VDMA @ Forschungsfabrik Karlsruhe, 22.06.2022gn

open62541 - Secure by Design?

Dr.-Ing. Julius Pfrommer, Fraunhofer IOSB

Agenda

- Introduction
- What is OPC UA and why should you care?
- Security by design in the protocol
- Security by processes and tools used by the open62541 implementation
- Time for questions

Dr.-Ing. Julius Pfrommer

- Head of the research group “Distributed Cyber-Physical Systems” at Fraunhofer IOSB
  - Flexible Production Control
  - Machine-Learning for Industrial Applications
- PhD in D*istributed Planning for Self-Organizing Production Systems*

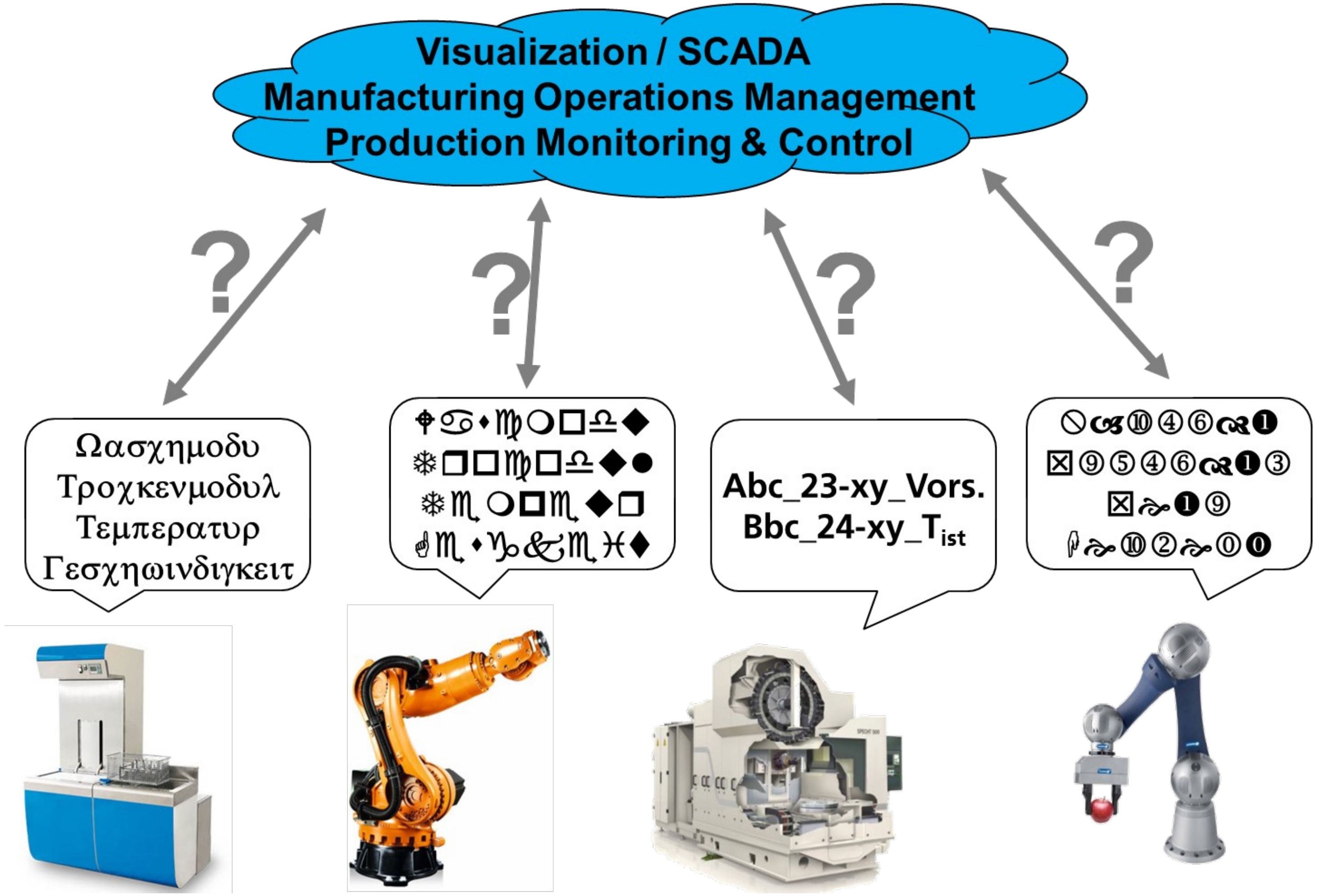
Activities (Excerpt)

- Scientific Director of the *Competence Center for Artificial Intelligence in Engineering* (CC-KING) (https://www.ki-engineering.eu/)
- Scientific Head of the *Karlsruhe Research Factory* (https://www.forschungsfabrik-ka.de)
- University Lecture at KIT Karlsruhe: *Methods of Convex Optimization for ML and Engineering*

Contact

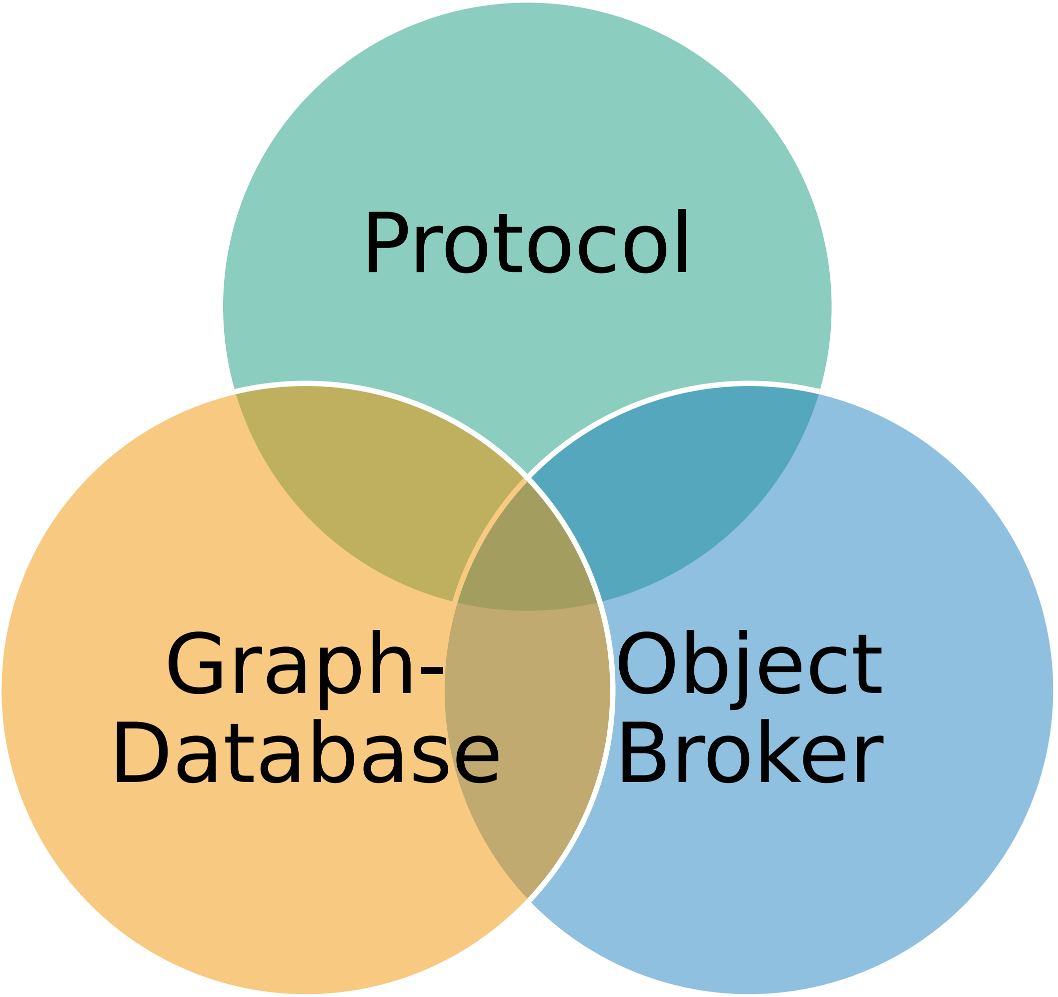
- julius.pfrommer@iosb.fraunhofer.de
- https://www.linkedin.com/in/juliuspfrommer

