Open Geospatial Consortium, Inc.

Date: 2009-07-29

Reference number of this document: OGC 08-176**r1**

Version: 0.3.0

Category: Public Engineering Report

Editor: Andreas Matheus

OGC® OWS-6 Secure Sensor Web Engineering Report

Warning

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements.

Document type:	OpenGIS® Engineering Report
Document subtype:	NA
Document stage:	Approved for Public release
Document language:	English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable

Preface

The main purpose of this Engineering Report is to introduce standards-based security solutions for making the existing OGC Sensor Web Services, as described in the OWS-6 SWE baseline, ready towards the handling of sensors in the intelligence domain.

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

Forward

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

OWS-6 Testbed

OWS testbeds are part of OGC's Interoperability Program, a global, hands-on and collaborative prototyping program designed to rapidly develop, test and deliver Engineering Reports into OGC's Specification Program, where they are formalized for public release. In OGC's Interoperability Initiatives, international teams of technology providers work together to solve specific geoprocessing interoperability problems posed by the Initiative's sponsoring organizations. OGC Interoperability Initiatives include test beds, pilot projects, interoperability experiments and interoperability support services - all designed to encourage rapid development, testing, validation and adoption of OGC standards.

In April 2008, the OGC issued a call for sponsors for an OGC Web Services, Phase 6 (OWS-6) Testbed activity. The activity completed in June 2009. There is a series of on-line demonstrations available here:

http://www.opengeospatial.org/pub/www/ows6/index.html

The OWS-6 sponsors are organizations seeking open standards to address their urgent interoperability requirements. After analyzing their requirements, the OGC Interoperability Team recommended to the sponsors that the content of the OWS-6 initiative be organized around the following threads:

1. Sensor Web Enablement (SWE)
2. Geo Processing Workflow (GPW)
3. Aeronautical Information Management (AIM)
4. Decision Support Services (DSS)
5. Compliance Testing (CITE)

Additional background on these threads and the Request for Quotation / Call For Participation (RFQ/CFP) issued by OGC can be found at:

http://www.opengeospatial.org/projects/initiatives/ows-6.

The OWS-6 sponsoring organizations were:

- U.S. National Geospatial-Intelligence Agency (NGA)
- Joint Program Executive Office for Chemical and Biological Defense (JPEO-CBD)
- GeoConnections - Natural Resources Canada
- U.S. Federal Aviation Agency (FAA)
- EUROCONTROL
- EADS Defence and Communications Systems
- US Geological Survey

- Lockheed Martin
- BAE Systems
- ERDAS, Inc.

The OWS-6 participating organizations were:

52North, AM Consult, Carbon Project, Charles Roswell, Compusult, con terra, CubeWerx, ESRI, FedEx, Galdos, Geomatys, GIS.FCU, Taiwan, GMU CSISS, Hitachi Ltd., Hitachi Advanced Systems Corp, Hitachi Software Engineering Co., Ltd., iGSI, GmbH, interactive instruments, lat/lon, GmbH, LISAsoft, Luciad, Lufthansa, NOAA MDL, Northrop Grumman TASC, OSS Nokalva, PCAvionics, Snowflake, Spot Image/ESA/Spacebel, STFC, UK, UAB CREAF, Univ Bonn Karto, Univ Bonn IGG, Univ Bunderswehr, Univ Muenster IfGI, Vightel, and Yumetech.

Contents

Page

Figures

Page

Tables

Page

OGC® OWS-6 Secure Sensor Web Engineering Report

1 Introduction

1.1 Scope

The main purpose of this Engineering Report is to introduce standards-based security solutions for making the existing OGC Sensor Web Services, as described in the OWS-6 SWE baseline, ready towards the handling of sensors in the intelligence domain. This brings in the requirement for handling sensors that eventually produce classified information and the main objective of accreditation. In order to fulfill this, it would require a holistic security approach, but as this report is documenting the scientific findings under the OWS-6 initiative, it is limited to the given use case and its scenarios as well as the underlying architecture.

However, in order to proceed in this direction, this Engineering Report has the burden to identify a firm set of requirements first. This can be achieved with the objective "classified information" and the Trusted Computer System Evaluation Criteria (TCSEC). The TCSEC (also called The Orange Book) (see [63]) defines the evaluation class "B" for trusted systems that are certified to handle classified information. In particular, the following mandatory, and for this Engineering Report relevant, requirements are:

- Mandatory Access Right Management
- Authorization based on user identity and resource classification / user clearance
- Integrity of the classification labels and its protection against modification
- Tracking of actions

Providing solutions for the Sensor Web Services requires to take under consideration the distributed aspect of the system. This requires to extend the requirements from the TCSEC to reflect this. In addition, it is important to understand the difference between "distributed trusted systems" and a Service Oriented Architecture, as outlined in A SECURITY ARCHITECTURE FOR NET-CENTRIC ENTERPRISE SERVICES (NCES) (see [66]). In order to ensure that all security requirements are taken care off in an appropriate and holistic way, it is also important to define the Threat Model. For this report the Internet Threat Model, as defined in RFC 3552, is assumed. As outlined in the "Internet Threat Model", we have to assume an insecure network and the capabilities of an adversary to gain control over the communication and exercise different attacks towards espionage and sabotage. This requires to take additional and more specific requirements under consideration, as stated in (all parts of) ISO 10181, "SECURITY FRAMEWORKS FOR OPEN SYSTEMS". Basically, the distributed property of the Sensor Web System might not take affect, compared to a non-distributed system. But due to the distributed property, the implementation of requirements such as persistent protection of classified information needs to be ensured not only for a local system but

for multiple systems that are connected with each other over insecure communication channels. And even more complex for a Service Oriented Architecture, as it is the basis for the Sensor Web Services, the orchestration of services is dynamic which limits the applicability of network- or transport layer security.

In order to propose a secure sensor web, we also need to analyze the vulnerabilities and potential attacks that exist in the baseline and in the different ways of implementing the identified requirements. This will be done for the baseline Sensor Web Services and the proposed security standards. Because this analysis is so exhausting that the scope is limited to a given use case and its scenarios.

1.2 Non-goals

- Any aspects related to physical and operational computer safety
- How to establish federated identity management
- Configuration/administration of security services
- Multi national aspects
- Service recovery after an attack was exercised

1.3 Assumptions

For this document, it is assumed that OGC Sensor Web Services are deployed on secure and trusted systems, as defined in [63]. Therefore, no threats are discussed nor taken under considerations that result from an intrusion into these systems.

It is further assumed that each actor in the Sensor Web (either a person, a sensor or a service) is uniquely identified.

1.4 Relation to other documents in the OWS-6 initiative

The Engineering Report “OGC Web Services Security” is a work item of the GPW thread in OWS-6. It addresses security aspects for all OGC Web Services. As the Sensor Web includes services that supersede the services from the GPW thread, this document is a complement to that report.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

1.5 Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Andreas Matheus	AM Consult
Thomas Everding	Institute for Geoinformatics, University of Muenster
Ingo Simonis	Geospatial Research

1.6 Revision history

Date	Release	Editor	Primary clauses modified	Description
2008-12-11	0.1.0	Andreas Matheus	All	Creation
2009-09-03	0.2.0	Andreas Matheus	9.4, 9.5, 9.6, 9.7	Threats for SWE services Vulnerabilities and attack analysis
2009-19-03	0.3.0	Andreas Matheus Thomas Everding	9.8 12	Rating the attacks Event pattern based communication and firewalls
2009-09-04	0.4.0	Andreas Matheus	9.5, 9.6, 9.7 all	Corrects Editorial issues
2009-27-05	0.5.0	Andreas Matheus	13	Recommendation added
2009-10-06	0.6.0	Andreas Matheus	9	Comments by Ingo Simonis incorporated
2009-15-06	0.7.0	Andreas Matheus & Ingo Simonis	9	SWE Services operations analysis clarification
2009-10-09	0.3.0	Carl Reed	Various	Prepare document for public release

1.7 Future work

Improvements in this document by OGC Sensor Web experts are strongly recommended prior to voting on the publication of this Engineering Report (see Preface note in red).

It is recommended to focus (i) on sanity checking of the writing and (ii) on providing example attacks that are related to the provided CCSI use case.

It is also recommend that a short executive summary be written highlighting the possible vulnerabilities and attacks on the baseline and introduce a mitigating or prevention strategy.

For the next OWS, the author recommends to pick one Sensor Web Service, e.g. the SPS, and practice the findings from this ER towards the implementation of a secure SPS.

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

[1] ISO 10181 (all parts), *DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS - INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – SECURITY FRAMEWORKS FOR OPEN SYSTEMS*, 1996

[2] **IPSec:** IP Security – IETF RFC 4301 (2005) (soboletes RFC 2401 from1998): http://tools.ietf.org/html/rfc4301

[3] **TLS:** Transport Layer Security – IETF RFC 2246 (1999): http://tools.ietf.org/html/rfc2246

[4] HTTP/HTTPS

[5] **Web Services Security**: SOAP Message Security 1.1 (WS-Security 2004) – OASIS Standard Specification, 1 February 2006: http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[6] SOAP

[7] **XML Digital Signature:** XML-Signature Syntax and Processing – W3C Recommendation 12 February 2002: http://www.w3.org/TR/xmldsig-core/

[8] **XML Encryption**: XML Encryption Syntax and Processing – W3C Recommendation 10 December 2002: http://www.w3.org/TR/xmlenc-core/

[9] **XKMS**: XML Key Management Specification (XKMS) – W3C Note 30 March 2001: http://www.w3.org/TR/xkms/

[10] **SAML**: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[11] **SAML**-Bindings: Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005: http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf

[12] **SAML**-Profiles: Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[13] **HTTPS:** HTTP Over TLS – IETF RFC 2818 (2000): http://tools.ietf.org/html/rfc2818

[14] **X.509 / PKI**: Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks, ITU-T Standard, 08/2005: http://www.ietf.org/html.charters/pkix-charter.html

[15] **CRL:** Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile – IETF RFC 3280: http://tools.ietf.org/html/rfc3280

[16] **Kerberos:** The Kerberos Network Authentication Service (V5) – IETF RFC 4120 (2005) obsoletes 1510 (1993): http://tools.ietf.org/html/rfc4120

[17] **LDAP:** Lightweight Directory Access Protocol (LDAP): The Protocol – IETF RFC 4511 (2006): http://tools.ietf.org/html/rfc4511
[18] **XCBF**: XML Common Biometric Format, OASIS Standard, August 2003: http://www.oasis-open.org/committees/download.php/3353/oasis-200305-xcbf-specification-1.1.doc
[19] **XACML:** eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS Standard, 1 Feb 2005: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
[20] **XACML RBAC Profile:** Core and hierarchical role based access control (RBAC) profile of XACML v2.0, OASIS Standard, 1 February 2005: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf
[21] **XACML SAML Profile**: SAML 2.0 profile of XACML v2.0, OASIS Standard, 1 February 2005: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf
[22] **XACML DSIG Profile**: XML Digital Signature profile of XACML v2.0, OASIS Standard, 1 February 2005: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-dsig-profile-spec-os.pdf
[23] **GeoXACML:** Geospatial eXtensible Access Control Markup Language (GeoXACML) v1.0, Open Geospatial Consortium, Inc., 2008/02/20: http://portal.opengeospatial.org/files/?artifact_id=25218
[24] **GeoXACML Extension A:** Geospatial eXtensible Access Control Markup Language (GeoXACML) Extension A – GML2 Encoding Version 1.0: http://portal.opengeospatial.org/files/?artifact_id=25219
[25] **GeoXACML Extension B**: Geospatial eXtensible Access Control Markup Language (GeoXACML) Extension B – GML3 Encoding Version 1.0: http://portal.opengeospatial.org/files/?artifact_id=25220
[26] **XrML:** XrML - eXtensible rights Markup Language, ContentGuard: http://www.xrml.org/
[27] **REL**: Information technology -- Multimedia framework (MPEG-21) -- Part 5: Rights Expression Language, ISO/IEC 21000-5:2004: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=36095
[28] **ODRL**: Open Digital Rights Language (ODRL) Version 1.1, W3C Note, 19 September 2002: http://www.w3.org/TR/odrl/
[29] **SOAP**: Simple Object Access Protocol (SOAP), W3C Recommendation (Second Edition) 27 April 2007: http://www.w3.org/TR/soap/
[30] **WSDL**: Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001: http://www.w3.org/TR/wsdl
[31] **WS-Addressing:** Web Services Addressing 1.0 – Core W3C Recommendation 9 May 2006: http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/
(This Recommendation supersedes WS-Routing and WS-Referral as proposed by Microsoft in 2001)
[32] **WS-PAEPR**: Web Services Policy Attachment for Endpoint Reference (WS-PAEPR), W3C Member Submission 20 July 2007: http://www.w3.org/Submission/WS-PAEPR/

[33] **WS-Policy**: Web Services Policy 1.5 – Framework, W3C Recommendation 04 September 2007: http://www.w3.org/TR/2007/REC-ws-policy-20070904/
[34] **WS-Policy Attachment**: Web Services Policy 1.5 – Attachment, W3C Recommendation, 04 September 2007: http://www.w3.org/TR/ws-policy-attach/
[35] **WS-SecurityPolicy:** WS-SecurityPolicy 1.2, OASIS Standard, 1 July 2007: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf
[36] **WS-Trust:** WS-Trust 1.3, OASIS Standard, 19 March 2007: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf
[37] **WS-SecureConversation:** WS-SecureConversation 1.3, OASIS Standard, 1 March 2007: http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf
[38] **WS-Reliable Messaging:** Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.2, Committee Draft, 28 February 2008: http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.2-spec-cd-01.pdf
[39] **WS-RM Policy**: Web Services Reliable Messaging Policy Assertion (WS-RM Policy) Version 1.2, Committee Draft, 28 February 2008: http://docs.oasis-open.org/ws-rx/wsrmp/200702/wsrmp-1.2-spec-cd-01.pdf
[40] **WS-MakeConnection:** Web Services Make Connection (WS-MakeConnection) Version 1.1, Committee Draft, 28 February 2008: http://docs.oasis-open.org/ws-rx/wsmc/200702/wsmc-1.1-spec-cd-01.pdf
[41] **WS-Federation / WS-Authorization / WS-Privacy:** Web Services Federation Language (WS-Federation) Version 1.2, Editors Draft – 06, May 21, 2008: http://www.oasis-open.org/committees/download.php/28360/ws-federation-1.2-spec-ed-06.doc
[42] **WS-MetadataExchange:** Web Services Metadata Exchange (WS-MetadataExchange), Version 1.1, August 2006, Microsoft, IBM, Sun and SAP: http://specs.xmlsoap.org/ws/2004/09/mex/WS-MetadataExchange.pdf
[43] **WS-Transfer:** Web Services Transfer (WS-Transfer), W3C Member Submission, 27 September 2006: http://www.w3.org/Submission/WS-Transfer/
[44] **WS-RT:** Web Services Resource Transfer (WS-RT), Version 1.0, August 2006: http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer/WS-ResourceTransfer.pdf
[45] **ISO/TS 15000-1:** Electronic business eXtensible Markup Language (ebXML) -- Part 1: Collaboration-protocol profile and agreement specification (ebCPP), ISO 2004: http://www.iso.org/iso/catalogue_detail?csnumber=39972
[46] **ISO/TS 15000-2**: Electronic business eXtensible Markup Language (ebXML) -- Part 2: Message service specification (ebMS), ISO 2004: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39973
[47] **ISO/TS 15000-3**: Electronic business eXtensible Markup Language (ebXML) – Part 3: Registry information model specification (ebRIM), ISO 2004: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39974
[48] **ISO/TS 15000-4**: Electronic business eXtensible Markup Language (ebXML) – Part 4: Registry services specification (ebRS), ISO 2004:

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39975

[49] **ISO/TS 15000-5**: Electronic business eXtensible Markup Language (ebXML) – Part 5: ebXML Core Components Technical Specification, Version 2.01(ebCCTS), ISO 2005: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=41022

[50] **ISO/IEC 15408-1:** Information technology — Security techniques — Evaluation criteria for IT security —Part 1: Introduction and general model, ISO 2005: http://standards.iso.org/ittf/PubliclyAvailableStandards/c040612_ISO_IEC_15408-1_2005(E).zip

[51] **ISO/IEC 15408-2:** Information technology — Security techniques — Evaluation criteria for IT security —Part 2: Security functional requirements, ISO 2005: http://standards.iso.org/ittf/PubliclyAvailableStandards/c040613_ISO_IEC_15408-2_2005(E).zip

[52] **ISO/IEC 15408-3:** Information technology — Security techniques — Evaluation criteria for IT security —Part 3: Security assurance requirements, ISO 2005: http://standards.iso.org/ittf/PubliclyAvailableStandards/c040614_ISO_IEC_15408-3_2005(E).zip

[53] **ISO/IEC 15443-1**: Information technology - Security techniques - A framework for IT security assurance - Part 1: Overview and framework, ISO 2005: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39733

[54] **ISO/IEC 15443-2**: Information technology - Security techniques - A framework for IT security assurance - Part 2: Assurance methods, ISO 2005: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39271

[55] **ISO/IEC 15443-3**: Information technology - Security techniques - A framework for IT security assurance - Part 3: Analysis of assurance methods, ISO 2007: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=41693

[56] **ISO/IEC 10181-1:** Information technology -- Open Systems Interconnection -- Security frameworks for open systems: Overview, ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24404

[57] **ISO/IEC 10181-2:** Information technology -- Open Systems Interconnection -- Security frameworks for open systems: Authentication framework, ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=18198

[58] **ISO/IEC 10181-3:** Information technology -- Open Systems Interconnection -- Security frameworks for open systems: Access control framework, ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=18199

[59] **ISO/IEC 10181-4:** Information technology -- Open Systems Interconnection -- Security frameworks for open systems: Non-repudiation framework, ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=23615

[60] **ISO/IEC 10181-5:** Information technology -- Open Systems Interconnection -- Security frameworks for open systems: Confidentiality framework, ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24329

[61] **ISO/IEC 10181-6:** Information technology -- Open Systems Interconnection -- Security frameworks for open systems: Integrity framework, ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24330

[62] **ISO/IEC 10181-7:** Information technology -- Open Systems Interconnection -- Security frameworks for open systems: Security audit and alarms framework, ISO 1996: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=18200

[63] United States Government Department of the Defense, TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA, 1985

[64] OGC 06-121r3, *OpenGIS® Web Services Common Specification*, 2006

[65] OGC 02-112, *OpenGIS® Service Architecture*, 2002

[66] A SECURITY ARCHITECTURE FOR NET-CENTRIC ENTERPRISE SERVICES (NCES)

[67] ITU, X.800, *Data Communication Networks; Open Systems Interconnection (OSI); Se-. curity Structure and Application — Security Architecture for Open Systems*, 1991

[68] IETF, RFC3552, *Guidelines for Writing RFC Text on Security Considerations*, 2003: http://tools.ietf.org/html/rfc3552

[69] [1] SWE High Level Architecture" (OGC #07-165)

[70] [2] Sensor Web Enablement (ISO TC211 http://www.isotc211.org/Outreach/Newsletter/Newsletter_07_2005/TC_211_Newsletter_07.pdf)

[71] [3] Web Services Architecture Usage Scenarios (W3C http://www.w3.org/TR/ws-arch-scenarios/)

[72]

NOTE This OWS Common Specification contains a list of normative references that are also applicable to this Implementation Specification.

3 Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Specification [OGC 06-121r3] and in OpenGIS® Abstract Specification Topic 12: The OpenGIS Service Architecture [OGC 02-112] shall apply. In addition, the following terms and definitions apply.

3.1
Authentication
"The provision of assurance of the claimed identity of an entity."[1]

Access Control
"The provision of assurance of the claimed identity of an entity." [1]

3.2
Data Integrity
"The property that data has not been altered or destroyed in an unauthorized manner" [1]

3.3
Confidentiality
"The property that information is not made available or disclosed to unauthorized individuals, entities, or processes." [1]

3.4
Availability
"The property of being accessible and useable upon demand by an authorized entity." [1]

3.5
Repudiation
"Denial by one of the entities involved in a communication of having participated in all or part of the communication." [1]

3.6
principal
"An entity whose identity can be authenticated." [1]

3.7
security domain
"A set of elements, a security policy, a security authority and a set of security-relevant activities in which the set of elements are subject to the security policy for the specified activities, and the security policy is administered by the security authority for the security domain." [1]

3.8
security token
"A set of data protected by one or more security services, together with security information used in the provision of those security services, that is transferred between communicating entities." [1]

3.9
trust
"Entity X is said to trust entity Y for a set of activities if and only if entity X relies upon entity Y behaving in a particular way with respect to the activities." [1]

3.10
verifier
"An entity which is or represents the entity requiring an authenticated identity. A verifier includes the functions necessary for engaging in authentication exchanges." [1]

4 Conventions

4.1 Abbreviated terms

ACI	Access Control Information
AI	Authentication Information
AIM	Aeronautical Information Management
CSW	Catalog Service for the Web
DMZ	Demilitarized Zone
EDA	Event Driven Architecture
IP	Internet Protocol
NAT	Network Address Translation
OWS-6	Open Web Services, Phase 6
SAS	Sensor Alert Service
SOS	Sensor Alert Service
SI	Security Information
SPS	Sensor Planning Service
SWE	Sensor Web Enablement
WNS	Web Notification Service
XMPP	Extensible Messaging and Presence Protocol

4.2 UML notation

Sequence diagrams that appear in this Engineering Report are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 06-121r3].

4.3 Used parts of other documents

This document uses significant parts of document [OGC 06-121r3]. To reduce the need to refer to that document, this document copies some of those parts with small modifications. To indicate those parts to readers of this document, the largely copied parts are shown with a light grey background (15%).

5 Use case “Fire at the airport”

As it seems to be impossible to describe a standards based approaches for the implementation of these security frameworks in a general way, this Engineering Report focuses on a particular use case in which it is possible to register sensors with a catalog and task the sensors to fulfill a certain function.

The use case for this ER is based on the use case “fire at the airport” as described in the GPW thread of the OWS-6 initiative. But, it is slightly modified to accommodate the use of CCSI sensors and to become specific to the SWE thread. Further on, the use case was extended by incorporating a unattended aerial vehicle (UAV) in order to elaborate additional security relevant aspects.

5.1 Use Case Description

In the first responders dispatch office (FRDO), administrators connect different CCSI sensors to different Sensor Web Services to allow monitoring of fire alerts at an airport. After the sensor registration is completed, alert conditions are configured. In order to have fire alerts be reported to specific operators of the FRDO, the operator subscribes for the (pre-configured) fire alert events. During runtime of the system, fire alerts will be published to the operators.

After an operator received a fire alert, they have the responsibility to dispatch fire fighters and gather further information, relevant to support the fire fighters. In this respect, the operators can task UAVs to take pictures of the scene that can be obtained and used by the firefighters to understand the overall scene better. They can also obtain additional sensor observations, for example of plume sensors, to get additional scene information. For this scenario, it is assumed that imagery produced by UAVs and the additional observations are classified. Therefore, operators need a particular clearance to see that information.

5.2 Use Case Architecture

For this use case, different security domains (SD) exist:

- FR-SD – The First Responder security domain consists of mobile clients operated by fire fighters and first responder personnel out in the field.
- FRDO-SD – The First Responder Dispatch Office security domain consists of Sensor Web Services and clients that allow receiving alerts from connected fire detection sensors.
- AA-SD – The Airport Authority security domain consists of sensors that report about measuring of air pollution locations on site (plume sensor). The AA-SD provides classified sensor alerts through service interfaces that can only be used by other security domains under certain conditions.
- FL-SD – The Federal Level security domain provides secure Sensor Web Services for tasking UAVs. The FL-SD provides classified imagery through

service interfaces that can only be used by other security domains under certain conditions.

5.3 Use Case Scenarios

This use case described above can be separated into different scenarios, related to the Publish-Find-Bind pattern, as it exists for Web Services in general:

(i) Sensor Registration – Registering available fire detection sensors and their capabilities/characteristics in a CSW,

(ii) Sensor Find/Bind – Finding sensors in a CSW and connect (register) them with appropriate Sensor Web Services,

(iii) Sensor Web Services Find/Bind – Setting up and configuring instances of Sensor Web Services, to

a. receive fire alerts,

b. obtain air pollution, and

c. task UAV sensors

(iv) Sensor Web Services Processing – Running the instance of the sensor web to

a. receive fire alerts

b. obtain air pollution, and

c. task UAV sensors and obtain produced imagery.

5.3.1 Sensor Register

For each security domain, in which sensors exist, an administrator uses a catalog service (CSW) to register available sensors and their capabilities to allow binding to the sensors. For this use case it is assumed that the CSW is deployed in the security domain of the administrator and is accessible for search operations by “configuration” administrators located in the same security domain.

5.3.2 Sensor Find/Bind

Interactions during this scenario focus on the registration of particular sensor instances to Sensor Web Services provided by the security domain. In order to achieve this, the setup administrator (potentially different from the administrator of Register scenario) uses the search operation of the CSW to find applicable sensor instances. After finding a sensor, the administrator will register the sensor with the appropriate service. The type of service depends on the capabilities of the sensor. For example, a temperature or smoke detection sensor would be registered with the SPS and the SAS; the SPS can be used to configure the sensor and the SAS takes care of operator subscription and alert notifications. The

UAV sensors are only registered with a SPS to allow tasking. In order to obtain the produced imagery, a SOS is used.

After the administrator has registered all relevant sensors with the appropriate Sensor Web Services, the setting up of the sensor web instance can begin.

