Date: ~~December 23, 2014~~September 17, 2015

Automated Source Code Security Measure (ASCSM)

Beta ~~2~~1.0

OMG Document Number: **ptc/2015-09-04**~~admtf/2014-06-07~~
Standard document URL:
http://www.omg.org/spec/ASCSM/20150904/AutomatedSourceCodeSecurityMeasure - change bars~~http://www.omg.org/spec/ASCSM~~
Machine consumable files:
http://www.omg.org/spec/ASCSM/20150815/AutomatedSourceCodeSecurityMeasure.xmi

http://www.omg.org/spec/ASCSM/20150816/AutomatedSourceCodeSecurityMeasureSMM.xmi
~~http://www.omg.org/spec/CWE25SM/20140608/AutomatedSourceCodeCWESANSTop25BasedSecurityMeasure.xmi~~
~~http://www.omg.org/spec/CWE25SM/20140609/AutomatedSourceCodeCWESANSTop25BasedSecurityMeasureSMM.xmi~~

Field Code Changed

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an OMG specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG), a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. CISQ AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL CISDQ, THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT,

INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 250 First Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

IMM®, MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , MOF™ , OMG Interface Definition Language (IDL)™ , and OMG SysML™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

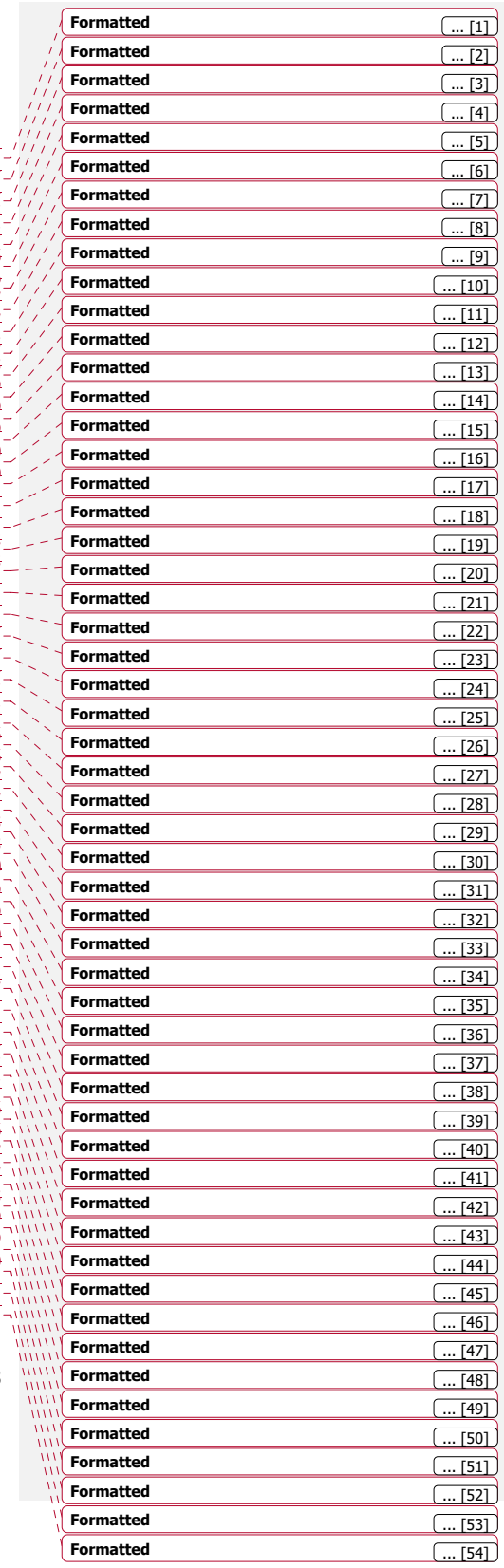
OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http:// www.omg.org*, under Documents, Report a Bug/Issue (*http://www.omg.org/report_issue.htm*).

Table of Contents

ASCSM-91: Reorder CWEs numerically -- reorder sequence of CWEs to be in numerical order

Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font, Check spelling and grammar



Formatted: Default Paragraph Font, Check spelling and grammar

Formatted: Default Paragraph Font, Check spelling and grammar

Formatted: Default Paragraph Font

Formatted: Default Paragraph Font

Formatted: Default Paragraph Font

Formatted: Default Paragraph Font

Formatted: Default Paragraph Font

Formatted: Default Paragraph Font

Formatted: Default Paragraph Font

Formatted: Default Paragraph Font

Preface

About the Object Management Group

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG″s specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG″s specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at http://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A Specifications Catalog is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm
Specifications are organized by the following categories:

Business Modeling Specifications

Middleware Specifications

OMG Modeling Specifications
- **CORBA/IIOP**
- **Data Distribution Services**
- **Specialized CORBA**

IDLO/Language Mapping Specifications

Modeling and Metadata Specifications
- **UML, MOF, CWM, XMI**
- **UML Profile**

Modernization Specifications

Platform Independent Model (PIA), Platform Specific Model (PSM), Interface Specifications
- **CORBAServices**
- **CORBAFacilities**

OMD Domain Specifications

CORBA Embedded Intelligence Specifications

CORBA Security Specifications

All of OMG"s formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
109 Highland Avenue
Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO/IEC standards. Please consult http://www.iso.org

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier - 10 pt. Bold:` Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

Note – Terms that appear in *italics* are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to *http://www.omg.org/ report_issue.htm*.

1 Scope

1.1 Purpose

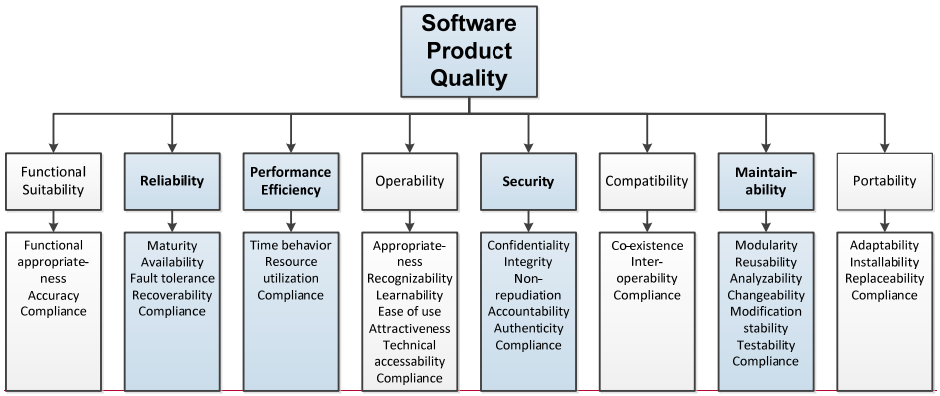
ASCSM-7: Shorten and revise section 1.1--Shorten section to a single paragraph that is revised to cover only the purpose of the measure

The purpose of this specification is to establish a standard measure of security based on detecting violations of good architectural and coding practices that could result in unauthorized entry into systems, theft of confidential information, and the malicious compromise of system integrity. Establishing a standard for this measure is important because such measures are being used in outsourcing and system development contracts without having an approved international standard to reference. They are also critical to other software-intensive OMG initiatives such as the Internet of Things Consortium.

~~The purpose of the Consortium for IT Software Quality (CISQ) is to develop specifications for automated measures of software quality characteristics taken on source code. These measures were designed to provide international standards for measuring software structural quality that can be used by IT organizations, IT service providers, and software vendors contracting, developing, testing, accepting, and deploying software applications. Executives from the member companies that joined CISQ prioritized Reliability, Security, Performance Efficiency, and Maintainability to be developed as measurement specification.~~

~~CISQ intends to maintain consistency with ISO/IEC standards to the extent possible, and in particular with the ISO/IEC 25000 series that replaces ISO/IEC 9126 and defines quality measures for software systems. In order to maintain consistency with ISO/IEC 25010, software quality characteristics are defined for the purpose of this specification as attributes that can be measured from the static properties of software, and can be related to the dynamic properties of a computer system as affected by its software. However, the 25000 series, and in particular ISO/IEC 25023 does not define quality characteristic measures at the source code level. Thus, this and other CISQ quality characteristic specifications will supplement ISO/IEC 25023 by providing this deeper level of software measurement.~~

~~Companies interested in joining CISQ held executive forums in Frankfurt, Germany; Arlington, VA; and Bangalore, India to set strategy and direction for the consortium. In these forums four quality characteristics were selected as the most important targets for automation—reliability, security, maintainability, and performance efficiency. These targets cover four of the eight quality characteristics described in ISO/IEC25010. Figure 1 displays the ISO/IEC25010 software product quality model with the four software quality characteristics selected for automation by CISQ highlighted in blue. Each software quality characteristic is shown with the sub-characteristics that compose it.~~



~~**Figure 1. Software Quality Characteristics from ISO/IEC 25010.**~~

~~This specification defines a method for automating the measurement of Security from violations of secure architectural and coding practice in source code. These violations were drawn from the Common Weakness Enumeration (CWE) maintained by Mitre Corporation, a cyber-security community repository of over 800 known weaknesses in software that can be exploited for unauthorized intrusion into a system. This specification was developed from the CWE/SANS Institute Top 25 most commonly exploited weaknesses, nineteen of which can be detected in source code. The CWE/SANS Top 25 Most Dangerous Software Errors provides a list of the most widespread and commonly exploited 25 security anti-patterns and associated rules that can be found at http://cwe.mitre.org/top25/#Listing.~~

1.2 Overview of Software Quality Characteristic Measurement

ASCSM-9: Shorten and revise section 1.2 -- Shorten section 1.2 to provide briefer overview of quality characteristic measurement that is focused on the type of measure proposed.

Measurement of the internal or structural quality aspects of software has a long history in software engineering (Curtis, 1980). Software quality characteristics are increasingly being incorporated into development and outsourcing contracts as the equivalent of service level agreements. That is, target thresholds based on quality characteristic measures are being set in contracts for delivered software. Currently there are no standards for most of the software quality characteristic measures being used in contracts. ISO/IEC 25023 purports to address these measures, but only provides measures of external behavior and does not define measures that can be developed from source code during development. Consequently, providers are subject to different interpretations and calculations of common quality characteristics in each contract. This specification addresses one aspect of this problem by providing a specification for measuring one quality characteristic, Security, from the source code. This specification is one of four specifying source code level measures of quality characteristics. The other three specify quality characteristic measures for Security, Performance Efficiency, and Maintainability.

The most recent advance in measuring the structural quality of software is based on the analysis and measurement of violations of good architectural and coding practice that can be detected by statically analyzing the source code. The CWE/SANS 25 and OWASP Top Ten lists of security weaknesses are examples of this approach. These lists are drawn from the Common Weakness Enumeration (CWE)

repository maintained by MITRE Corporation. CWE contains descriptions of over 800 weaknesses that represent violations of good architectural and coding practice in software that can be exploited to gain unauthorized entry into a system. The Software Assurance community has been a leader in this area of measurement by championing the detection of code weaknesses as a way of improving one aspect of software quality—software security.

Unfortunately there are no equivalent repositories of weaknesses for Reliability, Performance Efficiency, or Maintainability. Knowledge of these weaknesses is spread across software engineering textbooks, expert blogs, and information sharing sites such as github. An OMG standard for Reliability can fill the void for a consensus body of knowledge about the most egregious Security problems that should be detected and remediated in source code.

Using violations of good architectural and coding practices in software quality metrics presents several challenges for establishing baselines. Growth in the number of unique violations to be detected could continually raise the bar for measuring quality, reducing the validity of baseline comparisons. Further, different vendors will detect different sets of violations, making comparisons difficult across commercial software quality measurement offerings. One solution to this problem is to create a stable list of violations that are used for computing a baseline for each quality characteristic. The Automated Source Code Security Measure was developed by a team of industry experts to form the basis for a stable baseline measure.

~~Measurement of the internal or structural quality aspects of software has a long history in software engineering. Internal software quality measurement can be traced to pioneering work in the 1970s by Halstead (1976), McCabe (1977), Boehm et al. (1978), and McCall et al. (1977). Curtis, et al. (1979a,b) empirically validated that structural quality measures could predict developer comprehension, defect detection times, development time, and accuracy of modifications. Currently there are two primary approaches to measuring internal software quality characteristics — measuring structural elements and measuring anti-patterns.~~

~~Software quality characteristics are increasingly being incorporated into development and outsourcing contracts as the equivalent of service level agreements. That is, target thresholds based on quality characteristic measures are being set in contracts for delivered software. When thresholds are not met the supplier is subject to rework or financial penalties. Currently there are no standards for most of the software quality characteristic measures being used in contracts. ISO/IEC 25023 purports to address these measures, but only provides measures of external behavior and does not define measures that can be developed from source code. Consequently, providers are subject to different interpretations and calculations of common quality characteristics in each contract. This specification addresses one aspect of this problem by providing a specification for measuring one quality characteristic, Security from the source code. This specification will be accompanied by specifications for three other quality characteristic measures; Reliability, Performance Efficiency, and Maintainability.~~

~~***Structural Elements*** — The first and historical approach is based on counts of the structural elements of software. Halstead's Software Science, McCabe's Cyclomatic Complexity, Henry and Kafura's information flow metrics, and Chidamber and Kemmerer's Object-Oriented Metric Suite are examples of measurement based on formulas derived from counts of various structural elements.~~

~~Counts of structural characteristics have a 20-30 year history and are backed by numerous validation studies (Curtis, 1980). Counts of structural elements do not of themselves constitute a defect in the software. Rather they are indicators of potential defects or problems. That is, the probability that the code possesses defects or will be the site of future defect injections increases with higher values of these software quality characteristic measures. Consequently, these measures are often used to set threshold values that, when exceeded, require the offending component to be remediated.~~

~~***Anti-patterns*** – The second, and more recent addition to assessing the structural quality of software is based on the analysis and measurement of anti-patterns – violations of good architectural and coding practice that can be detected by statically analyzing the source code. The CWE/SANS 25 and OWASP Top Ten lists of security vulnerabilities are examples of this second approach. The Software Assurance community has been a leader in this area of measurement by championing the detection of anti-patterns as a way of improving one aspect of software quality—software security. Although the Software Assurance community has developed methods for scoring the severity of individual vulnerabilities, standards have not been developed for calculating component or application-level security measures that aggregate security-related anti-patterns detected through static code analysis into application-level security measures.~~

~~The use of anti-patterns in quality characteristic metrics presents several challenges for establishing baselines. Growth in the number of anti-patterns could continually raise the bar for measuring quality, reducing the validity of baseline comparisons. Further, different vendors will have different sets of anti-patterns they detect, making comparisons difficult across commercial software quality measurement offerings.~~

~~One solution to this problem is to create a stable list of anti-patterns that are used for computing a baseline for each quality characteristic. The Automated Source Code Security Measure uses the CWE/SANS Top 25 anti-patterns to form the basis for a stable baseline measure. For each quality characteristic~~ such a list would provide a minimum set of anti-patterns that must be included in calculating the attribute measure.

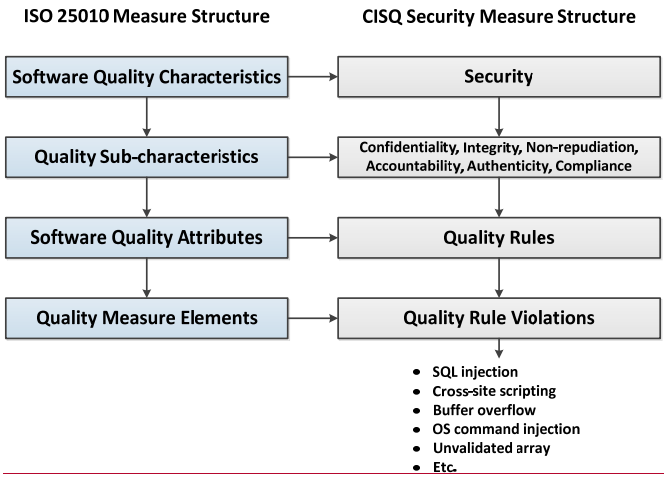
1.3 Development of the Automated Source Code Security Measure

ASCSM-11: Shorten and revise section 1.3 -- Shorten section 1.3 and focus it only on the process through which the measure was created.

The Consortium for IT Software Quality (CISQ) was formed as a special interest group of OMG to create specifications for automating standard measures of software quality attributes and submit them to OMG for approval. The Objective of the Consortium for IT Software Quality (CISQ) is to develop specifications for automated measures of software quality characteristics taken on source code. These measures were designed to provide international standards for measuring software structural quality that can be used by IT organizations, IT service providers, and software vendors contracting, developing, testing, accepting, and deploying software applications. Executives from the member companies that joined CISQ prioritized Reliability, Security, Performance Efficiency, and Maintainability to be developed as measurement specification.

The original 24 CISQ member companies decided to base the security measure on an existing security community body of knowledge concerning exploitable weaknesses. This specification defines a method for automating the measurement of Security from violations of secure architectural and coding practice in source code. These violations were drawn from the Common Weakness Enumeration (CWE) maintained by Mitre Corporation, a cyber-security community repository of over 800 known weaknesses in software that can be exploited for unauthorized intrusion into a system. This specification was developed from the CWE/SANS Institute Top 25 most commonly exploited weaknesses, nitwenty-two of which can be detected in source code. The CWE/SANS Top 25 Most Dangerous Software Errors provides a list of the 25 most widespread and commonly exploited security anti-patterns and associated rules that can be found at http://cwe.mitre.org/top25/#Listing.

~~ISO/IEC 25010 defines a quality characteristic as being composed from several quality sub-characteristics. Each quality sub-characteristic consists of a collection of quality attributes that can be quantified as Quality Measure Elements. Figure 2 presents an example of the ISO/IEC25010 quality measurement framework using a partial decomposition for the Automated Source Code Security Measure.~~



~~**Figure 2. ISO/IEC 25010 Framework for Software Quality Characteristics Measurement**~~

~~The non-normative portion of this specification describes the security issue underlying each of the CWEs included in the Automated Source Code Security Measure. These issues were then translated into software security rules worded as architectural or coding practices or conventions to avoid the problem described in the security issue. These rules were then transformed into software quality measure elements by counting violations of these practices and conventions. These violations of secure architectural and coding practices constitute software anti-patterns.~~

~~The normative portion of this specification represents each quality measure element developed from a CWE in the Implementation Pattern Meta-model for Software Systems (IPMSS). The calculation of the Automated Source Code Security Measure from its quality measure elements is then represented in the Structured Metrics Meta-model. This calculation is presented as the simple sum of quality measure elements without being adjusted by a weighting scheme.~~

~~There are several weighting schemes that can be applied to structural quality measures. The most effective weighting often depends on the measure's use such as assessing operational risk or estimating maintenance costs. The CWE-based anti-patterns included in this specification were considered to be severe violations of secure architectural and coding practices that would need to be remediated. Therefore, weightings based on severity would add little useful information to the measure since the variance among weights would be small. CISQ will provide guidelines for applying weighting schemes for varied uses, but decided they should not be part of this specification.~~

1.4 Structure of the Automated Source Code Security Measure

ASCSM-13: Insert new section 1.4 for structural information --
Insert new section 1.4 to contain all information about the structure of the measure. Renumber the CWE/SANS Top 25 Weaknesses section to become 1.5

ISO/IEC 25010 defines a quality characteristic as being composed from several quality sub-characteristics. This framework for software product quality is presented in Figure 1 for the eight quality characteristics presented in 25010. The quality characteristics and their sub-characteristics selected for source code measurement by CISQ are indicated in blue.

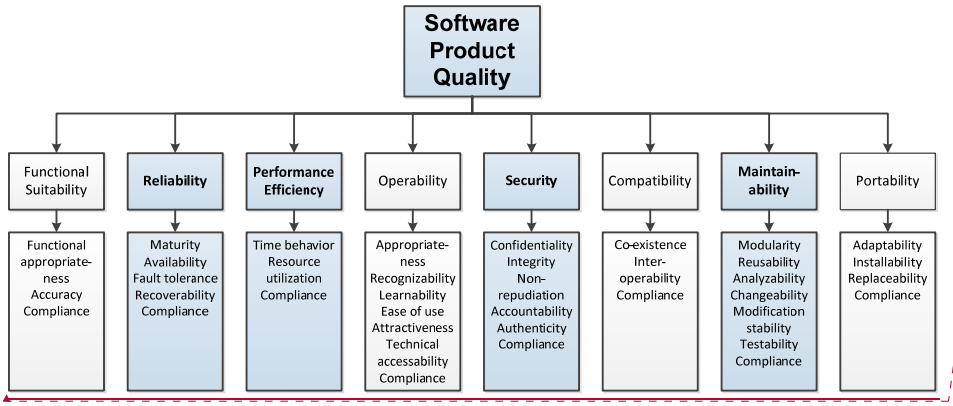


Figure 1. Software Quality Characteristics from ISO/IEC 25010 with CISQ focal areas highlighted.