The Bottleneck of Industrial Communication



Visualization / SCADA
Manufacturing Operations Management
Production Monitoring & Control
?
?
?
?
Ωασχημοδυ Τροχκενμοδυλ Τεμπερατυρ Γεσχηωινδιγκειτ
⌖♋⬩♍❍□♎◆ ❄❐□♍□♎◆● ❄♏❍◘♏◆❒ ☝♏⬩♑&♏♓♦
Abc_23-xy_Vors. Bbc_24-xy_T_ist
⃠ꝏ⑩④⑥ꝏ❶ ☒⑨⑤④⑥ꝏ❶③ ☒❧❶⑨ ☛❧⑩②❧⓪⓿

Three Perspectives on OPC UA



Client-Server Protocol

- OPC UA defines a protocol for request/response message exchange
- Message Encoding: Binary, JSON, XML
- Transport Protocols: TCP/IP, Websockets, HTTP/S, (SOAP)
- TCP/IP + Binary Encoding is the most common transport mechanism

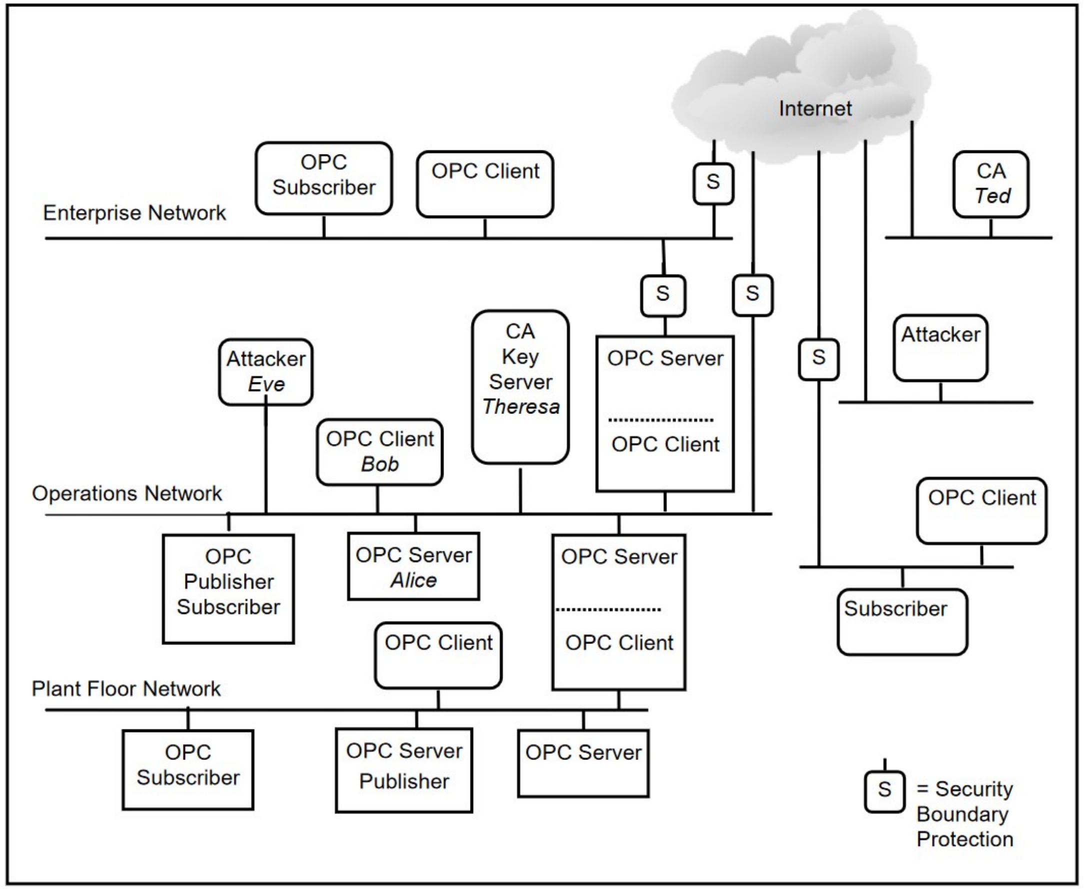
Object Broker

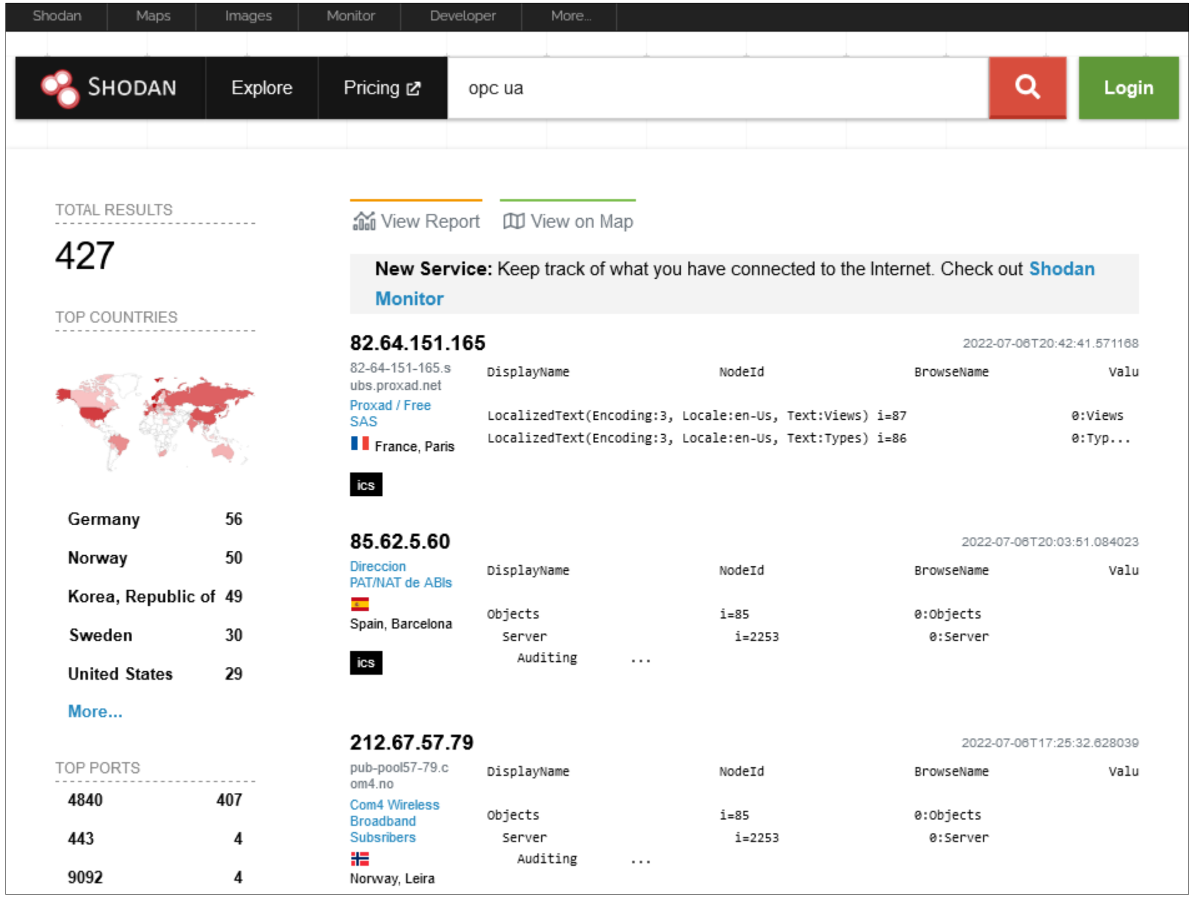
- Objects live in a server-side information model
- Dynamic changes to information model

Graph-Database

- Introspection of the information model
  in a graph with typed relations

Why would you connect your robot to the outside world?





TOTAL RESULTS

427

TOP COUNTRIES

Germany	56
Norway	50
Korea, Republic of	49
Sweden	30
United States	29

More...