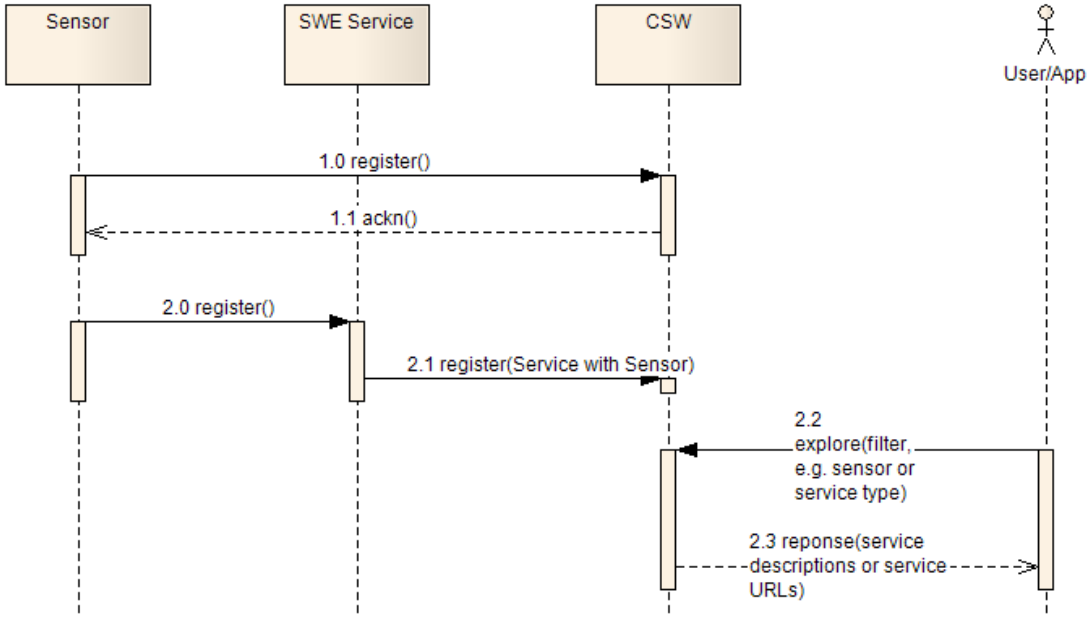

Figure 1: Registration of sensors with a CSW and a Sensor Web Service

5.3.3 Sensor Web Serivces Find/Bind

During the interactions in this scenario, an administrator searches available catalog services to find applicable services for configuring a sensor web to fulfill particular needs. For the airport fire use case, fire detection sensors and smoke sensors are required. The administrator would link the associated Sensor Planning Services (SPS) and Sensor Alert Services (SAS) into the own portal to enable

- sensor configuration and
- subscription for fire (and smoke) alerts for operators.

The administrator would further configure sensors via specific interfaces provided by the SPS to accommodate the specific needs. For example, the administrator would set the threshold for the smoke detectors to reflect local environment specific aspects.

5.3.4 Processing

The actors in this scenario are the operators at the FRDO-SD, which are the consumer of the information that the instance of a sensor web is producing. Operators with specific

clearance will receive classified alerts and operators with specific rights will be able to task available UAV sensors via the SPS interface. They will also have access to different SOS providing air pollution information and the imagery production of a UAV.

For the purpose of discussing security aspects in a later section of the ER, two different communication patterns can be identified within this scenario:

- The request/response communication pattern that is used by operators to configure and task sensors via a SPS or to subscribe for alerts at a SAS
- The notification communication pattern, where an SAS is broadcasting alerts to all subscribed users. If the SPS EO profile is used, it will also notify operators after certain operations completed.

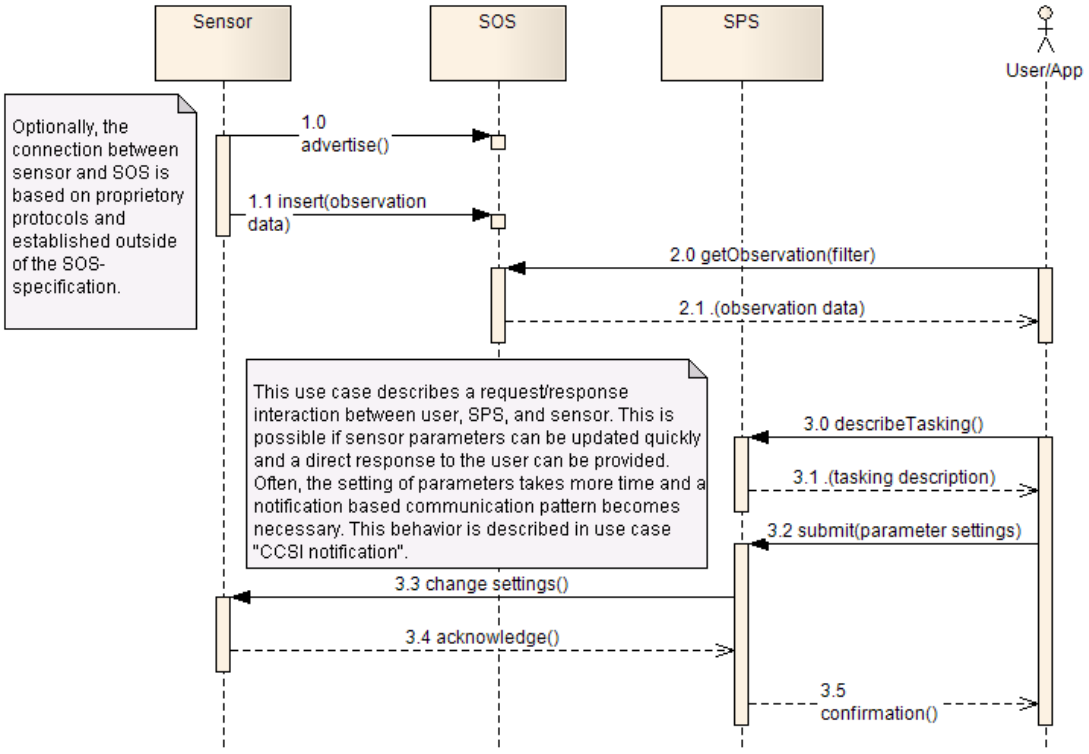

Figure 2: Request/Response Communication initiated by the user

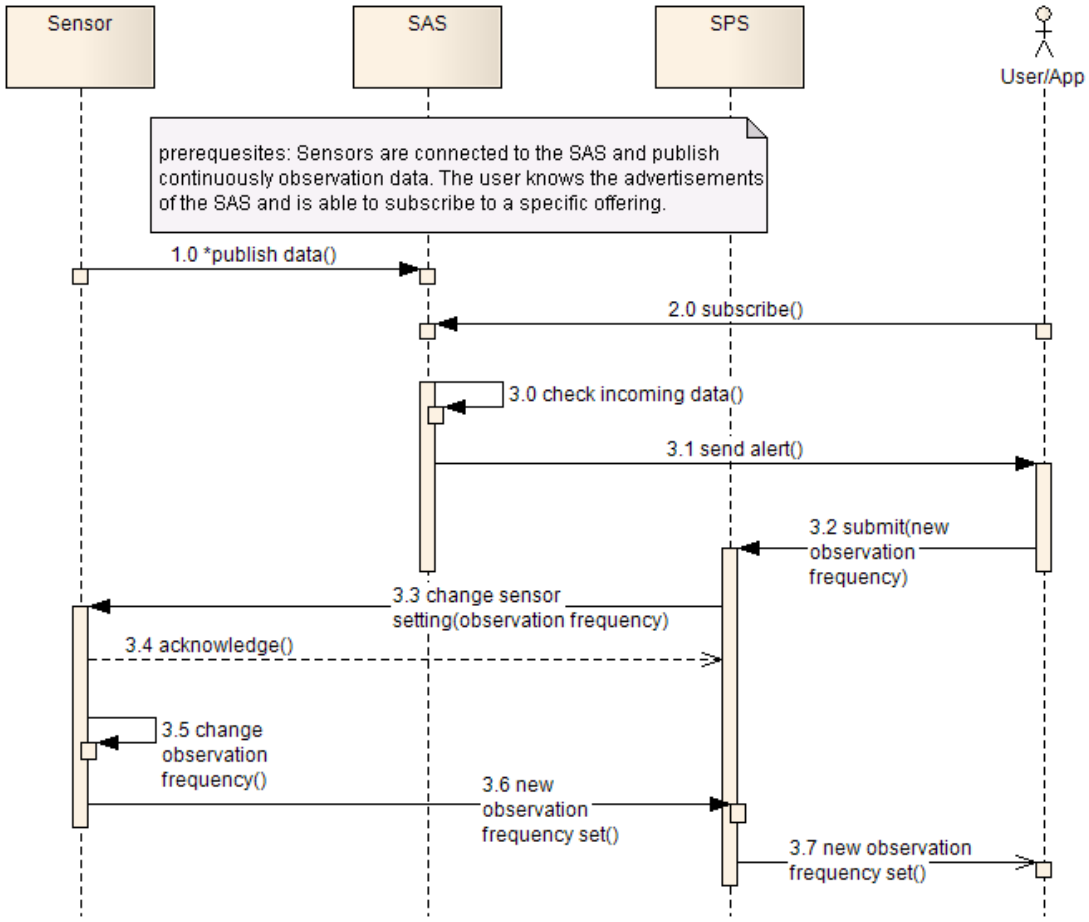

Figure 3: Notification Communication initiated by the service

6 Identifying applicable Requirements for a Secure Sensor Web

The US Department of Defense describes in their 1985 release of the TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA (TSEC) also known as "The Orange Book" requirements, certification classes and evaluation criteria for trusted (not distributed) systems.

For a distributed system, ISO defines in their 10181 series, called "INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – SECURITY FRAMEWORKS FOR OPEN SYSTEMS" (see [1]), security frameworks and provides security specific requirements. In order to get a complete picture, definitions from ITU X.800 (see [67]) are also mandatory as they are used in 10181.

Any approach to describe a secure system requires determining requirements and their applicability to firmly define what "secure" means and which parts of the entire system it affects. For the purpose of this Engineering Report, we define "secure" in close relationship to The Orange Book. For secure computer systems, the Orange Book states that *"secure systems will control, through use of specific security features, access to information such that only properly authorized individuals, or processes operating on their behalf, will have access to read, write, create, or delete information."* [3] It further defines six different types of requirements that can be implemented in different ways to assert trusted behavior compliant to different classes.

The Orange Book mainly targets at securing a computer system through requirements and evaluation procedures for operating systems. It is therefore not directly useable for securing the Sensor Web, as it is an open and distributed system based on a Service Oriented Architecture (SOA). So the question arises, how the requirements and implementations of assurance classes from The Orange Book can be mapped to target architecture. But in general, the property of the system either being distributed or service oriented shall not have any impact on the security definition from The Orange Book, as cited above. An answer to the question about applicable requirements for distributed systems is given in ISO 10181 (all parts). And for a Service Oriented Architecture, [66] defines "tailored" requirements based on ISO 10181, specific for net-centric systems.

6.1 Introduction to TCSEC "The Orange Book"

In 1985, the United States Government Department of Defense released the standard named Trusted Computer System Evaluation Criteria (TCSEC). It defines requirements for assessing the effectiveness of computer security controls, build into a computer system. The TCSEC was updated in 2005 by the Common Criteria standard from ISO (see [2]).

The TCSEC has been developed to serve different purposes. The interesting purpose in the context of this ER is that it provides manufacturers information as to what security controls they have to build into their computer system in order to satisfy trusted processing of sensitive information to prevent its unauthorized disclosure. The

requirements in order to do so are the following:

- **Policy**: A security policy must exist with explicit and well-defined rights to be enforced by the system. Two different kinds of control models are differentiated:
    - Discretionary Access Control (DAC) that supports the "rights of the owner concept" for management of access rights. The rights enforcement is based on the identity of the subject (and the action and resource).
    - Mandatory Access Control (MAC) that restricts the management of domain specific rights only through an administrator. The rights enforcement is based on the identity of the subject and context information. For controlling access to classified information, this can be the classification level of the object and the clearance level of the identity.
- **Marking**: For systems that implement MAC, it is mandatory to store and preserve the integrity of the classification labels with the objects even after exporting.
- **Identification**: Individual subjects must be identified. The identification and the associated rights must be securely maintained by the system.
- **Accountability**: *"Audit information must be selectively kept and protected so that actions affecting security can be traced to the responsible party."* And, *"data must be protected from modification and unauthorized destruction to permit detection and after-the-fact investigations of security violations."* [63]
- **Assurance**: The system must provide individual functions that provide assurance for the requirements Policy, Marking, Identification and Accountability. The Orange Book states that this is typically embedded in the operating system.
- **Continuous Protection**: "The trusted mechanisms that enforce these basic requirements must be continuously protected against tampering and/or unauthorized changes." [63]

The standard further specifies different evaluation classes that assert the implementation of certain security controls. These classes are separated into four divisions. Their main objectives, important in context to this document are:

- **Division D** does contain only one class, which is reserved for those computer systems that have been evaluated but fail to meet the requirements for a higher evaluation class.
- **Division C** is reserved for computer systems that have been successfully evaluated to provide discretionary protection, audit and accountability of subjects and their actions.
- **Division B** is reserved for computer systems that have been successfully evaluated to preserve integrity of classification labels and to provide mandatory protection based on these labels.
- **Division A** is reserved for all computer systems that have succeeded the formal verification methods to assure that mandatory and discretionary security controls are effectively protecting classified information from unauthorized disclosure.

Remark: The Orange Book does not explicitly define the requirement Confidentiality. In a closed system, this is implicitly taken care of by access control. But for a distributed system, confidentiality needs to be applied to any information that is in transit from one

system to the other.

6.2 Definition of Security Requirements based on ISO 10181

ISO 10181 provides a definition for security, focusing to protect the information: "*Information held by IT products or systems is a critical resource that enables organisations to succeed in their mission. Additionally, individuals have a reasonable expectation that their personal information contained in IT products or systems remain private, be available to them as needed, and not be subject to unauthorised modification. IT products or systems should perform their functions while exercising proper control of the information to ensure it is protected against hazards such as unwanted or unwarranted dissemination, alteration, or loss. The term IT security is used to cover prevention and mitigation of these and similar hazards."*[2] The standard further defines security frameworks and the associated requirements, applicable to security services in Open System environments. The term "Open System" is defined to include *"... areas such as Database, Distributed Applications, ODP and OSI."* [2]

In order to protect the exchange of information between secured systems and the management of the stored data, the standard states that*"... security services may apply to the communicating entities of systems as well as to data exchanged between systems, and to data managed by systems."* [2] In subsequent parts of the standard, the requirements and the following security frameworks are defined:

- **Authentication Framework:** ISO 10181-2 defines all basic concepts of authentication in Open Systems: It identifies different classes of authentication mechanisms, the services for their implementation and the requirements for supporting protocols. It further identifies requirements for the management of identity information.

- **Access Control Framework:** ISO 10181-3 defines all basic concepts for access control in Open Systems and the relation to other frameworks such as the Authentication and Audit Frameworks.

- **Non-repudiation Framework:** ISO 10181-4 refines and extends the concepts of non-repudiation, given in ISO 7598-2. It further defines general non-repudiation services and the mechanisms to provide these services.

- **Confidentiality Framework:** ISO 10181-5 defines the basic concepts of confidentiality, identifies classes of confidentiality mechanisms and their maintenance. It further addresses the interactions of the confidentiality mechanisms with other services.

- **Integrity Framework:** ISO 10181-6 defines the basic concepts of integrity, identical to the Confidentiality Framework.

- **Security Audits and Alarms Framework:** ISO 10181-7 defines the basic concepts for security audit and alarms and the relationship to other security services.

7 The Threat Model, Vulnerabilities and Attacks

Protecting a system against all kinds of threats is almost impossible. And, the statement *"we are secure, we have a firewall"* is dangerous as it limits the view towards possible threats to (dis)allow communication typically from the outside world to your internal applications. The unfortunate with this limited view is that when you provide web services, you have to open firewall port 80 and optionally 443, otherwise the outside world cannot execute your service. Therefore, the firewall is just one component in the big picture, when trying the holistic security approach. In order for this approach to meet the expectations, it needs to cover aspects such as securing the network, the computer that hosts the applications, in particular the web service(s) available to the outside world and the applications itself. Securing the applications includes securing the presentation, business and data access logic. In addition, care needs to be taken with maintaining the operating system of the host computers, the runtime services (other than the web services and the platform specific services. The firewall actually belongs to the elements that need to be secured under the network category; Router and Switches also fall under this category. Securing the host basically deals with appropriate configuration of (user) accounts, operating system services, directory and file access as well as file shares. Securing the applications deal with implementing countermeasures or prevention of vulnerabilities towards input validation, authentication, authorization, protection of sensitive data, cryptography, exception handling as well as auditing and logging.

7.1 Defining the Threat Model

One of the first things to do when describing a secure architecture is to define the threat model to expect. CCITT X.800 defines threat as *"A potential violation of security"* and it separates into passive and active threats. For this ER, we understand "security" as outlined in section 6. X.800 defines an active threat to be *"a deliberate unauthorized change to the state of the system"*. They define a passive threat as *"unauthorized disclosure of information without changing the state of the system"*. A vulnerability is defined to be *"any weakness that could be exploited to violate a system or the information it contains."* and an asset is explained to be anything in the system that has value. An attack is the action taken that exploits vulnerability or enacts a threat with the purpose to harm an asset.

A thread model basically describes, which resources are directly available to an attacker or become available as a result of a previously succeeded attack. Even the safest system (whatever that might be) is vulnerable to one thread or another. Let's take the example of the safest safe on this planet that keeps your secret documents. This is great, but not very helpful as you cannot work with the documents, locked up in the safe. Working with the documents requires to open the safe and take the documents out. Here, two aspects are important: First, in order to get the documents out, you have to enter a secret code to open the safe. How can you be sure that no one has installed a hidden camera that frames you whilst entering the secret code and will re-use the code after you are gone to obtain unauthorized access? Second, after you have taken the documents out of the safest environment, they might also become available to spy cameras in your office. What we

wanted to emphasize here is that is extremely difficult to imagine all possible attacks before hand. But nevertheless, when describing a secure distributed system, we need to carefully think about all possible attacks and if the system shall either prevent, detect or can tolerate certain attacks. In order to determine which attacks fall into which category, attack tree analysis can be used.

But let's go back to the threat model. For this Engineering Report, we assume the commonly known Internet Threat Model as defined in RFC 3552 of the IETF (see [68]). This model basically assumes three things:

- The end systems used for communication have not been compromised
- Attackers have full access to the network and can therefore read the traffic and most likely also forge it
- Attackers have reasonable computational ability and computing power and are willing to use it to succeed

From this threat model, we can directly derive vulnerabilities to exercise espionage and sabotage, as they are the main concerns for this security architecture:

Espionage, understood as *"... to obtain information about the plans and activities especially of a foreign government or a competing company"*[1] focuses on the fact to obtain confidential information without being recognized. Therefore, for example the wire taping attack can be exercised to gain information without notice. In later sections, we will layout possible and potential attacks for the baseline.

Sabotage, understood as *"deliberate subversion"*[2] can (in the context of this ER) be exercised by shutting down communication or system(s). Compared to espionage, the attackers are willing to take risks that might exploit them. For Service Oriented Architectures and the assumed Internet Threat Model, as they define the baseline for this ER, typical attacks towards sabotage can be summarized as denial-of-service attacks.

In addition to the assumptions undertaken by the Internet Threat Model, we also need to face possible attacks towards unauthorized disclosure by re-play attacks using modified requests to a service with fraudulent authentication or authorization information.

7.2 Threat Modeling Techniques

For the purpose of identifying threats and potential countermeasures, the following threat modeling process can be leveraged:

1. **Identify the Assets**
   This step concerns the identification of all assets that the system must protect against unauthorized disclosure.
2. **Create an architecture overview**

---

[1] http://www.merriam-webster.com/dictionary/espionage

[2] http://www.merriam-webster.com/dictionary/sabotage

This step focuses on sketching out the parts of the architecture that are relevant for security, such as trust boundaries and communication "crossing" these boundaries.

3. **Decompose the Application**
   The aim of this step is to identify vulnerabilities in the software design, the implementation or the deployment.
4. **Identify the Threats**
   For producing good results during this step, it is essential to think like an adversary. With the attackers goals in mind, take a look at the potential vulnerabilities and take advantage of the fact that you know the software architecture.
5. **Document the Threats**
   Use the same template to document identified threats and use it for keeping track when correcting software design or implementation.
6. **Rate the Threats**
   Depending on the business model or typical usage of the system, try to prioritize the threats. This can be done by estimating the probability of the threat against the damage it can cause.

7.3 Threats and Attacks applicable to a Service Oriented Architecture

CCITT X.800 defines different types of threats:

- Accidental Threads are those that exist because of system malfunctions or software bugs
- Intentional Threats are those that take advantage from special knowledge of the system
- Passive Threats are those that would not change the state of the system, hence modify any information. But, the result of a passive threat if realized is obtaining information in an unauthorized way.
- Active Threats are those that change the state of the system, hence change information if realized.

For this ER, we limit further analysis to passive and active threats, as they are the relevant threats for the assumed Internet Thread Model.

Specific attacks that can be exercised under the given assumptions are:

- **Masquerade**
  This type of attack is applicable to active threats in combination with other attacks such as replay and modification of message attacks. Masquerade attacks try to take advantage over vulnerabilities that exist in implementing Authentication.
- **Replay** (of messages)
  This type of attack is exercised by an adversary to produce an unauthorized effect by re-sending recorded messages either in full or in parts. Replay attacks try to take advantage over vulnerabilities that exist in implementing Access Control (or Authorization).

- **Modification** (of messages)
  This type of attack is exercised by an adversary to produce an unauthorized effect by modifying the content of a message while in transit. The successful completion of the attack requires that the modification is not detected by the receiver. Modification attacks try to take advantage over vulnerabilities that exist in implementing Integrity.
- **Denial** (of service)
  This type of attack is exercised by an adversary to prevent other entities to use a service when required. For a Service Oriented Architecture, the attacker can either suppress all traffic to the service (e.g. firewall drops network packages) or produce that much traffic to the service that it prevents processing of legitimate requests. For Web Services operating on XML messages, it is also possible to send fraudulent messages to the service such that processing of the request consumes all or significant amount of resources of the service (e.g. CPU, memory, sockets, etc.).

7.3.1 Denial of Service Attacks applicable to (OGC) Web Services

Attacks to web services can occur on different layers of the ISO/OSI model. For this ER we assume that countermeasures are in place for denial of service attacks or all layers below the application layer. So for example, we assume that countermeasures are in place that prevent the well-known syn-flood attack (direct or distributed) that result from a fraudulent TCP-handshake, where the client suppresses the final ACK. This typically causes an overflow of the connection table, which ends in rejection of further (perhaps legitimate) connection requests.

Attacks for web services that take place on the application layer of the ISO/OSI model are typically based on corrupted XML input messages. There selective corruption can aim at malfunctioning of different software components of the underlying hardware infrastructure.

1) Attack on XML validation

Before an XML formatted input message is accepted by the service, it is typically validated. Basically two options are supported by XML: First of all, the parser can use the XSD from the “schemaLocation” attribute. This is dangerous as the adversary can reference his own corrupted XSD to make the message become valid, even it is not.

A possible countermeasure is to configure the parser to use a locally stored and trusted XSD to verify the structure of the received XML formatted input message.

2) Attack on XML processing

One common attack is to use entity expansion by injecting DTD based definitions in the XML formatted input message. This will result in a high memory consumption by the parsing process with the side-effect that other processes or threads do not have enough memory or might not be scheduled by the operating system. If the parser is accepting DTD content in the XML formatted input message, the adversary can easily create a 100

byte large input messages that can blow up to 100MB when processed. An example of such an input message is shown below. When processed, it will be expanded to 10^6 (Secure Sensor Web ER9.

A possible countermeasure is to reject XML messages if they contain DOCTYPE snippets.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE foobar [
<!ELEMENT foobar (#PCDATA) >

