Federal Office
for Information Security

OPC UA Security Analysis

01/06/2022

Revisions

Table 1 Revision history

Version	*Date*	*Description*
1.0	01/02/2022	Publication
1.1	01/04/2022	Correction of defective references
1.2	01/06/2022	Correction of defective references and formatting

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

Acknowledgements

The BSI would like to thank the authors of this study:

M.Sc. Johannes vom Dorp

Group Manager *Applied System Analysis* at Fraunhofer FKIE

M. Sc. Sven Merschjohann

Research Associate at Fraunhofer IEM

M. Sc. David Meier

Research Associate at Fraunhofer IOSB

M. Sc. Florian Patzer

Research Associate at Fraunhofer IOSB

M. Sc. Markus Karch

Research Associate at Fraunhofer IOSB

Dr.-Ing. Christian Haas

Group Manager *Industrial Cybersecurity* at Fraunhofer IOSB

Content

1 Introduction

The Open Platform Communication Unified Architecture (OPC UA) is an open communication standard enabling the communication between arbitrary industrial machines. From individual sensors to complete productions lines plants, they can be represented in a server-side information model and controlled by a client application. OPC UA is a vendor independent standard and a key technology for Industry 4.0 applications. The client-server architecture of OPC UA provides, compared to other industrial communication protocols, built-in security mechanisms to ensure the authenticated, integrity-protected and encrypted communication, as well as mechanisms for the authorised access of information in the OPC UA address pace for applications and users. The specified security mechanisms are sufficient to ensure a high level of security. This was already examined and confirmed in a study conducted on behalf of the BSI for OPC UA version 1.02 in 2016. Since the study of 2016, there have been major changes both in the OPC UA specification and in the ANSI C implementation provided by the OPC Foundation. In the meantime, OPC UA version 1.04 has been adopted and the OPC Foundation's ANSI C implementation is now only available in a legacy version. Simultaneously, more and more products and applications with OPC UA support have been developed and deployed.

In context of this publication, an update of the OPC UA security analysis from 2016 was conducted, based on OPC UA version 1.04 and the open source C implementation open62541. The methodology of the 2016 analysis was mostly maintained and was limited to the Server/Client use case. In addition, a user survey was carried out to determine the security of OPC UA products and applications in real world environments, and to help identify the problems and challenges that the developers face when implementing the OPC UA specification.

The following results were developed during this analysis:

- **Overview of relevant publications with respect to the IT security of OPC UA**: Based on an internet research, 36 publications were examined with regard to their impact on the security properties of OPC UA. Of these publications, the relevant 9 publications were further grouped by the respective areas they address or criticize. Five of these publications directly addressed problematic aspects of the specification that were further examined.
- **Status of editorial remarks from 2016**: The implementation state of original recommendation from the 2016 analysis was examined. Most of the recommendations were found to be completely or at least partially addressed. Several of the recommendations were reevaluated and new remarks were provided to the OPC Foundation.
- **New editorial remarks and recommendations based on OPC UA version 1.04**: In total, 192 editorial remarks and recommendations were identified and discussed with the security working group of the OPC Foundation. The result of these discussion was acknowledged and 171 tickets were created in the Foundation's tracking system "Mantis".
- **Analysis of the protection mechanism on the parameter level**: The results from the 2016 analysis in regards to the server/client communication with created session are still applicable to version 1.04 of OPC UA.
- **Survey results about the implementation of OPC UA security features**: As part of this analysis, a market survey was conducted in order to get insights into the practical usage of OPC UA while concentrating on security relevant aspects. For this purpose, a questionnaire with over 100 question was created and distributed among manufacturers of certified products as well as other contacts via a public link. Afterwards, the 138 collected answers were evaluated. Several recommendations and insights could be gained from this survey. Overall, the implementation state of security features in real products has shown that products still do not meet the expectations in all security relevant areas.
- **Dynamic analysis of the OPC UA protocol**: The security of OPC UA in protocol version 1.04 was reviewed, mostly based on the open62541 library, using three methods: two different fuzzing campaigns

were conducted, one black box fuzzing targeting the protocol on network level and the other white box fuzzing directly on the open62541 implementation. The third was a test of the certificate path validation. All three methods provided newly discovered issues, most prominently the white box fuzzing found a reproducible bug in open62541 that was reported and fixed during the time of the study.

- **Static analysis of open62541**: For the analysis of open62541, automated tools as well as a manual code analysis for the security critical parts of the implementation were employed. The automated code analysis tools used were Cppcheck, FramaC and Clang. To summarize the analysis, no major vulnerabilities were found and the code is in general at a very high security level. Nevertheless, all findings were reported to the open62541 project. They were accordingly accepted and will be fixed in future versions of open62541.

2 Subject of the analysis

The Platform Industrie 4.0 has chosen OPC UA as one of the first protocols when implementing the Industry 4.0 future strategy. The Client/Server architecture of OPC UA provides, compared to other industrial communication protocols, built-in security mechanisms to ensure the authenticated, integrity-protected and encrypted communication, as well as mechanisms for the authorised access of information in the OPC UA address pace for applications and/or users. The specified security mechanisms are basically sufficient to ensure a high level of security. This was already examined and confirmed in a study conducted on behalf of the BSI for OPC UA version 1.02 in 2016 [3]. In addition to the specification and threat analysis, the reference implementation offered by the OPC UA Foundation in ANSI C was analysed via static and dynamic code analysis.

Since the study of 2016, there have been major changes both in the OPC UA specification and in the implementations provided by the OPC Foundation. OPC UA is now specified in version 1.04 and one of the most fundamental changes is the new PubSub architecture. Other parts of the standard, covering certificate management for example, have also undergone major changes. The ANSI C implementation (the implementation analysed in the 2016 study) is now only available as a legacy version. In addition, more products are appearing with (certified) OPC UA support, although it is not always clear which security functions have been implemented w.r.t. Security Policies and User Identity Token.

Considering this situation, an update and extension of the 2016 study is necessary. Therefore, considering version 1.04 of the OPC UA specification, the goal of this document is to update the 2016 study.

For the update of the 2016 study, all cybersecurity-relevant changes made to the OPC UA specification from version 1.02 to 1.04 were identified first. Afterwards, the new version of the specification was analysed w.r.t vulnerabilities and improvements. It was decided together with the BSI that PubSub (Part 14) will not be considered as part of the update, as the major differences between the two communication paradigms, Client/Server and PubSub, would not ensure comparability of the results. Furthermore, at the time of updating the study, no complete (or certified) implementations of PubSub were available. PubSub and the required infrastructure should therefore be analysed in a separate analysis in the future.

As part of the update of the study, an open source implementation of the OPC UA protocol is investigated via static and dynamic code analysis. In this context, open62541 [18] was selected as the open source implementation. For once, it was investigated whether parts of the vulnerabilities identified in 2016 could also be found in the new implementation, additionally what other inconsistencies occurred during the analysis. As far as possible, the identified vulnerabilities are implemented in proof-of-concept exploits.

In addition, a market survey is carried out to identify which security mechanisms have been implemented in the individual products and what difficulties the manufacturers encountered. The results are to be documented in recommendations for manufacturers and integrators.

3 Overview of relevant publications with respect to the IT security of OPC UA

Since the publication of the 2016 security analysis [3], additional publications have become available that are relevant to the security properties of OPC UA. This chapter provides an overview of the publications that were considered relevant for this analysis. A list of all considered publications can be found in Annex A.

3.1 Approach

Based on an internet research, 36 publications were examined with regard to their impact on the security properties of OPC UA. Of these publications, the relevant 9 publications were further grouped by the respective areas they address or criticize. The results are presented in the following section.

3.2 Relevant Publications

In the following, the relevant publications are presented that criticize different areas of the OPC UA specification in regards to security. Table 2 presents an overview of these areas.

Table 2 Overview of criticized areas and respective publications

Criticized area	*Publication*
Mapping of application security methods on the security model	[1]
Usage of SHA-1 in security policies (has since been fixed already)	[2], [3]
OpenSecureChannel and CreateSession enable Man-in-the-Middle attacks	[4]
Extensibility and Separation-of-Concerns authentication and authorization scheme	[5]
Usage of RBAC	[5], [7]
Trust establishment and management	[6], [8]
Deployed and active OPC UA application that are inadequately configured and accessible via public internet	[16]
Unsecure trust list configuration in proprietary OPC UA applications and libraries	[17]

The authors of [1] criticize the possibility of mapping application security methods to the security model of OPC UA. As this work is from 2010, assessing its relevance for the current version of the specification is not possible without further investigation. The publication was added for completeness.

In [2] and [3], the usage of SHA-1 in certain security policies was criticized. As SHA-1 has been removed from the specification in the meantime, these remarks are not relevant anymore.

The possibility of Man-in-the-Middle attacks on the SecureChannel establishment, that enables replay attacks and intercepts channel keys was presented in [4]. The publication recommends adding key-wrapping for nonces to counter this attack method. However, the specification demands in Part 6 that AsymmetricKeyWrapAlgorithm is not to be used for OPC UA Secure Conversation (UASC).

In [4], the possibility of a MitM attack on the OpenSecureChannel service enabling replay attacks is presented, even when using the SignAndEncrypt security mode. The authors recommend adding key wrapping for the nonces as a countermeasure. However, Part 6 of the OPC UA specifiation states the AsymmetricKeyWrapAlgorithm is not to be used for OPC UA Secure Conversation (UASC). The authors of [4] also describe an attack on the CreateSession service that can also be prevented by using key wrapping. The underlying issue seems to be not yet considered by the OPC UA specification. While the authors raise some interesting concerns, their results seem to be at least partially based on false assumptions. For example, the claim that nonces are transmitted unsecured in the Sign security mode does not match the current definition in the specification (see OPC UA, Part 4, 5.5.2.1). Nonetheless, a more in-depth investigation of the claimed issues found in the paper should be carried out.

In [5], the author discovered that the addition of an additional security attribute was not possible because of the already completely used 32bit masks of NodeClassType, WriteMask and UserWriteMask. They also deemed the usage of RBAC as unsuitable, as this would not be compatible with the dynamic requirements of modern use cases. This was also seen negatively in [7], while addressing an inconsistency in the specification where detailed authorization rules are recommended, but at the same time not supported by the authorization concept. Additionally, the authors in [5] also criticize the missing separation between user authorization and authentication, as well as the information and authorization model. The criticized points should be discussed in the future by the standardization body.

The authors of [6] and [8] criticize the missing specification of secure trust establishment and management. While the OPC UA specification recommends the employment of a Public Key Infrastructure (PKI), the concrete integration of the PKI is not described. This makes it unclear how server and clients should be provisioned with certificates, as well as how the integration of PKI services should be implemented in the OPC UA architecture. For the latter, the specification part 12 designates the Global Discovery Server (GDS) to fulfill certain PKI services. For this, practically and available implementations are missing.

Newer publications criticize the poor configuration of publicly accessible OPC UA applications [16] as well as trust lists in proprietary OPC UA applications and libraries [17]. While not directly addressing the security in the OPC UA specification, this creates the impression of the security of state of the art. Still, the area of security is neglected in operating OPC UA systems.

3.3 Summary of results

In this analysis, 36 publications were examined. The predominant part of these concentrate on the application of OPC UA for various use cases with regards to security functionality, but they do not contribute any direct improvements of the OPC UA specification. Nine publications have been further considered based on their criticism of the specification or its current application. Five of these directly address problematic aspects of the specification that should be considered further. Also, based on this information, recommendations have been developed as presented in section 4.3.

4 Editorial remarks, security relevant changes and recommended changes to the specification

The 2016 security analysis suggested various editorial corrections and recommendations based on several parts of the specification. This chapter starts with a current overview of these corrections and recommendations by investigating how they were addressed by the OPC Foundation.

As the 2016 analysis used version 1.02 of the OPC UA specification, the section proceeds with an overview of all security relevant changes between version 1.02 and 1.04, the current OPC UA version during this analysis.

Based on this overview, the chapter concludes with new editorial remarks and compiled recommendations.

4.1 Status of editorial remarks

The 2016 analysis gave over 60 editorial remarks and recommendation in regards to the OPC UA specification. The following presents the results of an analysis of the status of these recommendations, including the feedback of the OPC Foundation.

4.1.1 Approach

The editorial remarks and recommendations of the 2016 analysis were examined by conducting a manual review of each recommendation and its implementation status. This review was based on Version 1.04 of the specification, including corresponding errata as of 18th August 2020. Afterwards, all unaddressed recommendations were again discussed with the security working group of the OPC Foundation.

4.1.2 Overview of the Implementation Status of Editorial Recommendations

Table 3 gives an overview of the number of recommendations based on their implementation status. The status is categorized into three types. Completely addressed recommendations include recommendations that became obsolete, for example, because of the removal of a functionality or because they are not applicable to the current version of the specification. Partially addressed recommendations comprise areas of the specification that were altered but do not completely cover the intent of the original recommendation. The remaining recommendations have not been addressed at all.

Table 3 Status overview of the editorial recommendations

Completely addressed	*Partially addressed*	*Not addressed*
44	10	6

Table 4 gives an overview of the recommendations that were not addressed since the 2016 analysis, as well as an updated assessment of the initial recommendation, based on the current version 1.04 of the OPC UA specification, as well as the answer to the recommendation by the OPC Foundation, as given in 2017[1]. During the course of this analysis, the OPC Foundation has identified further required action for some of these recommendations. As a result, new entries in the Foundation's tracking system "Mantis" were created (Mantis-IDs #6248, #6250, #6251, #6252, #6253). Respective recommendations are highlighted in grey in Table 4.

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

Table 4 Overview of unaddressed recommendations

Document	*Chapter*	*Recommendation for 1.02*	*Answer from OPC Foundation*	*Finding for 1.04*
Part 4	5.5.2.2	"channelId" should be replaced by "secureChannelId"	The XML mapping was removed in V1.03.	The answer of the OPC Foundation is incomprehensible and seems to be a copy&paste error. Nevertheless, the "channelId" field is part of the locally defined ChannelSecurityToken. Therefore, a renaming is not considered necessary.
Part 4	5.6.2.2	It should be specified in detail what "omitted", "empty" and "null" means.	Replaced 'omitted' with 'null' in Part 4 v1.04	Correction was not implemented, text still includes "empty".
Part 6	D.3	The data type for ValidationOptions should be changed to byte or UInt32, as no negative values are permissible.	ValidationOptions is a mask with a finite set of bits used. Negative values are excluded since there are only 6 possible bits.	Recommendation has not been implemented and can still lead to avoidable security bugs.
Part 12	7.6.4	Addition of minimum requirements or recommendations for the quality of the password	Part 2 provides this level of information.	The specification is still not defining any password requirements. But it is not uncommon for a communication standard to omit specifying such a requirement.
General		It is recommended that securityPolicyUri 'None' is prohibited in the securityMode 'Sign' and 'SignAndEncrypt' and that this validation of the configuration is also added as conformity test.	It should be obvious that "None" implies that neither signing nor encryption are implemented. Certification can check for invalid EndpointDescriptions.	The recommendation from 2016 was not implemented and is still valid. The described combination of policies and modes is only prevented in two services by the means of conformity tests.

Similar to Table 4, Table 5 shows all partially addressed recommendations and updated findings.