TOP PORTS

4840	407
443	4
9092	4

New Service: Keep track of what you have connected to the Internet. Check out **Shodan Monitor**

82.64.151.165 2022-07-06T20:42:41.571168

82-64-151-165.subs.proxad.net
Proxad / Free SAS
France, Paris

```
DisplayName                                         NodeId              BrowseName     Valu

LocalizedText(Encoding:3, Locale:en-Us, Text:Views) i=87                               0:Views
LocalizedText(Encoding:3, Locale:en-Us, Text:Types) i=86                               0:Typ...
```

85.62.5.60 2022-07-06T20:03:51.084023

Direccion PAT/NAT de ABIs
Spain, Barcelona

```
DisplayName                NodeId              BrowseName           Valu

Objects                    i=85                0:Objects
  Server                     i=2253              0:Server
    Auditing       ...
```

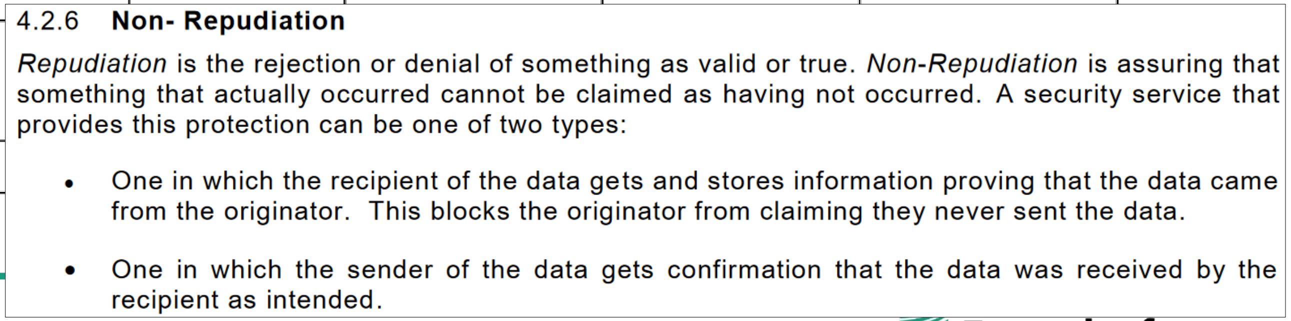
212.67.57.79 2022-07-06T17:25:32.628039

pub-pool57-79.com4.no
Com4 Wireless Broadband Subsribers
Norway, Leira

```
DisplayName                NodeId              BrowseName           Valu

Objects                    i=85                0:Objects
  Server                     i=2253              0:Server
    Auditing       ...
```

Security Objectives and Attacks [OPC UA Spec, Part 2]

Attacks	Authentication	Authorization	Confidentiality	Integrity	Auditability	Availability	Non-Repudiation
Denial of Service						X	
Eaves Dropping	X	X	X				
Message Spoofing		X					
Message Alteration	X	X		X	X		X
Message Replay	X	X					
Malformed Messages						X	
Server Profiling	(X)	(X)	(X)	(X)	(X)	(X)	(X)
System Hijacking	X	X	X	X	X	X	X
Rogue Server	X	X	X		X	X	
Compromising User Credentials	X	X	X				
Repudiation							



4.2.6 **Non- Repudiation**

Repudiation is the rejection or denial of something as valid or true. *Non-Repudiation* is assuring that something that actually occurred cannot be claimed as having not occurred. A security service that provides this protection can be one of two types:

- One in which the recipient of the data gets and stores information proving that the data came from the originator. This blocks the originator from claiming they never sent the data.
- One in which the sender of the data gets confirmation that the data was received by the recipient as intended.

OPC UA Security Architecture

TCP/IP

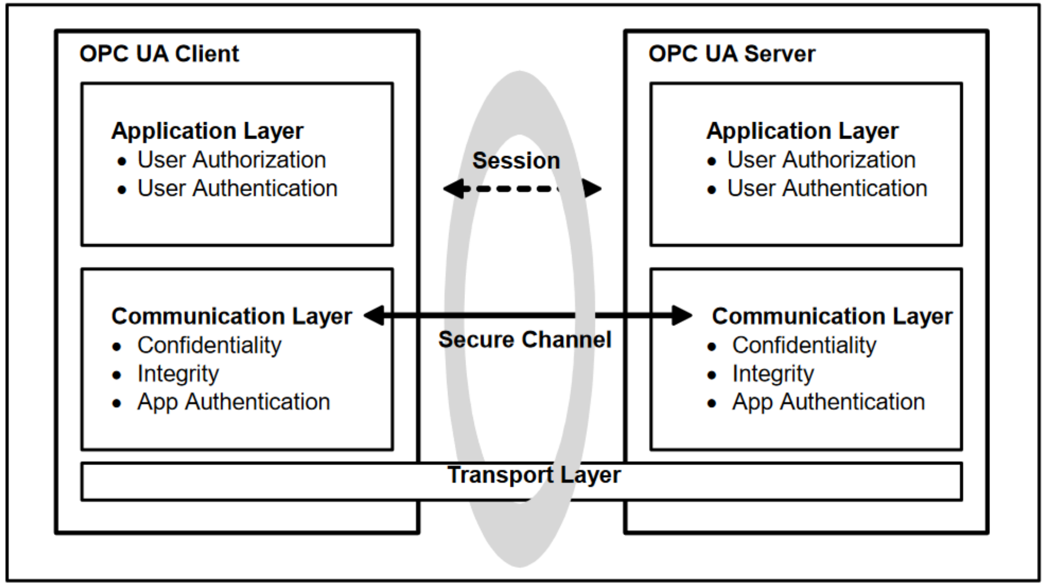
- Possibility to use Software-Defined Networking, VPN Tunnels, etc.

SecureChannel

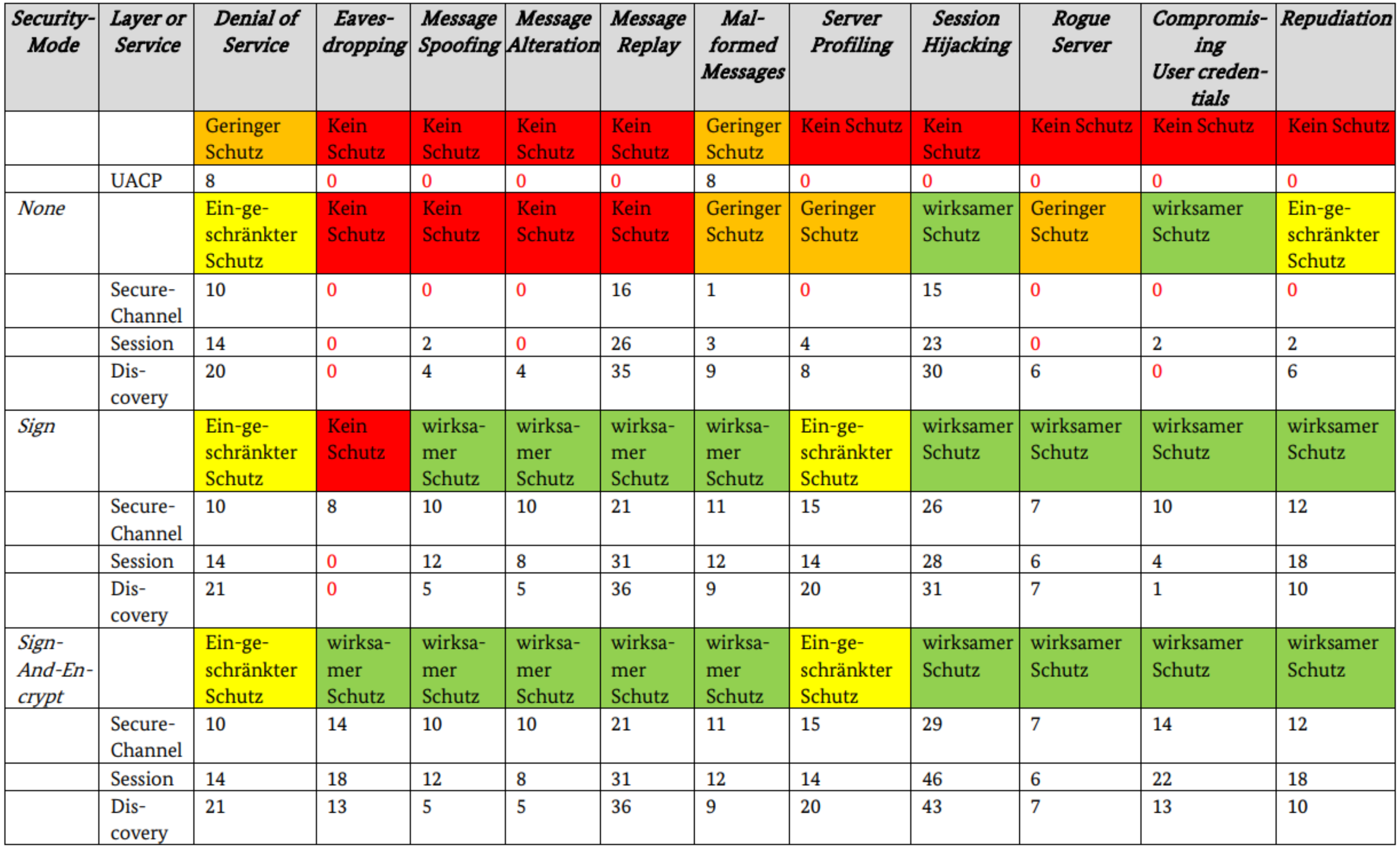
- Security Modes
  - None / Sign / Sign+Encrypt
- RSA for the handshake, AES at runtime
  - Profiles with crypto suites updated over time
  - ECC-based encryption upcoming
- Validation of x509 Certificates
  - Typical PKI backend similar to TLS