<!ENTITY x0 "Secure Sensor Web ER">
<!ENTITY x1 "&x0;&x0;&x0;&x0;&x0;&x0;&x0;&x0;&x0;&x0;">
<!ENTITY x2 "&x1;&x1;&x1;&x1;&x1;&x1;&x1;&x1;&x1;&x1;">
<!ENTITY x3 "&x2;&x2;&x2;&x2;&x2;&x2;&x2;&x2;&x2;&x2;">
<!ENTITY x4 "&x3;&x3;&x3;&x3;&x3;&x3;&x3;&x3;&x3;&x3;">
<!ENTITY x5 "&x4;&x4;&x4;&x4;&x4;&x4;&x4;&x4;&x4;&x4;">
<!ENTITY x6 "&x5;&x5;&x5;&x5;&x5;&x5;&x5;&x5;&x5;&x5;">
]>
<foobar>&x6;</foobar>
```

Figure 4: Use of DTD entity expansion to attack XML processing

3) Attack on the application logic

This kind of attack is extremely promising for the adversary as defense is only possible through application specific countermeasures. However, it requires the adversary to be some kind of an expert, as the attacks are application specific. One example for a malicious WFS or WMS input request can use repetition of layer parameters. So for example instead of requesting a map from layer A, the adversary can request layers A{,A}*. Or for a WFS for example, an adversary could create an extremely complex FILTER statement that causes the consumption of all existing resources. According to the principles of Boolean expressions, a true statement can be expressed as A==A or A&A==A&(A|Â). But more dangerous for OGC filter expressions is the vulnerability to extend the Boolean logic using geometries. For example, an adversary can take advantage over the fact that geometry A equals A.intersect(A) . If then the number of points in A is some 1,000,000 or the polygon has some 1,000,000 holes, processing of the intersect operation becomes quite resource consuming.

A possible countermeasure is to do a good sanity check on all incoming request before dispatching it for local processing.

4) Attack on the database logic

This kind of attack focuses on SQL code injection to cause the database logic consume all free resources of the system. The OGC FILTER standards seems to be vulnerable against these attacks if the implementation does not do a sanity check on the SQL statement that is created from the WFS request.

A possible countermeasure is to do a good sanity check on the SQL statement dispatching it to the database.

7.3.2 Example Attacks applicable to XMPP Servers

The Sensor Alert Service (SAS) is leveraging XMPP with Multi User Chat (MUC) rooms to notify subscribed users about alerts. It is therefore relevant for this ER to also take a look at attacks, specific to XMPP servers. For this ER, we will only look at the potential attacks that can occur on the application level and those specific to our scenarios. In particular we assume that the communication between the sender' and receiver' client and the associated XMPP servers is secure. However, it is important to differentiate between an architecture, where the receiver's XMPP server can be trusted or not.

1) Effect of stolen (Multi User) Chat room id

This attack assumes that the adversary is in the possession of a valid chat room id and can spoof the client's identity to gain access to the chat room. For the Sensor Alert Service no authentication is used. It is therefore sufficient for the adversary to read the SAS response on a subscribe operation to connect to the MUC room.

After gaining access to the multi user chat room, the adversary can see other subscribed clients for this alert and can (of course) read all alerts in that chat room and he can also flood the channel that might result in the fact that alerts are not being delivered caused by the lack of available bandwith if these functions for the MUC are not disabled by the creator of the channel or the XMPP server. In order to prevent the adversary to spy out other subscribers, appropriate privacy settings need to be set. Because it can be assumed that the provided multi-user-chats applicable to SWE are not moderated, a "client dropped" criteria reflecting the maximum throughput can be set. The original intension was to prevent spamming over XMPP channels. Another effect of the stolen id is that the adversary can record all alerts that are broadcasted by the SAS on that channel. The security features of TLS and SASL that are already included in the XMPP protocol do not prevent this. In order to ensure that the adversary cannot tape alerts by simply connecting to an alert channel can be mitigated by using MUC with authentication. In order to obey to the requirement that classified alerts need to be confidential to users of a particular clearance, authentication alone is not sufficient.

2) High jacking the receiver's (untrusted) XMPP server

For the scenario, where communication takes place between a trusted and a temporary trusted domain, the risk exists that the administrator of the temporary trusted domain has bad "friends". The high jacking of the XMPP server causes a lot of trouble. We assume that the intent for high jacking was not to shutdown communication as that would exploit the attack. Instead we assume that the adversary likes to take advantage by taping alerts and spying on other clients subscriptions. In contrast to the previous attack where the adversary could only spy on presence information in that one channel, now the adversary has access to the presence information of all clients connected to the XMPP server. In order to prevent the taping of alerts, individual message confidentiality can be applied by

the sender. But because the adversary can potentially record a large amount of alerts, cryptographic strength needs to be ensured.

8 The Approach for Securing the OGC Sensor Web Services

In order to propose a security architecture for the OGC Sensor Web Services to make them usable in the intelligence domain, it is important that the result can pass evaluation criteria similar to those outlined in ISO 15408 (Common Criteria). The challenge for this ER is that those evaluation criteria for the OGC Sensor Web Services does not exist, but still a security architecture is to be proposed in such a way that it is likely to pass evaluation tests created in the future. In order to proceed in this direction, it is important to define a firm set of requirements, as outlines in TCSEC and ISO 10181 and propose standards based implementations, as we see this as the most promising way forward.

Looking at the challenge from Mars, it is important that the end user of the Sensor Web Services can have assured confidence in the architecture in such a way that

- sensor tasking and configuration of alerts can be trusted, hence not modified by any unauthorized user, and
- all alerts from a SAS and observation notifications from a SOS show up at the end user's client, and
- that the end user can trust the incoming alerts to get the attention and become an important part of decisions that might have huge impact.

Zooming in and looking at the challenge from Moon unveils the fact that the system is distributed, based on SOA and is cross security domains. This makes it important that the end user can actually apply trust verification methods to information either received by request or via a notification. In order to achieve that, trusted authentication services are required that issue identity credentials and allow secure verification. In addition, it is important that the information is protected towards integrity; all the way from the originator to the ultimate receiver. In a Service Oriented Architecture, it might happen that the information is forwarded (processed) by many intermediate services, perhaps providing additional pieces of information. But still the integrity of certain parts of a message needs to be guaranteed. This can be compared to the statement in the ISO standard that the distributed aspect of the system shall not make any difference to applying security to a single system. Applying message level security can assure the integrity of the information and its authenticated origin and in addition provide flexible protection towards confidentiality. However, the entire system is only useful, if the services and the communication is available whenever needed. This requires countermeasures for denial of service attacks.

Zooming in and looking at the challenge from a technology point of view, the implementation of message level security fits very good to Web Services, as they exchange XML structures messages. WS-Security and related standards build the foundation to do so. It supports that integrity and confidentiality of classified information assets can be assured, but also involves extremely complex configuration of different security services and the secure distribution of X.509 certificates, definition of access rights to prevent unauthorized disclosure. All so much more complicated than the simple

deployment of a Virtual Private Network (VPN) or leveraging SSL/TLS connections. But for the Secure Sensor Web architecture, it is important to ensure the flexibility of applications to use any service of a trusted federation in any orchestration. This makes the roll-out of a virtual network almost impossible; and in use cases where no direct trust relationship exists between the communication partners, the roll-out of a VPN is probably not possible. With Web Services, trust between these entities can be brokered using Secure Token Services for example. Also, it is important to distinguish between the identity of communication end-points such as security gateway that is being used for establishing a SSL or TLS connection and the actual identity of the end-users, which execute the services and which exchange the information. It is also important to notice that the (mutual) authentication on a SSL connection only lives as long as the session lasts, but according to the requirements from the TCSEC, we need to ensure persistent authenticity, integrity and confidentiality. Using a SSL or TLS communication would protect the information only as long as it is in the secure pipe, but would be unveiled immediately afterwards. Thinking of a sequence of services, all intermediate services would have to be trusted by all partners, as they see all information in the clear. This also does not seem to be feasible for the context of this ER.

In summary, the flexibility of message level security can be leveraged to individually implement all mandatory requirements for implementing a Secure Sensor Web. However, many different aspects of attacks and vulnerabilities need to be investigated to ensure that the overall system, based on OGC Sensor Web Service instances can be trusted and pass evaluation.

As the SAS leverages XMPP for the notification of alerts it is also essential to apply security measures to that protocol, as it might transport alerts of different classification levels inside the same MUC. The vulnerabilities and potential attacks on XMPP to get unauthorized access to the alerts (read, modify or even delete them) are aspects for the evaluation. Looking at the list of extension available for XMPP (XEPs), it becomes clear that the concept of message level security can also be leveraged with XMPP. This would provide a seamless security model that where easy to implement and evaluated as the security concepts are the same for OGC Sensor Web Services and XMPP.

During the next sections of this ER, we will evaluate for each OGC Web Service the interface vulnerabilities and the possible attacks as well as their outreach if exercised, based on the Internet Threat Model. Then, we will introduce concepts for securing the service interfaces to meet the outlined requirements, using existing standards and technologies. We will also introduce concepts for ensuring authenticity, confidentiality and integrity of the service requests and responses. To be comprehensive, we will introduce known vulnerabilities in standards used to make the reader aware of potential pit-falls. For example, the W3C released a XML Digital Best Practices Draft Paper on November 14, 2008 talking about the danger in using certain transform algorithms.

8.1 The Foundation of Message-Level-Security

Before going into the details, it is important to point out that establishing Message-Level Security for distributed systems is lacking of mature standards. All existing standards are relatively new which imposes “teething problems” and; they are not ground-braking -

they just are extensions of well-established security standards (e.g. ISO 10181) for securely exchanging XML structured data.

In short, the foundation of Message-Level Security is given by a handful of standards:

- XML by W3C
- SOAP by the W3C
- WS-Security by OASIS
- XML Digital Signature by W3C
- XML Encryption by W3C
- SAML by OASIS

The most important standard is WS-Security from OASIS. It basically provides the means to secure a SOAP message towards integrity and confidentiality. Even though OGC Sensor Web Services do currently not support SOAP but HTTP/POST binding, this provides an excellent basis for discussing SOAP based implementations of message level security. It is also important to point out that a strong OGC TC motion[3] exists that makes a SOAP interface description of all future version of OGC Web Services mandatory. Therefore, WS-Security is a good foundation for applying security to the OGC Sensor Web Services.

Service Oriented Architectures as they leverage a Web Services alike implementation exchange XML structured messages using SOAP and the services are typically loosely coupled together to fulfill a particular function. And because typically a particular orchestration of services is not known at the point in time when security is implemented, it is important that a message sender can reply on the fact that the message remains unchanged and confidential, until it has reached the ultimate receiver. For XML structured information, the W3C standards XML Digital Signature and XML Encryption provide the functionality necessary to implement this assurance.

XML Digital Signature can be used to apply message integrity using asymmetric encryption. In cases where the authenticity of the origin (the sender) is topic for verification by the ultimate (and perhaps any intermediate) receiver, X.509 certificates must be used. This is because the X.509 certificate ties together the public key and the assured identity of the owner.

---

[3] Approved at the June 2006 Meetings. Going forward, all future revisions of existing and all new OWS (including OLS) interface specifications: (1.) Should include an optional SOAP (messaging) binding and (2.) Should express that binding in WSDL. Exceptions may only be granted through appeal to the OAB. The process will be to write a short argument as to why a SOAP binding is not required for a given implementation specification. The argument might document specific market requirements or technology constraints. For example, a given implementation environment might not support SOAP or there are bandwidth restrictions.

In order to insure that all OGC WS specifications use a consistent approach, the recommendation also included the requirement for OWS Common to describe a consistent pattern for SOAP/WSDL bindings on OGC interface specifications.

For the foreseeable future, OGC will maintain existing GET and POST bindings. Further, there is considerable work being done in the OWS Common RWG on defining a consistent pattern for SOAP binding. Further, please note that SOAP can be thought of as another POST binding. The membership should consider joining into that activity so that we can define a common and consistent patter for implementing SOAP bindings for relevant OGC W*S interface specifications.

XML Encryption can be used to apply message confidentiality to an XML structured message, either in full or in parts. It is important to understand that for performance reasons, a symmetric key is generated on the sender's site to be used for the ciphering of the information to be made confidential. In order to have the receiver de-cipher the protected (hence ciphered) data, the generated key must be send with the message. In order to ensure that only the ultimate receiver can use it, it is encrypted with the public key of the receiver. As the de-ciphering can only take place by the legitimate owner of the associated private key, it is confidential to all other parties or services, receiving the message too.

WS-Security is also significant for describing the foundations of message-level-security, as it provides interoperability when securing SOAP messages towards integrity and confidentiality but also exchanging security context information. WS-Security basically defines three major XML elements for a secure SOAP header: security tokens, XML Encryption and/or XML Signature. The security tokens are used to attach authenticity or authorization information: For authenticity information for example, WS-Security supports simple XML formatted tokens such as username/password tokens (where perhaps the password is given in clear text or digested), XCBF and SAML authentication and assertion tokens as well as binary security tokens such as Kerberos tickets and X.509 certificates. For authorization information for example, WS-Security supports XrML tokens that describe licensed rights of a user and SAML authorization tokens.

As pointed out earlier, for a Service Oriented Architecture, a service instance provides one or more network-endpoints that can be used by any other party to execute the service. In order to give the caller of the service (either a client or another service) sufficient information to actually invoke the service can be provided in a WSDL document. That document describes the service towards network endpoints and their protocol binding, input and output (and error) messages and their structure. If it comes to execute a secured service, in addition the caller needs to have information about the existing security constraints. For example the caller has to know about the accepted security tokens and if the request message has to be encrypted or digitally signed to be accepted and which public key (perhaps from an X.509 certificate) to be used. On a high level, the WS-Policy standard from W3C can be used to express the security constraints and allow the caller and the service to conclude (negotiate) a common set of security constraints; the security context necessary to execute the service. In order to publish the security constraints expressed using a WS-Policy to the caller, they can be included into the WSDL document.

9 Security Discussion for the Sensor Web Services as defined by the Baseline

9.1 The Services Baseline

The objectives of the OGC Sensor Web Enablement (SWE) Initiative are outlined in [69] and [70] as to *"... encompass specifications for interfaces, protocols and encodings that enable discovery, tasking and access of sensors, acquisition of sensor data, and discovery and access of sensor-processing services."* [70]. The baseline documents for the Sensor Web to be considered within this Engineering Report are:

Service - Standards

- Sensor Planning Service (SPS), version 1.0, OGC #07-014r3
- Sensor Observation Service (SOS), version 1.0, OGC #06-009r6

Service - Best Practices Document

- Sensor Alert Service (SAS), version 0.9, OGC #06-028r3

Language - Standard

- SensorML, version 1.0.1, OGC #07-122r2
- O&M, version 1.0, OGC #07-022r1
- TML, version 1.0, OGC #06-010r6

9.2 Communication Patterns applicable to the Baseline

The most important aspect from this ER's perspective is that none of these documents addresses security issues such as possible attacks and their outreach. In order to find the existing vulnerabilities, see the potential attacks and understand the impact of exercised attacks, the following sections outline the services interfaces and the exchanged information under the security focus.

But before we do this, we like to give a short introduction into the two communication patterns and their security implications important for this ER:

The **request/response communication pattern** is described in [71] section S003 Request/Response, such that *"Two parties wish to conduct electronic business by the exchange of business documents. The sending party packages one or more documents into a request message, which is then sent to the receiving party. The receiving party then processes the message contents and responds to the sending party."* [71]. The document further states that it is implementation specific/dependent if this communication pattern can be considered synchronous or asynchronous. A synchronous implementation would be based on HTTP, where the same communication channel is used for exchanging request and response. However, SOAP over HTTP could allow an asynchronous implementation, depending on the processing logic of the receiver. If the receiving service queues incoming messages for further processing, the client service might only

receive a confirmation about this as a synchronous result of the request. After processing is finished, the service could send the response to the client. But this (obviously) requires the client to provide a network endpoint to establish a communication. The latter implementation also requires the use of unique message IDs that are protected towards integrity, so that the calling and responding service can reference the request and the response, not exchanged within the same communication.

The **notification communication pattern** is described in [71] section 3.24 S200 Event notification, such that *"An application subscribes to notifications of certain named events from an event source. When such events occur, notifications are sent back to the originating application (first party notification) or to another application (third party notification)."* [71]. This pattern actually requires a subscription of the receiver to the sensor of the event that can be implemented according to the request/response communication pattern. The actual notification can (according to [71]) be implemented according to the *"fire-and-forget to multiple receivers scenario"* also described in [71], section S002. This scenario does not guarantee the delivery to the sender, as it is implemented without acknowledgment from the receiver to the sender. The implementation can be based on any distribution technology. One popular implementation is XMPP, but also SOAP-enabled event receiver services are possible, so long the implementation manages to send notifications to subscribed receivers only.

9.3 Interface Summary for Baseline Services

9.3.1 Sensor Planning Service (SPS)

Operation name	Input encoding	Output encoding	HTTP Binding
GetCapabilities	KVP or valid XML	Valid XML	GET or POST
GetFeasibility	Valid XML	Valid XML	POST
Submit	Valid XML	Valid XML	POST
GetStatus	Valid XML	Valid XML	POST
Update	Valid XML	Valid XML	POST
Cancel	Valid XML	Valid XML	POST
DescribeResultAccess	Valid XML	Valid XML	POST
DescribeTasking	Valid XML	Valid XML	POST

Table 1 – SPS operation summary

Summary: All operations can be invoked by a valid XML request and return a valid XML

document using the HTTP POST binding and adhere to the request/response communication pattern.

9.3.2 SPS EO Profile

A SPS EO Profile supports all SPS operations as listed above, plus the following:

Operation name	Input encoding	Output encoding	HTTP Binding
DescribeGetFeasibility	Valid XML	Valid XML	POST
DescribeSensor	Valid XML	SensorML	POST
EstimateSenorWorkload	Valid XML	Valid XML + SensorML	POST
DescribeSubmit	Valid XML	Valid XML	POST

Table 2 – SPS-EO Profile operation summary

Summary: All operations can be invoked by a valid XML request and return a valid XML document using the HTTP POST binding. Different to the SPS core specification (see above) is that the GetFeasibility and Submit operations operate according to the notification communication pattern. The invocation of the operation (GetFeasibility or Submit) is used to subscribe to the SPS and the GetFeasibiltiyResponse or the SubmitRespnse message represents the notification.

9.3.3 Sensor Observation Service (SOS)

Operation name	Input encoding	Output encoding	HTTP binding
GetCapabilities	KVP or Valid XML	Valid XML	GET or POST
GetObservation	Valid XML	O&M	POST
DescribeSensor	Valid XML	TML or SensorML	POST
GetObservationById	Valid XML	O&M	POST
GetResult	Valid XML**	O&M	POST

DescribeResultModel	Valid XML	Valid XML	POST
GetFeatureOfInterest	Valid XML	GML	POST
GetFeatureOfInterestType	Valid XML	GML	POST
DescribeObservationType	Valid XML	Valid XML	POST
DescribeFeatureType	Valid XML	Valid XML	POST
*RegisterSensor**	TML or SensorML + O&M document	Valid XML	POST
*InsertObservation**	O&M	Valid XML	POST

Table 3 – SOS operation summary

*: Mandatory for transaction profile

**: plus an O&M instance template document from previous GetObservation

Summary: All operations can be invoked by a valid XML request and return a valid XML document using the HTTP POST binding and leverage the request/response communication pattern.

9.3.4 Sensor Alert Service (SAS)

Operation name	**Input encoding**	**Output encoding**	**HTTP Binding**
GetCapabilities	KVP or Valid XML	Valid XML	GET or POST
Advertise	Valid XML	Valid XML*	POST
RenewAdvertisement	Valid XML	Valid XML	POST
CancelAdvertisement	Valid XML	Valid XML	POST
Subscribe	Valid XML	Valid XML*	POST
RenewSubscription	Valid XML	Valid XML	POST
CancelSubscription	Valid XML	Valid XML	POST

Table 4 – SAS operation summary

*: Contains XMPP MUC URI

Summary: All operations can be invoked by a valid XML request and return a valid XML document using the HTTP POST binding and leverage the request/response communication pattern.

Remark: The actual notification of alerts is operated via XMPP using the MUC URI provided by SAS. Even though XMPP supports user authentication, it is unclear how the user and the sensor get the appropriate login information.

9.3.5 Summary

After we have given the condensed service interface summary focusing on input and output encoding as well as service endpoint binding, it becomes clear that the security implementations need to obey the Sensor Web Services specific limitations. Important for introducing a security concept for Sensor Web Services based on message level security is that HTTP/Get binding is only applicable to the GetCapabilities operation and that all other operations operate on XML request/response messages that can be validated against XML schemata. However, it is not possible to apply message level security to the service interfaces directly, as outlined in a later section.

It is also important to note that the SOS operates on the request/response and notification communication pattern with XML structured messages for in- and output. But the SAS outsources the notification functionality to an XMPP infrastructure.

9.4 Vulnerabilities and Attacks for the Baseline Services

The goal of the security analysis for the OGC SWE services as defined in the OWS-6 baseline is undertaken with the objectives to show possible attacks, derive the possible impact and give recommendations how to make a future version of the service resistant against the attack. In order to gain a complete picture of possible attacks, how to exercise them and what it takes for the adversary to succeed, an analysis for each service operation was done.

For the security analysis, we assumed the Internet Threat Model that basically assumes that systems are secure (cannot be high jacked) and that communication is insecure so that communication can be compromised. This assumption certainly holds so long we look at direct client service interactions, as it usually is the fact for SWE services. But when it comes to prepare a security analysis for other OGC services, it is recommended to also assume the System Threat Model, where the system or applications of intermediate services of a workflow might be compromised. For a direct client service interaction, the Browser Threat Profile would be the counterpart.

In order to rate the possible attacks, different focuses exist:

- How likely is it that a particular attack can successfully be exercised?

- How large is the impact of the successful completion of the attack?
- How difficult is it to detect a particular attack?

Different attacks require different condition. Assuming the Internet Threat Model includes that we have to assume that network configuration can be modified by an adversary in order to get in the position for exercising an attack. This is a relevant assumption, as Eavesdropping, Man-In-The-Middle and ARP-Spoofing are typically attacks that can be leveraged by the adversary to exercise certain attacks. For this Engineering Report we assume that only "in house" attackers are able to modify the settings of a LAN in order to get in the position of exercising associated attacks. For the analysis we assume that the attacker has the knowledge how to modify the network in order to support the intended attack and that enough time is available to exercise the relevant attack(s) to succeed.

- **Eavesdropping** is the attack that will allow to record traffic on a LAN. In case the attacker is in the same network segment, it is relative simple to succeed. In case of switched network segments, some switches can be compromised by poisoning the ARP cache, which causes most switches to fall back to into hub mode. But after the attacker succeeded with switch poisoning, it is just a question of time and the network administrator will detect it.
- **ARP-Spoofing** is an attack on OSI layer 2 that allows spoofing of MAC to IP addresses. As a consequence, the attacker can redirect network communication so that communication will end at the adversary's computer. For the context of this analysis, we will assume that communication from a user client will end at the service of the adversary and not at the actual service.
- **Man-In-The-Middle** is the attack that leverages ARP-spoofing in order to route communication through the adversary's computer. The typical scenario is that the default gateway of a LAN is substituted by the computer of the adversary so that all communication can be recorded and tampered. It is not possible to use this constellation for recording and tampering of HTTPS communication. But with the real existence of to context security verification and self-signed certificates, "dumb" users sometimes simply click "OK" without understanding the implications. For these users, the router of the adversary provides self signed certificates to the client, which should be detected.

In addition to the attacks that require network modification, we also take under consideration an attack type that can be exercised by simply executing the service. The only condition for the attacker is that the network endpoint of the service is known (e.g. can be obtained from a catalog search) and that any operation can be executed. But this is the case for all baseline services, as there is no access control defined. Therefore, these attacks are ones that are exercised most likely.

Also, the overall likelihood that attacks can be exercised and succeed depend on the fact if the adversary has access to the LAN or from a WAN. The options are much larger for exercising different attacks if the adversary is in the LAN, but also does (or at least should) any network modification alarm the network administrator so that it always is a

question of time until the attacker unveils himself. Because the attacker from the WAN is less likely to be identified, the possible attacks are to be taken much more serious.

Another criteria to determine the overall likelihood that an attack results in the desired goal, we can differentiate simple or complex attacks. A simple attack can be described as a one-time invocation of the service operation that already has immediate affect on the asset. On the other hand a complex attack requires the adversary to use a particular sequence of interactions with one or multiple services in order to succeed. For example, if the adversary likes to cancel a running assignment served by an SPS, the adversary needs a taskID. Therefore, the adversary has to obtain the taskID first. This requires either other attacks to exercise of interaction with other services. For example, all information that can be obtained from a GetCapabilites request is immediately at hand of the adversary.

Criteria Summary:

- Does it require modification to the network to succeed with an attack?
- Does it take more than one attack to succeed?
- Does the attacker need application specific knowledge and if so, how much?

Furthermore, we need to take under consideration what the aim of the attack is: Espionage, Sabotage (specific) or DoS (global sabotage):

- Espionage Scenario 1) Attacker wants to obtain certain information about an asset of someone else.
- Espionage Scenario 2) Attacker wants to obtain the produced information of an asset of someone else.
- Sabotage Scenario 1) Attacker wants to tamper a particular asset or assets of a particular entity without unveiling himself.
- Sabotage Scenario 2) Attacker wants to tamper asset, independent if being unveiled.
- DoS Scenario 1) Making certain that no other entity is able to use the asset (sensor, data, service), but without unveiling himself.
- DoS Scenario 2) Making certain that no other entity is able to use the asset (sensor, data, service), even though that requires to immediately unveiling of the attacker.

In order to provide facts for answering as many detailed questions as possible, we created an analysis template that is used for describing the vulnerabilities for each operation of a service:

Cause	Why can this attack be exercised? • Eavesdropping • ARP-Spoofing • Man-In-The-Middle • Execute S*S

Effect	What does it mean for the service?
Result	What does it mean for the client, user or attacker?
Scope	What does the attacker need in order to succeed with the attack?
Example	
Likelihood	The likelihood that an adversary can exercise the described attack • Low = The adversary has to be a domain expert that understands the service and associated specifications (e.g. structure and semantics of requests and responses and the sequence of requests in order to get going to obtain relevant information for exercising the attack) • Medium = The adversary can exercise the attack by reading the service specification; no application (service instance) specific knowledge is required • High = The adversary does not need application or service specific knowledge to exercise the attack
Impact on Asset	The affect on the asset by successfully exercising the attack • Low = There is no direct affect on the asset (e.g. fraudulent/fictitious offerings provided by a SAS and SOS) • Medium = There is a direct affect on the asset but effective to a single client only (e.g. modification of observation request/response) • High = There is a direct affect on the asset effective to all clients for the service under attack (e.g. fraudulent InsertObservation operation with SOS)
Impact on User	The affect impacting the user upon the immediate or future use of the asset by successfully exercising the attack • Low = There is no impact on the user and no affect on the future use of the asset • Medium = There is an impact on the user but no affact on the future use of the asset. • High = There is an impact on the user and an affect on the future use of the asset
Potential	Which information can the adversary gain from the attack to exercise further attacks?
