Bundesamt
für Sicherheit in der
Informationstechnik

Deutschland
Digital•Sicher•BSI•

Sicherheitsanalyse OPC UA

Stand: 01.06.2022

Änderungshistorie

Version	*Datum*	*Beschreibung*
1.0	01.02.2022	Veröffentlichung
1.1	01.04.2022	Korrektur von fehlerhaften Verweisen
1.2	01.06.2022	Korrektur von Verweisen und Formatierung

Tabelle 1: Beschriftung

Bundesamt für Sicherheit in der Informationstechnik Postfach 20 03 63
53133 Bonn
Tel.: +49 22899 9582-0
E-Mail: ics-sec@bsi.bund.de
Internet: https://www.bsi.bund.de

Danksagung

Das BSI dankt den Autoren der Studie:

M.Sc. Johannes vom Dorp

Gruppenleiter *Applied System Analysis* bei Fraunhofer FKIE

M. Sc. Sven Merschjohann

Wissenschaftlicher Mitarbeiter bei Fraunhofer IEM

M. Sc. David Meier

Wissenschaftlicher Mitarbeiter bei Fraunhofer IOSB

M. Sc. Florian Patzer

Wissenschaftlicher Mitarbeiter bei Fraunhofer IOSB

M. Sc. Markus Karch

Wissenschaftlicher Mitarbeiter bei Fraunhofer IOSB

Dr.-Ing. Christian Haas

Gruppenleiter *Industrielle Cybersicherheit* bei Fraunhofer IOSB

Inhalt

1 Einleitung

OPC UA, die Open Platform Communications Unified Architecture, ist ein offener Kommunikationsstandard, mit dem jegliche Art von Maschinen im industriellen Kontext miteinander kommunizieren können. Einzelne Sensoren bis zu kompletten Produktionsstraßen, Kraftwerken oder Ölplattformen werden in einem serverseitigen Informationsmodell abgebildet und von einer Client-Software angesteuert. So werden Datenaustausch, intelligente Datenauswertung und Prozesssteuerung möglich. Als herstellerübergreifender Standard gilt OPC UA als eine Basis-Technologie für Industrie-4.0-Anwendungen. OPC UA bietet im Vergleich zu vielen anderen industriellen Kommunikationsprotokollen integrierte Sicherheitsmechanismen für die authentisierte, integritätsgeschützte und ggf. verschlüsselte Kommunikation, sowie Mechanismen, um Anwendungen oder Nutzer für den Zugriff auf die entsprechenden Informationsmodelle zu autorisieren. Die spezifizierten Sicherheitsmechanismen sind grundsätzlich geeignet, um eine sichere Kommunikation nach Stand der Technik zu gewährleisten. Dies wurde bereits 2016 in einer im Auftrag des BSI durchgeführten Studie für die OPC UA Version 1.02 untersucht und bestätigt. Seit dieser Studie aus dem Jahr 2016 ergaben sich sowohl in der OPC UA-Spezifikation als auch in Bezug auf die damals untersuchte ANSI C Implementierung der OPC Foundation große Änderungen. OPC UA ist mittlerweile in der Version 1.04 spezifiziert, die damalige ANSI C Implementierung der OPC Foundation ist mittlerweile lediglich als Legacy-Version verfügbar. Gleichzeitig werden immer mehr Produkte und Anwendungen entwickelt und angeboten, die OPC UA unterstützten.

Im Rahmen dieser Veröffentlichung wurde eine Aktualisierung der Studie aus dem Jahr 2016 auf Basis der OPC UA Version 1.04 und für die quelloffene C-Implementierung open62451 durchgeführt. Die Methodik der Studie aus 2016 wurde weitestgehend beibehalten. Die Untersuchung beschränkte sich dabei auf OPC UA Client/Server. Zusätzlich wurde im Rahmen einer Marktbefragung untersucht, welche Sicherheitsfunktionen in OPC UA Produkten und Anwendungen vorzufinden sind und welche Probleme Entwickler bei der Umsetzung des OPC UA Standards vorfanden.

Folgende Ergebnisse sind dabei entstanden:

- **Bestandsaufnahme des Kenntnisstands bzgl. der IT-Sicherheit von OPC UA**: Für diese Studie wurden 36 Veröffentlichungen begutachtet. Der überwiegende Teil der identifizierten Veröffentlichungen bezieht sich auf die Anwendung von OPC UA für verschiedene Anwendungsfälle und besitzt dabei zwar einen Security-Fokus, kann jedoch nicht unmittelbar zur Verbesserung des Standards beitragen. Neun Beiträge wurden jedoch wegen ihrer Kritik am OPC-UA-Standard und dem aktuellen Stand der Technik als relevant erachtet. Fünf dieser Beiträge motivieren direkt problematische Aspekte der Spezifikation, die weiter betrachtet wurden.
- **Umsetzungsstand Handlungsempfehlungen von 2016**: Es wurde überprüft, ob die Handlungsempfehlungen aus der Studie 2016 von der OPC Foundation umgesetzt wurden. Der größte Teil der Handlungsempfehlungen wurde vollständig oder in Teilen umgesetzt. Einige Handlungsempfehlungen wurden neu bewertet und als Änderungsvorschläge an die OPC Foundation übergeben.
- **Neue Handlungsempfehlungen auf Basis der OPC UA Version 1.04**: Insgesamt wurden 192 redaktionelle Anmerkungen und Handlungsempfehlungen identifiziert und mit der Arbeitsgruppe für IT-Sicherheit der OPC Foundation besprochen. Das Ergebnis dieser Diskussionen ist, dass 171 von den insgesamt 192 gemeldeten redaktionellen Anmerkungen und Handlungsempfehlungen bestätigt wurden und dementsprechende Einträge im Verwaltungssystem „Mantis“ der OPC Foundation erstellt wurden.
- **Analyse der Schutzmechanismen auf Parameterebene**: Die in der Studie aus dem Jahr 2016 gewonnenen Schlussfolgerungen zum Client/Server-Kommunikationsparadigma mit aufgebauter Session sind mit der Version 1.04 des OPC UA Standards weiterhin aktuell.
- **Ergebnisse der Marktbefragung zum Umsetzungstand von OPC UA Security-Funktionen**: Im Rahmen der Marktbefragung wurden Hersteller von OPC UA Produkten und Anwendung mit Hilfe einer aus

100 Fragen bestehenden Umfrage zum Umsetzungsstand von OPC UA Security-Funktionen befragt. Insgesamt wurden 138 Beantwortungen gezählt, wobei davon 129 Teilnehmer über einen öffentlichen verfügbaren Link und 9 Teilnehmer über herstellerindividuelle Links für zertifizierte Produkte auf die Umfrage zugegriffen haben. Aus den Erkenntnissen zur Durchführung der Marktbefragung und Nutzerstudie, sowie den Ergebnissen daraus, können verschiedene Rückschlüsse auf sicherheitsrelevante Aspekte beim praktischen Einsatz von OPC UA gezogen werden. Insgesamt ist anzumerken, dass der Umsetzungsstand von Sicherheitsfunktionen zeigt, dass die im Markt verfügbaren Produkte der Spezifikation in sicherheitsrelevanten Bereichen noch nicht vollständig gerecht werden.

- **Dynamische Codeanalyse von open62541**: Die Sicherheit des OPC UA Protokolls in Version 1.04 wurde anhand von open62541 als zertifizierte Serverimplementierung auf drei Arten dynamisch untersucht. Es wurden zwei Fuzzing-Ansätze verfolgt, ein Blackbox- und ein Whitebox-Ansatz, sowie ein Test auf Zertifikatsvalidierung umgesetzt. Das Whitebox-Fuzzing hat einen reproduzierbaren Fehler in der open62541-Bibliothek identifiziert der gemeldet und vor Ablauf der Studie bereits behoben wurde.
- **Statische Codeanalyse von open62541**: Zur Analyse von open62541 wurden sowohl automatische Programme eingesetzt, als auch eine manuelle Codeanalyse für sicherheitskritische Bereiche der Implementierung durchgeführt. Als automatische Codeanalysetools kamen dabei Cppcheck, FramaC und Clang zum Einsatz. Zusammenfassend lässt sich festhalten, dass bei der Analyse keine schwerwiegenden Schwachstellen gefunden wurden und der Code allgemein auf einem sehr hohen Sicherheitsniveau ist. Alle gefundenen Punkte wurden dem open62541 Projekt gemeldet. Diese Punkte wurden entsprechend akzeptiert und werden in zukünftigen Versionen von open62541 ausgebessert.

2 Gegenstand der Analyse

OPC UA wurde von der Plattform Industrie 4.0 als eines der ersten Protokolle als Industrie 4.0 Standard gesetzt. OPC UA in der Variante des Client-Server-Kommunikationsparadigmas bietet im Vergleich zu anderen industriellen Kommunikationsprotokollen integrierte Sicherheitsmechanismen für die authentisierte, integritätsgeschützte und ggf. verschlüsselte Kommunikation, sowie Mechanismen um Anwendungen oder Nutzer für den Zugriff auf die entsprechenden Informationsmodelle zu autorisieren. Die spezifizierten Sicherheitsmechanismen sind grundsätzlich geeignet, um eine sichere Kommunikation nach Stand der Technik zu gewährleisten. Dies wurde bereits 2016 in einer im Auftrag des BSI durchgeführten Studie für die OPC UA Version 1.02 untersucht und bestätigt [3]. Neben der Analyse des Standards sowie einer durchgeführten Bedrohungsanalyse wurde zudem die ANSI C-Implementierung der OPC Foundation mittels statischer und dynamischer Codeanalyse inkl. Fuzzing untersucht.

Seit dieser Studie aus dem Jahr 2016 ergaben sich sowohl in der OPC UA-Spezifikation als auch in Bezug auf die von der OPC Foundation bereitgestellten Implementierungen große Änderungen. OPC UA ist mittlerweile in der Version 1.04 spezifiziert und als umfassendste Änderung ist hierbei sicherlich das neu hinzugekommene Kommunikationsparadigma PubSub zu nennen. Auch andere Teile des Standards, die beispielsweise die Zertifikatsverwaltung abdecken, haben größere Änderungen erfahren. Die ANSI C-Implementierung (untersuchte Implementierung der Studie aus 2016) ist mittlerweile lediglich als Legacy-Version verfügbar. Zudem erscheinen immer mehr Produkte mit (zertifizierter) OPC UA-Unterstützung, wobei nicht immer ersichtlich ist, welche Sicherheitsfunktionen über die Security Policies und User Identity Tokens jeweils umgesetzt worden sind. Dazu zählen beispielsweise die Unterstützung eines Global Discovery Server (GDS).

Auf Basis dieser Ausgangssituation ist es sinnvoll, die Studie aus 2016 zu aktualisieren und um neue Untersuchungsaspekte zu ergänzen. Ziel dieser Studie ist es daher, ausgehend von der Version 1.04 des OPC UA Standards, eine Aktualisierung der Studie aus 2016 zur Verfügung zu stellen.

Für die Aktualisierung der Studie wurden daher alle sicherheitsrelevanten Änderungen an der OPC UA Spezifikation von Version 1.02 auf 1.04 gesammelt und die neue Version der Spezifikation analog zur Studie aus 2016 auf Schwachstellen und Verbesserungen hin untersucht. In Abstimmung mit dem Bundesamt für Sicherheit wurde PubSub (Part 14) als neues Kommunikationsparadigma nicht im Rahmen der Aktualisierung betrachtet, da die großen Unterschiede zwischen den beiden Kommunikationsparadigmen Client/Server und PubSub eine Vergleichbarkeit der Ergebnisse nicht gewährleisten würde. Zum Zeitpunkt der Aktualisierung der Studie waren zudem keine vollständigen (oder zertifizierten) Implementierungen verfügbar. PubSub und die dafür notwendige Infrastruktur sollten daher zu einem späteren Zeitpunkt im Rahmen einer eigenen Analyse untersucht werden.

Im Rahmen der Aktualisierung der Studie soll eine offene (Open Source) Implementierung des OPC UA Protokolls mittels statischer und dynamischer Code-Analysemethoden untersucht werden. Es wurde open62541 [18] als offene Implementierung für die Untersuchung in der aktualisierten Studie ausgewählt. Es wird einerseits untersucht, ob Teile der 2016 identifizierten Schwachstellen gegebenenfalls auch in der neuen Implementierung zu finden sind, aber auch welche weiteren Auffälligkeiten bei den Tests auftreten. Gefundene Schwachstellen werden so weit wie möglich in Proof-of-Concept Exploits umgesetzt und Angriffe auf die Implementierung bzw. den Standard demonstriert.

Zusätzlich wird im Rahmen einer Marktsichtung untersucht, welche Sicherheitsmechanismen in den einzelnen Produkten umgesetzt wurden und welche Schwierigkeiten die Hersteller dabei vorfanden. Die Ergebnisse sollen in Handlungsempfehlungen zur sicheren Umsetzung bei Herstellern und Integratoren festgehalten werden.

3 Bestandsaufnahme des Kenntnisstands bzgl. der IT-Sicherheit von OPC UA

Seit der Studie im Jahr 2016 [3] wurden weitere Veröffentlichungen mit Bezug zur Sicherheit von und durch OPC UA vorgestellt. Dieses Kapitel vermittelt einen Überblick über die, im Kontext dieses Berichts, relevanten Publikationen. Eine Liste aller betrachteten Veröffentlichungen ist Anhang A: Liste betrachteter Veröffentlichungen aufgeführt.

3.1 Vorgehen

Im Rahmen einer Online-Recherche wurden 36 Veröffentlichungen begutachtet. Diese wurden bezüglich ihres potenziellen Beitrags zur Verbesserung des Standards evaluiert. Dabei fielen bereits 27 Beiträge aus der Betrachtung heraus, die verbleibenden konnten anhand ihrer Kritikpunkte bezüglich OPC UA kategorisiert werden. Dies wird in folgendem Abschnitt vorgestellt.

3.2 Relevante Beiträge

Im Folgenden werden Veröffentlichungen vorgestellt, die Kritikpunkte bezüglich verschiedener Security-Aspekte von OPC UA aufweisen. Tabelle 2 bietet eine Übersicht dieser Punkte.

Tabelle 2 Übersicht über Kritikpunkte und entsprechende Veröffentlichungen

Kritikpunkt	*Veröffentlichung*
Abbildbarkeit von Anwendungs-Security-Strategien auf das Security Model	[1]
Den Vorschlag aus u.a. Part 6 Anhang D Tabelle D2, SHA-1 einzusetzen (Wurde seit [3] jedoch behoben)	[2], [3]
OpenSecureChannel und CreateSession ermöglicht Man-in-the-Middle-Angriffe	[4]
Erweiterbarkeit und Separation-of-Concerns Authentifizierungs- und Autorisierungsschema	[5]
Einsatz von RBAC	[5], [7]
Vertrauensaufbau und -management	[6], [8]
Produktiv eingesetzte und öffentlich zugreifbare OPC UA Anwendungen mangelhaft konfiguriert.	[16]
Konfiguration der Vertrauenslisten in proprietären OPC UA Anwendungen und OPC UA Bibliotheken mangelhaft.	[17]

In [1] wurde die Abbildbarkeit von Anwendungs-Security-Strategien auf das Security Model von OPC-UA kritisiert. Da diese Arbeit bereits aus dem Jahr 2010 stammt, ist eine Einschätzung der aktuellen Relevanz ohne weitere Untersuchungen nicht möglich. Sie wurde demnach der Vollständigkeit halber aufgenommen.

In [2] und [3] wurde der Vorschlag des Einsatzes von SHA-1 kritisiert. Dieser Vorschlag wurde zwischenzeitlich entfernt, wodurch der entsprechende Kritikpunkt nicht mehr relevant ist.

In [4] wurde die Möglichkeit eines MitM-Angriffs auf den OpenSecureChannel trotz SignAndEncrypt beschrieben, der Replay-Angriffe ermöglicht und die Channel-Keys abfängt. Als Gegenmaßnahme wird empfohlen das Key-Wrapping für die Nonces einzusetzen. Allerdings steht im Standard Part 6, dass der AsymmetricKeyWrapAlgorithm nicht für OPC UA Secure Conversation (UASC) genutzt wird. In [4] wird auch ein Angriff auf das CreateSession-Protokoll gefunden, der mit Key-Wrapping verhindert werden kann. Das beschriebene Problem wird im Standard bisher nicht beachtet, da weder Key-Wrapping verpflichtend ist, noch genauer erläutert wird.

Allerdings ist nicht eindeutig erkennbar, ob alle Annahmen, die zur Herleitung der Kritik und zur Durchführbarkeit der Angriffe getroffen wurden, standardkonform sind. Hierfür ist eine tiefergehende Untersuchung notwendig.

In [5] wurde unter anderem bemerkt, dass die Hinzunahme eines zusätzlichen Security-Attributs durch die bereits belegten 32-Bit-Masken von NodeClassType, WriteMask und UserWriteMask nicht möglich ist. Auch wurde der Einsatz von RBAC als ungeeignet bewertet, da dies nicht mit den dynamischen Anforderungen modernen Anwendungsfälle vereinbar ist. Dies wurde auch in [7] negativ interpretiert, wobei auch eine Inkonsistenz im Standard adressiert wurde, die dadurch entsteht, dass der Standard zwar feingranulare Autorisierungsregeln empfiehlt, das Autorisierungskonzept von OPC UA diese aber nicht ermöglicht. Zudem bemängelten die Autoren von [5] die fehlende Trennung von Nutzerautorisierung und -authentifizierung, sowie von Informations- und Autorisierungsmodell. Diese Kritikpunkte sollten zukünftig im Standardisierungsgremium diskutiert werden.

Die Veröffentlichungen [6] und [8] bemängelten die fehlende Spezifikation eines sicheren Vertrauensaufbaus und -managements. Zwar wird der Einsatz einer Public Key Infrastruktur (PKI) vom Standard empfohlen, die Integration dieser PKI wird jedoch offengelassen. Dabei ist sowohl unklar, wie Clients und Server mit Zertifikaten provisioniert werden, als auch, wie die Integration der PKI-Dienste in die OPC UA Architektur umgesetzt werden soll. Für letzteres definiert der Standard in Part 12 die Übernahme einzelner PKI-Dienste durch den Global Discovery Server (GDS). Hierzu fehlt es momentan zudem noch an praxistauglichen Implementierungen. In Abschnitt 5.3 werden zudem relevante Standardisierungslücken adressiert.

Verhältnismäßig neue Veröffentlichungen kritisieren zudem die mangelhafte Konfiguration öffentlich zugreifbarer OPC UA Anwendungen [16] und der Vertrauenslisten in proprietären OPC UA Anwendungen und OPC UA Bibliotheken [17]. Dies bezieht sich zwar nicht auf die Sicherheit der Spezifikation, vermittelt jedoch einen Eindruck des Stands der Technik. Oft wird demnach beim Einsatz von OPC UA das Thema Security weiterhin nicht hinreichend betrachtet.

3.3 Zusammenfassung der Ergebnisse

Für diese Studie wurden 36 Veröffentlichungen begutachtet. Der überwiegende Teil der identifizierten Veröffentlichungen bezieht sich auf die Anwendung von OPC UA für verschiedene Anwendungsfälle und besitzt dabei zwar einen Security-Fokus, kann jedoch nicht unmittelbar zur Verbesserung des Standards beitragen. Neun Beiträge wurden jedoch wegen ihrer Kritik am OPC-UA-Standard und dem aktuellen Stand der Technik als relevant erachtet. Fünf dieser Beiträge motivieren direkt problematische Aspekte der Spezifikation, die weiter betrachtet werden sollten. Unter anderem bezüglich dieser Kritik, werden in Abschnitt 4.3 Neue Handlungsempfehlungen vorgeschlagen.

4 Redaktionelle Anmerkungen, sicherheitsrelevante Spezifikationsänderungen und Handlungsempfehlungen

In der 2016 Studie wurden verschiedene redaktionelle Anmerkungen und Handlungsempfehlungen basierend auf den verschiedenen Spezifikationsteilen erarbeitet und präsentiert. In diesem Kapitel werden zunächst diese Anmerkungen und Empfehlungen noch einmal betrachtet und überprüft, ob und wie die aufgezeigten Hinweise durch die OPC Foundation adressiert wurden.

Da die Studie 2016 auf Grundlage der Version 1.02 der Spezifikation erstellt wurde, werden anschließend alle seitdem vorgenommenen, sicherheitsrelevanten Änderungen an der Spezifikation betrachtet.

Auf der Basis dieser Änderungen wurden neue Anmerkungen und Empfehlungen erstellt, die im Abschluss dieses Kapitels präsentiert werden.

4.1 Umsetzungsstand Handlungsempfehlungen

In der Studie von 2016 wurden 60 auf die Spezifikation bezogene, redaktionelle Anmerkungen erstellt. Im Rahmen dieser Analyse wurden diese Anmerkungen erneut betrachtet, sowie der Umsetzungsstand und Einschätzung der OPC Foundation analysiert.

4.1.1 Vorgehen

Die in der Studie von 2016 identifizierten redaktionellen Anmerkungen und Handlungsempfehlungen wurden im ersten Schritt durch ein manuelles Review auf ihren Umsetzungsstand hin überprüft. Hierzu wurde die OPC UA Spezifikation in der aktuellen Version 1.04 (Stand: 18.08.2020) herangezogen. Anschließend wurden alle identifizierten redaktionellen Anmerkungen und Handlungsempfehlungen, die in der aktuellen OPC UA Spezifikation nicht adressiert wurden, erneut mit der IT-Sicherheitsgruppe der OPC Foundation diskutiert.

4.1.2 Überblick zum Umsetzungsstand der Handlungsempfehlungen

Tabelle 3 gibt eine Übersicht über die Anzahl der Anmerkungen basierend auf ihrem Umsetzungsstand. Vollständig adressierte Anmerkungen schließen dabei Anmerkungen mit ein, die durch den Wegfall von Funktionen und ähnliches nicht mehr auf die aktuelle Version der Spezifikation übertragbar sind. Teilweise adressierte Anmerkungen beinhalten Bereiche der Spezifikation, die zwar angepasst wurden, dem eigentlichen Grund der Anmerkung aber nicht vollständig gerecht werden. Die nicht adressierten Anmerkungen vervollständigen die Liste der 60 Anmerkungen.

Tabelle 3 Übersicht des Umsetzungsstands der Handlungsempfehlungen

Vollständig adressiert	*Teilweise adressiert*	*Nicht adressiert*
44	10	6

In Tabelle 4 werden die nicht adressierten Anmerkungen noch einmal aufgegriffen und eine erneute Einschätzung auf Basis der aktuellen Version 1.04 der Spezifikation, sowie die Einschätzung der OPC Foundation von 2017[1], wiedergegeben. Zu einigen der aufgeführten Handlungsempfehlungen hat die OPC Foundation im Rahmen dieser Studie neuen Handlungsbedarf erkannt. Für die grau hinterlegten Empfehlungen wurden neue Einträge im „Mantis“ der OPC Foundation angelegt (Mantis-IDs #6248, #6250, #6251, #6252, sowie #6253).

[1] https://opcfoundation.org/wp-content/uploads/2017/04/OPC_UA_security_analysis-OPC-F-Responses-2017_04_21.pdf

Tabelle 4 Übersicht der nicht adressierten Handlungsempfehlungen

Dokument	*Kapitel*	*Empfehlung für 1.02*	*Anmerkung OPC Foundation (engl.)*	*Feststellung für 1.04*
Part 4	5.5.2.2	„channelId“ sollte durch „secureChannelId“ ersetzt werden.	The XML mapping was removed in V1.03.	Die Anmerkung der Foundation ist nicht nachvollziehbar und scheint ein Copy-and-Paste-Fehler. Das Feld “channelId” ist jedoch Teil des lokal definierten „ChannelSecurity-Token“. Eine explizite Bezeichnung als „secureChannelId“ ist nicht notwendig.
Part 4	5.6.2.2	„Omitted“, „empty“ und „null“ sollten genauer spezifiziert werden.	Replaced ‘omitted’ with ‘null’ in Part 4 v1.04	Korrektur fand nicht statt, Text verwendet weiterhin „empty“.
Part 6	D.3	Datentyp für „ValidationOptions“ sollte auf „byte“ oder „UInt32“ geändert werden, um nur positive Werte zuzulassen.	ValidationOptions is a mask with a finite set of bits used. Negative values are excluded since there are only 6 possible bits.	Empfehlung wurde nicht umgesetzt und Situation kann weiterhin bei Programmierfehlern zu möglichen Schwachstellen führen.
Part 12	7.6.4	Mindestanforderungen an Passwörter sollten ergänzt werden.	Part 2 provides this level of information.	In der Spezifikation werden weiterhin keine Passwortanforderungen definiert, da es sich um einen Kommunikationsstandard handelt kann ggf. darauf verzichtet werden.
Allgemein		Kombination von Security-Mode ‚Sign‘ bzw. ‘SignAndEncrypt’ und SecurityPolicy ‘None” sollte verboten werden.	It should be obvious that “None” implies that neither signing nor encryption are implemented. Certification can check for invalid EndpointDescriptions.	Die Empfehlung von 2016 wurde nicht umgesetzt und ist weiterhin gültig. Die Kombination ist nur in zwei Services durch die Konformitätstests ausgeschlossen.

In Tabelle 5 sind die teilweise adressierten Anmerkungen dargestellt und kommentiert.

Tabelle 5 Übersicht der teilweise adressierten Handlungsempfehlungen

Dokument	*Kapitel*	*Empfehlung für 1.02*	*Anmerkung OPC Foundation (engl.)*	*Feststellung für 1.04*
Part 2	4.2	Einheitliche, anerkannten Standards folgende Begriffs-definitionen sollten verwen-det werden.	We need more specifics on the terms, which are problematic.	Begriffsdefinitionen wurden teilweise er-weitert und ergänzt, weiterhin aber nicht basierend auf definier-ten Standards.
Part 2	4.3.2	„Client flooding“ sollte er-gänzt werden.	The response size is limited by the client with a configured max. The client auto-matically closes connections if they are ex-ceeded. This is also true if extra responses are returned.	Die Beschreibung wurde verändert, Cli-ent flooding findet aber weiterhin keine explizite Erwähnung.
Part 4	5.4.3.1	Korrektur der Erwähnung von „CreateSecureChannel“ statt „OpenSecureChannel“.	Will change text as suggested in Part 2.	In diesem Abschnitt behoben, Fehler tritt aber an neuer Stelle im Dokument auf (Fig. 36).
Part 4	5.5.2.2	Es sollten Unter- und Ober-grenzen für „requested-Lifetime“ empfohlen werden.	Will add a refer-ence to the dis-cussion in Part 2 that explains the factors that af-fect the choice of value.	Sicherheitshinweis hinzugefügt, aber keine Empfehlung für die Grenzwerte.
Part 4	5.6.2.2	Es sollte ein Standardwert o-der Untergrenze für „maxResponsesMessageSize“ empfohlen werden.	Part 4 v1.04 now refers to Part 6 for protocol-specific mini-mum or default values.	Hinweis auf Part 6 be-züglich möglicher Un-tergrenze hinzugefügt, aber bisher kein sol-cher Wert angegeben.
Part 6	5.1.6	Für verschachtelte Daten-strukturen sollten maximale Rekursionstiefen festgelegt werden.	-	Keine Maximalgrenze vorgegeben, aber Emp-fehlung für eine Min-desttiefe von 100 und Fehlermeldungshin-weis.