Table 5 Overview of partially addressed recommendations

Document	*Chapter*	*Recommendation for 1.02*	*Answer from OPC Foundation*	*Finding for 1.04*
Part 2	4.2	The definitions of IT security terms do not correspond to the definitions of internationally recognised standards such as ISO 27000.	We need more specifics on the terms, which are problematic.	Definitions have partially been extended, but are still not based on a particular security standard.
Part 2	4.3.2	Please add "client flooding"	The response size is limited by the client with a configured max. The client automatically closes connections if they are exceeded. This is also true if extra responses are returned.	Description has been edited, but “client flooding” is still not explicitly addressed.
Part 4	5.4.3.1	"CreateSecureChannel" occurs several times in figures, tables or text passages. Replace by "OpenSecureChannel”	Will change text as suggested in Part 2.	Addressed in this section, but incorrect term introduced at different place (Fig. 36).
Part 4	5.5.2.2	Lower and/or upper limits are missing for the requestedLifetime parameter. Please recommend (ranges of) values.	Will add a reference to the discussion in Part 2 that explains the factors that affect the choice of value.	Security remark was added, but still missing recommended limit values.
Part 4	5.6.2.2	Recommendations for a lower limit or a default value for maxResponsesMessageSize.	Part 4 v1.04 now refers to Part 6 for protocol-specific minimum or default values.	Reference to Part 6 added in regard to recommended values, but Part 6 does not include such values.
Part 6	5.1.6	A reasonable maximum permissible recursion depth should be specified, e.g. 10.	-	Still no maximum depth specified, but recommendation added for a depth of 100 and corresponding error definition.

Document	*Chapter*	*Recommendation for 1.02*	*Answer from OPC Foundation*	*Finding for 1.04*
Part 6	6.7.2	Change the data type of SecurityPolicyUriLength and ReceiverCertificateThumbprintLength to byte or UInt32. Change data type of SenderCertificateLength to UInt32.	Added text indicating that negative values are invalid in Part 6 v1.04.	Original recommendation for converting to ByteStrings was not implemented, mainly because of compatibility concerns. But problem was still addressed by adjustments of the definitions.
Part 7	5.5	A category "documentation – security best practices/recommended settings" could be added.	Part 2 already provides this level of information.	Various recommendations and requirements have been added at several places. However, the specification still lacks explicit requirements for security related configurations or best practices.
Part 7	6.5.124-126	Why is a PKI required for these profiles? Can self-signed certificates not be used with trust lists?	-	Affected SecurityPolicies are deprecated and no longer described in the specification. However, unexplained remarks in regard to self-signed certificates can still be found in the Basic256Sha256 definition.
Part 12	F.2	How are rogue servers identified in the provisioning state?	-	More mentions of „rogue server” have been added, but still no further details on their identification.
General		The configuration of an OPC UA application includes a number of parameters which determine lower or upper limits, e.g. for buffer sizes or number of messages. In most cases, however, default value information or a recommendation regarding the range of reasonable values is missing. It would make the work of developers and administrators significantly easier if a list of such default values would be available in the specification.	-	In some passages new limits have been added, but the specification is still missing a compiled list of recommended values.

4.2 Cybersecurity-related changes

In order to update the OPC UA study, all cybersecurity-relevant specification changes had to be identified since version 1.02. First, the procedure for identifying the cybersecurity-related changes is presented, followed by a presentation of the most important updates.

4.2.1 Approach

On 18. August 2020, all available OPC UA specification parts since version 1.02 were downloaded from the OPC Foundation archive. Each of these specification documents have a tabular overview of the general specification changes (after the table of contents) compared to the previous versions. First, these tables were reviewed and all identified cybersecurity-related changes were collected. In the second step, a manual review of all OPC UA parts was carried out. For each OPC UA specification part, release version 1.02 was compared with release version 1.03 and all identified security-relevant changes were collected. Subsequently, the release version 1.03 was compared with the current release version 1.04. Annex B summarises the collected security-relevant changes in detail.

4.2.2 Overview of the major cybersecurity-related changes

Since version 1.02, several new concepts have been introduced and existing concepts have been adapted. Table 6 summarizes the most significant security-related specification changes since version 1.02 and provides an overview.

Table 6 Overview of cybersecurity-related changes

New and modified concepts	***Description***
Roles	OPC UA provides a standard approach for implementing role-based access control with version 1.04. Permissions are assigned to roles. Corresponding specification modifications can be found in parts 2, 3, 5 and 12.
PubSub	OPC UA introduces the Publish - Subscribe communication architecture in Part 14. Therefore, the security architecture in Part 2 was adapted. The PubSub communication model provides confidentiality and integrity. *Security Key Servers* (SKS) are distributing symmetric keys. PubSub can be deployed in environments with or without a broker. An analysis of the PubSub architecture did not take place in this study.
Discovery Services	Two new discovery services were added: *FindServersOnNetwork* and *RegisterServer2*. The *FindServersOnNetwork* service is especially interesting because it can be used without any security mechanisms.
Certificate validation steps	The certificate validation steps described in Part 4 have been extended (Table 106, v.1.04). Furthermore, a section on certificate chains has been added to Part 6.
UserIdentityToken	IssuedIdentityTokens used by the ActivateSession service to authenticate the user are no longer limited to WS-SecurityTokens (e.g. Kerberos), but now allow the use of e.g. JSON Web Tokens.
Session-less Services	Session-less Services are introduced in version 1.04. Authentication can still take place using the *authenticationToken* in the *requestHeader*.
Reverse Connect	Part 6 defines a reverse connect mechanism where the Server is able to initiate the logical connection.
TransportProtocols	WebSockets were introduced as a new transport protocol in Part 6. In addition, SOAP/HTTP has been marked as "deprecated".

New and modified concepts	***Description***
Global Discovery Server (GDS)	The KeyCredential Manager and the Authorization Service were added to the GDS in Part 12. The *KeyCredential* manager allows the management and distribution of *KeyCredentials, which OPC UA Applications* use to access *Authorization Services* and *Brokers.* *The Authorization Services* provide *Access Tokens* to *Clients* that may use them to access resources. Furthermore, the concept of *CertificateGroups* was introduced and the information model of the *CertificateManagers* was adapted.
Profiles und Conformance Units	Part 7 has been restructured. This analysis does not include a detailed list of the security-relevant changes for Part 7 because the specification available as a PDF was last generated on 22.11.2017 and differs significantly from the version available in the reporting tool (https://profiles.opcfoundation.org/v104/Reporting/). However, it can be summarized that for each new security-relevant feature/concept, a corresponding profile and/or conformance unit has been defined.
Elliptic-curve cryptography (ECC)	On 18.12.2020, the OPC Foundation published an amendment to support ECC. Due to the fact that this change was published after the cut-off date defined for this study, it could not be considered. However, as it is a significant change, it should at least be mentioned here.

4.3 Editorial analysis

The editorial analysis was carried out again, due to the numerous security-relevant changes in version 1.04. Once again, different aspects were revealed which could lead to comprehension and/or implementation problems (e.g. simple typing errors, ambiguous wording, inconsistencies etc.) or be even relevant to IT security.

4.3.1 Approach

Based on the identified cybersecurity-relevant changes since version 1.02, the editorial analysis was narrowed down to Parts 2, 3, 4, 5, 6 and 12. First, all security-related changes identified since version 1.02 were colour-coded in the PDF documents. Thus, the focus could be placed on the newly added text passages. Afterwards, these documents were reviewed manually by different members of the Fraunhofer IOSB research team. Then, the editorial comments and recommendations identified in this process were discussed with the security working group of the OPC Foundation.

4.3.2 Overview of the editorial analysis

Table 7 summarises the aspects found: A total of 192 editorial comments and recommendations were identified and discussed with the security working group of the OPC Foundation. The outcome of these discussions was that 171 of the total 192 editorial comments and recommendations reported were confirmed. Corresponding "Mantis" entries were created to make the handling of the comments and recommendations more comprehensible.

The findings of each OPC UA Part can be requested via the mail address ics-sec@bsi.bund.de. Each document contains a reference to the text passage, table or figure, the reported editorial comment and, in some cases, recommendations. If a comment or recommendation was confirmed by the OPC UA Security Group, a corresponding link to the "Mantis" entry is provided.

Table 7 Amount of editorial comments and recommendations reported and confirmed

Document	*Reported comments/recommendations*	*Confirmed comments/recommendations*
Part 2	46	35
Part 3	7	5
Part 4	47	44
Part 5	5	5
Part 6	16	14
Part 12	71	68
Total	**192**	**171**

4.3.3 Summary of the findings

The main findings for the identified editorial comments and recommendations can be grouped into the following four categories:

4.3.3.1 Lack of alignment between specification parts when introducing new features or requirements

As of 18.08.2020, the OPC UA standard consists of 17 specification parts (companion specs, errata and amendments not included). During the analysis, it was noticed that additions to the standard were often not aligned across the specification parts. This also applies to features relevant to IT security.

Examples:

- Session-less Services are introduced in version 1.04. Several parts of the specification do not take this new mechanism into account and state several times that the user can only be authenticated by establishing a session [Part 2, ID 33] [Part 2, ID 34] [Part 3, ID 2] [Part 4, ID 10] [Part 6, ID 4]. However, authentication can also take place via the authenticationToken in the requestHeader for Session-less Services.
- Part 12 introduces the recommendation, that a client should ignore the top-level domain "local" in the URL provided by a Local Discovery Server with Multicast Extension (LDS-ME) while checking the server certificate. However, such a note is missing in Parts 4 and 6 [Part 12, ID 16].
- The management and distribution of certificates and trust lists in OPC UA is realized via certificate groups, which are introduced in Part 12. Unfortunately, part 2 does not mention this concept [Part 2, ID 45]. Moreover, the concept is not included in the XML Schema for security settings [Part 6, ID 16].

4.3.3.2 Lack of description

For some concepts defined in OPC UA, a security assessment is not possible because some concepts are barely specified. In the discussions with the IT Security Working Group of the OPC Foundation Security Group, it has been noticed that this mainly applies to concepts that have not yet been implemented in practice.

Examples:

- The concept of a gateway server is barely specified in Part 4. Since the information about a gateway server is contained in the response of the GetEndpoint service, which is executed without any security mechanism, a security assessment must be carried out. Unfortunately, such an assessment is not possible due to the lack of information in the specification [Part4, ID 8] [Part 4, ID 19]
- The KeyCredential Manager and the Authorization Service were added to the GDS in Part 12. The numerous missing descriptions lead to the assumption that the concepts have not been implemented yet [Part 12, ID 62] [Part 12, ID 63] [Part 12, ID 67] [Part 12, ID 68] [Part 12, ID 70].

- When updating an ApplicationInstanceCertificate, the standard only describes how an OPC UA server has to behave. The description for an OPC UA Client is missing [Part 4, ID 31].
- OPC UA defines multiple steps for the verification of information provided by a discovery endpoint via an established session. However, the procedure for the new session-less service calls is not described [Part 2, ID 33] [Part 2, ID 34].

4.3.3.3 Inaccurate definitions and wording

In several places, inaccurate definitions and wording have been identified, which can lead to comprehension and/or implementation problems.

Examples:

- Section 6.1.2 of Part 4 incorrectly states that the public key is always issued by a CA.
- The "SecurityAdmin" role defined in Part 3 has to be used for security-relevant settings. However, what security-relevant settings are is not defined.
- A reference to Part 4 is provided by Part 12 for the validation process while updating the ApplicationInstanceCertificate via the CertificateManager. It is not specified how to deal with the errors described in Part 4 that can be suppressed and whether the same settings as in Part 4 should be used [Part 12, ID 39] [Part 12, ID 40] [Part 12, ID 41].

4.3.3.4 Complexity of the OPC UA standard

Cybersecurity-relevant concepts are defined in several parts throughout the specification and could therefore lead to comprehension and/or implementation problems. Thus, one could assume from OPC UA Part 2 with the title "Security Model" that this specification part provides an overview of all security-relevant concepts in OPC UA. Unfortunately, this is not the case. An overview of all cybersecurity-relevant concepts in OPC UA should be obtained by reading at least Parts 2, 3, 4, 5, 6, 7, 12 and 14.

Examples:

- Part 2 illustrates a general OPC UA network example in Figure 1. This example does not contain a CertificateManager, KeyCredential Service, Authorisation Service or Security Key Server (SKS). At least one network example with all security-relevant components defined in the OPC UA standard is recommended and expected in Part 2 with the title "Security Model".

The complexity of the OPC UA Standard could be reduced by providing a reference implementation. However, after enquiring with the OPC Foundation, no certified reference implementation is currently available.

5 Analysis of the protection mechanism at the parameter level

In the 2016 study, an analysis of the protection mechanisms was performed. This section is an update of that analysis. First, the procedure is described, whereby the change log for the document of the 2016 study "Detailed analysis of the protection mechanisms of the OPC UA specification at parameter level.xls" is also provided. Afterwards, the analysis table is presented and discussed.

5.1 Procedure and change log for the analysis table

In the 2016 study, the analysis of the protection mechanisms only considers the Client/Server architecture with the three service sets: Discovery, SecureChannel and Session. This scope has been retained for the update performed in this chapter. Based on the cybersecurity-related changes identified in section 4.2, two new discovery services (FindServersOnNetwork and RegisterServer2) have been added since the 2016 study. In addition, the new reverse connect mechanism was introduced, where a Server is able to initiate the logical connection. In a first step, the document "Detailed analysis of the protection mechanisms of the OPC UA specification at parameter level.xls" was expanded to include the new services and mechanisms. Afterwards, the document was then reviewed and updated w.r.t. outdated terms, references and default values. All changes made are summarized in Table 8.

In addition to the document "Detailed analysis of the protection mechanisms of the OPC UA specification at parameter level.xls", some text passages from the final report of 2016 also had to be updated. Since only minor adjustments had to be made, it was decided that the text sections from the final report of 2016 would be copied into this document and that changed or newly added text passages would be highlighted with red font colour.

Table 8 Change log for "Detailed analysis of the protection mechanisms of the OPC UA specification at parameter level.xls"

ID	*Message/Service*	*Description*
1	ReverseHello	Added *ReverseHello*-Message (RHE).
2	FindServersOnNetwork, RegisterServer2	Added these two new discovery services.
3	Hello, Acknowledge	*MessageChunkSize* shall be at least 8 192 Bytes.
4	OpenSecureChannel	The calculated numbers for this service are incorrect. The document from 2016 includes the Part 4 and Part 6 definition for this service. Numbers were assigned and accumulated for both definitions. Thus, numbers were wrongly accumulated twice. This error has been corrected.
5	OpenSecureChannel, CloseSecureChannel	For these two services, the message must now consist of exactly one single/final chunk.
6	OpenSecureChannel	ReceiverCertificateThumbprintLength no longer needs to be 20 Bytes in SecurityMode None, can also take the values 0 or -1.
7	OpenSecureChannel	Changed SecureChannelId to BaseDataType.
8	CreateSession	Recommendation for the ServerEndpoints [] parameter of type EndpointDescription has been adapted.
9	CreateSession	Request can now contain the AIC for *securityPolicyUri None.* However, the certificate shall be ignored by the server.
10	ActivateSession	clientSoftwareCertificates is marked as “Reserved for future “.
11	ActivateSession	New default value: Null or empty user token shall always be interpreted as anonymous.

ID	*Message/Service*	*Description*
12	ActivateSession	Changed *omitted* to *null* for userTokenSignature parameter
13	All Services - Footer	The order of the fields in the OPC UA Secure Conversation Message footer has been changed: PaddingSize, Padding, ExtraPaddingSize, Signature.
14	-	The abstract OPC UA Connection Protocol was introduced. Since the handshake implemented by OPC UA TCP can also be applied to other transport protocols (e.g. WebSockets), the abstract OPC UA Connection Protocol (UACP) was introduced. The table was adapted accordingly.
15	-	The SecurityPolicyUris that are no longer up to date have been removed.
16	-	The outdated table and page references in the document have been removed.

5.2 Explanation of the analysis table

OPC UA seems to provide mechanisms as protection against all threat types introduced in Part 2. This was tested up to the parameter level on the basis of a detailed analysis. Due to their extent, the results of this analysis of the OPC UA security based on the parameters contained in the exchanged messages can be requested via the Mail address ics-sec@bsi.bund.de.