Reason	What does the adversary have in mind? • Sabotage = Affect on assets effective to at least one entity • Denial of Service (DoS) = Prevent the regular use of the service or its resources (sensor, observation, data, etc) which effects all users • Espionage = Gain information in an unauthorized way
Requirement	What needs to be done for a future version of the service to either mitigate or prevent the attack?

Table 5 – Analysis Template

9.5 Sensor Planning Service

9.5.1 Identify the Assets

The definition of the asset for SPS is manning fold: It includes the service as such, as well as the physical asset as it is operated by the service. As each operation has different effects on the service and/or physical sensor, the concrete asset is defined prior to the operation analysis for each operation individually.

9.5.2 Identify the Threats for GetCapabilities() operation

Asset: Sensor metadata and Phenomena Offerings

Cause	Man-In-The-Middle
Effect	Client will receive fraudulent sensor metadata and/or phenomena offerings. This can include fictitious or removed offerings or an empty list of offerings (<SensorOfferingList/> and/or <PhenomenonOfferingList/>). The same is true for the sensor metadata.
Result	User client uses fraudulent sensor metadata and phenomenon offerings.
Scope	Attacker has to have service instance specific knowledge about the structure and the semantics of the Capabilities document in order to derive fraudulent information that is acceptable by the client but leads to erroneous interactions with the SPS.
Example	Assuming the SPS provides a sensor that measures the temperature in Degree Centigrade for Munich, Germany. A simple modification could be change Centigrade to Fahrenheit. Another possibility is to change the location of the sensor so that it is not reporting temperature for Munich, Germany (48.160131,11.580276) but for Munich, ND (48.666988,-98.834295).
Likelihood	Medium
Impact on Asset	None
Impact on User	Impact on the use of the asset as the metadata available to the user has changed.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 6: Modify GetCapabilities() response

Cause	Adversary's client can execute SPS

Effect	Adversary's client will send GetCapabilities() request to SPS
Result	Adversary's client will receive metadata about sensor(s) and phenomena offerings.
Scope	Requires knowledge how to create the GetCapabilities() request
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	Important for exercising other attacks, such as the metadata contains all sensorIDs that are served by the SPS.
Reason	Future Espionage, Sabotage, DoS
Requirement	None

Table 7: Create GetCapabilites() request

Cause	Eavesdropping and adversary can execute SPS
Effect	Adversary's client will send recorded GetCapabilities() request to SPS.
Result	Adversary's client receives SPS capabilities.
Scope	The attacker does not have to have any application specific knowledge.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	Important for exercising other attacks, such as the metadata contains all sensorIDs that are served by the SPS.
Reason	Future Espionage, Sabotage, DoS
Requirement	None

Table 8: Replay GetCapabilites() request

Cause	Eavesdropping

Effect	Adversary's client will record GetCapabilities() request/response to SPS.
Result	Adversary's client receives SPS capabilities.
Scope	No application specific knowledge required to exercise this attack.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	Important for exercising other attacks, requiring sensorID as input.
Reason	Future Espionage, Sabotage, DoS
Requirement	Allow execution of GetCapabilities() for authenticated users only and protect response with confidentiality to prevent unveiling of the metadata.

Table 9: Record GetCapabilites() request/response

9.5.3 Identify the Threats for DescribeTasking() operation

Asset: Sensor metadata

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent metadata for a sensor assignment.
Result	User client might not be able to task the sensor due to the fraudulent information.
Scope	Attacker has to have application specific knowledge to tamper the response "properly".
Example	
Likelihood	Low
Impact on Asset	None
Impact on User	There is a potential affect on the future use of the asset effective to the actual user in cases where the user client makes the actual tasking of a sensor dependent on the response from the DescribeTasking() operations.
Potential	N/A
Reason	Sabotage

Requirement	Integrity

Table 10: Modify DescribeTasking() response

Cause	Adversary's client can execute SPS
Effect	Adversary's client will receive sensor metadata about a sensor that is relevant for submitting an assignment request (Submit() operation).
Result	Adversary obtains sensor metadata.
Scope	Adversary has to have application specific knowledge and a valid sensor ID. But this is not problematic; it just requires to issue a GetCapabilities() request first.
Example	
Likelihood	Low
Impact on Asset	None
Impact on User	None
Potential	The adversary gains all parameters of a sensor for tasking.
Reason	Espionage
Requirement	None

Table 11: Create DescribeTasking() request

Cause	Eavesdropping and adversary's client can execute SPS
Effect	Adversary's client will send recorded DescribeSensor() request to SPS.
Result	Adversary's client will receive sensor metadata.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	The adversary gains all parameters of a sensor for tasking.
Reason	Espionage

	But the (Shannon) entropy of the information gained is probably zero unless the SPS is serving new sensors since the last attack.
Requirement	None

Table 12: Replay DescribeTasking() request

Cause	Eavesdropping
Effect	N/A
Result	Adversary obtains sensorID and tasking parameter.
Scope	No application specific knowledge is required to exercise the attack.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	The adversary gains all parameters of a sensor for tasking, including the sensor ID.
Reason	Fetch information required to exercise other attacks.
Requirement	Confidentiality of the sensorID and the tasking parameters as they are valuable for the attacker.

Table 13: Record DescribeTasking() request/response

9.5.4 Submit() operation

Asset: sensor

Cause	Man-In-The-Middle
Effect	SPS will receive fraudulent assignment request
Result	1) If the request was modified in such a way that the assignment cannot be accepted by the SPS, the user client will receive an error and no fraudulent tasking of the sensor will occur. 2) If the request was modified in such a way that the SPS accepts the assignment request, the SPS will task a sensor in a fraudulent way but the user client will receive O.K. not able to determine that the O.K. is associated to a fraudulent assignment.

Scope	The adversary has to have application specific knowledge how to “properly” tamper the request in order to undertake the desired goal.
Example	
Likelihood	Medium
Impact on Asset	Direct affect on the asset if the request is tampered in such a way that the SPS accepts the fraudulent request.
Impact on Asset	No affect on the asset if the tampered request is rejected by the SPS.
Impact on User	Direct affect on the use of the asset if the attack always tampers the request in such a way that it is always rejected by the SPS. The consequence is that the user can never task the asset.
Impact on User	User cannot evaluate integrity of request and response, i.e. service might task sensor differently.
Potential	N/A
Reason	Sabotage
Requirement	Integrity of the request, Authentication and Access Control to protect execution of the operation for trusted users only. As a consequence, the successful creation of a taskID requires that the identity of the user is associated with it so that the legitimate owner can only execute other operations on the task.

Table 14: Modify Submit() request

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent information about the success of the submitted assignment request.
Result	User client might take wrong action(s) based on the fraudulent information.
Scope	Adversary has to have application specific knowledge.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	A potential for affecting further interactions between the user client and the SPS towards the use of the asset exists. This in particular if the response was modified from "not_feasible" to "feasible", as the user is not aware of the fact that the wanted tasking was rejected by the SPS and will therefore never try another attempt to task the asset.

Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 15: Modify Submit() response

Cause	Adversary's client can execute SPS.
Effect	SPS will receive a fictitious assignment request for a sensor ID.
Result	Attacker's client will receive a response by the SPS containing a taskID, created for the request. Sending another request to the SPS will return another taskID.
Scope	The attacker needs to know a valid sensor ID and tasking parameter that can be obtained from a DescribeTasking() request. By exercising this attack a couple of times might unveil the structure of the taskID which puts the attacker in the position to start successfully guessing taskIDs that are associated with running assignments. Using of the GetStatus() or even UpdateRequest() or Cancel() operation with a correctly guessed taskID brings the attacker in the position to unveil the status of a running task or even allow to change or cancel a task.
Example	Assuming that the SPS creates taskIDs like natural numbers (1, 2, 3, 4, …), will unveil the pattern quite quickly. If the first request of the attacker will result in a taskID 4711, the next in 4712, etc. Will make almost certain that the SPS has used the numbers 1, 2, …, 4709, 4710 already. Therefore, a Cancel() request using taskID 4710 might end in a successful cancelation of that task!
Likelihood	Low if the attacker actually intends to task a sensor, because the request must be correct which can require to set complex parameters appropriately.
Likelihood	High if the attacker wants to obtain a taskID for learning its pattern, because it is sufficient to receice the error message which does not require to set all parameters with appropriate values.
Impact on Asset	No direct affect on asset if the request is not accepted by the SPS. But this attack can be exercised to learn about the structure of task IDs. Depending on the pattern used to create task IDs, it is more or less likely that the adversary can successfully guess a valid tasked. And with that knowledge undertake other attacks towards espionage or sabotage.
Impact on Asset	Direct affect on asset if the request is accepted by the SPS.
Impact on User	Affect on the use of the asset effective to all users of the SPS as the asset might not be available due to fraudulent tasking by the adversary.
Potential	Learn taskID pattern for exercising other attacks that require a taskID as input.
Reason	Espionage: Task a sensor to obtain production data. Sabotage: Adversary's client could cancel or update running assignments if taskID

	is known.
Requirement	Make taskID a number difficult to guess, e.g. a random number and do not submit a taskID in the error response.

Table 16: Create Submit() request

Cause	Eavesdropping
Effect	Adversary's client will send recorded Submit() request to SPS.
Result	The SPS will potentially accept this request after the addressed sensor becomes available. E.g. the previous assignment has ended.
Scope	The adversary does not have to have application specific knowledge
Example	
Likelihood	High
Impact on Asset	Potential for direct affect on the asset if the attacker has recorded a valid request that most likely will allow the successful re-tasking of a sensor at a later time.
Impact on User	A successful replay of a previously recorded Submit() request will block availability of the sensor effective to all users of the SPS.
Potential	It is possible to use the gained information for issuing a GetFeasibility() request which might cause the SPS to undertake heavy processing. It is also possible to block the availability of a sensor by re-tasking effective to all other users.
Reason	Espionage, Sabotage, Denial of Sensor Availability
Requirement	Unique request ID and timestamp

Table 17: Replay Submit() request

Cause	Eavesdropping
Effect	Adversary's client will record Submit() request/response.
Result	Adversary's client will gain information which entity has submitted an assignment request and if it was accepted by the SPS. The response does contain a taskID.
Scope	The adversary does not have to have application specific knowledge.
Example	

Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	The adversary can use the taskID or the assignment parameters to exercise further attacks at any later time: e.g. GetStatus(), Cancel() or RequestUpdate() operation
Reason	Espionage, Sabotage
Requirement	Confidentiality of the taskID in the response and the assignment parameters in the request.

Table 18: Record Submit() request/response

Cause	ARP-Spoofing
Effect	User client will send Submit() request to adversary's SPS.
Result	User client will receive the response from the adversary's SPS.
Scope	Application specific knowledge is required to "properly" respond to the request.
Example	
Likelihood	Low
Impact on Asset	No direct affect on asset as the actual sensor will not be tasked.
Impact on User	As the response is coming from the adversary's SPS, the user will never be able to undertake the desired tasking of the actual sensor, because it is impossible to determine from the response that was sent by the adversary's SPS.
Potential	N/A
Reason	Sabotage
Requirement	Authentication of SPS and authenticity on the response.

Table 19: Redirect Submit() request

9.5.5 DescribeResultAccess() operation

Asset: Information produced by a sensor provided by the SPS

Cause	Man-In-The-Middle

Effect	SPS will receive a request where to obtain production data for a ficticious taskID.
Result	User client might obtain fraudulent information where to access sensor production data.
Scope	Adversary has to have application specific knowledge. Furthermore, the adversary needs to know valid sensorID(s) or taskID(s). This is required if the response of the SPS shall point to a data production of another entity. In order to gain information about completed tasks, the adversary must guess a proper taskID. This can be undertaken by exercising the "Create Submit() request" attack and then invoke GetStatus().
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Direct affect on use of the asset as the user obtains wrong access information which prevents the user from retrieving the produced data. Depending on the goal of the attack, the adversary will change the request such that it results in an error. This would prevent the user client from obtaining the access information. If the goal is to give the user client the access information to another production (perhaps of another entity or an earlier production from the same entity), the change of the taskID needs to be "properly" done.
Potential	N/A
Reason	Sabotage
Requirement	Integrity, Access Control: Only the owner of a task can request the information where to obtain the produced information. The rights management should be discretionary so that the owner of the task can decide whom to grant access to the information where the production data can be obtained.

Table 20: Modify DescribeResultAccess() request

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent information where to obtain sensor production data.
Result	User will use fraudulent access information.
Scope	Adversary has to have application specific knowledge and know how to "properly" modify access information for the "download" service.
Example	
Likelihood	Medium

Impact on Asset	None
Impact on User	As the user client receives tampered information where to obtain the produced data (e.g. points to a service provided by the adversary), this attack can have immediate affect to the use of the asset as further interactions undertaken by the user rely on the information given by the adversary.
Impact on User	User might access fraudulent data in case of fraudulent service reference.
Potential	N/A
Reason	Sabotage
Requirement	Integrity and Authenticity

Table 21: Modify DescribeResultAccess() response

Cause	Adversary's client is able to execute SPS
Effect	SPS will receive request from adversary's client where to access sensor production data for a sensorID or a (guessed) taskID.
Result	Adversary's client might obtain access information to sensor production data that belongs to assignments, submitted by other entities.
Scope	Adversary has to have application specific knowledge and a valid sensorID or taskID.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Potential for a direct affect on asset exists if the adversary successfully deletes it.
Impact on User	The adversary might obtain the sensor production data and perhaps can delete it afterwards if the service provides such an operation. This would have affect on the asset effective to all users.
Potential	Fetch the access information to production data of another entity.
Reason	Espionage if the production data of another entity will be obtained. Sabotage if the production data of another entity will be tampered or deleted.
Requirement	Authentication and Access Control to ensure that only a task owner can request the access information.

Table 22: Create DescribeResultAccess() request

Cause	Eavesdropping and adversary's client can execute SPS
Effect	Adversary's client will send recorded DescribeResultAccess() request.
Result	Adversary's client will receive access information to obtain sensor production data.
Scope	No application specific information is required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	For using the access information, some (other) knowledge is required how to succeed when using the access information.
Reason	Espionage, Sabotage
Requirement	None

Table 23: Replay DescribeResultAccess() request

Cause	Eavesdropping
Effect	N/A
Result	Adversary's client will receive access information to obtain sensor production data.
Scope	No application specific information is required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	For using the access information, some (other) knowledge is required how to succeed when using the access information.
Reason	Espionage, Sabotage
Requirement	Only authenticated users are allows to request the access information and the response is confidential for the legitimate user.

Table 24: Record DescribeResultAccess() request/response

9.5.6 GetFeasibility() operation

Asset: service (assignment request metadata)

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent feasibility information on the submitted assignment parameter.
Result	User client receives the feasibility that is fictitious and not associated to the intended request. This might prevent that the user is ever trying to task the sensor.
Scope	Adversary has to have application specific knowledge.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Direct affect if feasible request is tampered as "not-feasible" and user consequently doesn't issue "submit" requests
Potential	N/A
Reason	Denial of Sensor – The user might never try to task the sensor with the assignment parameters, as the feasibility does not indicate that it is possible.
Requirement	Integrity

Table 25: Modify GetFeasibility() respone

Cause	Man-In-The-Middle
Effect	SPS will receive fraudulent GetFeasibility() request.
Result	SPS will inform user client about the feasibility on its request but the answer was derived by the SPS on fraudulent information.
Scope	Adversary has to have application specific knowledge.
Example	
Likelihood	Medium
Impact on Asset	Direct affect on asset as tampered GetFeasibility() request parameters can cause the SPS to undertake costly processing. And as the user might not task a sensor before the parameters are “cleared” by the GetFeasibility() operation, the tasking will either

	move into the future or actually never happen.
Impact on User	Direct affect on the use of the asset effective to the active client as the response is not associated to the actual request sent by the user.
Potential	N/A
Reason	Denial of Sensor – The user might never try to task the sensor with the assignment parameters, as the feasibility does not indicate that it is possible.
Requirement	Integrity

Table 26: Modify GetFeasibility() request

Cause	Adversary's client is able to execute SPS
Effect	Adversary's client sends a GetFeasibility() request with (fictitious) assignment paramters to the SPS.
Result	The SPS will derive the feasibility of the assignment request that is part of the GetFeasibility() request.
Scope	Adversary has to have application specific knowledge. In particular, he needs to know how to create a fictitious assignment request.
Example	
Likelihood	Low
Impact on Asset	None However, in cases where the fictitious assignment is extremely complex, the calculation of the feasibility might consume SPS resources and result in slower processing of other requests.
Impact on User	None
Potential	The adversary can submit concrete assignment parameters to test the feasibility for actually tasking the sensor.
Reason	DoS if the adversary submits extremely complex assignment parameters. Espionage if the adversary submits concrete assignment parameters prior to actually tasking the sensor.
Requirement	Sanity check on the request to detect fictitious complexity.

Table 27: Create GetFeasibility() request

Cause	Eavesdropping

Effect	Adversary's client will send recorded GetFeasibility() request messages to SPS.
Result	SPS is processing GetFeasibility() requests
Scope	No application specific knowledge required as recorded messages are used.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	Depending on the implementation, the continuous replay of GetFeasibility() requests might prevent the acceptance of Submit() requests by the SPS.
Reason	DoS In cases where the recorded request contains a complex assignment, sending this request in a bulk might cause the SPS to slower processing.
Requirement	Unique message ID and time-stamp as well as integrity to detect the replay. The implementation has to guarantee that continuous replay of GetFeasibility() requests do not prevent the actual tasking of a sensor.

Table 28: Replay GetFeasibility() request

Cause	ARP spoofing
Effect	User client will send GetFeasibility() request to fraudulent SPS.
Result	User client will receive a fictitious (probably fraudulent) feasibility that most likely is not associated to the original request.
Scope	No application specific knowledge is required. However, the attacker has to be able to set up a service that returns an appropriate result back to the user client.
Example	
Likelihood	Medium
Impact on Asset	None – As the processing of the feasibility will not take part for the actual sensor.
Impact on User	Direct affect on use of asset effective to the active client.
Potential	The adversary might obtain a large set of assignment parameters for sensor. The adversary is able to determine the kind of sensor and its operation based on the obtained information.
Reason	Sabotage

	Assuming that the user client will issue a GetFeasibilityRequest() prior to submitting an assignment request for tasking of a complex sensor (e.g. a satellite), the user client might never actually issue the Submit() request, if the response of the feasibility request is permanently negative.
Requirement	Authentication for the SPS and Authenticity of the response so that the user client can determine that the response came from the attacker's SPS.

Table 29: Redirect GetFeasibility() request

Cause	Eavesdropping
Effect	N/A
Result	The adversary might obtain a large set of assignment parameters for sensor. The adversary is able to determine the kind of sensor and its operation based on the obtained information.
Scope	No application specific knowledge is required to exercise this attack.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	The adversary can obtain information such as assignment parameters that can be used for exercising other attacks.
Reason	Espionage
Requirement	Confidentiality

Table 30: Record GetFeasibility() request/response

9.5.7 GetStatus() operation

Asset: sensor assignment (task)

Cause	Man-In-The-Middle
Effect	SPS will receive fraudulent GetStatus() request.
Result	User client will receive fictitious status on any assignment but the one requested.
Scope	Adversary has to have application specific knowledge.

Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Affect on use of asset effective to the active client as the wrong status information might prevent actually requesting of the production data (the status request was tampered such that the taskID refers to a running task).
Potential	N/A
Reason	Sabotage, as the user will receive the status for a different taskID.
Requirement	Integrity on the request.

Table 31: Modify GetStatus() request

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent status information.
Result	User will not know the status about his request.
Scope	Adversary has to have application specific knowledge.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Direct affect on use of asset as the user will never receive a task completion notification and therefore never try to obtain the production data.
Potential	N/A
Reason	Sabotage
Requirement	Integrity on the response.

Table 32: Modify GetStatus() response

Cause	Adversary's is able to execute SPS
Effect	GetStatus() operation of the SPS is invoked.
Result	The adversary might receive status information about the task if the GetStatus() request contained a valid taskID.

Scope	Adversary has to have application specific knowledge. In addition, the attacker has to know a valid taskID.
Example	
Likelihood	Low
Impact on Asset	None
Impact on User	None
Potential	N/A
Reason	Espionage
Requirement	Access Control to prevent unveiling of task status information to other entities than the owner.

Table 33: Create GetStatus() request

Cause	Eavesdropping
Effect	Adversary's client will send recorded GetStatus() requests to SPS.
Result	SPS returns the status for the requested task to adversary's client.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	N/A
Reason	Espionage It is important to note that the adversary can only obtain status information as long as the task is active.
Requirement	Unique request ID and time stamp as well as integrity

Table 34: Replay GetStatus() request

Cause	Eavesdropping

Effect	N/A
Result	The adversary obtains information about taskID and status.
Scope	No application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	None
Potential	The adversary can use the obtained taskID(s) to update or even cancel the task.
Reason	Sabotage
Requirement	Confidentiality on the taskID in the request.

Table 35: Record GetStatus() request/response

9.5.8 Update() operation

Asset: sensor assignment (task)

Cause	Man-In-The-Middle
Effect	SPS receives fraudulent Update() request.
Result	SPS will change the processing of a running task according to the fraudulent values.
Scope	Application specific knowledge is required. In particular knowledge is required to change the request “properly” in order to reach the goal.
Example	
Likelihood	Medium
Impact on Asset	Immediate affect on the asset.
Impact on User	Immediate affect on the use of the asset as the success of the desired modification is associated to a tamprered request.
Potential	N/A
Reason	Sabotage

Requirement	Access Control to allow changing of running assignments for task owners only.

Table 36: Modify Update() request

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent response on success of the assignment update.
Result	User will not know the correct status of the update.
Scope	Application specific knowledge is required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	The user does not receive the correct status associated to the issued Update() request. (potentially red, as it might affect further use)
Potential	N/A
Reason	Sabotage
Requirement	Integrity of the request.

Table 37: Modify Update() response

Cause	Adversary's client is able to execute SPS
Effect	Update() operation of the SPS is invoked containing a fraudulent request.
Result	The adversary might change the processing of one or many running assignments.
Scope	Application specific knowledge is required. In particular, the attacker has to know the valid taskID that shall be sabotaged and the correct assignment parameters.
Example	
Likelihood	Low
Impact on Asset	Immediate affect on asset
Impact on User	Affect on use of asset as the assignment created by the user is modified. It is therefore very likely that the production data are not associated to the actual tasking done by the user.
Potential	N/A

Reason	Sabotage
Requirement	Access Control to prevent that only the owner of a task can issue an Update().

Table 38: Create Update() request

Cause	Eavesdropping and adversary's client is able to execute SPS
Effect	Adversary's client will send recorded Update() request to SPS.
Result	SPS will modify associated assignment.
Scope	No application specific knowledge is required.
Example	
Likelihood	High
Impact on Asset	Immediate affect on asset if the previously recorded request can be applied another time
Impact on Asset	None if the replayed update request is rejected by the SPS.
Impact on User	Affect on the use of the asset and the production data if the replayed Update() request was accepted by the SPS. Particularly true if user sends multiple update requests himself.
Impact on User	None if the replayed Update() request was rejected by the SPS.
Potential	N/A
Reason	Sabotage
Requirement	Unique request ID and time stamp as well as integrity.

Table 39: Replay Update() request

Cause	Eavesdropping
Effect	N/A
Result	The adversary gains information about assignment parameters for a sensor.
Scope	No application specific knowledge is required.
Example	
Likelihood	Medium

Impact on Asset	None
Impact on User	None
Potential	Adversary can obtain assignment parameters for a sensor and taskID that can be used for future attacks.
Reason	Espionage
Requirement	Confidentiality

Table 40: Record Update() request/response

9.5.9 Cancel() operation

Asset: sensor assignment (task)

Cause	Man-In-The-Middle
Effect	SPS receives fraudulent Cancel() request.
Result	SPS will cancel the processing of a running task according to the fraudulent taskID if valid. Intended task to be cancelled remains active.
Scope	Application specific knowledge is required. And the attacker has to know the taskID to be sabotaged.
Example	
Likelihood	Medium
Impact on Asset	Immediate affect on asset if taskID is valid.
Impact on User	Immediate impact on at least one user as his running task might be the one cancelled by the attack.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 41: Modify Cancel() request

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent response on success status of the requested

	assignment cancellation.
Result	User will not know the correct status of the request to cancel the assignment.
Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Potential affect on further use of the asset effective to the active client. as the user does not know if the request was accepted or rejected by the SPS. Therefore, a potential erroneous processing of the Cancel() request could not become aware to the user.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 42: Modify Cancel() response

Cause	Adversary's client is able to execute SPS.
Effect	Adversary's client will send Cancel() request to the SPS.
Result	The SPS might cancel the processing of a running assignment (task) if the fraudulent request from the adversary's client is applicable to a task.
Scope	Application specific knowledge is required. In addition, the attacker needs to know the taskID to be sabotaged.
Example	
Likelihood	Low
Impact on Asset	Immediate affect on asset.
Impact on User	Immediate impact to potentially all users as the attack might have cancelled his task.
Reason	Sabotage
Requirement	Access control to ensure cancellation of a running task is possible for task owner only.

Table 43: Create Cancel() request

Cause	Eavesdropping and adversary's client can execute SPS
Effect	Adversary's client will send recorded Cancel() requests to SPS.
Result	SPS will try to cancel assignment that has already be cancelled.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None as the request is already cancelled.
Impact on User	None
Potential	N/A
Reason	Sabotage
Requirement	The SPS shall not submit the same taskID twice.

Table 44: Replay Cancel() request

Cause	Eavesdropping
Effect	N/A
Result	N/A
Scope	No application specific knowledge is required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	None
Reason	Espionage
Requirement	None

Table 45: Record Cancel() request/response

9.5.10 Summary of the Attacks

For all operations of the SPS it is required to ensure integrity of the request and the response to prevent attacks that modify the content. In order to prevent spoofing attacks, the SPS has to authenticate to the client.

- The Create/Record GetCapabilities() attacks unveil sensor metadata that is important information to the adversary to exercise other attacks. In order to mitigate the exploitation of the capabilities to an adversary, the GetCapabilities() operation can be put under access control to ensure only authenticated users can execute the operation and that the response (the capabilities of the SPS) is confidential for the identified user.
- The Record DescribeTasking() attack unveils assignment parameters that is valuable input for the adversary. To prevent future attacks based on that information, it is required to ensure confidentiality on the sensor URI in the request and the assignment parameters in the response.
- Attacks on the Submit() operation require that the assignment parameters in the request and the taskID in the response are confidential. It is also important that the SPS associates the identity of the caller to the taskID to control future operations (e.g. Update() or Cancel()) on the task for owners only.
- Attacks leveraging the GetStatus(), Update() and Cancel() operations can be prevented if the SPS establishes Access Control to prevent execution of these operations for entities other than the owner. Further more it is required that the taskID in the request is confidential (exept the Cancel() operation) so that it cannot be recorded by the adversary and misused in future attacks.
- For the GetFeasibility() operation it is required to ensure confidentiality of the assignment parameters and the sensor URI in the request and the feasibilityID in the response to prevent its misuse by the adversary in future attacks.
- In order to prevent the misuse of assignment parameters that can be obtained by the adversary upon leveraging the Create DescribeTasking() attack, it is required that this operation can only be executed by authenticated users and that the assignment parameters in the response are confidential.