Dokument	*Kapitel*	*Empfehlung für 1.02*	*Anmerkung OPC Foundation (engl.)*	*Feststellung für 1.04*
Part 6	6.7.2	SecurityPolicyUri, SenderCertificate und ReceiverCertificateThumbprint sollten als ByteStrings definiert werden.	Added text indicating that negative values are invalid in Part 6 v1.04.	Der Handlungsempfehlung zur Umwandlung in ByteStrings wurde zwar nicht entsprochen, hauptsächlich aus Überlegungen der Kompatibilität, durch die Definitionsanpassung wurde das Problem jedoch beseitigt.
Part 7	5.5	Kategorie „documentation – security best practices/recommended settings" könnte hinzugefügt werden.	Part 2 already provides this level of information.	Teilweise wurden Hinweise/Anforderungen zur sicheren Konfiguration im Kontext der Dokumentation aufgenommen. Eine explizite Anforderung, dass zu sicherheitsrelevanten Einstellungen oder Best-Practices fehlt.
Part 7	6.5.124-126	Klarstellung, warum/ob für bestimmte SecurityProfile eine PKI benötigt wird.	-	Betroffene SecurityPolicies teilweise veraltet und nicht mehr beschrieben, Hinweis jedoch bei Basic256Sha256 noch immer vorhanden ohne Begründung.
Part 12	F.2	Klarstellung, wie „Rogue Server" während der Provisionierung erkannt werden.	-	Zwar an weiterer Stelle „Rogue Server" erwähnt, aber keine weiteren Details zur Erkennung gegeben.
Allgemein		Es sollten Standardwerte für verschiedene sicherheitsrelevante Konfigurationsparameter gegeben werden.	-	An wenigen Stellen sind neue Angaben zu Grenzen hinzugekommen, eine gesammelte Liste mit empfohlenen Werten existiert jedoch nicht.

4.2 Sicherheitsrelevante Spezifikationsänderungen

Zur Aktualisierung der OPC UA Studie mussten alle sicherheitsrelevanten Spezifikationsänderungen seit der Version 1.02 identifiziert werden. Im Folgenden wird zunächst das Vorgehen zur Identifikation der sicherheitsrelevanten Spezifikationsänderungen erläutert, bevor die wesentlichen Änderungen vorgestellt werden.

4.2.1 Vorgehen

Am 18.08.2020 wurden alle seit der Version 1.02 vorhandenen OPC UA Spezifikationsteile aus dem Archiv der OPC Foundation heruntergeladen. Dies umfasste insbesondere die OPC UA Parts 2, 3, 4, 5, 6, 7 und 12, welche die wesentlichen sicherheitsrelevanten Spezifikationsänderungen vorzuweisen haben. Jedes dieser Dokumente verfügt nach dem Inhaltsverzeichnis über eine tabellarische Übersicht zu den allgemeinen Spezifikationsänderungen im Vergleich zu den vorangegangenen Versionen. Im ersten Schritt wurden diese Tabellen untersucht und alle identifizierten sicherheitsrelevanten Änderungen in einem Dokument zusammengetragen. Im zweiten Schritt erfolgte ergänzend ein manuelles Review aller OPC UA Parts. Hierbei wurde für jeden OPC UA-Spezifikationsteil die Release-Version 1.02 mit der Release-Version 1.03 verglichen und alle identifizierten sicherheitsrelevanten Änderungen zusammengetragen. Im Anschluss wurde die Release-Version 1.03 mit der aktuellen Release-Version 1.04 verglichen. In Anhang B: Sicherheitsrelevante Spezifikationsänderungen seit Version 1.02 sind die betrachteten Dokumente detailliert aufgeführt und Veränderungen zusammengefasst.

4.2.2 Überblick zu den wesentlichen sicherheitsrelevanten Spezifikationsänderungen

Seit der Version 1.02 wurden zahlreiche neue Konzepte in OPC UA eingeführt und vorhandene Konzepte angepasst. Tabelle 6 fasst die wesentlichen sicherheitsrelevanten Spezifikationsänderungen seit der Version 1.02 zusammenfassen und dient als erster Überblick.

Tabelle 6 Wesentliche sicherheitsrelevante Spezifikationsänderungen

Neue und angepasste Konzepte	***Beschreibung***
Rollenbasierte Zugriffskontrolle	Das Konzept von Rollen und damit assoziierten Berechtigungen wurde neu in die Version 1.04 mit aufgenommen. Somit ermöglicht OPC UA die Umsetzung der rollenbasierten Zugriffskontrolle (RBAC). Entsprechende Spezifikationsanpassungen sind insbesondere in den Spezifikationsteilen 2, 3, 5 und 12 finden.
PubSub	Das PubSub-Kommunikationsmodell ist mit Part 14 neu hinzugekommen. Dies hat insbesondere eine komplette Neuausrichtung der Sicherheitsarchitektur in Part 2 zur Folge. Das Kommunikationsmodell ermöglicht durch die Einführung des *Security Key Servers (SKS)* und den sogenannten *SecurityGroups* eine vertrauliche und integre Kommunikation zwischen Publisher und Subscribern. Außerdem ist für eine Sicherheitsbetrachtung entscheidend, ob das Kommunikationsmodell mit oder ohne Broker eingesetzt wird. Eine genauere Betrachtung der PubSub-Funktionen hat im Rahmen dieser Studie jedoch nicht stattgefunden.
Discovery Services	Mit FindServersOnNetwork und RegisterServer2 sind zwei neue Dienste in Part 4 hinzugekommen, die dem Discovery Service Set angehören. Diese Dienste wurden in Part 12 der Version 1.02 bereits erwähnt, jedoch wurden sie noch nicht in Part 4 definiert. Der Dienst FindServersOnNetwork ist besonders relevant, da dieser ohne Sicherheitsmechanismen aufgerufen werden kann.
Zertifikatsvalidierung	Die in Part 4 beschriebenen Schritte zur Zertifikatsvalidierung wurden zur Version 1.02 erheblich ausgeweitet und können in der Version 1.04 in der Tabelle 106 in Part 4 eingesehen werden. Außerdem wurde Part 6 um einen zusätzlichen Abschnitt zu Zertifikatsketten ergänzt.

Neue und angepasste Konzepte	*Beschreibung*
UserIdentityToken	Die im ActivateSession-Dienst zur Authentifizierung des Benutzers einsetzbaren *IssuedIdentityToken* beschränken sich nicht mehr ausschließlich auf WS-SecurityToken (z.B. Kerberos) sondern ermöglichen nun insbesondere die Nutzung von JSON Web Token (z.B. OAuth2).
Session-less Services	Mit Version 1.04 wurde das Konzept der Session-less Dienstaufrufe eingeführt und in Abschnitt 6.3 in Part 4 beschrieben. Auch Dienstaufrufe über den neuen Session-less Mechanismus erlauben die Authentifikation des Benutzers über ausgestellte *authenticationToken* im *RequestHeader*.
Reverse-Connect	Der neue in Part 6 definierte ReverseConnect-Mechanismus erlaubt es OPC UA-Servern, den Verbindungsaufbau zum Client zu initiieren.
Transportprotokolle	WebSockets wurden in Part 6 als neues Transportprotokoll eingeführt. Außerdem wurde SOAP/HTTP als „*deprecated*“ gekennzeichnet.
Global Discovery Server (GDS)	Der GDS wurde in Part 12 um den *KeyCredential-Manager* und den *Authorization Service* erweitert. Ersterer hat die Verwaltung und Verteilung von Anmeldedaten, die einem *Authorization Service* oder einem Broker übergeben werden, als Ziel. Letzterer stellt Access Token bereit, welche während der Aktivierung einer Session als *IssuedIdentityToken* übergeben werden können. Zusätzlich wurde das Informationsmodell des *CertificateManagers* angepasst und das Konzept der *CertificateGroups* eingeführt.
Profiles und ConformanceUnits	Part 7 wurde umstrukturiert. Es wurde im Rahmen dieser Analyse auf eine detaillierte Auflistung der sicherheitsrelevanten Änderungen verzichtet, weil die als PDF verfügbare Spezifikation das letzte Mal am 22.11.2017 generiert wurde und sich signifikant von der im Reporting-Tool (https://profiles.opcfoundation.org/v104/Reporting/) vorhandenen Version unterscheidet. Grundsätzlich kann aber festgestellt werden, dass zu jedem neuen sicherheitsrelevanten Feature/Konzept auch entsprechende Profiles und ConformanceUnits definiert wurden.
Elliptic-curve cryptography (ECC)	Die OPC Foundation veröffentlichte zum 18.12.2020 ein Amendment zur Unterstützung von ECC. Da diese Änderung nach dem für diese Studie definierten Stichtag veröffentlicht wurde, konnte sie nicht mehr berücksichtigt werden. Da es sich jedoch um eine signifikante Änderung handelt, sollte diese mindestens an dieser Stelle erwähnt werden.

4.3 Neue Handlungsempfehlungen

Aufgrund der zahlreichen sicherheitsrelevanten Änderungen wurde die redaktionelle Analyse auf Basis der Version 1.04 des OPC UA Standards nochmals durchgeführt. Im Rahmen dieser Analyse sind erneut verschiedene Punkte aufgefallen, die zu Verständnis- und/oder Implementierungsproblemen führen können oder Relevanz für die IT-Sicherheit haben.

4.3.1 Vorgehen

Basierend auf den identifizierten sicherheitsrelevanten Spezifikationsänderungen wurde entschieden, dass für die redaktionelle Analyse die OPC UA Spezifikationsteile 2, 3, 4, 5, 6 und 12 betrachtet werden. Im ersten Schritt wurden alle seit Version 1.02 identifizierten sicherheitsrelevanten Änderungen in den entsprechenden PDF-Dokumenten farblich hervorgehoben. Somit konnte der Fokus auf die neu hinzugefügten Textstellen gelegt werden. Im zweiten Schritt wurden diese Dokumente in einem manuellen Review von mehreren Mitarbeitern des Fraunhofer IOSB untersucht. Die hierbei identifizierten redaktionellen Anmerkungen und Handlungsempfehlungen wurden schriftlich festgehalten und mit der Arbeitsgruppe für IT-Sicherheit der OPC Foundation diskutiert.

4.3.2 Überblick zu den identifizierten neuen Handlungsempfehlungen

Tabelle 7 gibt einen Überblick zur Anzahl neuer Handlungsempfehlungen: Insgesamt wurden 192 redaktionelle Anmerkungen und Handlungsempfehlungen identifiziert und mit der Arbeitsgruppe für IT-Sicherheit der OPC Foundation besprochen. Das Ergebnis dieser Diskussionen ist, dass 171 von den insgesamt 192 gemeldeten redaktionellen Anmerkungen und Handlungsempfehlungen bestätigt wurden und dementsprechende Einträge im „Mantis“ der OPC Foundation erstellt wurden. Auf diese Weise ist die weitere Behandlung der Anmerkungen und Handlungsempfehlungen einfacher nachvollziehbar.

Die Ergebnisse jedes Spezifikationsteils können über die E-Mail-Adresse ics-sec@bsi.bund.de angefragt werden. Die Ergebnisse enthalten eine eindeutige ID Verweise auf die Textstellen, Tabellen oder Abbildungen, die gemeldeten redaktionellen Anmerkungen sowie gegebenenfalls entsprechende Handlungsempfehlungen. Wurde eine Anmerkung oder Handlungsempfehlung von der OPC Security Group bestätigt, wurde ein entsprechender Eintrag im „Mantis“ der OPC Foundation angelegt und der entsprechende Link festgehalten.

Tabelle 7 Anzahl der neu gemeldeten und bestätigten redaktionellen Anmerkungen und Handlungsempfehlungen

Analysierter Spezifikationsteil	*Gemeldete Anmerkungen/Empfehlungen*	*Bestätigte Anmerkungen/Empfehlungen*
2	46	35
3	7	5
4	47	44
5	5	5
6	16	14
12	71	68
Gesamt	**192**	**171**

4.3.3 Zusammenfassung der wesentlichen Erkenntnisse

Die wesentlichen Erkenntnisse zu den identifizierten redaktionellen Anmerkungen und Handlungsempfehlungen lassen sich in die folgenden vier Kategorien einordnen.

4.3.3.1 Neue Features und Anforderungen werden nicht in alle relevante Spezifikationsteile eingepflegt

Der OPC UA Standard besteht zum 18.08.2020 aus insgesamt 17 Spezifikationsteilen (Companion Specs, Errata und Amendments nicht einbezogen). Im Rahmen der hier durchgeführten Analyse ist der Eindruck entstanden, dass bei der Einführung neuer Features in den OPC UA Standard nicht alle relevanten Spezifikationsteile angepasst werden, was an der hohen Anzahl an Spezifikationsteilen und unterschiedlichen verantwortlichen Personen liegen kann. Dies trifft auch für die IT-Sicherheit relevanten Features zu.

Beispiele:

- Mit der Version 1.04 des OPC UA Standards wurde das Konzept der Session-less Dienstaufrufe eingeführt und in Part 4 beschrieben. In mehreren Spezifikationsteilen fehlt eine Berücksichtigung dieses neuen Mechanismus und es wird mehrmals aufgeführt, dass eine Authentifizierung des Benutzers ausschließlich mit dem Aufbau einer Session erfolgen kann [Part 2, ID 33] [Part 2, ID 34] [Part 3, ID 2] [Part 4, ID 10] [Part 6, ID 4]. Die Authentifikation des Benutzers ist jedoch auch bei Session-less Dienstaufrufen über ausgestellte authenticationToken im RequestHeader möglich.
- Part 12 führt die Bedingung ein, dass ein Client die Top Level Domain „local“ in der von einem Local Discovery Server mit Multicast-Erweiterung (LDS-ME) bereitgestellten URL bei der Überprüfung des Serverzertifikats zu ignorieren hat. Ein entsprechender Hinweis in den Parts 4 und 6 fehlt jedoch [Part 12, ID 16].

- Das Ausstellen, Widerrufen und Verwalten von Zertifikaten und Vertrauenslisten erfolgt in OPC UA über Zertifikatsgruppen (engl. Certificate Groups) die in Part 12 definiert und eingeführt werden. Leider fehlt die Erwähnung dieses Konzeptes in Part 2 [Part 2, ID 45] und ist auch nicht im XML-Schema der Sicherheitskonfigurationen von OPC UA Anwendungen enthalten [Part 6, ID 16].

4.3.3.2 Unterspezifizierte Konzepte (ggf. aufgrund fehlender Implementierungen)

Für einige in OPC UA definierte Konzepte ist eine Sicherheitsbetrachtung aufgrund unterspezifizierter Textabschnitte nicht möglich. In den Diskussionen mit der Arbeitsgruppe für IT-Sicherheit der Sicherheitsgruppe der OPC Foundation ist aufgefallen, dass dies hauptsächlich auf Konzepte zutrifft die noch nicht praktisch umgesetzt wurden oder kaum Verwendung in der Praxis finden.

Beispiele:

- Der Gateway Server wird in Part 4 nur rudimentär spezifiziert. Da die Information zur Verwendung eines Gateway Servers in der Antwort des GetEndpoints-Dienstes enthalten ist, welcher ohne Sicherheitsmechanismen aufgerufen wird, ist eine entsprechende Sicherheitsbetrachtung durchzuführen, bspw. auf Bedrohungen hinsichtlich möglicher Man-in-the-Middle Angriffe. Leider ist solch eine Betrachtung aufgrund fehlender Informationen in der Spezifikation nicht möglich. [Part4, ID 8] [Part 4, ID 19]
- OPC UA wurde in Part 12 mit dem KeyCredential-Manager und dem Authorization Service, welche beide als zusätzliche Komponenten des Global Discovery Servers betrachtet werden können, erweitert. Der momentane Stand der Spezifikation zu diesen Komponenten lässt vermuten, dass noch keine konkrete Umsetzung implementiert wurde [Part 12, ID 62] [Part 12, ID 63] [Part 12, ID 67] [Part 12, ID 68] [Part 12, ID 70].
- Bei der Aktualisierung eines Applikationszertifikates (engl. ApplicationInstanceCertificate) geht der OPC UA Standard nur darauf ein, wie sich ein OPC UA Server zu verhalten hat. Die Beschreibung für einen OPC UA Client fehlt [Part 4, ID 31].
- OPC UA definiert in Part 4 Schutzmechanismen zur Überprüfung von Informationen, welche von einem DiscoveryEndpoint bereitgestellten werden. Dies ist notwendig, da die entsprechenden Dienstaufrufe (z.B. GetEndpoints) ohne Sicherheitsmechanismen aufgerufen werden. Diese Schutzmechanismen basieren jedoch darauf, dass eine Session aufgebaut wird: So muss bspw. die in der Antwort des CreateSession-Dienstaufrufes enthaltene EndpointDescription identisch zur EndpointDescription sein, welche vom DiscoveryEndpoint bereitgestellt wurde. Wie jedoch im Fall der neuen Session-less Dienstaufrufe verfahren wird, ist nicht beschrieben [Part 2, ID 33] [Part 2, ID 34].

4.3.3.3 Ungenaue Definitionen und sprachliche Ausführungen

An mehreren Stellen sind teils ungenaue Definitionen und sprachliche Ausführungen identifiziert worden, welche zu Verständnis- und/oder Implementierungsproblemen führen können.

Beispiele:

- In Abschnitt 6.1.2 von Part 4 wird fälschlicherweise definiert, dass der öffentliche Schlüssel von der CA ausgegeben wird [Part 4, ID 20].
- Die in Part 3 definierte „SecurityAdmin“-Rolle soll für sicherheitsrelevante Einstellungen zum Einsatz kommen. Was aber sicherheitsrelevante Einstellungen sind wird nicht definiert [Part 3, ID 5].
- In Part12 wird beim Updaten eines Applikationszertifikates über den CertificateManager des GDS hinsichtlich des Validierungsprozesses (teilweise) auf Part 4 verwiesen. Die Referenz sollte konkretisiert werden, um das Auffinden in Part 4 zu vereinfachen. Es ist nicht spezifiziert, wie an dieser Stelle mit den in Part 4 beschriebenen Fehlern, die unterdrückt werden können, umzugehen ist und ob dieselben Einstellung wie in Part 4 zu verwenden sind oder anderer Parameter als bei den Verbindungen zum Einsatz kommen können. [Part 12, ID 39] [Part 12, ID 40] [Part 12, ID 41].

4.3.3.4 Komplexität des OPC UA Standards

Bei der Durchsicht des OPC UA Standards ist aufgefallen, dass sicherheitsrelevante Konzepte in unterschiedlichen Spezifikationsteilen definiert sind und somit zu Verständnis- und/oder Implementierungsproblemen führen können. So könnte man vom OPC UA Part 2 mit dem Titel „Security Model“ annehmen, dass dieser Spezifikationsteil einen Überblick über alle sicherheitsrelevanten Konzepte in OPC UA bietet. Leider ist dies nicht der Fall. Möchte man einen Überblick aller sicherheitsrelevanter Konzepte in OPC UA erhalten, sollten mindestens die Parts 2, 3, 4, 5, 6, 7, 12 und 14 gelesen werden. Da in Zukunft neue Spezifikationsteile mit neuen sicherheitsrelevanten Konzepten spezifiziert werden, ist eine bessere Übersicht/Auflistung wünschenswert.

Beispiel:

- Part 2 stellt mit Abbildung 1 ein einziges allgemeines OPC UA Netzwerkbeispiel vor. Dieses Beispiel enthält weder einen CertificateManager, KeyCredential Service, Authorization Service oder Security Key Server (SKS). Mindestens ein Netzwerkbeispiel mit allen im OPC UA Standard definierten sicherheitsrelevanten Komponenten ist zu empfehlen und in Part 2 mit dem Titel „Security Model“ zu erwarten [Part 2, ID 11].

Zur Reduzierung der Komplexität des OPC UA Standards und dem Vermeiden von Verständnis- und/oder Implementierungsproblemen können auch Referenzimplementierung hilfreich sein. Nach Anfrage bei der OPC Foundation ist jedoch momentan keine zertifizierte Referenzimplementierung verfügbar.

5 Analyse der Schutzmechanismen auf Parameterebene

In der Studie von 2016 wurde eine Analyse der Schutzmechanismen auf Parameterebene durchgeführt. Dieser Abschnitt stellt eine Aktualisierung dieser Analyse dar. Zunächst wird das genaue Vorgehen beschrieben, wobei auch das Änderungsprotokoll zum Dokument "Detaillierte Analyse der Schutzmechanismen der OPC UA Spezifikation auf Parameterebene.xls“ aus der Studie 2016 bereitgestellt wird. Anschließend wird die in diesem Dokument enthaltene Analysetabelle erläutert und die daraus resultierenden Ergebnisse vorgestellt.

5.1 Vorgehen und Änderungsprotokoll zur Analysetabelle

Die in der Studie 2016 durchgeführte Analyse der Schutzmechanismen auf Parameterebene bezieht sich ausschließlich auf das Client-Server-Kommunikationsparadigma unter Berücksichtigung der drei Service Sets: *Discovery*, *SecureChannel* und *Session*. Dieser Rahmen wurde auch für die hier durchgeführte Aktualisierung der Analyse auf Parameterebene beibehalten. Auf Basis der in Abschnitt 4.2 identifizierten sicherheitsrelevanten Änderungen konnte festgestellt werden, dass seit der Studie von 2016 die zwei Discovery Dienste *FindServersOnNetwork* und *RegisterServer2* neu hinzugekommen sind. Außerdem wurde identifiziert, dass es einen neuen *ReverseConnect*-Mechanismus gibt, welcher OPC UA Servern das Initiieren des Verbindungsaufbaus ermöglicht. Im ersten Schritt wurde das Dokument "Detaillierte Analyse der Schutzmechanismen der OPC UA Spezifikation auf Parameterebene.xls“ um diese neuen Dienste bzw. Mechanismen ergänzt. Anschließend wurde das Dokument hinsichtlich veralteter Begriffe, Referenzen und Standardwerte überprüft und aktualisiert.

Neben dem Dokument "Detaillierte Analyse der Schutzmechanismen der OPC UA Spezifikation auf Parameterebene“ mussten auch die Textabschnitte aus dem Abschlussbericht von 2016 aktualisiert werden. Da mit den neuen Diensten, Mechanismen und Begrifflichkeiten nur kleine Anpassungen vorzunehmen waren, wurde entschieden, dass die Textabschnitte aus dem Abschlussbericht von 2016 in dieses Dokument übernommen werden und geänderte bzw. neu hinzugefügte Textpassagen mit roter Schriftfarbe hervorgehoben werden.

Tabelle 8 Änderungsprotokoll zum Dokument "Detaillierte Analyse der Schutzmechanismen der OPC UA Spezifikation auf Parameterebene.xls“

ID	*Nachrichtentyp*	*Beschreibung*
1	ReverseHello	Die *ReverseHello*-Nachricht (RHE) ist neu hinzugekommen und wurde der Tabelle hinzugefügt.
2	FindServersOnNetwork, RegisterServer2	Diese beiden Dienste sind im Discovery Service Set neu hinzugekommen. Sie wurden in der Tabelle hinzugefügt.
3	Hello, Acknowledge	Die Mindestgröße eines *MessageChunk* muss nun nicht größer, sondern größer-gleich 8 192 Bytes betragen.
4	OpenSecureChannel	Die für diesen Dienst angegebenen Ziffern, welche Angaben zum Schutz gegen bestimmte Bedrohungen entsprechen, ist in der Studie von 2016 fehlerhaft. Das Dokument von 2016 fasst für diesen Dienst sowohl die Definition aus Part 4 als auch aus Part 6 auf. Für beide Definition wurden Ziffern vergeben und akkumuliert. Somit wurden Ziffern fälschlicherweise doppelt einberechnet. Dieser Fehler wurde korrigiert.
5	OpenSecureChannel, CloseSecureChannel	Bei diesen beiden Diensten muss die Nachricht nun aus genau einem einzigen/finalen Chunk bestehen.
6	OpenSecureChannel	Der ReceiverCertificateThumbprintLength-Parameter muss im SecurityMode None nicht mehr 20 Bytes betragen und kann auch die Werte 0 oder -1 annehmen.

ID	*Nachrichtentyp*	*Beschreibung*
7	OpenSecureChannel	SecureChannelId wurde auf den Typ BaseDataType verallgemeinert.
8	CreateSession	Die Empfehlung für den ServerEndpoints []-Parameter vom Typ Endpoint-Description wurden angepasst.
9	CreateSession	Im Request wird nicht mehr explizit untersagt (*securityPolicyUri* von *None)* ein Zertifikat mitzusenden. Dieses soll jedoch weiter vom Server ignoriert werden.
10	ActivateSession	clientSoftwareCertificates sind nun als „Reserved for future“ markiert.
11	ActivateSession	Neue Default-Wert: Null oder fehlendes UserdientityToken soll nun immer als anonymer Benutzer gehandhabt werden.
12	ActivateSession	Beschreibung des userTokenSignature-Parameters angepasst: omitted -> null
13	Alle Dienste - Footer	Die Reihenfolge der Felder im OPC UA Secure Conversation Message footer wurde verändert: PaddingSize, Padding, ExtraPaddingSize, Signature. Die Tabelle wurde entsprechend angepasst.
14	-	Es wurde das abstrakte OPC UA Connection Protokoll eingeführt. Da vor allem der vom OPC UA TCP umgesetzte Handshake auch mit anderen Transportprotokollen eingesetzt werden kann (z.B. WebSockets), wurde das abstrakte OPC UA Connection Protokoll (UACP) eingeführt. Die Tabelle wurde entsprechend angepasst.
15	-	Die in dem Dokument nicht mehr aktuellen SecurityPolicyUris wurden entfernt.
16	-	Die in dem Dokument nicht mehr aktuellen Tabellen- und Seitennummern wurden entfernt.

