierungseinheit wurde das TCP verwendet. Ein großer Vorteil des TCP ist der einmalige Aufbau der Verbindung, die während der ganzen Simulation genutzt werden kann. Dieses Vorgehen sorgt für eine schnellere Laufzeit und mehr Sicherheit. Allerdings war es nicht umsetzbar, eine Verbindung aufzubauen und diese zu halten, da die verwendete Bibliothek “opcua-asyncio” eine gewisse Asynchronität vorsieht. Wie bereits erwähnt, arbeitet Mosaik mit Simulationsschritten. Mosaik verwendet hierfür eine feste Struktur und arbeitet mit synchronen Methoden. Um aber die Verbindung von der RTU zum Server aufrecht zu erhalten, müssen die Methoden, in denen der Client verwendet wird, als asynchron deklariert werden, auch wenn sie nicht asynchron verwendet werden. Andersherum können synchrone Prozesse von der Bibliothek nicht asynchron aufgerufen werden. Also müsste für eine dauerhafte Verbindung die $Step$-Methode asynchron gemacht werden. Diese Deklaration würde dazu führen, dass die Methoden von Mosaik nicht erkannt werden und Mosaik nicht lauffähig ist. Kurzum, es ist es nicht möglich mit “opcua-asyncio” die Verbindung zum Server aufrecht zu erhalten und eine funktionsfähige Implementierung einer Mosaik-Simulation für die RTU zu haben. Stattdessen wird innerhalb dieser Arbeit auf eine dauerhafte Verbindung und somit auf einen großen Vorteil von TCP verzichtet und eine geringfügig längere Laufzeit in Kauf genommen, welche bei ersten Tests nicht von großer Relevanz war.

7 Fazit

Nachfolgend wird das Ergebnis der Arbeit zusammengefasst betrachtet, um anschließend Antworten auf die Forschungsfragen zu finden.

7.1. Zusammenfassung

Durch eine genauere Betrachtung des aktuellen Aufbaus von Stromnetzen wurde festgestellt, dass es keinen ausreichenden Schutz vor Angriffen gibt. Es werden zwar neue Technologien wie Smart-Grids eingesetzt, aber die bisher verwendeten und veralteten Protokolle sind weiterhin im Einsatz (siehe Abschnitte 2.1 und 2.2). Diese können, wenn Intrusion Prevention Systeme auf Netzwerk-Ebene versagen, dazu verwendet werden, das Stromnetz zu steuern und so entweder Schäden an Leitungen oder anderen Komponenten zu verursachen oder Stromausfälle herbeizuführen. Um den daraus resultierenden Schaden zu reduzieren, wurde das Konzept des Digital Twins betrachtet und die Idee übernommen, relevante Daten an eine externe Komponente zu übergeben, um so das Stromnetz virtuell abbilden zu können.

Das Testbed von Menzel et al. mit den Erweiterungen aus dem Projektseminar bildeten eine gute Grundlage, ein Konzept für eine Validierung zu entwickeln, indem neue Instanzen des Testbeds für die Validierung verwendet werden. Dieses Konzept ist, wie durch die Umsetzung der Implementierung zu sehen ist, aufgegangen und alle im Konzept aufgestellten Punkte 1 bis 8 (siehe Liste) konnten erfolgreich implementiert werden. Die Simulation läuft weiterhin stabil und wurde durch eine Validierungskomponente ergänzt. Die Attack-Engine hat es ermöglicht, die Validierungskomponente zu evaluieren. Es wurden verschiedene Testszenarien entwickelt und umgesetzt, die gezeigt haben, dass Vorhersagen über die Auswirkungen von Befehlen möglich sind. Zum einen können Überlastungen der Leitungen und zum anderen kleine oder große Stromausfälle erkannt werden. Wie in der Sektion Keine weitreichenden Vorhersagen beschrieben wurde, ist die Reichweite der Vorhersagen allerdings sehr eingeschränkt. Um diese zu erhöhen, wäre es nötig, das System auf

einem Server mit mehr Leistung aufzusetzen oder eine andere Art der Absprache, wie in der Sektion vorgeschlagen wurde, zu verwenden. Das zweite resultierende Problem ist durch Verwendung von Mosaik aufgetreten und hat in Bezug auf reale Stromnetze keine Relevanz für die Beantwortung der Forschungsfragen.