The document is structured as follows:

Which protection OPC UA offers crucially depends on the securityMode parameter. Therefore, the results are presented in three tables for the sake of clarity: Using the tabs, the results for securityMode 'None', 'Sign' and 'SignAndEncrypt' can be displayed. The fourth tab summarises the results of all modes and is explained in more detail below.

The first three tables have the same structure: In the form of a tree structure that can be opened and closed on the left side, the lines represent how the messages are structured in UACP (OPC UA Connection Protocol) and UASC (UA Secure Conversation). The messages which are crucial for the communication and thus play an important role in IT security were chosen, i.e. finding the communication partners (Discovery service) and establishing the connection (SecureChannel and Session): In UACP, there are four possible message types (HEL, ACK, ERR and REV). In UASC, the messages for the SecureChannel, Session and the Discovery services were taken into account.

The columns represent two types of information: The columns on the left side (A to L) are used to describe the message parameters, whereas the columns on the right side (M to X) show the results regarding the evaluation of the security.

Below, a detailed breakdown of the column contents can be found:

- The columns A to F represent the different levels of the protocols, message parts and parameters.
- In column G, the data types of the parameters are specified and column H shows the corresponding default values if available.
- By using colours, the columns I and J highlight which parts of the respective messages are signed or encrypted:
- Column I: grey = unsignedblue = signed
- Column J: grey = unencrypted green = encrypted
- Moreover, inside the surfaces it is shown with which key the operation was carried out.
- The columns K and L contain explanations for a better understanding of the parameters.
- In the columns M to W, the threats from Part 2 are listed.

The results of the analysis are entered in the table as follows: If a parameter supports the protection against a specific threat, '1' is entered in the respective line (parameter) and column (threat). Example: The SecureChannelId contributes to the protection against the re-use of messages. Therefore, '1' has been entered into cell Q53.

In summary, the entries for the individual threats on the message level are added up. If this result is a number greater than 0, this means that this message type provides protection against this threat. This addition of the entries per threat was also continued at the service and protocol level. The results at the service level were transferred to the last tab in the table.

The interpretation as to whether a parameter increases the protection against a threat is defined in relatively broad terms: In particular, this does not mean that one parameter alone is sufficient as protection against the respective threat, but only that it contributes to such protection. Furthermore, the fact that several parameters contribute to the protection against a threat does not automatically mean that the threat is thus fended off completely.

5.3 Detailed explanation of the results of the analysis table

Adding the numbers sorted according to the securityMode at the service level results in table 9.

It is important to stress that the numbers were only used as an aid for the interpretation. Only one thing, however is clear: if the number 0 is entered for one threat for the three services in one securityMode, there can be no protection. However, a quantitative evaluation is not possible: Higher scores do not necessarily refer to better protection.

The findings from the 2016 study on the session-based Client/Server communication for version 1.04 are still up to date. Only the OPC UA Connection Protocol introduced in version 1.04 requires the text from the 2016 study to be adapted. The detailed explanation of the results is listed below:

5.3.1 OPC UA Connection Protocol

Since the parameter securityMode is part of UASC, it does not have any impact on the underlying layer OPC UA Connection Protocol (UACP).

Denial of service: UACP provides only minimum protection against flooding attacks.

Malformed message: Defining upper limits for buffer and message sizes and the maximum number of partial messages also provides protection against incorrect messages in a very restricted manner, since a message of an impermissible size, for example, is rejected at an early stage.

Server profiling: Since the number of different OPC UA SDKs available worldwide is relatively small, it is not excluded that the information gained on the UACP level alone could be sufficient to create a clear server profile.

5.3.2 SecurityMode None

Denial of service: The precautions against flooding attacks are the same for the parameters as for the two other modes, but with different impacts: On the one hand, it is more difficult for an attacker to force the victim to carry out complex computer operations, since there is no signing nor encryption. On the other, an attacker has more options of overloading the memory and filling the hard disk by writing due to the fact that there is no application authentication.

Eavesdropping: Since, with the exception of secret-based userIdentityTokens, there is no encryption in the securityMode None, the security objective 'confidentiality' is not complied with.

Message spoofing, message alteration. securityMode None does not provide any protection at all against the creation and alteration of messages, such as in the event of a man-in-the-middle attack, due to the lack of application authentication.

Message replay and session hijacking: Adequate mechanisms, such as verifying the sequenceNumber provide protection against messages being sent again and the takeover of sessions. This requires, however, that the attacker does not eavesdrop additionally.

Table 9 Effectiveness of the OPC UA measure

Security-Mode	*Layer or Service*	*Denial of Service*	*Eaves-dropping*	*Message Spoofing*	*Message Alteration*	*Message Replay*	*Malformed Messages*	*Server Profiling*	*Session Hijacking*	*Rogue Server*	*Compromising User credentials*	*Repudiation*
		low prot.	no prot.	no prot.	no prot.	no prot.	low prot.	no prot.	no prot.	no prot.	no prot.	no prot.
	UACP	8	0	0	0	0	8	0	0	0	0	0
None		restricted	no prot.	no prot.	no prot.	no prot.	low prot.	low prot.	effective	low prot.	effective	restricted
	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
	Discovery	20	0	4	4	35	9	8	30	6	0	6
Sign		restricted	no prot.	effective	effective	effective	effective	restric-ted	effective	effective	effective	effective
	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
	Discovery	21	0	5	5	36	9	20	31	7	1	10
Sig-nAndEn-crypt		restricted	effective	effective	effective	effective	effective	re-stricted	effective	effective	effective	effective
	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
	Discovery	21	13	5	5	36	9	20	43	7	13	10

Restricted protection (prot.) - the possibilities of an attacker are restricted, but this type of attack is not prevented.
Effective protection - attacks of this type require cryptographic attacks.

malformed message: Due to the missing application authentication, an attacker has many options of creating incorrect messages and having them evaluated by the victim. User authentication alone, if absolutely necessary, prevents a session from being established.

server profiling: The protection against the creation of a server profile is limited to the fact that user authentication, if absolutely necessary, prevents a session from being established.

rogue server: Application authentication for the RegisterServer and the RegisterServer2 Discovery service prevents an attacker from registering their unauthorised server with Discovery servers and thus from being found via FindServers of clients in this manner. In Part 12, however, it is also possible that a rogue server introduces itself via mDNS.

Due to the lack of application authentication, a Client is then no longer able to distinguish an untrusted server from a trusted server.

compromising user credentials: If the securityPolicyUri or a userTokenPolicy is set as required by the specification, i.e. not 'None', the security of a secret-based userIdentityToken is ensured for user authentication.

repudiation: Thus, non-repudiation regarding user authentication is also provided. Non-repudiation regarding the application cannot be complied with, since there is no such authentication.

5.3.3 SecurityMode Sign and SignAndEncrypt:

eavesdropping: As expected, the two modes basically differ in the protection of confidentiality: In the securityMode Sign, the OpenSecureChannel requests and responses themselves are encrypted, but the further exchange of information is performed in unencrypted form. In particular, the contents of all requests and responses which are exchanged within an existing session can be read by an eavesdropping attacker, because they are sent in plaintext. Only secret-based userIdentityTokens which are sent for the authentication of the users in case of activateSession requests are encrypted and thus protected.

With SignAndEncrypt, however, all messages are encrypted for SecureChannel and the Session. Only for the Discovery service, all applications, irrespective of the securityMode set otherwise, must support FindServers, FindServersOnNetwork and GetEndpoints requests and responses even without security, i.e. unencrypted and unsigned.

denial of service: Certain parameters restrict the possibilities of an attacker in the case of flooding attacks. Forced application authentication in particular makes it considerably more difficult to perform a flooding attack on the session level and/or makes it impossible without knowing further information such as private keys. Especially due to the measures involving complex decryption and signature verification steps, the attacker has nevertheless the possibility of also putting a considerably stronger load on the processor, as is the case with securityMode None, in addition to a high network load. This means that UASC only offers inadequate protection against flooding attacks.

This vulnerability, however, must be qualified, since an attacker would probably achieve comparable results with significantly less effort by flooding on the IP or TCP protocol level.

server profiling: Due to the required application authentication, an attacker has less possibilities of gaining information via a SecureChannel, whereas failed authentication might also disclose information about the server. With the securityModes, an attacker cannot establish sessions, but FindServers, FindServersOnNetwork and GetEndpoints are also possible without authentication.

The two modes offer adequate protection against all other threats, with slightly better protection when SignAndEncrypt is chosen. If confidentiality does not play a role, securityMode Sign is probably sufficient.

5.4 Conclusion

The conclusions from the 2016 study on the session-based Client/Server communication for version 1.04 are still up to date and listed below:

securityMode None provides little to no protection against IT security attacks and should be avoided in all cases. If confidentiality does not play a role, securityMode Sign offers adequate protection against the threats examined. If confidential data is exchanged, securityMode SignAndEncrypt is required.

The threats ´*denial of service*´ and ´*server profiling*´ can only be reduced using OPC UA protection mechanisms, but not fended off completely. Accordingly, additional measures outside the OPC UA infrastructure must contribute to the protection against these threats.
Moreover, the analysis has shown that all threats except for ´*denial of service*´ and ´*server profiling*´ can be counteracted adequately with the securityMode SignAndEncrypt. Thus, the security objectives are complied with except for 'availability'.

6 Survey

As part of this analysis, a market survey was conducted in order to get insights into the practical usage of OPC UA while concentrating on security relevant aspects.

6.1 Approach

One aim of the survey was to find out which security mechanisms are implemented by the products using OPC UA and what challenges the manufacturers faced when implementing them. To acquire this information, a questionnaire was designed with the intention to address a target group as broad as possible, in order to get a realistic picture of the practical application of OPC UA. The questionnaire was also designed to be accessible to participants with varying knowledge of OPC UA. Participant were asked to answer the question in relation to a specific OPC UA product.

The invitation for questionnaire was distributed through two separate channels:

- Manufacturers of certified OPC UA products were directly contacted to participate in the survey. Respective products were gathered from the OPC Foundation's website.
- All other manufacturers were invited to participate mainly through the newsletter of the OPC Foundation via a public link.

The questionnaire was available for participation from September 2020 to March 2021.

6.2 Structure and Execution

For this survey, a collection containing over 100 questions was built based on general best practices of the IT security domain, as well as conformance units of the OPC UA specification.

From these questions, a questionnaire was implemented for the survey and conducted using the online platform SurveyMonkey. For this, the questions were distributed into 16 pages. Individual pages could be shown or hidden based on a defined rule set. This enabled the questionnaire to adapt to participant and their individual referenced products.

The questions were assigned to the following eight different categories:

Table 10 Categories of the Survey

Category	*Description*
Basic Questions	Basic question about the OPC UA product, its type and certifications. Also includes optional question about the participants.
Product Features	Features of the product, for example, the utilized SDK or supported OPC UA versions.
Basic Security Questions	Basic question about security properties of the product, for example, secure default configurations or supported authentication methods.
Expert Security Questions	Further question about the product's security properties, for example, followed security concepts and employed certificate checks.
Product Development	Questions about the secure development of the products, for example, product security requirements and employed methods for secure software development.
Discovery Server	Questions about the employment of discovery services.
Global Discovery Server	Questions about the support of the GDS and its features, for example, the certificate management.
PubSub	Questions about the support of OPC UA's PubSub and supported security level.

6.3 Evaluation of the results

In total, 138 participants took part in the study, with 129 of these using the public link. Nine participants used the individual links sent to the manufacturers of the certified products.

The following paragraphs will summarize the major findings of each question category. Because of the adaptability of the questionnaire to the knowledge of the participant and the possibility of skipping over individual questions, the following number of average answers were collected for each category:

Table 11 Number of questions per category

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 Basic Questions

On average, 131 participants answered in this category.

These basic questions can give insight into the types of participants and their respective products. The covered products were majorly for the OPC UA server (77,54%), but also clients (67,39%), as shown in Figure 1. 16,67% of participant's products were described as an SDK (in total 23 participants). Multiple selections were possible for this question.

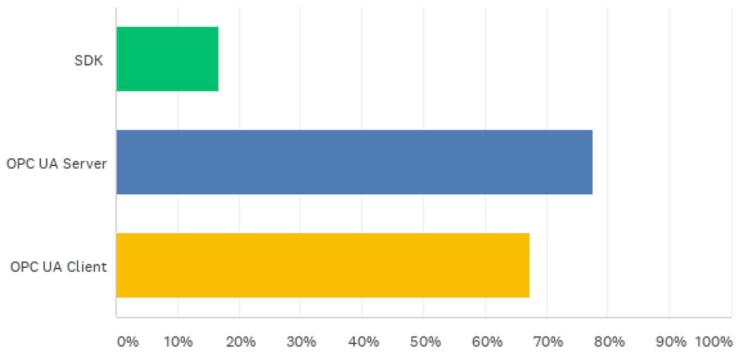

Figure 1 OPC UA product types

Amongst the participants, approx. 55% identified as developers of OPC UA products, 30% as product managers and 5% as product support members. Over 60% of participants stated that their product was not officially certified. Together, the certified products covered all available certification profiles, even outdated ones. The majority of certified products implemented the Standard 2017 UA Server Profile (16,38%) or Standard 2017 Client Profile (12,93 %), respectively.

6.3.2 Product Features

On average, 79 participants answered in this category.

The employed stacks, as stated by the participants, show a broad distribution, as shown in Figure 2

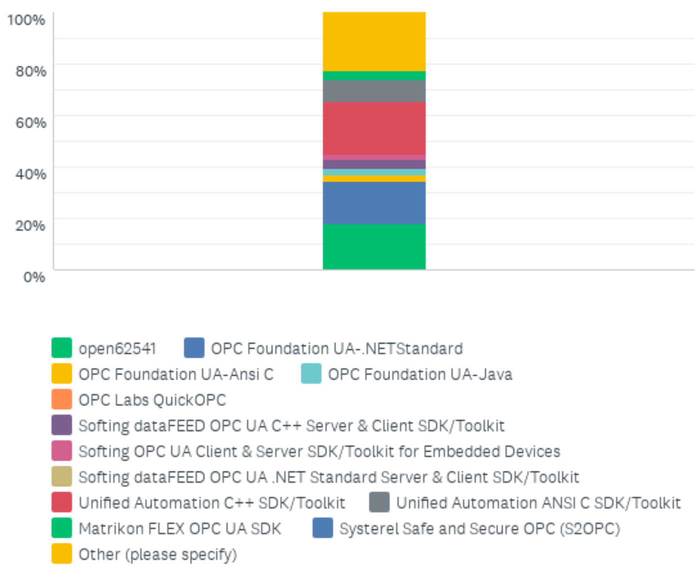

Figure 2 Employed OPC UA stacks

In Figure 3 the supported OPC UA versions are shown. While the current version 1.04 has the majority with 72,84%, Version 1.0 is still supported by 25% of the participants. The majority of participants are planning to support the current OPC UA version in the future, with 30% not planning to update the supported version.

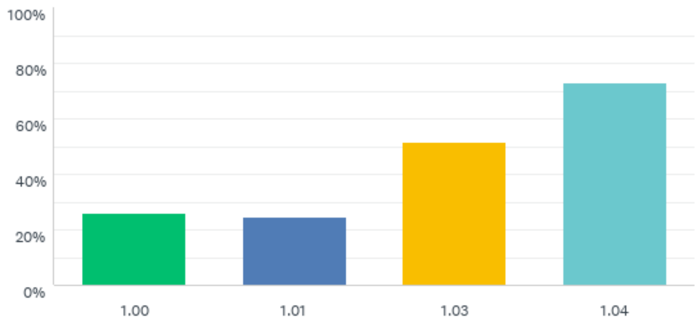

Figure 3 Supported OPC UA versions

Just over one third of participants (33,8%) stated that their product supports at least some aspects of the Global Discovery Server. Another question referred to the importance of different product requirements on the development. Here, for most participants the feature set (41.18%) was the most important requirement. Security was also seen as an important requirement, but one of secondary importance.