5.2 Erläuterung der Analysetabelle

Basierend auf den inzwischen in Part 2 eingeführten Bedrohungstypen scheint OPC UA Mechanismen zum Schutz aller aufgelisteten Bedrohungstypen zu bieten. Dies wurde anhand einer detaillierten Analyse bis auf Parameterebene geprüft.

Die detaillierten Ergebnisse dieser Analyse der Sicherheit von OPC UA auf der Grundlage der in den ausgetauschten Nachrichten enthaltenen Parameter können über die E-Mail-Adresse ics-sec@bsi.bund.de angefragt werden, da diese aufgrund des Umfangs und fehlender Barrierefreiheit nicht veröffentlicht werden.

Das Dokument ist wie folgt aufgebaut:

Welchen Schutz OPC UA bietet, hängt entscheidend vom Parameter securityMode ab. Deshalb wurde die Darstellung der Ergebnisse der Übersichtlichkeit halber in drei Tabellen aufgeteilt: Über die Reiter kann man sich die Ergebnisse für securityMode 'None', 'Sign' und 'SignAndEncrypt' anzeigen lassen. Der vierte Reiter fasst die Ergebnisse aller Modi zusammen und ist weiter unten erklärt.

Die drei ersten Tabellen sind gleich aufgebaut: Die Zeilen stellen in Form einer links auf- und zuklappbaren Baumstruktur dar, wie die Nachrichten bei UACP (OPC UA Connection Protocol) und UASC (UA Secure Conversation) aufgebaut sind. Es wurden die Nachrichten ausgewählt, die für die Kommunikation entscheidend sind und somit eine wichtige Rolle bei der IT Sicherheit spielen, nämlich die Findung der Kommunikationspartner (Discovery Dienst) und der Verbindungsaufbau (SecureChannel und Session): Bei UACP handelt es sich um vier mögliche Nachrichtentypen (HEL, ACK, ERR und REV). Bei UASC wurden die Nachrichten für den SecureChannel, die Session und den Discovery Dienst berücksichtigt.

Die Spalten stellen zwei Arten von Informationen dar: Die linken Spalten (A bis L) dienen der Beschreibung der Nachrichtenparameter, die rechten (M bis X) zeigen die Ergebnisse bezüglich der Auswertung der Sicherheit an.

Hier eine detailliertere Aufschlüsselung der Spalteninhalte:

- Die Spalten A bis F stellen die verschiedenen Ebenen der Protokolle, Nachrichtenteile und Parameter dar.
- In der Spalte G sind die Datentypen der Parameter spezifiziert, Spalte H zeigt, falls vorhanden, passende Defaultwerte an.
- Die Spalten I und J heben farblich hervor, welche Teile der jeweiligen Nachrichten signiert, respektive verschlüsselt sind:

Spalte I:

- grau = unsigniert
- blau = signiert

Spalte J:

- grau = unverschlüsselt
- grün = verschlüsselt

Außerdem wird innerhalb der Flächen angezeigt, mit welchem Schlüssel die Operation durchgeführt wurde.

- Die Spalten K und L enthalten Erläuterungen zum besseren Verständnis der Parameter.
- In den Spalten M bis W werden die Bedrohungen aus Part 2 aufgelistet. Die Bedrohungen wurden auch bereits in der Studie von 2016 eingeführt.

Die Ergebnisse der Analyse sind folgendermaßen in der Tabelle eingetragen: Unterstützt ein Parameter beim Schutz gegen eine bestimmte Bedrohung, wird in der jeweiligen Zeile (Parameter) und Spalte (Bedrohung) eine '1' eingetragen. Beispiel: Die SecureChannelId trägt zum Schutz gegen die Wiederverwendung von Nachrichten bei. Deshalb ist in der Zelle Q53 eine '1' eingetragen.

Zusammenfassend werden außerdem die Einträge für die einzelnen Bedrohungen auf Nachrichtenebene aufsummiert. Kommt dabei eine Zahl größer als 0 heraus, bedeutet dies, dass dieser Nachrichtentyp einen Schutz gegen diese Bedrohung bietet. Diese Aufsummierung der Einträge je Bedrohung wurde auch auf Dienst- und Protokollebene fortgesetzt. Die Ergebnisse auf Dienstebene wurden in der Tabelle im letzten Reiter übertragen.

Die Interpretation, ob ein Parameter den Schutz gegen eine Bedrohung erhöht, ist relativ weit gefasst: Es bedeutet insbesondere nicht, dass ein Parameter alleine ausreichend ist, um gegen die jeweilige Bedrohung zu schützen, sondern lediglich, dass er dazu beiträgt. Und die Tatsache, dass mehrere Parameter zum Schutz gegen eine Bedrohung beitragen, bedeutet auch nicht automatisch, dass die Bedrohung damit vollständig abgewehrt ist.

5.3 Detaillierte Erläuterung der Ergebnisse der Analysetabelle

Mit den im vorherigen Abschnitt durchzuführenden Änderungen ergibt sich folgende aktualisierte Tabelle.

Tabelle 9 Wirksamkeit der OPC UA Schutzmaßnahmen

Security-Mode	*Layer or Service*	*Denial of Service*	*Eaves-dropping*	*Message Spoofing*	*Message Alteration*	*Message Replay*	*Mal-formed Messages*	*Server Profiling*	*Session Hijacking*	*Rogue Server*	*Compromis-ing User creden-tials*	*Repudiation*
		Geringer Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Geringer Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Kein Schutz
	UACP	8	0	0	0	0	8	0	0	0	0	0
None		Ein-ge-schränkter Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Geringer Schutz	Geringer Schutz	wirksamer Schutz	Geringer Schutz	wirksamer Schutz	Ein-ge-schränkter Schutz
	Secure-Channel	10	0	0	0	16	1	0	15	0	0	0
	Session	14	0	2	0	26	3	4	23	0	2	2
	Dis-covery	20	0	4	4	35	9	8	30	6	0	6
Sign		Ein-ge-schränkter Schutz	Kein Schutz	wirksa-mer Schutz	wirksa-mer Schutz	wirksa-mer Schutz	wirksa-mer Schutz	Ein-ge-schränkter Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz
	Secure-Channel	10	8	10	10	21	11	15	26	7	10	12
	Session	14	0	12	8	31	12	14	28	6	4	18
	Dis-covery	21	0	5	5	36	9	20	31	7	1	10
Sign-And-En-crypt		Ein-ge-schränkter Schutz	wirksa-mer Schutz	wirksa-mer Schutz	wirksa-mer Schutz	wirksa-mer Schutz	wirksa-mer Schutz	Ein-ge-schränkter Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz
	Secure-Channel	10	14	10	10	21	11	15	29	7	14	12
	Session	14	18	12	8	31	12	14	46	6	22	18
	Dis-covery	21	13	5	5	36	9	20	43	7	13	10

Eingeschränkter Schutz bedeutet, dass die Möglichkeiten eines Angreifers einschränkt sind, diese Art von Angriffen aber nicht verhindert

Es ist wichtig zu betonen, dass die Zahlen lediglich als Hilfe für die Interpretation herangezogen wurden: Eindeutig ist nur, falls bei einem securityMode für eine Bedrohung bei den drei Diensten eine 0 steht, dann kann kein Schutz vorhanden sein. Aber eine quantitative Auswertung ist nicht möglich: Höhere Werte deuten nicht zwangsläufig auf einen besseren Schutz hin.

Die in der Studie aus dem Jahr 2016 gewonnenen Erkenntnisse zum Client/Server-Kommunikationsparadigma mit aufgebauter Session sind mit der Version 1.04 des OPC UA Standards weiterhin aktuell. Lediglich das in Version 1.04 eingeführte OPC UA Connection Protocol führt dazu, dass der Text aus der Studie von 2016 angepasst werden musste. Die detaillierte Erläuterung zu den Ergebnissen wird im Folgenden erneut aufgeführt:

5.3.1 OPC UA Connection Protocol

Da der Parameter securityMode Teil von UASC ist, hat er keine Auswirkung auf die darunterliegende Schicht OPC UA Connection Protocol (UACP).

denial of service: UACP bietet nur minimalen Schutz gegen flooding Angriffe.

malformed message: Die Festlegung von Obergrenzen für Puffer- und Nachrichtengrößen und der maximalen Anzahl an Teilnachrichten bietet auch sehr eingeschränkt Schutz gegen fehlerhafte Nachrichten, weil z. B. eine Nachricht mit unzulässiger Größe früh verworfen wird.

server profiling: Weil die Anzahl der weltweit verfügbaren unterschiedlichen OPC UA SDKs übersichtlich ist, ist nicht ausgeschlossen, dass allein die auf UACP Ebene gewonnenen Informationen zur Erstellung eines eindeutigen Serverprofils ausreichen könnten.

5.3.2 SecurityMode None

denial of service: Die Schutzvorkehrungen gegen flooding Angriffe sind für die Parameter die gleichen, wie bei den zwei anderen Modi auch, allerdings mit unterschiedlichen Auswirkungen: Einerseits ist es für einen Angreifer schwieriger, das Opfer zu aufwendigen Rechenoperationen zu zwingen, da weder signiert, noch verschlüsselt wird. Andererseits hat ein Angreifer durch das Wegfallen der Applikationsauthentifizierung mehr Möglichkeiten, den Arbeitsspeicher zu überlasten und die Festplatte vollzuschreiben.

Eavesdropping: Weil, mit der Ausnahme von geheimnisbasierten userIdentityTokens, bei securityMode None keine Verschlüsselung stattfindet, ist das Schutzziel Vertraulichkeit nicht erfüllt.

message spoofing, message alteration: Gegen das Erzeugen und die Änderung von Nachrichten, wie z. B. bei einem Man-in-the-Middle Angriff, schützt securityMode None wegen der fehlenden Applikationsauthentifizierung überhaupt nicht.

message replay und session hijacking: Gegen das erneute Versenden von Nachrichten und die Übernahme von Sessions greifen geeignete Mechanismen, wie z. B. die Prüfung der sequenceNumber. Allerdings setzt dies zwingend voraus, dass der Angreifer nicht zusätzlich mitlauscht.

malformed message: Durch die nicht vorhandene Applikationsauthentifizierung hat ein Angreifer viele Möglichkeiten, fehlerhafte Nachrichten zu erzeugen und vom Opfer auswerten zu lassen. Allein die Benutzerauthentifizierung, falls zwingend erforderlich, verhindert den Aufbau einer Session.

server profiling: Der Schutz gegen die Erstellung eines Serverprofils beschränkt sich auf die Tatsache, dass die Benutzerauthentifizierung, falls zwingend erforderlich, den Aufbau einer Session verhindert.

rogue server: Die vorgeschriebene Applikationsauthentifizierung beim RegisterServer und RegisterServer2 Discovery Dienst verhindert, dass ein Angreifer seinen nicht autorisierten Server bei Discovery Servern anmeldet und somit dieser über FindServers oder FindServersOnNetwork von Clients gefunden wird. Allerdings wird in Part 12 auch die Möglichkeit geboten, dass sich ein rogue server über mDNS bekannt macht.

Durch die fehlende Applikationsauthentifizierung hat ein Client anschließend keine Möglichkeit mehr, einen nicht vertrauenswürdigen Server von einem vertrauenswürdigen zu unterscheiden.

compromising user credentials: Ist, wie von der Spezifikation gefordert, entweder die securityPolicyUri oder eine userTokenPolicy gesetzt, also nicht 'None', so ist die Sicherheit eines geheimnisbasierten userIdentity-Tokens bei der Benutzerauthentifizierung gewährleistet.

repudiation: Dadurch ist die Nichtabstreitbarkeit bezüglich der Benutzerauthentifizierung auch gegeben. Die Nichtabstreitbarkeit bezüglich der Applikation kann nicht eingehalten werden, da keine solche Authentifizierung stattfindet.

5.3.3 SecurityMode Sign und SignAndEncrypt:

eavesdropping: Wie erwartet, unterscheiden sich die zwei Modi im Wesentlichen in dem Schutz der Vertraulichkeit: Im securityMode Sign werden zwar die OpenSecureChannel Anfragen und Antworten verschlüsselt, der weitere Nachrichtenaustausch findet aber unverschlüsselt statt. Insbesondere die Inhalte aller Anfragen und Antworten, die im Rahmen einer bestehenden Session ausgetauscht werden, können von einem mitlauschenden Angreifer gelesen werden, weil sie im Klartext versendet werden. Nur geheimnisbasierte userIdentityTokens, die bei activateSession Anfragen zur Authentifizierung des Benutzers gesendet werden, sind verschlüsselt und somit geschützt.

Dagegen werden bei SignAndEncrypt alle Nachrichten beim SecureChannel und der Session verschlüsselt. Allein beim Discovery Dienst müssen alle Applikationen, unabhängig vom sonst eingestellten securityMode, FindServers, FindServersOnNetwork und GetEndpoints Anfragen und Antworten auch ohne Sicherheit unterstützen, also unverschlüsselt und unsigniert.

denial of service: Bestimmte Parameter schränken die Möglichkeiten eines Angreifers bei flooding Angriffen ein. Insbesondere die erzwungene Applikationsauthentifizierung erschwert ganz erheblich einen flooding Angriff auf Sessionebene bzw. macht diesen ohne Kenntnis weiterer Informationen wie privaten Schlüsseln unmöglich. Nichtsdestotrotz hat der Angreifer gerade wegen der Schutzmaßnahmen, die aufwendige Entschlüsselungs- und Signaturverifizierungsschritte mit sich bringen, die Möglichkeit, zusätzlich zu einer hohen Netzwerklast auch den Prozessor deutlich stärker zu beanspruchen, als es mit securityMode None der Fall ist. Also bietet UASC nur einen unzureichenden Schutz gegen flooding Angriffe an.

Diese Schwachstelle muss jedoch relativiert werden, weil ein Angreifer mit deutlich geringerem Aufwand durch flooding auf IP oder TCP Protokollebene vermutlich vergleichbare Ergebnisse erzielen würde.

server profiling: Durch die erforderliche Applikationsauthentifizierung hat ein Angreifer weniger Möglichkeiten über den SecureChannel Informationen zu gewinnen, wobei auch eine fehlgeschlagene Authentifizierung eventuell Informationen über den Server preisgibt. Sessions kann ein Angreifer bei den securityModes nicht, aber FindServers, FindServersOnNetwork und GetEndpoints sind ohne Authentifizierung möglich.

Gegen alle anderen Bedrohungen schützen beide Modi ausreichend, mit minimal besserem Schutz bei SignAndEncrypt. Spielt die Vertraulichkeit keine Rolle, reicht wahrscheinlich securityMode Sign.

5.4 Zusammenfassung & Schlussfolgerung

Die in der Studie aus dem Jahr 2016 gewonnenen Schlussfolgerungen zum Client/Server-Kommunikationsparadigma mit aufgebauter Session sind mit der Version 1.04 des OPC UA Standards weiterhin aktuell und werden im Folgenden erneut aufgeführt:

securityMode None bietet wenig bis keinen Schutz vor IT Sicherheitsangriffen und sollte unbedingt vermieden werden. Spielt die Vertraulichkeit keine Rolle, bietet securityMode Sign einen angemessenen Schutz gegen die betrachteten Bedrohungen. Werden vertrauliche Daten ausgetauscht, ist securityMode SignAndEncrypt erforderlich.

Die Bedrohungen „*denial of service*“ und „*server profiling*“ können, wie bei jedem TCP/IP-Protokoll, mit OPC UA Schutzmechanismen nur abgemildert, aber nicht komplett abgewehrt werden. Entsprechend müssen zusätzliche Maßnahmen außerhalb von der OPC UA Infrastruktur zum Schutz gegen diese Bedrohungen beitragen.

Zudem zeigt die Analyse, dass mit securityMode SignAndEncrypt allen Bedrohungen bis auf „*denial of service“* und „*server profiling*“ ausreichend begegnet werden kann. Somit werden auch die Schutzziele bis auf die Verfügbarkeit erreicht.

6 Marktbefragung

Im Zuge dieser Studie wurde auch eine Marktbefragung durchgeführt. Ziel war es dabei, einen Einblick in den praktischen Einsatz von OPC UA zu gewinnen und dabei besonders auf die sicherheitsrelevanten Eigenschaften zu achten.

6.1 Vorgehen

Ziel der Marktbefragung war es herauszufinden, welche Sicherheitsmechanismen in Produkten, die OPC UA unterstützen, umgesetzt wurden und welche Schwierigkeiten die Hersteller dabei vorfanden. Zur Ermittlung der Daten wurde eine Umfrage entworfen, die eine möglichst breite Zielgruppe ansprechen sollte, um ein möglichst breites Abbild über den praktischen Einsatz von OPC UA zu erhalten. Die Umfrage sollte dabei so aufgebaut sein, dass auch Personen mit unterschiedlich tiefem Wissen über OPC UA zu den Ergebnissen beitragen konnten. Dabei sollten die Teilnehmer die Fragen jeweils auf ein bestimmtes OPC UA Produkt beziehen.

Die Einladung zur Teilnahme an der Umfrage wurde über zwei Arten verbreitet:

- Hersteller, die zertifizierte OPC UA Produkte anbieten, wurden direkt zur Teilnahme an der Umfrage eingeladen. Hierbei wurden die zertifizierten Produkte der Webseite der OPC Foundation entnommen.
- Alle anderen Hersteller wurden u.a. durch eine Einladung im Newsletter der OPC Foundation eingeladen, an der Umfrage teilzunehmen.

Die Umfrage war von September 2020 bis März 2021 für Antworten geöffnet.

6.2 Aufbau und Durchführung

Zur Durchführung der Umfrage wurde zunächst ein Fragenkatalog mit über 100 Fragen aufgebaut. Die Fragen basierten dabei auf allgemeinen Best Practices aus der IT-Sicherheitsbereich, aber auch z.B. auf den Conformance Units der OPC UA Spezifikation.

Aus diesen Fragen wurde eine Umfrage mithilfe der Online-Plattform SurveyMonkey implementiert und durchgeführt. Dabei wurden die Fragen auf 16 Seiten aufgeteilt. Einzelne Fragen und Seiten konnten dabei regelbasiert ein- und ausgeblendet werden. Hierdurch war eine individuelle Anpassung an die Teilnehmer und die referenzierten Produkte möglich.

Insgesamt wurden die Fragen in den folgenden 8 verschiedenen Kategorien eingeteilt:

Tabelle 10 Umfragekategorien der Marktanalyse

Kategorie	*Beschreibung*
Basic Questions	Grundlegende Fragen zum OPC UA-Produkt, wie dessen Art oder vorhandene Zertifizierungen, und freiwillige Angaben zum Teilnehmer.
Product Features	Funktionen des Produkts, wie z.B. zu Grunde liegendes SDK oder unterstützte OPC UA-Versionen.
Basic Security Questions	Grundlegende Fragen zu den Sicherheitseigenschaften, wie z.B. sichere Standardkonfiguration oder unterstützte Authentifizierungsmethoden.
Expert Security Questions	Weitergehende Fragen zu den Sicherheitseigenschaften des Produkts, wie z.B. unterstützte Sicherheitskonzepte oder verfügbare Zertifikatschecks.
Product Development	Fragen zur sicheren Entwicklung des Produkts, wie z.B. Produktanforderungen oder eingesetzte Methoden zur sicheren Softwareentwicklung.
Discovery Server	Fragen zum Einsatz von Discovery-Diensten.
Global Discovery Server	Fragen zur Unterstützung des GDS und seiner Funktionen, wie z.B. dem Zertifikatsmanagement.

Kategorie	*Beschreibung*
PubSub	Fragen zur Unterstützung von PubSub in OPC UA, wie z.B. unterstützte Sicherheitslevels.

6.3 Auswertung der Ergebnisse

Insgesamt wurden 138 (Teil)Beantwortungen gezählt. 129 Teilnehmer nutzten dabei einen öffentlichen Link. Neun Teilnehmer nutzten individuelle Links, die an alle Produkthersteller von offiziell zertifizierten Produkten verteilt wurden.

Im Folgenden werden die wesentlichen Erkenntnisse aus den einzelnen Fragekategorien kurz zusammengefasst. Aufgrund des Kenntnisstands der Teilnehmer und der dadurch angepassten individuell angezeigten Fragen bzw. durch das Überspringen von Fragen ergaben sich die folgenden Werte für die durchschnittlichen Beantwortungszahlen der einzelnen Kategorien:

Tabelle 11 Anzahl der gegebenen Antworten pro Fragekategorie

Basic Questions	*Product Features*	*Basic Security Questions*	*Expert Security Questions*	*Product Development*	*Discovery Server*	*Global Discovery Server*	*PubSub*
131	79	73	21	26	42	13	9

6.3.1 Basis Fragen

Insgesamt gaben durchschnittlich 131 Teilnehmer eine Antwort in dieser Kategorie ab.

Die grundlegenden Fragen gaben einen guten Einblick über die Art der Teilnehmer und ihrer Produkte. Die behandelten Produkte waren zum Großteil OPC UA-Server (77,54%), aber auch Client-Produkte (67,39%), wie in Abbildung 1 dargestellt. Immerhin 16.67% gaben an, dass ihr Produkt ein eigenes SDK darstellt (insgesamt 23 Teilnehmer). Mehrfachnennungen waren hier möglich.

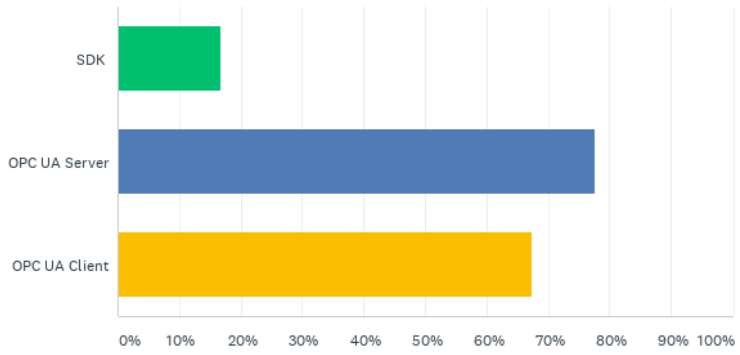

Abbildung 1 Art des OPC UA-Produkts

Von den Teilnehmern waren knapp 55% eigenen Angaben zufolge Entwickler von OPC UA-Produkten, 30% Produktmanager und 5% aus dem Produktsupport. Über 60% der Teilnehmer gaben an, dass ihr Produkt nicht offiziell zertifiziert sei. Die zertifizierten Produkte deckten dabei alle verfügbaren und teilweise auch veralteten Zertifizierungsprofile ab. Die Mehrzahl der zertifizierten Produkte wurde nach den Profilen Standard 2017 UA Server Profile (16,38%) bzw. Standard 2017 Client Profile (12,93 %) zertifiziert.

6.3.2 Produkt Funktionen

Insgesamt gaben durchschnittlich 79 Teilnehmer eine Antwort in dieser Kategorie ab.

Es zeigt sich eine breite Streuung bei der Nutzung der eingesetzten Stacks, dargestellt in Abbildung 2.

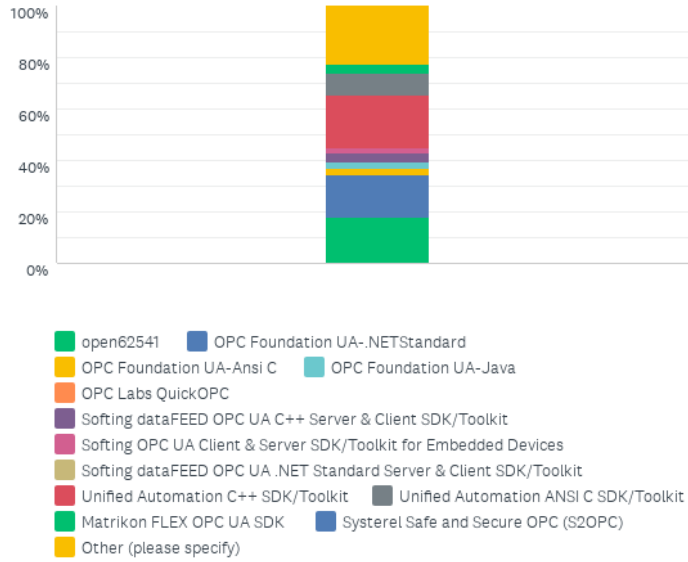

Abbildung 2 Eingesetzte OPC UA-Stacks

Bei den unterstützen OPC UA-Versionen, dargestellt in Abbildung 3, lag zwar die aktuelle Version 1.04 mit 72,84% auf dem ersten Platz, aber auch Version 1.00 wurde von 25% der Teilnehmer als noch immer unterstützt angegeben. Hierbei plant aber die Mehrheit in Zukunft auch die aktuelle Version zu unterstützen. Rund 30 % plant jedoch kein Update.

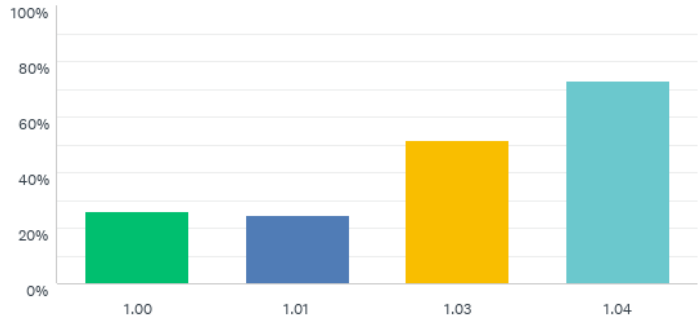

Abbildung 3 Eingesetzte OPC UA-Versionen

Knapp über ein Drittel der Teilnehmer (33,8%) gaben an, dass ihr Produkt Aspekte des Global Discovery Servers unterstützen würde. Eine weitere Frage bezog sich auf die Gewichtung der Produktanforderungen Sicherheit, Performanz, Benutzbarkeit und Funktionsumfang. Hierbei wurde der Funktionsumfang am häufigsten (41,18%) als wichtigstes Ziel gesehen. Insgesamt wird aber auch die Sicherheit als wichtig eingestuft und als zweitwichtigstes Ziel gewertet.

Bei der Produktentwicklung gaben 80 % an, dass zumindest interne Sicherheits-Experten mit einbezogen werden. Nur ein Teilnehmer gab hier an, dass auch externe Experten konsultiert werden.

6.3.3 Basis Security Fragen

Insgesamt gaben durchschnittlich 73 Teilnehmer eine Antwort in dieser Kategorie ab.