7.2. Beantwortung der Forschungsfragen

Im Folgenden wird versucht, eine Antwort auf die am Anfang formulierten Forschungsfragen auf Grund der in dieser Arbeit entwickelten Lösung zu finden.

Wie kann eine Validierung von Befehlen an eine RTU vor ihrer Ausführung in das bestehende IDS von Menzel et al. integriert werden?[1]

Eine Integration ist mittels eines Validierungsservers und einer angepassten Kopie des Testbeds denkbar. Dieser Ansatz wurde erfolgreich in der Arbeit umgesetzt und es wurde bewiesen, dass eine Validierung von Befehlen vor ihrer Ausführung in das IDS von Menzel et al. integriert werden kann.

Welche Angriffe auf ein Stromnetz können mit dieser Art der Validierung verhindert werden?

Es handelt sich bei der Implementation um einen Schutz nach dem Eindringen, vielleicht eine Art Intrusion Protection. Sie greift erst ein, wenn der Angreifer bereits in das System eingedrungen ist. Es dient daher dazu, größere Schäden zu vermeiden, wenn die Intrusion Prevention versagt hat. Man-in-the-Mittle-Angriffe durch die Attack-Engine können weiterhin ausgeführt werden, sie haben allerdings eine weniger verheerende Auswirkung auf das Netz und sind deutlich auffälliger. Wie in der Evaluation zu sehen war, werden Stromausfälle vermieden (siehe Abschnitt 6.1.3) und Schäden an Leitungen können durch die Anzeige im Testbed besser vorgebeugt werden (Abschnitt 6.1.4). Es ist also sowohl gelungen Angriffe, die einen größeren Stromausfall zur Folge hätten, abzuwehren, als auch solche zu erkennen, die zu Schäden am Stromnetz führen sollten.

Ist es möglich die Validierung auf ein echtes Netz zu übertragen und wie erfolgversprechend wäre eine Validierung durch eine Simulation?

Eine Übertragung auf das reale Stromnetz ist denkbar, da es auch bei dem echten Netz Zeitfenster zwischen der Übertragung der Sensordaten gibt und ggf. eine Anpassung wie im Abschnitt Keine weitreichenden Vorhersagen denkbar wäre. Eine Form der geschilderten Angriffe ist durchaus realistisch und so wäre ein weiterer Schutz durch die in der Arbeit geschilderte Form sinnvoll. Es sollte allerdings beachtet werden, die Kommunikation zwischen RTU und Validierungsserver sicher zu verschlüsseln, was mit der verwendeten Bibliothek “opcua-asyncio” möglich ist. Es könnte dort zusätzlich zu Man-In-The-Middle-Angriffen kommen und beispielsweise alle Befehle genehmigt oder verweigert und Warnungen unterdrückt oder fehlerhaft ausgegeben werden. Außerdem müsste der Server vor DDOS-Angriffen geschützt werden, weil ein Server der nicht erreichbar ist, auch nicht validieren kann. Auf die Steuerung hätte das, wie in dem Szenario Nicht-Erreichbarkeit der Validierungskomponente zu sehen, mit der Implementation aus dieser Arbeit keine Auswirkungen.

8 Ausblick

In dieser Arbeit wurde der erste Ansatz zu einer Validierung von Befehlen an eine RTU im Testbed von Menzel et al. implementiert. Es wurde eine Brücke zu einer externen Validierungskomponente geschaffen. Über diese Brücke werden bereits die jeweils aktuellen Daten der RTU übermittelt und es ist vordefiniert, welche Datenstruktur die RTU als Ergebnis erwartet. Durch diese offene Implementation ist es möglich, die Methode der Validierung zu erweitern oder zu ersetzen, so dass sich anschließende Forschung in diesem Bereich mit anderen Validierungsmöglichkeiten beschäftigen könnte. Eine denkbare Option wäre die Graphenanalyse oder mittels künstlicher Intelligenz der Versuch, Vorhersagen von Verbrauch und Produktion im kleinen Rahmen zu treffen. Die vorhergesagten Daten könnten dann an Stelle der fest definierten Daten für eine Simulation mittels Mosaik verwendet werden. Alternativ ist es auch denkbar eine KI nicht die Produktions- und Verbrauchsdaten sondern direkt die Sensordaten vorhersagen zu lassen und dort Verstöße zu erkennen. Hierbei könnte die Validierung durch Mosaik weiterhin als Referenz dienen.