80% of participants stated that at least internal security experts are involved in the development process of their product. Only one participant stated that also external experts are consulted during development.

6.3.3 Basic Security Questions

On average, 73 participants answered in this category.

Over 70% of participant answered that there are instructions available on how to securely configure their products. At the same time, as depicted in Figure 4, 50% of participants stated that security functions are not enabled in the default configuration of their product or that there was not a defined default configuration.

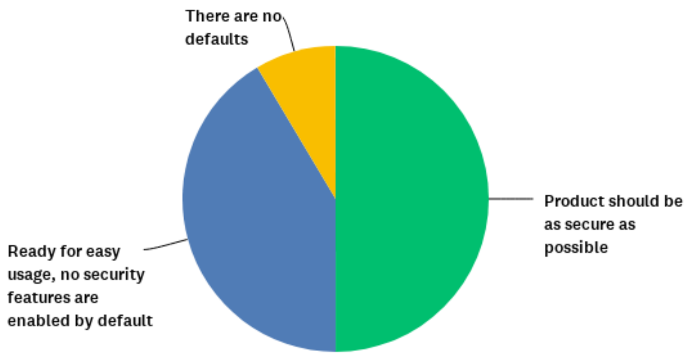

Figure 4 Default configurations and security settings

6.3.4 Expert Security Questions

On average, 21 participants answered in this category.

The most common transport protocol for OPC UA was „OPC UA TCP", which is supported by all products of all answering participants. OPC UA over https (15%) and WSS (2%) are not as widespread.

Part 7 of the OPC UA specification defines several security relevant recommendations, for example, handling of timeouts or random number generation. The attention to these recommendations is quite varying (14% - 63%), with no recommendation being followed by all products, as shown in Figure 5.

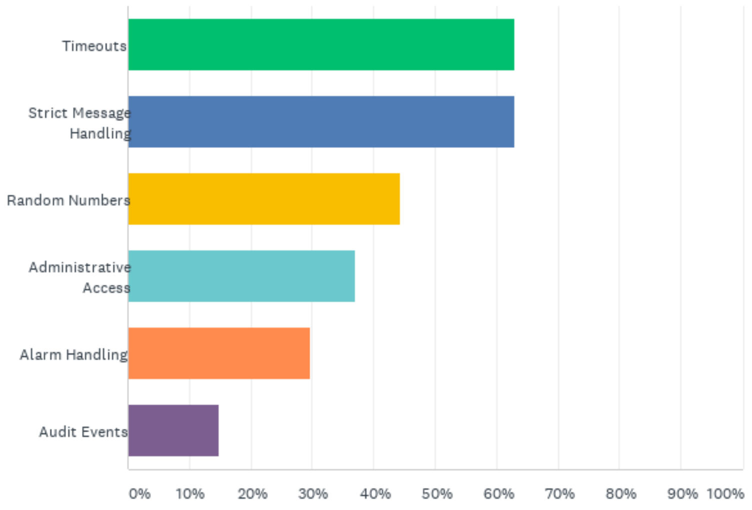

Figure 5 Implemented security recommendations

Default roles defined by the specification are only supported by 12% of products, with only 25% of products supporting the OPC UA role concept at all.

As shown in Figure 6, not all certificate checks demanded by the specification are implemented or enforced by the included products, with supported individual checks varying from 36% (validation of the URI) to 86% (validity date checks). The answer rate of basic checks like certificate structure and signature, only 70% and 80% respectively, could possibly indicate a misunderstanding or lack of knowledge of the product functionality.

Most products (96%) do not seem to support certificate specific alert messages, which are defined by the specification.

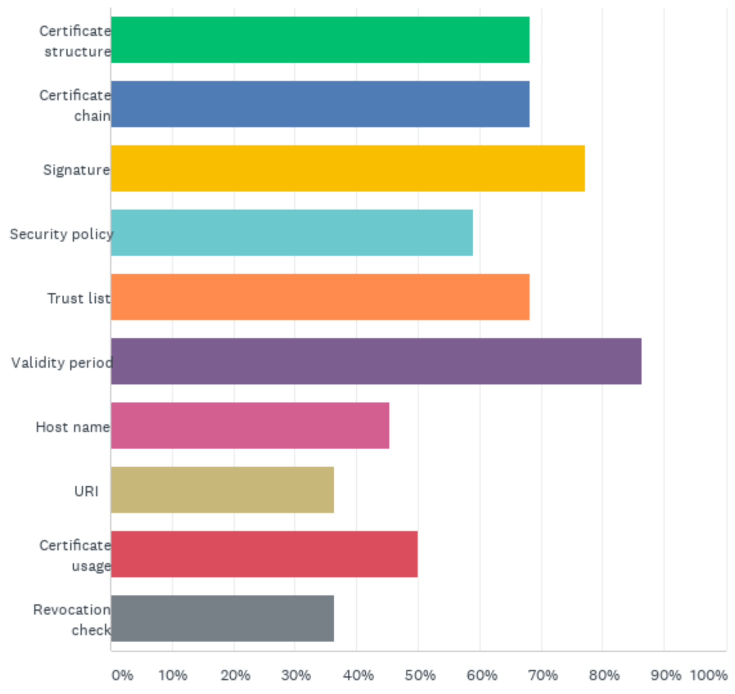

Figure 6 Supported certificate checks

Additionally, 11 participants stated that their product does not support replacement of the application certificate, for example, in order to install a PKI generated certificate.

6.3.5 Product Development

On average, 26 participants answered in this category.

In this category, the employed methods for secure software development were queried. Many participants already employ such methods, for example, security focused code reviews (47%), static code analysis (60%) or risk analysis (52%), as shown in Figure 7. Amongst the participants, 60% state that penetration tests are carried out against the final product.

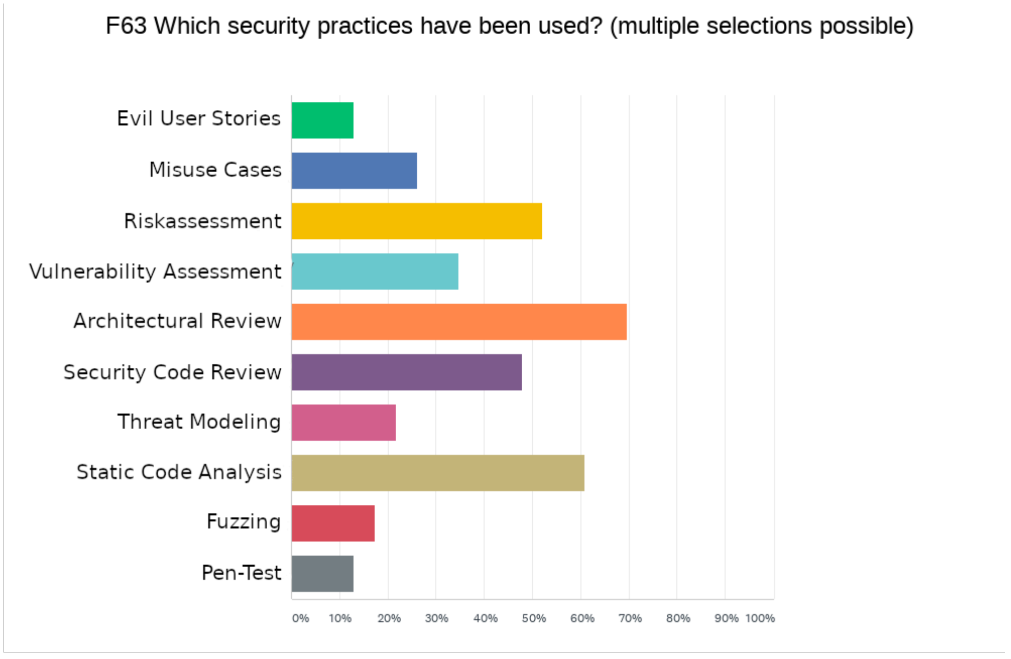

Figure 7 Employed secure software development methods

Updates are supported by most products, but mainly deployed manually (93%). Most (56%) do not provide a fixed update cycle.

Most participants (69%) employ security analysis during development, but only 17% continue after the release.

From the participants, 23% state to not having an overview of the used third-party dependencies, like software libraries.

6.3.6 Discovery Server

On average, 42 participants answered in this category.

As shown in Figure 8, only 9% of participants state that their product supports discovery services using the multicast extension that enables automatic network-wide discovery.

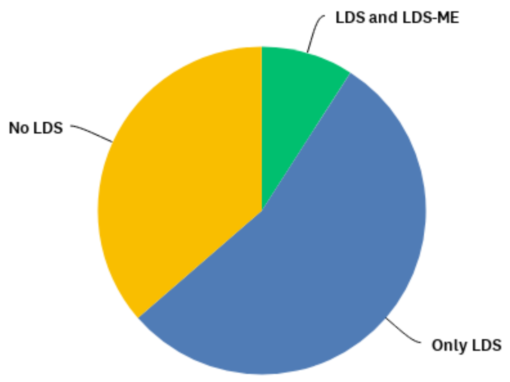

Figure 8 Support for LDS

6.3.7 Global Discovery Server

On average, 13 participants answered in this category. Because of this low number, the results of this category are not reliable.

Almost all participants, 95% state that their GDS-enabled product does not support the highly automated registration process based on LDS-ME.

Regarding the certificate management, 18% support the push, as well as the pull model, with 25% only supporting the push-model, as shown in Figure 9. The majority of these answers are from participants with SDK products.

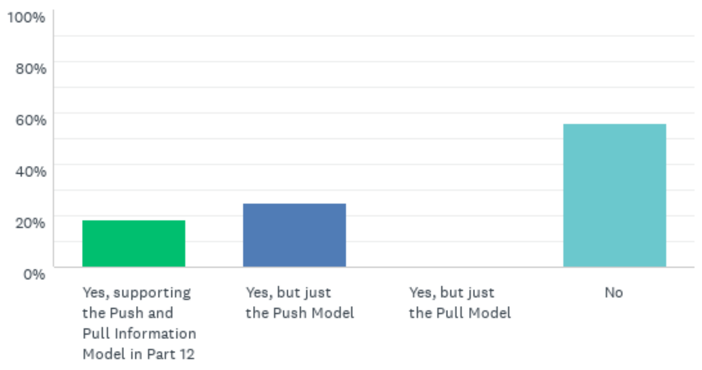

Figure 9 Support for the GDS Push- and Pull-Model

Only one product in each case stated that it also supports the KeyCredential services and the Authorization services.

6.3.8 PubSub

On average, 9 participants answered in this category. Because of this low number, the results of this category are not reliable.

Only 14 participants stated basic support of their product for PubSub with 11 supporting broker-less operation (for example via UDP), and 5 supporting broker-based operation (for example via AMQP or MQTT).

6.4 Summary of the Results

The findings of this survey provide the possibility to draw various conclusion about the security related aspects of the practical usage of OPC UA and enable the deduction of recommendations:

1 Certified Product and Manufacturers
Because of the low number of Feedback from the manufacturers of certified OPC UA products, remarks about security relevant features, other than about SecurityPolicies and UserIdentityTokens, are not possible.

2 Employed stacks
OPC UA is provided by a large number of available stacks that enable the development of respective products. These include proprietary as well as open source solutions suitable for different environments and programming languages. The majority, at least of the participant of this survey, base their applications on stacks with officially certified example applications.
This emphasizes the importance of highlighting security requirements in the certification process, as this would already benefit the majority of end users. The certification process is already incorporating this to a certain extend.

3 Employed Versions
Version 1.04 that was targeted in this analysis is adopted and available since 2017. Still, it is not supported by over one quarter of the products in this survey and older version are still relevant in the active operation. Additionally, in some cases, update is not even planned.
Therefore, it should be made sure that security related developments should also consider impacts on older version of OPC UA and support these older versions in security related aspects.

4 Secure Configuration
A secure default configuration is not a main target of the majority of products in this survey, which means that many of the security recommendations and requirements of the specification are not implemented in practice.
The OPC UA specification should thus provide improved assistance for such secure default configurations and also demand them. As shown in 2), the certification process as well as a precise documentation of product configuration play a key role in these efforts.

5 Supported Security Features
The implementation state of security features, like certain product functions or available certificate checks, conveys the need to transparently show the user which features are available and which are not. The available products still do not meet the expectations in all security relevant areas.
This especially holds true when the results of the security analysis of the specification are projected on the available implementations and products. It is not possible to infer that all OPC UA enabled products can be assumed to have implemented or enabled all security features of OPC UA.

6 Supported Features
The supported features, for example in the area of discovery services or PubSub, emphasize once more

that the feature set of available OPC UA products only covers a subset of the specified features. Support for new features prevails very slowly.

7 Dynamic Security Analysis

7.1 Approach

The dynamic security analysis of the OPC UA protocol in version 1.04 was conducted in three ways: The first and second way are fuzzing campaigns, one as a black box analysis and another a white box analysis. The third way is an analysis of the certificate path validation. Combining the results of each analysis, it is intended to cover as many security related aspects of the protocol as possible while allowing the discovery of deep flaws in addition to more obvious shallow flaws.

The black box fuzzing especially allows a broad application of the analysis, as it is independent of specific implementations as well as made publicly available [9]. The white box fuzzing, while specifically set up on the open62541 implementation, allows for a deeper analysis and demonstrates the stability of a mature implementation against runtime threats. Finally, the certificate path validation test observes the *SecureChannel* of OPC UA and investigates its handling of manipulated or invalid authentication attempts.

7.2 Black Box Fuzzing of the OPC UA Protocol

Fuzzing of a program that processes files, network traffic or any data based on a well-defined protocol, can generally be done in one of two ways: One way is to tailor a fuzzing campaign towards the specific program, considering its unique data flow and internal functionality. The other way is to design the campaign based on the protocol itself, modelling and mutating the individual protocol fields, and thus applying to an arbitrary implementation of the given protocol. In most cases, a program specific fuzzer will perform better on the target implementation while the protocol-based fuzzer will be more widely applicable.

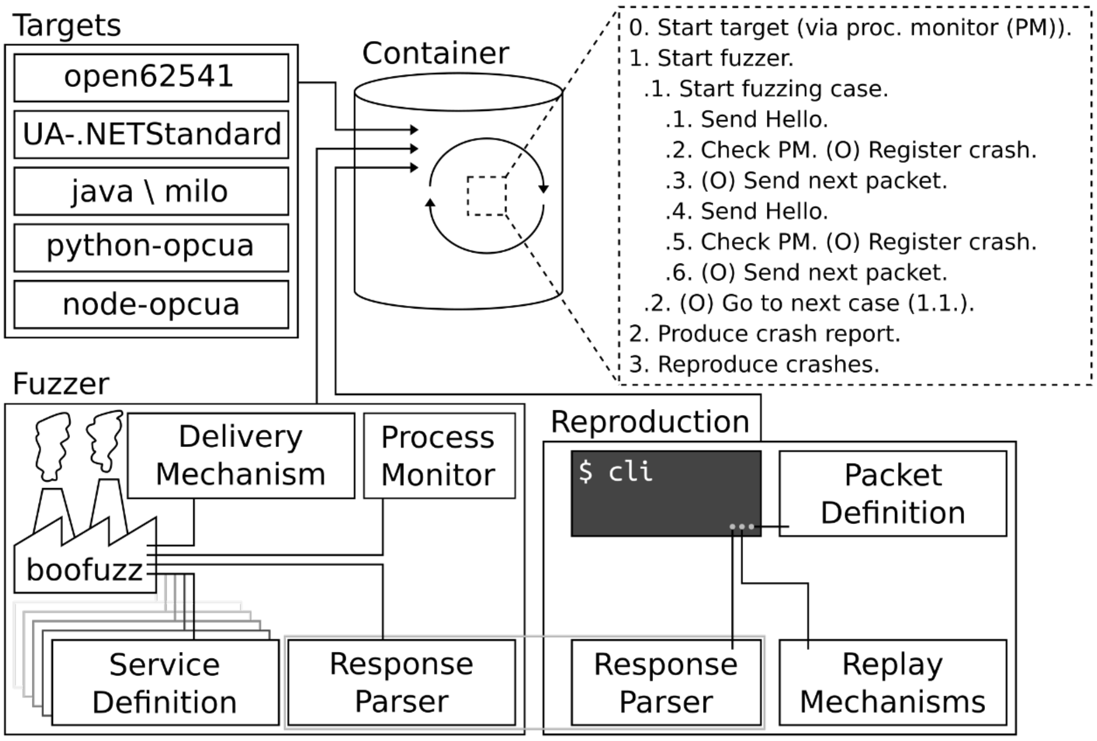

Figure 10 Complete black box fuzzing mechanism including individual fuzzing steps inside the container environment.