Über 70% der Teilnehmer gaben an, dass für ihr Produkt Anweisungen zur sicheren Konfiguration vorhanden sind. Gleichzeitig sind in der Standardkonfiguration laut 50% der Teilnehmer die Sicherheitsfunktionen nicht aktiviert bzw. gibt es keine Standardvorgaben, wie in Abbildung 4 dargestellt.

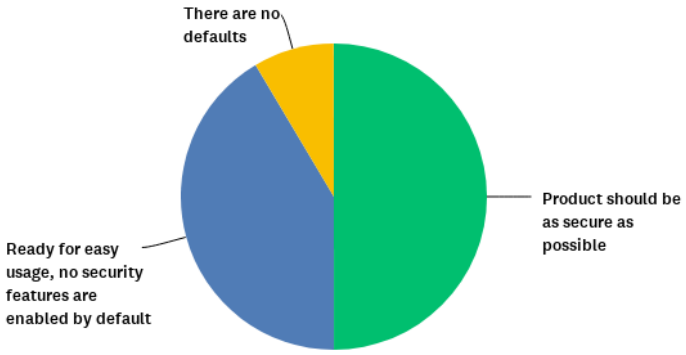

Abbildung 4 Standardkonfigurationen und Sicherheitseinstellungen

6.3.4 Experten Security Fragen

Insgesamt gaben durchschnittlich 21 Teilnehmer eine Antwort in dieser Kategorie ab.

Das am meisten verbreitete Transport-Protokoll für OPC UA ist „OPC UA TCP“, das von allen Produkten unterstützt wird. OPC UA über https (15%) bzw. WSS (2%) spielen eine sehr untergeordnete Rolle.

Part 7 der Spezifikation definiert einige sicherheitsrelevante Vorgehensempfehlungen, wie z.B. den Umgang mit Timeouts oder Zufallszahlen. Die Beachtung dieser Vorgaben ist jedoch recht unterschiedlich (14% - 63%) und keine Vorgabe wurde von jedem Produkt beachtet, wie dargestellt in Abbildung 5.

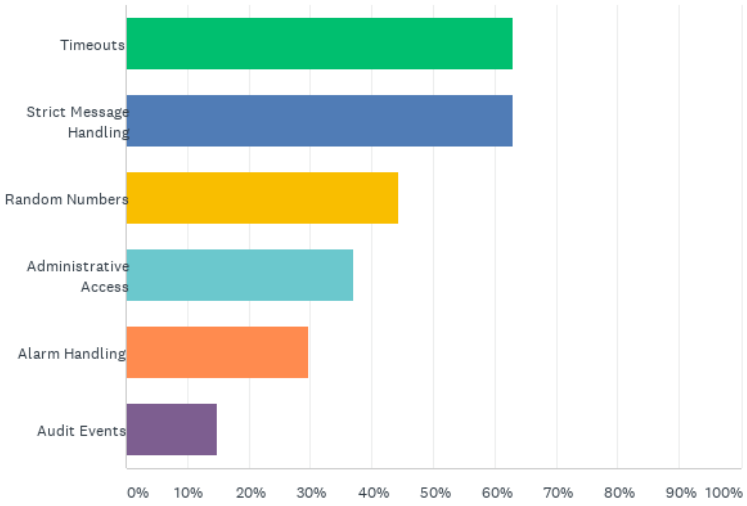

Abbildung 5 Umgesetzte Vorgehensempfehlungen

Durch die Spezifikation vorgegebene Standardrollen werden nur von 12% der Produkte unterstützt, u.a. da nur 25% der Produkte überhaupt das Rollenmodell von OPC UA bereits unterstützen.

Auch bei den unterstützen Zertifikatsüberprüfungen werden scheinbar nicht alle durch die Spezifikation vorgegeben Prüfungen zwingend durchgeführt oder die Ergebnisse unterdrückt. Wie in Abbildung 6 zu sehen, schwanken die Unterstützungsraten bei den einzelnen Tests zwischen 36% (Überprüfung der URI) und 86% (Überprüfung des Gültigkeitszeitraums). Die Zustimmung zur Prüfung der Zertifikatsstruktur oder der Signatur von unter 70 % bzw. 80 % kann auch auf mögliche Missverständnisse hinsichtlich der Fragestellung oder Unkenntnis der Produktfunktionen hinweisen.

Die meisten Produkte (96%) scheinen keine zertifikatsspezifischen Alarme zu unterstützen.

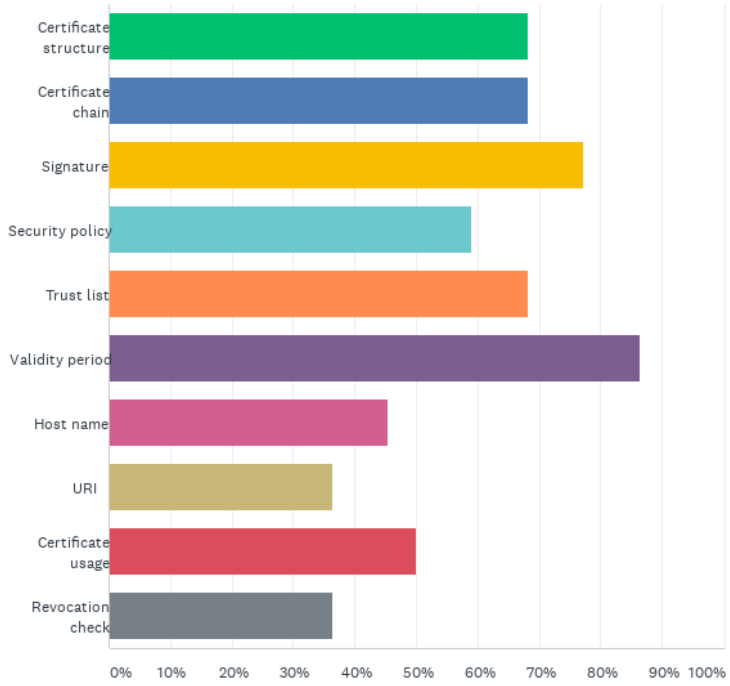

Abbildung 6 Unterstütze Zertifikatsüberprüfungen

Darüber hinaus gaben 11 Teilnehmer an, dass ihr Produkt nicht den Austausch des Applikations-Zertifikats unterstützt, z.B. durch ein von einer PKI ausgestelltes Zertifikat.

6.3.5 Produkt Entwicklung

Insgesamt gaben durchschnittlich 26 Teilnehmer eine Antwort in dieser Kategorie ab.

In dieser Kategorie wurde u.a. nach eingesetzten Methoden der sicheren Software-Entwicklung gefragt. Hier setzten bereits viele Teilnehmer solche ein, wie z.B. Sicherheitscodereviews (47%), statische Codeanalyse (60%) oder Risikoanalyse (52%), dargestellt in Abbildung 7. 60% der Teilnehmer berichten auch von Penetrationstests gegen das finale Produkt.

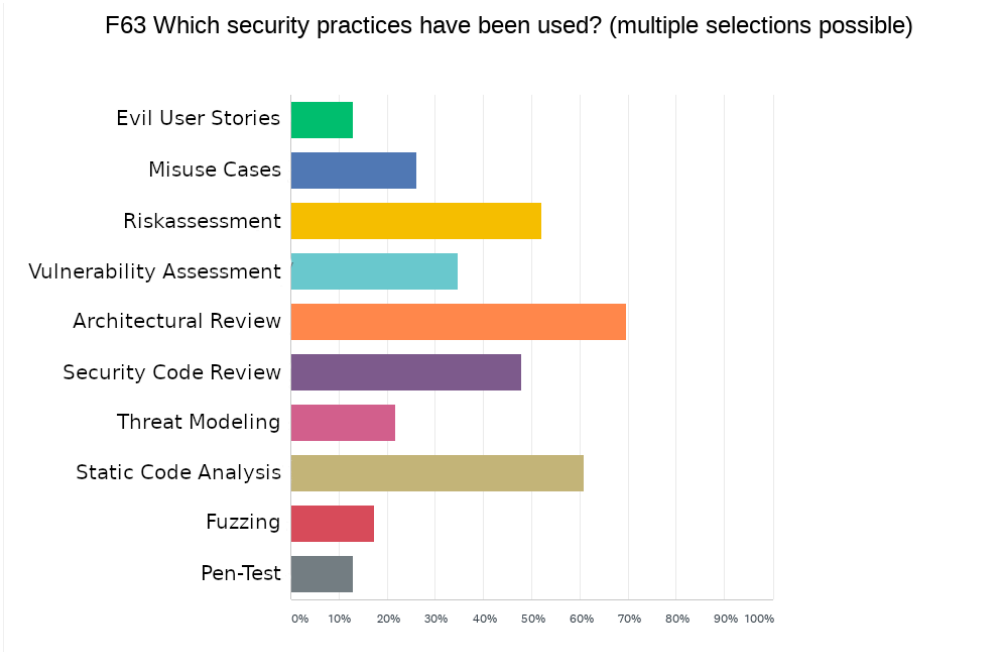

Abbildung 7 Eingesetzte Softwareentwicklungsmethoden

Updates werden bei den meisten Produkten (93%) manuell bereitgestellt und eingespielt, wobei ein Großteil keine festen Updatezyklen anbietet (56%).

Die meisten Teilnehmer setzen Sicherheitsanalysen entwicklungsbegleitend ein (69%), jedoch nur 17% auch nach der Produktveröffentlichung.

Von den Teilnehmern geben 23% an, keinen Überblick über die eingesetzten Drittanbieterabhängigkeiten (wie z.B. eingesetzte Software-Bibliotheken) zu haben.

6.3.6 Discovery Server

Insgesamt gaben durchschnittlich 42 Teilnehmer eine Antwort in dieser Kategorie ab.

Wie in Abbildung 8 dargestellt, geben nur 9% der Teilnehmer an, dass ihr Produkt Discovery-Dienste mit Multicast-Extension zur automatisierten Funktion unterstützt.

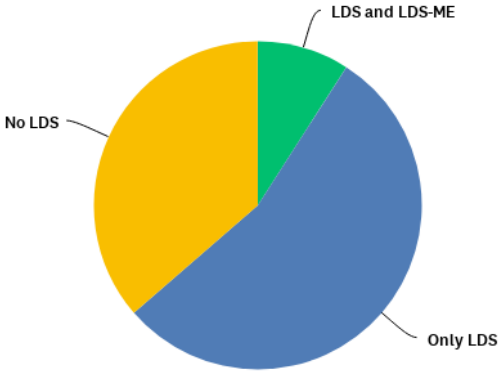

Abbildung 8 LDS-Unterstützung

6.3.7 Global Discovery Server

Insgesamt gaben durchschnittlich 13 Teilnehmer eine Antwort in dieser Kategorie ab. Aufgrund der geringen Teilnehmerzahl in diesem Bereich sind die Ergebnisse nicht belastbar.

Nahezu alle Teilnehmer geben an (95%), dass ihr GDS-unterstützendes Produkt nicht die vollständige Automatisierung des Registrierungsprozesses auf Basis der LDS-ME unterstützt.

Beim Zertifikatsmanagement unterstützen 18% sowohl das Push, als auch das Pull-Modell, bzw. 25% nur das Push-Modell, wie in Abbildung 9 zu sehen. Dabei kamen die Antworten aber zum Großteil von Teilnehmern mit SDK-Produkten.

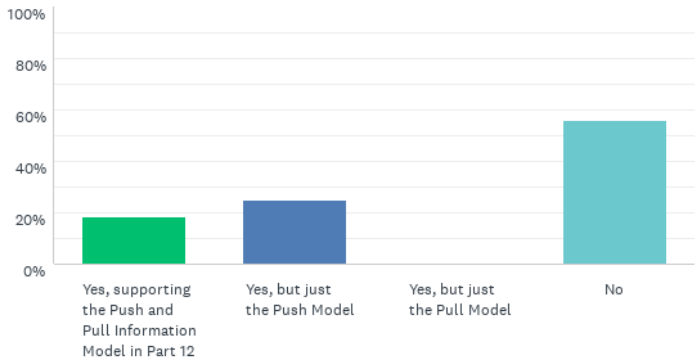

Abbildung 9 Unterstützung GDS-Push- und Pull-Modell

Jeweils nur ein Produkt gibt an, auch die KeyCredential-Dienste sowie die Authorization-Dienste zu unterstützen.

6.3.8 PubSub

Insgesamt gaben durchschnittlich 9 Teilnehmer eine Antwort in dieser Kategorie ab.

Aufgrund der besonders geringen Teilnehmerzahl in diesem Bereich sind die Ergebnisse nicht belastbar. Nur 14 Teilnehmer geben für ihr Produkt eine grundlegende Unterstützung von PubSub an, wobei 11 einen broker-losen Betrieb unterstützen (z.B. über UDP), und 5 einen broker-basierten Betrieb unterstützen (z.B. über AMQP oder MQTT).

6.4 Zusammenfassung der Ergebnisse

Aus den Erkenntnissen zur Durchführung der Marktbefragung, sowie den Ergebnissen daraus, können verschiedene Rückschlüsse auf sicherheitsrelevante Aspekte beim praktischen Einsatz von OPC UA gezogen und Empfehlungen daraus abgeleitet werden:

1 Zertifizierte Produkte und Hersteller
Aufgrund der wenigen Rückmeldungen von Herstellern zertifizierter OPC UA Produkte kann, über die Security Policies und UserIdentity Tokens hinaus, keine Aussage zur Umsetzung von weiteren sicherheitsrelevanten Funktionen gemacht werden.

2 Eingesetzte Stacks
Für das OPC UA Protokoll gibt es eine Vielzahl an verfügbaren Stacks, die die Entwicklung entsprechender Produkte ermöglichen. Diese reichen dabei von kommerziellen bis zu quelloffenen Varianten für verschiedene Umgebungen und Programmiersprachen. Dabei setzen, zumindest bei den Teilnehmern der Befragung, über dreiviertel auf Stacks mit zertifizierten Demo Applikationen.
Dies verleiht insbesondere den sicherheitsrelevanten Vorgaben der Zertifizierung besonderes Gewicht, da diese potentiell bereits einem Großteil der Endanwender zugutekommen. Im Zertifizierungsprozess wird dem teilweise schon Rechnung getragen.

3 Eingesetzte Versionen
Die untersuchte Version 1.04 der Spezifikation ist seit 2017 verabschiedet und verfügbar. Trotzdem wird diese von über einem Viertel der Produkte noch nicht unterstützt und ältere Versionen sind weiterhin relevant im praktischen Einsatz. Teilweise sind zudem keine Updates geplant.
Bei sicherheitstechnischen Entwicklungen sollten auch Auswirkungen auf ältere Versionen betrachtet werden und auch diese weiterhin sicherheitstechnisch gepflegt werden.

4 Sichere Konfiguration
Die sichere Grundkonfiguration steht für einen großen Teil der Produkte laut Umfrage nicht im Fokus, d.h. vom Standard nur empfohlene und nicht geforderte Sicherheitskonfigurationen sind in vielen Fällen nicht umgesetzt.
Die OPC UA Spezifikation sollte daher verbesserte Hilfestellungen für solche Grundkonfigurationen geben und fordern. Wie in Punkt 2) beschrieben, spielt der Zertifizierungsprozess und eine klare Dokumentation der Produktkonfiguration eine wichtige Rolle.

5 Unterstützte Sicherheitsfunktionen
Der Umsetzungsstand von Sicherheitsfunktionen, wie bestimmte Produktfunktionen oder die verfügbaren Zertifikatstests, zeigt, dass transparent für die Nutzer sein muss, welche Funktionen vorhanden sind und welche nicht. Die im Markt verfügbaren Produkte der Spezifikation werden im sicherheitsrelevanten Bereichen diesem Anspruch noch nicht vollständig gerecht.

6 Bei der Übertragung der Ergebnisse von den Sicherheitsuntersuchungen der Spezifikation auf verfügbare Produkte ist dies besonders zu beachten. Denn es kann nicht automatisch davon ausgegangen werden, dass alle Produkte, die OPC-UA einsetzen, auch alle sicherheitsrelevanten Funktionen implementiert haben und diese in dem gewünschten Anwendungsfall auch genutzt werden.

7 Unterstütze Funktionen/Features
Die unterstützen Funktionen, z.B. im Bereich der Discovery-Dienste oder PubSub, betonen ein weiteres Mal, dass der Funktionsumfang von Produkten im Markt die Möglichkeiten von OPC UA nur teilweise ausnutzt. Die Unterstützung von neuen Funktionalitäten setzt sich nur langsam durch.

7 Dynamische Sicherheitsanalyse

7.1 Vorgehen

Die Sicherheit des OPC UA Protokolls in Version 1.04 wurde anhand von open62541 als zertifizierte Serverimplementierung auf drei Arten dynamisch untersucht. Es wurden zwei Fuzzing-Ansätze verfolgt, ein Blackbox- und ein Whitebox-Ansatz, sowie ein Test auf Zertifikatsvalidierung umgesetzt. Die unterschiedlichen Analysen sollen möglichst viele Aspekte des Protokolls abdecken, sowie eine Mischung aus Tiefe und Breite in der Art der Analyse ermöglichen.

Das Blackbox-Fuzzing erlaubt eine breite Anwendbarkeit des verfolgten Ansatzes, da es eine implementierungsunabhängige Analyse erlaubt, die zudem frei verfügbar veröffentlicht wurde [9]. Das Whitebox-Fuzzing, basierend auf der open62541 Implementierung, zeigt, wie sicher eine gereifte Umsetzung des Protokolls gegenüber Laufzeitfehlern ist. Mit den Tests zur Zertifikatsvalidierung wird zudem ein Blick auf den SecureChannel geworfen und geprüft wie open62541, gegenüber manipulierten bzw. invaliden Authentifizierungsversuchen ist.

7.2 Black-Box Fuzzing des OPCUA Protokolls

Fuzzing von Programmen die Dateien, Netzwerkverkehr oder andere Daten verarbeiten, die in einem wohldefinierten Protokoll definiert sind, können i.d.R. auf zwei Arten mittels Fuzzing analysiert werden: Zum einen kann ein programmspezifischer Fuzzer entwickelt werden, der sich am Datenfluss innerhalb des Programms orientiert und somit auf das Programm zugeschnitten wird. Zum anderen kann ein Fuzzer auf Basis des Protokolls entwickelt werden, der einzelne oder mehrere Felder innerhalb des Protokolls mutiert und damit auf beliebige Programme angewendet werden kann die mittels des Protokolls interagieren. Programmspezifische Fuzzer sind meist effizienter für das Zielprogramm, während protokollspezifische Fuzzer breiter einsetzbar sind.

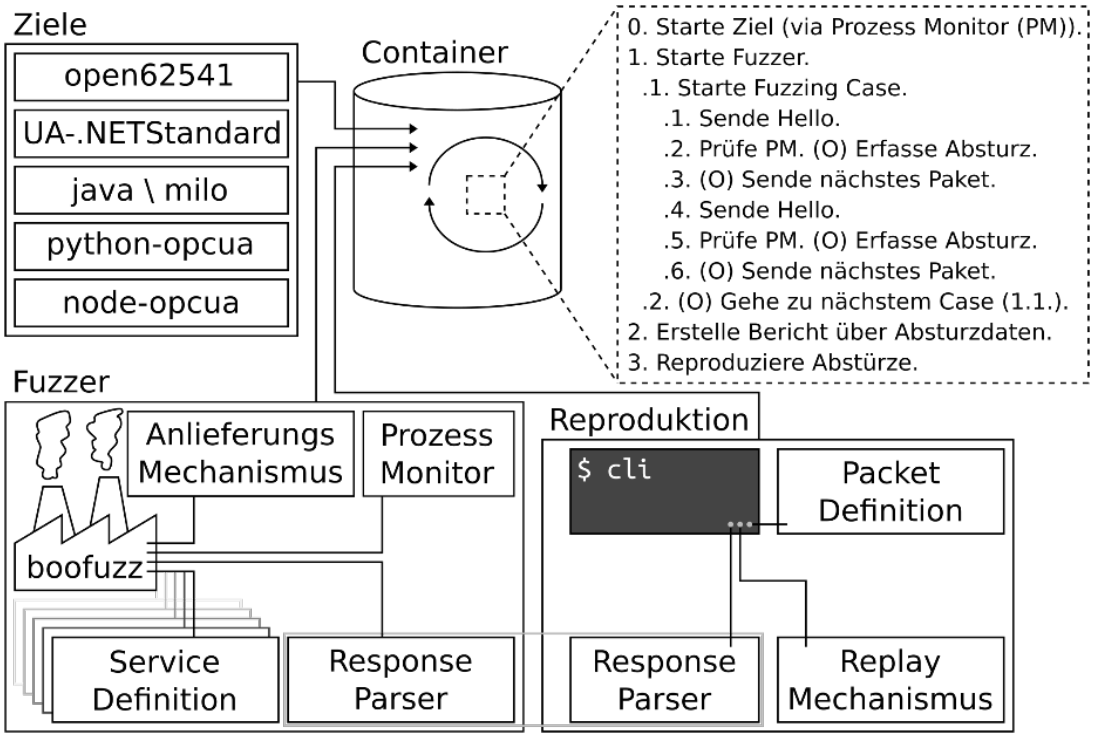

Abbildung 10 Vollständiger Mechanismus zum Black-Box-Fuzzing mit Ablauf innerhalb des erzeugten Containers.

7.2.1 Fuzzing Mechanismus

Der Begriff Fuzzer ist in der Fachliteratur überladen. Während in der ursprünglichen Bedeutung ein Fuzzer lediglich für die Mutation von Eingabedaten verwendet wird, bezeichnet dieser oft auch Frameworks die neben der Mutation auch noch andere Aspekte der Analyse, die beispielweise den Anlieferungsmechanismus beinhaltet oder sogar fertige Werkzeuge. Im Folgenden wird Fuzzer als Synonym mit dem Zusammenschluss der verschiedenen Aspekte der Analyse verwendet.

Für die Umsetzung des Black-Box-Fuzzing wurde das Fuzzing-Framework boofuzz [10] als Grundlage ausgewählt. boofuzz wurde zum Fuzzen von Netzwerk-Protokollen entwickelt und bietet mehrere wichtige Komponenten, die für unsere Entwicklung genutzt werden konnten. Zunächst liefert boofuzz eine Bibliothek zur Beschreibung von Protokollstrukturen. Diese erlaubt es die Schritte eines Protokolls einzeln zu definieren und gleichzeitig festzulegen, welche der Felder später mutiert werden sollen. Zusätzlich bietet boofuzz einen Anlieferungsmechanismus, der die zuvor definierten Schritte an einen Server senden kann. Zuletzt bringt boofuzz auch einen einfachen Prozessmonitor mit, der erkennt, ob der Server abgestürzt ist und diesen neustarten kann, während er die Fehlermeldungen beim Absturz speichert und für die Analyse nutzbar macht.

In Abbildung 10 sind alle Komponenten des entwickelten Fuzzing-Mechanismus dargestellt. Neben dem eigentlichen Fuzzer gibt es noch den Reproduktionsmechanismus und die Fuzzing-Ziele, sowie ein zentrales Programm das den kompletten Ablauf der Analyse steuert. Alle Komponenten werden in einem Container, einer isolierten Laufzeitumgebung innerhalb des Analyse-PCs, aufgesetzt und dann vom zentralen Programm aus angesteuert. Das Aufsetzen des Containers erfolgt einmalig für jeden Analysedurchlauf. Jeder Analysedurchlauf wiederum besteht aus einem vollständigen Test einer einzelnen Implementierung, also z.B. von open62541. Nach Ende des Analysedurchlaufs liegen die Ergebnisse außerhalb des Containers vor und dieser kann gelöscht werden. Die vier übergeordneten Schritte jedes Durchlaufs bestehen aus

1. der Einrichtung des Containers, inklusive Kompilierung oder Einrichtung der gewählten Zielimplementierung,
2. dem Fuzzing-Durchlauf auf allen umgesetzten Protokolldefinitionen,
3. der Sammlung von Absturzinformationen und
4. der Reproduktion aller erkannter Abstürze.

Die Einrichtung des Containers, abgesehen von der Zielimplementierung, ist für alle Fälle gleich und besteht in erster Linie aus der Einrichtung von boofuzz. Für jede Zielimplementierung wurde ein repräsentativer Beispielserver verwendet, für dessen Einrichtung eigene Installationsskripte entwickelt wurden.

Tabelle 12 Mutationen in der Hello-Nachricht

Feld	*Typ (vereinfacht)*	*Mutation*
Nachrichten ID (HEL)	String	Nein
Chunk Typ (F)	Char	Nein
Länge (Dynamisch)	Int	Nein
Protokoll Version	Int	Ja
Empfänger Buffer-Größe	Int	Ja
Sender Buffer-Größe	Int	Ja
Maximale Nachrichtenlänge	Int	Ja
Maximale Chunk-Anzahl	Int	Ja
Länge der Endpunkt URL (Dynamisch)	Int	Ja
Endpunkt URL	String	Ja

Der vollständige Fuzzing-Durchlauf kann in Nachrichten und Felder untergliedert werden. Die Hello Nachricht besteht etwa aus 10 Feldern von denen 7 mutiert werden. Aus der Mutation dieser 7 Felder werden insgesamt 2261 Testfälle erstellt. Tabelle 12 zeigt die Felder der Hello Nachricht und ihren Typ. Die ersten drei

Felder, Nachrichten ID, Chunk Typ und Länge, existieren bei allen OPC UA Nachrichten. Mutiert man diese Felder wird die Kommunikation zwangsläufig korrumpiert, da entweder die Nachricht aufgrund einer fehlenden ID nicht als OPC UA Nachricht erkannt wird, der falsche Chunk Typ dazu führt, dass auf ein Paket gewartet wird, das niemals kommt oder die Nachricht aufgrund falscher Länge zu früh oder zu spät abgeschnitten wird und sich daraus korrupte Nachrichten ergeben. Alle weiteren Felder werden der Reihe nach untersucht und ergeben in Summe den Fuzzing-Durchlauf für die Hello-Nachricht.