- The Create DescribeResultAccess() request attack unveils information to the adversary to obtain the sensor production data of other entities. In order to prevent this, access control shall ensure that only task owners can execute the operation. Confidentiality of the response shall ensure that the access information cannot be recorded by the adversary leveraging the Record DesccribeResultAccess() attack.

As the taskID and the feasibilityID are handles between different operations, it is important to keep them confidential. It is also important to create secure random task- and feasibility IDs to prevent guessing by the adversary.

9.6 Sensor Observation Service

9.6.1 GetCapabilities() operation

Asset: Observation Offerings

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent observation offerings. This can include fictitious offerings or removed offerings or an empty list of offerings (<ObservationOfferingList/>).
Result	User might not find the expected (required) offering even though it might exist.
Scope	Application specific knowledge is required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Affect on the use of the asset effective to the active client, as existing offerings might be fraudulent or removed, an indirect affect on use of assets exists.
Potential	N/A
Reason	Sabotage Denial of Service use if the ObservationOfferingList is empty
Requirement	Integrity

Table 46: Modify GetCapabilities() response

Cause	Adversary's client can execute SOS.
Effect	Adversary's client will send GetCapabilities() messages to SOS and receive service offerings.
Result	Adversary can use the obtained information to execute other service operations.
Scope	Very little application specific knowledge is required to create the request. But full application specific knowledge is required to understand the response and how to use it in future attacks.
Example	
Likelihood	High
Impact on Asset	None

Impact on User	None
Potential	The adversary obtains information about offerings served by the SOS that can be used in future attacks.
Reason	Espionage Possible intent to sabotage as the information from the capabilities document is the baseline for other attacks.
Requirement	None

Table 47: Create GetCapabilites() request

Cause	Eavesdropping and adversary's client can execute SOS.
Effect	Adversary's client will send recorded GetCapabilities() request to SOS
Result	Adversary obtains information about the available offerings.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	N/A
Reason	Espionage Possible intent to sabotage as the information from the capabilities document is the bases for everything else.
Requirement	None

Table 48: Replay GetCapabilites() request

Cause	Eavesdropping
Effect	Adversary's client will record GetCapabilities() request/response to SOS.
Result	Adversary's client receives SOS capabilities.
Scope	No application specific knowledge required to exercise this attack.

Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	Important for exercising other attacks, requiring sensor metadata and information about observation offerings as input.
Reason	Future Espionage, Sabotage, DoS
Requirement	Allow execution of GetCapabilities() for authenticated users only and protect response with confidentiality to prevent unveiling of the metadata.

Table 49: Record GetCapabilites() request/response

9.6.2 DescribeSensor() operation

Asset: Sensor

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent metadata about a sensor and observation offering(s) served by the SOS.
Result	User gets fraudulent sensor metadata.
Scope	Application specific knowledge required.
Example	
Likelihood	Low
Impact on Asset	None
Impact on User	Direct impact on further use of the asset effective to the active client as it is based on the tampered metadata.
Reason	Sabotage
Requirement	Integrity

Table 50: Modify DescribeSensor() respone

Cause	Adversary's client is able to execute SOS

Effect	Client will receive metadata about a sensor which observation offering(s) are served by the SOS.
Result	Adversary can obtain metadata information relevant for other attacks.
Scope	Application specific knowledge required.
Example	
Likelihood	Low
Impact on Asset	None
Impact on User	None
Reason	Espionage Possible intent to sabotage as the information from the response is the baseline for other attacks.
Requirement	None

Table 51: Create DescribeSensor() request

Cause	Eavesdropping
Effect	Adversary's client will send recorded DescribeSensor() request to SOS.
Result	Adversary will receive sensor metadata.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	N/A
Reason	Espionage
Requirement	None

Table 52: Replay DescribeSensor() request

Cause	Eavesdropping

Effect	N/A
Result	Adversary will receive sensor metadata.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	The adversary can obtain detailed information on a sensor encoded in SensorML or TML that can be used for future attacks.
Reason	Espionage
Requirement	None

Table 53: Record DescribeSensor() request/response

9.6.3 GetObservation() operation

Asset: Observation data

Cause	Man-In-The-Middle
Effect	SOS will receive fraudulent GetObservation() request.
Result	User receives observation data that is not associated to the actual request, if the request was modified in such a way that the SOS can still match the request to existing observation(s). User will receive error in all other cases.
Scope	Application specific knowledge required. In particular, the adversary has to know a valid taskID to have the SOS return associated observation data.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Direct affect on asset effective to the active client.
Potential	N/A
Reason	Sabotage

Requirement	Integrity

Table 54: Modify GetObservation() request

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent GetObservation response.
Result	User gets observation data that is not associated to the request.
Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Immediate affect on asset
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 55: Modify GetObservation() response

Cause	Eavesdropping
Effect	None
Result	Adversary obtains observation data.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	The adversary can obtain observation data of other entities.
Reason	Espionage
Requirement	Confidentiality

Table 56: Record GetObservation() response

Cause	Attacker's client can execute the SOS
Effect	Adversary's client will receive observation data about an observation offering(s), served by the SOS.
Result	Adversary gets the observation offerings of the SOS.
Scope	Application specific knowledge is required. In particular the offering URI and the URI referencing the phenomena.
Example	
Likelihood	Low
Impact on Asset	None
Impact on User	None
Potential	The adversary can obtain observation data for a particular phenomenon.
Reason	Espionage
Requirement	Access Control to prevent unauthorized access to observation data.

Table 57: Create GetObservation() request

Cause	Eavesdropping and adversary's client can execute SOS.
Effect	Adversary's client will send recorded GetObservation() request to SOS.
Result	Adversary's client will receive observation from the SOS.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	N/A
Reason	Espionage
Requirement	Unique request ID and time-stamp to detect replay.

Table 58: Replay GetObservation() request

Cause	Man-In-The-Middle
Effect	User client's GetObservation() request will be send to adversary's SOS.
Result	User receives fraudulent observation data from adversary's SOS.
Scope	Application specific knowledge is required as the adversary's SOS has to response "properly".
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Direct affect on asset.
Potential	N/A
Reason	Sabotage
Requirement	Authentication for SOS and Authenticity on the response so that the user client can determine that the result came from another service.

Table 59: Redirect GetObservation() request

9.6.4 RegisterSensor() operation

Asset: Observation offerings

Cause	Man-In-The-Middle
Effect	SOS will receive fraudulent RegisterSensor() request
Result	SOS will trust fraudulent sensor provided by the attacker and provide offerings based on the sensor.
Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	Direct affect on asset.
Impact on User	Impact to the user of the active client exists as the response is not associated to the originally request.

Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 60: Modify RegisterSensor() request

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent RegisterSensor() response. In particular, the response might contain a fraudulent AssignedSensorId URI.
Result	User might use wrong AssignedSensorId URI to insert observation data (via InsertObservation() request). The correct AssignedSensorId can be misused by the adversary to send fraudulent/fictitious observation data.
Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Direct affect on asset.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 61: Modify RegisterSensor() response

Cause	Adversary's client is able to execute SOS.
Effect	RegisterSensor() operation of the SOS is invoked.
Result	Theoretically, infinite sensors can get registered with SOS. This can be used by the adversary as InsertOffering() requests can be issued for all the fraudulent sensors. Also, these fraudulent sensors can be used by clients/users.
Scope	Application specific knowledge required.
Example	

Likelihood	Low
Impact on Asset	Direct affect on the asset as the sensor can be used to create offerings which are then based on the sensor(s) of the adversary.
Impact on User	Direct impact to all users of the SOS exist as they can use the fictitious/fraudulent sensor.
Reason	Sabotage
Requirement	Access Control to prevent unauthorized registration of sensors.

Table 62: Create RegisterSensor() request

Cause	Eavesdropping
Effect	Adversary's client will send recorded RegisterSensor() request to SOS.
Result	SOS RegisterSensor() operation is invoked. This should not affect the registration table as the request was already processed earlier and should result in an error.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Reason	Sabotage
Requirement	None

Table 63: Replay RegisterSensor() request

Cause	Man-In-The-Middle
Effect	User client RegisterSensor() request will be send to adversary's SOS.
Result	The user will receive a fraudulent response indicating that the registration was successful. All subsequent InsertObservation() requests to the actual SOS will result in a processing error, as the sensor is not registered.
Scope	Application specific knowledge is required.
Example	

Likelihood	Medium
Impact on Asset	None
Impact on User	Direct affect on asset.
Potential	N/A
Reason	Sabotage
Requirement	Authenticity of the response to allow the user client to determine that the response came from another service. Service Authentication.

Table 64: Redirect RegisterSensor() request

Cause	Eavesdropping
Effect	N/A
Result	Adversary obtains detailed information about a sensor and the AssignedSensorId.
Scope	No application specific knowledge is required to exercise this attack.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	None
Potential	The adversary can use the AssignedSensorId to inject fictitious observations to the SOS.
Reason	Espionage
Requirement	Confidentiality

Table 65: Record RegisterSensor() request/response

9.6.5 InsertObservation() operation

Asset: observation offerings

Cause	Man-In-The-Middle
Effect	SOS will receive fraudulent InsertObservation() request

Result	SOS will provide fraudulent observation offerings to other users.
Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	Direct affect on asset.
Impact on User	Direct impact effective to all users of the SOS.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 66: Modify InsertObservation() request

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent InsertObservation response, in particular a fraudulent ObservationId URI
Result	User might use wrong ObservationId URI to obtain observation data (via GetObservationById() request).
Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Direct impact on the further use of the asset effective to the active client.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 67: Modify InsertObservation() response

Cause	Adversary's client is able to execute SOS
Effect	InsertObservation() operation of the SOS is invoked.

Result	The fraudulent observation might overwrite a correct observation. But in order for that to happen, the created request must match an existing AssignedSensorId URI.
Scope	Application specific knowledge required. In particular, the attacker has to know a valid AssignedSensorId URI.
Example	
Likelihood	Low
Impact on Asset	Direct affect on asset.
Impact on User	Direct impact effective to all users of the SOS.
Reason	Sabotage
Requirement	Access Control to prevent unauthorized insertion of observation(s).

Table 68: Create InsertObservation() request

Cause	Eavesdropping
Effect	Adversary's client will send recorded InsertObservation() request to SOS
Result	SOS InsertObservation() operation is invoked and observation is updated with values from the old request.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	Potentially affecting asset. In cases, where new observations have been send by the sensor since the recording time of the replayed request, they get "updated" with the old values.
Impact on User	Impact on all users that access this observation.
Potential	N/A
Reason	Sabotage
Requirement	Unique request Id and time-stamp.

Table 69: Replay InsertObservation() request

Cause	Man-In-The-Middle
Effect	User client's InsertObservation() request will be send to adversary's SOS
Result	The adversary's SOS will receive the observation data.
Scope	Application specific knowledge required, as the request must be answered properly.
Example	
Likelihood	High
Impact on Asset	Direct affect on the asset as the actual SOS has not received the observation data and will therefore serve outdated values to the registered clients.
Impact on User	Impact to all users at registered clients.
Potential	N/A
Reason	Sabotage
Requirement	Service authentication and authenticity of the response.

Table 70: Redirect InsertObservation() request

Cause	Eavesdropping
Effect	N/A
Result	The adversary's client receives the observation data.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	N/A
Reason	Espionage
Requirement	Confidentiality

Table 71: Record InsertObservation() request/response

9.6.6 GetObservationById() operation

Asset: observation offering provided by SOS

Cause	Man-In-The-Middle
Effect	SOS will receive fraudulent GetObservtationById() request sent by the user client.
Result	SOS will provide observation offering to user if the tampered ObservationId is served by the SOS.
Scope	Application specific knowledge is available. In particular, the attacker needs to have a valid ObservationId so that the response contains sabotaged data.
Example	
Likelihood	Medium
Impact on Asset	No direct affect on the asset but the unveiling of the observation based on the tampered ObservationId in case it is served by the SOS.
Impact on User	Impact on the user of the active client as the returned observation is not associated to the actual request.
Potential	N/A
Reason	Espionage
Requirement	Integrity

Table 72: Modify GetObservationById() request

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent GetObservationById response that might contain fictitious observation data.
Result	User does not get the observation data associated with the actual request.
Scope	Application specific knowledge is required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Direct affect on asset.
Potential	N/A

Reason	Sabotage
Requirement	Integrity

Table 73: Modify GetObservationById() response

Cause	Adversary's client is able to execute SOS.
Effect	GetObservationById() operation of the SOS is invoked.
Result	The adversary might receive observation data if the ObservationId of the created request is served by the SOS.
Scope	Application specific knowledge is required. In particular, the attacker needs to have a valid ObservationId.
Example	
Likelihood	Low
Impact on Asset	Unveiling of asset.
Impact on User	None
Potential	N/A
Reason	Espionage
Requirement	Access Control to prevent unauthorized requests.

Table 74: Create GetObservationById() request

Cause	Eavesdropping and adversary's client can execute SOS.
Effect	Adversary's client will send recorded GetObservationById() requests to SOS
Result	Adversary gets observation data, associated with the recorded request.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	Unveiling of asset
Impact on User	None

Potential	N/A
Reason	Espionage
Requirement	Unique request id and time stamp to detect replay.

Table 75: Replay GetObservationById() request

Cause	ARP spoofing
Effect	User client will send GetObservationById() request to adversary's SOS.
Result	The adversary's SOS will receive the request and return fictitious observation data to the user.
Scope	Application specific knowledge is required.
Example	
Likelihood	Low
Impact on Asset	Direct affect on asset.
Impact on User	Impact on the user of the active client as the response will not come from the actual SOS.
Potential	N/A
Reason	Sabotage
Requirement	Service authentication and authenticity on the response.

Table 76: Redirect GetObservationById() request

Cause	Eavesdropping
Effect	N/A
Result	The adversary can obtain the observation.
Scope	No application specific knowledge is required to exercise the attack.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None

Potential	N/A
Reason	Espionage
Requirement	Confidentiality

Table 77: Record GetObservationById() request/response

9.6.7 GetResult() operation

Asset: Observation offering

Cause	Man-In-The-Middle
Effect	SOS will receive fraudulent GetResult() request.
Result	User will get wrong observation based on the modified parameters from the request.
Scope	Application specific knowledge is required. In particular, the adversary has to know the ObservationTemplateId that was created by the SOS as a result of an earlier GetObservation() request.
Example	
Likelihood	Low
Impact on Asset	None
Impact on User	Direct affect on asset if the ObservationTemplateId is valid but not associated to the actual request effective to active client.
Impact on User	Direct effect for calling client
Potential	N/A
Reason	Sabotage, Espionage
Requirement	Integrity

Table 78: Modify GetResult() request

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent observation data.
Result	User will receive observation data that is associated with request but tampered.

Scope	Application specific knowledge is required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Direct affect on asset effective to active client.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 79: Modify GetResult() response

Cause	Adversary is able to execute SOS
Effect	GetResult() operation of the SOS is invoked.
Result	The adversary might receive observation data from the SOS if the ObservationTemplateId is valid.
Scope	Application specific knowledge. In particular, the attacker needs to know a valid ObservationTemplateId served by the SOS
Example	
Likelihood	Low
Impact on Asset	No affect on asset but its unveiling.
Impact on User	None
Potential	N/A
Reason	Espionage
Requirement	Access Control to prevent unauthorized access.

Table 80: Create GetResult() request

Cause	Eavesdropping
Effect	Adversary's client will send recorded GetResult() requests to SOS.

Result	Adversary receives updated observation data associated to the ObservationTemplateId.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	N/A
Reason	Espionage
Requirement	Unique request id and time-stamp to detect replay.

Table 81: Replay GetResult() request

Cause	ARP spoofing
Effect	User client GetResult() request is send to the adversary's SOS.
Result	Adversary's SOS will receive the request and return fictitious observation data.
Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Impact on the user of the active client as the response will come from the adversary's SOS.
Reason	Sabotage
Requirement	Service authentication and authenticity on the response.

Table 82: Redirect GetResult() request

Cause	Eavesdropping
Effect	N/A
Result	Adversary receives observation data and obtains a valid ObservationTemplateID.

Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	None
Potential	Adversary can use the ObservationTemplateId to request updates of the observation based on the template.
Reason	Sabotage
Requirement	Service authentication and authenticity on the response.

Table 83: Record GetResult() request/response

9.6.8 Summary of the Attacks

For all operations of the SOS it is required to ensure integrity of the request and the response to prevent attacks that modify the content. In order to prevent spoofing attacks, the SOS has to authenticate to the client.

- The Create/Record GetCapabilities() attacks unveil sensor metadata that is important information to the adversary to exercise other attacks. In order to mitigate the exploitation of the capabilities to an adversary, the GetCapabilities() operation can be put under access control to ensure only authenticated users can execute the operation and that the response (the capabilities of the SPS) is confidential for the identified user.
- For the DescribeSensor() operation it is required to ensure confidentiality on the request to prevent unveiling of the sensorId to the adversary and the response in order to prevent unveiling of the detailed sensor metadata provided in SensorML or TML. The information could be used by the adversary to register a fraudulent sensor with the SOS.
- The GetObservation() and GetObservationById() operations require access control to prevent unauthorized fetching of observation data. In addition the response element gml:name requires confidentiality in order to prevent the adversary to leverage the Create GetResult() attack.
- The InsertObservation() operation requires confidentiality of the actual observation inserted to prevent recoding. Authentication by the sensor is required to ensire that only trusted sensors can insert and observation for a particular AssignedSensorId.
- The Create RegisterSensor() attack can be leveraged by the adversary to inject fictitious information for a sensor. In order to prevent that, a mutual authentication is required for self-registering sensors to guarantee only trusted sensors are accepted by the SPS. In cases where an administrator registers as

sensor, the RegisterSensor() operation shall contain two identities: The identity of the admin and the identity of the sensor to be registered. The SOS shall establish access control based on the admin's identity to prevent unauthorized registrations and use the sensor identity information for verification that the sensor is trustworthy. The AssignedSensorId element in the response requires confidentiality to prevent the adversary to leverage Create InsertObservation() attack.

As it is essential for the user to get assurance that no observation reported by a sensor is missing, the SOS shall create a sequence number to be incremented each time a sensor inserts an observation. The sequence number can be used by the client to check, if the list of observations is complete.

9.7 Sensor Alert Service

9.7.1 GetCapabilities() operation

Asset: Subscription Offerings

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent subscription offerings from the SAS. This can include fictitious offerings or removed offerings or an empty list of offerings (< SubscriptionOfferingList/>).
Result	User gets fraudulent offerings or does not know about existing offerings.
Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Direct affect on use of asset.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 84: Modify GetCapabilities() response

Cause	Adversary's client can execute SAS.
Effect	SAS GetCapbilities() operation is invoked.
Result	Adversary will get the capabilities of the SAS.
Scope	Application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	The adversary can obtain subscription offerings. This information can be used by the adversary to inject fraudulent alerts using the Advertise() operation and Subscribe() operation to obtain the related alerts.
Reason	Espionage Sabotage as the information from the capabilities can be the baseline for other attacks.
Requirement	None

Table 85: Create GetCapabilites() request

Cause	Eavesdropping and adversary's client can execute SAS.
Effect	Adversary's client sends recorded GetCapabilities() request messages to SAS.
Result	SAS GetCapbilities() operation is invoked and capabilities are returned to the adversary.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	N/A
Reason	Espionage
Requirement	None

Table 86: Replay GetCapabilites() request

9.7.2 Advertise() operation

Asset: Alert Subscription Offerings

Cause	Man-In-The-Middle
Effect	SAS will receive fraudulent advertisements and create fraudulent offerings.
Result	Clients will subscribe to the fraudulent offerings and users will receive fraudulent alerts or no alerts at all.
Scope	Application specific knowledge required.
Example	The adversary changes the location of the sensor in the advertise() message from (51.96,7.607) to (48.8,11,34). This will create an offering with a location of the sensor at (48.8,11,34). If a client subscribes for this offering, it will receive alerts directly from the sensor via the XMPP MUC that report the sensor's location at (51.96,7.607).
Likelihood	Medium
Impact on Asset	Direct affect, as SAS will produce fraudulent offerings.
Impact on User	Direct affect on asset effective to all subscribed clients.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 87: Modify Advertise() request

Cause	Man-In-The-Middle
Effect	User client will receive confirmation with a fraudulent ID and fraudulent XMPP MUC.
Result	User client will connected to the MUC given by the adversary and will either never receive alerts or receive fraudulent alerts.
Scope	Application specific knowledge is required.
Example	
Likelihood	Medium

Impact on Asset	None
Impact on User	Indirect affect on asset as the client will connect to the fraud MUC.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 88: Modify Advertise() response

Cause	Adversary's client can execute SAS.
Effect	SAS will receive fraudulent offering.
Result	Clients/users that subscribe for these offerings will either never receive an alert or receive fraudulent (fictitious) alerts. Depending on the implementation, the SAS might crash due to an overflow of offerings to be managed or perform slow.
Scope	Application specific knowledge required.
Example	
Likelihood	Low
Impact on Asset	Direct affect on asset.
Impact on User	Impact to all users that receive the advertisement.
Potential	PublicationID and XMPP URI can be obtained by the adversary. The PublicationID can be used for cancellation of publications and he XMPP URI can be used for recording alerts.
Reason	Espionage, Sabotage
Requirement	Access Control to ensure only authorized owners of an offering can use this operation.

Table 89: Create Advertise() request

Cause	Eavesdropping and adversary's client can execute SAS.
Effect	SAS will create offering that might be outdated.
Result	If the adversary replays a request after the sensor has send a CancelAdvertisement() message, the SAS will re-create the offering and provide this (dead) offering for which no sensor will send data or the adversary will send data.

	Clients might subscribe to dead or fraudulent offerings.
Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	Affect on all assets that have been cancelled in the meantime.
Impact on User	Impact on all users as all assets that have been cancelled in the meantime become again available.
Potential	N/A
Reason	Sabotage
Requirement	Unique request id and time stamp to detect replay.

Table 90: Replay Advertise() request

Cause	Man-In-The-Middle
Effect	The advertisement of alerts will be received by the adversary's SAS.
Result	The subscribed users will not receive the alert.
Scope	Application specific knowledge required.
Example	
Likelihood	Low
Impact on Asset	None
Impact on User	Impact on all users as the advertisement will not be available at the actual SAS.
Reason	Sabotage
Requirement	SAS authentication and authenticity on the response.

Table 91: Redirect Advertise() request

Cause	Eavesdropping
Effect	N/A
Result	The adversary fetches information important to exercise attacks that allow the injection of fictitious, hence fraudulent alerts and record alerts after subscription.

Scope	No application specific knowledge required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Reason	Espionage and preparation for further attacks.
Requirement	Confidentiality

Table 92: Record Advertise() request/response

9.7.3 RenewAdvertisement() operation

Asset: Alert Subscription Offerings

Cause	Man-In-The-Middle
Effect	Actual SAS will delete existing offering after regular expiration as no renewal was received.
Result	Sensors will send data longer than SAS offering exists. Subscribed clients will still receive alerts via the XMPP channel, as this communication is not effected by the attack. New subscriptions will not take place as the SAS will not provide that offering longer as originally assured by the sensor in the previous Advertise() message.
Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	Direct affect on asset.
Impact on User	Impact on all users as the renewal request will not be received by the actual SAS.
Potential	N/A
Reason	Sabotage
Requirement	SAS authentication and authenticity on the response to allow the client determine that the response came from another service.

Table 93: Redirect RenewAdvertisement() request

Cause	Man-In-The-Middle
Effect	SAS will receive fraudulent RenewAdvertisement() request.
Result	Two extremes are likely: The renewal date (until the sensor will send data) can (i) be very short. This is comparable to the "redirect" attack if the renewal time is extremely close to the current time. The sensor will continue sending data even though the SAS will delete the offering in the very near future. (ii) be pushed into the invite future. The offering in the SAS will then exist forever, but the sensor will stop sending data at the originally posted renewal date.
Scope	Application specific knowledge required.
Example	Attacker changes the RenewAdvertisement() for ID 4711 to end on April 1, 2009 to December 1, 2009. As the SAS returns to the client only "4711 confirmed" and not that the date has erroneously been changed to Dec 1, 2009, the sensor is in the impression that everything went O.K., which is actually wrong! So perhaps at April 1, 2009 the sensor will stop sending data, as it reported to the SAS earlier. But the SAS will keep the offering up until December 1, 2009 not knowing that there will be no more data coming in.
Likelihood	Medium
Impact on Asset	Direct affect on asset.
Impact on User	Impact on all usres as the actual renewal request is tampered before received by the SAS.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 94: Modify RenewAdvertisement() request

Cause	Adversary's client can execute SAS.
Effect	SAS will receive fictitious RenewAdvertisement() request.
Result	If the PublicationID of a fictitious RenewAdvertisement() message matches an existing offering, the SAS would change it accordingly, similar to the "change" attack. Unnoticeable for user clients with existing subscriptions but effective for new subscriptions.

Scope	Application specific knowledge is required.
Example	
Likelihood	Low
Impact on Asset	Direct affect on asset.
Impact on User	Impact on all users that will receive the advertisement.
Potential	N/A
Reason	Sabotage
Requirement	Access Control to allow only advertisement owners to apply this operation.

Table 95: Create RenewAdvertisement() request

Cause	Eavesdropping and adversary's client can execute SAS.
Effect	SAS will receive old renewal requests.
Result	RenewalAdvertisement() request for existing offering that have been processed in the meantime become effect less. If a replayed RenewalAdvertisement() request is received for which the SAS received a CancelAdvertisement() request earlier, the response will be an Exception.
Scope	No application specific knowledge required.
Example	Adversary records a RenewAdvertisement() message that instructs the SAS to change the date on offering 4711 until February 1, 2009. By the end of January, the sensor will renew 4711 until April 1, 2009. If the adversary re-sends the recorded message, the SAS will change the end of the advertisement back to February 1, 2009.
Likelihood	High
Impact on Asset	Direct affect on asset.
Impact on User	Impact on all users as the replayed request overwrites other related requests issued in the meantime.
Potential	N/A
Reason	Sabotage
Requirement	Unique request id and time stamp to detect replay.

Table 96: Replay RenewAdvertisement() request

Cause	Man-In-The-Middle
Effect	Sensor will receive fraudulent response that does not reflect the result of the processing done by the SAS.
Result	Advertisements that have resulted in a processing error will not be available as offerings. But the sensor does not know that as the status might have set to "confirmed" by the attack.
Scope	Application specific knowledge is required.
Example	
Likelihood	Medium
Impact on Asset	Leads to potentially invalid offerings (no data production)
Impact on User	Impact on the client if the response is modified from success to failure This would cause the user to re-initiate the RenewAdvertisement() over and over again.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 97: Modify RenewAdvertisement() response

9.7.4 CancelAdvertisement() operation

From the attacker's point of view, the semantic for this operation is identical to RenewAdvertisement("current time"). Therefore, the possible attacks and effects are identical with the attacks for the RenewAdvertisement() operation as described above with using the current time as a parameter.

Requirement: Access Control to prevent unauthorized cancellation of advertisements.

9.7.5 Subscribe() operation

Asset: Alert Subscription

Cause	Man-In-The-Middle
Effect	User client Subscribe() request is send to the adversary's SAS.
Result	User receives fraudulent or no alerts from the adversary's SAS.

Scope	Application specific knowledge is required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Impact on the user of the active client as the request will not be received by the actual SAS and therefore the user will not receive alerts.
Potential	N/A
Reason	Sabotage
Requirement	SAS authentication and authenticity on the response.

Table 98: Redirect Subscribe() request

Cause	Man-In-The-Middle
Effect	SAS will receive fraudulent conditions for sending alerts to the user client.
Result	The user receives fraudulent alerts on the spoofed MUC or is not able to connect to the spoofed MUC.
Scope	Application specific knowledge is required. In particular, the attacker needs to know how to operate a XMPP server to provide spoofed MUCs to user clients.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Impact on the user of the active client as the request received by the actual SAS is modified and therefore the user will not receive the intended alerts.
Potetial	N/A
Reason	Sabotage
Requirement	Integrity

Table 99: Modify Subscribe() request

Cause	Adversary's client can execute SAS.