Session

- Different Authentication Mechanisms
  - Anonymous / Username+PW / Certificate
- Sessions are bound to a SecureChannel
- Sessions can switch to a new



Protocol Audit (BSI)

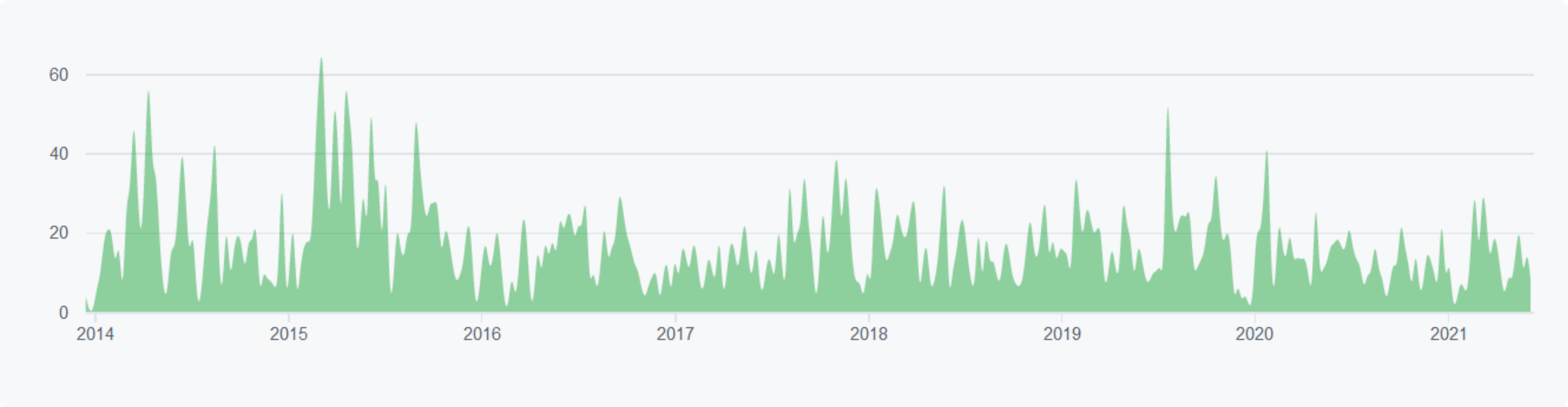


SecurityMode	*Layer or Service*	*Denial of Service*	*Eavesdropping*	*Message Spoofing*	*Message Alteration*	*Message Replay*	*Malformed Messages*	*Server Profiling*	*Session Hijacking*	*Rogue Server*	*Compromising User credentials*	*Repudiation*
		Geringer Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Geringer Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Kein Schutz
	UACP	8	0	0	0	0	8	0	0	0	0	0
None		Eingeschränkter Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Kein Schutz	Geringer Schutz	Geringer Schutz	wirksamer Schutz	Geringer Schutz	wirksamer Schutz	Eingeschränkter Schutz
	SecureChannel	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		Eingeschränkter Schutz	Kein Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz	Eingeschränkter Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz
	SecureChannel	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
SignAndEncrypt		Eingeschränkter Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz	Eingeschränkter Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz	wirksamer Schutz
	SecureChannel	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

https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/OPCUA/OPCUA_2022.pdf

The open62541 41 Open Source OPC UA SDK

- Open Source OPC UA SDK (Server / Client / PubSub)
- Written in platform-independent C
  - Linux, Windows, MacOS, Embedded, …
- Distributed as a open62541.c/.h file pair for easy integration
- License: MPLv2 (can be used in commercial projects)
- Large community, consistent development over time



We are doing everything wrong!

- Don‘t roll your own crypto
- Don‘t roll your own database
- Don‘t expose systems to the Internet
- Regularly update and maintain your deployed system
- Don‘t write software in C!

✳ Use processes and tools to ensure code quality

The origin of open62541

Picture: OPC UA Workshop & open62541 User Meeting (September 2015)

- Developed since late 2013
- Core maintainers from 4 German research institutes
- ~8,500 commits from >200 individual contributors

Fraunhofer IOSB

fortiss

ACPLT Lehrstuhl für Prozessleittechnik

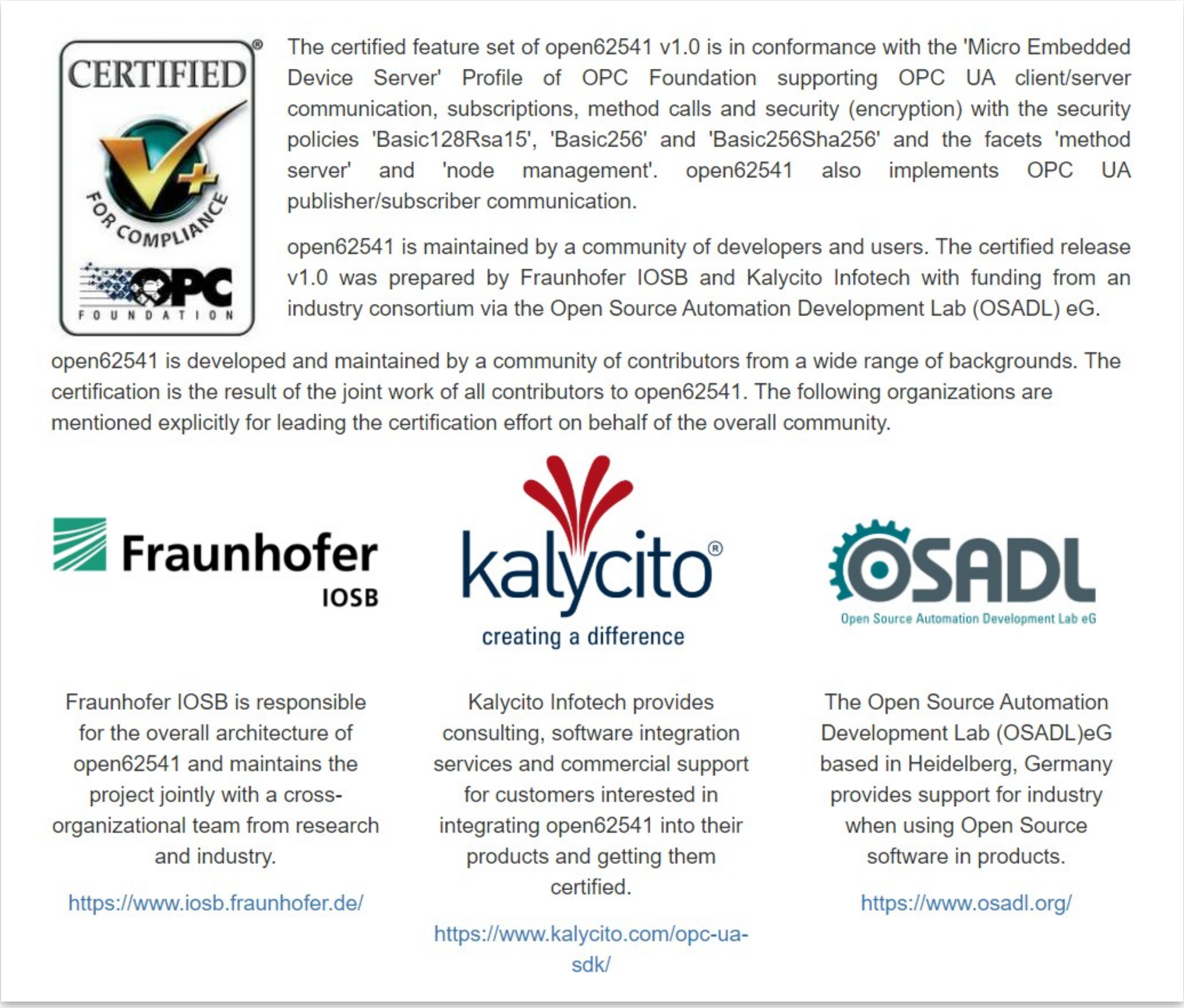
Professur für PROZESSLEITTECHNIK

kalycito®

basysKom

Support Partners

open62541 (example server) officially certified



The certified feature set of open62541 v1.0 is in conformance with the 'Micro Embedded Device Server' Profile of OPC Foundation supporting OPC UA client/server communication, subscriptions, method calls and security (encryption) with the security policies 'Basic128Rsa15', 'Basic256' and 'Basic256Sha256' and the facets 'method server' and 'node management'. open62541 also implements OPC UA publisher/subscriber communication.

open62541 is maintained by a community of developers and users. The certified release v1.0 was prepared by Fraunhofer IOSB and Kalycito Infotech with funding from an industry consortium via the Open Source Automation Development Lab (OSADL) eG.

open62541 is developed and maintained by a community of contributors from a wide range of backgrounds. The certification is the result of the joint work of all contributors to open62541. The following organizations are mentioned explicitly for leading the certification effort on behalf of the overall community.