7.2.1 Fuzzing Mechanism

In literature, the term fuzzer has different meanings. Originally, the term refers to a mutation engine. Over time, it has come to mean a set of tools or single larger tool that handles the complete process of fuzz testing an application. Besides the mutation of inputs, fuzz testing involves at least a delivery mechanism and some kind of response handling. In the following, we use the term fuzzer in the latter meaning of describing the full-fledged tool.

The black box fuzzing solution utilizes the boofuzz framework [10]. boofuzz is a fuzzing framework specifically developed for fuzzing protocols and includes several important components for this purpose. One component is a library for protocol definition. For a network protocol, it allows to define each specific packet type as well as the order in which the packets are sent. The crafted definitions then function as basis for the mutation engine, as the protocol definition also includes if a given field should be mutated. boofuzz also provides a delivery mechanism that uses the definitions and specified sequences to iterate possible test cases including the sending of packets and receiving of responses. Finally, boofuzz offers a simple process monitor that monitors the server process to detect crashes and collect error messages and logs. The process monitor is also able to restart the server after a crash to allow continued fuzzing.

Figure 10 shows all components of the developed fuzzing mechanism. Besides the fuzzer itself, the mechanism incorporates a reproduction component as well as the target implementations. Finally, a script applies fuzzer and reproduction on the implementation of choice and collects the resulting reports. All four components are built into a container environment for each analysis, isolated from the host system, to allow for scaling and reproducible results. Each analysis then consists of a complete fuzzing campaign on one implementation, for example. the open62541 server. After each analysis, the resulting report is stored on the host system and the container can be removed. The four general steps of each analysis are

1. building the container, including compilation or set up of the target implementation,
2. the fuzzing campaign on all of the defined protocol fields and packet types,
3. collection of crash reports, and
4. the reproduction of all detected crashes.

The only variable part in the container creation is the target set up (1.). Since the fuzzing is independent of implementation, no other components need modification. Each implementation is represented by a suitable example server, which is set up by a custom installation script.

Table 12 Mutations of the Hello message

Field	*Type (simplified)*	*Mutation*
Message ID (HEL)	string	no
Chunk type (F)	char	no
Length (dynamic)	int	no
Protocol version	int	yes
Receiver buffer size	int	yes
Sender buffer size	int	yes
Maximum message length	int	yes
Maximum chunk count	int	yes
Endpoint URL length (dynamic)	int	yes
Endpoint URL	string	yes

The fuzzing campaign can be divided into messages and fields. The *Hello* message consists of about ten fields, seven of which are mutated. Mutating these seven fields yield 2261 unique test cases. Table 12 Mutations of the Hello message shows the *Hello* fields and their respective types. The first three, message ID, chunk type and length exist for all OPC UA messages. Mutating these fields necessarily corrupts communication, as the server will reject the messages. Reasons include a missing ID, a bad chunk type leading to more packets being expected and a packet being cut to early or to late due to the length fields. Hence, these fields are not fuzzed for any of the messages. All other fields of *Hello* are tested in order, each yielding a static number of test cases, resulting in the fuzzing campaigns analysis of *Hello*.

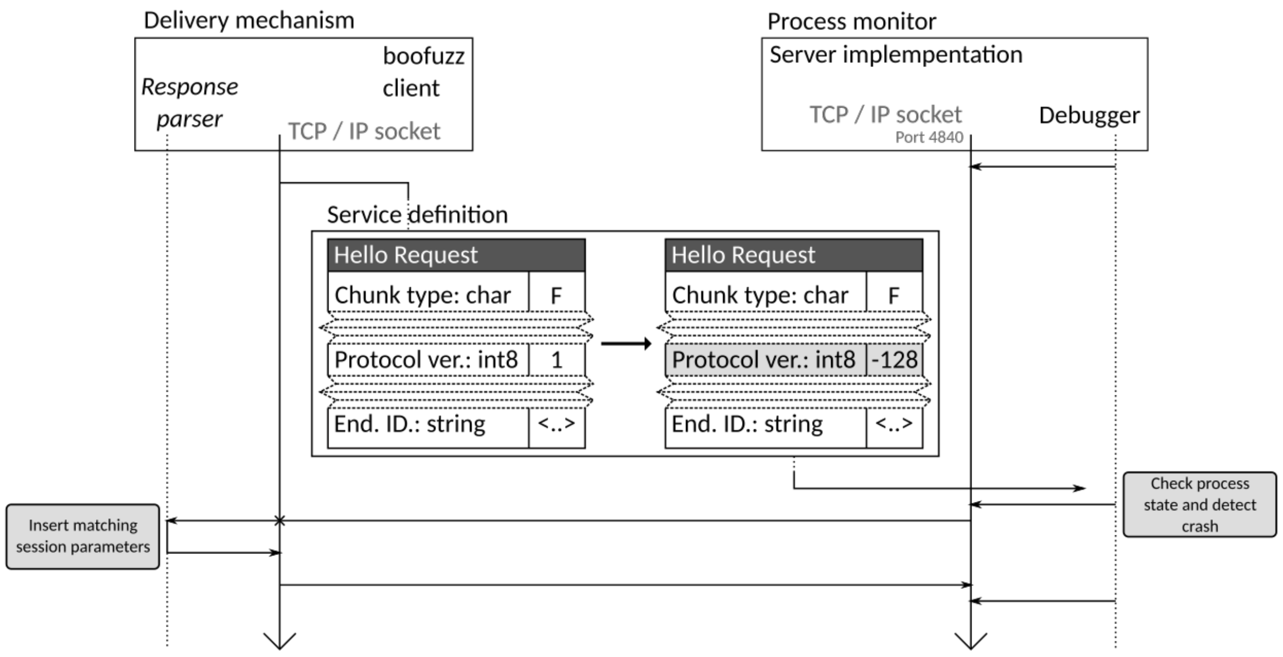

Figure 11 Fuzzing mechanism consisting of delivery mechanism with attached response parser and mutation engine and process monitor for starting and observing the server.

Figure 11 depicts the general sequence of a single test case for the Hello message. On the client side, the participating components are the delivery mechanism, the attached response parser and the service definition of Hello. On the server side the process monitor instruments the target implementation and attaches a debugger. Based on the Hello service definition a single field is mutated and the resulting message sent to the server. In this case the protocol version is mutated. Instead of the original value of 1, the mutated message contains the lowest possible value for the 8-bit integer type. The attached debugger observes the server process, detecting crashes and logging error code as well console logs on a detected crash. This can later be used for triage.

A central mechanism in supporting OPC UA messages inside a secure channel is the dynamic insertion of session parameters. Similar to other connection-oriented protocols, OPC UA keeps a set of values to identify packets with a session and their placement in a sequence of messages. These have to be adhered to allow valid communication. Unlike the static default values for other fields, such as the above depicted protocol version, the session parameters have to be inserted dynamically based on prior messages from the server. The response parser handles the extraction of these session parameters from the server messages and the computation of correct parameters for the client message. The delivery mechanism then inserts them into the mutated packet.

On conclusion of the fuzzing campaign, process monitor data and state information from the delivery mechanism are combined into a single report for each crash to allow reproduction. The resulting information for each crashing test case include:

- Test case number / identifier
- Message type
- Mutated field and exact mutation
- All sent and received messages in binary representation
- Console logs and error code of the server

Boofuzz tracks all test cases in a sqlite database which can be queried to extract only the cases leading to a crash. This gives us the first four data points. The fifth data point comes from the process monitor. The implementation of the boofuzz process monitor does not adhere to a data format that can be used for automated parsing and data mining. Thus, to allow the automated reproduction, a contribution was made to boofuzz to allow receiving the process monitor data in JSON form. Based on the sqlite data from delivery mechanism and JSON data from process monitor, a new JSON file was created to store the combined data for triage and reproduction.

The final step of the fuzzing mechanism is the reproduction, where the combined crash data is used to replay the exact test cases on a freshly spawned server for each test case. This removes possible side effects introduced through previous test cases. The crashing test cases are replayed message by message, with the session parameters being the only fields changed. If the test case again results in a crash, the test case is defined as reproducible and this is added to the crash information stored in the combined JSON file.

7.2.2 Black box fuzzing results on open source implementations

For the black box fuzzing, a total of 13 message types were modelled. Table 13 shows the messages, split into the initial handshake messages for setting up the secure channel, the discovery service set and the session service set. Along with the messages the number of unique mutations for each message type is stated. The number of mutations is stable as boofuzz does not create mutations arbitrarily, but uses a well-honed set of mutation variants for each data type. As seen in the example in Figure 11 setting an integer value to the minimum value of its value range is a mutation that regularly leads to internal errors. For string types, very long values might fit the criteria as well as strings containing unusual characters. Besides a much better efficiency compared to a dumb fuzzing approach, the determination of the test cases allows for easier reproduction of test cases as well as a better comparison between implementations. This also factors into a better usability for certification purposes.

Table 13 Packets / Services that were modeled for fuzzing, including number of mutations for each packet.

Packet	***# Mutations***
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
ActivateSession	4032

The dynamic analysis was conducted on five open source implementations of the protocol. Three of these, open62541, node-opcua and the OPC UA .NET Standard, offer a distinct server executable for CTT based certification that could be used as fuzzing target. For the other two a prime example was used. All five projects use different programming languages, C, C#, Java, python and JavaScript, offering a variety in possible realizations as well as application scenarios. In a recent study targeting openly accessible OPC UA servers (Dahlmanns, et al., 2020), all five implementations could be found deployed, underlining the relevance of the chosen servers.

While the determined nature of boofuzz makes repetition of the fuzzing campaigns unnecessary in theory, each campaign was run thrice with no deviation in the results. Fuzzing was done on a standard PC with a Linux operating system with eight physical processor cores and 32 GB of memory. The container containing the complete fuzzing environment described in section 7.2.1 was created from scratch for each campaign, removing all possible side effects from previous campaigns. Runtime deviated between 1:37 h and 11:07 h over all campaigns, however, it was mostly stable over each set of three campaigns per implementation.

Table 14 List of analyzed products stating programming language, produced crashes and reproduced crashes.

Product	*Language*	*Number of crahes*	*Reproducible Crashes*
open62541	C	184	0
.NET-Refererence Implementation	C#	0	0
Eclipse Milo	Java	167	0
python-opcua	python	188	0
node-opcua	JavaScript	13	13

Table 14 lists the results of the black box fuzzing analysis in terms of crashes. Two observations jump out: Reproduction of crashes was only possible for the JavaScript implementation and only the C# implementation did not crash at all. A detailed comparison of detected crashes between the three iterations per campaign show that all specific crashes were produced in each independent iteration. That means, while reproduction outside of the fuzzing was not possible in most cases, the crashes reappear in each campaign. This leads to the assumption, that the crashes do not derive solely on the test case that trigger them. More likely, a series of test cases up to and including the one that triggers the crash are responsible. Closer introspection into the server processes even showed that the servers were still running in most cases, but simply did not accept any further connections.

For open62541, there are two causes for this behavior: firstly, there is a limit for open connections, once reached it forbids further new connections. Since the fuzzing does not gracefully close connections due to its mutations, this behavior is expected and not viewed as unwanted or erroneous. The second cause is the refusal of packets with bad node ids that stem from mutation of the node id field. This is expected as well and thus also not viewed as erroneous. Further adaptation of the boofuzz framework could allow detecting these specific cases, however requires considerable development effort.

Similarly, for the Java implementation Eclipse Milo, most crashes stem from discarding of the mutated packets. The server logs show an error message stating termination of the current session. The server still accepts new connections even without restart.

Some errors in both Java and python implementations could not be successfully traced to a cause, mostly due to the fact that reproduction was not possible outside of the full fuzzing campaign. It can be assumed that the crashes or errors are caused based on some previous task case. Proof of this assumption would need a detailed review of the targets' source code though.

The 13 reproducible crashes on the JavaScript implementation can be traced to a common bug in the data parsing routine. While the mutations deviate between each case, the error log allows the identification of the exact same line of code causing the crash. The developers of node-opcua closed the issue in the meantime based on our reporting on the issue.

Analyzing all fuzzing campaigns offers a positive conclusion regarding the stability and security of the observed implementations and thus the protocol in its current version. The deterministic manner of boofuzz fuzzing producing the same mutation for each campaign, while not offering an exhaustion of the theoretically possible packets, allows for an arbitrary number of missed bugs. Based on the limitations of network-based fuzzing, this non-exhaustive fuzzing allows for a finite time for the analysis though. Adapting the mutations towards an exhaustive fuzzing has to be contrasted to exhaustion of resources mostly in regards to time. Since the sophisticated choice of mutations offered by boofuzz did identify some bugs and allowed to observe other interesting behavior in the implementations, the choice seems sensible for our use case.

7.3 White Box Fuzzing of open62541

The open62541 implementation was integrated into the google oss-fuzz project (Google, 2016). For this the open62541 developers designed interfaces to 12 potentially critical code areas to allow fuzzing with libfuzzer. The code under test is mostly concerned with encoding and decoding the submitted or received packets. As these routines directly work on outside data by design, bugs in these code areas would make it likely that an adversary can trigger and potentially exploit them. The integration with oss-fuzz offers, if done correctly and with continued tests on newly released code, a continuous observation of the security and thus transparency regarding the security level. The fuzzing builds in oss-fuzz did break a number of times (clusterfuzz, 2020). Still the project found a number of bugs in the implementation since the integration in 2020 (clusterfuzz, 2021). This suggests that continuous extensive fuzzing using the provided test cases offer the possibility to find additional bugs.

For this security analysis, the oss-fuzz integration was reconstructed locally to both confirm the high security level of the implementation in a controlled environment and potentially identify additional bugs or even vulnerabilities.

At first, a realization based on the oss-fuzz base framework clusterfuzz was found, though this showed to be overly complicated. Directly setting up libfuzzer proved more straight forward and easier to manage. The additional instrumentation options offered by clusterfuzz were not necessary for the given purpose. Based on the latest stable oss-fuzz build (open62541, 2020), the fuzzing test cases were ported into a libfuzzer harness directly running on a Linux PC. The PC offers a similar environment as was used for the black box fuzzing: 8 physical CPU cores and 32 GM memory. The libfuzzer harness ran continuously for 117 days. In this time none of the bugs previously found by oss-fuzz were reproduced.

During the fuzzing campaign, libfuzzer with address sanitizer found a set of memory leaks in the **fuzz_json_decode_encode** test case. Most of the leaks show a similar 4-byte leak and show the same stack trace on reproduction. Thus, all identified crashes could be traced to the same cause.