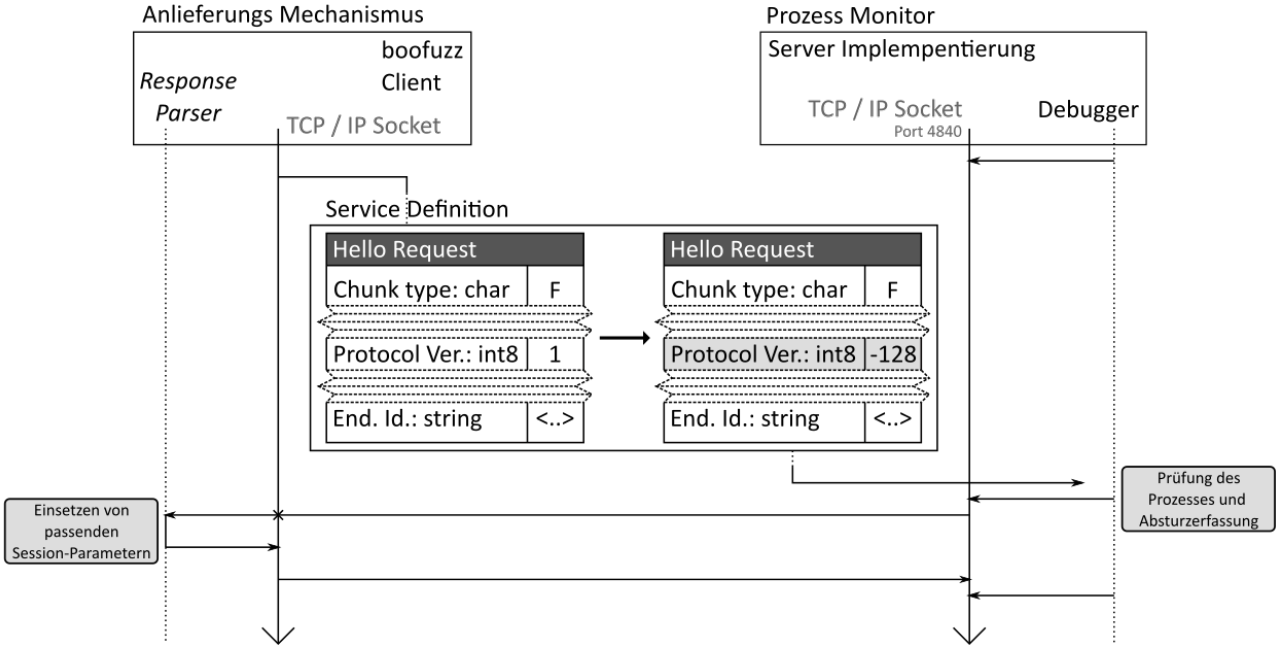

Abbildung 11 Fuzzing Mechanismus bestehend aus Anlieferungsmechanismus mit angeschlossenen Komponenten zum Parsern und Mutieren der Nachrichten und dem Prozess Monitor zum Starten und Überwachen des Servers.

Abbildung 11 zeigt den Ablauf eines einzelnen Tests der Hello Nachricht am Beispiel des Protokoll Version Felds. Die Client-Seite beinhaltet Anlieferungsmechanismus, Response Parser und die Grammatiken der Protokoll-Definitionen. Die Server-Seite beinhaltet den Prozessmonitor sowie die instrumentierte Zielimplementierung. Basierend auf der in der boofuzz-Grammatik umgesetzten Definition der Nachricht wird ein einzelnes Feld mutiert und das resultierende Paket an den Server geschickt. Das resultierende Feld hat im dargestellten Fall den niedrigsten möglichen Wert den ein Feld vom Typ Int8, also einem 8-Bit Integer, annehmen kann. Der Prozessmonitor überwacht den Server und erkennt, dank des angeschlossenen Debuggers, falls der Prozess abstürzt. Im Falle eines Absturzes erfasst der Prozessmonitor den Fehlercode, die Ausgabe und die Konsolenausgabe des Servers. Diese Informationen können später für die Auswertung des Absturzes genutzt werden. Ein wichtiger Punkt in der Umsetzung tieferer Nachrichten im OPC UA Handshake ist die dynamische Anpassung der Session-Parameter. Während die Modellierung des Protokolls mittels der boofuzz-Grammatik grundsätzlich statische Basiswerte, die in Folge mutiert werden, vorsieht, müssen für zustandsbasierte Protokolle wie OPC UA die Zustandsparameter dynamisch gesetzt werden. Bei OPC UA sind das die Kanal- und Session-Parameter. Diese Parameter werden im Fuzzing-Mechanismus mittels des Response Parsers aus den Servernachrichten gewonnen und dann in die folgenden Client-Nachrichten eingesetzt.

Die Datensammlung nach Abschluss des vollständigen Fuzzing-Durchlaufs verbindet dann die vom Prozess-Monitor gesammelten Information und die Log-Daten des Fuzzers um eine vollständige Reproduzierbarkeit der gefundenen Abstürze zu ermöglichen. Zu den relevanten Informationen gehören

- Nummer des Testfalls
- Typ der Nachricht
- Mutiertes Feld und exakte Mutation

- Alle gesendeten und empfangenen Nachrichten in Binärdarstellung
- Ausgabe und Absturznachrichten des Servers, sowie Fehlercodes.

Die Protokollierung der Testfälle erfolgt durch boofuzz automatisch in Form einer SQLite Datenbank. Diese assoziiert jeden Schritt eines Testfalls mit dessen Nummer, sodass die ersten vier Informationen direkt aus der Datenbank gewonnen werden können. Ein erkannter Absturz findet sich auch in der Datenbank, sodass die Fälle diesbezüglich gefiltert werden können. Die Serverausgabe, Absturzmeldungen und Fehlercodes werden vom Prozessmonitor erkannt, der keinen Zugriff auf die Datenbank hat und daher ein eigenes Ausgabeformat nutzt. Da dieses unstrukturiert ist und daher nicht fehlerfrei verarbeitet werden kann, wurde boofuzz an dieser Stelle erweitert, um eine Ausgabe im JSON-Format zu erlauben. Diese Anpassung erlaubt es dann die Daten aus Datenbank und JSON-Datei zu verbinden, um ein detailliertes Protokoll aller Abstürze zu erhalten.

Basierend auf den Absturzdaten wird abschließend eine Reproduktion der Abstürze versucht. Dazu wird nacheinander für jeden Testfall, bei dem ein Absturz erkannt wurde, der Server gestartet und alle Nachrichten des Testfalls inklusive der letzten mutierten Nachricht wiederholt. Kommt es dabei erneut zum Absturz, wird der Absturz als reproduzierbar definiert. Die Information ob ein Testfall reproduzierbar ist wird im Absturzprotokoll ergänzt.

7.2.2 Ergebnisse des Blackbox-Fuzzing auf Open Source Implementierungen

Mithilfe der Bibliothek zur Protokollbeschreibung wurden verschiedene OPC UA Dienste definiert. Abbildung 11Tabelle 13 zeigt die Sammlung aller Pakete bzw. Dienste die definiert wurden und damit auch mittels Fuzzing untersucht werden können. Wie in der Tabelle erkennbar, ist die Anzahl an Mutationen für jeden Dienst festgelegt. Dies resultiert aus der Methodik die boofuzz für die Mutation einsetzt. Einzelne Felder des Protokolls werden nicht zufällig, sondern auf Basis von bewährten Manipulationen verändert. Für Felder mit Zahlen, wie etwa Identifikatoren, werden u.a. 0, -1 und die größte sowie kleinste mögliche Zahl eingesetzt. Felder mit Text werden u.a. stark verkürzt und verlängert und es werden bestimmt Buchstaben durch Bytes ersetzt, die keine Buchstaben, sondern Kontrollsymbole wie *end of file* (dt.: Dateiende) darstellen. Neben einer besseren Effizienz im ressourcenbeschränkten Netzwerkkontext bietet diese Determiniertheit auch die Möglichkeit, Fälle auf Basis ihres Indexes zu wiederholen, sowie ganze Testläufe zu reproduzieren. Das Werkzeug bietet somit auch eine sinnvolle Ergänzung für Prüfzyklen.

Tabelle 13 Pakete bzw. Dienste die im Fuzzing Mechanismus umgesetzt wurden mit Angabe der Anzahl an Mutationen pro Paket.

Paket	*# Mutationen*
Handshake	8392
Hello	2261
OpenSecureChannel	5287
CloseSecureChannel	844
Discovery Service Set	13357
FindServers	2624
FindServersOnNetwork	1264
RegisterServer2	6845
GetEndpoints	2624
Session Service Set	10016
CreateSession	5984

Paket	*# Mutationen*
ActivateSession	4032

Die dynamische Analyse des Protokolls wurde anhand von insgesamt fünf Implementierungen durchgeführt. Bei drei Implementierungen, open62541, node-opcua und dem OPC UA .NET Standard konnte dabei ein als CTT-Ziel definierter Server verwendet werden. Bei den anderen beiden Implementierungen wurde ein Referenzbeispiel verwendet. Alle fünf Implementierungen basieren auf unterschiedlichen Programmiersprachen, open62541 auf C, die Implementierung der OPC-Foundation auf C#, Eclipse Milo auf Java, pyhton-opcua auf Python und node-opcua auf JavaScript. Die Ziele decken damit natürlicherweise eine Reihe von Anwendungsszenarien ab. Alle Implementierungen wurden in einer aktuellen Analyse von öffentlich zugänglichen OPC UA Servern nachgewiesen [8].

Aufgrund der Determiniertheit der boofuzz-Mutationen war es prinzipiell nicht nötig, die Fuzzing-Durchläufe zu wiederholen. Dennoch wurde das Fuzzing für jede Implementierung drei Mal wiederholt. Es ergaben sich dabei keine Abweichungen. Alle Durchläufe wurden auf einem normalen Desktop PC mit einer 8-Kern AMD CPU und 32 GB RAM durchgeführt. Der Desktop PC wurde mittels eines Ubuntu 20.04 Host-Betriebssystem betrieben. Die Fuzzing Durchläufe fanden innerhalb des in Abschnitt 7.1.1 vorgestellten Containers statt, der für jede Implementierung und jeden vollständigen Durchlauf neu erstellt wurde. Die Laufzeit der einzelnen Durchläufe war, ebenfalls relativ konstant zwischen den jeweils drei Durchläufen pro Ziel und, bewegte sich zwischen 1:37 h und 11:07 h.

Tabelle 14 Liste der untersuchten Produkte mit Angabe von Sprache, gefundenen Abstürzen und reproduzierbaren Abstürzen

Produkt	*Sprache*	*Anzahl Abstürze*	*Reproduzierbare Abstürze*
open62541	C	184	0
.NET-Refererenzimplementierung	C#	0	0
Eclipse Milo	Java	167	0
python-opcua	Python	188	0
node-opcua	JavaScript	13	13

Tabelle 14 zeigt eine Übersicht der gefundenen Abstürze. Zwei Beobachtungen sind dabei besonders auffällig. Nur für die JavaScript-Implementierung wurden reproduzierbare Abstürze gefunden und nur für die C#-Implementierung wurde kein Absturz gefunden. Die Tatsache, dass nicht-reproduzierbare Abstürze reproduzierbar sind, wenn man vollständige Fuzzing-Durchläufe wiederholt, suggeriert, dass die Abstürze durch die Abfolge mehrerer Verbindungen mit mutierten Paketen zurückgehen. Genaue Analysen der nicht-reproduzierbaren Abstürze zeigen, dass die Server zwar noch laufen, aber keine Nachrichten mehr entgegennehmen.

Im Fall von open62541 etwa gibt es zwei Ursachen: Zum einen wird das Limit an Verbindungen erreicht. Dies ist das erwartete Verhalten und wird nicht als Fehler eingestuft. Zum anderen werden mutierte NodeIDs nicht erkannt und daher die Verbindungen beendet. Auch dies ist ein normales Verhalten. Für beide Ursachen könnte eine Anpassung an boofuzz ermöglichen, die fehlenden Antworten korrekt einzuordnen.

Auch bei Eclipse Milo, der Java Implementierung, handelt es sich in den meisten Fällen um falsch erkannte Abstürze. Es treten zwar interne Fehler bei der Verarbeitung der mutierten Pakete auf, aber die Fehler werden zentral abgefangen und in Konsequenz die Verbindung beendet. Eine neue Verbindung ist auch ohne Neustart des Servers möglich.

Andere Fehler in der Java-Implementierung und der Python-Implementierung konnten nicht genau nachvollzogen werden, da das Verhalten nicht außerhalb der Fuzzing-Durchläufe reproduziert werden konnte. Eine mögliche Ursache für die Abstürze sind Folgefehler aus den vorangegangenen Verbindungen. Für eine genaue Begründung müsste allerdings eine detaillierte Analyse des Quellcodes der Ziele durchgeführte werden.

Die 13 Abstürze auf der JavaScript-Implementierung gehen auf einen gemeinsamen Fehler in der Implementierung der Datenverarbeitung zurück. Die Abstürze werden zwar durch das Mutieren verschiedener

Felder in verschiedenen Nachrichten herbeigeführt, anhand der Absturznachrichten kann aber ein gemeinsames Problem identifiziert werden. Die Fehler wurden durch das Projekt nach der Information geschlossen.

Insgesamt zeigt die Auswertung der Fuzzing Durchläufe eine positive Bilanz für die untersuchten Implementierungen. Aufgrund der endlichen Anzahl an Testfällen, die durch die boofuzz-Mutation vorgegeben werden, kann nicht ausgeschlossen werden, dass ein ausschöpfendes mutierende der Nachrichtenfelder weitere Abstürze hervorbringen würde. Insbesondere in einem Netzwerk-Szenario geht eine Ausweitung der Testfälle aber auch mit einem deutlichen Anstieg an Laufzeit einher. Während boofuzz also mittels kleiner Anpassungen eine erschöpfende Mutation des Wertebereiches von Feldern fester Größe zulässt, muss der Mehrwert dieser zusätzlichen Testfälle mit dem Mehr an Laufzeit in Verhältnis gesetzt werden.

7.3 White-Box Fuzzing der Implementierung

Die open62541 Implementierung wurde in Googles oss-fuzz Projekt [11] integriert. Von Seiten der open62541 Implementierung wurden dafür mit Hilfe von libfuzzer 12 potentiell kritische Implementierungsdetails in Form von Fuzzing Testfällen abgedeckt. Diese konzentrieren sich auf das Verarbeiten von Binärnachrichten im Allgemeinen als auch das En- bzw. Dekodieren dieser innerhalb des Protokollstandards. Da diese Komponenten direkt mit Daten interagieren die ein Angreifer manipulieren kann, ist die Wahrscheinlichkeit, dass Programmierfehler zu Sicherheitsproblemen führen können, besonders hoch. Die Einbindung in das vollständig automatisierte oss-fuzz liefert damit eine gewisse Grundsicherheit, sofern das Fuzzing auch durchgängig auf dem aktuellsten Stand der Implementierung durchgeführt wird. Während die oss-fuzz Integration seit August 2020 regelmäßig fehlerhafte Builds aufweist [12] wurden in der Vergangenheit auch eine Reihe von relevanten Fehlern aufgedeckt [13], so dass die Annahme naheliegt, dass durch weiteres extensives Fuzzing auch weiterhin noch neue Fehler aufgedeckt werden können.

Daher wurde im Rahmen dieser Sicherheitsanalyse versucht, die Integration in oss-fuzz lokal nachzustellen, um zu überprüfen, ob die Ergebnisse valide sind. Zusätzlich soll durch die Durchführung der Untersuchung auf kontrollierter Hardware zusätzliche Gewissheit über die Prüfumgebung gegeben werden, die bei der cloudbasierten oss-fuzz Integration nicht gegeben ist. Ein mögliches Ausbleiben von gefundenen Fehlern bestätigt dann den Qualitätsanspruch der durch die oss-fuzz Integration gestellt wird.

Nachdem zunächst eine Umsetzung mittels der zentralen oss-fuzz Komponente clusterfuzz angestrebt wurde, hat sich eine direkte Nutzung von libfuzzer als erfolgreicher und wartbarer ergeben. Auf Basis des letzten stabilen oss-fuzz Builds [14] wurden die 12 vorhandenen Fuzzing Testfälle in eine eigene libfuzzer gestützte Umgebung portiert. Das Fuzzing lief daraufhin lokal für 117 Tage auf einer Ubuntu 18.04.5 LTS Maschine mit einem 8 Kernen und 32 GB RAM. In dieser Laufzeit konnten die durch oss-fuzz gefundenen und reproduzierbaren Fehler nicht erneut gefunden werden. Beim genaueren Betrachten unterliegen alle dort noch als "Status: New" hinterlegten Fehler einer Regression.

Für den Testfall **fuzz_json_decode_encode** wurden durch libfuzzer mit ASAN Unterstützung eine Reihe von *Memory Leaks* (dt. Speicherleck) erkannt. Diese haben alle eine ähnliche Form. Die meisten verweisen auf einen 4-Byte Leak und weisen beim Reproduzieren den gleichen Stack-Trace auf. Damit lassen sich nach erster Analyse alle aufgezeichneten Abstürze dieser Art auf die gleiche Ursache zurückführen.

Für den Testfall **fuzz_json_decode** wurde durch libfuzzer mit ASAN Unterstützung ein reproduzierbarer Absturz gefunden. Es handelt sich um einen *Heap Buffer Overflow*. Der Fehler wurde an die Entwickler gemeldet und in der Zwischenzeit behoben.

Darüber hinaus wurden zwei nicht reproduzierbare Abstürze, als auch ein nicht reproduzierbarer Memory Leak für den Testfall **fuzz_tcp_message** gefunden.

Setzt man die Summe der gefundenen Probleme, einen Absturz und eine Reihe von Leaks, in Verhältnis zu der großen Menge an getesteten Code, hat das umgesetzte Whitebox Fuzzing den Qualitätsanspruch des OPC UA Protokolls und der open62541 Implementierung eher bestätigt. Die Ergebnisse sind zudem in etwa

vergleichbar zu der cloudbasierten oss-fuzz Integration, wenn man die Summe der dort gefundenen Abstürze betrachtet. Die Tatsache, dass weitere mögliche Probleme gefunden wurde bekräftigt aber die Notwendigkeit einer ständigen Integration des Whitebox-Fuzzing in die Entwicklung. Es sollte also darauf geachtet werden, dass Build-Fehler in der oss-fuzz Integration erkannt und behoben werden. Ein nächster Schritt wäre die vorhandenen Testfälle, wo sinnvoll, zu erweitern um weitere Codebereiche abzudecken.

7.4 Test der Zertifikatsbehandlung

Für die Untersuchung der Sicherheit des neuen OPC UA Standards 1.04 wurde auf Basis von open62541 ein Test der Zertifikatvalidierung umgesetzt, der vergleichbar zur Studie des BSI aus dem Jahr 2016 ist. Als Werkzeug für die Erzeugung von Testzertifikaten wurde das vom BSI bereitgestellte Certification Path Validation Test Tool (CPT) verwendet [15].

7.4.1 Prüfmechanismus

Abbildung 12 zeigt die Komponenten des entwickelten Prüfmechanismus. Wie im in Abschnitt 7.2.1 beschriebenen Fuzzing-Mechanismus wird auch hier auf einen Container zurückgegriffen um ein isoliertes Zielsystem umzusetzen. Der Container, der den Zielserver, wiederum der CTT Server aus den open62541 Quellen, sowie die zum jeweiligen Client-Zertifikat passende Zertifikatskette enthält, wird für den Test jedes Client-Zertifikats neu aufgesetzt und danach gelöscht. Dadurch ist jeder Test voneinander isoliert Als Kryptographiebibliothek wird mbedTLS verwendet. Während open62541 sowohl mit mbedTLS als auch OpenSSL kompatibel ist, erfolgt die Zertifizierung mit CTT basierend auf der mbedTLS-Variante. Der Client, basierend auf python-opcua, erhält jeweils das zu untersuchende Client-Zertifikat und startet einen Verbindungsaufbau zum durch den Container bereitgestellten Server. Die Hello Nachricht enthält noch keine Zertifikate, sodass erst nach der Übermittlung des *OpenSecureChannelRequest* entschieden werden kann, wie der Server das Client-Zertifikat behandelt. Wird das Zertifikat akzeptiert, antwortet der Server mit einer *OpenSecureChannelResponse*-Nachricht. Lehnt der Server das Zertifikat ab, ist die Antwort eine *ErrorMessage*-Nachricht. Basierend auf der Antwort erkennt der Prüfmechanismus also, ob das Zertifikat akzeptiert wurde oder nicht. Die Ausgabe des Prüfmechanismus‘ erfolgt mittels eines einfachen Text-Logs.

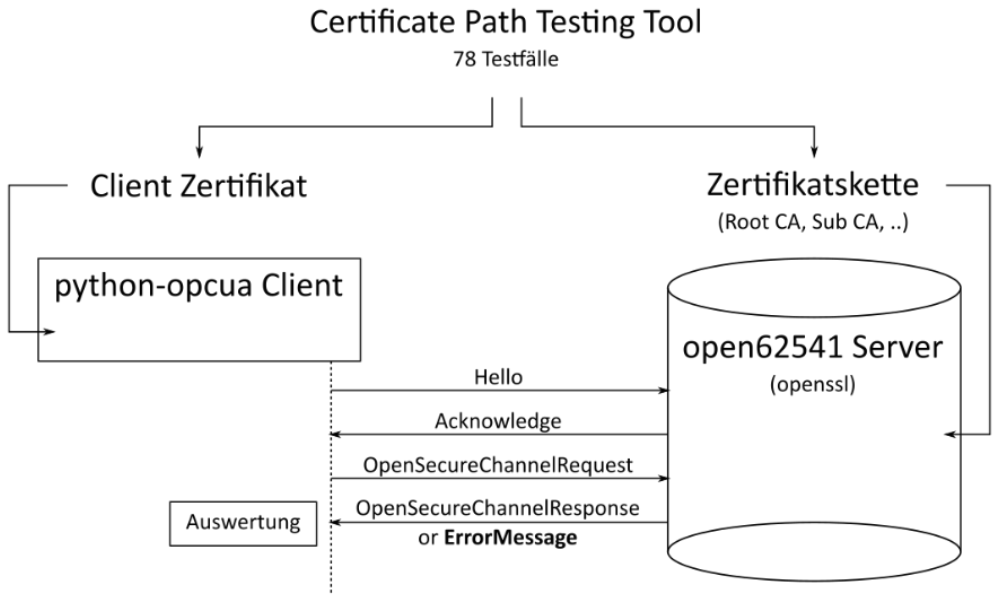

Abbildung 12 Prüfmechanismus für die Zertifikatsvalidierung, basierend auf 78 vom CPT generierten Kombinationen aus Client Zertifikat und dazugehörigen Zertifikatsketten.

Je nach Testfall beinhaltet die Zertifikatskette mindestens ein Root-Zertifikat, sowie ein oder zwei Zwischenzertifikate. Die Zertifikatskette wird dem Server mittels der Option `--trustlistFolder` mitgeteilt. Zusätzlich wurden für alle Tests, die keine eigene Widerrufsliste definie-

ren, leere Widerrufslisten erzeugt. Widerrufslisten werden verwendet um Zertifikate als unvertraut zu markieren, die ansonsten valide sind. Für den Test war es nötig jeder Kette Widerrufslisten zuzuordnen, da der Prüfzyklus im OPC UA Standard dies explizit vorsieht. CPT generiert Widerrufslisten nur zu den Tests, die den Umgang mit den Listen prüfen. Die Widerrufslisten werden mit der Option `--revocationlistFolder` übergeben. Neben dem Generieren von Widerrufslisten, wurde CPT so konfiguriert, dass die im OPC UA Standard geforderten Erweiterungen innerhalb der Zertifikate korrekt gesetzt werden. Dazu gehören unter anderem die Werte der Key Usage-Erweiterung und die URI der Zielapplikation.

7.4.2 Auswertung der Ergebnisse

Das CPT definiert insgesamt 94 einzigartige Testfälle. Es gibt fünf Klassen von Tests mit insgesamt 41 Tests die auf diese Prüfung nicht anwendbar sind. Dazu gehören E-Mail Zertifikate, TLS- und IPsec-spezifische Test sowie Tests des Online Certificate Status Protocol (OCSP), einem Mechanismus um den Widerrufsstatus von Zertifikaten Online abzufragen. Von den übrigen 53 Testfällen waren zwei nicht nutzbar, da CPT diese nicht erzeugen konnte. Eine initiale Untersuchung deutet auf einen Fehler im CPT Mechanismus hin.

Die verbleibenden 51 Testfälle ergaben folgende Resultate:

- 7 Zertifikate wurden akzeptiert.
- 2 Tests führten zu einem Fehler im Testzyklus.
- 42 Zertifikate wurden abgelehnt.

Von den zwei Fällen die zu einem Fehler führen, handelt es sich einmal (COMMON_05) um ein strukturell korruptes Client-Zertifikat. Dieses kann der python-opcua Client nicht parsen und daher auch nicht zur Erstellung einer Nachricht an den Server senden. Da der Testfall gerade die Behandlung von strukturell korrupten Zertifikaten testen soll, ist dies zu erwarten. Es kann keine Aussage über die Behandlung des korrupten Zertifikats seitens des Servers getroffen werden. Der zweite Fehler (EXT_14) tritt auf, weil für den Testfall kein Client Zertifikat generiert wird. Dies scheint ein Fehler im CPT zu sein.

Von den sieben akzeptierten Zertifikaten ist eines (COMMON_01) ein positiver Testfall, der prüfen soll, dass der Server valide Zertifikate akzeptiert. Zwei der Testfälle (COMMON_06, COMMON_12) sollten zwar abgelehnt werden, CPT definiert hier aber Ermessensspielraum. Es handelt sich um Tests zu überlangen Namen und negativen Seriennummern. Die Fälle haben keinen direkten Einfluss auf die Sicherheit der Zertifikate. In einem Fall (COMMON_14) wird geprüft, ob ein Server ein Zertifikat ablehnt, wenn es sowohl eine vertraute Kette, als auch eine invalide Kette gibt. Der Aufbau des Prüfzyklus führt hier zu einem Falsch-Positiv, da nur die korrekte Kette hinterlegt wird. Es kann keine Aussage getroffen werden, wie der Server in einem korrekt konstruierten Szenario reagiert. Die drei übrigen Fälle (ALGO_STRENGTH_01, CRYPT_01, EXT_17) behandeln kryptographische Unsauberkeiten bzw. Erweiterungen in Zwischenzertifikaten. Während die Fälle nicht in direktem Widerspruch zum OPC UA Standard stehen, deuten die Tests auf eine suboptimale Konstruktion der Vertrauenskette auf Seiten der open62541-Implementierung hin. Die Ergebnisse der Tests wurden an die Entwickler gemeldet und die Konstruktion der Vertrauenskette soll dahingehen überarbeitet werden, dass hier das gewünschte Verhalten eintritt.

Bei den 42 abgelehnten Zertifikaten handelt es sich sämtlich um negative Testfälle. Somit tritt mit Ablehnung das erwartete Verhalten ein. Stichprobenartige Untersuchungen der Begründungen haben ergeben, dass die Zertifikate aus den in CPT beschriebenen Gründen abgelehnt wurden. Ein vollständiger Abgleich der Begründungen wurde nicht durchgeführt.