Literatur

[1] V. Menzel, J. L. Hurink und A. Remke, „Securing SCADA networks for smart grids via a distributed evaluation of local sensor data,“ in *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2021, S. 405–411. DOI: 10.1109/SmartGridComm51999.2021.9632283.

[2] J. J. Chromik, A. Remke und B. R. Haverkort, „An integrated testbed for locally monitoring SCADA systems in smart grids,“ *Energy Inform*, 2018. DOI: https://doi.org/10.1186/s42162-018-0058-7. Adresse: https://energyinformatics.springeropen.com/articles/10.1186/s42162-018-0058-7.

[3] *Electric Grid Reliability and Interface with Nuclear Power Plants*, Ser. Nuclear Energy Series NG-T-3.8. Vienna: INTERNATIONAL ATOMIC ENERGY AGENCY, 2012, ISBN: 978-92-0-126110-6. Adresse: https://www.iaea.org/publications/8754/electric-grid-reliability-and-interface-with-nuclear-power-plants.

[4] *Smart-Grid Umweltbundesamt*, Accessed: 2023-01-27, 2013. Adresse: https://www.umweltbundesamt.de/service/uba-fragen/was-ist-ein-smart-grid.

[5] K. Sayed und H. Gabbar, „Chapter 18 - SCADA and smart energy grid control automation,“ in *Smart Energy Grid Engineering*, H. A. Gabbar, Hrsg., Academic Press, 2017, S. 481–514, ISBN: 978-0-12-805343-0. DOI: https://doi.org/10.1016/B978-0-12-805343-0.00018-8. Adresse: https://www.sciencedirect.com/science/article/pii/B9780128053430000188.

[6] M. Safiuddin, „HISTORY OF ELECTRIC GRID,“ in 2013, S. 6–11, ISBN: 978-1-60263-070-3.

[7] S. Borenius, H. Hämmäinen, M. Lehtonen und P. Ahokangas, „Smart grid evolution and mobile communications—Scenarios on the Finnish power grid,“ *Electric Power Systems Research*, Jg. 199, S. 107 367, 2021, ISSN: 0378-7796. DOI: https://doi.

org/10.1016/j.epsr.2021.107367. Adresse: https://www.sciencedirect.com/science/article/pii/S0378779621003485.

[8] O. Smith, O. Cattell, E. Farcot, R. D. O'Dea und K. I. Hopcraft, „The effect of renewable energy incorporation on power grid stability and resilience," *Science Advances*, Jg. 8, Nr. 9, eabj6734, 2022. DOI: 10.1126/sciadv.abj6734. eprint: https://www.science.org/doi/pdf/10.1126/sciadv.abj6734. Adresse: https://www.science.org/doi/abs/10.1126/sciadv.abj6734.

[9] *FAQ Modbus Site*, Accessed: 2023-02-03. Adresse: https://modbus.org/faq.php.

[10] *Overview of DNP3 Protocol*, Accessed: 2023-02-03. Adresse: https://www.dnp.org/About/Overview-of-DNP3-Protocol.

[11] D. Bailey und E. Wright, *Practical SCADA for Industry*, D. Bailey und E. Wright, Hrsg. Oxford: Newnes, 2003, ISBN: 978-0-7506-5805-8. DOI: https://doi.org/10.1016/B978-0-7506-5805-8.X5000-4. Adresse: https://www.sciencedirect.com/book/9780750658058/practical-scada-for-industry.

[12] V. M. Igure, S. A. Laughter und R. D. Williams, „Security issues in SCADA networks," *Computers & Security*, Jg. 25, Nr. 7, S. 498–506, 2006, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2006.03.001. Adresse: https://www.sciencedirect.com/science/article/pii/S0167404806000514.

[13] M. DenHartog, „DNP3 Tutorial: Learn the Industry-Standard SCADA Protocol," 2012. Adresse: https://www.dpstele.com/pdfs/white_papers/dnp3_tutorial.pdf.

[14] M. Organization, „MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE V1.0b," 2006. Adresse: https://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf.

[15] *PyModbus Doc*, Accessed: 2023-01-27. Adresse: https://pymodbus.readthedocs.io/en/v3.1.1/readme.html#example-code.