For the **fuzz_json_decode** test case, libfuzzer with address sanitizer found a reproducible *heap buffer overflow*. The issue was reported directly to the developers and was fixed in the meantime.

In addition, two crashes were found for the **fuzz_tcp_message** test case that could not be reproduced.

Based on the coverage of the given test cases and the extensive runtime of the fuzzing campaign, the high-quality standard claimed by open62541, and the OPC UA protocol in general, could mostly be confirmed. In addition, the number of fuzzing results were in line with the oss-fuzz results when scale of runtime and computing power are considered. That an additional, independent fuzzing campaign finds different bugs also underlines the necessity for continuous testing. It is advised to closely monitor the state of oss-fuzz builds and fix build problems immediately. Designing additional test cases could lead to an even better quality assurance.

7.4 Testing the certificate validation

Similar to the original 2016 study on OPC UA, an analysis of the certificate handling and validation for version 1.04 of the standard was conducted. The test cases were generated using the BSI Certification Path Validation Test Tool (CPT) (MTG AG, 2018).

7.4.1 Test mechanism

The components of the developed test mechanism for the certificate validation are depicted in Figure 12. Similar to the fuzzing mechanism described in 7.2.1, the certificate validation uses a container environment to enable reproducible and isolated results. For the target server, the open62541 CTT target was used, just as in the open62541 fuzzing campaign. For each test case a new container is created, containing the target server and the server certificate chain for the given test. The container is removed after each test case so that each case is completely independent from other tests. As open62541 offers two options for the crypto library used in communication and validation, mbedTLS was chosen over openSSL, since the certification with CTT was done using mbedTLS as well. For the client part of the validation, python-opcua was used. For each test case the implemented client is initialized with the correct client certificate for the case and then starts a connection to the open62541 server hosted inside the container.

Since Hello does not contain certificate information yet, the first two packets of the OPC UA handshake are needed before the validation can be tested. The OpenSecureChannelRequest as the third packet then contains the client certificate for the test. On acceptance of the client certificate, the server sends an *OpenSecureChannelResponse* packet. On decline, an *ErrorMessage* packet is sent. The client implementation registers the response, parses the *ErrorMessage* on decline to observe the server's reasoning and stores the collected data into a simple text log.

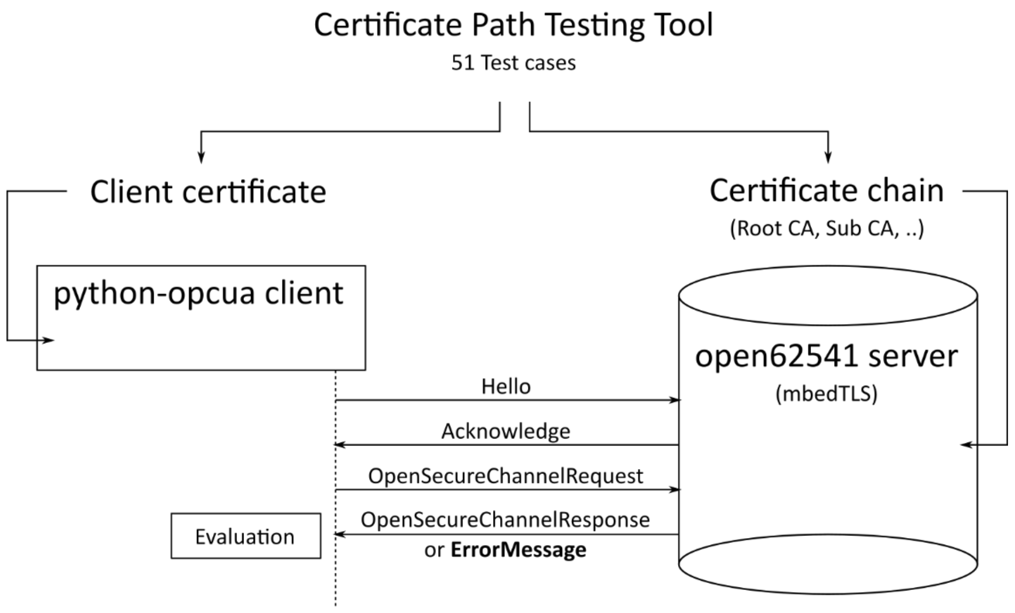

Figure 12 Test mechanism for the certificate validation based on 78 test cases generated from CPT each containing a client certificate and a corresponding certificate chain containing potentially multiple certificates.

Depending on each case, the certificate chain stored inside the container contains at least a root certificate and one or two intermediate certificates. The open62541 CTT server offers a command line option called `--trustlistFolder` that can be used to hand over the certificates in the certificate chain. The revocation list for each test was handed over using the `--revocationlistFolder` option. Revocation lists are used to keep track of untrusted certificates that might look trustworthy based on their certification path. While not all test cases include a revocation list, the OPC UA standard defines the necessity of keeping such a list, a feature also enforced by the open62541 implementation. To allow proper testing, an empty revocation list was generated for each test case that did not contain a list. Furthermore, some adaptations to the CPT were necessary to ensure conformity with the OPC UA standard, as CPT is not developed for OPC UA specifically and the protocol defines multiple obligatory fields in addition to the X.509 standard. Two such fields are the *Key Usage* extension and the target URI.

7.4.2 Analysis of the results

In total CPT defines 94 unique test cases covering a multitude of scenarios in 10 test categories, 5 of which are not applicable to our analysis. The five categories that can't be applied are mail certificate tests, TLS client and TLS server tests, IPsec tests as well as tests incorporating the Online Certificate Status Protocol (OCSP). While the first 4 categories target specific protocols and only make sense when testing implementations for these protocols, the OCSP tests require a remote but controlled OCSP server to work correctly. As OCSP handles revocation of certificates and verification of revocation status, omitting OCSP was viewed a non-issue as revocation status of certificates is already handled in other test cases.

The 5 categories that could be applied contain 53 test cases. Two test cases could not be created as CPT crashes on their invocation. A cursory observation suggests a bug in the CPT mechanism. The remaining 51 test cases resulted in the following behavior:

- 7 Certificates were accepted.
- 2 Test cases led to an error in the test cycle.
- 42 Certificates were declined.

Of the two test cases that led to an error, one also seems to stem from a bug in CPT, while the other stems from the client library. The case that stems from CPT is called EXT_14. For the case, CPT does not generate a client certificate. Since the case description suggests a test of some extension inside a server certificate, this does not seem to be the intended behavior. A source for this error in CPT could not be identified. The case stemming from the client library is called COMMON_05. As the case description suggests, the client certificate for this test has a wrong encoding. Since python-opcua can’t parse this certificate and the parsing routine can’t be bypassed easily, this test case could not be evaluated with our mechanism.

One of the seven accepted certificates, the COMMON_01 test case is correctly validated as it represents a valid certificate to verify that correct certificates are accepted. For the test cases COMMON_06 and COMMON_12, CPT defines the expected behavior as decline, but states that the misconfigurations which are not allowed should be handled gracefully. The tests configure a negative serial number and a too-long subject domain respectively. The fourth case, COMMON_14, defines two different certificate chains with one being valid and another being invalid. Due to the setup of the testing mechanism, the server was only initialized with the valid certificate chain, making the test result itself invalid. The behavior with both chains present can't be inferred based on our data.

The remaining three test cases that led to acceptance of the certificate, ALGO_STRENGTH_01, CRYPT_01 and EXT_17, represent different insecure uses of cryptographic primitives or extensions that are not specifically handled in the OPC UA standard. While this might explain the incorrect handling of the cases on the side of open62541, especially CRYPT_01 where an intermediate certificate has a wrong signature and thus represents a clear trust issue in the chain, it does show potential vectors for abuse. The analysis results for this certificate validation test in general and the incorrectly handled cases specifically were given to the open62541 developers who are taking steps to mitigate the issues in the path validation algorithms.

All of the 42 declined certificates represent test cases where the certificate should be declined. Multiple random samples of these cases showed that the reasoning for rejection was in line with the intended issue. A full evaluation of all 42 cases was not conducted.

7.5 Result summary

The dynamic analysis of the OPC UA protocol was conducted in three partial analyses: Black box fuzzing, white box fuzzing and certificate validation tests. While each analysis observed issues with implementation or protocol, the totality of results is small, giving room for a high grade regarding the security for protocol and implementations under test.

The black box fuzzing mainly found an error in the analyzed JavaScript implementation. Since the implemented fuzzing routine is independent of platform, language and product, it can be reused for future tests as well as in conformance testing and certification. The extensive white box fuzzing campaign produced one real bug in the open62541 library that was reported to the developers and fixed before the final release of this study. The analysis of certificate path validation has mostly yielded the expected behavior, though a small number of test cases led to an unexpected acceptance due to differences in the definition of X.509 in general and OPC UA specifically.

8 Static Code Analysis

8.1 Approach

For the analysis of open62541, automated programs as well as a manual code analysis for security-critical areas of the implementation were used. The automated code analysis tools used in the analyses were Cppcheck, FramaC and Clang.

8.2 Cppcheck

Cppcheck is a static code analysis tool for C and C++. It finds possible vulnerabilities, such as dead pointer, integer overflows, null pointer, or uninitialized variables. In particular, buffer overflows can have security implications. Cppcheck in version 1.90 was used to perform the test.

For the analysis via cppcheck the following command was used:

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

The following results are concerning the analysis of open62541 in version 1.1.1. The output consists of 636 findings. In order to reduce these findings to the important ones that can have an influence on IT security, the deps/ directory has been excluded as these contain external dependencies. Findings of the type *note*, *information*, and *style* have also been filtered out, since these have no influence on IT security.

Finally, 70 findings remain. Among these are 11 errors:

1. 2x `(Error) Memory leak`
   a. `./src/server/ua_server_discovery_mdns.c, Line 236: Memory Leak: newUrl`
      This is an error, which leads to memory not being freed again, which can lead in heavy usage scenarios to noticeable memory leaks. The memory leaks result of the url->data pointer which is overwritten by a new pointer before the old one got freed.
      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, Line 642: Memory Leak: array`
      This is not an error. The respective variable array is allocated and stored in data->mis. Data in turn is stored in cc->clientData and finally returned. Thus, no memory leak occurs here.
2. `(Error) Uninitialized variable`
   a. `./src/pubsub/ua_pubsub_networkmessage.c, Lines 702 + 1122 + 1298, Uninitialized Variable: byte`
      In all three places the variable is only created to query the size of a byte by means of UA_Byte_calcSizeBinary, for which the variable and its contents are of no importance. Therefore, this is not an error.
   b. `./src/server/ua_services_view.c, Line 134, Uninitialized Variable: dummy`
      This variable is not initialized, but values are assigned to the required fields directly afterwards. This could also be set during an initialization. This does not generate an error.
   c. `./src/server/ua_services_attribute.c, Lines 185 + 1189 + 1249, Uninitialized struct member: range.dimensions`
      In all three cases, range.dimensions is not initialized, but a second variable (rangeptr) is checked before trying to release range.dimensions again. In the case that rangeptr contains a pointer, range.dimensions was also assigned a pointer by UA_NumericRange_parse. Therefore, there is no error here.

d `./plugins/ua_nodestore_ziptree.c, Lines 193 + 196, Uninitialized variable: dummy`
  In both cases, dummy is not initialized, but corresponding struct members are set afterwards. In the further usage only these are important, therefore no error occurs here.

In addition to these errors, Cppcheck reported further findings in the categories Portability and Warnings:

3 `(Portability) Shifting signed 64-bit value by 63 bits is implementation-defined behaviour.`
  `./src/ua_types_encoding_binary.c, Line 321`
  In this place, an algorithm was taken from Beej's Network programming guide. It should be checked whether an unsigned long long can be used instead of the signed one.

4 `(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, Lines 243 + 274 + 398 + 1049`
  In all four cases, UA_new is used to allocate memory for the variable. The returned void Pointer is then casted to a pointer of the appropriate datatype (UA_INT64*, UA_UINT64*, UA_DateTime*, UA_UtcTime*) and returned. This should not lead to any problems.

5 `(Portability) %lu in format string (no. 1) requires 'unsigned long' but the argument type is 'size_t {aka unsigned long}'.`
  `./examples/client_historical.c, Line 90`
  `./tests/server/check_server_historical_data.c, Lines 325 + 372 + 465 + 468`
  The length of size_t depends on the system and compiler (for example 32bit vs. 64bit) and can therefore lead to compiler warnings. Instead of %lu, %zu should be used for structs. However, as this is a newer option, it might not be possible for compability reasons. In this case, the value to be output can be cast to maximum size (unsigned long long) and output using %llu.

6 `(Warning) The obsolete function 'alloca' is called.`
  `./src/server/ua_server_discovery.c, Line 54`
  `./src/pubsub/ua_pubsub_writer.c, Lines 283 + 284 + 1802 + 1887 + 1928 + 2010 + 2011`
  `./src/pubsub/ua_pubsub_ns0.c, Line 231 + 531 + 747 + 915`
  `./src/server/ua_services_view.c, Line 612 + 1119`
  `./src/server/ua_services_discovery.c, Line 161 + 229`
  `./src/server/ua_services_discovery_multicast.c, Line 193 + 463 + 500`
  `./src/client/ua_client_highlevel.c, Line 792`
  `./src/client/ua_client_subscriptions.c, Line 339 + 545`
  `./src/server/ua_subscription.c, Line 490`
  `./src/server/ua_subscription_datachange.c, Line 98`
  `./examples/encryption/server_encryption.c, Line 45`
  `./examples/encryption/client_encryption.c, Line 37`
  `./examples/access_control_encrypt/client_access_control_encrypt.c, Line 38`
  These warnings concern the use of the UA_STACK_ARRAY macro. Due to compiler compability reasons, alloca is still used here. In the case that gcc or clang is used, the variable length arrays (VLA) introduced with C99 are used. The use of alloca or VLAs could also be included as an expert build option to provide an easy way of setting up other compilers.

7 `(Warning) %d in format string (no. X) requires 'int' but the argument type is 'unsigned int.`
  `./examples/discovery/client_find_servers.c, Line 44 (no. 1)`
  `./examples/server_ctt.c, Lines 222 (no.1), 286 (no. 1 & 2) + 589 (no. 1)`

`./examples/client.c, Lines 24(no. 2,3 & 5) + 76 (no. 2)`
`./examples/client_async.c, Lines 26(no.2) + 36(no.2) + 49(no.2) + 59(no.2)`
`./tests/server/check_server_historical_data.c, Lines 963 + 970 + 975 + 980 + 1000 + 1005 + 1010 (all no.1)`
These warnings have no impact on IT security. In all cases, an unsigned int is output, while a signed int is expected by the formatting. Instead, %u should be used.

8 `(Warning) %u in format string (no. 2) requires 'unsigned int' but the argument type is 'signed int'`
`./examples/custom_datatype/client_types_custom.c, Line 92`
This warning has no impact on IT security. Here, a signed int is output, while an unsigned int is expected by the formatting. Instead, %i or %d can be used.

9 `(Warning) Possible null pointer dereference: dataA`
`./tests/client/check_client_historical_data.c, Line 207`
This null pointer is potentially possible, if dataSize is not properly set. This is not the case here and therefore does not happen.

8.3 Frama C

FramaC is another static code analysis tool, which was developed explicitly for C code and is open source. It offers additional analyses and requires programs for their static code analysis. For this purpose, the example programs (examples) and tests (tests) were used. For these program codes, FramaC was executed including the module Eva (evolved value analysis). This module activates further tests that are performed on the code without having to make additional annotations in the source code, for example, pre- and post-conditions of individual methods.

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

To speed up the execution, the whole source code of the analyzed files was compiled and saved first.

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

Subsequently, this save was used to analyze all source code files. The call is always structured the same, as an example in the following for client.c:

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

For open62541 in version 1.1.1, the previously described module Eva found the following 23 findings, of which 13 are alarms and 10 warnings concerning the same problem.