7.5 Zusammenfassung der Ergebnisse

Die dynamischen Analysen des OPC UA Protokolls, aufgeteilt in Blackbox- und Whitebox-Fuzzing sowie Tests der Zertifikatsvalidierung haben wenige Probleme in den untersuchten Implementierungen aufge-

zeigt. Mittels des Blackbox-Fuzzing wurde in erster Linie ein Fehler in der untersuchten JavaScript-Implementierung gefunden. Da das dazu entwickelte Werkzeug quelloffen und frei verfügbar gemacht wird, kann es auch für zukünftige Sicherheitsprüfungen verwendet werden. Das Whitebox-Fuzzing hat einen reproduzierbaren Fehler in der open62541-Bibliothek identifiziert, der gemeldet und vor Ablauf der Studie bereits behoben wurde. Die fehlgeschlagenen Tests in der Untersuchung der Zertifikatsvalidierung haben eine standardkonforme, wenn auch nicht optimale Konstruktion der Vertrauenskette in open62541 aufgezeigt. Die überwiegende Mehrheit der Zertifikate wurde jedoch korrekt behandelt.

8 Statische Codeanalyse

8.1 Vorgehen

Zur Analyse von open62541 wurden sowohl automatische Programme eingesetzt, als auch eine manuelle Codeanalyse für sicherheitskritische Bereiche der Implementierung durchgeführt. Als automatische Codeanalysetools kamen dabei Cppcheck, FramaC und Clang zum Einsatz.

8.2 Cppcheck

Cppcheck ist ein statisches Codeanalysetool für C und C++ und findet mögliche Schwachstellen, wie beispielsweise Dead Pointer, Integer Overflows, Null Pointer oder uninitialized Variables. Insbesondere können Buffer Overflows Auswirkungen auf die Security haben. Für die Durchführung des Tests wurde Cppcheck in der Version 1.90 verwendet.

Für die Analyse mittels Cppcheck wurde der folgende Aufruf verwendet:

```
cppcheck --project=open62541/build/compile_commands.json --enable=all --
std=c99 -i open62541/build/
```

Die folgenden Ergebnisse beziehen sich auf die Analyse von open62541 in der Version 1.1.1. In der Ausgabe sind 636 Meldungen. Um diese Meldungen auf die wichtigen Meldungen, die Einfluss auf die IT-Sicherheit haben können zu reduzieren, wurde das Verzeichnis deps/ exkludiert, da diese externe dependencies beinhalten. Meldungen vom Typ note, information und style sind ebenfalls ausgefiltert worden, da diese ebenfalls keinen Einfluss auf die IT-Sicherheit haben.

Somit verbleiben 70 Meldungen. Unter diesen sind 11 Fehler:

1 2x `(Error) Memory leak`

   a `./src/server/ua_server_discovery_mdns.c, Zeile 236: Memory Leak: newUrl`
   Dies ist ein Fehler und kann bei häufigen Aufrufen dazu führen, dass Speicher nicht wieder freigegeben wird. Das Memory Leak entsteht dadurch, dass der url->data Pointer mittels eines neuen Pointers überschrieben wird, bevor der alte freigegeben wurde.
   Score: 5.3 (CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L)

   b `./src/client/ua_client_subscriptions.c, Zeile 642: Memory Leak: array`
   Dies ist kein Fehler. Die betreffende Variable array wird hier allokiert und in data->mis abgelegt. Data wiederum wurde in cc->clientData gespeichert und letztlich auch zurückgegeben. Somit entsteht hier kein memory leak.

2 `(Error) Uninitialized variable`

   a `./src/pubsub/ua_pubsub_networkmessage.c, Zeile 702 + 1122 + 1298, Uninitialized Variable: byte`
   An allen drei Stellen wird die Variable nur angelegt, um mittels UA_Byte_calcSizeBinary, die Größe eines Byte abzufragen, wozu die Variable und deren Inhalt nicht von Bedeutung sind. Daher ist dies kein Fehler.

   b `./src/server/ua_services_view.c, Zeile 134, Uninitialized Variable: dummy`
   Diese Variable wird zwar nicht initialisiert, allerdings werden den benötigten Feldern direkt im Anschluss Werte zugewiesen. Dies könnten auch bei einer Initialisierung gesetzt werden. Dies erzeugt keinen Fehler.

   c `./src/server/ua_services_attribute.c, Zeile 185 + 1189 + 1249, Uninitialized struct member: range.dimensions`

In allen drei Fällen wird range.dimensions zwar nicht initialisiert, allerdings wird eine zweite Variable (rangeptr) überprüft, bevor versucht wird range.dimensions wieder freizugeben. In dem Fall, dass rangeptr einen Pointer enthält, wurde auch range.dimensions mittels UA_NumericRange_parse ein Pointer zugewiesen. Daher kommt es hier zu keinem Fehler.

d `./plugins/ua_nodestore_ziptree.c, Zeile 193 + 196, Uninitialized variable: dummy`
In beiden Fällen wird dummy nicht initialisiert, sondern im Anschluss entsprechende struct members gesetzt. In der weiteren Verwendung sind nur diese wichtig, daher kommt es hier zu keinem Fehler.

Neben diesen Fehlern gab CPPCheck noch weitere Meldungen der Kategorien Portability und Warnings aus:

1 `(Portability) Shifting signed 64-bit value by 63 bits is implementation-defined behaviour.`
`./src/ua_types_encoding_binary.c, Zeile 321`
An dieser Stelle wurde ein Algorithmus aus Beej's Network programming guide übernommen. Es sollte überprüft werden, ob anstatt signed auch ein unsigned long long verwendet werden kann.

2 `(Portability) Returning an integer (int/long/etc) in a function with pointer return type is not portable across different platforms and compilers.`
`./build/src_generated/open62541/types_generated_handling.h, Zeile 243 + 274 + 398 + 1049`
In allen vier Fällen wird UA_new verwendet, um Speicher für die Variable zu allokieren. Der so zurückgegebene void Pointer wird in einen Pointer für den entsprechenden Datentyp (UA_INT64*, UA_UINT64*, UA_DateTime*, UA_UtcTime*) konvertiert und zurückgegeben und sollte zu keinem Problem führen.

3 `(Portability) %lu in format string (no. 1) requires 'unsigned long' but the argument type is 'size_t {aka unsigned long}'.`
`./examples/client_historical.c, Zeile 90`
`./tests/server/check_server_historical_data.c, Zeile 325 + 372 + 465 + 468`
Die Länge von size_t ist abhängig von dem System und Compiler (beispielsweise 32bit vs. 64bit) und kann daher zur Compilerwarnungen führen. Es sollte anstatt %lu, %zu für structs verwendet werden. Dies ist allerdings eine neuere Option, falls dies aus Kompatibilitätsgründen nicht möglich ist, kann der auszugebende Wert auf maximale Größe gecastet werden (unsigned long long) und mittels %llu ausgegeben.

4 `(Warning) The obsolete function 'alloca' is called.`
`./src/server/ua_server_discovery.c, Zeile 54`
`./src/pubsub/ua_pubsub_writer.c, Zeile 283 + 284 + 1802 + 1887 + 1928 + 2010 + 2011`
`./src/pubsub/ua_pubsub_ns0.c, Zeile 231 + 531 + 747 + 915`
`./src/server/ua_services_view.c, Zeile 612 + 1119`
`./src/server/ua_services_discovery.c, Zeile 161 + 229`
`./src/server/ua_services_discovery_multicast.c, Zeile 193 + 463 + 500`
`./src/client/ua_client_highlevel.c, Zeile 792`
`./src/client/ua_client_subscriptions.c, Zeile 339 + 545`
`./src/server/ua_subscription.c, Zeile 490`
`./src/server/ua_subscription_datachange.c, Zeile 98`
`./examples/encryption/server_encryption.c, Zeile 45`
`./examples/encryption/client_encryption.c, Zeile 37`
`./examples/access_control_encrypt/client_access_control_encrypt.c, Zeile 38`

Diese Meldungen betreffen die Verwendung des Makros UA_STACK_ARRAY. Aus Compilerkompatibilitätsgründen wird hier noch alloca verwendet. In dem Fall, dass gcc oder clang verwendet wird, werden die mit C99 eingeführten Variable-Length-Arrays (VLA) verwendet. Die Verwendung von alloca oder VLAs könnte auch noch als Experten Build-Option aufgenommen werden, um so eine einfache Möglichkeit der Einstellung bei speziellen Compilern zu bieten.

5 `(Warning) %d in format string (no. X) requires 'int' but the argument type is 'unsigned int.`
`./examples/discovery/client_find_servers.c, Zeile 44 (no. 1)`
`./examples/server_ctt.c, Zeile 222 (no.1) , 286 (no. 1 & 2) + 589 (no. 1)`
`./examples/client.c, Zeile 24(no. 2,3 & 5) + 76 (no. 2)`
`./examples/client_async.c, Zeile 26(no.2) + 36(no.2) + 49(no.2) + 59(no.2)`
`./tests/server/check_server_historical_data.c, Zeile 963 + 970 + 975 + 980 + 1000 + 1005 + 1010 (alle no.1)`
Dies sind Fehler ohne Auswirkungen auf die IT-Sicherheit. In allen Fällen wird ein unsigned int ausgegeben, während als Formatierung ein signed int erwartet wird. Es sollte stattdessen %u verwendet werden.

6 `(Warning) %u in format string (no. 2) requires 'unsigned int' but the argument type is 'signed int'`
`./examples/custom_datatype/client_types_custom.c, Zeile 92`
Dies ist ein Fehler ohne Auswirkungen auf die IT-Sicherheit. Hier wird ein signed int ausgegeben, während ein unsigned int erwartet wird. Es kann stattdessen %i oder %d verwendet werden.

7 `(Warning) Possible null pointer dereference: dataA`
`./tests/client/check_client_historical_data.c, Zeile 207`
Dies ist potenziell möglich, falls dataSize falsch angegeben wird. Dies ist hier allerdings nicht der Fall und kommt daher nicht vor.

8.3 Frama C

FramaC ist ein weiteres statisches Codeanalysetool, welches explizit für C Code entwickelt wurde und open source ist. Es bietet weitere Analysen und benötigt Programme, welche zur statischen Codeanalyse verwendet werden. Für diesen Zweck wurden die Beispielprogramme (examples) und Tests (tests) verwendet. Für diese Programmcodes wurde FramaC inklusive des Moduls eva (evolved value analysis) ausgeführt. Dieses Modul aktiviert weitere Tests, die auf dem Code durchgeführt werden, ohne zusätzliche Annotationen, wie beispielsweise Vor- und Nachbedingungen einzelner Methoden, im Sourcecode vornehmen zu müssen.

```
SRCS=$(frama-c-script list-files | grep -v '../examples\|../tests' | tr -d '\\\n' | cut -d "#" -f 1 | cut -d "=" -f 2)
```

Zur Beschleunigung der Ausführung wurde der gesamte Sourcecode der zu analysierenden Dateien zuerst kompiliert und abgespeichert.

```
frama-c -cpp-extra-args="-std=c99-I/usr/include" -json-compilation-database compile_commands.json $SRCS -save open62541.sav
```

Im Anschluss wurde dies verwendet, um alle Sourcecode-Dateien zu analysieren. Der Aufruf ist für alle diese Dateien stets gleich aufgebaut, im Folgenden für die client.c:

```
frama-c -load open62541.sav -then client.c –eva
```

Das oben beschriebene genutzte Modul Eva hat im open62541 Code der Version 1.1.1. die folgenden 23 Meldungen gefunden, wovon 13 Alarms und die restlichen 10 Warnings für dasselbe Problem sind.

1 `[eva:Alarm] Warning: out of bounds read`

a `./examples/client.c, Zeile 47, assert \valid_read(&(endpointArray + i)->endpointUrl.data)`

Dies ist kein Fehler, da der index nicht zu hoch gezählt wird und auch der endPoint entsprechend gesetzt ist.

b `./examples/server_ctt.c, Zeile 954, assert \valid_read(argv + i)`
Dies ist kein Fehler, hier wurde die Kommandozeile eingelesen und entsprechend mittels argc und argv die Parameter überprüft. Der Index wird korrekt verwendet.

2 `[eva:Alarm] Warning: out of bounds write`

a `./examples/server_loglevel.c, Zeile 52, assert \valid(&config->logger)`
Dies ist kein Fehler, da ein entsprechender Logger bei dem Anlegen einer Konfiguration initialisiert wird und somit zugreifbar ist.

b `./examples/tutorial_client_events.c, Zeile 87, assert \valid(&(selectClauses + 0)->attributeId)`
Dies ist kein Fehler, da selectClauses entsprechend initialisiert und übergeben wurde, somit ist der Zugriff auf attributeId nicht out of bounds.

c `./include/open62541/client.h, Zeile 138, assert \valid(&identityToken->userName)`
Dies ist kein Fehler, da das identityToken entsprechend allokiert und initialisiert wurde, wozu auch userName gehört.

d `./include/open62541/client.h, Zeile 139, assert \valid(&identityToken->password)`
Dies ist kein Fehler, da das identityToken entsprechend allokiert und initialisiert wurde, wozu auch password gehört.

e `[eva:alarm] Warning: signed overflow`
`./examples/server_settimestamp.c, Zeile 66, assert -9223372036854775808 < currentTime - (long long)(1800 * (long long)((long long)(10LL * 1000LL) * 1000LL))`
Dies ist kein Fehler, da es zu keinem negativen Überlauf kommen kann. Dies liegt daran, dass currentTime stets positiv ist und daher die Subtraktion nicht zu einem Überlauf führen kann.

3 `[eva:alarm] function “x”: precondition “y” got status unknown`

a `./examples/client_connect.c, Zeile 39, function: ‘strcmp’, precondition: ‘valid_string_s1’`
Dies ist kein Fehler, hier wird von der Kommandozeile ein Parameter eingelesen, welcher anschließend verglichen wird.

b `./examples/common.h, Zeile 16, function: ‘fopen’, precondition: 'valid_filename'`
Dies ist kein Fehler, da nach fopen überprüft wird, ob der Zugriff erfolgreich war und diese Helperklasse einen allgemeinen Zugriff auf Dateien erlaubt. Ansonsten ließe sich der Zugriff auf mögliche Dateien weiter einschränken.

c `./examples/common.h, Zeile 28, function ‘fread’, precondition: 'valid_ptr_block'`
Dies ist kein Fehler, da zuvor sichergestellt wurde, dass die Datei geöffnet werden konnte und fseek mit legitimen Werten genutzt wird.

d `./examples/server_ctt.c, Zeile 672, function: ‘snprintf_va_4’, precondition: ‘valid_read_string(param0)’`
Dies ist kein Fehler, allerdings ist auffällig, dass eine andere als die initialisierte Länge übergeben wird. Es sollte nochmal geprüft werden, ob dies so beabsichtigt ist.

e `./examples/server_loglevel.c, Zeile 37, function: ‘atoi’, precondition: 'valid_nptr'`
Dies ist kein Fehler, optarg wird von getopt_long verwendet, um übergeben Argumente einzelner Optionen zurückzugeben. Die Verwendung geschieht hier korrekt.

f `./include/open62541/types.h, Zeile 175, function ‘strlen’, precondition: 'valid_string_s'`
Dies ist kein Fehler, falls allerdings die maximale Länge bekannt ist lässt sich auch strnlen nutzen, um sicherzugehen das auch bei einem nicht NULL-terminierten String aufgehört wird zu lesen.

4 `[eva:locals-escaping] locals {name} escaping the scope of a block ./examples/server_ctt.c, Zeile 592 (10 Warnings)`
Dies sind Fehler, welche 10 Warnings erzeugt haben. Das Problem ist, dass Pointer auf die lokale Variable name den Scope von name verlassen. Dies geschieht durch Speicherung des Pointers in object_attr.description (Zeile 590), object_attr.displayname (Zeile 591) und dem hinzufügen über UA_QUALIFIEDNAME in UA_Server_addObjectNode (Zeile 592). Dies kann zu undefiniertem Verhalten führen, da die lokale Variable name nach dem Scope freigegeben wird. Es sollte stattdessen Speicher für name allokiert werden.

8.4 Clang

Clang ist ein Compiler Front End für C und C++ und nutzt dazu das LLVM Framework. Darüber hinaus liefert Clang weitere statische Codeanalysen, welche für die Analyse von open62541 genutzt wurden. Dazu wurde die CMakeLists.txt dahingehend angepasst, dass das Flag -Werror entfernt wurde, um auch dann komplett durchzulaufen, falls Warnings gefunden wurden. Standardmäßig sind schon mehr Flags eingeschaltet als mit -Wall und -Wextra abgedeckt werden. Lediglich die Warnings zu -Wno-static-in-inline, -Wno-overlength-strings und -Wno-unused-parameter werden ausgeschaltet, welche keinen Einfluss auf die Security haben. Mittels dieser Einstellungen wurden auch keine weiteren Warnungen generiert, was nicht verwunderlich ist, da durch die standardmäßige Verwendung von -Werror nicht mehr erfolgreich kompiliert wird, sobald eine Warnung erzeugt wird.

8.5 Manuelle Analyse

Für die manuelle Analyse wurde der Code hinsichtlich sicherheitskritischer Fehler untersucht. Dabei wurden insbesondere Verschlüsselung, Signatur, Integrität, Sessionhandling, Securitypolicies, Accesscontrol, Server- und Clientbeispiele betrachtet. Im Folgenden sind die betrachteten Dateien und Funktionen aufgelistet. Die Ergebnisse werden im Anschluss dargestellt.

- `client/ua_client_connect.c`
signActivateSessionRequest, encryptUserIdentityToken, checkCreateSessionSignature, processERRResponse, processACKResponse, SendHELMessage, processOPNResponseDecoded, sendOPNAsync, renewSecureChannel, responseActivateSession, activateSessionAsync, responseSessionCallback, createSessionAsync, connectSync, connectASync, UA_client_connectSecureChannel, closeSecureChannel
- `client/ua_client.c`
UA_Client_init, UA_Client_newWithConfig, UA_ClientConfig_deleteMembers, UA_Client_deleteMembers, UA_Client_delete, UA_Client_getState, UA_Client_getConfig, notifyClientState, sendSymmetricServiceRequest, processAsyncResponse, processServiceResponse, receiveResponse, receiveResponseAsync, __UA_Client_Service, UA_Client_AsyncService_cancel, UA_Client_AsyncService_removeAll, __UA_Client_AsyncServiceEx, __UA_Client_AsyncService, UA_Client_sendAsyncRequest, asyncServiceTimeoutCheck, backgroundConnectivityCallback, UA_Client_backgroundConnectivity, UA_Client_run_iterate, UA_Client_sendAsyncRequest

- `ua_securechannel.c`
  UA_SecureChannel_init, UA_SecureChannel_setSecurityPolicy, UA_SecureChannel_close, UA_SecureChannel_processHELACK, UA_SecureChannel_sendAsymmetricOPNMessage, sendSymmetricChunk, processSequenceNumberSym, decryptVerifySymmetricChunk
- `ua_types.c`
  fnv32, UA_Guid_random
- `ua_securechannel_crypto.c`
  checkAsymHeader, UA_SecureChannel_generateLocalNonce, UA_SecureChannel_generateLocalKeys, prependHeadersAsym, hideBytesAsym, padChunkAsym, signAndEncryptAsym, calculatePaddingSym, padChunkSym, signChunkSym, encryptChunkSym, setBufPos, decodeChunkPadding, verifySignature, decryptAndVerifyChunk, processSequenceNumberAsym, checkAsymHeader, checkSymHeader, UA_SecurityPolicy_getRemoteAsymEncryptionBufferLengthOverhead, generateRemoteKeys
- `server/ua_session.c`
  UA_Session_attachToSecureChannel, UA_Session_detachFromSecureChannel, UA_Session_generateNonce, UA_Session_addSubscription, UA_Session_deleteSubscription, UA_Session_getSubscriptionById, UA_Session_dequeuePublishReq, UA_Session_queuePublishReq
- `server/ua_services_securechannel.c`
  removeSecureChannel, UA_Server_deleteSecureChannels, UA_Server_cleanupTimedOutSecureChannels, purgeFirstChannelWithoutSession, UA_Server_createSecureChannel, UA_Server_configSecureChannel, UA_SecureChannelManager_open, UA_SecureChannelManager_renew, UA_Server_closeSecureChannel, Service_OpenSecureChannel, Service_CloseSecureChannel
- `server/ua_services_session.c`
  signCreateSessionResponse, UA_Server_createSession, Service_CreateSession, checkSignature, decryptPassword, selectEndpointAndTokenPolicy, Service_ActivateSession, Service_CloseSession
- `server/ua_services_attribute.c`
  getUserWriteMask, getAccessLevel, getUserAccessLevel, getUserExecutable, ReadWithNode, copyAttributeIntoNode
- `plugins/ua_pki_default.c`
  verifyCertificateAllowAll, verifyApplicationURIAllowAll, clearVerifyAllowAll, UA_CertificateVerification_AcceptAll, bstrchr, UA_Bstrstr, fileNamesFromFolder, reloadCertificates, certificateVerification_verify, certificateVerification_verifyApplicationURI, certificateVerification_clear, UA_CertificateVerification_Trustlist, UA_CertificateVerification_CertFolders
- `plugins/securitypolicies/openssl/ua_pki_openssl.c`
  UA_CertContext_sk_Init, UA_CertContext_sk_free, UA_CertContext_Init, UA_CertificateVerification_clear, UA_skTrusted_Cert2X509, UA_skIssuer_Cert2X509, UA_skCrls_Cert2X509, UA_Certificate_Filter_der, UA_Certificate_Filter_crl, UA_BuildFullPath, UA_loadCertFromFile, UA_ReloadCertFromFolder, UA_X509_Store_CTX_Error_To_UAError, UA_CertificateVerification_Verify, UA_VerifyCertificateAllowAll, UA_CertificateVerification_VerifyApplicationURI, UA_CertificateVerification_Trustlist, UA_CertificateVerification_CertFolders
- `plugins/securitypolicies/ua_securitypolicy_none.c`
  verify_none, sign_none, length_none, encrypt_none, decrypt_none, makeThumbprint_none, compareThumbprint_none, generateKey_none, generateNonce_none, newContext_none, deleteContext_none, setContextValue_none, compareCertificate_none, updateCertificateAndPrivateKey_none, policy_clear_none, UA_SecurityPolicy_None
- `plugins/securitypolicies/ua_securitypolicy_basic128rsa15.c`
- `plugins/securitypolicies/ua_securitypolicy_basic256.c`

- `plugins/securitypolicies/ua_securitypolicy_basic256sha256.c`
- `plugins/securitypolicies/securitypolicy_mbedtls_common.c`
  swapBuffers, mbedtls_hmac, mbedtls_generateKey, mbedtls_verifySig_sha1, mbedtls_sign_sha1, mbedtls_thumbprint_sha1, mbedtls_encrypt_rsaOaep, mbedtls_decrypt_rsaOaep
- `plugins/securitypolicies/openssl/securitypolicy_openssl_common.c`
  UA_Openssl_Init, UA_copyCertificate, UA_OpenSSL_RSA_Public_Verify, UA_OpenSSL_RSA_PKCS1_V15_SHA256_Verify, UA_Openssl_X509_GetCertificateThumbprint, UA_Openssl_RSA_Private_Decrypt, UA_Openssl_RSA_Oaep_Decrypt, UA_Openssl_RSA_Public_Encrypt, UA_Openssl_RSA_OAEP_Encrypt, P_SHA256_Ctx_Create, P_SHA256_Hash_Generate, UA_Openssl_Random_Key_PSHA256_Derive, UA_Openssl_RSA_Public_GetKeyLength, UA_Openssl_RSA_Private_GetKeyLength, UA_Openssl_RSA_Private_Sign, UA_Openssl_RSA_PKCS1_V15_SHA256_Sign, UA_OpenSSL_HMAC_SHA256_Verify, UA_OpenSSL_HMAC_SHA256_Sign, UA_OpenSSL_Decrypt, UA_OpenSSL_Encrypt, UA_OpenSSL_AES_256_CBC_Decrypt, UA_OpenSSL_AES_256_CBC_Encrypt, UA_OpenSSL_X509_compare, UA_OpenSSL_RSA_PKCS1_V15_SHA1_Verify, UA_Openssl_RSA_PKCS1_V15_SHA1_Sign, P_SHA1_Ctx_Create, P_SHA1_Hash_Generate, UA_Openssl_Random_Key_PSHA1_Derive, UA_OpenSSL_HMAC_SHA1_Verify, UA_OpenSSL_HMAC_SHA1_Sign, UA_Openssl_RSA_PKCS1_V15_Decrypt, UA_Openssl_RSA_PKCS1_V15_Encrypt, UA_OpenSSL_AES_128_CBC_Decrypt, UA_OpenSSL_AES_128_CBC_Encrypt
- `plugins/securitypolicies/openssl/ua_openssl_basic128rsa15.c`
- `plugins/securitypolicies/openssl/ua_openssl_basic256.c`
- `plugins/securitypolicies/openssl/ua_openssl_basic256sha256.c`
- `plugins/ua_config_default.c`
  UA_Server_new, UA_ServerConfig_addSecurityPolicyNone, UA_ServerConfig_addSecurityPolicyBasic128Rsa15, UA_ServerConfig_addSecurityPolicyBasic256, UA_ServerConfig_addSecurityPolicyBasic256Sha256, UA_ServerConfig_addAllSecurityPolicies, UA_ServerConfig_setDefaultWithSecurityPolicies, UA_ServerConfig_setMinimalCustomBuffer, UA_Client_new, UA_ClientConfig_setDefault, UA_ClientConfig_setDefaultEncryption
- `plugins/ua_accesscontrol_default.c`
  - activateSession_default, clear_default, UA_AccessControl_default
- `examples/access_control/server_access_control.c`
- `examples/server.cpp`
- `examples/server_ctt.c`
- `examples/client.c`
- `include/open62541/util.h`
  UA_constantTimeEqual
- `CMakeLists.txt`

Das Review ergab die folgenden Ergebnisse:

1 Memory Leak

   a `ua_securechannel_crypto.c, Zeile 115`
   Im Fehlerfall wird hier der allokierte Speicher von buf nicht wieder freigegeben.