Fraunhofer IOSB is responsible for the overall architecture of open62541 and maintains the project jointly with a cross-organizational team from research and industry.

https://www.iosb.fraunhofer.de/

Kalycito Infotech provides consulting, software integration services and commercial support for customers interested in integrating open62541 into their products and getting them certified.

https://www.kalycito.com/opc-ua-sdk/

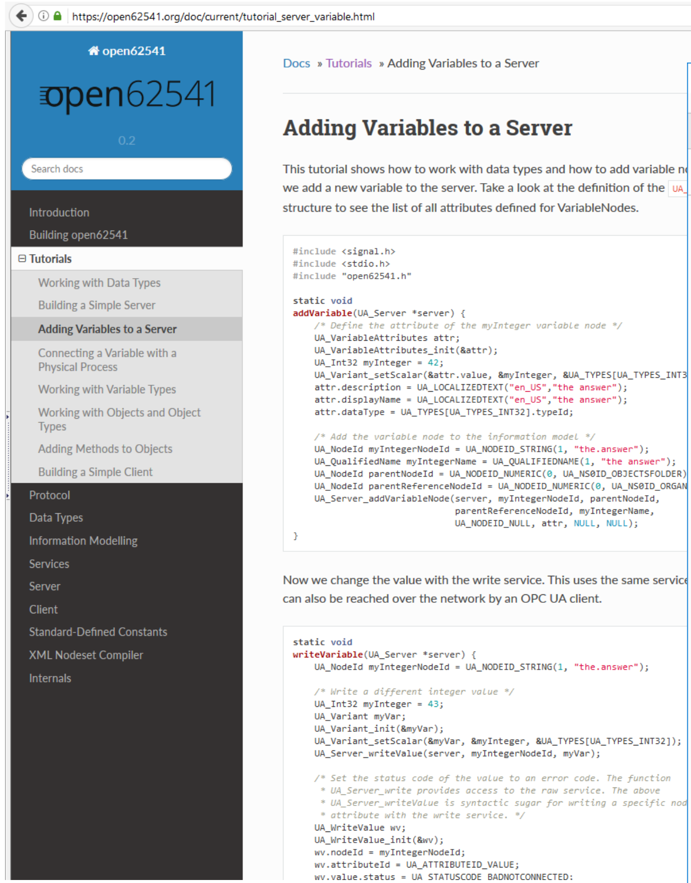
The Open Source Automation Development Lab (OSADL)eG based in Heidelberg, Germany provides support for industry when using Open Source software in products.

https://www.osadl.org/

CERTIFIED
FOR COMPLIANCE
OPC FOUNDATION
Fraunhofer IOSB
kalycito creating a difference
OSADL Open Source Automation Development Lab eG

- The example server from the v1.0 release was officially certified in 2019 by the OPC Foundation
- Hence, solutions based on that release are certifiable (not automatically certified)
- Certified Feature Set:
  - Micro Embedded Server
  - Encryption
  - Methods
  - Node Management
- Certification for the next set of profiles intended for 2022

Extensive Documentation (~250 pages PDF or HTML)



https://open62541.org/doc/current/tutorial_server_variable.html
open62541
open62541
0.2
Search docs
Introduction
Building open62541
⊟ Tutorials
Working with Data Types
Building a Simple Server
Adding Variables to a Server
Connecting a Variable with a Physical Process
Working with Variable Types
Working with Objects and Object Types
Adding Methods to Objects
Building a Simple Client
Protocol
Data Types
Information Modelling
Services
Server
Client
Standard-Defined Constants
XML Nodeset Compiler
Internals
Docs » Tutorials » Adding Variables to a Server

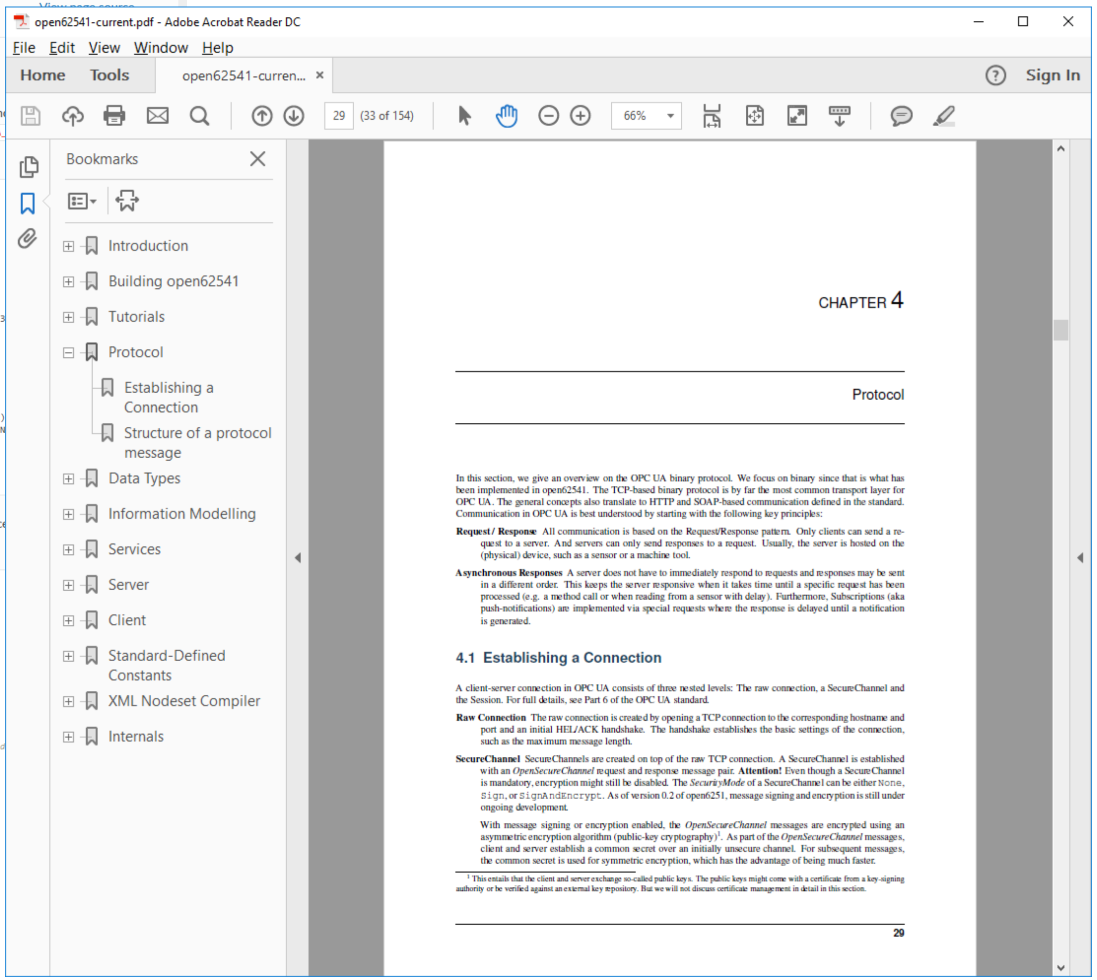
Adding Variables to a Server

This tutorial shows how to work with data types and how to add variable n
we add a new variable to the server. Take a look at the definition of the UA_
structure to see the list of all attributes defined for VariableNodes.

```
#include <signal.h>
#include <stdio.h>
#include "open62541.h"

static void
addVariable(UA_Server *server) {
    /* Define the attribute of the myInteger variable node */
    UA_VariableAttributes attr;
    UA_VariableAttributes_init(&attr);
    UA_Int32 myInteger = 42;
    UA_Variant_setScalar(&attr.value, &myInteger, &UA_TYPES[UA_TYPES_INT3
    attr.description = UA_LOCALIZEDTEXT("en_US","the answer");
    attr.displayName = UA_LOCALIZEDTEXT("en_US","the answer");
    attr.dataType = UA_TYPES[UA_TYPES_INT32].typeId;

    /* Add the variable node to the information model */
    UA_NodeId myIntegerNodeId = UA_NODEID_STRING(1, "the.answer");
    UA_QualifiedName myIntegerName = UA_QUALIFIEDNAME(1, "the answer");
    UA_NodeId parentNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_OBJECTSFOLDER)
    UA_NodeId parentReferenceNodeId = UA_NODEID_NUMERIC(0, UA_NS0ID_ORGAN
    UA_Server_addVariableNode(server, myIntegerNodeId, parentNodeId,
                              parentReferenceNodeId, myIntegerName,
                              UA_NODEID_NULL, attr, NULL, NULL);
}
```