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

   a `./examples/client.c, Line 47, assert \valid_read(&(endpointArray + i)->endpointUrl.data)`
   This is not an error, as the index is not counted up too high and the endPoint is properly set.

   b `./examples/server_ctt.c, Line 954, assert \valid_read(argv + i)`
   This is not an error. Here the command line is read and the parameters accordingly with argc and argv are checked. The index is correctly used.

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

   a `./examples/server_loglevel.c, Line 52, assert \valid(&config->logger)`
   This is not an error, as a corresponding logger is initialized when the configuration is created and is therefore accessible.

b `./examples/tutorial_client_events.c, Line 87, assert \valid(&(selectClauses + 0)->attributeId)`
This is not an error, while selectClauses is initialized and passed, such access to attributeId is not out of bounds.

c `./include/open62541/client.h, Line 138, assert \valid(&identityToken->userName)`
This is not an error, as the identityToken is allocated and initialized, part of that being userName.

d `./include/open62541/client.h, Line 139, assert \valid(&identityToken->password)`
This is not an error, as the identityToken is allocated and initialized, part of that being password.

e `[eva:alarm] Warning: signed overflow ./examples/server_settimestamp.c, Line 66, assert -9223372036854775808 < currentTime - (long long)(1800 * (long long)((long long)(10LL * 1000LL) * 1000LL))`
This is not an error, as a negative overflow is not possible. Reason being that the currentTime is always positive and therefore the subtraction cannot lead to an overflow.

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

a `./examples/client_connect.c, Line 39, function: ‘strcmp’, precondition: ‘valid_string_s1’`
This is not an error. Here the command line is used to read a parameter, which is compared afterwards.

b `./examples/common.h, Line 16, function: ‘fopen’, precondition: 'valid_filename'`
This is not an error. After fopen, it is checked whether the access was successful and this helperclass allows a general access on files. Otherwise, the access to possible files could be further restricted.

c `./examples/common.h, Line 28, function ‘fread’, precondition: 'valid_ptr_block'`
This is not an error, as it is ensured that the file can be opened beforehand and fseek is using appropriate values.

d `./examples/server_ctt.c, Line 672, function: ‘snprintf_va_4’, precondition: ‘valid_read_string(param0)’`
This is not an error, however, it should be remarked that a different length than the initialized one is passed. It should be verified, whether this is intended.

e `./examples/server_loglevel.c, Line 37, function: ‘atoi’, precondition: 'valid_nptr'`
This is not an error. Optarg is used by getopt_long to return options of passed arguments. The usage is handled correctly here.

f `./include/open62541/types.h, Line 175, function ‘strlen’, precondition: 'valid_string_s'`
This is not an error. If the maximum length is known, strnlen can be used to ensure that reading will also stop when reading non NULL-terminated strings.

4 `[eva:locals-escaping] locals {name} escaping the scope of a block ./examples/server_ctt.c, Line 592 (10 Warnings)`
These are errors, which created 10 warnings. The problem is that the pointer to the locale variable name leave the scope of name. This happens through saving the pointer in object_attr.description (Line 590),

object_attr.displayname (Line 591) and the addition via UA_QUALIFIEDNAME in UA_Server_addObjectNode (Line 592). This can lead to undefined behavior, as the local variable name is released after the scope of name. Instead, memory should be allocated for name.

8.4 Clang

Clang is a compiler front-end for C and C++ and uses the LLVM Framework. In addition to that, Clang delivers further static code analyses, which were used for the analysis of open62541. For this purpose, CMakeLists.txt has been modified by removing the -Werror flag, so it runs to completion even when warnings are found. By default, the project enabled more flags than that are covered by -Wall and -Wextra. Only the warnings for -Wno-static-in-inline, -Wno-overlength-strings and -Wno-unused-parameter are switched off, which have no influence on security. No further warnings were generated using these settings, which is not surprising as the default use of -Werror means that the compilation is no longer successful as soon as a warning is generated.

8.5 Manual Analysis

For the manual analysis, the code was examined for security-critical errors. In particular, encryption, signature, integrity, session handling, security policies, access control and server and client examples were considered. The files and functions considered are listed below. The results are presented afterwards.

- `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`

The review produced the following results:

1 Memory Leak

  a `ua_securechannel_crypto.c, Line 115`
  In case of an error, the allocated memory of buf is not released again.

2 Certificate Verification
  `plugins/ua_pki_default.c`

  a Line 297: The minimal key length is set to 1024bits for all certificates. However, this should depend on the used policy.

  b Line 310-429: After a failed verification, it is checked whether the certificate is included in the trusted certificates. If this is the case, the issuer certificates are regarded as trusted and the certificate is verified again. The verification appears faulty at this point and the code is difficult to understand. In particular, the entire certificate chain should be checked using mbedTLS (certificate -> intermediate -> trusted root). In addition, the check whether a revocation list exists for the CA certificate should only be implemented in one place to avoid duplicate code. It is recommended to revise these places and to document them comprehensibly.

  c Line 491: Verification of the specified ApplicationUri of the certificate is allowed as soon as it is contained anywhere in the v3_ext field. Here it should be ensured that the ApplicationUri is its own entry and not part of a longer entry.

3 Security Policies

  a All implemented security policies do not check the lower and upper bounds of the asymmetric key lengths. Thus, certificates that are too weak could also be used for a policy.
  For mbedTLS policies (ua_securitypolicy_basic128rsa15.c line 707, ua_securitypolicy_basic256.c line 657, ua_securitypolicy_basic256sha256.c line 699): In addition to the constant, an additional value could be derived, for example, from the configuration, to use as the personalization entropy. This way, different personalization entropies can be obtained for different systems.

  b plugins/securitypolicies/openssl/ua_openssl_basic256sha256.c, Lines 259 + 273 + 547
  UA_Assert is used to check whether the key length corresponds to 2048 bits. However, the policy also allows longer key lengths in the specification.

4 Access Control

  a Plaintext Passwords
  In line 91 of plugins/ua_accesscontrol_default.c, the passed password is directly compared with the stored password. This means that the passwords are stored in plain text. At least salting and hashing should be used for password storage, for example, by using Argon2 or scrypt. Otherwise, there is a risk that this procedure will be adopted by developers for their own implementation and that an attacker will simply be able to access all passwords when the OPC UA server is taken over.
  Furthermore, the storage of additional plaintext passwords in the code can then be removed as well, since these must then also be stored as hash values (ua_config_default.c line 112, examples/access_control/server_access_control.c line 51+52).

  b Default Passwords
  Users, including their passwords, are defined in plugins/ua_config_default.c (line 112) and are configured as the default users through the following functions: UA_ServerConfig_setMinimalCustomBuffer (line 454) and UA_ServerConfig_ setDefaultWithSecurityPolicies (line 661). These two functions are additionally used by two further functions: UA_ServerConfig_setMinimal and UA_ServerConfig_ setDefault. This behavior can be overlooked by a developer and should be avoided. In general, default users and passwords should be avoided and changed at the latest when used for the first time.

  c Default Access Rights
  The functions of plugins/ua_accesscontrol_default.c to check whether a certain right exists are only placeholders and grant full rights. From a security point of view, it is better to deny all access by default in case no custom permissions are granted or implemented.

5 Example Server

  a Default Passwords
  Server.cpp and server_ctt.c use the default configuration, which uses the previously mentioned default users. This is not clear to a developer when using this example and could lead to these default users finding its way in further implementations.

6 Example Client

  a Hardcoded Passwords
  In line 55 of /examples/client.c, the source code contains credentials for establishing an OPC UA connection. This is not a problem here as these are neither real credentials nor is a real application. Nevertheless, an example can contribute to similar implementations being adopted by developers. For this reason, it would be better not to store the credentials in the source code, but, for example, to let the user enter them instead.

7 Build

  a The default of the flag Enable_Encryption is set to OFF. Following the secure-by-default approach, it would be better to set this to ON by default.

8.6 Summary

In summary, no serious vulnerabilities were found during the analysis and the code is generally at a very high security level. Only a few minor issues were found, but these do not contain any direct security vulnerabilities. However, the examples are using access controls with hardcoded plaintext passwords. This could lead to them being copied and used in production. It would be desirable if the passwords were exemplary stored as a hash including a salt instead, in order to serve the developers as a good example of a password storage.

All findings were reported to the open62541 project. The findings were accepted and will be improved upon in future versions of open62541.

9 Conclusion by BSI

As this analysis has shown, the large number of changes in OPC UA specification have made an update of the previous analysis advisable. Just as the 2016 analysis, no conceptual security flaws were identified. The new recommendations present suggestions for improving the specification, by avoiding implementation errors or by clarifying parts of the specification. These recommendations have been positively received by the OPC Foundation and will be discussed in the standardization bodies.

New features, like PubSub, could not be analyzed yet and are therefore not part of this security analysis.

This analysis has shown the insufficient implementation of many security features in existing products. This results in a rather unsatisfactory current situation. The OPC UA specification defines security features that lead to a generally secure system. However, the implementation of these features in products is strongly lagging behind or only covers them partially.

Therefore, manufacturer of automation components and libraries are asked to intensively study and implement security related features, and also ensure that their clients and end users are getting briefed and trained appropriately. Awareness of operators, security relevant documentation and a „secure by default” deployment of OPC UA products, require the willingness of the user side to attend respective trainings.

The open62541 implementation of OPC UA, that has been investigated in the context of this analysis, has shown that employing strict testing during development leads to a high code quality.

10 Annex A: List of surveyed publications

Table 15 List of surveyed publications

Title	*Year*	*Authors*	*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

Title	*Year*	*Authors*	*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

Title	*Year*	*Authors*	*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 Annex B: Cybersecurity-related changes since version 1.02

This section summarizes the cybersecurity-relevant changes of the OPC UA specification since version 1.02 (1.02.47 for Part 12).

11.1 Part 1

11.1.1 Considered documents

The following documents have been published by the OPC Foundation since version 1.02 (as of 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 Comparison between [1] and [3]

Editorial changes:

- Software changed to Digital (X.509)
  - „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]

Added, removed or modified security concepts:

- The section "Redundancy" has been moved. In addition, the concepts of client, server and network redundancy are introduced. Furthermore, the server modes transparent and non-transparent are introduced (see [3, 6.4]).

11.1.3 Comparison between [3] and [7]

Editorial changes:

- The definition of certificates was incorrect and has been adjusted:
  - "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]

Added, removed or modified security concepts:

- Terms, definitions, concepts and sections on PubSub have been added:
  - Terms and definitions:
    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]

- Sections:
  Added PubSub to the OPC UA specification [7, 4.1] [7, 4.3] [7, 5.3] [7, 6.5] [7, 6.6]
- The description "Authority-generated software Certificates" has been removed in [3, 5.4.1.2]

Other changes:

- JSON was added as a third encoding format [7, 5.3]
- SOAP/http has been removed as transport protocol, WebSockets have been added [7, 5.3]
- WS-SecureConversation has been removed as an example in [7,7.3]

11.2 Part 2

11.2.1 Considered documents

The following documents have been published by the OPC Foundation since version 1.02 (as of 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 Comparison between [1] and [2]

Editorial changes:

- Definition of non-repudiation adapted [2, 3.1.24]
- Added note to the definition Private Key [2, 3.1.27]
- Added note to the definition Public Key Infrastructure (PKI) [2, 3.1.28]

Extended sections:

- Security objective Authentication [1, 5.2.2] extended to Application [2, 5.2.2] and User Authentication [2, 5.2.3]
- Section Strict Message processing [2, 6.3]
- Section Random Number Generation [2, 6.3]
- Section Audit event management [2, 6.11]

Added sections:

- Cryptographic Keys [2, 6.8]
- Unsecured Services [2, 7]
- Certificate Management [2, 8] (First details already described in [1, 6.11])

11.2.3 Comparison between [2] and [5]

Editorial changes:

- Changed term “Digital Certificate” to “Certificate” [5, 3.1.4] [5, 3.1.11] [5, 8] ...
- Note for definition Application Instance Certificate extended [5, 3.1.4]

Added/removed terms and definitions:

- 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 removed [3, 3.1.16]

Introduction of PubSub with extensions and adaptations of already existing sections:

- Generalization: [5, 4.1] [5, 4.3.2] [5, 4.3.5] [5, 4.3.6] [5, 4.3.7]
- Section Rogue Publisher added [5, 4.3.11]
- Security architecture modified [5, 4.5.1] [5, 4.5.3]
- Section Security Policies modified [5, 4.6]
- Section OPC UA security related Services extended [5, 4.13]
- Added PubSub in section Security reconciliation [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]

Additions to authentication and authorization:

- Authorization [5, 4.2.3] section extended
- User Authentication [5, 4.9] section modified w.r.t. PubSub
- Application Authentication [5, 4.10] section extended
- User Authorization [5, 4.11] section modified
- Section Roles added + Figure [5, 4.12]
- Application authentication [5, 5.2.2] section extended
- User Authentication [5, 5.2.3] section modified
- OAuth2, JWT and User roles [5, 6.12] section added

Extended and modified sections:

- Removed descriptions and references for XML/SOAP and WS-SecureConversation [2, 1]
- Extended Message flooding [2, 4.3.2] to DoS [5, 4.3.2] [5, 5.1.2]
- Security threat Message alteration adapted [5, 4.3.5]
- Security threat Eavesdropping adapted [5, 4.3.3]
- Message replay [5, 4.3.6] section extended
- Rouge Server [5, 4.3.10] section extended
- Security threat Compromising user credentials adapted [5, 4.3.12]
- Introduction of the online Profile-Tools in section Security Profiles [5, 4.7]
- Added Table in [5, 5.1.1]
- Message spoofing [5, 5.1.4] section extended
- Message replay section extended [5, 5.1.6]
- Confidentiality [5, 5.2.5] section adapted
- Administrative access [5, 6.7] section extended
- Description for Certificate Store extended [5, 8.1.4.2]

New sections:

- Added security objective Non-Repudiation [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 Considered documents

The following documents have been published by the OPC Foundation since version 1.02 (as of 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 Comparison between [1] and [3]

Extended concepts:

- Extended AccessLevel and UserAccessLevel in table 8 [3, 5.6.2]
    - StatusWrite, TimestampWrite

11.3.3 Comparison between [3] and [4]

Editorial changes:

- The unreliability for clients for the UserAccess, UserExcecutable and UserWriteMask attributes is pointed out [7, 5.2.8] [7, 5.6.2] [7, 5.7]
- Description for the SemanticChange attribute adapted [7, 5.6.2]

New concepts:

- Added the concept for roles [7, 4.8] [7, 5.2.9] [7, 5.2.10] [7, 5.2.11]
  - Added section Roles [7, 4.8]
  - Added attributes to Base NodeClass [3, 5.2.1]
    - RolePermission [7, 5.2.9]
    - UserRolePermissions [7, 5.2.10]
    - AccessRestrictions [7,5.2.11]
- Changed data type of WriteMask and UserWriteMask in Base NodeClass [7, 5.2.1] [7, 5.2.7] [7, 5.2.8]
  - Changed UInt32 to AttributeWriteMask
- Added attribute AccessRestrictions to Base NodeClass [7, 5.2.1] [7, 5.2.11]
- Added attribute AccessLevelEx to Variable NodeClass [7, 5.6.2]
- Adapted description for attributes AccesLevel and UserAccesLevel [7, 5.6.2]
  - StatusWrite and TimeStampWrite [7, 8.57]
  - Type changed from byte to AccessLevelType

11.4 Part 4

11.4.1 Considered documents

The following documents have been published by the OPC Foundation since version 1.02 (as of 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 Comparison between [1] and [3]

New terms and definitions:

- 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]