2 Zertifikatsverifizierung
`plugins/ua_pki_default.c`

a Zeile 297: Die Mindestschlüssellänge wird für alle Zertifikate auf 1024 bits festgelegt. Dies sollte allerdings von der genutzten Policy abhängen.

b Zeile 310-429: Hier wird nach einer fehlgeschlagenen Verifizierung überprüft, ob das Zertifikat in den trusted Zertifikaten enthalten ist. Ist dies der Fall werden die Issuer Zertifikate als trusted angesehen und das Zertifikat erneut verifiziert. Die Verifikation erscheint an dieser Stelle fehlerhaft und der Code ist nur schwer nachvollziehbar. Insbesondere sollte die gesamte Zertifikatskette mittels mbedTLS geprüft werden (Zertifikat -> Intermediate -> Trusted root). Zusätzlich sollte die Überprüfung, ob eine Revocation List zu dem CA Zertifikat existiert, nur an einer Stelle implementiert werden, um doppelten Code zu vermeiden. Es wird empfohlen diese Stellen zu überarbeiten und verständlich zu dokumentieren.

c Zeile 491: Die Überprüfung der angegebenen ApplicationUri des Zertifikats wird zugelassen sobald sie irgendwo im v3_ext Feld enthalten ist. Hier sollte sichergestellt werden, dass die ApplicationUri tatsächlich eigenständig ist und nicht Teil eines längeren Eintrags ist.

3 SecurityPolicies

a Alle implementierten SecurityPolicies kontrollieren nicht die untere und obere Grenze der asymmetrischen Schlüssellängen. So könnten auch zu schwache Zertifikate für eine Policy verwendet werden. Für mbedTLS Policies (ua_securitypolicy_basic128rsa15.c Zeile707, ua_securitypolicy_basic256.c Zeile 657, ua_securitypolicy_basic256sha256.c Zeile 699): Als Personalization Entropy ließe sich auch zusätzlich zu der Konstanten ein Wert aus beispielsweise der Konfiguration ableiten, um so für unterschiedliche Systeme auch unterschiedliche Personalization Entropy zu erhalten.

b plugins/securitypolicies/openssl/ua_openssl_basic256sha256.c, Zeile 259 + 273 + 547
Es wird mittels UA_Assert überprüft, ob die Schlüssellänge 2048 bit entspricht. Die Policy erlaubt in der Spezifikation allerdings auch längere Schlüssellängen.

4 Access Control

a Klartextpasswörter
In Zeile 91 von plugins/ua_accesscontrol_default.c wird das übergebene Passwort direkt mit dem gespeicherten Passwort verglichen. Dies bedeutet, dass die Passwörter im Klartext gespeichert werden. Es sollte für die Passwortablage mindestens ein Salt und Hash verwendet werden, beispielsweise durch Verwendung von Argon2 oder scrypt. Ansonsten besteht die Gefahr, dass dieses Vorgehen durch Entwickler für ihre eigene Implementierung übernommen wird und bei Übernahme des OPC UA Servers ein Angreifer einfach alle Passwörter abgreifen kann.
Des Weiteren kann dann auch auf die Ablage von weiteren Klartextpasswörtern im Code verzichtet werden, da diese dann ebenfalls als Hashwerte abgelegt werden müssen (ua_config_default.c Zeile 112, examples/access_control/server_access_control.c Zeile 51+52)

b Standardpasswörter
In plugins/ua_config_default.c (Zeile 112) werden Standardbenutzer mitsamt Passwort definiert und durch folgende Funktionen als Standard konfiguriert: UA_ServerConfig_setMinimalCustomBuffer (Zeile 454) und UA_ServerConfig_ setDefaultWithSecurityPolicies (Zeile 661). Diese zwei Funktionen werden außerdem durch zwei weitere Funktionen genutzt: UA_ServerConfig_setMinimal und UA_ServerConfig_ setDefault. Dieses Verhalten kann durch einen Entwickler übersehen werden und sollte vermieden werden. Generell sind Standard Benutzer und Passwörter zu vermeiden und spätestens bei Erstbenutzung zu ändern.

c Standardzugriffsrechte
Die Funktionen von plugins/ua_accesscontrol_default.c zur Überprüfung, ob ein bestimmtes Recht vorliegt, sind nur Platzhalter und gewähren volle Rechte. Aus Security Sicht ist es besser für den Fall, dass keine eigenen Rechte vergeben bzw. implementiert werden, standardmäßig nichts zu erlauben.

5 Beispielserver

a Standardpasswörter
Server.cpp und server_ctt.c nutzen beide die Standardkonfiguration, welche die oben erwähnten Standardbenutzer verwenden. Durch die Verwendung dieses Beispiels ist dies einem Entwickler nicht klar und könnte dazu führen, dass diese Standardbenutzer auch in weiteren Implementierungen übernommen werden.

6 Beispielclient

a Hardcoded Passwords
In Zeile 55 von /examples/client.c sind im Sourcecode Zugangsdaten für den Aufbau einer OPC UA Verbindung hinterlegt. Dies ist hier kein Problem, da es sich weder um echte Zugangsdaten noch um eine reale Implementierung handelt. Dennoch kann ein Beispiel dazu beitragen, dass ähnliche Implementierungen durch den Entwickler übernommen werden. Aus diesem Grund wäre es besser, die Zugangsdaten nicht im Code abzulegen, sondern stattdessen beispielsweise den Benutzer diese eingeben zu lassen.

7 Build

a Das Flag Enable_Encryption ist standardmäßig auf OFF gestellt. Dem Ansatz Secure-by-Default folgend wäre es besser dieses standardmäßig auf ON zu stellen.

8.6 Zusammenfassung

Zusammenfassend lässt sich festhalten, dass bei der Analyse keine schwerwiegenden Schwachstellen gefunden wurden und der Code allgemein auf einem sehr hohen Sicherheitsniveau ist. Lediglich ein paar kleinere Stellen wurden gefunden, die aber keine direkte Schwachstelle hinsichtlich Security beinhalten. Allerdings wurden in den Beispielen Zugriffskontrollen mit hartkodierten Klartext Passwörtern verwendet. Dies könnte dazu führen, dass dies so kopiert wird und in den Produktivbetrieb gelangt. Hier wäre es wünschenswert, wenn stattdessen die Passwörter beispielhaft als Hash unter Verwendung eines Salt abgelegt werden, um so für Anwender auch als ein gutes Beispiel für die Passwortablage zu dienen.

Alle gefundenen Punkte wurden dem open62541 Projekt gemeldet. Diese Punkte wurden entsprechend akzeptiert und werden in zukünftigen Versionen von open62541 ausgebessert.

9 Fazit des BSI

Es hat sich im Verlauf der Studie gezeigt, dass eine Aktualisierung der Studie aufgrund der Vielzahl der Veränderungen im OPC-UA-Standard sinnvoll war. Wie in der der Studie 2016 wurden keine konzeptionellen Schwachstellen hinsichtlich der Sicherheitsfunktionen identifiziert. Bei den neuen Handlungsempfehlungen handelt sich um Verbesserungsvorschläge, die dazu beitragen sollen, Implementierungsfehler zu vermeiden und vorhandene Unklarheiten innerhalb des Standards zu schließen. Diese Anmerkungen wurden von der OPC-Foundation positiv aufgenommen und werden in den Standardisierungsgremien diskutiert werden.

Neuere Funktionen, wie PubSub, konnten aufgrund des Umfangs nicht im Rahmen der Analyse der Spezifikationen berücksichtigt werden.

Die Studie zeigt jedoch auch eine mangelnde Implementierung vieler Sicherheitsfunktionen in vorhandenen Produkten auf. So kommt es zu einer wenig zufriedenstellenden Situation. Der Standard definiert Sicherheitsfunktionen, die in der Gesamtschau als sicher zu bezeichnen sind. Die Umsetzung in den Produkten hinkt hier dem Standard jedoch stark hinterher oder setzt diesen nur teilweise um.

Hersteller von Automatisierungskomponenten und Bibliotheken sind daher aufgerufen sich intensiv mit Security relevanten Funktionalitäten auseinander zu setzen, die Funktionalitäten zu implementieren und nicht zuletzt auch ihre Kunden/Endanwender zu informieren und zu schulen. Die Sensibilisierung der Anwender, die (security relevante) Dokumentation, und auch der Auslieferungszustand "secure by default" für UA Produkte, erfordert die Bereitschaft auf der Anwenderseite an entsprechenden Schulungen teilzunehmen.

Bei der untersuchten Implementierung „open62541“ hat sich gezeigt, dass durch entwicklungsbegleitende Test eine hohe Qualität erreicht werden kann.

10 Anhang A: Liste betrachteter Veröffentlichungen

Tabelle 15 Liste der betrachteten Veröffentlichungen

Titel	*Jahr*	*Autoren*	*DOI/Link*
Research on OPC UA security	2010	Renjie, H., Feng, L., & Dongbo, P.	10.1109/ICIEA.2010.5514836
OPC UA information model, data exchange, safety and security for IEC 61131–3	2011	Miyazawa, I., Murakami, M., Matsukuma, T., Fukushima, K., Maruyama, Y., Matsumoto, M., ... & Yamashita, E.	https://ieeexplore.ieee.org/document/6060212
Certificate management in OPC UA applications: An evaluation of different trust models	2012	Fernbach, A., & Kastner, W.	10.1109/ETFA.2012.6489675
An OPC UA based approach for dynamic-configuration of security credentials and integrating a vendor independent digital product memory	2014	Blume, M., Koch, N., Imtiaz, J., Flatt, H., Jasperneite, J., Schleipen, M., ... & Dosch, S.	https://www.iosb.fraunhofer.de/content/dam/iosb/iosbtest/documents/projekte/secure-plug-and-work/Produktblatt_SecurePLUGandWORK.pdf
Research of Integrity and Authentication in OPC UA Communication Using Whirlpool Hash Function	2015	Wu, K., Li, Y., Chen, L., & Wang, Z.	10.3390/app5030446
A Cyber Security Architecture for Microgrid Deployments	2015	Mohan, A., Brainard, G., Khurana, H., & Fischer, S.	10.1007/978-3-319-26567-4_15
The Study of Security Issues for the Industrial Control Systems Communication Protocols	2015	Wanying, Q., Weimin, W., Surong, Z., & Yan, Z.	10.2991/jimet-15.2015.129
Formal Analysis of Security Properties on the OPC-UA SCADA Protocol	2016	Puys, M., Potet, M. L., & Lafourcade, P.	10.1007/978-3-319-45477-1_6
Challenges and research directions for heterogeneous cyber–physical system based on IEC 61850: Vulnerabilities, security requirements, and security architecture	2016	Yoo, H., & Shon, T.	10.1016/j.future.2015.09.026
Sicherheitsanalyse von OPC UA für Industrie 4.0	2016	Wichmann, A.	10.17560/atp.v58i07-08.573
Analysis of the Cyber-Security of industry 4.0 technologies based on RAMI 4.0 and identification of requirements	2016	Flatt, H., Schriegel, S., Jasperneite, J., Trsek, H., & Adamczyk, H.	10.1109/ETFA.2016.7733634
OPC-UA communications integration using a CPPS architecture	2016	Garcia, M. V., Irisarri, E., Pérez, F., Estévez, E., & Marcos, M.	10.1109/ETCM.2016.7750838
Interoperability and Security Challenges of Industry 4.0	2017	Watson, V., Tellabi, A., Sassmannahausen, J., & Lou, X.	10.18420/in2017_100

Titel	*Jahr*	*Autoren*	*DOI/Link*
Introducing remote attestation and hardware-based cryptography to OPC UA	2017	Birnstill, P., Haas, C., Hassler, D., & Beyerer, J.	10.1109/ETFA.2017.8247591
PKI and User Access Rights Management for OPC UA based Applications	2018	Karthikeyan, G., & Heiss, S.	10.1109/ETFA.2018.8502603
A Security Model based Authorization Concept for OPC Unified Architecture	2018	Wallis, K., Merzinger, M., Reich, C., & Schindelhauer, C.	10.1145/3291280.3291799
OPC UA-Integrated Authorization Concept for the Industrial Internet of Things (IIoT)	2018	Gamer, T., Schmitt, J. O., Braun, R., & Schramm, A. M.	10.1007/978-3-030-01174-1_81
Security Challenges in Control Network Protocols: A Survey	2018	Volkova, A., Niedermeier, M., Basmadjian, R., & de Meer, H.	10.1109/COMST.2018.2872114
Secure Framework and Key Agreement Mechanism for OPC-UA in Industrial IoT	2018	Wei, M., Mo, L., Zhuang, Y., & Kim, K.	10.1145/3164541.3164568
A Unified Architecture for Industrial IoT Security Requirements in Open Platform Communications	2019	Hansch, G., Schneider, P., Fischer, K., & Böttinger, K.	10.1109/ETFA.2019.8869524
Secure Granular Interoperability with OPC UA	2019	Watson, V., Sassmannshausen, J., & Waedt, K.	10.18420/inf2019_ws34
Assessing the impact of attacks on OPC-UA applications in the Industry 4.0 era	2019	Polge, J., Robert, J., & Le Traon, Y.	10.1109/CCNC.2019.8651671
Comparative assessment of different OPC UA open–source stacks for embedded systems	2019	Cenedese, A., Frodella, M., Tramarin, F., & Vitturi, S.	10.1109/ETFA.2019.8869187
Formal Security Verification of Industry 4.0 Applications	2019	Nigam, V., & Talcott, C.	10.1109/ETFA.2019.8869428
Open-Source OPC UA Security and Scalability	2020	Mühlbauer, N., Kirdan, E., Pahl, M. O., & Carle, G.	10.1109/ETFA46521.2020.9212091
Assessing the Security of OPC UA Deployments	2020	Roepert, L., Dahlmanns, M., Fink, I. B., Pennekamp, J., & Henze, M.	10.15496/publikation-41813
Improving security in industry 4.0 by extending OPC-UA with usage control	2020	Martinelli, F., Osliak, O., Mori, P., & Saracino, A.	10.1145/3407023.3407077
Easing the Conscience with OPC UA: An Internet-Wide Study on Insecure Deployments	2020	Dahlmanns, M., Lohmöller, J., Fink, I. B., Pennekamp, J., Wehrle, K., & Henze, M.	10.1145/3419394.3423666
OPC UA: Kommunikation und Security: Herausforderungen und Lösungen	2020	Jänicke, L., & Förder, T.	10.17560/atp.v62i3.2457
Research on OPC UA Security Encryption Method	2020	Luo, Z., & Zhang, X.	10.1109/ICIBA50161.2020.9276932
Hybrid OPC UA: Enabling Post-Quantum Security for the Industrial Internet of Things	2020	Paul, S., & Guerin, E.	10.1109/ETFA46521.2020.9212112

Titel	*Jahr*	*Autoren*	*DOI/Link*
Towards Post-Quantum Security for Cyber-Physical Systems: Integrating PQC into Industrial M2M Communication	2020	Paul, S., & Scheible, P.	10.1007/978-3-030-59013-0_15
OTG: A Gateway for Cybersecurity in the Context of OPC UA PubSub Pattern	2020	Liu, J., Huang, J., & Sun, Z.	10.1088/1742-6596/1693/1/012016
Portable Trust Anchor for OPC UA Using Auto-Configuration	2020	Meier, D., Patzer, F., Drexler, M., & Beyerer, J.	10.1109/ETFA46521.2020.9211904
Research on security protection of OPC UA PubSub Protocol	2021	Li, X., Huang, J., & Sun, Z.	10.1088/1742-6596/1738/1/012119
Practical Pitfalls for Security in OPC UA	2021	Erba, A., Müller, A., & Tippenhauer, N. O.	https://arxiv.org/pdf/2104.06051.pdf

11 Anhang B: Sicherheitsrelevante Spezifikationsänderungen seit Version 1.02

Dieser Abschnitt fasst die sicherheitsrelevanten Änderungen der OPC UA Spezifikation seit Version 1.02 zusammen (1.02.47 für Part 12).

11.1 Part 1

11.1.1 Untersuchungsgegenstand

Folgende Dokumente sind seit Version 1.02 von der OPC Foundation veröffentlicht worden (Stand 2020-08-18):

[1] Part 1: Overview and Concepts, Version 1.02 Release, 2012-07-10

[2] Part 1: Overview and Concepts, Version 1.02.4 Release Candidate, 2012-02-07

[3] Part 1: Overview and Concepts, Version 1.03 Release, 2015-10-10

[4] Part 1: Overview and Concepts, Version 1.03.03 Release Candidate, 2015-07-21

[5] Part 1: Overview and Concepts, Version 1.04.04 Release Candidate, 2017-01-16

[6] Part 1: Overview and Concepts, Version 1.04.07 Release Candidate, 2017-07-18

[7] Part 1: Overview and Concepts, Version 1.04 Release 2017-11-22

11.1.2 Vergleich zwischen [1] und [3]

Redaktionelle Korrekturen:

- Software zu Digital (X.509) Certificate
  - „Software Certificates are utilized to identify the Client and Server and the capabilities that they provide.“ [1, 5.4.1.2]
  - “Digital (X.509) Certificates are utilized to identify the Client and Server and the capabilities that they provide.” [3, 5.4.1.2]

Neue, entfernte oder geänderte Sicherheitskonzepte:

- Der Abschnitt Redundancy wurde verschoben. Zusätzlich werden die Konzepte Client, Server und Network Redundancy eingeführt. Außerdem werden die Server Modi transparent und non-transparent eingeführt (siehe [3, 6.4])

11.1.3 Vergleich zwischen [3] und [7]

Redaktionelle Korrekturen:

- Die Definition von Zertifikaten war inkorrekt und wurde angepasst:
  - “digitally signed data structure that describes capabilities of a Client or Server “ [3, 3.2.5]
  - “digitally signed data structure that contains a public key and the identity of a Client or Server” [7, 3.8]

Neue, entfernte oder geänderte Sicherheitskonzepte:

- Definitionen, Konzepte und Abschnitte zu Pub/Sub wurden hinzugefügt.
  - Definitionen

        - Broker [7, 3.7], DataSet [7, 3.13], DataSetMessage [7, 3.13], Message Oriented Middleware [7, 3.20], Network Message [7, 3.23], Publisher [3.32], PubSub [7, 3.33], Subscriber [7, 3.42], Subscription [7, 3.43]
    - Abschnitte
        - Aufnahme von PubSub in die OPC UA-Spezifikation [7, 4.1] [7, 4.3] [7, 5.3] [7, 6.5] [7, 6.6]
- Unter [3, 5.4.1.2] wurde die Beschreibung „Authority-generated software Certificates“ entfernt

Sonstiges:

- JSON wurde als drittes Encoding-Format hinzugefügt [7, 5.3]
- SOAP/http wurde als Transportprotokoll entfernt, WebSockets sind neu hinzugekomment [7, 5.3]
- WS-SecureConversation wurde als Beispiel unter [7,7.3] entfernt

11.2 Part 2

11.2.1 Untersuchungsgegenstand

Folgende Dokumente sind seit Version 1.02 von der OPC Foundation veröffentlicht worden (Stand 2020-08-18):

[1] Part 2: Security Model, Version 1.02 Release, 2012-07-10

[2] Part 2: Security Model, Version 1.03 Release, 2015-11-25

[3] Part 2: Security Model, Version 1.03.05, Release Candidate, 2015-08-20

[4] Part 2: Security Model, Version 1.04.18. Release Candidate, 2018-05-15

[5] Part 2: Security Model, Version 1.04 Release, 2018-08-03

11.2.2 Vergleich von [1] und [2]

Redaktionelle Korrekturen:

- Definition Non-Repudation angepasst [2, 3.1.24]
- Note zur Definition Private Key hinzugefügt [2, 3.1.27]
- Note zur Definition Public Key Infrastructure (PKI) hinzugefügt [2, 3.1.28]

Erweiterte Abschnitte:

- IT-Schutzziel Authentication [1, 5.2.2] zu Application [2, 5.2.2] und User Authentication [2, 5.2.3] erweitert
- Abschnitt Strict Message processing [2, 6.3]
- Abschnitt Random Number Generation [2, 6.3]
- Abschnitt Audit event management [2, 6.11]

Neue Abschnitte:

- Cryptographic Keys [2, 6.8]
- Unsecured Services [2, 7]
- Certificate Management [2, 8] (Erste Inhalte bereits in [1, 6.11] beschrieben)

11.2.3 Vergleich von [2] und [5]

Redaktionelle Korrekturen:

- Begriff “Digital Certificate” zu “Certificate” geändert [5, 3.1.4] [5, 3.1.11] [5, 8] ...
- Notiz zur Definition Application Instance Certificate erweitert [5, 3.1.4]

Neue/Entfernte Definitionen :

- Access Restriction [5, 3.1.1]
- Access Token [5, 3.1.2]
- Authorization Service [5, 3.1.12]
- Claim [5, 3.1.16]
- Identity Provider [5, 3.1.24]
- Permission [5, 3.1.30]
- Resource [5, 3.1.34]
- Role [5, 3.1.36]
- Scope [5, 3.1.37]
- Security Key Service [5, 3.1.38]
- Security Group [5, 3.1.42]
- Definition Digital Certificate entfernt [3, 3.1.16]

Einführung des Pub/Sub-Paradigmas - Erweiterung/Anpassung bereits vorhandener Abschnitte:

- Verallgemeinerung: [5, 4.1] [5, 4.3.2] [5, 4.3.5] [5, 4.3.6] [5, 4.3.7]
- Neuer Abschnitt Rogue Publisher [5, 4.3.11]
- Sicherheitsarchitektur angepasst [5, 4.5.1] [5, 4.5.3]
- Abschnitt Security Policies angepasst [5, 4.6]
- Abschnitt OPC UA security related Services erweitert [5, 4.13]
- PubSub in die Betrachtung in Abschnitt Security reconciliation aufgenommen [5, 5]
  - Denial of Service [5, 5.1.2]
  - Message spoofing [5, 5.1.4]
  - Message replay [5, 5.1.6]
  - Rogue Server or Publisher [5, 5.1.10]
  - Application Authentication [5, 5.2.2]
  - Authorization [5, 5.2.4]
  - Confidentiality [5, 5.2.5]
  - Integrity [5, 5.2.6]

Erweiterungen bzgl. Authentifikation und Autorisation:

- Abschnitt Authorization erweitert [5, 4.2.3]
- Abschnitt User Authentication hinsichtlich Pub/Sub angepasst [5, 4.9]
- Abschnitt Application Authentication erweitert [5, 4.10]
- Abschnitt User Authorization angepasst [5, 4.11]
- Abschnitt Roles hinzugefügt + Abbildung [5, 4.12]

- Abschnitt Application authentication erweitert [5, 5.2.2]
- Abschnitt User Authentication angepasst [5, 5.2.3]
- Abschnitt OAuth2, JWT and User roles [5, 6.12] hinzugefügt

Erweiterte/Angepasste Abschnitte:

- Entfernung von XML/SOAP und WS-SecureConversation Beschreibungstexten und Referenzen [2, 1]
- Message flooding [2, 4.3.2] zu DoS [5, 4.3.2] [5, 5.1.2] erweitert
- IT-Schutzziel bzgl. Message alteration angepasst [5, 4.3.5]
- IT-Schutzziel bzgl. Eavesdropping angepasst [5, 4.3.3]
- Abschnitt Message replay erweitert [5, 4.3.6]
- Abschnitt Rouge Server erweitert [5, 4.3.10]
- IT-Schutzziel bzgl. Compromising user credentials angepasst [5, 4.3.12]
- Einführung des online Profile-Tools in Abschnitt Security Profiles [5, 4.7]
- Tabelle unter [5, 5.1.1] eingefügt
- Abschnitt Message spoofing erweitert [5, 5.1.4]
- Abschnitt Message replay erweitert [5, 5.1.6]
- Abschnitt Confidentiality angepasst [5, 5.2.5]
- Abschnitt Administrative access erweitert [5, 6.7]
- Beschreibung zum Certificate Store erweitert [5, 8.1.4.2]

Neue Abschnitte:

- IT-Schutzziel Non-Repudiation hinzugefügt [5, 4.2.6]
- Repudiation [5, 4.3.13] [5, 5.1.12]
- Security Mode Setting [5, 4.8]
- HTTPs, SSL/TLS & Websockets [5. 6.13]
- Reverse Connect [5, 6.14]

11.3 Part 3

11.3.1 Untersuchungsgegenstand

Folgende Dokumente sind seit Version 1.02 von der OPC Foundation veröffentlicht worden (Stand 2020-08-18)

[1] Part 3: Address Space Model, Version 1.02 Release, 2012-07-10

[2] Part 3: Address Space Model, Version 1.02.19 Release Candidate,

[3] Part 3: Address Space Model, Version 1.03, Release, 2015-07-20

[4] Part 3: Address Space Model, Version 1.03.09. Release Candidate, 2015-03-18

[5] Part 3: Address Space Model, Version 1.04.10, Release Candidate, 2017-02-09

[6] Part 3: Address Space Model, Version 1.04.16, Release Candidate, 2017-07-18

[7] Part 3: Address Space Model, Version 1.04 Release, 2017-11-22

11.3.2 Vergleich von [1] und [3]

Erweiterte Sicherheitskonzepte

- AccessLevel und UserAccessLevel in Tabelle 8 wurden erweitert [3, 5.6.2]
  - StatusWrite, TimestampWrite

11.3.3 Vergleich von [3] und [7]

Redaktionelle Korrekturen:

- Auf die Unzuverlässigkeit der Attribute UserAccess, UserExcecutable und UserWriteMask wird hingewiesen [7, 5.2.8] [7, 5.6.2] [7, 5.7]
- Beschreibung zu SemanticChange Attribut angepasst [7, 5.6.2]

Neue (Sicherheits-)Konzepte:

- Rollenbasierte Zugriffskontrolle eingeführt [7, 4.8] [7, 5.2.9] [7, 5.2.10] [7, 5.2.11]
  - Abschnitt Roles eingeführt [7, 4.8]
  - Optionale Attribute für Rollen zur Base NodeClass hinzugefügt [3, 5.2.1]
    - RolePermission [7, 5.2.9]
    - UserRolePermissions [7, 5.2.10]
    - AccessRestrictions [7,5.2.11]
- Datentyp von WriteMask und UserWriteMask der Base NodeClass geändert [7, 5.2.1] [7, 5.2.7] [7, 5.2.8]
  - Von UInt32 zu AttributeWriteMask
- Optionales Attribut AccessRestrictions zur Base NodeClass hinzugefügt [7, 5.2.1] [7, 5.2.11]
- Attribute AccessLevelEx zur Variable NodeClass hinzugefügt [7, 5.6.2]
- Beschreibung zu AccesLevel und UserAccesLevel Attribut angepasst [7, 5.6.2]
  - StatusWrite und TimeStampWrite [7, 8.57]
  - Typ von Byte zu AccessLevelType angepasst