Effect	Adversary's client will send Subscribe() requests to SAS.
Result	Adversary can create (theoretically) unlimited offerings which might prevent the SAS to operate properly (too many subscriptions to handle). Exercising this attack frequently can cause slow processing of the SAS.
Scope	Application specific knowledge is required. In particular, the attacker needs to know offerings as contained in the capabilities document.
Example	
Likelihood	Low
Impact on Asset	None
Impact on User	None
Potential	N/A
Reason	Denial of Service
Requirement	Access Control to ensure only authorized users can execute the operation.

Table 100: Create Subscribe() request

Cause	Eavesdropping and adversary's client can execute SAS.
Effect	Adversary's client sends a recorded Subscribe() message to SAS.
Result	If the adversary re-sends the recorded message after the SAS has received a CancelSubscription() message for that subscription, the SAS will keep a subscription (and a MUC) for the client of the adversary.
Scope	No application specific knowledge required unless the adversary wants to receive alerts on the MUC. Then, the attacker needs to know how to use a XMPP client.
Example	
Likelihood	High
Impact on Asset	Unveiling of the asset to the adversary.
Impact to User	None
Potential	N/A
Reason	Espionage
Requirement	Unique request id and timestamp to detect the replay.

Table 101: Replay Subscribe() request

Cause	Eavesdropping
Effect	None
Result	Recorded Subscribe() requests that contain a MUC, can be replayed by the adversary to connect to that MUC and record the published alerts.
Scope	No application specific knowledge is required but the attacker needs to know how to use an XMPP client.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	The adversary can fetch XMPP MUC URI (if provided by the client) to connect to in order to record alerts.
Reason	Espionage
Requirement	Confidentiality on requests that contain a MUC address.

Table 102: Record Subscribe() request

Cause	Eavesdropping
Effect	None
Result	A recorded Subscribe() response that contain both a MUC and a subscription ID, the adversary can disconnect the client from that MUC by sending a CancelSubscription() request message, using the obtained subscription ID.
Scope	Application specific knowledge is required.
Example	
Likelihood	High
Impact on Asset	None
Impact on User	None
Potential	The adversary can fetch XMPP MUC URI to connect to in order to record alerts.
Reason	Espionage

Requirement	Confidentiality

Table 103: Record Subscribe() response

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent response that does not reflect the result of the user processing done by the SAS.
Result	(i) XMPPURI might be fraudulent with the effect that the user client would either not receive any or fraudulent alerts. (ii) The processing status might be fraudulent, e.g. "OK" which would hide processing errors.
Scope	Application specific knowledge is required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on Asset	Impact on user of the active client as he will not receive the MUC information associated to the request.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 104: Modify Subscribe() response

9.7.6 RenewSubscription() operation

Asset: Alert Subscriptions

Cause	ARP spoofing
Effect	User client will send RenewSubscription() request to adversary's SAS.
Result	Actual SAS will delete existing subscription at its regular expiration time as no renewal was received. Subscribed clients will not receive alerts via the XMPP channel after the subscription is expired. In case the used MUC was created by the client, it will exist but the client will not receive any more alerts on that MUC. In case the SAS created the MUC, the client might receive a XMPP error when the

	MUC is closed by the SAS. The response will come from the adversary's SAS and therefore contain a faked processing status, e.g. "OK".
Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Impact on all subscribed users.
Potential	N/A
Reason	Sabotage
Requirement	Access Control to prevent unauthorized renewal of subscriptions.

Table 105: Redirect RenewSubscription() request

Cause	Man-In-The-Middle
Effect	SAS will receive a modified RenewSubscription() request message sent by the client. Basically two modifications can occur: (i) The renewal time is changed to be before or after the actual time in the request. (ii) The new date until the client is expecting alerts can be pushed into the invite future.
Result	If the modified renewal time is changed to be earlier than the actual renewal time, the SAS would close the MUC earlier than expected by the client. If the modified renewal time is changed to be after the actual renewal time, the SAS would keep the MUC but the client will no longer listen to it.
Scope	Application specific knowledge is required.
Example	
Likelihood	Medium
Impact on Asset	None, as SAS terminates abandoned MUCs.
Impact on User	Impact on all subscribed users.
Potential	N/A

Reason	Sabotage
Requirement	Integrity

Table 106: Modify RenewSubscription() request

Cause	Adversary's client is able to execute SAS
Effect	SAS will receive fictitious RenewSubscription() request messages.
Result	If the SubscriptionID of a fictitious RenewSubscription() message matches an existing offering, the SAS would change it accordingly. The adversary can theoretically push all existing subscriptions into the infinite future if either knowing or guessing all valid SubscriptionIDs.
Scope	Application specific knowledge required. In particular, this attack does only make sense if the attacker knows valid subscription IDs.
Example	
Likelihood	Low
Impact on Asset	Direct affect on asset effective to all subscribed clients.
Impact on User	Impact on all subscribed users.
Potential	N/A
Reason	Sabotage
Requirement	Access Control to prevent unauthorized renewal.

Table 107: Create RenewSubscription() request

Cause	Eavesdropping and adversary's client is able to execute SAS
Effect	SAS will receive outdated renewal requests for existing subscriptions.
Result	RenewSubscription() messages that have been processed in the meantime become affectless.
Scope	No application specific knowledge required.
Example	Adversary records a RenewSubscription() message that instructs the SAS to change the date on subscription 4711 until February 1, 2009. By the end of January, the client will renew 4711 until April 1, 2009. If the adversary re-sends the recorded message, the SAS will change the end of the subscription back to February 1, 2009 and stop sending alerts.

Likelihood	High
Impact on Asset	Affect on asset effective to subscriptions that have not been cancelled in the meantime.
Impact on User	Impact on all subscribed users.
Potential	N/A
Reason	Sabotage
Requirement	Unique request ID and timestamp to detect replay.

Table 108: Replay RenewSubscription() request

Cause	Man-In-The-Middle
Effect	User client will receive fraudulent response that does not reflect the result of the processing done by the SAS.
Result	Subscriptions that resulted in a processing error will not be available for sending alerts. Therefore, the status change to "OK" is critical as it hides any errors that might have occurred when the SAS processed the RenewSubscription() message. And because it is hidden to the client, the user cannot undertake relevant actions to correct the error.
Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	Affect on asset effective to the active client only.
Potential	N/A
Reason	Sabotage
Requirement	Integrity

Table 109: Modify RenewSubscription() response

Cause	Eavesdropping
Effect	N/A
Result	The adversary receives SubscriptionId.

Scope	Application specific knowledge required.
Example	
Likelihood	Medium
Impact on Asset	None
Impact on User	None
Potential	SubscriptionID can be used to cancel the associated subscription.
Reason	Espionage and preparation for future Sabotage.
Requirement	Confidentiality

Table 110: Record RenewSubscription() request/response

9.7.7 CancelSubscription() operation

From the attacker's point of view, the semantic for this operation is identical to RenewSubscription("current time"). Therefore, the possible attacks and effects are identical with the attacks for the RenewSubscription() operation as described above with using the current time as a parameter.

Requirement: Access Control to prevent unauthorized cancellation of subscriptions.

9.7.8 Summary of the Attacks

For all operations of the SAS it is required to ensure integrity of the request and the response to prevent attacks that modify the content. In order to prevent spoofing attacks, the SAS has to authenticate to the client. For the XMPP functionality it is required to use MUCs that require the client to authenticate. This prevents from unauthorized injection of alerts and/or the recording of alerts by an adversary. The login information for the MUCs must be send in a confidential manner from the SAS to the client upon Advertise() response or Subscribe() operations.

In order to prevent that the adversary injects old advertisement into the SAS, it is required to include a unique request ID into the Advertise() request.

The operations RenewAdvertisement() and CancelAdvertisement() require to establish access control based on the owner of the PublicationId.

It is not possible to establish access control for the RenewSubscription() and CancelSubscription() as the SubscriptionId cannot be associated to a particular owner; they are potentially associated to many clients.

As potentially many clients share the same MUC it is required to ensure privacy preventing to exploit other clients in the MUC.

9.8 Rate the attacks for the Baseline Services

A rating of the attacks that have been identified in the previous sections can be undertaken from different viewpoints:

- Which attacks does have the most likelihood to be exercised.
- Which of the attacks does have the highest chance to complete successfully.
- Which of the attacks has the highest impact.
- Which of the attacks can be exercised where the attacker cannot be unveiled.
- Which of the attacks are based on information that can be obtained easily.
- Which of the attacks require a network compromise or address fraud.

9.8.1 Likelihood to exercise an attack and likelihood of success

For the context of this ER, two different kinds of likelihood are important:

- the likelihood that an attack can be exercised at all, and
- the likelihood that an exercised attack is actually successful.

In general, it is difficult to estimate the absolute likelihood as it depends on many factors. One factor is how the service implementation works. Therefore, the absolute likelihood for the same attack exercised on different service implementations could be totally different. One good example here is how an SPS creates the task IDs. If it is a strong random number, generated with a very high entropy, the guessing of valid task ids is much harder as if the task IDs are natural numbers in ascending order. Another factor is the correctness and the actuality of the information required to exercise the attack. This becomes in particular important, if the attacker has gathered the information from different sources over a longer time window. For example, the attacker is using task id information from an eavesdropping attack that was running over quite some time. By re-using fetched task ids, a likelihood exists that they are outdated by the time the attack is excercised.

Even though it might be impossible to estimate absolute likelihoods for attacks, it sounds reasonable to provide an estimation of relative likelihood between attacks. This estimation can be used to create a kind of ranking between the different attacks. This can be helpful to determine which of the attacks needs to be taken care of first or with which priority and monetary resources.

In order to rank the attacks by estimating the relative likelihood among them, one important factor is whether the attacker does have all information at hand by simply reading the service specification or if information gathering is required. For the latter, the likelihood drops the more service interactions are required to obtain all required

information. The likelihood drops dramatically further if interactions to other service are required for the gathering of required information.

For example, for all S*S (actually all OGC Web Services) the GetCapabilities() request can be submitted by reading the OWS Common specification. The only information required is the service URL, which can often be obtained from a catalogue service or a Google search. But the cancellation of an assignment with the SPS requires knowing a valid task ID. In order to gather a valid task ID, the attacker has to eavesdrop communication with the service with the goal of fetching task IDs. As the attacker cannot know if the wire-taping attack will unveil a task ID at all, an alternative approach might be challenged: By submitting a number of Submit() requests to the SPS with invalid parameters, each response will contain a task ID and "rejected". From those responses, the attacker might guess task IDs with a good certainty and start exercising the actual attacks.

Another factor is determined by the difficulty of gathering the required information. The easiest way is to read the specification as it is publically available and does contain many examples. If those examples (e.g. for task IDs) are screenshots from input/output from prototype implementations, the attacker has ease of use. If other attacks need to succeed, it is certainly easier to succeed with attacks that do not require compromising the network than attacks that require a specific network configuration tampering.

9.8.2 Impact Discussion

For the outlined attacks from the previous sections, different impacts on the asset can be outlined:

- No affect on asset but its unveiling
  The success of the attack does not change the value(s) or state of the affect but makes its values (content) available to the adversary. For security domains that are required to be protected against sabotage, the attacks with this impact can be tolerated. For a security domain that shall be protected against espionage, this impact is not acceptable.
- Affect to asset effective immediately or at a later time
  The success of the attack does change the value or state of the asset at the time when the attack is exercised or at a later time after the attack has succeeded. For the identified attacks of this ER, an example of an attack with later affect is RenewSubscription() if the tampered time is further in the future than the actual time.
- Affect to asset effective to one entity, some entities or all entities
  The success of the attack does change the value or state of the asset and is effective to a different number of entities. An example where the attack success is affecting one entity is the tampering of the response from the service to the client. For example if the response of the GetStatus() of the SPS is tampered by the attacker, the result becomes visible to the active client only. Attacks that use an operation with write characteristics have potential to affect all other clients interacting with the service. However, some attacks might only be effective to some clients only. The success of tampering the Advertise() request of the SAS

has influence to all clients that are subscribed to that particular alert. Tampering the InsertSensor() request of the SOS is effective to all clients as he SOS cannot provide offerings for the sensor.

9.8.3 Risk discussion

The opening questions is: How long can an attacker exercise attacks before he gets unveiled and how many attacks can an attacker exercise and how many succeeding attacks can an attacker undertake before he gets unveiled?Exercising attacks always carries a certain risk for the attacker to be unveiled. Depending on the kind of the attack and its affect on the asset and its effectiveness, the potential to be unveiled can be determined by the following factors:

- Does the exercising of an attack require the tampering of the underlying network as it is relevant to attacks leveraging Man-in-The-Middle or ARP spoofing. In these cases, it is very likely that the adversary cannot exercise too many attacks, as he must expect the unveiling at almost any time. For attacks that can be carried out by just executing the service (Adversary's client can execute S*S) it is much harder for the network admin to differentiate between a normal request and an attack.
- For a certain attack to succeed, how many other requests must be exercised successfully. The higher that number, the more likely it is that the attacker gets unveiled and suspended from the network, before the actual devastating attack can be exercised.
- All attacks that are related to espionage typically carry less risk to be unveiled than attacks that are related to sabotage. Therefore attacks that do not result in a change of the asset or its state are perhaps less risky than attacks that affect the asset and is also be effective to more than one user.

9.8.4 Overall Rating

In the context of this ER, the overall rating of an attack shall reflect the likelihood that an attack succeeds, its impact and the risk involved for that attacker to get unveiled. To estimate an overall rating seems almost impossible, taking under consideration that the reason why the attacker wants to exercise might vary extremely. However, we can perhaps say that it is proportional to the likelihood that the attack succeeds times the impact divided by number of the attacks potentially required to succeed. But also, the selection of appropriate attacks might still depend on the situation and the context and therefore be different from that rule of thumb.

However, it is possible to say that the attacker is probably favorable to attacks that do not require network tampering, do not require information gathering – at least not from exercising other attacks – and have a high impact. In that sense, all transactional operations of a service are "interesting" as they carry the potential with high impact, if the adversary is up to sabotage.

9.8.5 Attack suitability discussion

Which kinds of attacks provide the most flexibility for the attacker to reach the desired goal?

In case has to make the adversary make his mind up which attack to exercise in order to succeed and reach the desired goal. This is in particular important if different options exist but with different likelihood and involved risk to get unveiled.

In addition to that, the attacker has to make his mind up which attack is suitable to reach the desired goal. Attacks, leveraging the Man-In-The-Middle cause provide in general the maximum flexibility as they can modify the request from the user client to the service and the response going back. Certainly, if it requires to modify the request as the attack aims at changing the asset stored at the service, the adversary has no choice. But if the aim is to provide tampered data to the client, the adversary can exercise an attack that modified the request or the response In that respect, the attacker can categorize the possible attacks in two categories:

- Modifications to the service response
  Attacks based on modification of services responses can be tampered in any respect so long the result is still acceptable to the client and the user is not suspicious about the response. gives more flexibility over With modification of the response, the adversary can change almost anything to influence the user in any desired way.
- Modifications to the service request
  Attacks based on modification of the service request are less flexible, as the response still comes from the service. So the variety of the responses is limited by the processing semantics of the service.

10 Introduction to relevant Security Standards

As discussed earlier, many different requirements exist that need to be met in order to secure the Sensor Web architecture to be used in the intelligence domain. As illustrated in the section “Approach”, Message-Level-Security seems to be an extremely promising security foundation towards accreditation. The following figure lays out the different existing security standards (draft standards and recommendation) that can be used to implement Message-Level-Security in an interoperable way. The figure also shows standards that can be used to secure conversation between network-endpoints, applying security on the Binding- and Network-Layer.

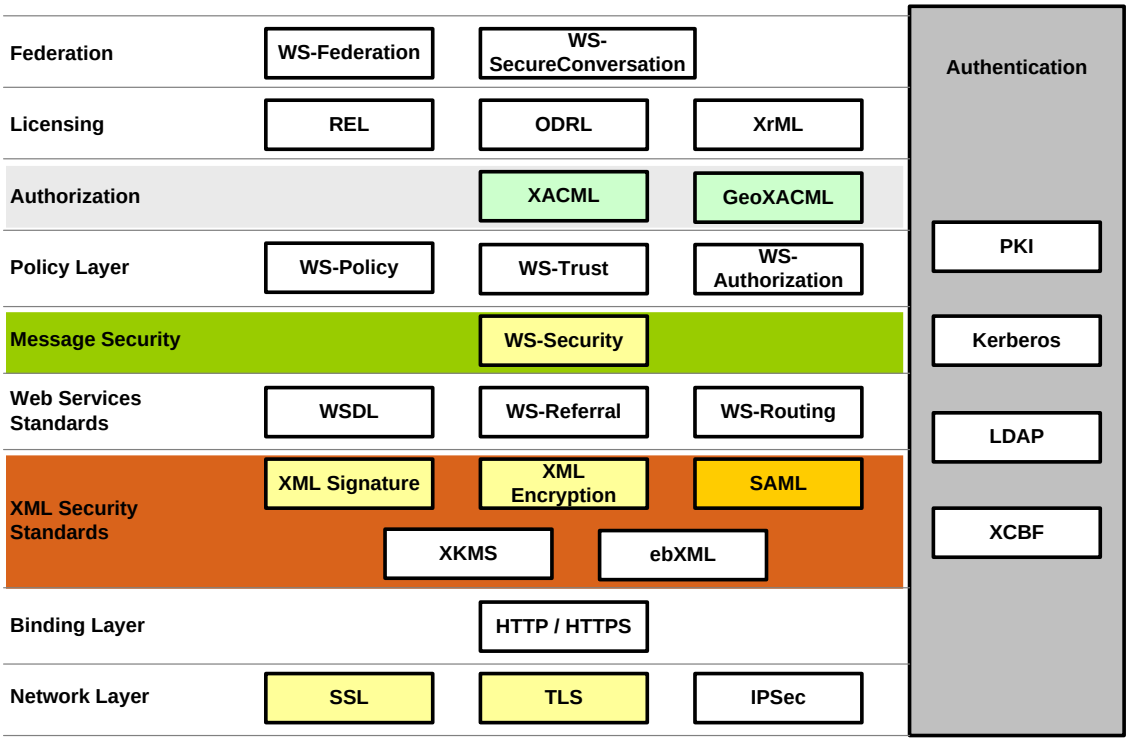

Figure 5: Security Standards Overview (excerpt)

The following sections of this document introduce different standards, draft standards, recommendations and other literature that defines the realization of security requirements for the Network Layer and Message Layer. Both can be applied independent from each other or in combination, depending on the overall architecture and requirements.

10.1 Standards for securing Communication on the Network Layer

This section of the document provides an overview of standards, recommendations and other literature related for establishing secure communication.

10.1.1 IPSec (see [2])

IPSec defines a protocol that secures Internet Protocol (IP) based communication between network endpoints on ISO/OSI layer 3 (network layer). It thereby creates secure tunnels through untrusted/unsecure networks ensuring confidential and authenticated communication. Sites connected by these tunnels form Virtual Private Networks (VPNs).

The following protocols are used in IPsec:

- ESP (Encapsulating Security Payload) is the encrypted information that is transported,
- AH (Authentication Header) provides authentication for data packets and
- IKE (Internet Key Exchange) negotiates connection parameters.

The strength of IPSec is that applications can use the secure communication established (provided) by IPSec without any knowledge. Even though this is a strength, it needs to be remembered that IPSec does not establish an end-to-end secure communication, as it is provided by message layer security. This is important to understand when building a network topology that consists of multiple segments, each using their own IPSec configuration.

10.1.2 TLS / (SSL) (see [3])

The TLS/SSL protocol enables applications to communicate in a point-to-point fashion by establishing a secure communication channel that supports integrity and confidentiality of the exchanged information. It requires that the server authenticates itself. Also, TLS/SSL provides optional mutual (client) authentication, which is almost never used. Based on a challenge request/response handshake that involves asymmetric encryption, the client and server establish (agree on) a shared secret (symmetric key) to encrypt all further communication that is associated to the current session.

Because TLS/SSL secures the entire information that is exchanged between communication partners, it cannot be used if individual parts of one message are or the entire message is confidential for receivers different from the client and the server. Also, transparent proxy connections are not possible.

In addition, the use of TLS/SSL is not sufficient if message repudiation is important, as the encryption is based on a shared secret. Here, message layer protection must be established to enable secure and trusted audit.

10.2 Standards for securing Communication on the Binding Layer

10.2.1 HTTP(S) (see [13])

HTTPS is defined as HTTP over TLS in the IETF RFC 2818. It defines how HTTP leverages TLS to establish a secure communication over the Internet using the https:// URI scheme. Simply speaking is the result of an HTTPS connection communication of encrypted messages using the standard port 443.

10.3 Standards for securing Communication on the Message Security

10.3.1 WS-Security (see [5])

The prime goal of this OASIS specification is to enable secure exchange of XML messages using the SOAP (see [6]) protocol between communication end-points. It provides support implementing message integrity and confidentiality as well as client (user) authentication. This can be obtained by applying XML Digital Signature (see [7]) and XML Encryption (see [8]) to an XML message in a specific fashion. This standard describes the processing rules in order to create message integrity or confidentiality. It also describes the structure of SOAP messages and the structure or relevant metadata so that they can be processed (by web services) in an interoperable way.

This standard also supports different security tokens to obtain client authentication. It defines processing rules of how to attach security tokens to messages. These security tokens are currently supported:

- "Username" token provides support to share knowledge about the identity of a user. "Password" expresses the password associated with this token. In addition, "Nonce" and "Created" are supported to enable strong digested passwords.
- "X.509" token supports exchange and use of X.509 certificates for the matter of authentication, digital signatures and encryption.
- "SAML" include SAML assertions as a token.
- "Kerberos" token allows to the use of Kerberos tickets.
- "REL" token can be used to attach license information.

10.4 Standards associated to Message Content Security

This section of the document provides an overview of standards and recommendations and other literature related for establishing message content security.

10.4.1 XML Digital Signature (see [7])

This W3C Recommendation specifies the processing rules how to apply digital signatures to any type of information; in particular XML structures information and represent the

result as well as the relevant metadata in XML. It supports different kinds of digital signatures:

- "Enveloped" signatures are processed over the content that includes the digital signature element itself.
- "Enveloping" signatures are processed over content that is part of the signature element.
- "Detached" signatures are processed over content that is external to the signature element.

10.4.2 XML Encryption (see [8])

This W3C Recommendation specifies the processing rules how to encrypt information and represent the result as well as relevant metadata in XML. It also defines processing rules for the associated decryption. The following types of encryption are supported:

- "Element Encryption" allows encrypting the embracing element and its name.
- "Element Content Encryption" allows encrypting the value of an XML element which leaves the embracing element name in clear text.
- "Any Data Encryption" allows encrypting entire documents.
- "Super-Encryption" supports to encrypt already encrypted data.

10.4.3 XKMS (see [9])

The XML Key Management Specification is a W3C Note comprises of two sections specifying a XML Key Information Service (X-KISS) and a XML Key Registration Service (X-KRSS) as well as the associated protocols for the distribution and registration of public keys that can be used in conjunction with the W3C Recommendations XML Digital Signature and XML Encryption.

- The Key Information Service Specification describes the protocols that allow an application delegating the processing of XML Digital Signatures (or parts of it) to a trusted service. The application hereby gains simplicity and performance issues concentrate on the trusted service.
- The Key Registration Service Specification describes the protocol to register (and revoke) public keys with a trusted service. The associated private key can be generated by the service or the client. This requires in the first case assertions by the client toward the proof of possession and in the latter case protocol mechanisms for securely sending the private key to the client. In order to allow a meaningful use of public keys and support for cryptographic verification, the client can request that the service registers particular information with a public key.

10.5 Standards for Authentication

This section of the document provides an overview of standards, recommendations and other literature related to authentication and identity management.

10.5.1 X.509 (see [14])

A X.509 certificate is an information bundle where an identity is bound to a public key. The format of the identity can be a X.500 name, an email address or a DNS entry. The information bundle is digitally signed by the CA which guarantees tamper resistance and authenticity. Today, version 3 of X.509 (x.509v3) is been used that allows the use of extension attributes that can be defined as necessary.

X.509 certificates are used to establish HTTPS communications, typically between a web browser and a web server. They are also been used for signing emails, electronic documents such as PDF files or XML formatted messages that are sent by web services.

Because X.509 certificates are based on asymmetric encryption, a private key is associated to the public key. In order to create confidential documents and emails, a X.509 certificate can also been used.

10.5.2 PKI (see [14])

Public Key Infrastructure (PKI) as described in ITU-T standard provides the means by which public keys can be bind to identities in such a way that identification is possible without prior authentication. It also describes management procedures that guarantee that identities are unique throughout the Internet. This can be ensured creating a unique root certificate for each CA and each CA ensures that all maintained identities are unique throughout the CA.

So in a PKI, proof of identity is realized by use of X.509 certificates that are released by CAs. It is therefore essential that a trust relationship with the CA (from which the X.509 certificate is released) is established. This can be set up by accepting (or installing) the X.509 (root) certificate of the CA. With all standard a web browsers, root certificates of all common CAs are pre-installed so that the user does not have to do that.

Beside the management of identities through a certain number of trusted CAs, PKI describes also the means of revocation for X.509 certificates. Each CA maintains a so called Certificate Revocation List (CRL) that contains the (permanently) revoked certificates. Even each certificate has a pre-defined lifecycle that is set by creation, it can perhaps be necessary that the certificate – so the assurance of the CA that a certain identity is bound to the certificate – expires prior to the pre-defined lifecycle. Reasons for revocation are given in the IETF RFC 3280 (see [15]). One reason is that the private key that is associated to the identity has been tampered. Another reason is that a certificate was released for a fraud identity. One well known example was the certificate that was issued to the fraud identity “Microsoft Incorporation”.

10.5.3 Kerberos (see [16])

Kerberos is a Computer Network Authentication Protocol that was developed by the Massachusetts Institute of Technology (MIT) that allows proving of identities between communication partners to each other using a non-secure network. Therefore, Kerberos provides mutual authentication so that the user and the server can verify each other's identity. The protocol protects against eavesdropping (wiretapping) and replay attacks. Today, Kerberos is mainly used for authentication in Microsoft Windows Systems.

Technically, authentication is based on so called Kerberos Tickets. After a successful login at the Authentication Server (AS) using a long term shared secret such as a username / password, the client receives a ticket from the AS. This AS-ticket can then be used to obtain shorter lifecycle tickets to be used with other servers.

10.5.4 LDAP (see [17])

The Lightweight Directory Access Protocol (LDAP) is a protocol for querying and modifying entries of a Directory Service (DS). A DS is a computer program that stores information (typically structured using X.500) about users and computers in a network. Each entry has a unique identifier, called the distinguished name (dn). Each entry can have additional attributes that have a name and a value that – as a whole – define the characteristics of the entry. The stored information is used by administrators to assign roles or access permissions to resources. In an Attribute Based Access Control (ABAC) System, the attributes and their values can be used to derive the authorization decision. In such systems, it is vital to keep the X.500 structure backward compatible.

The LDAP can be used by other authentication protocols to query/exchange identity information.

10.5.5 XCBF (see [18])

The XML Common Biometric Format (XCBF) is an OASIS standard that defines cryptographic messages, based on a common set of XML encodings for the Common Biometric Exchange File Format (CBEFF) that allow the secure collection, distribution and processing of biometric information for the purpose of authentication. In particular, it allows the verification of identity based on human characteristics such as DNA, fingerprints, iris scans and hand geometry.

10.5.6 SAML (see [10])

The Security Assertion Markup Language is an OASIS standard that specifies the structure, the exchange and the processing of assertions about the identity of a subject. An assertion is a structured package of information using the XML notation that is prepared and issued by a so called asserting party and consumed by a so called relying party. Constraints are specified by this standard that allows expressing the restrictions by the asserting party to guarantee appropriate consumption of assertions by the relying party. Also, assertions can be digitally signed to ensure integrity and authenticity. Also, encryption can be applied to make assertions or parts of it confidential. In addition, extension points are defined that allows the extension of assertion to meet project specific needs. Three types of assertions are specified by the standard supporting different use cases at the relying party:

- "Authentication Assertion" provides information about the asserted subject toward the means by which a subject was authenticated, by whom and at what time.
- "Attribute Assertion" provides information about the characteristics of the asserted subject.
- "Authorization Assertion" states that access to a particular resource is to be permitted/denied for the asserted subject.

In regard to exchange (request and response) assertions between the asserting and relying party, this standard specifies the following protocols (relevant excerpt) and the appropriate sequence of messages:

- "Assertion Query and Request Protocol" defines the processing rules of how existing assertions can be queried and the structure of the messages.
- "Authentication Request Protocol" enables the relying party to request assertion statements about the means by which a subject was authenticated.
- "Artifact Resolution Protocol" defines how SAML artefact references can be exchanged instead of the assertions itself.
- "Name Identifier Management Protocol" defines how an asserting party can change the name of an identifier that was previously established and is been used by relying parties.
- "Single Logout Protocol" defines a sequence of message exchange with the goal to terminate all existing sessions of the subject with other relying parties close to real time. However, there is no confirmation message because the logout with all relying parties cannot be guaranteed.