Now we change the value with the write service. This uses the same servic
can also be reached over the network by an OPC UA client.

```
static void
writeVariable(UA_Server *server) {
    UA_NodeId myIntegerNodeId = UA_NODEID_STRING(1, "the.answer");

    /* Write a different integer value */
    UA_Int32 myInteger = 43;
    UA_Variant myVar;
    UA_Variant_init(&myVar);
    UA_Variant_setScalar(&myVar, &myInteger, &UA_TYPES[UA_TYPES_INT32]);
    UA_Server_writeValue(server, myIntegerNodeId, myVar);

    /* Set the status code of the value to an error code. The function
     * UA_Server_write provides access to the raw service. The above
     * UA_Server_writeValue is syntactic sugar for writing a specific nod
     * attribute with the write service. */
    UA_WriteValue wv;
    UA_WriteValue_init(&wv);
    wv.nodeId = myIntegerNodeId;
    wv.attributeId = UA_ATTRIBUTEID_VALUE;
    wv.value.status = UA_STATUSCODE_BADNOTCONNECTED;
```



open62541-current.pdf - Adobe Acrobat Reader DC
File Edit View Window Help
Home Tools open62541-curren... ×
Sign In
29 (33 of 154)
66%
Bookmarks
Introduction
Building open62541
Tutorials
Protocol
Establishing a Connection
Structure of a protocol message
Data Types
Information Modelling
Services
Server
Client
Standard-Defined Constants
XML Nodeset Compiler
Internals

CHAPTER 4

Protocol

In this section, we give an overview on the OPC UA binary protocol. We focus on binary since that is what has been implemented in open62541. The TCP-based binary protocol is by far the most common transport layer for OPC UA. The general concepts also translate to HTTP and SOAP-based communication defined in the standard. Communication in OPC UA is best understood by starting with the following key principles:

Request / Response All communication is based on the Request/Response pattern. Only clients can send a request to a server. And servers can only send responses to a request. Usually, the server is hosted on the (physical) device, such as a sensor or a machine tool.

Asynchronous Responses A server does not have to immediately respond to requests and responses may be sent in a different order. This keeps the server responsive when it takes time until a specific request has been processed (e.g. a method call or when reading from a sensor with delay). Furthermore, Subscriptions (aka push-notifications) are implemented via special requests where the response is delayed until a notification is generated.

4.1 Establishing a Connection

A client-server connection in OPC UA consists of three nested levels: The raw connection, a SecureChannel and the Session. For full details, see Part 6 of the OPC UA standard.

Raw Connection The raw connection is created by opening a TCP connection to the corresponding hostname and port and an initial HEL/ACK handshake. The handshake establishes the basic settings of the connection, such as the maximum message length.

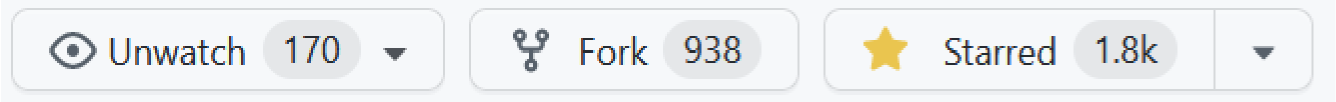
SecureChannel SecureChannels are created on top of the raw TCP connection. A SecureChannel is established with an *OpenSecureChannel* request and response message pair. **Attention!** Even though a SecureChannel is mandatory, encryption might still be disabled. The *SecurityMode* of a SecureChannel can be either `None`, `Sign`, or `SignAndEncrypt`. As of version 0.2 of open6251, message signing and encryption is still under ongoing development.

With message signing or encryption enabled, the *OpenSecureChannel* messages are encrypted using an asymmetric encryption algorithm (public-key cryptography)[1]. As part of the *OpenSecureChannel* messages, client and server establish a common secret over an initially unsecure channel. For subsequent messages, the common secret is used for symmetric encryption, which has the advantage of being much faster.

[1] This entails that the client and server exchange so-called public keys. The public keys might come with a certificate from a key-signing authority or be verified against an external key repository. But we will not discuss certificate management in detail in this section.

29

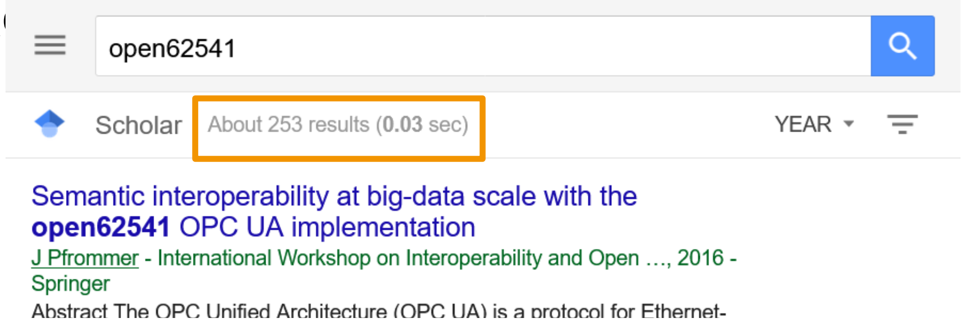
Usage of open62541



Prototyping and Product Development

- ~100k Downloads + git clones + Package Managers
- Commercial Support Partners
- BSI Survey 2021: *Which OPC UA stack / SDK is your product's OPC UA implementation based on?*
  ✳ 17.86% open62541

R



open62541

Scholar About 253 results (0.03 sec) YEAR

Semantic interoperability at big-data scale with the **open62541** OPC UA implementation
J Pfrommer - International Workshop on Interoperability and Open ..., 2016 - Springer
Abstract The OPC Unified Architecture (OPC UA) is a protocol for Ethernet-

Language Bindings

- Perl
- TCL
- C++
- Python (unreleased)
- Lua (unreleased)

Standardization

- OPC Foundation – FLC Prototyping
- umati

Large-Scale Physics Experiments

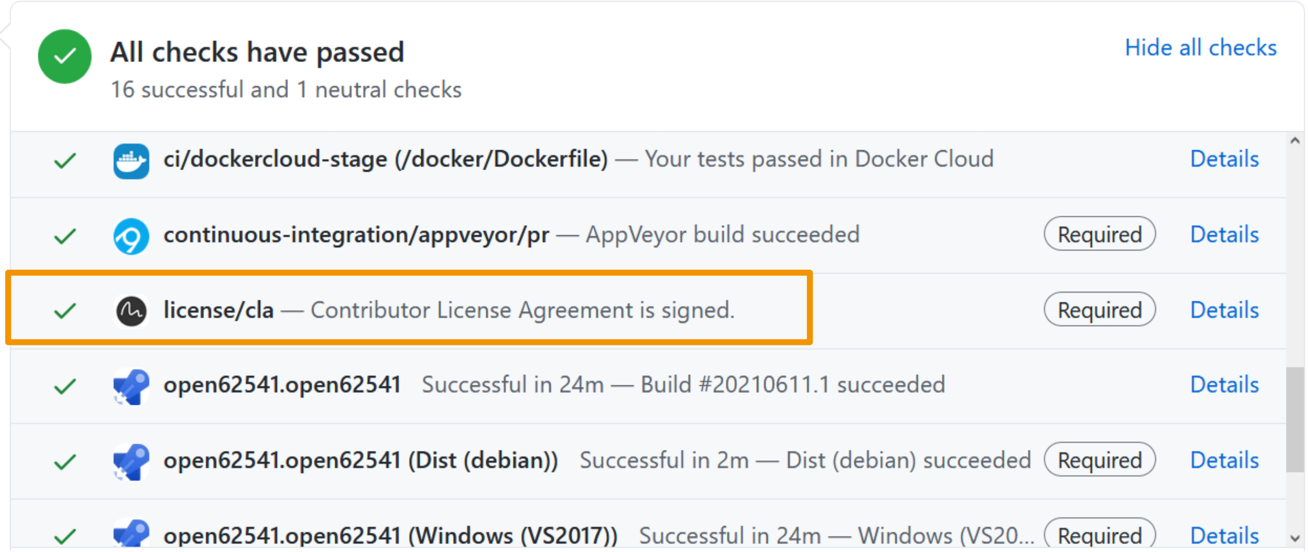
- Helmholtz ELBE
- CERN LHC

Particle Accelerators

- European Southern Observatory's Very Large Telescope

Community Contributions

- Outside contributions are highly welcome
- No Copyright Assignment Form or membership required to contribute code
  - Signing of the CLA required to assure legal backing of the contribution