11.4 Part 4

11.4.1 Untersuchungsgegenstand

Folgende Dokumente sind seit Version 1.02 von der OPC Foundation veröffentlicht worden (Stand 2020-08-18):

[1] Part 4: Services, Version 1.02 Release, 2012-03-18

[2] Part 4: Services, Version 1.02.14 Release Candidate, 2012-02-07

[3] Part 4: Services, Version 1.03, Release, 2015-07-20

[4] Part 4: Services, Version 1.03.09. Release Candidate, 2015-03-18

[5] Part 4: Services, Version 1.04.11, Release Candidate, 2017-02-09

[6] Part 4: Services, Version 1.04.18, Release Candidate, 2017-07-18

[7] Part 4: Services, Version 1.04 Release, 2017-11-22

11.4.2 Vergleich von [1] und [3]

Neue Definitionen:

- Active Server [3, 3.1.1]
- Failed Server [3, 3.1.4]
- Failover [3, 3.1.5]
- Redundancy [3, 3.1.10]
- Redundant Server Set [3, 3.1.11]

Änderungen am Discovery Service Set:

- Hinweis zu Abbildung 9 hinzugefügt: Aufbau des SecureChannels war nicht enthalten [3, 5.4.1]
- Hinweis unter [3, 5.4.1] hinzugefügt: EndpointDescription bezieht sich auf die CreateSession-Response
- Alle möglichen HostNames sollen in das Zertifikat des Servers aufgenommen werden [3, 5.4.1]
- Neue Discovery-Services
  - FindServersOnNetwork [3, 5.4.3]
  - RegisterServer2 [3, 5.4.6]
- FindServers
  - Es dürfen jetzt auch mehrere Einträge pro Server existieren [3, 5.4.2.1]
  - Hinweis zu non-transparent Servern hinzugefügt [3, 5.4.2.1]
  - Als LocalId soll der applicationName der ApplicationDescritption verwendet werden [3, 5.4.2.1]
- GetEndpoints
  - Server darf AIC bei SecurityPolicy None senden [3, 5.4.4.1]
- RegisterServer
  - Hinweis hinzugefügt: Von Discovery Servers zu implementieren [3, 5.4.5.1]
  - Hinweis hinzugefügt: Dienst setzt Client Authentication voraus [3, 5.4.5.1]
  - Auslagerung des Typs RegisteredServer nach Abschnitt 7.29 [3, 5.4.5.2]

Änderungen am SecureChannel Service Set:

- Referenzen zu SOAP und WS-SecureConversation entfernt [3, 5.5.1]
- Beschreibungstext für HTTPS-Verbindungen hinzugefügt [3, 5.5.1]
- OpenSecureChannel
  - Änderung bei der Handhabung von aktualisierten SecurityToken (should to shall) [3, 5.5.2.1]
  - Hinweis hinzugefügt: Sicherheitskonzepte gilt nicht für SecurityPolicy None [3, 5.5.2.1]
  - Text zur Validierung des HostNames hinzugefügt [3, 5.5.2.1]
  - Client darf AIC auch bei Security Policy None senden [3, 5.5.2.2]
  - Text zur Schließung von SecureChannels beim Erreichen des „Limits“ hinzugefügt [3, 5.5.2.1]
  - Text zu Bad_NonceInvalid hinzugefügt [3, 5.5.2.3]

Änderungen am Session Service Set:

- CreateSession

  - Text zu DoS-Attacks hinzugefügt [3, 5.6.2.1]
  - Client darf AIC auch bei Security Policy None senden [3, 5.6.2.2]
  - Server darf AIC auch bei Security Policy None senden [3, 5.6.2.2]
  - Text zu Bad_NonceInvalid hinzugefügt [3, 5.6.2.3]
- ActivateSession
  - Referenzen zu SoftwareCertificates entfernt
  - Fehlender userIdentityToken wird auf Anonymous gemappt [3, 5.6.3.2]
- CloseSession
  - Verhalten beim Schließen der Session vor Aktivierung der Session beschrieben [3, 5.6.4.1]

Sonstiges:

- Bad_SecurityModeInsufficient hinzugefügt [3, 5.10.2.4]
- Text zur Unabhängigkeit zwischen Subscription und Session hinzugefügt [3, 5.13.1.1]
- Text zum AIC angepasst [3, 6.1.2]
- Text und Tabelle zur Zertifikatsvalidierung angepasst [3, 6.1.3]
- Text zu SoftwareCertificates entfernt [3, 6.2]
- Kein AuditEvent für OpenSecureChannel mit renew-requestType, wenn erfolgreich [3, 6.3.5]
- Abschnitt Redundancy überarbeitet [3, 6.4]
  - Neuer Abschnitt Manually Forcing Failover [3, 6.4.5]
- Neuer Abschnitt Durable Subscriptions [3, 6.6]
- Neue Parameter DiscoveryConfiguration [3, 7.9] und MdnsDiscoveryConfiguration [3, 7.9.2]
- Reihenfolge des Parameters SignatureData korrigiert [3, 7.32]
- Text zu UserNameIdentityToken [3, 7.36.3] und IssuedIdentityToken überarbeitet [3, 7.36.5]
  - Hinweis im Fall der unverschlüsselten Übertragung von Passwörtern hinzugefügt [7.36.3]
- Abbildung 1 zu DiscoveryServiceSets korrigiert [3, 4.1]

11.4.3 Vergleich von [3] und [7]

Neue Definitionen:

- DiscoveryEndpoint [3.1.3]

Änderungen am SecureChannel Service Set:

- OpenSecureChannel
  - Hinweis hinzugefügt, dass AIC nicht mehr zwingend erforderlich sind [7, 5.5.2.1]
  - Datentyp von secureChannelId und channelId zu BaseDataType in geändert [7, 5.5.2.2]
  - Länge des client und serverNonce von der SecurityPolicy abhängig [7, 5.5.2.2]
  - Text zu requestedLifetime bzgl. Bedrohungen angepasst [7, 5.5.2.2]
- CloseSecureChannel
  - Text hinzugefügt: Response kann abh. vom Protokoll wegfallen [7, 5.5.3.2]
  - Datentyp von secureChannelId zu BaseDataType geändert

Änderungen am Session Service Set:

- CreateSession
  - Beschreibungstexte zu Parametern angepasst [7, 5.6.2.2]
    - maxResponseMessageSize, ServerEndpoints, serverSignature, maxRequestMessageSize
- ActivateSession
  - Text zum Schutz vor Angriffen auf UserIdentityToken hinzugefügt [7, 5.6.3.1]
    - Text zum Parameter clientSignature hinsichtlich Zertifikatsketten ergänzt [7, 5.6.3.2]

Sonstiges:

- Schritte zur Zertifikatsvalidierung in Tabelle 6 erweitert [7, 6.1.3]
  - Neu: Build Certificate Chain, Security Policy Check
- In Abbildung 22 und 23 Authorization Service mit Identity Provider ersetzt [7, 6.1.5] [7, 6.1.6]
- Text bzgl. User identity Token angepasst [7, 6.1.5]
- Neue Konzepte zu Authorization hinzugefügt
  - Neuer Abschnitt Authorization Services [6.2]
  - UserIdentityToken neu definiert/erheblich erweitert [7, 7.36]
    - IssuedIdentityToken neu definiert [7, 7.36.1]
    - EncryptedSecretFormat vs. Legacy Encrypted Token Secret Format
  - UserTokenPolicy angepasst [7, 7.37]
- Neues Konzept Session-less Serivce invocation eingeführt [7, 6.3]
- Abschnitt ClientRedundancy angepasst [7, 6.6.3]
- Text zum Aufbau neuer SecureChannels hinzugefügt [7, 6.7]
- Text zu ReverseConnect hinzugefügt [7, 6.7]
- Abbildung 38 angepasst [7, 7.31]
- Abschnitt UserIdentityToken parameters angepasst [7, 7.36]
- Klarstellung, dass Discovery-Dienste SecureChannel mit SecurityPolicy None verwenden
  - FindServers, GetEndpoints [7, 5.4.1]
- Beschreibung zu profileUrls in GetEndpoints-Service unter Tabelle 5 angepasst [7, 5.4.4.2]
- Text hinzugefügt: Server kann nun auch logische Verbindungen initiieren [7, 5.5.1]
- An mehreren Stellen Certificate mit Public oder Private Key ersetzt
  - [7, 5.5.2.1] [7, 5.5.2.2]
- Neue StatusCodes

11.5 Part 5

11.5.1 Untersuchungsgegenstand

Folgende Dokumente sind seit Version 1.02 von der OPC Foundation veröffentlicht worden (Stand 2020-08-18):

[1] Part 5: Information Model, Version 1.02 Release, 2015-03-18

[2] Part 5: Information Model, Version 1.02.22 Release Candidate

[3] Part 5: Information Model, Version 1.02.28, Release Candidate

[4] Part 5: Information Model, Version 1.03. Release, 2015-07-20

[5] Part 5: Information Model, Version 1.03.13, Release Candidate, 2015-03-18

[6] Part 5: Information Model, Version 1.04.09, Release Candidate, 2017-02-09

[7] Part 5: Information Model, Version 1.04.15 Release Candidate, 2017-07-18

[8] Part 5: Information Model, Version 1.04 Release, 2017-11-22

11.5.2 Vergleich von [1] und [3]

Redaktionelle Korrekturen

- Klarstellung bei Beschreibung des securityRejectedSessionCount und rejectSessionCount in Tabelle 143 [3, 12.9]
- Beschreibung in Tabelle 149 für authenticationMechanism angepasst [3, 12.12]
- Should zu Shall in AuditEvents
- Anpassungen aufgrund Redundancy
- NonTransparentRedundancyType [3, 6.3.9]
- RedundancySupport [3, 12.5]

11.5.3 Vergleich von [3] und [8]

Rollen hinzugefügt:

- RolePermissions, UserRolePermissions als Common Node Attribute hinzugefügt [8, 5.1]
- RoleSetType unter Tabelle 10 hinzugefügt [8, 6.3.2]
- DefaultRolePermission, DefaultUserRolePermissions und DefaulAccesRestrictions in Tabelle 21 hinzugefügt [8, 6.3.13]

Erweiterungen:

- AccessRestriction als Common Node Attribute hinzugefügt [8, 5.1]
- AccessLevelEx als Common Variable Attribute hinzugefügt [8, 5.3]
- StatusCode in den AuditSecurityEventType aufgenommen [8, 6.4.4]

Sonstiges:

- Common Method Attribute eingeführt [8, 5.5]
- EndointType eingeführt [8, 12.22]
- Annex F User Authentication hinzugefügt

11.6 Part 6

11.6.1 Untersuchungsgegenstand

Folgende Dokumente sind seit Version 1.02 von der OPC Foundation veröffentlicht worden (Stand 2020-08-18):

[1] Part 6: Mappings, Version 1.02 Release, 2012-03-18

[2] Part 6: Mappings, Version 1.02.9 Release Candidate

[3] Part 6: Mappings, Version 1.03. Release, 2015-07-20

[4] Part 6: Mappings, Version 1.03.12, Release Candidate, 2015-03-18

[5] Part 6: Mappings, Version 1.04.17, Release Candidate, 2017-02-09

[6] Part 6: Mappings, Version 1.04.27 Release Candidate, 2017-07-18

[7] Part 6: Mappings, Version 1.04 Release, 2017-11-22

11.6.2 Vergleich von [1] und [3]

Redaktionelle Korrekturen

- Wertebereich der Built-in Datentypen Byte und UInt32 korrigiert [3, 5.1.2]
- Hinweis hinzugefügt, dass OpenSecureChannel-Nachrichten sowohl bei SecurityPolicy SignAndEncrypted als auch bei Sign signiert und verschlüsselt werden [3, 6.7.4]
- Bad_TcpUrlRejected zu Bad_TcpEndpointUrlInvalid geändert[3, 7.1.2] [3, 7.1.5]

Anpassungen:

- Hinweise hinzugefügt, dass die Verschachtelung von Datentypen zu stack overflow errors führen kann. [3, 5.1.5] [3, 5.2.2.12] [3, 5.2.2.15] [3, 5.3.1.13]
- CertificateSignatureAlgorithm-Parameter in Tabelle 22-SecurityPolicy aufgenommen [3, 6.1]
- Beschreibung zum SubjectAltName in Tabelle 23 angepasst [3, 6.2.2]
- WS Secure Conversation ist nun deprecated [3, 6.6] [3, 7.2]
- Default für ReceiverCertificateThumprintLength in Tabelle 27 hinzugefügt, fall kein Zertifikat folgt [3, 6.7.2]
- ExtraPaddingSize und Padding Parameter im Message footer in Tabelle 30 vertauscht [3, 6.7.2]
- Datentyp für den Parameter CreatedAt in der OpenSecureChannel-Response Nachricht angepasst.
- Den Beschreibungstext für SenderCertificates unter [3, 6.7.6] angepasst
- Beschreibung zum XML-Encoding für SOAP über HTTPS angepasst [3, 7.3.2]

11.6.3 Vergleich von [3] und [7]

Redaktionelle Korrekturen

- “inclusive” in die Beschreibung der Built-in Datentypen hinzugefügt [7, 5.1.2]
- SecurityMode Sign zu SignOnly geändert [7, 6.1]
- “v3” zu X.509 hinzugefügt [6, 7.2.1]
- Anpassungen am Beschreibungstext des OPC UA Secure Conversation Mesage header in Tabelle 41 [7, 6.7.2.2]
- MaxCertificateSize mit MaxSenderCertificateSize in Tabelle 41 ersetzt [7, 6.7.2.3]
- Klarstellung zum Parameter ReceiverCertificateThumbprintLength in Tabelle 42 [7, 6.7.2.3]
- Referenz zu Part 2 bzgl. Zufallszahlengeneratoren [7, 6.7.4]
- Klarstellung zu HTTPS-Zertifikaten [7, 7.4.1]
- Text hinzugefügt, dass das Unterdrücken von Validierungsschritten beim Zertifikat zu unsicheren Konfigurationen führen können [7, E.6]

Anpassungen:

- Definition Certificate Digest hinzugefügt [7, 3.1.1]
- Abbildung 1 angepasst [7, 4]
  - TransportProtocols: SOAP/http entfernt, HTTPS und AMQP hinzugefügt
  - SecurityProtocols: WS Secure Conversation entfernt
  - Data Encoding: UA JSON hinzugefügt
- Abschnitt zu OPC UA JSON hinzugefügt [7, 5.4]
- SecureChannelNonceLength zu SecurityPolicy in Tabelle 35 hinzugefügt + Text angepast [7, 6.1]
- Beschreibung zu ServerSoftwareCertificates und ClientSoftwareCertificates entfernt [7, 6.2.1]
- Abschnitt SignedSoftwareCertificate entfernt [7, 6.2.3]
- Abschnitt Certificate Chains hinzugefügt [7, 6.2.3]
- Abschnitt JSON Web Token (JWT) hinzugefügt [7, 6.5.2]
- Abschnitt OAuth2 hinzugefügt [7, 6.5.3]
- SecureChannel wird geschlossen, wenn „Lücke“ in der Sequenznummer auftritt [7, 6.7.2.4]
- SignatureSize in der Berrechungsformel für MaxBodySize aus Floor gezogen [7,6.7.2.5]
- Datentyp vom RequestedLifetime-Attribut des OpenSecureChannel-Dienstes zu UInt32 geändert [7, 6.7.4]
- Beschreibung zum Abschnitt Deriving key ergänzt + neue Tabelle [7, 6.7.5]
- Abschnitt zu abstraktem OPC UA Connection Protocol hinzugefügt [7, 7.1] und OPC UA TCP nach 7.2 verschoben
- Mindestgröße der MessageChunkSize auf 8192 korrigiert [7, 6.7.2]
- Reverse-Connect für alle Transportprotokolle eingeführt
  - RHE in Tabelle 50 des UACP MessageHeaders eingeführt [7, 7.1.2.2]
  - Neuer Abschnitt ReverseHello message [7, 7.1.2.6]
  - Ergänzungen in Abschnitt Establishing a connection [7, 7.1.3]
- Beschreibungstext für den Fall, dass keine Resourcen mehr für den SecureChannel vorhanden sind hinzugefügt [6, 7.1.2.3]
- Konzept Error Recovery entfernt [3, 7.1.6]
- Neuer Abschnitt zu Session-less Services für HTTPS [6, 7.4.2]
- Neuer Abschnitt JSON Encoding [6, 7.4.5]
- Neuer Abschnitt zu neuem Transport Protokoll WebSockets [6, 7.5]
- Default Port für OPC UA HTTPS zu 443 geändert [6, 7.6]

11.7 Part 12

11.7.1 Untersuchungsgegenstand

Folgende Dokumente sind seit Version 1.02.47 von der OPC Foundation veröffentlicht worden (Stand 2020-08-18):

[1] Part 12: Discovery and Global Services, Version 1.03.52 Release Candidate, 2015-03-17

[2] Part 12: Discovery and Global Services, Version 1.03 Release, 2015-07-19

[3] Part 12: Discovery and Global Services, Version 1.04 Release, 2018-02-07

<Die in der vorherigen Spezifikationsanalyse verwendete Version 1.02.47 ist nicht mehr im Archiv der OPC Foundation verfügbar.>

11.7.2 Vergleich von [1] und [2]

CertificateGroups eingeführt

- Definition Certifcate Group hinzugefügt [2, 3.1.2]
- AuthorityType in CertificateGroupType geändert [2, 7.5.9]
- CertificateGroupFolderType eingeführt [2, 7.5.15]
- Informationsmodell zu Pull Cert.-Management angepassst [2, 7.6]
  - CertificateDirectoryType mit DefaultApplicationGroup, DefaultHttpsGroup und DefaultTokenGroup eingeführt [2, 7.6.1] [2, 7.6.2]
  - GetCertificateGroup-Methode eingeführt [2, 7.6.1] [2, 7.6.6]
  - Alle Methoden bzgl. CertificatesGroup angepasst, incl. Result Codes
  - CertificateRequestedAuditEventType angepasst [2, 7.6.9]
- Informationsmodell zu Push cert.-Management angepasst [2, 7.7.1]
  - GetRejectedList-Methode eingeführt
  - CertificateDirectoryType eingeführt
- Sonstiges
  - Flag isOffline in isOnline geändert [2, 4.2.2]
  - Referenz auf Part 2 hinzugefügt [2, 7.1]
  - Beschreibung der AddCertificate-Methode hinsichtlich der Validierung von Zertifikaten ergänzt [2, 7.5.5]
  - RsaBasicApplicationCertificateType in RsaMinApplicationCertificateType geändert [2, 7.5.13]
  - Rsa2048ApplicationCertificateType in RsaSha256ApplicationCertificateType geändert [2, 7.5.14]
  - Hinweis hinzugefügt, dass Aufruf der Methode nur mit verschlüsseltem SecureChannel zu erfolgen hat [2, 7.5.4] [2, 7.5.5]

11.7.3 Vergleich von [2] und [3]

Authorization Services eingeführt

- Definition Network Services hinzugefügt [3, 3.1.15]
- Abschnitt Authorization Services eingeführt [3, 9]

KeyCredential Mgmt. eingeführt

- Definition KeyCredential [3, 3.1.4] und KeyCredentialService eingeführt [3, 3.1.5]
- Abschnitt Key Credential Management eingeführt [3, 8]

Änderungen aufgrund ReverseConnect

- Verallgemeinerung von Server zu Application oder Einbezug von Clients [3, 4.1] [3, 6.1] ...

- GDS Informationsmodell angepasst
  - QueryServers-Methode zu QueryApplications-Methode geändert [3, 4.3.4] [3, 6.3.10] [3, 6.3.11]
  - FindApplicationsByEndpoint-Methode hinzugefügt [3, 6.3.1]
  - FindApplications hinzugefügt [3, 6.3.1]

Sonstiges:

- Registrierung durch RegisterServer2-Methode nun verpflichtend [3, 4.2.2]
- Beschreibungstext zu Schritt 4 vom Discovery Prozesses des GDSs angepasst [3, 6.2.1]
- Abschnitt Domain Names and MulticastSubnets hinzugefügt [3, 6.2.5]
- Beschreibung im Abschnitt Provisioning angepasst [3, 7.4]
- UpdateFrequency-Variable zum TrustListType eingeführt [3, 7.5.2]
- Beschreibung der AddCertificate und RemoveCertificate-Methode bzgl. Zertifikatsketten erweitert [3, 7.5.5] [3, 7.5.6]
- TrustListOutofDateAlarmType eingeführt [3, 7.5.9]
- CertificateGroupType um CertificateExpired und TrustListOutOfDate-Objekt erweitert [3, 7.5.10]
- UserCredentialCertificateType eingeführt [3, 7.5.11] [3, 7.5.14]
- Hinweis zur Änderung des AIC im laufenden Betrieb der ApplyChanges-Methode hinzugefügt [3, 7.7.5]

11.8 Errata

11.8.1 Untersuchungsgegenstand

[1] OPC 10000 - UA Specification Errata (PDF/ZIP), Release 1.04.06, 2020-07-15

11.8.2 Part 3

- AccessRestriction-Attribut um ApplyRestrictionsToBrowse erweitert -> Sehen der Node, auch wenn das Lesen der Werte SignAndEncrypt erfordert.

11.8.3 Part 4

- Hinweis zu Padding des UserIdentityTokens hinzugefügt
  - Adressiert CVE-2018-7559
- Hinweise, dass Client Server Zertifikat auch vor Verschlüsselung des UserIdentityToken validieren soll
  - Adressiert, CVE-2018-12087
- Redaktionelle Korrektur: Tabelle 190: Umbenennung UserIdentityTokenType zu UserTokenType

11.8.4 Part 5

- IdentityMapping für Anwendungen

11.8.5 Part 6

- Hinweis zur maximalen Schlüssellänge bei Issuer Certificates hinzugefügt
- Genannte aber nicht definierte StatusCodes ergänzt
  - Bad_CertificateUnknown

    - OpcUa_BadSequenceNumberInvalid
- Verhalten bei rollover der Sequenznummer definiert (kein CVE vorhanden)
- Redaktionelle Korrektur: Konkrete ProtocolVersion 0 eingeführt

11.8.6 Part 7

- SecurityPolicy – None deaktiviert, falls andere vorhanden

11.8.7 Part 12

- Klarstellung bei AddCertificate-Methode bzgl. Issuer Certificates
- Red. Korrektur: Angleichung des Textes von StartSigningRequest und CreateSigningRequest
- Subscriber als server capability entfernt, Registrierung nicht nötig
- Default Value für CertificateGroupId der GetTrustList-Methode definiert
- Generalisierung von RSA, ECC für KeyCredentialManagement notwendig

1

Literaturverzeichnis

[1] Huang, Renjie; Liu, Feng; Dongbo, Pan. 2010. Research on OPC UA security. In: The 5th IEEE Conference on Industrial Electronics and Applications (ICIEA). Taichung, Taiwan, 15 - 17 June 2010.

[2] Wu, Kehe; Li, Yi; Chen, Long; Wang, Zhuxiao. 2015. Research of Integrity and Authentication in OPC UA Communication Using Whirlpool Hash Function. Applied Science 5 (2015), S. 446-458.

[3] Damm; Gappmeier; Zugfil; Plöb; Fiat; Störkuhl. 2016. Sicherheitsanalyse OPC UA. Bundesamt für Sicherheit in der Informationstechnik. 2016.

[4] Puys, Maxime; Potet, Marie-Laure; Lafourcade, Pascal. 2016. Formal Analysis of Security Properties on the OPC-UA SCADA Protocol. In: Computer Safety, Reliability, and Security

35th International Conference, SAFECOMP 2016. Trondheim, Norway, September 20-23, 2016. ISBN 978-3-319-45477-1

[5] Wallis, Kevin; Merzinger, Marc; Reich, Christoph; Schindelhauer, Christian. 2018. A Security Model based Authorization Concept for OPC Unified Architecture. In: Proceedings of the 10th International Conference on Advances in Information Technology. Association for Computing Machinery, New York, NY, United States, 2018. ISBN 978-1-4503-6568-0

[6] Karthikeyan, Gajasri; Heiss, Stefan. 2018. PKI and User Access Rights Management for OPC UA based Applications. IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE, 2018. ISBN 978-1-5386-7108-5

[7] Watson, Venesa; Sassmannshausen, Jochen; Waedt, Karl. 2019. Secure Granular Interoperability with OPC UA. INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik–Informatik für Gesellschaft (Workshop-Beiträge). Gesellschaft für Informatik eV, 2019. ISBN 978-3-88579-689-3

[8] Meier, David; Patzer, Florian; Drexler, Matthias; Beyerer, Jürgen. 2020. Portable Trust Anchor for OPC UA Using Auto-Configuration. 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE, 2020. ISBN 978-1-7281-8956-7

[9] Fraunhofer FKIE. 2021. Blackbox OPC UA Fuzzing. [https://github.com/fkie-cad/blackbox-opcua-fuzzing] GitHub : s.n., 2021.

[10] Pereyda, Joshua. 2018. boofuzz. GitHub. [Online] 15. 6 2018. https://github.com/jtpereyda/boofuzz.

[11] Google. 2016. oss-fuzz. [https://github.com/google/oss-fuzz] GitHub : s.n., 2016.

[12] clusterfuzz. 2020. Fuzzing build failure. [https://bugs.chromium.org/p/oss-fuzz/issues/detail?id=25070&q=open62541] 2020.

[13] clusterfuzz. 2021. Gefundene open62541 Bugs. [https://bugs.chromium.org/p/oss-fuzz/issues/list?q=open62541] 2021.

[14] open62541. 2020. open62541 commit bb8ce4b. [https://github.com/open62541/open62541/commit/bb8ce4be345b23ab84606e91f05be06d446cfb6e] 2020.

[15] MTG AG. 2018. Certification Path Validation Test Tool. [https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Kryptografie/Certification-Path-Validation-Tool/certification-path-validation-test-tool_dvl.html] GitHub : s.n., 2018.

[16] Dahlmanns, Markus, et al. 2020. Easing the Conscience with OPC UA: An Internet-Wide Study on Insecure Deployments. In Proceedings of the ACM Internet Measurement Conference (IMC '20). Association for Computing Machinery, New York, NY, USA, 101–110.

[17] Erba, Alessandro; Müller, Anne; Tippenhauer, Nils Ole. 2021. Practical Pitfalls for Security in OPC UA [https://arxiv.org/abs/2104.06051]

[18] open62541. 2021. Open Source Implementierung des OPC UA Protokolls [https://open62541.org/]