[16] *Official Wireshark Site*, Accessed: 2023-02-22. Adresse: https://www.wireshark.org/.

[17] *DNP3 Protocol : Architecture, Working, Function codes, Data format & Its Applications*, Accessed: 2023-01-31. Adresse: https://www.elprocus.com/dnp3-protocol/.

[18] I. N. Fovino, A. Carcano, T. De Lacheze Murel, A. Trombetta und M. Masera, „Modbus/DNP3 State-Based Intrusion Detection System," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, S. 729–736. DOI: 10.1109/AINA.2010.86.

[19] M. Majdalawieh, F. Parisi-Presicce und D. Wijesekera, „DNPSec: Distributed Network Protocol Version 3 (DNP3) Security Framework," in *Advances in Computer, Information, and Systems Sciences, and Engineering*, K. Elleithy, T. Sobh, A. Mahmood, M. Iskander und M. Karim, Hrsg., Dordrecht: Springer Netherlands, 2006, S. 227–234, ISBN: 978-1-4020-5261-3.

[20] *MODBUS/TCP Security Protocol Specification*, Accessed: 2023-02-18. Adresse: https://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf.

[21] R. R. R. Barbosa, R. Sadre und A. Pras, „A first look into SCADA network traffic," in *2012 IEEE Network Operations and Management Symposium*, 2012, S. 518–521. DOI: 10.1109/NOMS.2012.6211945.

[22] M. F. Venkat Pothamsetty (Cisco), *Transparent Modbus/TCP Filtering with Linux*, Accessed: 2023-01-30. Adresse: https://modbusfw.sourceforge.net/.

[23] J. Nivethan und M. Papa, „A Linux-based firewall for the DNP3 protocol," in *2016 IEEE Symposium on Technologies for Homeland Security (HST)*, 2016, S. 1–5. DOI: 10.1109/THS.2016.7568963.

[24] J. Jow, Y. Xiao und W. Han, „A survey of intrusion detection systems in smart grid," *International Journal of Sensor Networks*, Jg. 23, S. 170–186, 2017. DOI: 10.1504/IJSNET.2017.083410.

[25] M. M. H. Sifat u. a., „Towards electric digital twin grid: Technology and framework review," *Energy and AI*, Jg. 11, S. 100 213, 2023, ISSN: 2666-5468. DOI: https://

doi.org/10.1016/j.egyai.2022.100213. Adresse: https://www.sciencedirect.com/science/article/pii/S2666546822000593.

[26] H. Bai und Y. Wang, „Digital power grid based on digital twin: Definition, structure and key technologies," *Energy Reports*, Jg. 8, S. 390–397, 2022, ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2022.10.328. Adresse: https://www.sciencedirect.com/science/article/pii/S2352484722022612.

[27] S. A. Varghese, A. Dehlaghi Ghadim, A. Balador, Z. Alimadadi und P. Papadimitratos, „Digital Twin-based Intrusion Detection for Industrial Control Systems," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2022, S. 611–617. DOI: 10.1109/PerComWorkshops53856.2022.9767492.

[28] *Official Mosaik Website*, Accessed: 2023-01-21.

[29] A. Halu, A. Scala, A. Khiyami und M. C. González, „Data-driven modeling of solar-powered urban microgrids," *Science Advances*, Jg. 2, Nr. 1, e1500700, 2016. DOI: 10.1126/sciadv.1500700. eprint: https://www.science.org/doi/pdf/10.1126/sciadv.1500700. Adresse: https://www.science.org/doi/abs/10.1126/sciadv.1500700.

[30] J. Chromik, „Process-aware SCADA traffic monitoring: A local approach," English, Diss., University of Twente, Netherlands, 2019, ISBN: 978-90-365-4801-4. DOI: 10.3990/1.9789036548014.

Eidesstattliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit über *“Lokale Intrusion Prevention über eine simulations-basierte Validierung von Befehlen im Kontext von Smart Grids”* selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind.

Daniel Krug, Münster, 6. März 2023

Ich erkläre mich mit einem Abgleich der Arbeit mit anderen Texten zwecks Auffindung von Übereinstimmungen sowie mit einer zu diesem Zweck vorzunehmenden Speicherung der Arbeit in eine Datenbank einverstanden.

Daniel Krug, Münster, 6. März 2023