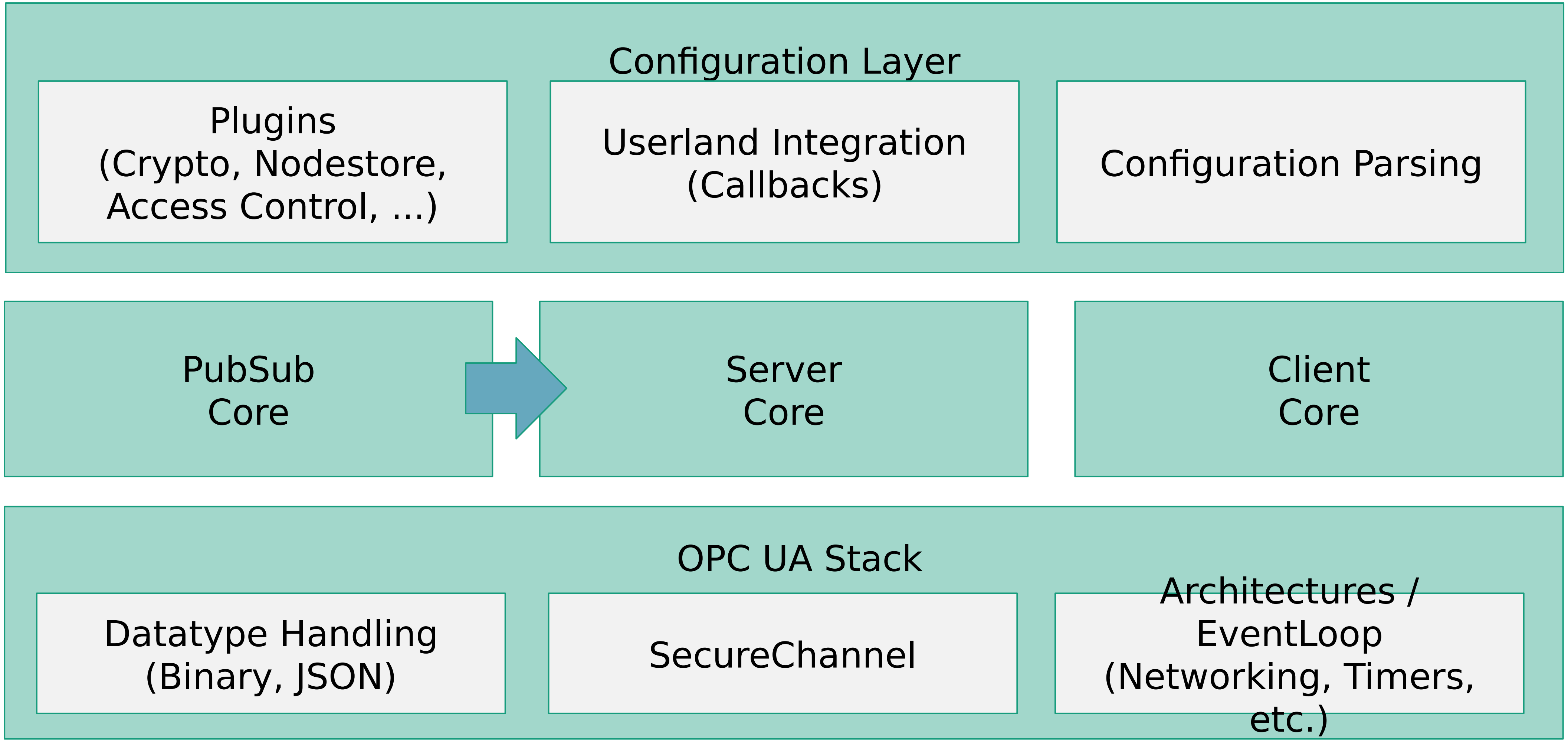


All checks have passed
16 successful and 1 neutral checks
Hide all checks

✓ ci/dockercloud-stage (/docker/Dockerfile) — Your tests passed in Docker Cloud — Details
✓ continuous-integration/appveyor/pr — AppVeyor build succeeded — Required — Details
✓ license/cla — Contributor License Agreement is signed. — Required — Details
✓ open62541.open62541 Successful in 24m — Build #20210611.1 succeeded — Details
✓ open62541.open62541 (Dist (debian)) Successful in 2m — Dist (debian) succeeded — Required — Details
✓ open62541.open62541 (Windows (VS2017)) Successful in 24m — Windows (VS20... — Required — Details

- Code reviews
  - Changing the code is easy. Changing the public API is hard.
  - Talk to us early about the API!
- Regular community conference calls to sync, align priorities and avoid double work
- Code Style & Commit Hygiene Guideline (CONTRIBUTING.md)

The Technical Architecture of open62541



Configuration Layer

Plugins (Crypto, Nodestore, Access Control, ...)

Userland Integration (Callbacks)

Configuration Parsing

PubSub Core

Server Core

Client Core

OPC UA Stack

Datatype Handling (Binary, JSON)

SecureChannel

Architectures / EventLoop (Networking, Timers, etc.)

Keeping open62541 lean and mean (w/o generated code, tools)

```
Statistics                   files          blank        comment           code
-------------------------------------------------------------------------------
/include/*                      21           1092           3370           4808
/src                            16           1461           1286           8904
/src/client                      6            626            388           3515
/src/server                     32           2888           3107          16446
/src/pubsub                     12           1214           1099           7762
-------------------------------------------------------------------------------
/plugins                        13            651            742           3861
/plugins/crypto/mbedtls          7            778            301           3350
/plugins/crypto/openssl          7            715            222           3583
-------------------------------------------------------------------------------
/tests                         117           6710           3272          33753
/examples                       77           2226           2704          13207
```

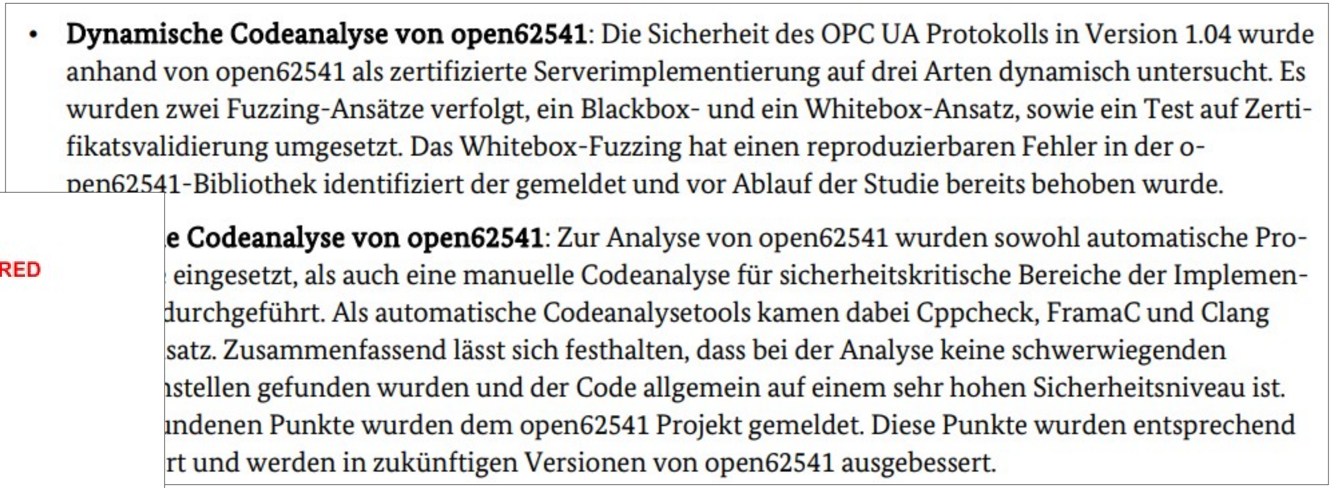
Code Quality Measures

- Every Pull Request has to pass the CI pipeline
- Unit and integration tests (80% coverage)
  - Compilers: GCC, Clang, TCC, MSVC 2008+,
    No warnings allowed
  - Compiles both as C and C++
  - Different standard libs:
    glibc, musl, MSVC CRT
  - Crypto: mbedTLS, OpenSSL
- Static code analysis: Clang Analyzer, Cppcheck
- Runtime sanitizers: Valgrind, Address Sanitizer,
  Memory Sanitizer, UB Sanitizer, …
- Fuzzing (Google oss-fuzz)