Changes to the Discovery Service Set:

- Description added to figure 9: Setup of the SecureChannel was not included [3, 5.4.1]
- Description added in [3, 5.4.1]: EndpointDescription refers to the CreateSession-Response
- All possible HostNames should be included in the server's certificate [3, 5.4.1]
- New services
  - FindServersOnNetwork [3, 5.4.3]
  - RegisterServer2 [3, 5.4.6]
- FindServers
  - Several entries per server are now allowed [3, 5.4.2.1]
  - Added description for non-transparent servers [3, 5.4.2.1]
  - The applicationName of the ApplicationDescritption is to be used as LocalId [3, 5.4.2.1]
- GetEndpoints
  - Server is allowed to send AIC with SecurityPolicy None [3, 5.4.4.1]
- RegisterServer
  - Added description: Implemented by Discovery Servers [3, 5.4.5.1]
  - Added description: Service requires Client Authentication [3, 5.4.5.1]
  - Moved type description for RegisteredServer to section 7.29 [3, 5.4.5.2]

Changes to the SecureChannel Service Set:

- Removed SOAP and WS-SecureConversation references [3, 5.5.1]
- Added description for HTTPS [3, 5.5.1]
- OpenSecureChannel
  - Changed the handling of updated SecurityTokens (should to shall) [3, 5.5.2.1]
  - Added description: Security concepts do not apply to SecurityPolicy None [3, 5.5.2.1]
  - Added text for validating the HostName [3, 5.5.2.1]
  - Client is allowed to send AIC even with Security Policy None [3, 5.5.2.2]
  - Added text for closing SecureChannels when the limit is reached [3, 5.5.2.1]
  - Added text for Bad_NonceInvalid [3, 5.5.2.3]

Changes to the Session Service Set:

- CreateSession
  - Added text for DoS-attacks [3, 5.6.2.1]
  - Client is allowed to send AIC even with Security Policy None [3, 5.6.2.2]
  - Server is allowed to send AIC even with Security Policy None [3, 5.6.2.2]
  - Added text for Bad_NonceInvalid [3, 5.6.2.3]
- ActivateSession
  - Removed references to SoftwareCertificates

    - Missing UserIdentityToken is mapped to Anonymous [3, 5.6.3.2]
- CloseSession
    - Behavior for closing the session before activating the session is described [3, 5.6.4.1]

Other changes:

- Bad_SecurityModeInsufficient added [3, 5.10.2.4]
- Added text w.r.t. the independence between Subscription and Session [3, 5.13.1.1]
- Adapted text for AIC [3, 6.1.2]
- Adapted text and table for the certificate validation steps [3, 6.1.3]
- Description for SoftwareCertificates removed [3, 6.2]
- No AuditEvent for OpenSecureChannel with renew-requestType (if successful) [3, 6.3.5]
- Adapted section Redundancy [3, 6.4]
    - New section Manually Forcing Failover [3, 6.4.5]
- New section Durable Subscriptions [3, 6.6]
- New parameter DiscoveryConfiguration [3, 7.9] and MdnsDiscoveryConfiguration [3, 7.9.2]
- Order of the parameter SignatureData adjusted [3, 7.32]
- Adapted text for UserNameIdentityToken [3, 7.36.3] and IssuedIdentityToken [3, 7.36.5]
    - Note added for unencrypted transmission of passwords [7.36.3]
- Fixed figure 1 to DiscoveryServiceSets [3, 4.1]

11.4.3 Comparison between [3] and [7]

New terms and definitions

- DiscoveryEndpoint [3.1.3]

Changes to the SecureChannel Service Set:

- OpenSecureChannel
    - Added note that AIC are not mandatory [7, 5.5.2.1]
    - Changed data type of secureChannelId and channelId to BaseDataType [7, 5.5.2.2]
    - Length of the client and serverNonce depends on the SecurityPolicy [7, 5.5.2.2]
    - Adapted text for requestedLifetime w.r.t. threats [7, 5.5.2.2]
- CloseSecureChannel
    - Text added: Response can be ommited depending on protocol [7, 5.5.3.2]
    - Changed data type from secureChannelId to BaseDataType

Changes to the Session Service Set:

- CreateSession
    - Adapted description for parameters [7, 5.6.2.2]
        - maxResponseMessageSize, ServerEndpoints, serverSignature, maxRequestMessageSize
- ActivateSession
    - Added text to protect against attacks on UserIdentityToken [7, 5.6.3.1]

- Added text for parameter clientSignature w.r.t. certificate chains [7, 5.6.3.2]

Other changes:

- Extended certificate validation steps in table 6 [7, 6.1.3]
  - New: Build Certificate Chain, Security Policy Check
- Replaced Authorization Service in Figure 22 und 23 with Identity Provider [7, 6.1.5] [7, 6.1.6]
- Adapted text for User identity Token [7, 6.1.5]
- New Authorization concepts introduced
  - Added section Authorization Services [6.2]
  - Added new definition for UserIdentityToken [7, 7.36]
    - Added IssuedIdentityToken [7, 7.36.1]
    - EncryptedSecretFormat vs. Legacy Encrypted Token Secret Format
  - UserTokenPolicy modified [7, 7.37]
- Session-less Service invocation added [7, 6.3]
- Adapted section ClientRedundancy [7, 6.6.3]
- Added description for the establishment of new SecureChannels [7, 6.7]
- Added text for ReverseConnect [7, 6.7]
- Adapted figure 38 [7, 7.31]
- Adapted section UserIdentityToken parameters [7, 7.36]
- Clarification that discovery services use SecureChannel with SecurityPolicy None
  - FindServers, GetEndpoints [7, 5.4.1]
- Adapted description w.r.t. profileUrls for the GetEndpoints-Service in table 5 [7, 5.4.4.2]
- Added text: Server can now also initiate logical connections [7, 5.5.1]
- Certificate replaced with public or private key at several places
  - [7, 5.5.2.1] [7, 5.5.2.2]
- New StatusCodes

11.5 Part 5

11.5.1 Considered documents

The following documents have been published by the OPC Foundation since version 1.02 (as of 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 Comparison between [1] and [3]

Editorial changes:

- Clarification of securityRejectedSessionCount and rejectSessionCount in table 143 [3, 12.9]
- Adapted description in table 149 for authenticationMechanism [3, 12.12]
- Should -> Shall in AuditEvents
- Adaptions w.r.t. Redundancy
- NonTransparentRedundancyType [3, 6.3.9]
- RedundancySupport [3, 12.5]

11.5.3 Comparison between [3] and [8]

Added Roles:

- Added RolePermissions, UserRolePermissions as Common Node attributes [8, 5.1]
- Added RoleSetType in table 10 [8, 6.3.2]
- Added DefaultRolePermission, DefaultUserRolePermissions and DefaultAccesRestrictions in table 21 [8, 6.3.13]

Additions:

- Added AccessRestriction as Common Node attribute [8, 5.1]
- Added AccessLevelEx as Common Variable attribute [8, 5.3]
- Added new property StatusCode to AuditSecurityEventType [8, 6.4.4]

Other changes:

- Added Common Method attributes [8, 5.5]
- Added EndpointType [8, 12.22]
- Added Annex F User Authentication

11.6 Part 6

11.6.1 Considered documents

The following documents have been published by the OPC Foundation since version 1.02 (as of 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 Comparison between [1] and [3]

Editorial changes:

- Range of the Built-in data types Byte and UInt32 corrected [3, 5.1.2]
- Added clarification that OpenSecureChannel messages are signed and encrypted with *SignAndEncrypted* and *Sign* [3, 6.7.4]
- Changed Bad_TcpUrlRejected to Bad_TcpEndpointUrlInvalid [3, 7.1.2] [3, 7.1.5]

Adjustments:

- Added notes that nesting of data types can lead to stack overflow errors [3, 5.1.5] [3, 5.2.2.12] [3, 5.2.2.15] [3, 5.3.1.13]
- Added CertificateSignatureAlgorithm-Parameter in table 22 [3, 6.1]
- Adapted text for SubjectAltName in table 23 [3, 6.2.2]
- WS Secure Conversation marked as deprecated [3, 6.6] [3, 7.2]
- Added default for ReceiverCertificateThumprintLength in table 27 (in case no certificate follows) [3, 6.7.2]
- ExtraPaddingSize and Padding parameter in the message footer in table 30 switched [3, 6.7.2]
- Data type for parameter CreatedAt in OpenSecureChannel-Response Message adapted
- Adapted text for SenderCertificates in [3, 6.7.6]
- Adapted text w.r.t. the XML-Encoding for SOAP over HTTPS [3, 7.3.2]

11.6.3 Comparison between [3] and [7]

Editorial changes:

- Added “inclusive” in the description for the built-in data types [7, 5.1.2]
- Changed SecurityMode Sign to SignOnly [7, 6.1]
- Added “v3” for X.509 [6, 7.2.1]
- Adapted text for OPC UA Secure Conversation Mesage header in table 41 [7, 6.7.2.2]
- Replaced MaxCertificateSize with MaxSenderCertificateSize in table 41 [7, 6.7.2.3]
- Clarification for parameter ReceiverCertificateThumbprintLength in table 42 [7, 6.7.2.3]
- Added Part 2 reference for random number generators [7, 6.7.4]
- Clarification for HTTPS certificates [7, 7.4.1]
- Added text for suppressing certificate validation steps [7, E.6]

Adjustments:

- Added definition for Certificate Digest [7, 3.1.1]
- Figure 1 adjusted [7, 4]
  - TransportProtocols: SOAP/http removed, HTTPS and AMQP added
  - SecurityProtocols: WS Secure Conversation removed
  - Data Encoding: UA JSON added
- Added section *OPC UA JSON* [7, 5.4]
- Added SecureChannelNonceLength to SecurityPolicy in table 35 + text adjusted [7, 6.1]

- Removed text for ServerSoftwareCertificates and ClientSoftwareCertificates [7, 6.2.1]
- Removed section *SignedSoftwareCertificate* [7, 6.2.3]
- Added section *Certificate Chains* [7, 6.2.3]
- Added section *JSON Web Token (JWT)* [7, 6.5.2]
- Added section *OAuth2* [7, 6.5.3]
- Close SecureChannel, when gap occurs in sequence number [7, 6.7.2.4]
- Extracted SignatureSize for the calculation of the MaxBodySize [7,6.7.2.5]
- Changed data type of RequestedLifetime attribute to UInt32 [7, 6.7.4]
- Description of section Deriving key extended + new table [7, 6.7.5]
- Section on abstract OPC UA Connection Protocol added [7, 7.1] and OPC UA TCP moved to 7.2
- Minimum size of MessageChunkSize corrected to 8192 [7, 6.7.2]
- Added Reverse-Connect for all transport protocols
  - Added RHE in table 50 of the UACP MessageHeaders [7, 7.1.2.2]
  - New Section *ReverseHello Message* [7, 7.1.2.6]
  - Section *Establishing a connection* adjusted [7, 7.1.3]
- Text added for the case there are no more resources for the SecureChannel [6, 7.1.2.3]
- Removed concept Error Recovery [3, 7.1.6]
- Added new section *Session-less Services* [6, 7.4.2]
- Added new section *JSON Encoding* [6, 7.4.5]
- Added new section WebSockets [6, 7.5]
- Default Port für OPC UA HTTPS zu 443 geändert [6, 7.6]

11.7 Part 12

11.7.1 Considered documents

The following documents have been published by the OPC Foundation since version 1.02.47 (as of 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

< Version 1.02.47 used in the previous study is no longer available in the OPC Foundation archive. >

11.7.2 Comparison between [1] and [2]

Added CertificateGroups

- Added definition for Certifcate Group [2, 3.1.2]
- Changed AuthorityType to CertificateGroupType [2, 7.5.9]
- Added CertificateGroupFolderType [2, 7.5.15]
- Adjusted information model for Pull Certificate Management [2, 7.6]

    - Added CertificateDirectoryType with DefaultApplicationGroup, DefaultHttpsGroup and DefaultTokenGroup [2, 7.6.1] [2, 7.6.2]
    - Added method GetCertificateGroup [2, 7.6.1] [2, 7.6.6]
    - All methods w.r.t. CertificatesGroup adjusted, incl. result codes
    - CertificateRequestedAuditEventType adjusted [2, 7.6.9]
- Adjusted information model for Push Certificate Management [2, 7.7.1]
    - GetRejectedList-Methode added
    - CertificateDirectoryType added

Other changes

- Changed flag isOffline to isOnline [2, 4.2.2]
- Added reference to Part 2 [2, 7.1]
- Added text for AddCertificate-Methode w.r.t. the validation of certificates [2, 7.5.5]
- Changed RsaBasicApplicationCertificateType to RsaMinApplicationCertificateType [2, 7.5.13]
- Changed Rsa2048ApplicationCertificateType to RsaSha256ApplicationCertificateType [2, 7.5.14]
- Added note that the method shall be called with an encrypted SecureChannel [2, 7.5.4] [2, 7.5.5]

11.7.3 Comparison between [2] and [3]

Added Authorization Services

- Added definition Network Services [3, 3.1.15]
- Added section *Authorization Services* [3, 9]

Added KeyCredential Mgmt.

- Added definition KeyCredential [3, 3.1.4] and KeyCredentialService [3, 3.1.5]
- Added section Key Credential Management [3, 8]

Changes w.r.t. ReverseConnect

- Generalization of the used term “server” to “application” or consideration of clients [3, 4.1] [3, 6.1]
- Adjusted GDS information model
    - Changed QueryServers method to QueryApplications method [3, 4.3.4] [3, 6.3.10] [3, 6.3.11]
    - Added method FindApplicationsByEndpoint [3, 6.3.1]
    - Added method FindApplications [3, 6.3.1]

Other changes:

- Registration with method RegisterServer2 now mandatory [3, 4.2.2]
- Description text for step 4 of the GDS discovery process adapted [3, 6.2.1]
- Added section *Domain Names and MulticastSubnets* [3, 6.2.5]
- Adjusted description in section *Provisioning* [3, 7.4]
- Added variable UpdateFrequency to TrustListType [3, 7.5.2]
- Enhanced description of the methods AddCertificate and RemoveCertificate w.r.t. certificate chains [3, 7.5.5] [3, 7.5.6]

- Added TrustListOutofDateAlarmType [3, 7.5.9]
- CertificateExpired and TrustListOutOfDate object added for CertificateGroupType [3, 7.5.10]
- Added UserCredentialCertificateType [3, 7.5.11] [3, 7.5.14]
- Added note about changing the AIC during the ApplyChanges method. [3, 7.7.5]

11.8 Errata

11.8.1 Considered documents

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

11.8.2 Part 3

- ApplyRestrictionsToBrowse bit has been defined

11.8.3 Part 4

- Added note for padding of UserIdentityTokens
  - Addressing CVE-2018-7559
- Added note for validating a server certificate before sending an encrypted UserIdentityToken
  - Addressing CVE-2018-12087
- Changed enumeration type UserIdentityTokenType to UserTokenType

11.8.4 Part 5

- IdentityMapping for applications

11.8.5 Part 6

- Added maximum key size for Issuer Certificates
- Added used but not defined StatusCodes
  - Bad_CertificateUnknown
  - OpcUa_BadSequenceNumberInvalid
- Behaviour defined for rollover of the sequence number
- ProtocolVersion 0 introduced

11.8.6 Part 7

- SecurityPolicy – None deactivated it other available

11.8.7 Part 12

- Clarification for method AddCertificate w.r.t. Issuer Certificates
- Alignment of the text of StartSigningRequest and CreateSigningRequest
- Subscriber removed as server capability, registration not necessary
- Default Value for CertificateGroupId of the GetTrustList method defined
- Generalisation of RSA, ECC requiered for KeyCredentialManagement

1

Bibliography

[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-tool_node.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/]