- "Name Identifier Mapping Protocol" defines an exchange of identifier names that can be used to establish identity federations.

An extension to the SAML standard (see [11]) defines the following bindings (relevant excerpt) that define an association of SAML protocol messages to the underlying communication/message protocols for a particular architecture:

- "SAML SOAP Binding" defines how SAML assertions are to be exchanged using SOAP messages and how SOAP header elements are to be used to do so.
- "Reverse SOAP (PAOS) Binding" describes a mechanism where the client is able to act as a SOAP responder or intermediary relevant for implementing the "Enhanced Client or Proxy (ECP) Profile".
- "HTTP Redirect Binding" enables the exchange of SAML messages as URL parameters. In order to ensure the length limit of a URL is not exceeded, message encryption is used. This binding is relevant, where HTTP user agents of restricted capabilities are involved in the message exchange.
- "HTTP POST Binding" defines how SAML messages can be send inside a HTML form using base64 encoding.
- "HTTP Artifact Binding" defines how SAML request and response messages are exchanged using a reference – an artefact. This binding is essential for implementing the "Artifact Resolution Profile".

An extension to the SAML standard (see [12]) defines the following profiles (relevant excerpt):

- "Web Browser SSO Profile" defines how a Single-Sign-On can be established using a (regular) web browser as the client.
- "Single Logout Profile" defines the sequence of messages relevant to ensure that a user is logged out at all participating services.
- "Enhanced Client or Proxy (ECP) Profile" defines the exchange of request/response messages for a client that knows which asserting party to contact and knowing that it supports PAOS Binding.
- "Identity Provider Discovery Profile" defines mechanisms by which a relying party can discover, which asserting parties a principal uses for the "Web Browser SSO profile".
- "Name Identifier Management Profile" defines mechanisms that can be used by the asserting/relying party to associate a different name to a principal.
- "Artefact Resolution Profile" defines a mechanism where client or client interface restrictions exist that prevents the direct exchange of SAML assertions. A SAML artefact a unique (one-time) reference in the Internet, issued by the asserting party that points to a particular assertion stored at the asserting party that can be requested by the relying party.
- "Assertion Query/Request Profile" defines the basic mechanisms to query/request assertions using synchronous communication.
- "SAML Attribute Profiles" defines a unique naming for SAML attributes of "build-in" types such as X.500/LDAP, UUID, DCE PAC and XACML.

10.6 Standards for Authorization (Attribute Based Access Control)

This section of the document provides an overview of standards, recommendations and other literature related to ABAC.

10.6.1 XACML (see [19], [20], [21], [22])

The eXtensible Access Control Markup Language (XACML) as specified in the OASIS standard describes a multi purpose Policy Language that allows the declaration of access rights in XML. It further defines the process of interpreting Policies in order to derive an authorization decision. In addition, it describes structures of request/response messages in XML that allows requesting an authorization decision from a Policy Decision Point (PDP) as it is useful in a Service Oriented Architecture.

Different profiles to XACML exist that define specific use of XACML. The following is an excerpt of important profiles:

- "RBAC Profile" (see [20]) defines how to declare XACML based access rights based on the Role Based Access Control (RBAC) Model. This profile supports RBAC0 (core RBAC) and RBAC1 (hierarchical RBAC). There is no support for RBAC2 (constraint RBAC).
- "SAML Profile" (see [21]) defines extensions to SAML so that XACML specific information can be securely exchanged. The following different extensions are defined:
    - "AttributeQuery" can be used for requesting one or more attributes from an Attribute Authority.
    - "AttributeStatement" defines a standard SAML statement that contains one or more attributes. This statement may be used in a SAML Response from an Attribute Authority, or it may be used in a SAML Assertion as a format for storing attributes in an Attribute Repository.
    - "XACMLPolicyQuery" can be used for requesting one or more policies from a Policy Administration Point (PAP).
    - "XACMLPolicyStatement" defines a SAML statement extension that can be used in a SAML response from a PAP.
    - "XACMLAuthzDecisionQuery" defines a SAML request extension that can be used by a PEP to request an authorization decision from an XACML PDP. This is an alternative to the XACMLAuthorizationDecisionRequest defined in XACML.
    - "XACMLAuthzDecisionStatement" defines a SAML statement extension that can be used in a SAML response from an XACML PDP. This is an alternative to the XACMLAuthorizationDecisionResponse defined in XACML.
- "DSIG Profile" (see [22]) defines a recommendation to exchange authorization decision request and responses based on the SAML Profile for XACML that supports applying digital signatures for the purpose of authentication and establishing message integrity. This is a relevant profile as XACML itself does not support to apply digital signatures to the XACML native authorization decision request and response messages.

10.6.2 GeoXACML (see [23], [24], [25])

The Geospatial eXtensible Access Control Markup Language (GeoXACML) is a standard by the Open Geospatial Consortium Inc. (OGC) that defines a geo-specific extension to XACML v2.0. It extends the XACML Policy Language by the new data type "Geometry" and several geo-specific functions that allow the declaration and enforcement of access rights that can be associated to geometric characteristics of the resource. The two extensions (see [24] and [25]) define particular XML encodings of a XACML AttributeValue element of type Geometry, based on the Geography Markup Language (GML). In particular, GeoXACML extension A provides support for GML2 and extension B provides support for GML3 formatted geometries.

10.7 Standards for Licensing

This section of the document provides an overview of standards, recommendations and other literature related to Licensing/Digital Rights Management.

10.7.1 XrML (see [26])

The eXtensible Rights Markup Language (XrML) is a proprietary XML dialect to express rights over digital content which is been used by Microsoft. It is not a standard and owned by ContentGuard (founded by Microsoft and Xerox) which holds related US patens. XrML version 1.0 is the successor of DPRL (Digital Property Rights Language) developed at Xerox PARC that defines computer work specific rights such as "copy", "backup", etc. Version 2.0 developed by ContentGuard was developed to be medium independent. Version 2.1 of XrML was standardized by ISO as Part 5 of the MPEG-21 standards suite (see next topic).

10.7.2 REL (Mpeg REL) (see [27])

The Rights Expressions Language as specified in ISO/IEC 21000-5 (see [27]) defines an XML dialect to express usage rights through tamper resistant enforceable licenses for moving pictures (MPEG) files. In order to protect the owners' assets, a Digital Rights Management System is required of which REL is one key component.

The kernel part of a license is the Rights Expression that grants defined usage rights to a particular consumer (user). Because the rights of a license are typically enforced on the user's computer the content owner relies on the tamper resistance of the license and of the component that interprets the licensed rights. Assuming a tamper resistant license, the meaning of the granted rights must be shared by the creator of the license (typically the content owner) and the software developer of the (MPEG) player. To ensure this, it is vital to standardize a certain set of rights and their semantics (e.g. play, print) as it is done by this standard.

10.7.3 ODRL (see [28])

The Open Digital Rights Language (ODRL) Version 1.1 is a W3C Note that specifies an Expression Language and the representation in XML. It further defines the semantics of core expressions.

The core entities of the ODRL Language are Assets, Rights and Parties. An Asset represents the content that is to be protected either in physical or digital form. Rights include Permissions that are the actual usage that are allowed on the asset. The Parties represent the end user (consumer) and the Rights holders that typically have been involved in the creation of the content or own it.

The standard defines in the ODRL Data Dictionary Semantics section a set of core rights and their semantics for Permissions, Constraints, Requirements, Rights Holders and Context. This standard also provides extension points for the definition of project specific of data dictionary elements. One example given in the standard is associated to the mobile community, where rights such as “ring” or “send” are relevant.

10.8 Standards for Web Services

This section of the document provides an overview of standards, recommendations and other literature related to securing Web Services.

10.8.1 SOAP (see [29])

SOAP provides the foundation of communication for web services. SOAP defines a particular XML structure that separates the information of a message into a “Header” and a “Body” part. The “Body” part of the message contains the actual information that is to be transported and the “Header” element can keep optional (security related) metadata as it relevant to protect the “Body” information as a whole or partially.

SOAP supports multiple bindings, where the HTTP (and HTTPS) binding is the most common one. It enables the communication between sites using the “standard” WWW port to pass through a firewall.

Based on SOAP, WS-Security defines mechanisms and XML structures how to protect SOAP messages in an interoperable way (so that it can be understand by the receiver) toward integrity and confidentiality using XML Digital Signatures and XML Encryption.

For some use cases, the input and/or output of a web service might be in binary format instead of XML. For these cases, a base64 encoding of the binary data can be transported in the SOAP Body. However this is possible, the base64 encoding increases the size of the information and XML parsing or digital signatures and encryption face a decrease in

performance. In order to exchange binary data via SOAP, SOAP with attachments can be used.

10.8.2 WSDL (see [30])

In order to bind to a web services, its network end points (operations and binding) and the (SOAP) structure of input and output message can be described using the Web Services Description Language (WSDL). More precise, WSDL is a W3C note that defines a model and the XML notation to describe web services to support ease of use by the following elements:

- The “types” element describes the messages that can be received and send by the web service
- The “interface” element contains information about the functionality of the web service
- The “binding” element has the information of how to access the web service
- The “service” element provides the actual network endpoint where the web service can be accessed

WSDL 2.0 supports a full HTTP binding including GET / POST (/ DELETE / PUT / etc.) and SOAP.

10.8.3 WS-Addressing (see [31])

Web Services Addressing is a W3C Recommendation that supersedes the WS-Referral & WS-Routing initiatives by Microsoft. It specifies a transport neutral mechanism to communicate addressing information for messages and service endpoint references. Using SOAP and HTTP(/HTTPS) the sender relies on TCP/IP to route the message to the right receiver. Once delivered, the receiver uses information from the SOAP message itself to figure out what to do with the message. WS-Addressing allows to disconnect this relationship by inserting WS-Addressing metadata information (structured in XML) into the SOAP Header. Looking at it from a security point of view, this enables communication partners to securely exchange synchronous but more important asynchronous (unsolicited) messages. In order to ensure a trusted processing, XML Digital Signature can be applied to make WS-Addressing metadata tamper resistant and authentic.

In “Web Services Policy Attachment for Endpoint Reference (WS-PAEPR)” (see [32]) is described, how to use WS-Policy (see [33]) Information into the Endpoint Reference provided by WS-Addressing. This enables to express service security requirements that ought to be met in order to access (execute) the referenced service.

10.8.4 WS-Policy: (see [33])

Web Services Policy is W3C Recommendation that allows to describe and advertise policies of a web service in XML. A policy can express requirements toward Quality of Service characteristics, privacy considerations, security constraints, etc.

From the standpoint of security, WS-Policy describes the capabilities and constraints of the security policies on intermediary services and end point services such as required security tokens, supported encryption algorithms, etc. WS-Policy also defines how to associate policies with web services. In addition, WS-Policy defines operators to combine and intersect policies.

10.8.5 WS-Policy Attachment (see [34])

Web-Services Policy Attachment is a W3C Recommendation that is based on WS-Policy. It specifies how to derive the effective policy for subjects from “scattered” policies by merging all relevant parts. This is important as constraints can be expressed at different levels (web service, operation, message, communication channel, environment, authorization, cryptographic algorithms, tokens, etc.) that must be taken under consideration at the moment when authorization is enforced.

In addition, this recommendation specifies two general-purpose mechanisms for associating policies to different versions of WSDL and UDDI. Universal Description, Discovery and Integration (UDDU) defines a registry service for publishing, searching and obtaining WSDL documents.

The specified model for attaching WS-Policies to WSDL includes how to partition a WSDL construct into “service”, “endpoint”, “operation” and “message” policy subjects and the semantics for attaching a policy to each policy subject. It further defines how to combine policies for a single policy subject that is attached to multiple WSDL components.

The defined mechanisms for associating policies to policy subjects through the use of UDDI involve two possibilities: Policies can be made available via direct (remote) reference or as tModels registered within UDDI. Independent from the approach, this recommendation defines how to calculate the effective policy.

10.8.6 WS-SecurityPolicy (see [35])

Web Services SecurityPolicy is an OASIS standard that defines a framework that allows to express web services security related constraints and requirements to be used in conjunction with WS-Policy.

In order to support that, WS-SecurityPolicy defines initial sets of assertions that are used by the service to express to the client how messages can be secured. The intent is to be flexible on the one hand side in terms of tokens and cryptographic algorithms but still been expressive to ensure interoperability toward assertion matching between

communication partners. Deriving the applicable policy out of a set of possible alternatives is based on the WS-Policy intersection mechanism and first-level, QName matching.

WS-SecurityPolicy supports the following types of assertions:

- "Protection assertions" define the parts of a message that are to be protected.
- "Conditional assertions" define preconditions of security such as which tokens can be used for integrity or confidentiality or which cryptographic algorithms can be used.
- "Security binding assertions" define how Conditional assertions are to be used to protect messages parts as declared using Protection assertions.
- "Supporting token assertions" define the types of tokens that can be used to secure individual operations of the service or messages.
- "Web Services Security and Trust assertions" define token referencing and additional trust options.

10.8.7 WS-Trust (see [36])

Web Services Trust is an OASIS standard that defines extensions to WS-Security for managing (issuing, renewing, cancelling, validating) security tokens for the purpose of establishing brokered trust relations between web services of communication partners through the exchange of secured messages. For supporting Brokered Trust this standard introduces the concept of a Security Token Service (STS). In order to use the STS in an interoperable way, XML message formats are defined for the messages to request and respond security tokens as well as negotiation and challenging mechanisms.

It is important to note that this specification does not define any security token types. It just specifies how to deal with them to establish trust between web services of not directly trusted communication partners.

10.8.8 WS-SecureConversation (see [37])

Web Services Secure Conversation is an OASIS standard that defines the concept of a Security Context (Security Context Token), how to establish and/or reference it in order to exchange a sequence of messages within a session instead of single messages, as supported by WS-Security. This standard defines three ways of how to establish a security context:

- Security Context Token (SCT) created by a security token service,
- SCT created by one of the communication parties and propagated with a message and
- SCT created by negotiation.

In addition the standard defines mechanisms for amending, renewing and cancelling an established security context. Because the encryption of the messages exchanged within an established security context is based on shared secrets, this standard also defines how to derive keys as well as the refreshing of keys in order to prevent providing too much encrypted data for analysis.

This standard is designed to be used in conjunction with other WS-* standards, in particular WS-Security and WS-Trust.

10.9 Draft Standards for Web Services

This section gives a short overview of current initiatives and draft standards in the area of security for web services and secure communication.

10.9.1 WS-Reliable Messaging (see [38])

WS-Reliable Messaging is an OASIS Draft that aims at providing modular mechanisms for reliable exchange of messages regardless to network failures. The defined SOAP based messaging protocol provides support to identify, track and manage reliable transfer of messages between a sender and a receiver.

This draft defines an extensible mechanism which use is anticipated with WS-Security standards such as WS-Policy to integrate other security requirements in an interoperable way.

10.9.2 WS-RM Policy (see [39])

Web Service Reliable Messaging Policy defines policy assertions applicable for reliable messaging to be used with WS-Policy and WS-Reliable Messaging.

The Sequence Security Policy assertion (extending WS-SecurityPolicy assertion) of this draft standard enables the destination and the source of a reliable communication to express the security requirements, particularly relevant for a sequence of messages.

10.9.3 WS-MakeConnection (see [40])

Web Services Make Connection is an OASIS Committee Draft that describes a mechanism to deliver a message between two endpoints if the sending end-point cannot establish a connection to the receiving end-point. In order to achieve this, WS-MakeConnection defines a mechanism to uniquely identify non-addressable endpoints. It does this for the SOAP binding.

This committee draft (specification) integrates with WS-Security, WS-Policy and WS-ReliableMessaging that supports the realization of security related aspects. Because the use of WS-Security secures messages by applying asymmetric keys, the performance might become an issue for large messages or high message throughput. WS-MakeConnection allows the use of WS-Trust and WS-SecureConversation to negotiate a shared secret (symmetric key) to encode messages.

10.9.4 WS-Federation / WS-Authorization / WS-Privacy (see [41])

Web Services Federation Language as of version 1.2 is an OASIS Editors Draft that defines mechanisms to protect resources from one security realm to subjects of another security realm. This requires a federation between the two security realms (identity and resource) such that the origin of authentication assertions from the authentication realm can be trusted by the access control realm. WS-Federation builds on WS-Trust to ensure this.

In addition, it is essential to ensure secure exchange of messages between the trusted realms. WS-Federation builds WS-Security to ensure this.

It is important to note that the federation mechanisms defined in this document are not limited to SOAP enabled Web Services; the Web Browser Environment is also supported. This is achieved by providing an HTTP encoding of the WS-Trust messages Request Security Token (RST) and Request Security Token Response (RSTR).

WS-Federation builds on Security Token Services (STSs) to exchange relevant security information. In order to ensure interoperability to an Authentication Service, this document defines a common profile of the STS as defined in WS-Trust. In addition, this document defines additional XML elements to become part of the RST that allows further specification of the authorization context in which a security token is requested.

Upon requesting a security token it might often be the case that some related information is private to a person or an organization. In order to obtain a security token that contains private information, the requestor can ask the provider to encrypt the private information. In order to express these constraints, this document defines an additional XML element for the RST message.

10.10 Standards for eBusiness

This section of the document provides an overview of standards related to electronic business.

10.10.1 ISO/TS 15000 (see [45], [46], [47], [48], [49])

This multi-part international ISO standard defines the electronic business eXtensible Markup Language (ebXML) that provides support for an interoperable exchange of

messages to facilitate global trade. In order to achieve the linking of business processes, each part of the standard defines certain (technical and non-technical) aspects such as Information Transfer, Meaning and Process. The main concern with Information Transfer is the safe and reliable exchange of information (messages) over the (un-secure) Internet. The Meaning aspect establishes a common (identical) understanding of the exchanged information about the order and/or deliverable. The Process aspect is related to the standardization of sequence of actions concerning messages to be sent and orders to be fulfilled. In addition, ebXML defines the structure of an ebXML registry, where process, messages and data definitions can be stored. In addition the standard defines mechanisms that guarantee inter-registry communication for the purpose of synchronisation.

Part 1 defines the collaboration-protocol profile (ebCPP) that can be used for business transactions between business communication partners. Part 1 also defines the agreement specification (CPA) that can be used as a message exchange agreement between the business partners. The CPA defines the minimum agreement toward message, communication security constraints, that are created by the intersection of the business partners' CPPs. The CPA also contains a binding to a Process Specification document that defines the interactions between the business partners, specific to the actual business collaboration.

Part 2 defines a communications-protocol (ebMS) neutral method for exchanging electronic business messages that ensures the reliable and secure delivery of business messages. In particular the ebXML message structure is defined and the behaviour of the message handling services that are used to send and receive ebXML messages. In order to achieve that, the ebXML SOAP Envelope extension is defined and the Reliable Messaging protocol is leveraged to ensure the once-and-only-once message delivery semantics.

Part 3 defines the registry information model (ebRIM) in which the term "repository item" is used to identify the actual information object that is stored in the registry (e.g. XML document) and the "RegistryEntry" which is used to refer to metadata about a repository item. The information, stored in an ebXML registry can be used to facilitate ebXML-based B2B partnerships or transactions. The Registry Information Model defines what types of objects are stored in the registry and how the stored objects are organized in the registry. It acts as a blue print for implementers to decide which types to include into the registry and which attributes and methods the actual objects might need. The actual Registry Information Model is provided as UML diagrams, in which different classes and their association are introduced: RegistryObject, Slot, Association, ExternalIdentifer, ExternalLink, ClassificationScheme, ClassificationNode, Classification, RegistryPackage, AuditableEvent, User, PostAddress, EmailAddress, Organization, Service, ServiceBinding and SpecificationLink.

Part 4 defines how to build ebXML registry services (ebRS) to provide access to the information stored in an ebXML registry. It therefore defines interfaces for the registry service, the interaction protocol and message structures.

10.11 ISO Standard for Common Criteria for Information Technology Security Evaluation (abbreviated as Common Criteria or CC)

This section of the document provides an overview of standards related to the security evaluation, abbreviated Common Criteria.

10.11.1 ISO/IEC 15408 (see [50], [51], [52])

This multi-part international ISO Standard defines what is well known as Common Criteria for Information Technology Security Evaluation (CC).

Based on this standard it is possible to compare the results of independent security evaluations for products such as operating systems, computer networks, distributed systems and applications. It supports that by providing a common set of requirements for security functions for a product to be certified and for applied assurance measures. The result of the security evaluation undertaken by competent and independent licensed laboratories that document how much the security requirements of a product meet the requirements might provide a help to the customer for evaluating if a product is suitable. CC knows seven assurance levels:

- EAL1: Functionally Tested
- EAL2: Structurally Tested
- EAL3: Methodically Tested and Checked
- EAL4: Methodically Designed, Tested and Reviewed
- EAL5: Semiformally Designed and Tested
- EAL6: Semiformally Verified Design and Tested
- EAL7: Formally Verified Design and Tested

In other words, ISO/IEC 15408 provides the capabilities for customers to specify certain security requirements, product (soft- and hardware) vendors can claim certain to have implemented those requirements and independent certification bodies can conduct tests on the product to actually proof the claim(s). A list of certified products according to the Common Criteria is available at http://www.commoncriteriaportal.org. For example, the "Interactive Link Data Diode Device" from Tenix Pty Limited, Sydney, Australia is the only product with the assurance level EAL7. It is used to separate high and low classified networks ensuring a secure unidirectional data flow to the high classified network only.

In particular, ISO/IEC 15408 can be applied to certify that products are not vulnerable to human or system initiated actions that cause the unwanted disclosure, (unnoticed) modification or loss of information processed or stored by a certified product. Therefore, this standard allows to certify that information confidentiality, integrity and availability is ensured.

This ISO International Standard is presented as three parts:

Part 1 (Introduction and general model) provides the introduction to ISO/IEC 15408, defines general concepts and principals for IT security evaluation and a general model for evaluation.

Part 2 (Security functional requirements) "defines the required structure and content of security functional components for the purpose of security evaluation" [51]. It also includes "a catalogue of functional components that will meet the common security functionality requirements of many IT products and systems" [51].

Part 3 (Security assurance requirements) defines the evaluation assurance levels and defines a scale for measuring assurance. It also contains the criteria for evaluation of assurance of Protection Profiles and Security Targets as specified in Part 2. For example, it defines assurance through evaluation by different techniques such as "verification of proofs" or "penetration testing". In addition, it defines assurance scales to state the minimal effort required to reach a particular assurance scale.

10.12 Standards for Security Techniques

This section of the document provides an overview of standards related to the security assurance.

10.12.1 ISO/IEC 15443 (see [53], [54], [55])

This international multi-part ISO Standard categorizes security assurance methods to a generic lifecycle model in order to gain high level of confidence when certifying security functionality of a deliverable. A deliverable in the context of this standard can be related but is broader than the definition of a TOE as defined in ISO/IEC 15408. Part 1 of this standard provides general definitions, an overview and a framework for assurance methods. Part 2 defines different assurance methods. Part 3 analyses different assurance methods and their applicability to the lifecycle: Concept/Specification, Design/Development, Integration, Deployment and Operation.

Part 1 defines three categories of assurance methods for the assessment of the deliverable, the process used to develop the deliverable and the environment such as personnel and facilities. It is stated that the selection of the right assurance method can be different for the same deliverable if the environment changes and that specific assurance methods can only be applied to certain time periods of the lifecycle.

Part 2 defines different security evaluation criteria for different markets and a visualization how it is to be used and to which timeframe of the lifecycle it applies to. For example, chapter 6.12 defines the "ITSEC/ITSEM Evaluation Criteria and Methodology for the European market". Its visualization is =>D=>, =>I=> and =>O=> meaning that it is applicable to Product/System/Service Design/Implementation, Integration/Verification and Operation.

Part 3 defines (as one most important aspect) which assurance approach will provide the most reliable results fitting the needs of the Assurance Authority. It therefore illustrates

the difference between Product vs. Product, Process vs. Environment and Product vs. Environment assurance. It also gives the (relative) value for each Assurance Approach indicating how applicable it is to the context of the Assurance Authority and how to deal with assurance of complex deliverables such as a combination of hard- or software components, security services, environmental aspects or any combination of them.

10.13 Standards for Open Systems Interconnection

This section of the document provides an overview of standards related to the definition of security requirements and concepts.

10.13.1 ISO/IEC 10181 (see [56], [57], [58], [59], [60], [61], [62])

This international multi-part ISO Standard defines security frameworks for Open System environments. It defines that "Open Systems" include Database, Distributed Applications, Open Distributed Processing (ODP) and Open Systems Interconnection (OSI). Security Frameworks are defined in order to provide protection for systems and objects within the systems as well as interactions between systems. The concept of Security Frameworks of this standard is meant as the base for further detailed specification in the other parts.

Part 1 describes the organization of Security Frameworks, defines relevant security concepts and describes relationships of the services of the frameworks. It hereby uses security architecture definitions from ISO/IEC 7498-2 such as access control, availability, denial of service, digital signature and encipherment. It also provides other relevant definitions such as security information, security domain, security policy, trust entities, trust and trusted third parties. For the security information it defines security labels, cryptographic checkvalues, security certificates and security tokens. In addition, it defines denial of service and availability in such a sense that a denial of service can not always be prevented. In theses cases, other security services can be used to detect the lack of availability and allows to apply corrective measures. Annex A of Part 1 provides an example of protection measures for security certificates.

Part 2 of this standard defines all aspects of Authentication in Open Systems and the relationship with other security functions such as access control.

Part 3 of this standard defines all aspects of Access Control in Open Systems as it applies to the interactions of user to processes, user to data, process to process and process to data. It also defines the relationships to other security functionality such as authentication and audit.

Part 4 of this standard refines all aspects of non-repudiation and extends the concepts defined in ISO/IEC 7498-2.

Part 5 of this standard defines confidentiality as a service "to protect information from unauthorized disclosure" in retrieval, transfer or managed.

Part 6 of this standard defines integrity as a property that “data has not been altered or destroyed in an unauthorized manner”. This applies to data in retrieval, transfer or management.

Part 7 of this standard defines the basic concepts of, a general model for and identifies relationships between services for security audit and alarms.

In addition, Part 1 defines the key management framework as its functions are applicable to any information technology environment where digital signatures and encipherment is used.

10.14 Other Literature

10.14.1 WS-MDE (see [42])

Web Services Metadata Exchange (WS-MetadataExchange) is a draft specification document that fits into the WS-* standards from OASIS but is not published by OASIS. It defines how service specific metadata that describes the conditions for establishing communication can be requested as a WS-Transfer resource. Therefore, the document defines the structure of a GetMetadata and Metadata element that can be inserted in a regular SOAP message. In addition, this document provides several mechanisms to aid service endpoints and requestors in bootstrapping communication, issuing a HTTP/GET request. The document strongly recommends the use of WS-Security to secure messages so that the exchanged metadata can be relied on.

10.14.2 WS-Transfer (see [43])