- Official Conformance Testing Tools
  - Provided by the OPC Foundation for corporate members
- Security audit perfomed as part of a BSI project

Code Audit Results

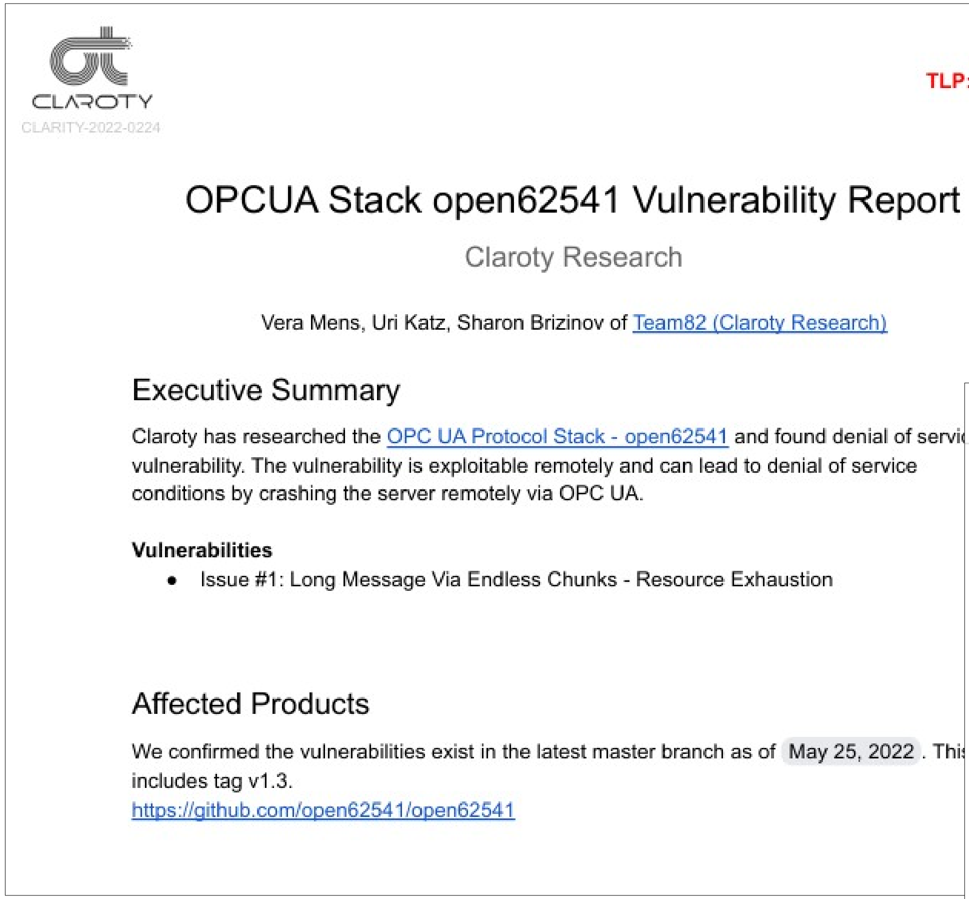
German Federal Office for Information Security (BSI)



- **Dynamische Codeanalyse von open62541**: Die Sicherheit des OPC UA Protokolls in Version 1.04 wurde anhand von open62541 als zertifizierte Serverimplementierung auf drei Arten dynamisch untersucht. Es wurden zwei Fuzzing-Ansätze verfolgt, ein Blackbox- und ein Whitebox-Ansatz, sowie ein Test auf Zertifikatsvalidierung umgesetzt. Das Whitebox-Fuzzing hat einen reproduzierbaren Fehler in der open62541-Bibliothek identifiziert der gemeldet und vor Ablauf der Studie bereits behoben wurde.

e Codeanalyse von open62541: Zur Analyse von open62541 wurden sowohl automatische Pro- eingesetzt, als auch eine manuelle Codeanalyse für sicherheitskritische Bereiche der Implemen- durchgeführt. Als automatische Codeanalysetools kamen dabei Cppcheck, FramaC und Clang satz. Zusammenfassend lässt sich festhalten, dass bei der Analyse keine schwerwiegenden nstellen gefunden wurden und der Code allgemein auf einem sehr hohen Sicherheitsniveau ist. ndenen Punkte wurden dem open62541 Projekt gemeldet. Diese Punkte wurden entsprechend rt und werden in zukünftigen Versionen von open62541 ausgebessert.

Claroty Research Responsible **Disclosure**



CLAROTY
CLARITY-2022-0224

TLP:RED

OPCUA Stack open62541 Vulnerability Report

Claroty Research

Vera Mens, Uri Katz, Sharon Brizinov of Team82 (Claroty Research)

Executive Summary

Claroty has researched the OPC UA Protocol Stack - open62541 and found denial of servi vulnerability. The vulnerability is exploitable remotely and can lead to denial of service conditions by crashing the server remotely via OPC UA.

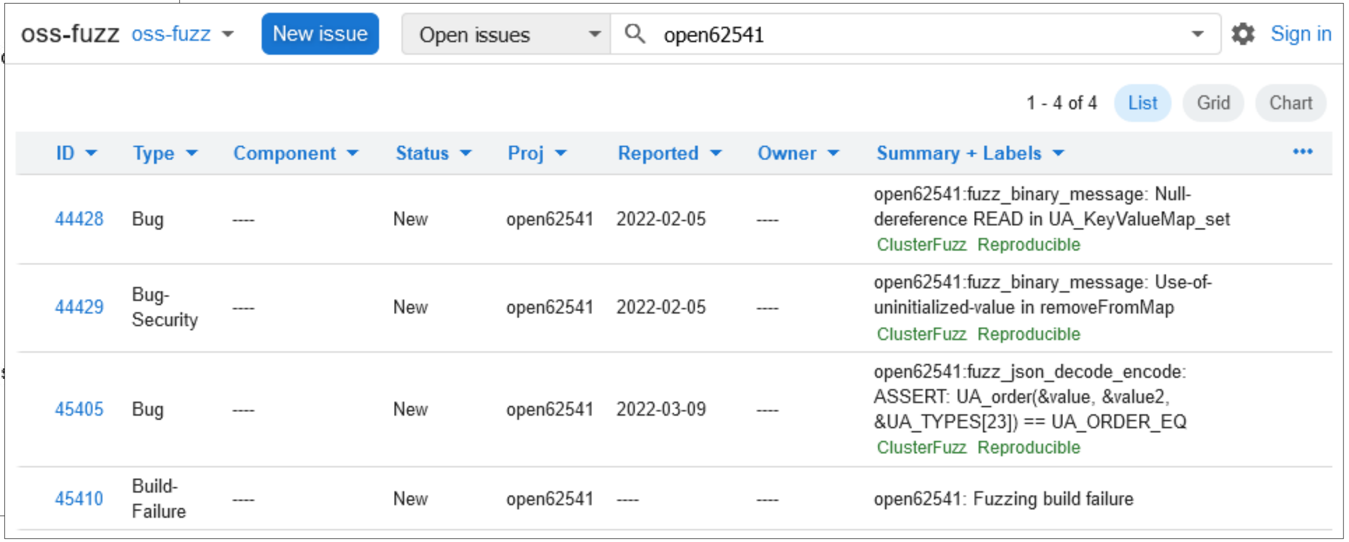
Vulnerabilities

- Issue #1: Long Message Via Endless Chunks - Resource Exhaustion

Affected Products

We confirmed the vulnerabilities exist in the latest master branch as of May 25, 2022 . Thi includes tag v1.3.
https://github.com/open62541/open62541

Automated Fuzzing Infrastructure



ID	Type	Component	Status	Proj	Reported	Owner	Summary + Labels
44428	Bug	----	New	open62541	2022-02-05	----	open62541:fuzz_binary_message: Null-dereference READ in UA_KeyValueMap_set ClusterFuzz Reproducible
44429	Bug-Security	----	New	open62541	2022-02-05	----	open62541:fuzz_binary_message: Use-of-uninitialized-value in removeFromMap ClusterFuzz Reproducible
45405	Bug	----	New	open62541	2022-03-09	----	open62541:fuzz_json_decode_encode: ASSERT: UA_order(&value, &value2, &UA_TYPES[23]) == UA_ORDER_EQ ClusterFuzz Reproducible
45410	Build-Failure	----	New	open62541	----	----	open62541: Fuzzing build failure

Bild kann bei Bedarf durch Klicken auf Symbol ausgetauscht werden

Contact

Dr. Julius Pfrommer
Head of the Research Group
„Cyberphysical Distributed Systems“
Phone +49 721 6091-286
Fax +49 721 6091-413
julius.pfrommer@iosb.fraunhofer.de

Thank you for the attention!