Web Services Transfer (WS-Transfer) is a W3C member submission that defines a SOAP based mechanism for acquiring, creating and deleting XML-based representations of entities using a web services infrastructure. More specific, it defines operations to Get, Put, Create and Delete representations of resources. Therefore, the document defines “Resources” that are addressable entities providing an XML representation and “Resource Factories” are web services that can create a new resource from an XML-based description.

10.14.3 WS-RT (see [44])

Web Services Resource Transfer (WS-RT) is a draft specification document that fits into the WS-* standards from OASIS but is not published by OASIS. It specifically defines extensions to WS-Transfer that allows to operate on fragments of resource representations using the WS-Transfer operations Get, Put, Create and Delete. In order to achieve that, it defines the QName and XPath Expression Dialect.

10.15 Applicable standards to implement the different Requirements

As illustrated and discussed earlier, a Service Oriented Architecture (SOA) can be understood as a network of distributed self-contained software components (services) that provide simple business functionality that can be orchestrated to build complex applications. Due to the nature of the architecture, it is important that users can relay on the availability of the services and the information exchanged. Therefore, the most common security requirements for a Service Oriented Architecture are Availability of services and Information security. In order to apply security to a Service Oriented Architecture, a core set of functionality as defined in ISO/IEC 10181 (see [56] - [62]) is required: Authentication, Access Control, Non-repudiation, Confidentiality, Integrity, Audit and Alarms.

The Availability of services is important to ensure that the provided functionality can be used at any time. This can be achieved by taking care of safety issues as they are associated to any operating system. For a SOA, it is also important that denial of service attacks do not cause any harm. This can be achieved by using certified (fail-safe and vulnerable-free) software components. The "Common Criteria" standard ISO/IEC 15408 (see [50] - [52]) provides good and solid facts for evaluating and comparing secure software products. In addition ISO/IEC 15443 (see [53] - [55]) defines concrete assertion criteria for different products depending on the context (use) of the product. Therefore ISO/IEC 15408 and 15443 can in conjunction been used to select safe software components.

In order to orchestrate services to accommodate electronic business, it is important to integrate business process across jurisdictions. ISO/TS 15000 (see [45] - [49]) defines ebXML, a framework and an XML dialect to do so. For example in a secure Sensor Web, services of different organizations (different security domains) might have the need for accounting and compensating of common used/shared observation data. This integration of monetary transactions can be integrated using ebXML.

In a modern service oriented architecture, the communication with a service / between services takes place using XML formatted messages that are structured according to the SOAP (see [6]) recommendation and services and their operations are described in the Web Services Description Language (WSDL), a W3C Note from 2001. The realization of different security requirements using message level security is possible by extending the SOAP protocol.

An interoperable realization of the above core functions from ISO 10181 can be achieved by using (a combination of) appropriate standards:

10.15.1 Authentication

The important standard for exchanging authentication information is the Security Assertion Markup Language (SAML) (see [10]). For this ER, it can be assumed that identity management systems are in place that provide localized (nationwide) authentication. For the interoperable and secure exchange of identity information on a project level, SAML can be used. In addition, XML Common Biometric Format (XCBF)

(see [18]) – can be used to collect, distribute and process biographic identity information. SAML integration with LDAP (see [17]) is possible in a seamless manner.

10.15.2 Access Control

Access control is one aspect to secure information while stored on a service where it is accessible for users and other services. For cross domain / cross jurisdiction access control, it is important to use a standard that supports the interoperable exchange and the collaborative process to define access rights as well as the (automated) electronic enforcement. The eXtensible Access Control Markup Language (XACML) from OASIS (see [19] - [22]) or the Geospatial eXtensible Access Control Markup Language (GeoXACML) from OGC (see [23] - [25]) support that. SAML also supports the means of Kerberos (see [16]) based authentication.

10.15.3 Digital Rights Management (DRM)

DRM provides functions to control the use and ensure the unauthorized disclosure of classified data even after it is obtained and stored on the local computer. This is in particular important for the observation data that is produced for dual use. This persistent protection can be achieved by a DRM- or GeoDRM-System. In addition to strong encryption to protect the data, licenses must be issued that contain the usage rights that prevent unauthorized use and disclosure. Up to now, Rights Expression Language standards for the expression of non geo-specific licenses exist; see [26] - [28]. There is (as of today) no standard available for expressing licensed rights for geospatial data.

10.15.4 Confidentiality

For a secure Sensor Web, two different aspects of confidentiality must be taken under consideration: (i) confidentiality of information while in transit and (ii) confidentiality of classified information.

- Confidentiality of information while in transit must be ensured when exchanging messages with services over insecure networks. This can be achieved on the network layer using IPSec ([2]) or VPN. However, this solution has shortcomings and might not always be possible as it depends on the constraints of the network topology. For single connections where end-to-end confidentiality is sufficient, HTTPS (see [3] and [4]) can also be used.
  Another solution that is independent from the security constraints of the network and its topology is provided by message level security. Based on SOAP messages, WS-Security (see [5]) defines how to apply XML Encryption (see [8]) to the information or parts of it. XML Encryption (and XML Digital Signatures) is based on X.509 (see [14]) and relies on a Public Key Infrastructure that can be established and maintained

using XKMS (see [9]). Whenever using X.509 certificates, revocation mechanisms are essential s defined in (see [15]).

- An information flow control must be established as part of the persistent control for confidentiality of classified information. For the Secure Sensor Web, the traditional flow control in the intelligence domain between two different classified networks through a network data diode is not applicable. This is, because the creator of a sensor tasking request can decide how much of the task information is confidential and to which other entity. Again, ABAC with XACML or GeoXACML can be used to ensure the correct flow of information according to the Bell-La Padula model independent from the network topology.

10.15.5 Integrity

For a secure Sensor Web, three different aspects of integrity must be taken under consideration: (i) integrity of information while in transit, (ii) integrity of information as part of common sensor tasking requests and (iii) integrity of produced results.

- Integrity of information while in transit must be ensured when exchanging messages with services over insecure networks. This can be achieved on the network layer using IPSec or VPN. However, this solution has shortcomings and might not always be possible as it depends on the constraints of the network topology. For single connections where end-to-end integrity is sufficient, HTTPS can also be used. Another solution that is independent from the security constraints of the network and its topology is provided by message level security. Based on SOAP messages, WS-Security defines how to apply XML Digital Signatures to the information or parts of it.

- Whenever a user creates a sensor tasking request that is dedicated for common use, it is important to ensure that certain information cannot be modified by others. In order to ensure integrity of information according to the Biba Model, ABAC and XACML or GeoXACML for geospatial information can be used. As an alternative, Digital Signatures can be applied.

- Whenever the result of a sensor tasking request is ready to be obtained, it is important that the information can never be modified without notice. This can be applied by digitally signing the information before storage or sending over the network. WS-Security and XML Digital Signature (see [7]) can be used to achieve this.

10.15.6 Non-repudiation

For a secure Sensor Web, different scenarios exist where non-repudiation of communication is applicable. For example, the creator of a sensor tasking request likes to be sure that the task is received by the operation control centre and that he gets informed upon completion. Also, non-repudiation is required for communication of classified

produced observation data. Here, the client must acknowledge the receiving of downloaded observation data.

Non-repudiation is also important with financial transactions associated to commercial use of dual-use observation data. In order to ensure non-repudiation, trusted audit is required. In addition, the OASIS Committee Draft WS-Reliable Messaging (see [38]) and the standards WS-Security (see [5]), WS-Trust (see [36]) and WS-Addressing (see [31]) can be used.

10.15.7 Audit and Alarms

For the purpose of creating trusted log-files for the purpose of audit, it is essential to have a 3rd party that stamps communicated messages with tamper resistant information at the sender and receiver side of the communication. This protocol functionality is one possible implementation for ensuring non-repudiation of a communication. There is currently no standard know that defines how to do this for the purpose of secure messages. However, the principal is that specific metadata of the communication (or even the entire message traffic) is digitally signed by the trusted audit components, which ensures integrity of the logged information.

For a Policy based Access Control system based on XACML or GeoXACML Policies, certain conditions can be defined to fire off alarms. This enables to inform personnel in charge (administrators) of certain violations to enforced policies.

In addition, it is important for a secure Service Oriented Architecture that communication with services, resp. between services takes place only if certain conditions are met. WS-Policy (see [33]), WS-Policy Attachment (see [34]) and WS-SecurityPolicy (see [35]) can be used to express these constraints and WS-MakeConnection (see [40]) can be used establish a secure communication. WS-MetadataExchange (see [42]) can be used to structure messages that are relevant to be exchanged during the connection negotiation sequence. The pure use of WS-Security is limited to use asymmetric encoding of messages to ensure confidentiality and integrity, which has a drawback on performance. In order to use symmetric encoding to ensure message integrity and confidentiality for a sequence of messages, WS-SecureConversation (see [37]) can be used.

10.16 Implementing Integrity and Confidentiality

For possible future certification, basically three different architectural alternatives exit to implement confidentiality and integrity:

10.16.1 Rely on secure Network and Access Control

The certification of this approach is based on the fundaments of existing procedures for certifying network security. Therefore, no additional cryptographic functionality is required to secure messages in transit towards integrity and confidentiality. By doing so, it is assumed that no man-in-the-middle attacks will take place on messages in transit, because the communication takes place over a secure communication.

In order to secure the integrity and confidentiality of information, while stored on services, Access Control is required. After authentication by username/password, a user can access certain information as defined in the Policy of the associated Access Control System. Towards certification it is important to note that authentication is also not based on keys.

The requirements for certification and the required certification procedure should be evolved.

10.16.2 Secure Messages in Transit based on PKI and Access Control

The certification of this approach is based on an existing, already certified Public Key Infrastructure (PKI). Based on the keys of the PKI, optional message integrity and confidentiality can be ensured by applying the WS-Security standard from OASIS.

In addition, the X.509 certificates of the PKI can be used for authentication to proof identities for users, clients, applications and services. This can be one part of information, relevant for Access Control.

The requirements for certification and the required certification procedure should be evolved.

10.16.3 Use of Security Token Service and Access Control

The use of a Security Token Service (STS) enables two things:

- It provides security tokens that can be used for ensuring message integrity and confidentiality and
- It provides authentication tokens and provides identity pseudonyms, relevant for identity federation.

This approach does not require a key management like the PKI, because the STS will provide applicable tokens for different means.

The requirements for certification and the required certification procedure should be evolved.

11 Discussion of the applicability of the security requirements and their relationship to the identified attacks

11.1.1 Applicability of Authentication

In general, the authentication of communication partners is required for implementing access control and authenticity.

In order to mitigate or prevent certain identified attacks, the implementation of authenticity is relevant for the following operations and services:

SPS: GetCapabilities(), Submit(), DescribeResultAccess(), GetFeasibility(), GetStatus(), Update(), and Cancel()

SOS: GetCapabilities(), GetObservation(), RegisterSensor(), InsertObservation(), GetObservationById(), and GetResult()

SAS: GetCapabilities(), Advertise(), RenewAdvertisement(), Subscribe(), and RenewSubscription()

11.1.2 Applicability of Access Control

In order to mitigate or prevent certain identified attacks, the implementation of access control is relevant for the following operation and services:

SPS: Submit(), DescribeResultAccess(), GetStatus(), Update(), and Cancel()

SOS: GetObservation(), RegisterSensor(), InsertObservation(), GetObservationById(), and GetResult()

SAS: Advertise(), RenewAdvertisement(), CreateSubscription(), and RenewSubscription()

11.1.3 Applicability of Data Integrity

In order to mitigate or prevent certain identified attacks, the implementation of data integrity is relevant for the following operations and services:

SPS: GetCapabilities(), DescribeTasking(), Submit(), DecribeResultAccess(), GetFeasibility(), GetStatus(), Update(), and Cancel()

SOS: GetCapabilities(), DescribeSesor(), GetObservation(), RegisterSensor(), InsertObservation(), GetObservationById(),and GetResult()

SAS: GetCapabilities(), Advertise(), RenewAdvertisement(), Subscribe(), and RenewSubscription()

11.1.4 Applicability of Confidentiality

As defined in ISO 10181-5, the purpose of implementing a Confidentiality Service is to ensure that certain information is only available to authorized entities. The main purpose of the service is to protect the information in a persistent manner by preventing disclosure of the information while the information is

- stored in a system,
- maintained, and
- in transit between communication entities

11.1.4.1 Mechanisms to protect stored information

One way of protection for stored information and information in transit can be applied by using encryption. Then, the confidentiality service relies on key management and the Access Control service that controls access to the keys, which can be used to decrypt the information.

An alternative way of protection that can also be applied as an additional protection mechanism leverages Access Control to control access to the information. Applying access control also to encrypted information limits the set of entities that can obtain the confidential information in the first place.

11.1.4.2 Mechanisms to protect information in transit

Protecting information in transit towards confidentiality can be applied by securing the communication between entities on different levels of the ISO/OSI stack. In cases where encryption is applied to the transport layer (e.g. TLS or SSL), the end-to-end confidentiality is only that strong as the weakest network segment over which the information is sent. This has implication for ensuring confidentiality for classified information. Here, the maximum end-to-end clearance is only that high as the lowest clearance of any network segment over which the message could be routed. The lack of protection based on the clearance of the communication channel can be compensated, by applying appropriate encryption to the information itself, before sending it.

11.1.4.3 Mechanisms to protect the flow of information

Protecting confidential information where different users can read/delete/modify the information, it is essential to control the passing on of the information to unauthorized users by removing the property of the information that it is confidential. The well-known model for implementing the information flow control towards confidentiality is known under the term “Bell – La Padula Model”.

11.1.5 Applicability to the Sensor Web

In order to mitigate or prevent certain identified attacks, the implementation of Confidentiality is relevant for all attacks with reason espionage and for those operations which provide relevant input by eavesdropping to exercise a certain attack. For example

the adversary likes to cancel all sensor assignments with a certain SPS requires to know task IDs. It is relatively easy to fetch task ids as they are part of many request and response messages and communicated in the clear. Applying confidentiality to the task ID eliminates the risk that an adversary fetches valid task IDs by Eavesdropping.

11.1.6 Applicability of Non-Repudiation

Non-Repudiation is a security requirement that does make sense only for systems that have implemented other basic security requirements. It does not make sense to talk about the applicability of non-repudiation for the baseline services of this ER as they are not implementing any security requirement.

11.1.7 Applicability of Security Audit and Alerts

Security Audits and Alerts are in principal applicable to the baseline services, as they support administrators to determine the fact that attacks are exercised or have been. For example would the auditing of the execution of the GetStatus() operation of the SPS that returns an error because the task ID from the request does not exist and the request come in with a high frequency (burst), this could trigger an alert for the administrator.

12 Notification pattern based communication and Firewalls

During the cross thread activities in OWS-6 between the Aeronautical Information Model (AIM) and the Sensor Web Enablement (SWE) threads some issues regarding notification pattern based communication (also called push-based communication) and firewalls arose. More precise, the issues were discovered in context of the integration of the SWE Event Service implementation into the AIM scenario.

The question that evolved was: „How can events be delivered to a consumer[4] whose client is behind a firewall?“. In order to answer this question, notification pattern communication is introduced and the problems for different security solutions that make use of firewalls are discussed.

This report does not specify solutions how a notification pattern based communication can be established in the different environments but presents general approaches to the problem.

12.1 Notification pattern based communication

Notification pattern based communication is an interaction pattern used in Event Driven Architectures (EDAs). In contrast to request-response based communication it is initiated by the data source (the publisher) every time new data is available. This behavior allows transmitting notifications (new data) as soon as possible without the need of (partly unnecessary) requests or the possibility to miss important notifications. Thus notification

[4] The consumer (or target) of a notification pattern based communication may be a client but can also be a service that for instance processes the received data and publishes the results.

pattern based communication is an important means in highly reactive and event driven applications such as early warning systems.

12.2 Firewall and NAT

Networks in a private household are usually secured by a firewall and Network Address Translation (NAT) integrated in a router. In addition also personal software firewalls on each computer may be used. This security solution in general works if communication is initiated from clients inside the private network.

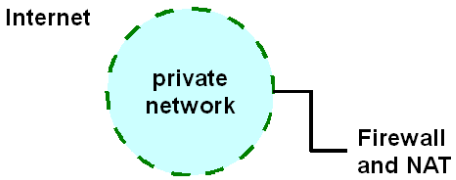

Figure 6: Private Network protected by one Firewall

When trying to establish notification pattern based communication, the incoming notifications are typically rejected by the firewall. In order to permit the communication requested from the outside, every firewall has to be configured to accept incoming communication. This is usually done on a port basis. Furthermore the Router has to be configured to forward the incoming notifications to the desired consumer (computer). This is necessary because the publisher does only know the external (IP) address of the router but not the internal address of the actual consumer. By adding a static route from the router to a particular computer on the private network, incoming notifications can be delivered to the desired consumer.

12.3 Perimeter networks

More sophisticated security solutions make use of perimeter networks (also called Demilitarized Zone, DMZ). Therefore the network is split in two parts (the inner network and the DMZ) secured using two firewalls (the inner and the outer firewall).

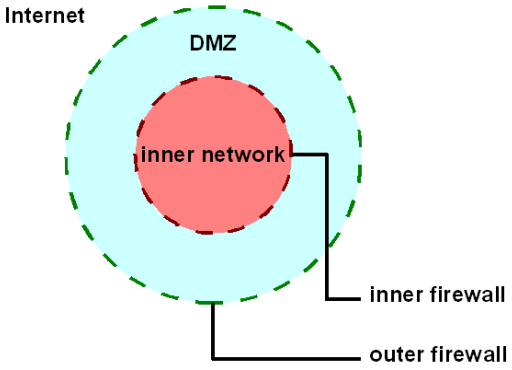

Figure 7: Firewalls with Perimeter Network

The inner network is secured by a highly restrictive (inner) firewall blocking all direct access to the Internet. Depending on the configuration, communication between the

perimeter network and the inner network is permitted or not. If a component located in the inner network wants to access a resource in the Internet the communication has to be transferred via a proxy in the perimeter network. Furthermore the inner firewall only allows communication initiated by components of the inner network to the perimeter network.

The outer firewall secures the perimeter network from the Internet. If this firewall allows notification pattern based communication, it has to be configured as described above to permit the communication into the perimeter network. In this network a proxy server has to be installed accepting (and possibly checking) the incoming notifications. The notifications cannot be forwarded to a consumer inside the inner network directly if the firewall restricts such communication.

12.4 More restrictive solutions

In more restrictive firewall settings it might be impossible to deliver events to a client in the private network at all. This is the case if communication is permitted through one of the firewalls at a time. In this case a component in the inner network has to send its requests to an agent located in the perimeter network when the inner firewall is permitting communications and the outer firewall is rejecting communication. When the outer firewall permits communication, the inner firewall blocks communication and the agent performs the submitted requests. The results are stored at the agent and can be requested when the firewalls switch back again. In such a system near relative communication is never possible, as the consumer and the publisher do not know the times when the inner or outer firewall gets opened.

13 Recommendations

As a result of this work, mainly based on the evaluation of vulnerabilities and potential attacks for the current OGC Sensor Web Services specifications, we like to give recommendations for implementing a Secure Sensor Web.

First of all we like to point out that it is important to implement all relevant requirements and not just one. For example, when implementing access control but the communication is not secured, an attacker could steal security context information such a a session token or an identity token which would most likely cause the access control system to grant requests, based on the stolen security context information.

Of course, it is not easy to say in general which requirements are to be implemented and in which way as this depend on many factors: (i) The architecture itself and which services are deployed in which security domain, (ii) is there direct trust relationship between security domains, (iii) which information/observation shall the system deal with and is it classified, etc. One dominant questions is “Do I have to use WS-Security with SOAP or can I do HTTP+TLS”? This mainly depends on the architecture and the orchestration of services. But as a rule of thumb, it is a good idea to use WS-Security and SOAP, even though the other variant using HTTP+TLS might also be applicable.

We like to point out the following recommendations knowing that there will always exist specific cases where these recommendations might not represent the most elegant solution. However, the list of recommendation can be understood as a framework for securing the Sensor Web.

13.1 Use Message Level Security

W think, that the sufficient and relevant level of ensurance for implementing all relevant security requirements as outlined in earlier sections of this ER can only be undertaken by leveraging message level security. The alternative – network level based implementation – is difficult to achieve as it very much depends on underlying network administration. And, because the joining and leaving of parties in a Secure Sensor Web can much easier be reflected and administrated based on security implementation on message level. The last argument for message level security is that the continuous protection of information exchanged among services provided by different organizations is to be ensured, independent of different network segments that are potentially owned and administrated by different parties using their own policies and service chains that might be required.

Independent of implementing security requirements such as Integrity, Confidentiality and Authenticity requires the existence of a Public Key Infrastructure (PKI). Even integrity and confidentiality could be guaranteed without a PKI, it is recommended to use X.509 certificates.

For the implementation of trusted Audit and Alarms, it is also relevant to have a PKI in place and trusted third parties that undertake and guarantee the correctness and confidentiality of audit information.

13.2 Services shall support SOAP and WS-Security

In simple architectures that are based on transparent chaining, only direct communication links exist between the client and many services. Here, the use of HTTPS can be leveraged in order to secure the communication and the security context information that is exchanged between the client and the secured service.

But in a more generic architecture, where a translucent or even opaque chaning of services shall be possible, the use of SOAP based interfaces leveraging WS-Securtiy for securing the communication is recommended. Depending on the security model that is to be supported, X.509 certificates or other types of tokens must be created. This requires naturally the existing of the associated infrastructure.

13.3 Describe security constraints for the service using WS-Policy/SecurePolicy

Whenever a secured service, e.g. a Sensor Web Service, is provided, it is important to give the caller (client or another service) relevant and sufficient information to understand the security constraints in order to execute the service. It is recommended to use WS-Policy or WS-SecurityPolicy to describe the constraints of the service. The link to such a policy document can be hooked into the WSDL description of the service.

13.4 Protect Transient Handles

Whenever a client (hence operator) of a sensor web has to achieve a particular takes, it typically involve to call different interfaces of either the same service or of different services. As the services are stateless, the relevant state information must be stored on the client side. This is done by so called transient handles. As the undertaken evaluation has shown, it is extremely important to protect these handles towards tampering and unauthorized reading as the effect can have different consequences on the asset.

For example, the SAS returns a PublicationID, SubscriptionID and an XMPP MUC URI; the SOS returns SensorID, AssignedSensorID, ObervationID, ObservationTemplateID; the SPS returns sensorID and taskID to the client. Transporting these transient handles in clear text creates the vulnerability that they can be read by an attacker and used in the context of another request. It is therefore necessary to protect this part of the message in a specific way. We thing that the use of SAML profiles is a sound approach, but we also thing that further investigations are relevant to proof this.

13.5 Support Asset/Identity based Access Control

In general, all interface operations except the GetCapabilities operation should be protected by access control. But as we have shown in the evaluation sections of this ER, the access is not simply to be regulated for registered users. Instead, the Discretionary Access Rights Management must be used to regulate the access. For example, upon a successful Submit() operation with the SPS, the created task is owned by the caller. And

therefore, only the owner – or all identites to which the owner has granted rights to – can call Cancel() or Modify() for that task. Any other user will receive deny.

13.6 Support Single-Sign-On and Identity Management Federations

We see that Sensor Web Services are mainly used with the Transparent Chaining option. This means that for protected services, the user has to be authenticated before access rights can be evaluated. For the ease of use, we recommend to establish a kind of Identity Management Federation with the support of Single-Sign-On. One good example is provided by Shibboleth, as it is used in various academic federations to show services among members of the different participating research and education institutions. One key feature of a Shibboleth based federation is the support for Single-Sign-On. We recommend to further explore the possibilities to integrate secured/procted OGC Sensor Web Services with a Shibboleth based Identity Management Federation as it is for example available in the UK, US, Australia and Germany.

13.7 Use Open Source Software but not out of the box

We recommend not to deploy Open Source Software out of the box, hence using the default installation script. It is likely that this procedure would carry different kinds of potential vulnerabilities such as XML validation vulnerabilities or unveiling of private exception information. In addition, we recommend to undertake a code review to eliminate or mitigate the vulnerabilities that exist by programming short comings. One example based on experience from creating this report is that the taskID in an SPS implementation was created in such a way that the pattern could easily be unveiled by the attacker. This might have the consequence that the attacker is able to successfully guess task IDs and use them with operations of the SPS to either cancel or modify a task.

13.8 Improve current Sensor Web Services Specifications

One result from the ER unveils that the quality of the baseline standards is extremely poor if it comes to define error processing, which information to return to the client and what is a safe state for a service after an attack was exercised.

We recommend that the SAS, SOS and SPS Standards Working Group improve the normative description of error handling.

In order to recommend the development of a series of Change Requests to reach the goal of matured interface specifications that are ready for Secure Sensor Web Services.

14 Outlook and Future Work

14.1 SAML Profiles

One important aspect is to guarantee integrity and confidentiality as well as authenticity of Sensor Web Service requests and responses. As SAML is a standard that describes mechanisms that allow the implementation of the requirements above, we like to encourage further investigations how SAML profiles can be created for security exchange messages with Sensor Web Services. This might include the service interfaces that support the registration of sensors, the publishing of alerts, the storing of and obtaining of observations as well as the tasking of sensors (including the process of evaluating feasibility). Examples how to define SAML profiles for securely exchange Policy Queries as well as Authorization Decision Requests and Responses can be found in the SAML 2.0 profile for XACML 2.0[5].

14.2 How can the proposed approach for securing the OGC Sensor Web be used in a multi-nation project?

For multi-nation projects, it shall be assumed that instances of Secure Sensor Services are deployed in different security domains that are hosted on different networks that are sovereign to different nations. Here, the problem arrives where to deploy the identified security services that apply integrity and confidentiality to classified messages. We think that each security domain must host a certain set of security services. As we assume the Internet Thread Model, all communication inside the security domain is protected to the highest level possible. It is therefore feasible to assume that all Secured Sensor Web Service instances leverages the security services of its security domain to request appropriate protection of outgoing messages and verification of incoming messages. Thinking towards accreditation, "only" the identified security services need to be evaluated. This seems to be more feasible than including the security functionality in each secured Sensor Web Services, as then all of the services need to pass evaluation criteria.

However, the challenge of configuring the security services and maintenance for distributing keys remains. Proposing a solution to do this maintenance and secure configuration was out of scope for this ER but could be a potential work item for future work.

[5] http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf