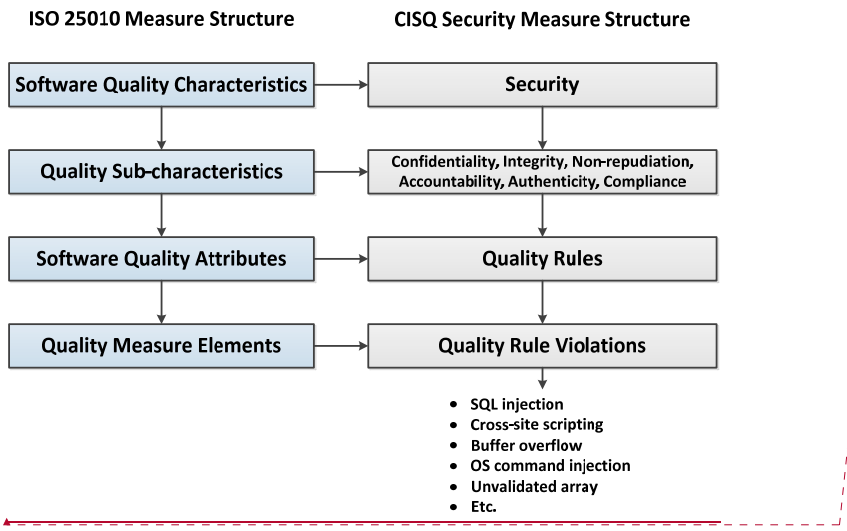
ISO/IEC 25023 establishes a framework of software quality characteristic measures wherein each quality sub-characteristic consists of a collection of quality attributes that can be quantified as quality measure elements. A quality measure element quantifies a unitary measurable attribute of software, such as the violation of a quality rule. Figure 2 presents an example of the ISO/IEC 25023 quality measurement framework using a partial decomposition for the Automated Source Code Security Measure.

The non-normative portion of this specification begins by listing the security issues that can plague software developed with poor architectural and coding practices. Quality rules written as architectural or coding practices are conventions that avoided the problem described in the security issue. These quality rules were then transformed into software quality measure elements by counting violations of these architectural and coding practices and conventions.

The normative portion of this specification represents each quality measure element developed from a security rule using the Structured Patterns Meta-model Standard (SPMS). The code-based elements in these patterns are represented using the Knowledge Discovery Meta-model (KDM). The calculation of the Automated Source Code Security Measure from its quality measure elements is then represented in the Structured Metrics Meta-model (SMM). This calculation is presented as the simple sum of quality measure elements without being adjusted by a weighting scheme.

There are several weighting schemes that can be applied in aggregating violation counts into structural quality measures. The most effective weighting often depends on the measure's use such as assessing operational risk or estimating maintenance costs. The quality measure elements included in this specification were considered to be severe violations of secure architectural and coding practices that would need to be remediated. Therefore, weightings based on severity would add little useful information to the measure since the variance among weights would be small. In order to support benchmarking among applications, this specification includes a measure of the violation density. This measure is created by dividing the total number of violations detected by a count of Automated Function Points (Object Management Group, 2014).

Formatted: Font: Calibri, 11 pt



Formatted: Font: Calibri, 11 pt

Figure 2. ISO/IEC 25010 Framework for Software Quality Characteristics Measurement

ASCSM-13: Insert new section 1.4 for structural information --

Insert new section 1.4 to contain all information about the structure of the measure. Renumber the CWE/SANS Top 25 Weaknesses section to become 1.5

1.45 CWE/SANS Top 25 Weaknesses

The foundation for this specification is the CWE/SANS Institute Top 25 Most Dangerous Software Errors that provides a list of the 25 most widespread and frequently exploited security weaknesses in software. The anti-patterns and associated rules that constitute these weaknesses can be found in the Common Weakness Enumeration accessible at http://cwe.mitre.org/top25/#Listing. This specification is developed from nineteen of the CWE/SANS Top 25 which can be detected and counted in source code. The CWE is a widely used industry source (http://cwe.mitre.org/community/citations.html) that provides a foundation for an ITU and ISO/IEC standard, in addition to 2 ISO/IEC technical reports:

- SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY Cybersecurity information exchange – Vulnerability/state exchange - Common weakness enumeration (CWE)
- ISO/IEC 29147:2014 Information Technology -- Security Techniques -- Vulnerability Disclosure"
- ISO/IEC TR 24772:2013 Information technology -- Programming languages -- Guidance to avoiding vulnerabilities in programming languages through language selection and use
- ISO/IEC Technical Report is ISO/IEC TR 20004:2012 Information Technology -- Security Techniques -- Refining Software Vulnerability Analysis under ISO/IEC 15408 and ISO/IEC 18045

The Automated Source Code Security Measure is a correlated measure rather than an absolute measure. That is, since it does not measure all possible security-related weaknesses it does not provide an absolute measure of security. However, since it includes counts of what industry experts have determined to be the top 25 known weaknesses, it provides a strong indicator of security that will be highly correlated with the absolute security of a software system and with the probability that it can be breached.

Since the CWE is recognized as the primary industry repository of security weaknesses (Lewis, 2010), it is supported by the majority of vendors providing tools and technology in the software security domain (http://cwe.mitre.org/compatible/compatible.html), such as Coverity, HP Fortify. Klockwork, IBM, CAST, Veracode, and others. These vendors already have capabilities for detecting many of the CWE/SANS Top 25 security weaknesses. Consequently, CWE/SANS Top 25 provides the best source for developing a measure that can be common among the majority of vendors in the software security domain. Industry experts who developed the CWE purposely worded the CWEs to be language and application agnostic in order to allow vendors to develop detectors specific to a wide range of languages and application types beyond the scope that could be covered in the CWE. Since some of the CWE/SANS Top 25 may not be relevant in some languages, the reduced opportunity for anti-patterns in those cases will be reflected in the scores.

Since the impact and frequency of specific violations in the CWE/SANS Top 25 could change over time, this approach allows specific violations to be included, excluded, amplified, or diminished over time in order to support the most effective benchmarking, diagnostic, and predictive use. This specification will be adjusted through controlled OMG processes to reflect changes in the threat environment while retaining the ability to compare baselines. Measurement vendors can compute this standard baseline measure, as well as their own extended measures that include other security anti-patterns.

1.6 Using and Improving This Measure

ASCSM-15: Add section on using the measure -- Add a section 1.6 Using and Improving This Measure

The Automated Source Code Security Measure is a correlated measure rather than an absolute measure. That is, since it does not measure all possible security-related weaknesses it does not provide an absolute measure of security. However, since it includes counts of what industry experts considered high severity security weaknesses, it provides a strong indicator of security that will be highly correlated with the absolute security of a software system and with the probability that it can experience unauthorized penetrations, data theft, malicious internal damage, and related problems.

Since the impact and frequency of specific violations in the Automated Source Code Security Measure could change over time, this approach allows specific violations to be included, excluded, amplified, or diminished over time in order to support the most effective benchmarking, diagnostic, and predictive use. This specification will be adjusted through controlled OMG specification revision processes to reflect changes in security engineering while retaining the ability to compare baselines. Vendors of static analysis and measurement technology can compute this standard baseline measure, as well as their own extended measures that include other security weaknesses not included as measure elements in this specification.

2 Conformance and Compliance

ASCSM-18: Eliminate section 2.1 -- Eliminate section 2.1 since it is extraneous material that does not discuss how to conform to this specification.

~~2.1 Conformance~~

~~This specification conforms to the definitions of software quality characteristics provided in ISO/IEC 25010. The measure specified for Security is based on and complies with definitions of anti-patterns listed in the CWE/SANS Top 25. The specification is expressed in the Implementation Pattern Meta-Model for Software Systems (IPMSS) and the Structured Metrics Meta-model (SMM).~~

~~2.2~~ Compliance

ASCSM-20: Add 'objective' to conformance criteria -- Added a bullet on 'objective' as a criteria for conformance with a description of its attributes. Eliminated subsection numbers since there is now only one section. Took 'compliance' out of the title since this is now only about conformance.

Implementations of this specification should be able to demonstrate the following attributes in order to claim compliance—automated, objective, transparent, and verifiable.

- ***Automated***—The analysis of the source code and the actual counting must be fully automated. These initial inputs include the source code of the application and vetted libraries being used to neutralize input data.
- ***Objective***—After the source code has been prepared for analysis using the information provided as inputs, the analysis, calculation, and presentation of results must not require further human intervention. The analysis and calculation must be able to repeatedly produce the same results and outputs on the same body of software.
- ***Transparent***—Implementations that comply with this specification must clearly list all software components entered into the analysis and list in the output each weakness that was detected.
- ***Verifiable***—Compliance with this specification requires that an implementation state the assumptions/heuristics it uses with sufficient detail so that the calculations may be independently verified by third parties. In addition, all inputs used are required to be clearly described and itemized so that they can be audited by a third party.

Formatted: Normal, Justified, Outline numbered + Level: 1 + Numbering Style: Bullet + Aligned at: 0.25" + Tab after: 0.5" + Indent at: 0.5"

3 References

3.1 Normative

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of any of these publications do not apply.

- Structured Patterns Meta-model Standard, admtf/14-02-01
- Knowledge Discovery Meta-model, version 1.3 (KDM), formal/2011-08-04
- Structured Metrics Meta-model, version 1.0 (SMM), formal/2012-01-05
- MOF/XMI Mapping, version 2.4.1 (XMI), formal/2011-08-09
- Automated Function Points (AFP), formal/2014-01-03
- ISO/IEC 25010 Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models

ASCM-22: Eliminate section 3.2 -- Eliminate non-normative references from this section

~~**3.2 Non-normative**~~

- ~~Common Weakness Enumeration. Mitre Corporation. http://cwe.mitre.org/~~
- ~~CWE/SANS Institute Top 25 Most Dangerous Software Errors. http://cwe.mitre.org/top25/#Listing.~~
- ~~International Organization for Standards. ISO/IEC 25010 Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models.~~
- ~~International Organization for Standards. ISO/IEC 25023 (in development) Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality.~~

4 Terms and Definitions

ASCSM-24: Add more terms for this measure to section 4 -- Add definitions for software security, violation, CWE, quality characteristic.

For the purposes of this specification, the following terms and definitions apply.

Automated Function Points—a specification for automating the counting of Function Points that mirrors as closely as possible the counting guidelines of the International Function Point User Group. (OMG, formal 2014-01-03)

Common Weakness Enumeration—a repository maintained by MITRE Corporation of known weaknesses in software that can be exploited to gain unauthorized entry into a software system. (cwe.mitre.org)

Cyclomatic Complexity—A measure of control flow complexity developed by Thomas McCabe based on a graph-theoretic analysis that reduces the control flow of a computer program to a set of edges, vertices, and their attributes that can be quantified. (McCabe, 1976)

Internal Software Quality
the degree to which a set of static attributes of a software product satisfy stated and implied needs for the software product to be used under specified conditions. This will be referred to as software structural quality, or simply structural quality in this specification. (ISO/IEC 25010)

Quality Measure Element—a measure defined in terms of a software quality attribute and the measurement method for quantifying it, including optionally the transformation by a mathematical function. (ISO/IEC 25010)

Formatted: Justified, Indent: Left: 0", Hanging: 0.25"

Formatted: Font: (Default) +Body (Calibri), 11 pt

Software Quality Property
measurable component of software quality. (derived from ISO/IEC 25010)

Security—
degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. (ISO/IEC 25010)

Software Anti-pattern
also referred to as an anti-pattern, is a violation of good architectural or coding practice that can, based on historical evidence, cause problems in software development, maintenance, or operations.

Software Product
a set of computer programs, procedures, and possibly associated documentation and data. (ISO/IEC 25010)

Software Product Quality Model
a model that categorizes product quality properties into eight characteristics (functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability and portability). Each characteristic is composed of a set of related sub-characteristics. (ISO/IEC 25010)

Software Quality
degree to which a software product satisfies stated and implied needs when used under specified conditions. (ISO/IEC 25010)

Software Quality Attribute

an inherent property or characteristic of software that can be distinguished quantitatively or qualitatively by human or automated means. (derived from ISO/IEC 25010)

Software Quality Characteristic
a category of software quality attributes that bears on software quality. (ISO/IEC 25010)

Software Quality Characteristic Measure
a software quality measure derived from measuring the attributes related to a specific software quality characteristic.

Software Quality Issue
architectural or coding practices that are known to cause problems in software development, maintenance, or operations and for which software quality rules can be defined that help avoid problems created by the issue.

Software Quality Measure
a measure that is defined as a measurement function of two or more values of software quality measure elements. (ISO/IEC 25010)

Software Quality Measurement
(verb) a set of operations having the object of determining a value of a software quality measure. (ISO/IEC 25010)

Software Quality Measure Element
a measure defined in terms of a software quality attribute and the measurement method for quantifying it, including optionally the transformation by a mathematical function. (ISO/IEC 25010)

Software Quality Model
a defined set of software characteristics, and of relationships between them, which provides a framework for specifying software quality requirements and evaluating the quality of a software product. (derived from ISO/IEC 25010)

Software Quality Property—measureable component of software quality. (derived from ISO/IEC 25010)

Formatted: Font: Bold

Software Quality Rule
an architectural or coding practice or convention that represents good software engineering practice and avoids problems in software development, maintenance, or operations. Violations of these quality rules produces software anti-patterns.

Software Quality Sub-characteristic
a sub-category of a software quality characteristic to which software quality attributes and their software quality measure elements are conceptually related. (derived from ISO/IEC 25010)

Software Security— degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. (ISO/IEC 25010)

Software Security Measure Element—a measure defined in terms of a quality attribute of software that affects it security and the measurement method for quantifying it, including optionally the transformation by a mathematical function. (adapted from ISO/IEC 25023)

Formatted: Indent: Left: 0", First line: 0"

Structural Element
a component of software code that can be uniquely identified and counted such as a token, decision, variable, etc.

Structural Quality

the degree to which a set of static attributes of a software product satisfy stated and implied needs for the software product to be used under specified conditions—a component of software quality. This concept is referred to as internal software quality in ISO/IEC 25010.

Violation—a pattern or structure in the code that is inconsistent with good architectural and coding practices and can lead to problems in operation or maintenance.

5 Symbols (and Abbreviated Terms)

ASCSM-26: Add abbreviated terms to section 5 -- Add KDM and change IPMSS to SPMS.

CISQ – Consortium for IT Software Quality

CWE – Common Weakness Enumeration

KDM – Knowledge Discovery Meta-model

SIPMSS – ~~Implementation~~ Software Pattern Meta-model ~~for Software Systems~~Standard

SMM – Structured Metrics Meta-model

Formatted: Justified, Space After: 0 pt, Line spacing: 1.5 lines

Formatted: Font: Calibri, 11 pt

6 Additional Information (Non-normative)

6.1 Software Product Inputs

The following inputs are needed by static code analyzers in order to interpret violations of the software quality rules that would be included in individual software quality measure elements.

- The entire source code for the application being analyzed
- A list of vetted libraries that are being used to "neutralize" input data
- What routines/API calls are being used for remote authentication, to any custom initialization and cleanup routines, to synchronize resources, or to neutralize accepted file types or the names of resources
- The encryption algorithms that are being used

Static code analyzers will also need a list of the anti-patterns that constitute each software quality measure element in the Automated Source Code Security Measure.

6.2 Automated Source Code Security Measure Elements

ASCSM-28: Insert revised Table 1-- Insert revised Table 1 with an additional column. The revised columns should be: Security Pattern, Consequence, Objective, and Measure Element.

The violations of good architectural and coding practice incorporated into the Automated Source Code Security Measure are listed and describe in the following Table 1. Some of the CWEs from the Common Weakness Enumeration repository that are included in the Security measure are also defects that can cause security problems. In order to retain consistency across measurement specifications, the original CWE numbers and titles have been retained for these security measure elements. In this sub-clause and in Clause 7 each security measure element from Table 1 will be labeled as ASCSM-#, where # can be replaced by its CWE number.

~~The anti-patterns incorporated into the Automated Source Code Security Measure were taken from the CWE/SANS Top 25 Most Dangerous Software Errors. Several of the top 25 were not measureable from the source code and were not included in this measure. The CWE was selected since it represents a Software Assurance community effort to catalogue all known weaknesses that open a software system to security problems such as unauthorized intrusion. Detecting and measuring the over 800 CWEs is impractical, especially since not all represent severe security risks. By selecting violations from the top 25 CWEs, the foundation for this measure is based on the most frequent and severe violations of good Security architectural and coding practice. The 19 CWE-based violations of secure architectural and coding practices incorporated into this measure are presented in Table 1 and are listed by their CWE number.~~

~~Table 1. Security Issues, Rules, and Quality Measure Elements~~

~~Security Issue~~	~~Security Rule~~	~~Security Quality Measure Element~~

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	**Rule 1:** Use a vetted library or framework that does not allow this weakness to occur or provides constructs that avoid this weakness, such as Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.	**Measure 1:** # of instances where an input does not use a vetted library or approved construct for neutralization
CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	**Rule 2:** Use a vetted library or framework that does not allow SQL injection to occur, or provides constructs that avoid SQL injection, or uses persistence layers such as Hibernate or Enterprise Java Beans.	**Measure 2:** # of instances where data is included in an SQL statement that is not passed through neutralization routines
CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	**Rule 3:** Use a vetted library or framework that does not allow path traversal to occur or provides constructs that make it easier to avoid.	**Measure 3:** # of path manipulation calls without validation mechanism
CWE-434: Unrestricted Upload of File with Dangerous Type	**Rule 4:** Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of all file types that the system can safely accept. Reject any input that does not strictly conform to specifications for those file types, or transform it into something that does. When performing input validation, consider all potentially relevant properties, including length of files, type of input files, the full range of acceptable values, missing or extra file name extensions, syntax, and conformance rules governing allowable uploads . Validation must be performed in each upload instance. For example, limiting filenames to alphanumeric characters can help to restrict the introduction of unintended file extensions.	**Measure 4:** # of upload opportunities not passed to sanitization calls

CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	**Rule 5:** Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, consider using the ESAPI Encoding control or a similar tool, library, or framework. These will help the programmer encode outputs in a manner less prone to error.	**Measure 5:** # of shell statements or operating system calls executed by the system without proper neutralization routines
CWE-798: Use of Hard-coded Credentials	**Rule 6:** For outbound authentication: store passwords, keys, and other credentials outside of the code in a strongly-protected, encrypted configuration file or database that is protected from access by all outsiders, including other local users on the same system. Properly protect the key (CWE-320). If you cannot use encryption to protect the file, then make sure that the permissions are as restrictive as possible. In Windows environments the Encrypted File System (EFS) may provide some protection.	**Measure 6:** # of remote authentication calls that use literal or fixed values as a user name or password
CWE-706: Use of Incorrectly-Resolved Name or Reference (also covers CWE-98 Improper Control of Filename for Include/Require Statement in PHP Program ('PHP File Inclusion'))	**Rule 7:** Use a vetted library or framework that does not allow user input to determine the names of resources to be used for execution, or provide constructs that avoid this problem.	**Measure 7:** # of names of resources with user input that aren't validated
CWE-129: Improper Validation of Array Index	**Rule 8:** Assume all input is malicious. When accessing a user-controlled array index, use a stringent range of values that are within the target array. Make sure that you do not allow negative values to be used. That is, verify the minimum as well as the maximum of the range of acceptable values.	**Measure 8:** # of array accesses with user input that are not range-checked

CWE-754: Improper Check for Unusual or Exceptional Conditions	**Rule 9:** Check the results of all functions involving system resources that return a value and verify that the value is expected. Notes: Checking the return value of the function will typically be sufficient, however beware of race conditions (CWE-362) in a concurrent environment. If using exception handling, catch and throw specific exceptions instead of overly-general exceptions (CWE-396, CWE-397). Catch and handle exceptions as locally as possible so that exceptions do not propagate too far up the call stack (CWE-705). Avoid unchecked or uncaught exceptions where feasible (CWE-248). Notes: Using specific exceptions, and ensuring that exceptions are checked, helps programmers to anticipate and appropriately handle many unusual events that could occur.	**Measure 9:** # of function calls involving system resources that do not check return values **Measure 10:** # of overly broad exceptions thrown (this will require language specific analysis of potential exceptions) **Measure 11:** # of overly broad exceptions caught (this will require language specific analysis of potential exceptions)
CWE-131: Incorrect Calculation of Buffer Size	**Rule 10:** Perform input validation on any numeric input by ensuring that it is within the expected range. Enforce that the input meets both the minimum and maximum requirements for the expected range.	**Measure 12:** # of allocations with tainted input that have no range check
CWE-327: Use of a Broken or Risky Cryptographic Algorithm	**Rule 11:** Select a well-vetted algorithm that is currently considered to be strong by experts in the field, and select well-tested implementations. As with all cryptographic mechanisms, the source code should be available for analysis. For example, US government systems require FIPS 140-2 certification.	**Measure 13:** Determine the version and type of libraries being used, and verify that they are well vetted implementations and are up to date. For example, FIPS 140-2 has a list of validated implementations. # of algorithms that are not vetted as current and strong.
CWE-134: Uncontrolled Format String	**Rule 12:** Ensure that all format string functions are passed a static string which cannot be controlled by the user and that the proper number of arguments are always sent to that function as well. If at all possible, use functions that do not support the %n operator in format strings.	**Measure 14:** # format strings with user input

CWE-456: Missing Initialization	**Rule 13:** Supply an initial value for all non-static variables	**Measure 15:** # of non-static variables that do not supply an initial value
CWE-672: Operation on a Resource after Expiration or Release	**Rule 14:** Once resources have been released, references to the resource should be cleared and they should not be accessed again.	**Measure 16:** # of released resources whose references have not been cleared and that can be used although they are released (free, file close, socket close, etc.)
CWE-834: Excessive Iterations	**Rule 15:** Do not use user-controlled data for loop conditions. **Rule 16:** Limit the number of recursive calls to a reasonable number.	**Measure 17:** # loop conditions that are specified by a user without a range check or neutralization process **Measure 18:** # of recursive functions that do not move toward a base case on each call
CWE-681: Incorrect Conversion between Numeric Types	**Rule 17:** Type casting should only be performed between compatible types	**Measure 19:** # of type castings between incompatible types
CWE-667: Improper Locking	**Rule 18:** Use industry standard APIs to implement locking mechanism.	**Measure 20:** # of shared resources accessed without synchronization in concurrent context
CWE-772: Missing Release of Resource after Effective Lifetime	**Rule 19**: When the software no longer needs a resource, such as a file, network connection, or memory, it should be released back to the system.	**Measure 21:** # of resources allocated and not released within the same module
CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer	**Rule 20:** When moving in-memory data never exceed the bounds of the buffer and ensure that the source and the destination have compatible sizes	**Measure 22:** # of functions that move in-memory data between buffers of incompatible sizes

In the normative section to follow each quality measure element from Table 1 will be labeled as CISQ-#, where # refers to the measure number in the right column. The CWE # will also be referenced in the subsection title for each CISQ# quality measure element IPMSS specification.

Table 1. Security Patterns, Consequences, Objectives, and Measure Elements

Security Pattern	Consequence	Objective	Measure Element
ASCSM-CWE-22: Path Traversal Improper Input Neutralization	Software that is unaware of file path control incurs the risk of exposition of sensitive data, the risk of corruption of critical files, such as programs, libraries, or important data used in protection mechanisms	Avoid failure to sanitize user input in use in path manipulation operations	Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method

			control elements, and ultimately used in the file path creation statement; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.
ASCSM-CWE-78: OS Command Injection Improper Input Neutralization	Software unaware of OS command control incurs the risk of unauthorized command execution, possibly used to disable the software, or possibly leading to unauthorized read and modify data access	Avoid failure to sanitize user input in use as operating system commands	Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the in the platform action to be executed by the execution environment; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.
ASCSM-CWE-79: Cross-site Scripting Improper Input Neutralization	Software featuring weak output generation practices incurs the risk of arbitrary code execution, the risk of sensitive data being compromised, and many other nefarious consequences	Avoid failure to sanitize user input in use in output generation operations	Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the in the user interface WritesUI action; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.

ASCSM-CWE-89: SQL Injection Improper Input Neutralization	Software unaware of SQL command control incurs the risk of unauthorized read, modify, and delete access to sensitive data	Avoid failure to sanitize user input in use in SQL compilation operations	Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the in the SQL compilation statement; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.
ASCSM-CWE-99: Name or Reference Resolution Improper Input Neutralization	Software unaware of resource identification control incurs the risk of unauthorized access to or modification of sensitive data and system resources, including configuration files and files containing sensitive information	Avoid failure to sanitize user input in use as resource names or references	Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the in the platform action to access a resource by its name; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.
ASCSM-CWE-120: Buffer Copy without Checking Size of Input	Software that is unaware of buffer bounds incurs the risk of corruption of relevant memory, and perhaps instructions, possibly leading to a crash, the risk of data integrity loss, and the risk of	Avoid buffer operations among buffers with incompatible sizes	Number of instances in which the content of the first buffer is moved into the content of the second buffer while the size of the first buffer is greater than the size of the second buffer.

	unauthorized access to sensitive data		
ASCSM-CWE-129: Array Index Improper Input Neutralization	Software that is unaware of array index bounds incurs the risk of corruption of relevant memory, and perhaps instructions, possibly leading to a crash, the risk of data integrity loss, and the risk of unauthorized access to sensitive data	Avoid failure to check range of user input in use as array index	Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the read or write action to access the array; none of the callable or method control element of the transformation sequence being a range check callable and method control element with regards to the array index.
ASCSM-CWE-134: Format String Improper Input Neutralization	Software that is unaware of formatting control incurs the risk of execution of arbitrary code and the risk of information disclosure which can severely simplify exploitation of the software	Avoid failure to sanitize user input in use in formatting operations	Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the formating statement; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.
ASCSM-CWE-252-resource: Unchecked Return Parameter Value of named Callable and Method Control Element	Software unaware of execution status control incurs the risk of bad data being used in operations, possibly leading to a crash or	Avoid improper processing of the execution status of resource handling operations	Number of instances where the named callable control element or method control element executes a 'Read', 'Write', or 'Manage Access' action, yet the value of the return parameter

with Read, Write, and Manage Access to Platform Resource	other unintended behaviors		from the action is not used by any check control element
ASCSM-CWE-327: Broken or Risky Cryptographic Algorithm Usage	Software using broken or risky cryptographic algorithm incurs the risk of sensitive data being compromised	Avoid failure to use vetted cryptographic libraries	Number of instances where the application uses the cryptographic deployed component which is not part of the list of vetted cryptographic deployed components.
ASCSM-CWE-396: Declaration of Catch for Generic Exception	Software unaware of accurate execution status control incurs the risk of bad data being used in operations, possibly leading to a crash or other unintended behaviors	Avoid failure to use dedicated exception types	Number of instances where the named callable control element or method control element contains a catch unit which declares to catch an exception parameter whose data type is part of a list of overly broad exception data types
ASCSM-CWE-397: Declaration of Throws for Generic Exception	Software unaware of accurate execution status control incurs the risk of bad data being used in operations, possibly leading to a crash or other unintended behaviors	Avoid failure to use dedicated exception types	Number of instances where the named callable control element or method control element throws an exception parameter whose data type is part of a list of overly broad exception data types
ASCSM-CWE-434: File Upload Improper Input Neutralization	Software unaware of file upload control incurs the risk of arbitrary code execution	Avoid failure to sanitize user input in use in file upload operations	Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the file file upload action; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.
ASCSM-CWE-456: Storable and	Software featuring weak initialization practices	Avoid failure to explicitly	Number of instances where a storable data element or member

Member Data Element Missing Initialization	incurs the risk of logic errors within the program, possibly leading to a security problem	initialize software data elements in use	data element is declared by the 'Create' action, then is evaluated in a 'Read' action without ever being initialized by a 'Write' action prior to the evaluation
ASCSM-CWE-606: Unchecked Input for Loop Condition	Software unaware of iteration control incurs the risk of unexpected consumption of resources, such as CPU cycles or memory, possibly leading to a crash or program exit due to exhaustion of resources	Avoid failure to check range of user input in use in iteration control	Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the loop condition statement; none of the callable or method control element of the transformation sequence being a range check control element.
ASCSM-CWE-667: Shared Resource Improper Locking	Software featuring inconsistent locking discipline incurs the risk of deadlock	Avoid data corruption during concurrent access	Number of instances where the shared storable data element or member data element, declared with the Create action, is accessed outside a critical section of the application via the Read or Write action.
ASCSM-CWE-672: Expired or Released Resource Usage	Software unaware of resource lifecycle incurs the risk of unauthorized access to sensitive data that is associated with a different user or entity, and the risk of erroneous later attempts to access the resource, possibly leading to a crash	Avoid access to a released, revoked, or expired resource	Number of instances where the platform resource is deallocated in the Manage action using its unique resource handler value which is transported throughout the application via the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, then used later within the application to try and access the resource in the Read or Write action.
ASCSM-CWE-681: Numeric Types Incorrect Conversion	Software featuring weak numerical conversion practices incurs the risk of using the wrong	Avoid numerical data corruption during	Number of instances where a storable element or member element is declared with a numerical data type in the

	number and generating incorrect results, possibly introducing new vulnerability when related to resource allocation and security decision	incompatible mutation	'Create' action, and then is updated with a value which is cast via a type cast action into a second numerical data type, which is incompatible with the first data type
ASCSM-CWE-772: Missing Release of Resource after Effective Lifetime	Software unaware of resource lifecycle incurs the risk of preventing all other processes from accessing the same type of resource	Avoid resource hoarding and consequently resource depletion	Number of instances where a platform resource is allocated and assigned a unique resource handler value via a manage resource action, and its unique resource handler value is used throughout the application along a transformation sequence composed of action elements with data relations, some of which are part of named callable and method control elements, but none of which is a resource release statement
ASCSM-CWE-789: Uncontrolled Memory Allocation	Software that is unaware of buffer bounds incurs the risk of corruption of relevant memory, and perhaps instructions, possibly leading to a crash, the risk of data integrity loss, and the risk of unauthorized access to sensitive data	Avoid failure to check range of user input in use as buffer index	Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the buffer Read or Write access action; none of the callable or method control element of the transformation sequence being a range check control element.
ASCSM-CWE-798: Hard-Coded Credentials Usage for Remote Authentication	Software featuring weak authentication practices incurs the risk of exposing resources and functionality to unintended actors, possibly leading to compromised sensitive	Avoid the existence of hard-coded credentials elements	Number of instances where a storable data element or member data element is initialized by a 'Write' action, transported throughout the application along the transport sequence composed of ActionElements with DataRelations relations,

	information and even the execution of arbitrary code		some of which being part of named callable and method control elements, and ultimately used in the remote resource management action ; the transport sequence is composed of assignment operations as updates to the value would not be considered as hard-coded (literal) any more.
ASCSM-CWE-835: Loop with Unreachable Exit Condition ('Infinite Loop')	Software unaware of iteration control incurs the risk of unexpected consumption of resources, such as CPU cycles or memory, possibly leading to a crash or program exit due to exhaustion of resources	Avoid infinite iterations	Number of instances where the named callable control element or method control element features the executioon path whose entry element is found again in the path, while it has no path whatsoever to not return to itself and exit the recursion

7. ~~I~~SPMS~~S~~ Representation of the Quality Measure Elements ~~(CWEs)~~ (Normative)

ASCSM-32: Add introduction to section 7 -- Add an introduction to section 7 to explain the representation meta-models used and how to read and interpret the patterns.

7.1 Introduction

This chapter displays in a human readable format the content of the machine readable XMI format file attached to the current specification. The content of the machine readable XMI format file is the representations of the Quality Measure Elements

- according to the Implementation Patterns Metamodel for Software Systems (SPMS)
- and relating to the Knowledge Discovery Meta-Model (KDM) within their description as frequently as possible, so as to be as generic as possible yet as accurate as possible.

SPMS

More specifically, the machine readable XMI format file attached to the current specification uses the SPMS Definitions Classes:

- PatternDefinition (SPMS:PatternDefinition): the pattern specification. In the context of this document, each Quality Measure Element is basically the count of occurrences of the described patterns.
- Role (SPMS:Role): "A pattern is informally defined as a set of relationships between a set of entities. Roles describe the set of entities within a pattern, between which those relationships will be described. As such the Role is a required association in a PatternDefinition. […]. Semantically, a Role is a 'slot' that is required to be fulfilled for an instance of its parent PatternDefinition to exist."
- PatternSection (SPMS:PatternSection): "A PatternSection is a free-form prose textual description of a portion of a PatternDefinition." In the context of this document, there are 6 different PatternSections in use:
  - "Descriptor" to provide pattern signature, a visible interface of the pattern,
  - "Measure Element" to provide a human readable explanation of the measure,
  - "Description" to provide a human readable explanation of the pattern that is sought after, identifying "Roles" and KDM modeling information,
  - "Objective" to provide a human readable explanation of the intent to get rid of the occurrences of the pattern that is sought after,
  - "Consequence" to provide a human readable explanation of the issue the detection of the pattern is designed to solve,

- "Input" to provide a human readable of the parameters that are needed to fine-tune the behavior of the pattern detection (e.g.: the target application architectural blueprint to comply with)
- "Comment" to provide some additional information (until now, used to inform about situations where the same measure element is useful for another one of the categories)

As well as some of the SPMS Relationships Classes:

- MemberOf (SPMS:MemberOf): "An InterpatternRelationship specialized to indicate inclusion in a Category"
- Category (SPMS:Category): "A Category is a simple grouping element for gathering related PatternDefinitions into clusters." In the context of this document, the SPMS Categories are used to represent the 4 Quality Characteristics:
  - "Reliability",
  - "Security",
  - "Performance Efficiency",
  - And "Maintainability".

KDM

More specifically, the machine readable XMI format file attached to the current specification uses KDM entities in the "Description" section of the pattern definitions. Descriptions try to remain as generic yet accurate as possible so that the pattern can be applicable and applied to as many situations as possible: different technologies, different programming languages, etc. This means:

1. The descriptions include information such as (code:MethodUnit), (action:Reads), (platform:ManagesResource), … to identify the KDM entities the pattern definition involves
2. The descriptions only detail the salient aspects of the pattern as the specifics can be technology- or language-dependant

KDM is helpful for reading this chapter. However, for readers not familiar with KDM, Table 2 presents a primer which translates standard source code element terms into the KDM wording in this specification.

Reading guide

For each numbered sub-clause of this clause

- Sub-clause 7.2 represents the SPMS Category covered by the current specification
- Starting with sub-clause 7.3 represents a new SPMS PatternDefinition member of this SPMS Category

Table 2. Software elements translated into KDM wording

Software element	KDM wording
function, method, procedure, stored procedure, sub-routine etc.	named callable control element (code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored') or method control element (code:MethodUnit)
variable, field, member, etc.	storable data element (code:StorableUnit) or member data element (code:MemberUnit)
class	class element (code:StorableUnit with code:DataType code:ClassUnit)
interface	interface element (code:StorableUnit of code:DataType code:InterfaceUnit)
method	method element (code:MethodUnit)
field, member	member element (code:MemberUnit)
SQL stored procedures	stored callable control elements (code:CallableUnit with code:CallableKind 'stored') in a data manager resource (platform:DataManager)
return code value	value (code:Value) of the return parameter (code:ParameterUnit of code:ParameterKind 'return')
exception	exception parameter (code:ParameterUnit with code:ParameterKind 'exception')
user input data flow	an external value is entered is entered into the application through the 'ReadsUI' user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along the 'TransformationSequence' sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used as
execution path	execution path (action:BlockUnit composed of action:ActionElements with action:CallableRelations to code:ControlElements)
Libraries, etc.	deployed component (platform:DeployedComponent)
RDBMS	data manager resource (platform:DataManager)
loop body	loop body block (action:BlockUnit starting as the action:TrueFlow of the loop action:GuardedFlow and ending with an action:Flow back to the loop action:GuardedFlow)
loop condition	loop condition (action:BlockUnit used in the action:GuardedFlow)
singleton	class element (code:StorableUnit with code:DataType code:ClassUnit) that can be used only once in the 'to' assoction of a Create action (action:Creates)
checked	used by a check control element (code:ControlElement containing action:ActionElement with a kind from micro KDM list of comparison actions)

SPMS PatternDefinition sub-clauses are:

- Pattern category: the “SPMS:Category” category the pattern is related to through a “SPMS:MemberOf” relationship.
- Pattern sections: the list of "SPMS:PatternSection" sections from the pattern:
  - “Descriptor”,
  - “Description”,
  - “Objective”,
  - “Consequence”,
  - and, when applicable,
    - “Input”,
    - “Comment”.
- Pattern roles: the list of “SPMS:Role” roles used in the “Descriptor”, and “Description” sub-clauses above.

In the following sub-clauses,

- Data between square brackets (e.g.: [key Reliabity]) identifies “xmi:id” that are unique and used to reference entities. They are machine-generated to ensure unicity.
- Data between paranthesis (e.g.: (code:MethodUnit)) identifies KDM modeling information.
- Data between angle brackets (e.g.: <ControlElement>) identifies SPMS Roles in Description and Input sub-clauses.

~~Automated Source Code Security Measure — Quality Measure Elements~~
~~Sub-patterns~~
- ~~IsUserInput~~
- ~~IsUserOutput~~
- ~~IsTransformedFrom~~
- ~~IsSanitizationOperation~~
- ~~NotIncludeSpecificOperation~~
- ~~IsCompiledSQLStatement~~
- ~~IsUsedInPathCreationStatement~~
- ~~IsUsedInFileUploadStatement~~
- ~~IsUsedInExecuteRunTimeCommandStatement~~
- ~~IsAccessByNameStatement~~
- ~~IsFormatStringStatement~~
- ~~IsLiteralValue~~
- ~~IsAssignmentSequence~~
- ~~IsUsedInAuthenticationStatement~~
- ~~IsNotChecked~~
- ~~IsRangeCheckOperation~~
- ~~IsArrayAccessStatement~~
- ~~IsBufferAllocationStatement~~
- ~~IsResourceAllocationStatement~~
- ~~IsResourceReleaseStatement~~
- ~~IsResourceAccessStatement~~
- ~~IsMoveBufferStatement~~

- ~~AreIncompatibleSizes~~
- ~~IsUsedInLoopConditionStatement~~
- ~~AreIncompatibleTypes~~
- ~~HasNoExitExecutionPath~~
- ~~IsRecursiveExecutionPath~~
- ~~IsNonAtomicOperation~~
- ~~IsNotInCriticalSection~~
- ~~IsEvaluationStatement~~
- ~~IsSharedVariable~~
- ~~IsObjectCreationExpression~~
- ~~IsCastClassExpression~~
- ~~IsNullOrNotInitializedValue~~

~~CISQ patterns~~

- ~~CISQ-1~~
- ~~CISQ-2~~
- ~~CISQ-3~~
- ~~CISQ-4~~
- ~~CISQ-5~~
- ~~CISQ-6~~
- ~~CISQ-7~~
- ~~CISQ-8~~
- ~~CISQ-9~~
- ~~CISQ-10~~
- ~~CISQ-11~~
- ~~CISQ-12~~
- ~~CISQ-13~~
- ~~CISQ-14~~
- ~~CISQ-15~~
- ~~CISQ-16~~
- ~~CISQ-17~~
- ~~CISQ-18~~
- ~~CISQ-19~~
- ~~CISQ-20~~
- ~~CISQ-21~~
- ~~CISQ-22~~

ASCSM-30: Eliminate all sub-patterns -- Eliminate all sub-patterns as they are no longer needed.

~~Sub patterns~~

~~IsUserInput~~

~~Pattern Descriptor~~
~~IsUserInput(Value: v, InputStatement: is)~~

~~Pattern Definition~~
~~PatternDefinition.name = IsUserInput~~

PatternDefinition.roles = Value, InputStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsUserInput pattern identifies situations where a value is entered into the application through a user input statement; it filters out values that are not entered by an application user.

Roles
InputStatement
Role.name = InputStatement
Description: the statement in the application which assign a value entered by an application user to a variable
Implementation quidelines: UI::ReadUI

Value
Role.name = Value
Description: the value that is initialized with the input from the InputStatement

IsUserOutput

Pattern Descriptor
IsUserOutput(Value: v, OutputStatement: os)

Pattern Definition
PatternDefinition.name = IsUserOutput
PatternDefinition.roles = Value, OutputStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsUserOutput pattern identifies situations where a value is used to be displayed to the application user; it filters out values that are not fed back to the application user.

Roles
OutputStatement
Role.name = OutputStatement
Description: the statement in the application which displays a value to an application user
Implementation quidelines: UI::WriteUI

Value
Role.name = Value
Description: the value that is displayed with the OutputStatement

IsTransformedFrom

Pattern Descriptor
IsTransformedFrom(OriginalValue: ov, TransformedValue: tv, TransformationSequence: ts)

Pattern Definition
PatternDefinition.name = IsTransformedFrom
PatternDefinition.roles = OriginalValue, TransformedValue, TransformationSequence
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsTransformedFrom pattern identifies couples of values in which the original value is transformed into the transformed value, using assignment (i.e., not changing the content but the container), content update (such as concatenation, incrementation, etc.), ... excluding value replacement / overwrite (as it is considered a new initialization).

Roles
OriginalValue
Role.name = OriginalValue
Description: the starting value
TransformedValue
Role.name = TransformedValue
Description: the value that results from the transformation sequence
TransformationSequence
Role.name = TransformationSequence
Description: a given sequence of value transformation operations
Implementation guidelines:

IsSanitizationOperation

Pattern Descriptor
IsSanitizationOperation(SanitisationOperation: so, SanitizationOperationList: sol)

Pattern Definition
PatternDefinition.name = IsSanitizationOperation
PatternDefinition.roles = SanitisationOperation, SanitizationOperationList
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsSanitizationOperation pattern identifies operations that are considered vetted based on their belonging to a list of vetted operations; it filters out operations that are not considered to be efficient enough to protect against a specific threat.

Roles
SanitizationOperation
Role.name = SanitizationOperation
Description: a given value transformation operation
Implementation guidelines:
SanitizationOperationList
Role.name = SanitizationOperationList
Description: the list of operations that are considered to be valid to prevent a CWE-79 Cross-site Scripting injection
Implementation guidelines:

NotIncludeSpecificOperation

Pattern Descriptor
NotIncludeSpecificOperation(OperationSequence: ts, SpecificOperation: so)

Pattern Definition
PatternDefinition.name = NotIncludeSpecificOperation
PatternDefinition.roles = OperationSequence, SpecificOperation

PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = NotIncludeSpecificOperation pattern identifies sequence of operations that do not contain a specific operation

Roles
OperationSequence
Role.name = OperationSequence
Description: a given sequence of operations
Implementation guidelines:
SpecificOperation
Role.name = SpecificOperation
Description: a given operation
Implementation guidelines:

IsCompiledSQLStatement

Pattern Descriptor
IsCompiledSQLStatement(Value: v, SQLCompilationStatement: scs)

Pattern Definition
PatternDefinition.name = IsCompiledSQLStatement
PatternDefinition.roles = Value, SQLCompilationStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsCompiledSQLStatement pattern identifies situations where a value is sent to the database engine to be compiled (not even executed); it filters out values that are not sent for compilation to the database engine.

Roles
SQLCompilationStatement
Role.name = SQLCompilationStatement
Description: the statement in the application which requires to compile the value by the database engine

Value
Role.name = Value
Description: the value that is compiled with the SQLCompilationStatement

UsedInPathCreationStatement

Pattern Descriptor
IsUsedInPathCreationStatement(Value: v, pathCreationStatement: pcs)

Pattern Definition
PatternDefinition.name = IsUsedInPathCreationStatement
PatternDefinition.roles = Value, PathCreationStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description

PatternSection.body = IsUsedInPathCreationStatement pattern identifies situations where a value is used to create a file path; it filters out values that are not used to create such path.

Roles
PathCreationStatement
Role.name = PathCreationStatement
Description: the statement in the application which builds a file path

Value
Role.name = Value
Description: the value that is used within the PathCreationStatement

IsUsedInFileUploadStatement

Pattern Descriptor
IsUsedInFileUploadStatement(Value: v, fileUploadStatement: fus)

Pattern Definition
PatternDefinition.name = IsUsedInFileUploadStatement
PatternDefinition.roles = Value, FileUploadStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsUsedInFileUploadStatement pattern identifies situations where a value is used to upload a file path; it filters out values that are not used to upload files.

Roles
FileUploadStatement
Role.name = FileUploadStatement
Description: the statement in the application which uploads a file

Value
Role.name = Value
Description: the value that is used within the FileUploadStatement

IsUsedInExecuteRunTimeCommandStatement

Pattern Descriptor
IsUsedInExecuteRunTimeCommandStatement(Value: v, executeRunTimeCommandStatement: es)

Pattern Definition
PatternDefinition.name = IsUsedInExecuteRunTimeCommandStatement
PatternDefinition.roles = Value, ExecuteRunTimeCommandStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsUsedInExecuteRunTimeCommandStatement pattern identifies situations where a value is used as a command to execute in the run-time environment; it filters out values that are not used to execute such commands.

Roles
ExecuteRunTimeCommandStatement

Role.name = ExecuteRunTimeCommandStatement
Description: the statement in the application which requests the run-time environment to execute a command

Value
Role.name = Value
Description: the value that is used within the ExecuteRunTimeCommandStatement

IsAccessByNameStatement

Pattern Descriptor
IsAccessByNameStatement(Value: tv, AccessByNameStatement: as)

Pattern Definition
PatternDefinition.name = IsAccessByNameStatement
PatternDefinition.roles = Value, AccessByNameStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsAccessByNameStatement pattern identifies situations where a value is used as a name to access a resource; it filters out values that are not used as such names.

Roles
AccessByNameStatement
Role.name = AccessByNameStatement
Description: the statement in the application which accesses a resource by name

Value
Role.name = Value
Description: the value that is used within the AccessByNameStatement

IsFormatStringStatement

Pattern Descriptor
IsFormatStringStatement(FormatValue: tv, FormatStatement: fs)

Pattern Definition
PatternDefinition.name = IsFormatStringStatement
PatternDefinition.roles = Value, FormatStringStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsFormatStringStatement pattern identifies situations where a value is used as a display format request; it filters out values that are not used as format request.

Roles
FormatStatement
Role.name = FormatStatement
Description: the statement in the application which controls the format the display

Value
Role.name = Value

Description: the value that is used within the FormatStatement

IsLiteralValue

Pattern Descriptor
IsLiteralValue(Value: v, InitialisationStatement: is)

Pattern Definition
PatternDefinition.name = IsLiteralValue
PatternDefinition.roles = Value, InitialisationStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsLiteralValue pattern identifies situations where a value is hard-coded in the application; it filters out values that are computed and entered into the system by the application user.

Roles
InitialisationStatement
Role.name = InitialisationStatement
Description: the statement in the application which initializes the value

Value
Role.name = Value
Description: the value that is initialized by the InitialisationStatement

IsAssignmentSequence

Pattern Descriptor
IsAssignmentSequence(TransformationSequence: ts, AssignmentOperationList: tol)

Pattern Definition
PatternDefinition.name = IsAssignmentSequence
PatternDefinition.roles = TransformationSequence, AssignmentOperationList
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsAssignmentSequence pattern identifies operation sequences that only forward values through assignments as defined by the list of assignment operations; it filters out operation sequences that update the content of the value.

Roles
TransformationSequence
Role.name = TransformationSequence
Description: a given sequence of value transformation operations
Implementation guidelines:
AssignmentOperationList
Role.name = AssignmentOperationList
Description: list of operations considered as assignement
Implementation guidelines:

IsUsedInAuthenticationStatement

~~Pattern Descriptor~~
~~IsUsedInAuthenticationStatement(Value: v, AuthenticationStatement: as)~~

~~Pattern Definition~~
~~PatternDefinition.name = IsUsedInAuthenticationStatement~~
~~PatternDefinition.roles = Value, AuthenticationStatement~~
~~PatternDefintion.sections = Description~~

~~Pattern Sections~~
~~PatternSection.name = Description~~
~~PatternSection.body = IsUsedInAuthenticationStatement pattern identifies situations where a value is used as an authentication data (user, password, ...); it filters out values that are not used as authentication data.~~

~~Roles~~
~~AuthenticationStatement~~
~~Role.name = AuthenticationStatement~~
~~Description: the statement in the application which manages authentication to a remote system~~

~~Value~~
~~Role.name = Value~~
~~Description: the value that is used within the AuthenticationStatement~~

~~IsNotChecked~~

~~Pattern Descriptor~~
~~IsNotChecked(Value: v, CheckValueStatementList: cvsl)~~

~~Pattern Definition~~
~~PatternDefinition.name = IsNotChecked~~
~~PatternDefinition.roles = Value, CheckValueStatementList~~
~~PatternDefintion.sections = Description~~

~~Pattern Sections~~
~~PatternSection.name = Description~~
~~PatternSection.body = IsNotChecked pattern identifies values which are not checked using a valid value check statement~~

~~Roles~~

~~Value~~
~~Role.name = Value~~
~~Description: a Value~~
~~CheckValueStatementList~~
~~Role.name = CheckValueStatementList~~
~~Description: list of valid operations to check values~~

~~IsRangeCheckOperation~~

~~Pattern Descriptor~~
~~IsRangeCheckOperation(Operation: o, RangeCheckOperationList: rcol)~~

~~Pattern Definition~~
~~PatternDefinition.name = IsRangeCheckOperation~~

PatternDefinition.roles = Operation, RangeCheckOperationList
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsRangeCheckOperation pattern identifies operations which actually control the range of a value

Roles
Operation
Role.name = Operation
Description: an operation that processes a value
RangeCheckOperationList
Role.name = RangeCheckOperationList
Description: list of valid operations to check ranges

IsArrayAccessStatement

Pattern Descriptor
IsArrayAccessStatement(ArrayIndexValue: aiv, ArrayAccessStatement: aas, Array: a)

Pattern Definition
PatternDefinition.name = IsArrayAccessStatement
PatternDefinition.roles = Array, ArrayIndexValue, ArrayAccessStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsArrayAccessStatement pattern identifies situations where an index value is used to access an array.

Roles
ArrayAccessStatement
Role.name = ArrayAccessStatement
Description: the statement in the application which controls the array access
ArrayIndexValue
Role.name = ArrayIndexValue
Description: the value of the index that is used within the ArrayAccessStatement to access the Array
Array
Role.name = Array
Description: the array being accessed by the ArrayAccessStatement

IsBufferAllocationStatement

Pattern Descriptor
IsBufferAllocationStatement(RangeValue: tv, BufferAllocationStatement: as, Buffer: b)

Pattern Definition
PatternDefinition.name = IsBufferAllocationStatement
PatternDefinition.roles = Buffer, RangeValue, BufferAllocationStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description

PatternSection.body = IsBufferAllocationStatement pattern identifies situations where a range value is used to access a buffer.

Roles
BufferAllocationStatement
Role.name = BufferAllocationStatement
Description: the statement in the application which controls the buffer access
RangeValue
Role.name = RangeValue
Description: the value of the range that is used within the BufferAllocationStatement to access the Buffer
Buffer
Role.name = Buffer
Description: the buffer being accessed by the BufferAllocationStatement

IsResourceAllocationStatement

Pattern Descriptor
IsResourceAllocationStatement(UniqueResourceHandlerValue: urhv, ResourceAllocationStatement: ras)

Pattern Definition
PatternDefinition.name = IsResourceAllocationStatement
PatternDefinition.roles = UniqueResourceHandlerValue, ResourceAllocationStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsResourceAllocationStatement pattern identifies situations where a resource is allocated and assigned a unique resource handler value that will be used to access the resource afterwards.

Roles
ResourceAllocationStatement
Role.name = ResourceAllocationStatement
Description: the statement in the application which allocates the resource and assign it a unique resource handler value
UniqueResourceHandlerValue
Role.name = UniqueResourceHandlerValue
Description: the unique value that is assigned within the ResourceAllocationStatement and that let the application handle the resource

IsResourceReleaseStatement

Pattern Descriptor
IsResourceReleaseStatement(UniqueResourceHandlerValue: urhv, ResourceReleaseStatement: rrs)

Pattern Definition
PatternDefinition.name = IsResourceReleaseStatement
PatternDefinition.roles = UniqueResourceHandlerValue, ResourceReleaseStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsResourceReleaseStatement pattern identifies situations where a resource is released *using its assigned unique resource handler* value.

Roles
ResourceReleaseStatement
Role.name = ResourceReleaseStatement
Description: the statement in the application which releases the resource using its assigned unique resource handler value
UniqueResourceHandlerValue
Role.name = UniqueResourceHandlerValue
Description: the unique value that is assigned within the ResourceAllocationStatement and that let the application handle the resource

IsResourceAccessStatement

Pattern Descriptor
IsResourceAccessStatement(UniqueResourceHandlerValue: urhv, ResourceAccessStatement: ras)

Pattern Definition
PatternDefinition.name = IsResourceAccessStatement
PatternDefinition.roles = UniqueResourceHandlerValue, ResourceAccessStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsResourceAccessStatement pattern identifies situations where a resource is accessed (read, write) using its assigned unique resource handler value.

Roles
ResourceAccessStatement
Role.name = ResourceAccessStatement
Description: a statement in the application which accesses the resource using its assigned unique resource handler value
UniqueResourceHandlerValue
Role.name = UniqueResourceHandlerValue
Description: the unique value that is assigned within the ResourceAllocationStatement and that let the application handle the resource

IsMoveBufferStatement

Pattern Descriptor
IsMoveBufferStatement(SourceBuffer: b1, TargetBuffer: b2, MoveBufferStatement: mbs)

Pattern Definition
PatternDefinition.name = IsMoveBufferStatement
PatternDefinition.roles = SourceBuffer, TargetBuffer, MoveBufferStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsMoveBufferStatement pattern identifies situations where the content of a source buffer is moved onto the content of a target buffer.

Roles
SourceBuffer
Role.name = SourceBuffer

Description: a buffer, considered as the source of the MoveBufferStatement
TargetBuffer
Role.name = TargetBuffer
Description: a buffer, considered as the target of the MoveBufferStatement
MoveBufferStatement
Role.name = MoveBufferStatement
Description: the statement that moves the content of a source buffer onto a target buffer

AreIncompatibleSizes

Pattern Descriptor
AreIncompatibleSizes(SourceSizeValue: sv1, TargetSizeValue: sv2)

Pattern Definition
PatternDefinition.name = AreIncompatibleSizes
PatternDefinition.roles = SourceSizeValue, TargetSizeValue
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = AreIncompatibleSizes pattern identifies situations where a source size value is incompatible with a target size value when it relates to buffer move.

Roles
SourceSizeValue
Role.name = SourceSizeValue
Description: the size of a source buffer
TargetSizeValue
Role.name = TargetSizeValue
Description: the size of a target buffer

IsUsedInLoopConditionStatement

Pattern Descriptor
IsUsedInLoopConditionStatement(LoopConditionValue: lcv, LoopConditionStatement: lcs)

Pattern Definition
PatternDefinition.name = IsUsedInLoopConditionStatement
PatternDefinition.roles = LoopConditionValue, LoopConditionStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsUsedInLoopConditionStatement pattern identifies situations where a value is used in a loop condition; it filters out values that are not used in conditions that control loop behavior.

Roles
LoopConditionStatement
Role.name = LoopConditionStatement
Description: the statement which controls the loop behavior
LoopConditionValue
Role.name = LoopConditionValue
Description: the value that is used within the loop condition

AreIncompatibleTypes

Pattern Descriptor
AreIncompatibleTypes(SourceType: t1, TargetType: t2)

Pattern Definition
PatternDefinition.name = AreIncompatibleTypes
PatternDefinition.roles = SourceType, TargetType
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = AreIncompatibleTypes pattern identifies situations where a source type is incompatible with a target type when it relates to type casting.

Roles
SourceSizeValue
Role.name = SourceSizeValue
Description: the size of a source buffer
TargetSizeValue
Role.name = TargetSizeValue
Description: the size of a target buffer

HasNoExitExecutionPath

Pattern Descriptor
HasNoExitExecutionPath(ExecutableComponent: **executableComponent**, ExecutionPathList: **executionPathList**)

Pattern Definition
PatternDefinition.name = HasNoExitExecutionPath
PatternDefinition.roles = ExecutableComponent, ExecutionPathList
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = HasNoExitExecutionPath pattern identifies components where no execution path exits the component.

Roles
Component
Role.name = ExecutableComponent
Description: an executable component of the application
ExecutionPathList
Role.name = ExecutionPathList
Description: the list of all execution paths of the ExecutableComponent

IsRecursiveExecutionPath

Pattern Descriptor
IsRecursiveExecutionPath(ExecutableComponent: **executableComponent**, RecursiveExecutionPath: **recursiveExecutionPath**)

Pattern Definition

PatternDefinition.name = IsRecursiveExecutionPath
PatternDefinition.roles = ExecutableComponent, RecursiveExecutionPath
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsRecursiveExecutionPath pattern identifies a recursive path for a component.

Roles
Component
Role.name = ExecutableComponent
Description: an executable component of the application
RecursiveExecutionPath
Role.name = RecursiveExecutionPath
Description: a recursive execution path for the executable component

IsNonAtomicOperation

Pattern Descriptor
IsNonAtomicOperation(Variable: v, Operation: o)

Pattern Definition
PatternDefinition.name = IsNonAtomicOperation
PatternDefinition.roles = Variable, Operation
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsNonAtomicOperation pattern identifies an operation on a variable that is not atomic, that is, an operation that reads o writes the variable, while appearing to the rest of the system to occur instantaneously.

Roles
Variable
Role.name = Variable
Description: a variable of the application
Operation
Role.name = Operation
Description: an operation on a variable

IsNotInCriticalSection

Pattern Descriptor
IsNotInCriticalSection(Component: c, Operation: o)

Pattern Definition
PatternDefinition.name = IsNotInCriticalSection
PatternDefinition.roles = Component, Operation
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsNotInCriticalSection pattern identifies an operation which is not in a critical section of a component.

Roles
Component
Role.name = Component
Description: a component of the application
Operation
Role.name = Operation
Description: an operation

IsEvaluationStatement

Pattern Descriptor
IsEvaluationStatement(Value: v, EvaluationStatement: es)

Pattern Definition
PatternDefinition.name = IsEvaluationStatement
PatternDefinition.roles = Value, EvaluationStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsEvaluationStatement pattern identifies situations where a value is evaluated.

Roles
EvaluationStatement
Role.name = EvaluationStatement
Description: a statement in the application which evaluates the Value

Value
Role.name = Value
Description: a value which is evaluated by EvaluationStatement

IsSharedVariable

Pattern Descriptor
IsSharedVariable(Variable: v, VariableDeclarationStatement : vd)

Pattern Definition
PatternDefinition.name = IsSharedVariable
PatternDefinition.roles = Variable, VariableDeclarationStatement
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsSharedVariable pattern identifies situations where a variable is declared as a shared one.

Roles
VariableDeclarationStatement
Role.name = VariableDeclarationStatement
Description: a statement in the application which declares the variable
Variable
Role.name = Variable
Description: a variable which is declared by VariableDeclarationStatement

IsObjectCreationExpression

Pattern Descriptor
IsObjectCreationExpression(Value: v, Type: t, ObjectCreationExpression: oce)

Pattern Definition
PatternDefinition.name = IsObjectCreationExpression
PatternDefinition.roles = Value, Type, ObjectCreationExpression
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsObjectCreationExpression pattern identifies situations where an expression creates a value of a given type.

Roles
ObjectCreationExpression
Role.name = ObjectCreationExpression
Description: an expression in the application which declares a value of a given type

Value
Role.name = Value
Description: a value which is created by ObjectCreationExpression

Type
Role.name = Type
Description: the type of the value which is created by ObjectCreationExpression

IsCastClassExpression

Pattern Descriptor
IsCastClassExpression(Value: v, Type: t, CastClassExpression: cce)

Pattern Definition
PatternDefinition.name = IsCastClassExpression
PatternDefinition.roles = Value, Type, CastClassExpression
PatternDefintion.sections = Description

Pattern Sections
PatternSection.name = Description
PatternSection.body = IsCastClassExpression pattern identifies situations where a value is casted into a given type.

Roles
CastClassExpression
Role.name = CastClassExpression
Description: an expression in the application which casts a value in a given type

Value
Role.name = Value
Description: a value which is casted into a given type by CastClassExpression

Type

~~Role.name = Type~~
~~Description: the type into which the value is casted by CastClassExpression~~

~~IsNullOrNotInitializedValue~~

~~Pattern Descriptor~~
~~IsNullOrNotInitializedValue(Value: v, DeclarationStatement: ds)~~

~~Pattern Definition~~
~~PatternDefinition.name = IsNullOrNotInitializedValue~~
~~PatternDefinition.roles = Value, EvaluationStatement~~
~~PatternDefintion.sections = Description~~

~~Pattern Sections~~
~~PatternSection.name = Description~~
~~PatternSection.body = IsNullOrNotInitializedValue pattern identifies situations where a value is declared yet not initialized or initialized by "null".~~

~~Roles~~
~~DeclarationStatement~~
~~Role.name = DeclarationStatement~~
~~Description: a statement in the application which declares the Value~~

~~Value~~
~~Role.name = Value~~
~~Description: a value which is declared by DeclarationStatement~~

7.2 Category definition of Security

[key ASCSM_Security] Security

ASCSM-34: Replace CWE-22 description -- Replace CWE-22 description with KDM- & SPMS-based representation

7.3 Pattern definition of ASCSM-CWE-22: Path Traversal Improper Input Neutralization

Pattern Category
[key ASCSM-CWE-22-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective
[key ASCSM-CWE-22-objective]
Avoid failure to sanitize user input in use in path manipulation operations

Consequence

[key ASCSM-CWE-22-consequence]
Software that is unaware of file path control incurs the risk of exposition of sensitive data, the risk of corruption of critical files, such as programs, libraries, or important data used in protection mechanisms

Measure Element

[key ASCSM-CWE-22-measure-element]
Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the file path creation statement; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.

Description

[key ASCSM-CWE-22-description]
This pattern identifies situations where an external value is entered into the application through the <UserInput> user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used in the <PathCreationStatement> file path creation statement (platform:ManagesResource with platform:FileResource); none of the callable or method control element of the transformation sequence being a vetted sanitization callable and method control element (code:ControlElement) from the <PathTraversalSanitizationControlElementList> list of vetted sanitization control elements.

Descriptor

[key ASCSM-CWE-22-descriptor]
ASCSM-CWE-22(UserInput: userInput,PathCreationStatement: pathCreationStatement, TransformationSequence: transformationSequence, PathTraversalSanitizationControlElementList: pathTraversalSanitizationControlElementList)

Variable input

[key ASCSM-CWE-22-input]
<PathTraversalSanitizationControlElementList> list of control elements vetted to handle path traversal vulnerabilities

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-22-roles-userInput] UserInput
[key ASCSM-CWE-22-roles-pathCreationStatement] PathCreationStatement
[key ASCSM-CWE-22-roles-transformationSequence] TransformationSequence
[key ASCSM-CWE-22-roles-pathTraversalSanitizationControlElementList] PathTraversalSanitizationControlElementList

ASCSM: Replace description of CWE-78 -- Replace description of CWE-78 with KDM- & SPMS-based representation.

7.4 Pattern definition of ASCSM-CWE-78: OS Command Injection Improper Input Neutralization

Pattern Category

[key ASCSM-CWE-78-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-78-objective]
Avoid failure to sanitize user input in use as operating system commands

Consequence

[key ASCSM-CWE-78-consequence]
Software unaware of OS command control incurs the risk of unauthorized command execution, possibly used to disable the software, or possibly leading to unauthorized read and modify data access

Measure Element

[key ASCSM-CWE-78-measure-element]
Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the in the platform action to be executed by the execution environment; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.

Description

[key ASCSM-CWE-78-description]
This pattern identifies situations where an external value is entered into the application through the <UserInput> user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used in the <ExecuteRunTimeCommandStatement> platform action (platform:PlatformActions) to be executed by the execution environment (platform:ExecutionResource); none of the callable or method control element of the transformation sequence being a vetted sanitization callable and method control element from the <OSCommandSanitizationControlElementList> list of vetted sanitization callable and method control elements.

Descriptor

[key ASCSM-CWE-78-descriptor]

ASCSM-CWE-78(UserInput: userInput,ExecuteRunTimeCommandStatement: executeRunTimeCommandStatement, TransformationSequence: transformationSequence, OSCommandSanitizationControlElementList: oSCommandSanitizationControlElementList)

Variable input

[key ASCSM-CWE-78-input]
<OSCommandSanitizationControlElementList> list of control elements vetted to handle Command Injection vulnerabilities

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-78-roles-userInput] UserInput
[key ASCSM-CWE-78-roles-executeRunTimeCommandStatement] ExecuteRunTimeCommandStatement
[key ASCSM-CWE-78-roles-transformationSequence] TransformationSequence
[key ASCSM-CWE-78-roles-oSCommandSanitizationControlElementList] OSCommandSanitizationControlElementList

ASCSM-39: Replace CWE-79 description -- Replace description of CWE-79 with KDM- & SPMS-based representation.

7.5 Pattern definition of ASCSM-CWE-79: Cross-site Scripting Improper Input Neutralization

Pattern Category

[key ASCSM-CWE-79-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-79-objective]
Avoid failure to sanitize user input in use in output generation operations

Consequence

[key ASCSM-CWE-79-consequence]
Software featuring weak output generation practices incurs the risk of arbitrary code execution, the risk of sensitive data being compromised, and many other nefarious consequences

Measure Element

[key ASCSM-CWE-79-measure-element]
Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the in the user interface WritesUI action; none of the callable

or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.

Description

[key ASCSM-CWE-79-description]
This pattern identifies situations where an external value is entered into the application through the <UserInput> user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used in the <UserDisplay> user interface WritesUI action (ui:WritesUI); none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the <CrossSiteScriptingSanitizationControlElementList> list of vetted sanitization control elements.

Descriptor

[key ASCSM-CWE-79-descriptor]
ASCSM-CWE-79(UserInput: userInput,CrossSiteScriptingSanitizationControlElementList: crossSiteScriptingSanitizationControlElementList, UserDisplay: userDisplay, TransformationSequence: transformationSequence)

Variable input

[key ASCSM-CWE-79-input]
<CrossSiteScriptingSanitizationControlElementList> list of control elements vetted to deal with cross-site scripting vulnerability

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-79-roles-userInput] UserInput
[key ASCSM-CWE-79-roles-crossSiteScriptingSanitizationControlElementList] CrossSiteScriptingSanitizationControlElementList
[key ASCSM-CWE-79-roles-userDisplay] UserDisplay
[key ASCSM-CWE-79-roles-transformationSequence] TransformationSequence

ASCSM-41: Replace CWE-89 description -- Replace description of CWE-89 with KDM- & SPMS-based representation.

7.6 Pattern definition of ASCSM-CWE-89: SQL Injection Improper Input Neutralization

Pattern Category

[key ASCSM-CWE-89-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-89-objective]
Avoid failure to sanitize user input in use in SQL compilation operations

Consequence

[key ASCSM-CWE-89-consequence]
Software unaware of SQL command control incurs the risk of unauthorized read, modify, and delete access to sensitive data

Measure Element

[key ASCSM-CWE-89-measure-element]
Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the in the SQL compilation statement; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.

Description

[key ASCSM-CWE-89-description]
This pattern identifies situations where an external value is entered into the application through the <UserInput> user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used in the <SQLCompilationStatement> SQL compilation statement (data:ReadsColumnSet or data:WritesColumnSet or data:ManagesData or action:Calls to a code:CallableUnit stored in the data:DataResource); none of the callable or method control element of the transformation sequence being a vetted sanitization callable and method control elements from the <SQLInjectionSanitizationControlElementList> list of vetted sanitization control elements.

Descriptor

[key ASCSM-CWE-89-descriptor]
ASCSM-CWE-89(UserInput: userInput,SQLCompilationStatement: sQLCompilationStatement, TransformationSequence: transformationSequence, SQLInjectionSanitizationControlElementList: sQLInjectionSanitizationControlElementList)

Variable input

[key ASCSM-CWE-89-input]
<SQLInjectionSanitizationControlElementList> list of control elements vetted to handle SQL injection vulnerabilities.

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-89-roles-userInput] UserInput
[key ASCSM-CWE-89-roles-sQLCompilationStatement] SQLCompilationStatement
[key ASCSM-CWE-89-roles-transformationSequence] TransformationSequence
[key ASCSM-CWE-89-roles-sQLInjectionSanitizationControlElementList] SQLInjectionSanitizationControlElementList

ASCSM-69: Replace CWE-706 description and change number

-- Replace description of CWE-706 with KDM- & SPMS-based representation and change number to CWE-99 since it describes the more general and inclusive case of this violation.

7.7 Pattern definition of ASCSM-CWE-99: Name or Reference Resolution Improper Input Neutralization

Pattern Category

[key ASCSM-CWE-99-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-99-objective]
Avoid failure to sanitize user input in use as resource names or references

Consequence

[key ASCSM-CWE-99-consequence]
Software unaware of resource identification control incurs the risk of unauthorized access to or modification of sensitive data and system resources, including configuration files and files containing sensitive information

Measure Element

[key ASCSM-CWE-799measure-element]
Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the in the platform action to access a resource by its name; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.

Description

[key ASCSM-CWE-99-description]
This pattern identifies situations where an external value is entered into the application through the <UserInput> user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with

code:CallableKind 'regular', 'external' or 'stored'), and ultimately used in the <AccessByNameStatement> platform action (platform:PlatformActions) to access a resource (platform:ResourceType) by its name; none of the callable or method control element of the transformation sequence being a vetted sanitization callable and method control elements from the <NameOrReferenceResolutionSanitizationControlElementList> list of vetted sanitization callable and method control elements.

Descriptor

[key ASCSM-CWE-99-descriptor]
ASCSM-CWE-99(UserInput: userInput,AccessByNameStatement: accessByNameStatement, TransformationSequence: transformationSequence, NameOrReferenceResolutionSanitizationControlElementList: nameOrReferenceResolutionSanitizationControlElementList)

Variable input

[key ASCSM-CWE-99-input]
<NameOrReferenceResolutionSanitizationControlElementList> list of control elements vetted to handle Name or Reference Resolution vulnerabilities

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-99-roles-userInput] UserInput
[key ASCSM-CWE-99-roles-accessByNameStatement] AccessByNameStatement
[key ASCSM-CWE-99-roles-transformationSequence] TransformationSequence
[key ASCSM-CWE-99-roles-nameOrReferenceResolutionSanitizationControlElementList] NameOrReferenceResolutionSanitizationControlElementList

ASCSM-4: Replace CWE-119 description and correct CWE number -- Replace description of CWE-119 with KDM- & SPMS-based representation and correct the number from 119 to 120.

7.8 Pattern definition of ASCSM-CWE-120: Buffer Copy without Checking Size of Input

Pattern Category

[key ASCSM-CWE-120-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-120-objective]
Avoid buffer operations among buffers with incompatible sizes

Consequence

[key ASCSM-CWE-120-consequence]
Software that is unaware of buffer bounds incurs the risk of corruption of relevant memory, and perhaps instructions, possibly leading to a crash, the risk of data integrity loss, and the risk of unauthorized access to sensitive data

Measure Element

[key ASCSM-CWE-120-measure-element]
Number of instances in which the content of the first buffer is moved into the content of the second buffer while the size of the first buffer is greater than the size of the second buffer.

Description

[key ASCSM-CWE-120-description]
This pattern identifies situations where two buffer storable elements (code:StorableUnit) or member elements (code:MemberUnit) are allocated with specific sizes in <SourceBufferAllocationStatement> and <TargetBufferAllocationStatement> Create actions (action:Creates), transformed within the application via the <SourceTransformationSequence> and <TargetTransformationSequence> sequences (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), then ultimately used by the application to move the content of the first buffer (action:Reads) onto the content of the second buffer (action:Writes) through the <MoveBufferStatement> statement, while the size of the first buffer is greater than the size of the second buffer.

Descriptor

[key ASCSM-CWE-120-descriptor]
ASCSM-CWE-120(SourceBufferAllocationStatement: sourceBufferAllocationStatement,TargetBufferAllocationStatement: targetBufferAllocationStatement, SourceTransformationSequence: sourceTransformationSequence, TargetTransformationSequence: targetTransformationSequence, MoveBufferStatement: moveBufferStatement)

Variable input

(none applicable)

Comment

[key ASCSM-CWE-120-comment] Measure element contributes to Security and Reliability

List of Roles

[key ASCSM-CWE-120-roles-sourceBufferAllocationStatement] SourceBufferAllocationStatement
[key ASCSM-CWE-120-roles-targetBufferAllocationStatement] TargetBufferAllocationStatement
[key ASCSM-CWE-120-roles-sourceTransformationSequence] SourceTransformationSequence
[key ASCSM-CWE-120-roles-targetTransformationSequence] TargetTransformationSequence
[key ASCSM-CWE-120-roles-moveBufferStatement] MoveBufferStatement

ASCSM-45: Replace CWE-129 description -- Replace description of CWE-129 with KDM- & SPMS-based representation.

7.9 Pattern definition of ASCSM-CWE-129: Array Index Improper Input Neutralization

Pattern Category

[key ASCSM-CWE-129-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-129-objective]
Avoid failure to check range of user input in use as array index

Consequence

[key ASCSM-CWE-129-consequence]
Software that is unaware of array index bounds incurs the risk of corruption of relevant memory, and perhaps instructions, possibly leading to a crash, the risk of data integrity loss, and the risk of unauthorized access to sensitive data

Measure Element

[key ASCSM-CWE-129-measure-element]
Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the read or write action to access the array; none of the callable or method control element of the transformation sequence being a range check callable and method control element with regards to the array index.

Description

[key ASCSM-CWE-129-description]
This pattern identifies situations where an external value is entered into the application through the <UserInput> user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used in the <ArrayAccessStatement> read or write action (action:Reads or action:Writes) to access the <Array> array (code:StorableUnit or code:MemberUnit with code:DataType code:ArrayType); none of the callable or method control element of the transformation sequence being a range check callable and method control element with regards to the array index (code:IndexUnit).

Descriptor

[key ASCSM-CWE-129-descriptor]
ASCSM-CWE-129(UserInput: userInput,ArrayAccessStatement: arrayAccessStatement, Array: array, TransformationSequence: transformationSequence)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-129-roles-userInput] UserInput
[key ASCSM-CWE-129-roles-arrayAccessStatement] ArrayAccessStatement
[key ASCSM-CWE-129-roles-array] Array
[key ASCSM-CWE-129-roles-transformationSequence] TransformationSequence

ASCSM-47: Replace CWE-134 description -- Replace description of CWE-134 with KDM- & SPMS-based representation.

7.10 Pattern definition of ASCSM-CWE-134: Format String Improper Input Neutralization

Pattern Category

[key ASCSM-CWE-134-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-134-objective]
Avoid failure to sanitize user input in use in formatting operations

Consequence

[key ASCSM-CWE-134-consequence]
Software that is unaware of formatting control incurs the risk of execution of arbitrary code and the risk of information disclosure which can severely simplify exploitation of the software

Measure Element

[key ASCSM-CWE-134-measure-element]
Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the formating statement; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.

Description

[key ASCSM-CWE-134-description]
This pattern identifies situations where an external value is entered into the application through the <UserInput> user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along

the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used in the <FormatStatement> formating statement; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the <StringFormatSanitizationControlElementList> list of vetted sanitization control elements.

Descriptor

[key ASCSM-CWE-134-descriptor]
ASCSM-CWE-134(UserInput: userInput,FormatStatement: formatStatement, TransformationSequence: transformationSequence, StringFormatSanitizationControlElementList: stringFormatSanitizationControlElementList)

Variable input

[key ASCSM-CWE-134-input]
<StringFormatSanitizationControlElementList> list of control elements vetted to handle format string vulnerabilities

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-134-roles-userInput] UserInput
[key ASCSM-CWE-134-roles-formatStatement] FormatStatement
[key ASCSM-CWE-134-roles-transformationSequence] TransformationSequence
[key ASCSM-CWE-134-roles-stringFormatSanitizationControlElementList] StringFormatSanitizationControlElementList

ASCSM-51: Replace CWE-754 description and change CWE # --

Replace description of CWE-754 with KDM- & SPMS-based representation. Change the CWE # to 252 since this is the more standard and common version of failing to check for unusual or exceptional conditions and is better explained for automation..

7.11 Pattern definition of ASCSM-CWE-252-resource: Unchecked Return Parameter Value of named Callable and Method Control Element with Read, Write, and Manage Access to Platform Resource

Pattern Category

[key ASCSM-CWE-252-resource-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-252-resource-objective]

Avoid improper processing of the execution status of resource handling operations

Consequence

[key ASCSM-CWE-252-resource-consequence]
Software unaware of execution status control incurs the risk of bad data being used in operations, possibly leading to a crash or other unintended behaviors

Measure Element

[key ASCSM-CWE-252-resource-measure-element]
Number of instances where the named callable control element or method control element executes a 'Read', 'Write', or 'Manage Access' action, yet the value of the return parameter from the action is not used by any check control element

Description

[key ASCSM-CWE-252-resource-description]
This pattern identifies situations where the <ControlElement> named callable control element (code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored') or method control element (code:MethodUnit) executes the <ResourceAccessStatement> Read, Write, and Manage Access action (platform:ReadsResource, platform:WritesResource, and platform:ManagesResource) yet the value (code:Value) of the return parameter (code:ParameterUnit of code:ParameterKind 'return') from the action is not used by any check control element (code:ControlElement containing action:ActionElement with a kind from micro KDM list of comparison actions).

Descriptor

[key ASCSM-CWE-252-resource-descriptor]
ASCSM-CWE-252-resource(ControlElement: controlElement,ResourceAccessStatement: resourceAccessStatement)

Variable input

(none applicable)

Comment

[key ASCSM-CWE-252-resource-comment] Measure element contributes to Security and Reliability

List of Roles

[key ASCSM-CWE-252-resource-roles-controlElement] ControlElement
[key ASCSM-CWE-252-resource-roles-resourceAccessStatement] ResourceAccessStatement

ASCSM-49: Replace CWE-327 description -- Replace description of CWE-327 with KDM- & SPMS-based representation.

7.12 Pattern definition of ASCSM-CWE-327: Broken or Risky Cryptographic Algorithm Usage

Pattern Category

[key ASCSM-CWE-327-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-327-objective]
Avoid failure to use vetted cryptographic libraries

Consequence

[key ASCSM-CWE-327-consequence]
Software using broken or risky cryptographic algorithm incurs the risk of sensitive data being compromised

Measure Element

[key ASCSM-CWE-327-measure-element]
Number of instances where the application uses the cryptographic deployed component which is not part of the list of vetted cryptographic deployed components.

Description

[key ASCSM-CWE-327-description]
This pattern identifies situations where the <Application> application uses the <CryptographicDeployedComponentInUse> cryptographic deployed component (platform:DeployedComponent) while it is not part of the <VettedCryptographicDeployedComponentList> list of vetted cryptographic deployed components. As an example, FIPS 140-2 features a list of validated implementations.

Descriptor

[key ASCSM-CWE-327-descriptor]
ASCSM-CWE-327(CryptographicDeployedComponentInUse: cryptographicDeployedComponentInUse,VettedCryptographicDeployedComponentList: vettedCryptographicDeployedComponentList, Application: application)

Variable input

[key ASCSM-CWE-327-input]
<VettedCryptographicDeployedComponentList> list of vetted cryptographic deployed components

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-327-roles-cryptographicDeployedComponentInUse]
CryptographicDeployedComponentInUse
[key ASCSM-CWE-327-roles-vettedCryptographicDeployedComponentList]
VettedCryptographicDeployedComponentList
[key ASCSM-CWE-327-roles-application] Application

ASCSM-53: Split CWE-754 and replace CISQ-11 with CWE 396 --

Shift from CWE-754 to CWE-396 which is specific for missing generic exceptions. Replace description with KDM- and SPMS-based representation.

7.13 Pattern definition of ASCSM-CWE-396: Declaration of Catch for Generic Exception

Pattern Category

[key ASCSM-CWE-396-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-396-objective]
Avoid failure to use dedicated exception types

Consequence

[key ASCSM-CWE-396-consequence]
Software unaware of accurate execution status control incurs the risk of bad data being used in operations, possibly leading to a crash or other unintended behaviors

Measure Element

[key ASCSM-CWE-396-measure-element]
Number of instances where the named callable control element or method control element contains a catch unit which declares to catch an exception parameter whose data type is part of a list of overly broad exception data types

Description

[key ASCSM-CWE-396-description]
This pattern identifies situations where the <ControlElement> named callable control element (code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored') or method control element (code:MethodUnit) contains the <CatchElement> catch unit (action:CatchUnit) which declares to catch the <CaughtExceptionParameter> exception parameter (code:ParameterUnit with code:ParameterKind 'exception') whose datatype (code:DataType) is part of the <OverlyBroadExceptionTypeList> list of overly broad exception datatypes.
As an example, with JAVA, <OverlyBroadExceptionTypeList> is {'java.lang.Exception'}.

Descriptor

[key ASCSM-CWE-396-descriptor]
ASCSM-CWE-396(ControlElement: controlElement,CatchElement: catchElement, CaughtExceptionParameter: caughtExceptionParameter, OverlyBroadExceptionTypeList: overlyBroadExceptionTypeList)

Variable input

[key ASCSM-CWE-396-input]
<OverlyBroadExceptionTypeList> list of overly broad exception datatypes

Comment

[key ASCSM-CWE-396-comment] Measure element contributes to Security and Reliability

List of Roles

[key ASCSM-CWE-396-roles-controlElement] ControlElement
[key ASCSM-CWE-396-roles-catchElement] CatchElement
[key ASCSM-CWE-396-roles-caughtExceptionParameter] CaughtExceptionParameter
[key ASCSM-CWE-396-roles-overlyBroadExceptionTypeList] OverlyBroadExceptionTypeList

ASCSM-55: Change CWE-754 CISQ-10 to CWE-397 and replace text -- Split CWE-754 and change the CISQ 10 part to CWE-397 which is much more descriptive of failing to throw generic exceptions. Replace the description with a KDM- and SPMS-based representation.

7.14 Pattern definition of ASCSM-CWE-397: Declaration of Throws for Generic Exception

Pattern Category

[key ASCSM-CWE-397-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-397-objective]
Avoid failure to use dedicated exception types

Consequence

[key ASCSM-CWE-397-consequence]
Software unaware of accurate execution status control incurs the risk of bad data being used in operations, possibly leading to a crash or other unintended behaviors

Measure Element

[key ASCSM-CWE-397-measure-element]
Number of instances where the named callable control element or method control element throws an exception parameter whose data type is part of a list of overly broad exception data types

Description

[key ASCSM-CWE-397-description]
This pattern identifies situations where the <ControlElement> named callable control element (code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored') or method control element (code:MethodUnit) throws with the <ThrowsAction> Throws action (action:Throws) the <ThrownExceptionParameter> exception parameter (code:ParameterUnit with code:ParameterKind 'exception') whose datatype (code:Datatype) is part of the <OverlyBroadExceptionTypeList> list of overly broad exception datatypes.
As an example, with JAVA, <OverlyBroadExceptionTypeList> is {'java.lang.Exception'}.

Descriptor

[key ASCSM-CWE-397-descriptor]

ASCSM-CWE-397(ControlElement: controlElement,ThrowsAction: throwsAction, ThrownExceptionParameter: thrownExceptionParameter, OverlyBroadExceptionTypeList: overlyBroadExceptionTypeList)

Variable input

[key ASCSM-CWE-397-input]
<OverlyBroadExceptionTypeList> list of overly broad exception datatypes

Comment

[key ASCSM-CWE-397-comment] Measure element contributes to Security and Reliability

List of Roles

[key ASCSM-CWE-397-roles-controlElement] ControlElement
[key ASCSM-CWE-397-roles-throwsAction] ThrowsAction
[key ASCSM-CWE-397-roles-thrownExceptionParameter] ThrownExceptionParameter
[key ASCSM-CWE-397-roles-overlyBroadExceptionTypeList] OverlyBroadExceptionTypeList

ASCSM-57: Replace CWE-434 description -- Replace description of CWE-434 with KDM- & SPMS-based representation.

7.15 Pattern definition of ASCSM-CWE-434: File Upload Improper Input Neutralization

Pattern Category

[key ASCSM-CWE-434-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-434-objective]
Avoid failure to sanitize user input in use in file upload operations

Consequence

[key ASCSM-CWE-434-consequence]
Software unaware of file upload control incurs the risk of arbitrary code execution

Measure Element

[key ASCSM-CWE-434-measure-element]
Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the file file upload action; none of the callable or method control element of the transformation sequence being a vetted sanitization control element from the list of vetted sanitization control elements.

Description

[key ASCSM-CWE-434-description]
This pattern identifies situations where an external value is entered into the application through the <UserInput> user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used in the <FileUploadStatement> file upload action (platform:ManagesResources with platform:FileResource); none of the callable or method control element of the transformation sequence being a vetted sanitization callable and method control element from the <FileUploadSanitizationControlElementList> list of vetted sanitization callable and method control elements.

Descriptor

[key ASCSM-CWE-434-descriptor]
ASCSM-CWE-434(UserInput: userInput,TransformationSequence: transformationSequence, FileUploadStatement: fileUploadStatement, FileUploadSanitizationControlElementList: fileUploadSanitizationControlElementList)

Variable input

[key ASCSM-CWE-434-input]
<FileUploadSanitizationControlElementList> list of control elements vetted to handle File Upload vulnerabilities

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-434-roles-userInput] UserInput
[key ASCSM-CWE-434-roles-transformationSequence] TransformationSequence
[key ASCSM-CWE-434-roles-fileUploadStatement] FileUploadStatement
[key ASCSM-CWE-434-roles-fileUploadSanitizationControlElementList] FileUploadSanitizationControlElementList

ASCSM-59: Replace CWE-456 description -- Replace description of CWE-456 with KDM- & SPMS-based representation.

7.16 Pattern definition of ASCSM-CWE-456: Storable and Member Data Element Missing Initialization

Pattern Category

[key ASCSM-CWE-456-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-456-objective]
Avoid failure to explicitly initialize software data elements in use

Consequence

[key ASCSM-CWE-456-consequence]
Software featuring weak initialization practices incurs the risk of logic errors within the program, possibly leading to a security problem

Measure Element

[key ASCSM-CWE-456-measure-element]
Number of instances where a storable data element or member data element is declared by the 'Create' action, then is evaluated in a 'Read' action without ever being initialized by a 'Write' action prior to the evaluation

Description

[key ASCSM-CWE-456-description]
This pattern identifies situations where the <DataElement> storable data element (code:StorableUnit) or member data element (code:MemberUnit) is declared by the <DeclarationStatement> Create action (action:Creates), then evaluated in the <EvaluationStatement> Read action (action:Reads) without ever being initialized by a Write action (action:Writes) prior to the evaluation.

Descriptor

[key ASCSM-CWE-456-descriptor]
ASCSM-CWE-456(DataElement: dataElement,DeclarationStatement: declarationStatement, EvaluationStatement: evaluationStatement)

Variable input

(none applicable)

Comment

[key ASCSM-CWE-456-comment] Measure element contributes to Security and Reliability

List of Roles

[key ASCSM-CWE-456-roles-dataElement] DataElement
[key ASCSM-CWE-456-roles-declarationStatement] DeclarationStatement
[key ASCSM-CWE-456-roles-evaluationStatement] EvaluationStatement

ASCSM-61: Replace CWE-834 description and change CWE number -- Replace description of CWE-834 with KDM- & SPMS-based representation. Change CWE number to 606 since this CWE provides a clearer description of the violation for an unchecked range of input to a loop.

7.17 Pattern definition of ASCSM-CWE-606: Unchecked Input for Loop Condition

Pattern Category

[key ASCSM-CWE-606-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-606-objective]
Avoid failure to check range of user input in use in iteration control

Consequence

[key ASCSM-CWE-606-consequence]
Software unaware of iteration control incurs the risk of unexpected consumption of resources, such as CPU cycles or memory, possibly leading to a crash or program exit due to exhaustion of resources

Measure Element

[key ASCSM-CWE-606-measure-element]
Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the loop condition statement; none of the callable or method control element of the transformation sequence being a range check control element.

Description

[key ASCSM-CWE-606-description]
This pattern identifies situations where an external value is entered into the application through the <UserInput> user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used in the <LoopConditionStatement> loop condition statement (action:GuardedFlow with an action:TrueFlow returning to the same action:GuardedFlow); none of the callable or method control element of the transformation sequence being a range check control element (code:ControlElement containing action:ActionElement with a kind from micro KDM list of comparison actions).

Descriptor

[key ASCSM-CWE-606-descriptor]
ASCSM-CWE-606(UserInput: userInput,LoopConditionStatement: loopConditionStatement, TransformationSequence: transformationSequence)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-606-roles-userInput] UserInput
[key ASCSM-CWE-606-roles-loopConditionStatement] LoopConditionStatement
[key ASCSM-CWE-606-roles-transformationSequence] TransformationSequence

ASCSM-63: Replace CWE-667 description -- Replace description of CWE-667 with KDM- & SPMS-based representation.

7.18 Pattern definition of ASCSM-CWE-667: Shared Resource Improper Locking

Pattern Category

[key ASCSM-CWE-667-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-667-objective]
Avoid data corruption during concurrent access

Consequence

[key ASCSM-CWE-667-consequence]
Software featuring inconsistent locking discipline incurs the risk of deadlock

Measure Element

[key ASCSM-CWE-667-measure-element]
Number of instances where the shared storable data element or member data element, declared with the Create action, is accessed outside a critical section of the application via the Read or Write action.

Description

[key ASCSM-CWE-667-description]
This pattern identifies situations where the <PublicDataElement> shared (code:ExportKind 'public') storable data element (code:StorableUnit) or member data element (code:MemberUnit), declared with the <DataElementDeclarationStatement> Create action (action:Creates), is accessed outside a critical section (action:BlockUnit) of the application via the <DataElementAcessStatement> Read or Write action (action:Reads or action:Writes).
The critical nature of the section is technology and platform dependent. As examples, in C/C++, critical nature comes from the use of 'mtx_lock' and 'mtx_unlock' from the 'threads.h' standard C language API (code:LanguageUnit), or from the use of 'pthread_mutex_lock' and 'pthread_mutex_unlock' from the 'pthreads.h' C/C++ POSIX API, or from the use of 'EnterCriticalSection' and 'LeaveCriticalSection' from the 'windows.h' C/C++ Win32 API. As other examples, in JAVA, critical nature comes from the use of the 'syncrhonized' keyword, and in C#, critical nature comes from the use of the 'lock' keyword.

Descriptor

[key ASCSM-CWE-667-descriptor]
ASCSM-CWE-667(PublicDataElement: publicDataElement,DataElementDeclarationStatement: dataElementDeclarationStatement, DataElementAcessStatement: dataElementAcessStatement)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-667-roles-publicDataElement] PublicDataElement
[key ASCSM-CWE-667-roles-dataElementDeclarationStatement] DataElementDeclarationStatement
[key ASCSM-CWE-667-roles-dataElementAcessStatement] DataElementAcessStatement

ASCSM-65: Replace CWE-672 description -- Replace description of CWE-672 with KDM- & SPMS-based representation.

7.19 Pattern definition of ASCSM-CWE-672: Expired or Released Resource Usage

Pattern Category

[key ASCSM-CWE-672-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-672-objective]
Avoid access to a released, revoked, or expired resource

Consequence

[key ASCSM-CWE-672-consequence]
Software unaware of resource lifecycle incurs the risk of unauthorized access to sensitive data that is associated with a different user or entity, and the risk of erroneous later attempts to access the resource, possibly leading to a crash

Measure Element

[key ASCSM-CWE-672-measure-element]
Number of instances where the platform resource is deallocated in the Manage action using its unique resource handler value which is transported throughout the application via the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, then used later within the application to try and access the resource in the Read or Write action.

Description

[key ASCSM-CWE-672-description]
This pattern identifies situations where the <PlatformResource> platform resource (platform:ResourceType) is deallocated in the <ResourceReleaseStatement> manages action (platform:ManagesResource) using its unique resource handler value which is transported throughout the application via the <TransportSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), then used later within the application to try and access the resource in the <ResourceAccessStatement> read or write action (platform:ReadsResource or platform:WritesResource).

Descriptor

[key ASCSM-CWE-672-descriptor]
ASCSM-CWE-672(PlatformResource: platformResource,ResourceReleaseStatement: resourceReleaseStatement, TransportSequence: transportSequence, ResourceAccessStatement: resourceAccessStatement)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-672-roles-platformResource] PlatformResource
[key ASCSM-CWE-672-roles-resourceReleaseStatement] ResourceReleaseStatement
[key ASCSM-CWE-672-roles-transportSequence] TransportSequence
[key ASCSM-CWE-672-roles-resourceAccessStatement] ResourceAccessStatement

ASCSM-67: Replace CWE-681 description -- Replace description of CWE-681 with KDM- & SPMS-based representation.

7.20 Pattern definition of ASCSM-CWE-681: Numeric Types Incorrect Conversion

Pattern Category

[key ASCSM-CWE-681-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-681-objective]
Avoid numerical data corruption during incompatible mutation

Consequence

[key ASCSM-CWE-681-consequence]
Software featuring weak numerical conversion practices incurs the risk of using the wrong number and generating incorrect results, possibly introducing new vulnerability when related to resource allocation and security decision

Measure Element

[key ASCSM-CWE-681-measure-element]
Number of instances where a storable element or member element is declared with a numerical data type in the 'Create' action, and then is updated with a value which is cast via a type cast action into a second numerical data type, which is incompatible with the first data type

Description

[key ASCSM-CWE-681-description]
This pattern identifies situations where the <DataElement> storable element (code:StorableElement) or member element (code:MemberUnit) is declared with the <NumericalDataType> numerical datatype (code:IntegerType, code:DecimalType, or coce:FloatType) in the <DataElementDeclarationStatement> Create action (action:Creates), then updated with a value which is cast via the <TypeCastExpression> type cast action (action:ActionElement with micro KDM kind 'TypeCast' or 'DynCast') into the <TargetDataType> second numerical datatype, which is incompatible with the first one.

Descriptor

[key ASCSM-CWE-681-descriptor]
ASCSM-CWE-681(DataElement: dataElement,DataElementDeclarationStatement: dataElementDeclarationStatement, NumericalDataType: numericalDataType, TypeCastExpression: typeCastExpression, TargetDataType: targetDataType)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-681-roles-dataElement] DataElement
[key ASCSM-CWE-681-roles-dataElementDeclarationStatement] DataElementDeclarationStatement
[key ASCSM-CWE-681-roles-numericalDataType] NumericalDataType
[key ASCSM-CWE-681-roles-typeCastExpression] TypeCastExpression
[key ASCSM-CWE-681-roles-targetDataType] TargetDataType

ASCSM-71: Replace CWE-772 description -- Replace description of CWE-772 with KDM- & SPMS-based representation.

7.21 Pattern definition of ASCSM-CWE-772: Missing Release of Resource after Effective Lifetime

Pattern Category

[key ASCSM-CWE-772-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-772-objective]
Avoid resource hoarding and consequently resource depletion

Consequence

[key ASCSM-CWE-772-consequence]
Software unaware of resource lifecycle incurs the risk of preventing all other processes from accessing the same type of resource

Measure Element

[key ASCSM-CWE-772-measure-element]
Number of instances where a platform resource is allocated and assigned a unique resource handler value via a manage resource action, and its unique resource handler value is used throughout the application along a transformation sequence composed of action elements with data relations, some of which are part of named callable and method control elements, but none of which is a resource release statement

Description

[key ASCSM-CWE-772-description]
This pattern identifies situations where the <PlatformResource> platform resource (platform:ResourceType) is allocated and assigned a unique resource handler value via the <ResourceAllocationStatement> ManagesResource action (platform:ManagesResources), its unique resource handler value is used throughout the application, along the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), none of which being a resource release statement (platform:ManagesResource).

Descriptor

[key ASCSM-CWE-772-descriptor]
ASCSM-CWE-772(PlatformResource: platformResource,ResourceAllocationStatement: resourceAllocationStatement, TransformationSequence: transformationSequence)

Variable input

(none applicable)

Comment

[key ASCSM-CWE-772-comment] Measure element contributes to Security and Reliability

List of Roles

[key ASCSM-CWE-772-roles-platformResource] PlatformResource
[key ASCSM-CWE-772-roles-resourceAllocationStatement] ResourceAllocationStatement

[key ASCSM-CWE-772-roles-transformationSequence] TransformationSequence

ASCSM-73: Replace CWE-131 description and change number -

- Replace description of CWE-131 with KDM- & SPMS-based representation and change number to CWE-789 since it provides the more common and case for this violation.

7.22 Pattern definition of ASCSM-CWE-789: Uncontrolled Memory Allocation

Pattern Category

[key ASCSM-CWE-789-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-789-objective]
Avoid failure to check range of user input in use as buffer index

Consequence

[key ASCSM-CWE-789-consequence]
Software that is unaware of buffer bounds incurs the risk of corruption of relevant memory, and perhaps instructions, possibly leading to a crash, the risk of data integrity loss, and the risk of unauthorized access to sensitive data

Measure Element

[key ASCSM-CWE-789-measure-element]
Number of instances where an external value is entered into the application through the user interface ReadsUI action, transformed throughout the application along the sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the buffer Read or Write access action; none of the callable or method control element of the transformation sequence being a range check control element.

Description

[key ASCSM-CWE-789-description]
This pattern identifies situations where an external value is entered into the application through the <UserInput> user interface ReadsUI action (ui:ReadsUI), transformed throughout the application along the <TransformationSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used as an index element (code:IndexUnit) to access a storable or member data element (code:StorableUnit or code:MemberUnit) in the <BufferAccessStatement> buffer Read or Write access action (action:Reads, action:Writes, action:Addresses); none of the callable or method control element of the transformation sequence being a range check with regards to the 'Buffer' buffer that whose maximum size was defined in the <BufferAllocationStatement> buffer creation action (action:Creates).

Descriptor

[key ASCSM-CWE-789-descriptor]
ASCSM-CWE-789(UserInput: userInput,BufferAccessStatement: bufferAccessStatement, TransformationSequence: transformationSequence, BufferAllocationStatement: bufferAllocationStatement)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-789-roles-userInput] UserInput
[key ASCSM-CWE-789-roles-bufferAccessStatement] BufferAccessStatement
[key ASCSM-CWE-789-roles-transformationSequence] TransformationSequence
[key ASCSM-CWE-789-roles-bufferAllocationStatement] BufferAllocationStatement

ASCSM-75: Replace CWE-798 description -- Replace description of CWE-798 with KDM- & SPMS-based representation.

7.23 Pattern definition of ASCSM-CWE-798: Hard-Coded Credentials Usage for Remote Authentication

Pattern Category

[key ASCSM-CWE-798-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-798-objective]
Avoid the existence of hard-coded credentials elements

Consequence

[key ASCSM-CWE-798-consequence]
Software featuring weak authentication practices incurs the risk of exposing resources and functionality to unintended actors, possibly leading to compromised sensitive information and even the execution of arbitrary code

Measure Element

[key ASCSM-CWE-798-measure-element]
Number of instances where a storable data element or member data element is initialized by a 'Write' action, transported throughout the application along the transport sequence composed of ActionElements with DataRelations relations, some of which being part of named callable and method control elements, and ultimately used in the remote resource management action ; the transport

sequence is composed of assignment operations as updates to the value would not be considered as hard-coded (literal) any more.

Description

[key ASCSM-CWE-798-description]
This pattern identifies situations where a literal value (code:Value) is hard-coded in the application via the <InitialisationStatement> Write action (action:Writes), transported throughout the application along the <TransportSequence> sequence (action:BlockUnit) composed of ActionElements with DataRelations relations (action:Reads, action:Writes, action:Addresses), some of which being part of named callable and method control elements (code:MethodUnit or code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored'), and ultimately used in the <AuthenticationStatement> remote resource management action (platform:ManagesResource with platform:ResourceType); the transport sequence is composed of assignment operations as updates to the value would not be considered as hard-coded (literal) any more.

Descriptor

[key ASCSM-CWE-798-descriptor]
ASCSM-CWE-798(InitialisationStatement: initialisationStatement,AuthenticationStatement: authenticationStatement, TransportSequence: transportSequence)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-798-roles-initialisationStatement] InitialisationStatement
[key ASCSM-CWE-798-roles-authenticationStatement] AuthenticationStatement
[key ASCSM-CWE-798-roles-transportSequence] TransportSequence

ASCSM-77: Replace CWE-835 description -- Replace description of CWE-834 with KDM- & SPMS-based representation.

7.24 Pattern definition of ASCSM-CWE-835: Loop with Unreachable Exit Condition ('Infinite Loop')

Pattern Category

[key ASCSM-CWE-835-relatedPatts-security] ASCSM_Security

Pattern Sections

Objective

[key ASCSM-CWE-835-objective]
Avoid infinite iterations

Consequence

[key ASCSM-CWE-835-consequence]
Software unaware of iteration control incurs the risk of unexpected consumption of resources, such as CPU cycles or memory, possibly leading to a crash or program exit due to exhaustion of resources

Measure Element

[key ASCSM-CWE-835-measure-element]
Number of instances where the named callable control element or method control element features the executioon path whose entry element is found again in the path, while it has no path whatsoever to not return to itself and exit the recursion

Description

[key ASCSM-CWE-835-description]
This pattern identifies situations where the <ControlElement> named callable control element (code:CallableUnit with code:CallableKind 'regular', 'external' or 'stored') or method control element (code:MethodUnit) features the <RecursiveExecutionPath> execution path (action:BlockUnit composed of action:ActionElements with action:CallableRelations to code:ControlElements) whose entry element (action:EntryFlow) is found again in the path, while it has no path whatsoever to not return to itself and exit the recursion.

Descriptor

[key ASCSM-CWE-835-descriptor]
ASCSM-CWE-835(ControlElement: controlElement,RecursiveExecutionPath: recursiveExecutionPath)

Variable input

(none applicable)

Comment

(none applicable)

List of Roles

[key ASCSM-CWE-835-roles-controlElement] ControlElement
[key ASCSM-CWE-835-roles-recursiveExecutionPath] RecursiveExecutionPath

ASCSM 39 – Replace CWE-79 description

~~CISQ patterns~~

~~CISQ-1 (CWE-79)~~

~~Pattern Descriptor~~
~~CISQ-1(InputStatement: **inputStatement**, OutputStatement: **outputStatement**, TransformationSequence: **transformationSequence**, SanitizationOperationList: **sanitizationOperationList**)~~

~~Pattern Definition~~
~~PatternDefinition.name = CISQ-1: # of instances where output is not using library for neutralization~~
~~PatternDefinition.roles = InputStatement, OutputStatement, TransformationSequence, SanitizationOperationList~~

PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsUserInput, IsTransformedFrom, IsSanitizationOperation, NotIncludeSpecificOperation, IsUserOutput

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-1 pattern identifies situations where a value is entered into the application through a user input statement, transformed throughout the application, used in a display statement; the transformation sequence is composed of operations, none of which being vetted a sanitization method.
PatternSection.name = Formula
PatternSection.body = IsUserInput(Value: originalValue, InputStatement: inputStatement) IsTransformedFrom(OriginalValue: originalValue, TransformedValue: transformedValue, TransformationSequence: transformationSequence) IsSanitizationOperation(SanitisationOperation: sanitizationOperation, SanitizationOperationList: sanitizationOperationList) NOT EXIST IncludesSpecificOperation(OperationSequence: transformationSequence, SpecificOperation: sanitizationOperation) IsUserOutput(Value: transformedValue, OutputStatement: outputStatement)

Roles
InputStatement
Role.name = InputStatement = IsUserInput.roles(InputStatement)
IsUserInput(Value: **originalValue**, InputStatement: **inputStatement**)
IsTransformedFrom(OriginalValue: **originalValue**, TransformedValue: **transformedValue**, TransformationSequence: **transformationSequence**) IsSanitizationOperation(SanitisationOperation: **sanitizationOperation**, SanitizationOperationList: **sanitizationOperationList**)
NotIncludeSpecificOperation(OperationSequence: **transformationSequence**, SpecificOperation: **sanitizationOperation**) IsUserOutput(Value: **transformedValue**, OutputStatement: **outputStatement**)
OutputStatement
Role.name = OutputStatement = IsUserOutput.roles(OutputStatement)
TransformationSequence
Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)
SanitizationOperationList
Role.name = SanitizationOperationList = IsSanitizationOperation.roles(SanitizationOperationList)

ASCSM 41 – Replace CWE-89 description

CISQ-2 (CWE-89)

Pattern Descriptor
CISQ-2(InputStatement: **inputStatement**, SQLCompilationStatement: **sqlCompilationStatement**, TransformationSequence: **transformationSequence**, SanitizationOperationList: **sanitizationOperationList**)

Pattern Definition
PatternDefinition.name = CISQ-2: # of instances where data is included in SQL statements that is not passed through the neutralization routines.
PatternDefinition.roles = InputStatement, SQLCompilationStatement, TransformationSequence, SanitizationOperationList
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsUserInput, IsTransformedFrom, IsSanitizationOperation, NotIncludeSpecificOperation, IsCompiledSQLStatement

Pattern Sections
PatternSection.name = Description

PatternSection.body = CISQ-2 pattern identifies situations where a value is entered into the application through a user input statement, transformed throughout the application, used in a SQL compilation statement; the transformation sequence is composed of operations, none of which being vetted a sanitization method.
PatternSection.name = Formula
PatternSection.body = IsUserInput(Value: originalValue, InputStatement: inputStatement) IsTransformedFrom(OriginalValue: originalValue, TransformedValue: transformedValue, TransformationSequence: transformationSequence) IsSanitizationOperation(SanitisationOperation: sanitizationOperation, SanitizationOperationList: sanitizationOperationList) NOT EXIST IncludesSpecificOperation(OperationSequence: transformationSequence, SpecificOperation: sanitizationOperation) IsCompiledSQLStatement(Value: transformedValue, SQLCompilationStatement: sqlCompilationStatement)

Roles
InputStatement
IsUserInput(Value: **originalValue**, InputStatement: **inputStatement**)
IsTransformedFrom(OriginalValue: **originalValue**, TransformedValue: **transformedValue**, TransformationSequence: **transformationSequence**) IsSanitizationOperation(SanitisationOperation: **sanitizationOperation**, SanitizationOperationList: **sanitizationOperationList**)
NotIncludeSpecificOperation(OperationSequence: **transformationSequence**, SpecificOperation: **sanitizationOperation**) IsCompiledSQLStatement(Value: **transformedValue**, SQLCompilationStatement: **sqlCompilationStatement**)
SQLCompilationStatement
Role.name = SQLCompilationStatement = IsCompiledSQLStatement.roles(SQLCompilationStatement)
TransformationSequence
Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)
SanitizationOperationList
Role.name = SanitizationOperationList = IsSanitizationOperation.roles(SanitizationOperationList)

ASCSM 34 – Replace CWE-22 description

CISQ-3 (CWE-22)

Pattern Descriptor
CISQ-3(InputStatement: **inputStatement**, PathCreationStatement: **pathCreationStatement**, TransformationSequence: **transformationSequence**, SanitizationOperationList: **sanitizationOperationList**)

Pattern Definition
PatternDefinition.name = CISQ-3: # of path manipulation calls without validation mechanism.
PatternDefinition.roles = InputStatement, PathCreationStatement, TransformationSequence, SanitizationOperationList
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsUserInput, IsTransformedFrom, IsSanitizationOperation, NotIncludeSpecificOperation, IsUsedInPathCreationStatement

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-3 pattern identifies situations where a value is entered into the application through a user input statement, transformed throughout the application, used in a file path creation statement; the transformation sequence is composed of operations, none of which being vetted a sanitization method.
PatternSection.name = Formula

PatternSection.body = IsUserInput(Value: originalValue, InputStatement: inputStatement) IsTransformedFrom(OriginalValue: originalValue, TransformedValue: transformedValue, TransformationSequence: transformationSequence) IsSanitizationOperation(SanitisationOperation: sanitizationOperation, SanitizationOperationList: sanitizationOperationList) NOT EXIST IncludesSpecificOperation(OperationSequence: transformationSequence, SpecificOperation: sanitizationOperation) IsUsedInPathCreationStatement(Value: transformedValue, PathCreationStatement: pathCreationStatement)

Roles
InputStatement
IsUserInput(Value: **originalValue**, InputStatement: **inputStatement**)
IsTransformedFrom(OriginalValue: **originalValue**, TransformedValue: **transformedValue**, TransformationSequence: **transformationSequence**)
IsSanitizationOperation(SanitisationOperation: **sanitizationOperation**, SanitizationOperationList: **sanitizationOperationList**) NotIncludeSpecificOperation(OperationSequence: **transformationSequence**, SpecificOperation: **sanitizationOperation**)
IsUsedInPathCreationStatement(Value: **transformedValue**, PathCreationStatement: **pathCreationStatement**)
PathCreationStatement
Role.name = PathCreationStatement = IsUsedInPathCreationStatement.roles(PathCreationStatement)
TransformationSequence
Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)
SanitizationOperationList
Role.name = SanitizationOperationList = IsSanitizationOperation.roles(SanitizationOperationList)

ASCSM 57 – Replace CWE-434 description

CISQ-4 (CWE-434)

Pattern Descriptor
CISQ-4(InputStatement: **inputStatement**, FileUploadStatement: **fileUploadStatement**, TransformationSequence: **transformationSequence**, SanitizationOperationList: **sanitizationOperationList**)

Pattern Definition
PatternDefinition.name = CISQ-4: # of upload opportunities not passed to sanitization calls.
PatternDefinition.roles = InputStatement, FileUploadStatement, TransformationSequence, SanitizationOperationList
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsUserInput, IsTransformedFrom, IsSanitizationOperation, NotIncludeSpecificOperation, IsUsedInFileUploadStatement

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-4 pattern identifies situations where a value is entered into the application through a user input statement, transformed throughout the application, used in a file upload statement; the transformation sequence is composed of operations, none of which being vetted a sanitization method.
PatternSection.name = Formula
PatternSection.body = IsUserInput(Value: originalValue, InputStatement: inputStatement) IsTransformedFrom(OriginalValue: originalValue, TransformedValue: transformedValue, TransformationSequence: transformationSequence) IsSanitizationOperation(SanitisationOperation: sanitizationOperation, SanitizationOperationList: sanitizationOperationList) NOT EXIST IncludesSpecificOperation(OperationSequence: transformationSequence, SpecificOperation:

sanitizationOperation) IsUsedInFileUploadStatement(Value: transformedValue, FileUploadStatement: fileUploadStatement)

Roles
InputStatement
Role.name = InputStatement = IsUserInput.roles(InputStatement)
FileUploadStatement
Role.name = FileUploadStatement = IsUsedInFileUploadStatement.roles(FileUploadStatement)
TransformationSequence
Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)
SanitizationOperationList
Role.name = SanitizationOperationList = IsSanitizationOperation.roles(SanitizationOperationList)

ASCSM 36 – Replace CWE-78 description

CISQ-5 (CWE-78)

Pattern Descriptor
CISQ-5(InputStatement: **inputStatement**, ExecuteRunTimeCommandStatement: **executeRunTimeCommandStatement**, TransformationSequence: **transformationSequence**, SanitizationOperationList: **sanitizationOperationList**)

Pattern Definition
PatternDefinition.name = CISQ-5: # of shell statements or OS calls executed by the system without proper neutralization routines.
PatternDefinition.roles = InputStatement, ExecuteRunTimeCommandStatement, TransformationSequence, SanitizationOperationList
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsUserInput, IsTransformedFrom, IsSanitizationOperation, NotIncludeSpecificOperation, IsUsedInExecuteRunTimeCommandStatement

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-5 pattern identifies situations where a value is entered into the application through a user input statement, transformed throughout the application, used in a statement to be executuded by the run-time environment; the transformation sequence is composed of operations, none of which being vetted a sanitization method.
PatternSection.name = Formula
PatternSection.body = IsUserInput(Value: **originalValue**, InputStatement: **inputStatement**)IsTransformedFrom(OriginalValue: **originalValue**, TransformedValue: **transformedValue**, TransformationSequence: **transformationSequence**)
IsSanitizationOperation(SanitisationOperation **sanitizationOperation**, SanitizationOperationList **sanitizationOperationList**) NotIncludeSpecificOperation(OperationSequence: **transformationSequence**, SpecificOperation: **sanitizationOperation**)
IsUsedInExecuteRunTimeCommandStatement(Value: **transformedValue**, ExecuteRunTimeCommandStatement: **executeRunTimeCommandStatement**)

Roles
InputStatement
Role.name = InputStatement = IsUserInput.roles(InputStatement)
ExecuteRunTimeCommandStatement
Role.name = ExecuteRunTimeCommandStatement = IsUsedInExecuteRunTimeCommand.roles(ExecuteRunTimeCommandStatement)
TransformationSequence

Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)
SanitizationOperationList
Role.name = SanitizationOperationList = IsSanitizationOperation.roles(SanitizationOperationList)

ASCSM 75 – Replace CWE-798 description

CISQ-6 (CWE-798)

Pattern Descriptor
CISQ-6(InitialisationStatement: is, AuthenticationStatement: as, TransformationSequence: ts)

Pattern Definition
PatternDefinition.name = CISQ-6: # of remote authentication calls that use literal or fixed values as a user name or password.
PatternDefinition.roles = InitialisationStatement, AuthenticationStatement, TransformationSequence
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsLiteralValue, IsTransformedFrom, IsAssignmentSequence, IsUsedInAuthenticationStatement

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-6 pattern identifies situations where a value is hard-coded in the application, transformed throughout the application, used in a statement to authentication; the transformation sequence is composed of assignment operations as updates to the value would not be considered as literal any more.
PatternSection.name = Formula
PatternSection.body = IsLiteralValue(Value: originalValue, InitialisationStatement: initialisationStatement) IsTransformedFrom(OriginalValue: originalValue, TransformedValue: transformedValue, TransformationSequence: assignmentSequence) IsAssignmentSequence(AssignmentSequence: assignmentSequence, AssignmentOperationList: assignmentOperationList) IsUsedInAuthenticationStatement(Value: transformedValue, AuthenticationStatement: authenticationStatement)

Roles
InitialisationStatement
Role.name = InitialisationStatement = IsLiteralValue.roles(InitialisationStatement)
AuthenticationStatement
Role.name = AuthenticationStatement = IsUsedInAuthenticationStatement.roles(AuthenticationStatement)
TransformationSequence
Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)

ASCSM 69 – Replace CWE-706 description

CISQ-7 (CWE-706)

Pattern Descriptor
CISQ-7(InputStatement: **inputStatement**, OutputStatement: **outputStatement**, TransformationSequence: **transformationSequence**, SanitizationOperationList: **sanitizationOperationList**)

Pattern Definition
PatternDefinition.name = CISQ-7: # of names with user input that aren't validated

PatternDefinition.roles = InputStatement, AccessByNameStatement, TransformationSequence, SanitizationOperationList
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsUserInput, IsTransformedFrom, IsSanitizationOperation, NotIncludeSpecificOperation, IsAccessByNameStatement

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-7 pattern identifies situations where a value is entered into the application through a user input statement, transformed throughout the application, used as a name to access a resource; the transformation sequence is composed of operations, none of which being vetted a sanitization method.
PatternSection.name = Formula
PatternSection.body = IsLiteralValue(Value: originalValue, InitialisationStatement: initialisationStatement) IsTransformedFrom(OriginalValue: originalValue, TransformedValue: transformedValue, TransformationSequence: assignmentSequence) IsAssignmentSequence(AssignmentSequence: assignmentSequence, AssignmentOperationList: assignmentOperationList) IsUsedInAuthenticationStatement(Value: transformedValue, AuthenticationStatement: authenticationStatement)

Roles
InputStatement
Role.name = InputStatement = IsUserInput.roles(InputStatement)
Role.name = AccessByNameStatement = IsAccessByNameStatement.roles(AccessByNameStatement)
TransformationSequence
Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)
SanitizationOperationList
Role.name = SanitizationOperationList = IsSanitizationOperation.roles(SanitizationOperationList)

ASCSM 45 – Replace CWE-129 description

CISQ-8 (CWE-129)

Pattern Descriptor
CISQ-8(InputStatement: **inputStatement**, OutputStatement: **outputStatement**, TransformationSequence: **transformationSequence**, SanitizationOperationList: **sanitizationOperationList**)

Pattern Definition
PatternDefinition.name = CISQ-8: # of array accesses with user input that is not range checked
PatternDefinition.roles = InputStatement, ArrayAccessStatement, Array, TransformationSequence
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsUserInput, IsTransformedFrom, IsRangeCheckOperation, NotIncludeSpecificOperation, IsArrayAccessStatement

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-8 pattern identifies situations where a value is entered into the application through a user input statement, transformed throughout the application, used as an index to access an array; the transformation sequence is composed of operations, none of which being a range check.
PatternSection.name = Formula
PatternSection.body = IsUserInput(Value: originalValue, InputStatement: inputStatement) IsTransformedFrom(OriginalValue: originalValue, TransformedValue: transformedValue, TransformationSequence: transformationSequence) IsRangeCheckOperation(Operation:

rangeCheckOperation, RangeValue: transformedValue, MaxValue: sizeValue)
IsSizeOfArray(Array: array, Size: sizeValue) NOT EXIST IncludesSpecificOperation(OperationSequence: transformationSequence, SpecificOperation: rangeCheckOperation)
IsArrayAccessStatement(ArrayIndexValue: transformedValue, ArrayAccessStatement: arrayAccessStatement, Array: array)

Roles
InputStatement
Role.name = InputStatement = IsUserInput.roles(InputStatement)
ArrayAccessStatement
Role.name = ArrayAccessStatement = IsArrayAccessStatement.roles(ArrayAccessStatement)
TransformationSequence
Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)

ASCSM 51 – Replace CWE-754 description

CISQ-9 (CWE-754)

Pattern Descriptor
CISQ-9(ReturnCodeValue: **returnCode**, ResourceAccessStatement: **resourceAccessStatement**)

Pattern Definition
PatternDefinition.name = CISQ-9: # of function calls involving system resources that do not check return values.
PatternDefinition.roles = ReturnCodeValue, ResourceAccessStatement
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsNotChecked, IsResourceAccessStatement

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-9 pattern identifies situations where a resource is accessed within the application, yet the return code of the access statement is not checked.
PatternSection.name = Formula
PatternSection.body = IsResourceAccessStatement(ReturnCodeValue: returnCodeValue, ResourceAccessStatement: resourceAccessStatement) NOT EXIST IsChecked(Value: returnCodeValue, CheckValueStatement: checkValueStatement)

Roles
ReturnCodeValue
Role.name = ReturnCodeValue = IsResourceAccessStatement.roles(ReturnCodeValue)
ResourceAccessStatement
Role.name = ResourceAccessStatement = IsResourceAccessStatement.roles(ResourceAccessStatement)

ASCSM 55 – Replace CWE-754 description

CISQ-10 (CWE-754)

Pattern Descriptor
CISQ-10(ThrownException: **thrownExecption**, OverlyBroadExceptionTypeList: **overlyBroadExceptionTypeList)** IsResourceAccessStatement(ReturnCodeValue: **returnCodeValue**, ResourceAccessStatement: **resourceAccessStatement**) IsNotChecked(Value: **returnCodeValue**, CheckValueStatementList: **checkValueStatementList**) CISQ-9(ReturnCodeValue: **returnCodeValue**, ResourceAccessStatement: **resourceAccessStatement**)

~~Pattern Definition~~
~~PatternDefinition.name = CISQ-10: # of overly broad exceptions thrown.~~
~~PatternDefinition.roles = ThrownException, OverlyBroadExceptionTypeList~~
~~PatternDefintion.sections = Description~~

~~Pattern Sections~~
~~PatternSection.name = Description~~
~~PatternSection.body = CISQ-10 pattern identifies situations where an exception is thrown and its type is part of the list of overly broad exception type list.~~

~~Roles~~
~~ThrownException~~
~~Role.name = ThrownException~~
~~Description: an exception thrown~~
~~OverlyBroadExceptionTypeList~~
~~Role.name = OverlyBroadExceptionTypeList~~
~~Description: the list of exception types that are considered overly broad~~

ASCSM 53 – Replace CWE-754 description

~~CISQ-11 (CWE-754)~~

~~Pattern Descriptor~~
~~CISQ-11(CaughtException: **caughtExecption**, OverlyBroadExceptionTypeList: **overlyBroadExceptionTypeList)**~~

~~Pattern Definition~~
~~PatternDefinition.name = CISQ-11: # of overly broad exceptions caught.~~
~~PatternDefinition.roles = CaughtException, OverlyBroadExceptionTypeList~~
~~PatternDefintion.sections = Description~~

~~Pattern Sections~~
~~PatternSection.name = Description~~
~~PatternSection.body = CISQ-11 pattern identifies situations where an exception is caught and its type is part of the list of overly broad exception type list.~~

~~Roles~~
~~CaughtException~~
~~Role.name = CaughtException~~
~~Description: an exception caught~~
~~OverlyBroadExceptionTypeList~~
~~Role.name = OverlyBroadExceptionTypeList~~
~~Description: the list of exception types that are considered overly broad~~

ASCSM 73 – Replace CWE-131 description

~~CISQ-12 (CWE-131)~~

~~Pattern Descriptor~~
~~CISQ-12(InputStatement: **inputStatement**, BufferAllocationStatement: **bufferAllocationStatement**, TransformationSequence: **transformationSequence**)~~

~~Pattern Definition~~
~~PatternDefinition.name = CISQ-12: # of allocations with tainted input AND no range check~~
~~PatternDefinition.roles = InputStatement, BufferAllocationStatement, TransformationSequence~~

PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsUserInput, IsTransformedFrom, IsRangeCheckOperation, NotIncludeSpecificOperation, IsArrayAccessStatement, IsBufferAllocationStatement

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-12 pattern identifies situations where a value is entered into the application through a user input statement, transformed throughout the application, used as an index to access a buffer range; the transformation sequence is composed of operations, none of which being a range check.
PatternSection.name = Formula
PatternSection.body = IsUserInput(Value: originalValue, InputStatement: inputStatement) IsTransformedFrom(OriginalValue: originalValue, TransformedValue: transformedValue, TransformationSequence: transformationSequence) NOT EXIST IncludesSpecificOperation(OperationSequence: transformationSequence, SpecificOperation: rangeCheckOperation) IsRangeCheckOperation(Operation: rangeCheckOperation, RangeValue: transformedValue, SizeValue: sizeValue) IsBufferReferenceStatement(RangeValue: transformedValue, BufferReferenceStatement: bufferReferenceStatement, Buffer buffer) IsBufferAllocationStatement(SizeValue: sizeValue, BufferAllocationStatement: bufferAllocationStatement, Buffer buffer)

Roles
InputStatement
Role.name = InputStatement = IsUserInput.roles(InputStatement)
BufferAllocationStatement
Role.name = BufferAllocationStatement = IsBufferAllocation.roles(BufferAllocationStatement)
TransformationSequence
Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)

ASCSM 49 – Replace CWE-327 description

CISQ-13 (CWE-327)

Pattern Descriptor
CISQ-13(UsedCryptographicLibrary: **usedCryptographicLibrary**, VettedCryptographicLibraryList: **vettedCryptographicLibraryList**)

Pattern Definition
CISQ-13: Determine the version and type of libraries being used, and verify that they are well vetted implementations and are up-to-date.

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-13 pattern identifies situations where a cryptographic library is used while it is not part of the list of vetted cryptographic libraries. For example, FIPS 140-2 has a list of validated implementations.

Roles
UsedCryptographicLibrary
Role.name = UsedCryptographicLibrary
Description:
VettedCryptographicLibraryList
Role.name = VettedCryptographicLibraryList
Description:

ASCSM 47 – Replace CWE-134 description

~~**CISQ-14 (CWE-134)**~~

~~Pattern Descriptor~~
~~CISQ-14(InputStatement: **inputStatement**, FormatStatement: **formatStatement**, TransformationSequence: **transformationSequence**, SanitizationOperationList: **sanitizationOperationList**)~~

~~Pattern Definition~~
~~PatternDefinition.name = CISQ-14: # of instances where output is not using library for neutralization~~
~~PatternDefinition.roles = InputStatement, FormatStatement, TransformationSequence, SanitizationOperationList~~
~~PatternDefintion.sections = Description, Formula~~
~~PatternDefinition.relatedPatts = (Nature = "Requires") = IsUserInput, IsTransformedFrom, IsSanitizationOperation, NotIncludeSpecificOperation, IsFormatStatement~~

~~Pattern Sections~~
~~PatternSection.name = Description~~
~~PatternSection.body = CISQ-14 pattern identifies situations where a value is entered into the application through a user input statement, transformed throughout the application, used in a display statement; the transformation sequence is composed of operations, none of which being vetted a sanitization method.~~
~~PatternSection.name = Formula~~
~~PatternSection.body = IsUserInput(Value: originalValue, InputStatement: inputStatement) IsTransformedFrom(OriginalValue: originalValue, TransformedValue: transformedValue, TransformationSequence: transformationSequence) IsSanitizationOperation(SanitisationOperation: sanitizationOperation, SanitizationOperationList: sanitizationOperationList) NOT EXIST IncludesSpecificOperation(OperationSequence: transformationSequence, SpecificOperation: sanitizationOperation) IsFormatStatement(Value: transformedValue, FormatStatement: formatStatement)~~

~~Roles~~
~~InputStatement~~
~~Role.name = InputStatement = IsUserInput.roles(InputStatement)~~
~~FormatStatement~~
~~Role.name = FormatStatement = IsFormatStatement.roles(FormatStatement)~~
~~TransformationSequence~~
~~Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)~~
~~SanitizationOperationList~~
~~Role.name = SanitizationOperationList = IsSanitizationOperation.roles(SanitizationOperationList)~~

ASCSM 59 – Replace CWE-456 description

~~**CISQ-15 (CWE-456)**~~

~~Pattern Descriptor~~
~~CISQ-15(InputStatement: **inputStatement**, EvaluationStatement: **evaluationStatement**, TransformationSequence: **transformationSequence**)~~

~~Pattern Definition~~
~~PatternDefinition.name = CISQ-15: # of non-static variables that are evaluated but do not supply an initial value.~~
~~PatternDefinition.roles = InputStatement, EvaluationStatement, TransformationSequence~~
~~PatternDefintion.sections = Description, Formula IsUserInput(Value: **originalValue**, InputStatement: **inputStatement**) IsTransformedFrom(OriginalValue: **originalValue**, TransformedValue: **transformedValue**, TransformationSequence: **transformationSequence**)~~

IsSanitizationOperation(SanitisationOperation: **sanitizationOperation**, SanitizationOperationList: **sanitizationOperationList**) NotIncludeSpecificOperation(OperationSequence: **transformationSequence**, SpecificOperation: **sanitizationOperation**) IsFormatStatement(Value: **transformedValue**, FormatStatement: **formatStatement**) CISQ-14(InputStatement: **inputStatement**, FormatStatement: **formatStatement**, TransformationSequence: **transformationSequence**, SanitizationOperationList: **sanitizationOperationList**) PatternDefinition.relatedPatts = (Nature = "Requires") = IsNullOrNotInitializedValue, IsEvaluationStatement, IsTransformedFrom

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-15 pattern identifies situations where a value is declared as null or without initial value, transformed throughout the application, then evaluated.
PatternSection.name = Formula
PatternSection.body = IsNullOrNotInitializedVariable(Variable: variable, VariableDeclarationStatement: declarationStatement) IsEvaluationStatement(Variable: variable, VariableEvaluationStatement: evaluationStatement)

Roles
DeclarationStatement
Role.name = DeclarationStatement = IsNullOrNotInitializedValue.roles(DeclarationStatement)
EvaluationStatement
Role.name = EvaluationStatement = IsEvaluationStatement.roles(EvaluationStatement)
TransformationSequence
Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)

ASCSM 65 – Replace CWE-672 description

CISQ-16 (CWE-672)

Pattern Descriptor
CISQ-16(ResourceReleaseStatement: **resourceReleaseStatement**, ResourceAccessStatement: resourceAccessStatement, TransformationSequence: transformationSequence)

Pattern Definition
PatternDefinition.name = CISQ-16: # of resources used after they are released (free, file close, socket close, etc.).
PatternDefinition.roles = InputStatement, EvaluationStatement, TransformationSequence
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsResourceReleaseStatement, IsResourceAccessStatement, IsTransformedFrom, IsAssignmentSequence

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-16 pattern identifies situations where a resource is deallocated using its unique resource handler value which is used later within the application to try and access the resource.
PatternSection.name = Formula
PatternSection.body = IsResourceReleaseStatement(UniqueResourceHandlerValue: uniqueResourceHandlerValue, ResourceReleaseStatement: resourceReleaseStatement) IsTransformedFrom(OriginalValue: uniqueResourceHandlerValue, TransformedValue: transformedUniqueResourceHandlerValue, TransformationSequence: assignmentSequence) IsAssignmentSequence(AssignmentSequence: assignmentSequence, AssignmentOperationList: assignmentOperationList) IsResourceAccessStatement(UniqueResourceHandlerValue: transformedUniqueResourceHandlerValue, ResourceAccessStatement: resourceAccessStatement)

Roles

ResourceReleaseStatement
Role.name = ResourceReleaseStatement = IsResourceReleaseStatement.roles(ResourceReleaseStatement)
ResourceAccessStatement
Role.name = ResourceAccessStatement = IsResourceAccessStatement.roles(ResourceAccessStatement)
TransformationSequence
Role.name = TransformationSequence= IsTransformedFrom.roles(TransformationSequence)

ASCSM 61 – Replace CWE-834 description

CISQ-17 (CWE-834)

Pattern Descriptor
CISQ-17(InputStatement: **inputStatement**, LoopConditionStatement: **loopConditionStatement**, TransformationSequence: **transformationSequence**)

Pattern Definition
PatternDefinition.name = CISQ-17: # loop conditions that are specified by a user without some kind of range check or neutralization process
PatternDefinition.roles = InputStatement, LoopConditionStatement, TransformationSequence
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsUserInput, IsUsedInLoopConditionStatement, IsTransformedFrom, NotIncludeSanitization, IsRangeCheckOperation

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-17 pattern identifies situations where a value is entered into the application through a user input statement, transformed throughout the application, used in a condition loop statement; the transformation sequence is composed of operations, none of which being a range check operation.
PatternSection.name = Formula
PatternSection.body = IsUserInput(Value: originalValue, InputStatement: inputStatement) IsTransformedFrom(OriginalValue: originalValue, TransformedValue: transformedValue, TransformationSequence: transformationSequence) NOT EXIST IncludesSpecificOperation(TransformationSequence: transformationSequence, SpecificOperation: rangeCheckOperation) IsRangeCheckOperation(Operation: rangeCheckOperation, RangeValue: transformedValue, SizeValue: sizeValue) IsUsedInLoopConditionStatement(LoopConditionValue: transformedValue, LoopConditionStatement: loopConditionStatement)

Roles
InputStatement
Role.name = InputStatement = IsUserInput.roles(InputStatement)
LoopConditionStatement
Role.name = LoopConditionStatement = IsUsedInLoopConditionStatement.roles(LoopConditionStatement)
TransformationSequence
Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)

ASCSM 77 – Replace CWE-834 description

CISQ-18 (CWE-834)

Pattern Descriptor
CISQ-18(ExecutableComponent: **executableComponent**, RecursiveExecutionPath: **recursiveExecutionPath**)

~~Pattern Definition~~
~~PatternDefinition.name = CISQ-18: # of recursive functions that do not move toward a base case on each call.~~
~~PatternDefinition.roles = ExecutableComponent, RecursiveExecutionPath~~
~~PatternDefintion.sections = Description, Formula~~
~~PatternDefinition.relatedPatts = (Nature = "Requires") = HasNoExitExecutionPath, IsRecursiveExecutionPath~~

~~Pattern Sections~~
~~PatternSection.name = Description~~
~~PatternSection.body = CISQ-18 pattern identifies situations where a component has a recursive path to itself while it has no path whatsoever to not return to itself.~~
~~PatternSection.name = Formula~~
~~PatternSection.body = HasSpecificExecutionPath(Component: component, SpecificExecutionPath: recursiveExecutionPath) NOT EXIST HasSpecificExecutionPath(Component: component, SpecificExecutionPath: exitExecutionPath)~~

~~Roles~~
~~ExecutableComponent~~
~~Role.name = ExecutableComponent = IsRecursiveExecutionPath.roles(ExecutableComponent)~~
~~RecursiveExecutionPath~~
~~Role.name = RecursiveExecutionPath = IsRecursiveExecutionPath.roles(RecursiveExecutionPath)~~

ASCSM 67 – Replace CWE-681 description

~~**CISQ-19 (CWE-681)**~~

~~Pattern Descriptor~~
~~CISQ-19(ObjectCreationExpression: **objectCreationExpression**, CastClassExpression: **castClassExpression**, TransformationSequence: **transformationSequence**)~~

~~Pattern Definition~~
~~PatternDefinition.name = CISQ-19: # of type casting between incompatible types.~~
~~PatternDefinition.roles = ObjectCreationExpression, CastClassExpression, TransformationSequence~~
~~PatternDefintion.sections = Description, Formula~~
~~PatternDefinition.relatedPatts = (Nature = "Requires") = IsObjectCreationExpression, IsTransformedFrom, IsCastClassExpression, AreIncompatibleTypes~~

~~Pattern Sections~~
~~PatternSection.name = Description~~
~~PatternSection.body = CISQ-19 pattern identifies situations where an object is created with a given type then cast into a second type, which is incompatible with the first one.~~
~~PatternSection.name = Formula~~
~~PatternSection.body = IsObjectCreationExpression(Value: createdValue, Type: createdType, ObjectCreationExpression: objectCreationExpression) IsTransformedFrom(OriginalValue: createdValue, TransformedValue: transformedValue, TransformationSequence: transformationSequence) IsCastClassExpression(Value: transformedValue, Type: castType, CastClassExpression: castClassExpression) AreIncompatibleTypes(SourceType: createdType, TargetType: castType)~~

~~Roles~~
~~ObjectCreationExpression~~
~~Role.name = ObjectCreationExpression = IsObjectCreationExpression.roles(ObjectCreationExpression)~~
~~CastClassExpression~~
~~Role.name = CastClassExpression = IsCastClassExpression.roles(CastClassExpression)~~

TransformationSequence
Role.name = TransformationSequence = IsTransformedFrom.roles(TransformationSequence)

ASCSM 63 – Replace CWE-667 description

CISQ-20 (CWE-667)

Pattern Descriptor
CISQ-20(SharedVariableDeclaration: **sharedVariableDeclaration**, VariableAccessStatement:variableAccessStatement)

Pattern Definition
PatternDefinition.name = CISQ-20: # of shared resources accessed without synchronization in concurrent context
PatternDefinition.roles = SharedVariableDeclaration, VariableAccessStatement
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsSharedVariable, IsNonAtomicOperation, IsNotInCriticalSection

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-20 pattern identifies situations where a shared variable is accessed outside a critical section of the application.
PatternSection.name = Formula
PatternSection.body = IsSharedVariable(Variable: variable, VariableDeclarationStatement: variableDeclarationStatement) IsNonAtomicOperation(Variable: variable, Operation: operation) NOT EXIST IsInCriticalSection(Component: component, Operation: operation)

Roles
VariableDeclarationStatement
Role.name = VariableDeclarationStatement = IsSharedVariableDeclaration.roles(VariableDeclarationStatement)
NonAtomicOperation
Role.name = NonAtomicOperation = IsNonAtomicOperation.roles(NonAtomicOperation)

ASCSM 71 – Replace CWE-772 description

CISQ-21 (CWE-772)

Pattern Descriptor
CISQ-21(ResourceAllocationStatement: **resourceAllocationStatement**, TransformationSequence: **transformationSequence**)

Pattern Definition
PatternDefinition.name = CISQ-21: # of resources allocated and not released within the same module
PatternDefinition.roles = ResourceAllocationStatement, TransformationSequence
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsResourceAllocationStatement, IsTransformedFrom, IsAssignmentSequence, NotInclude SpecificOperation

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-21 pattern identifies situations where a resource is allocated and assigned a unique resource handler value which is used throughout the application, along an execution path which is composed of operations, none of which being a resource release statement.
PatternSection.name = Formula

PatternSection.body = IsSharedVariable(Variable: variable, VariableDeclarationStatement: variableDeclarationStatement) IsNonAtomicOperation(Variable: variable, Operation: operation) NOT EXIST IsInCriticalSection(Component: component, Operation: operation)

Roles
ResourceAllocationStatement
Role.name = ResourceAllocationStatement = IsResourceAllocationStatement.roles(ResourceAllocationStatement)
TransformationSequence
Role.name = TransformationSequence= IsTransformedFrom.roles(TransformationSequence)

ASCSM 43 – Replace CWE-119 description

CISQ-22 (CWE-119)

Pattern Descriptor
CISQ-22(SourceBufferAllocationStatement **sourceBufferAllocationStatement**, TargetBufferAllocationStatement **targetBufferAllocationStatement**, MoveBufferStatement **moveBufferStatement**, SourceTransformationSequence **sourceTransformationSequence**, TargetTransformationSequence **targetTransformationSequence**)

Pattern Definition
PatternDefinition.name = CISQ-22: # of functions that move in-memory data between buffers of incompatible sizes
PatternDefinition.roles = [Source|Target]BufferAllocationStatement, [Source|Target]TransformationSequence, MoveBufferStatement
IsResourceAllocationStatement(UniqueResourceHandlerValue: **uniqueResourceHandlerValue**, ResourceAllocationStatement: **resourceAllocationStatement**) IsTransformedFrom(OriginalValue: **uniqueResourceHandlerValue**, TransformedValue: **transformedValue**, TransformationSequence: **transformationSequence**) IsAssignmentSequence(TransformationSequence: **transformationSequence**, AssignmentOperationList: **assignmentOperationList**) NotIncludeSpecificOperation(OperationSequence: **transformationSequence**, SpecificOperation: **ressourceReleaseStatement**)
PatternDefintion.sections = Description, Formula
PatternDefinition.relatedPatts = (Nature = "Requires") = IsTransformedFrom, IsBufferAllocationStatement, IsMoveBufferStatement, AreIncompatibleSizes

Pattern Sections
PatternSection.name = Description
PatternSection.body = CISQ-22 pattern identifies situations where two buffers are allocated with specific sizes, then ultimately used by the application to move the content of the first buffer onto the content of the second buffer, while their sizes are incompatible.
PatternSection.name = Formula
PatternSection.body = IsBufferAllocationStatement(BufferValue originalValue1, SizeValue sizeValue1, BufferAllocationStatement bufferAllocationStatement1) IsBufferAllocationStatement(BufferValue originalValue2, SizeValue sizeValue2, BufferAllocationStatement bufferAllocationStatement1) IsTransformedFrom(OriginalValue originalValue1, TransformedValue transformedValue1, TransformationSequence transformationSequence1) IsTransformedFrom(OriginalValue originalValue2, TransformedValue transformedValue2, TransformationSequence transformationSequence2) IsMoveBufferStatement(SourceBuffer transformedValue1, TargetBuffer transformedValue2, MoveBufferStatement moveBufferStatement) AreIncompatibleSizes(SourceSizeValue sizeValue1, TargetSizeValue sizeValue2)

Roles
[Source|Target]BufferAllocationStatement

~~Role.name = [Source|Target]BufferAllocationStatement = IsBufferAllocationStatement.roles(BufferAllocationStatement)~~
~~[Source|Target]TransformationSequence~~
~~Role.name = [Source|Target]TransformationSequence = IsTransformedFrom.roles(TransformationSequence)~~
~~MoveBufferStatement~~
~~Role.name = MoveBufferStatement = IsMoveBufferStatement.roles(MoveBufferStatement)~~

8

Calculation of Security and Functional Density Measures (Normative)

~~Automated Source Code Security Measure Calculation (Non-Normative)~~

ASCM-81: Expand calculation description -- Revise and expand the description of the measure calculation

8.1 Calculation of the Base Measure

A count of total violations of quality rules was selected as the best alternative for measurement. Software quality measures have frequently been scored at the component level and then aggregated to develop an overall score for the application. However, scoring at the component level was rejected because many critical violations of security quality rules cannot be isolated to a single component, but rather involve interactions among several components. Therefore, the Automated Source Code Security Measure is computed as the sum of its 22 quality measure elements computed across the entire application.

The calculation of the Automated Source Code Security Measure begins with determining the value of each of the 22 security measure elements. Each security measure element is measured as the total number of violations of its associated quality rule that are detected through automated analysis. Thus the value of each of the 22 security measure elements is represented as CISQ-SecME$_i$ where the range for i runs from 1 to 22.

$$\text{CISQ-SecME}_i = \sum \text{(all violations of type CISQ-SecME}_i \text{ detected through automated analysis)}$$

The value of the un-weighted and un-normalized Automated Source Code Security Measure (CISQ-Sec) is the sum of the values of the 22 security measure elements.

$$\text{CISQ-Sec} = \sum_{i=1}^{22} \text{CISQ-SecME}_i$$

Higher values of CISQ-Sec indicate a larger number of security-related defects in the application.

ASCM-83: Add functional density calculation -- Add a description of how to calculate functional density of security violations

8.2 Functional Density of Security Violations

In order to better compare security results among different applications, the Automated Source Code Security Measure can be normalized by size to create a density measure. There are several size measures with which the density of security violations can be normalized, such as lines of code and function points.

Formatted: Font: Calibri, 11 pt

Formatted: Font: Calibri, 11 pt

These size measures, if properly standardized, can be used for creating a density measure for use in benchmarking applications. However, the OMG Automated Function Points measure offers an automatable size measure that, as an OMG Supported Specification, is standardized, adapted from the International Function Point User Group's (IFPUG) counting guidelines, and commercially supported. Although other size measures can be legitimately used to evaluate the density of security violations, the following density measure for security violations is derived from OMG supported specifications for Automated Function Points and the Automated Source Code Security Measure. Thus, the functional density of Security violations is a simple division expressed as follows.

~~8.1 Calculation Formula~~

~~Violations of each quality rule attached to a CWE will be aggregated into a quality measure element. A count of total violations of quality rules was selected as the best alternative for measurement. Scoring at the component level was rejected because many critical violations of quality rules cannot be isolated to a single component, but rather involve interactions among several components. The Automated Source Code Security Measure is computed as the sum of its 22 component quality measure elements. The formula for calculating Automated Source Code Security Measure scores is as follows:~~

$$\text{ASCSM} = \sum_{i=1}^{n} (\text{CISQ i})$$

~~Where CWE25SM = the score for the Automated Source Code Security Measure, and n = 22 as the measure is currently specified.~~

9. Alternative Weighted Measures and Uses (Informative)

ASCSM-85: Replace SMM representation with derived measures -- Replace SMM representation since the charts are hard to read and do not add useful information beyond the accompanying SMM code. Add a section indicating additional ways the Security measure can be weighted to derive new measures.

9.1 Additional Derived Measures

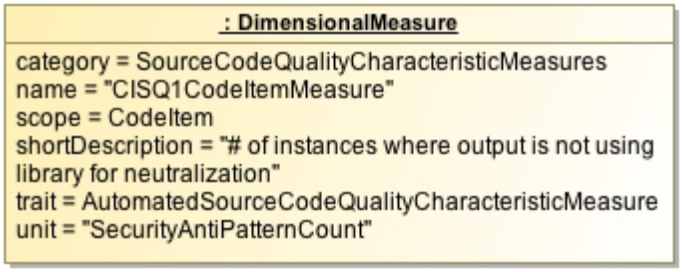
There are many additional weighting schemes that can be applied to the Automated Source Code Security Measure or to the security measure elements that compose it. Table 3 presents several candidate weighted measures and their potential uses. However, these weighting schemes are not derived from any existing standards and are therefore not normative.

Table 3. Informative Weighting Schemes for Security Measurement

Weighting scheme	Potential uses
Weight each Security measure by its severity	Measuring risk of security problems such as data theft and malicious internal damage
Weight each Security measure element by its effort to fix	Measuring cost of ownership, estimating future corrective maintenance effort and costs
Weight each module or application component by its density of Security violations	Prioritizing modules or application components for corrective maintenance or replacement

~~9 Structured Metrics Meta-Model (SMM) Representation (Normative)~~

~~In this section the Automated Source Code Security Measure specification is represented in the Structured Metrics Meta-Model (SMM). The referenced artifacts are modeled using the Knowledge Discovery Meta-Model. Figure 3 presents an example of an SMM class diagram for one of the 22 source code quality characteristic measures aggregated into the Automated Source Code Security Measure. The class diagram for the category of Source Code Quality Characteristic Measures, while Figure 4 presents the class diagram for the Automated Source Code Security Measure.~~



~~Figure 3. Class diagram for Source Code Quality Characteristic Measures~~

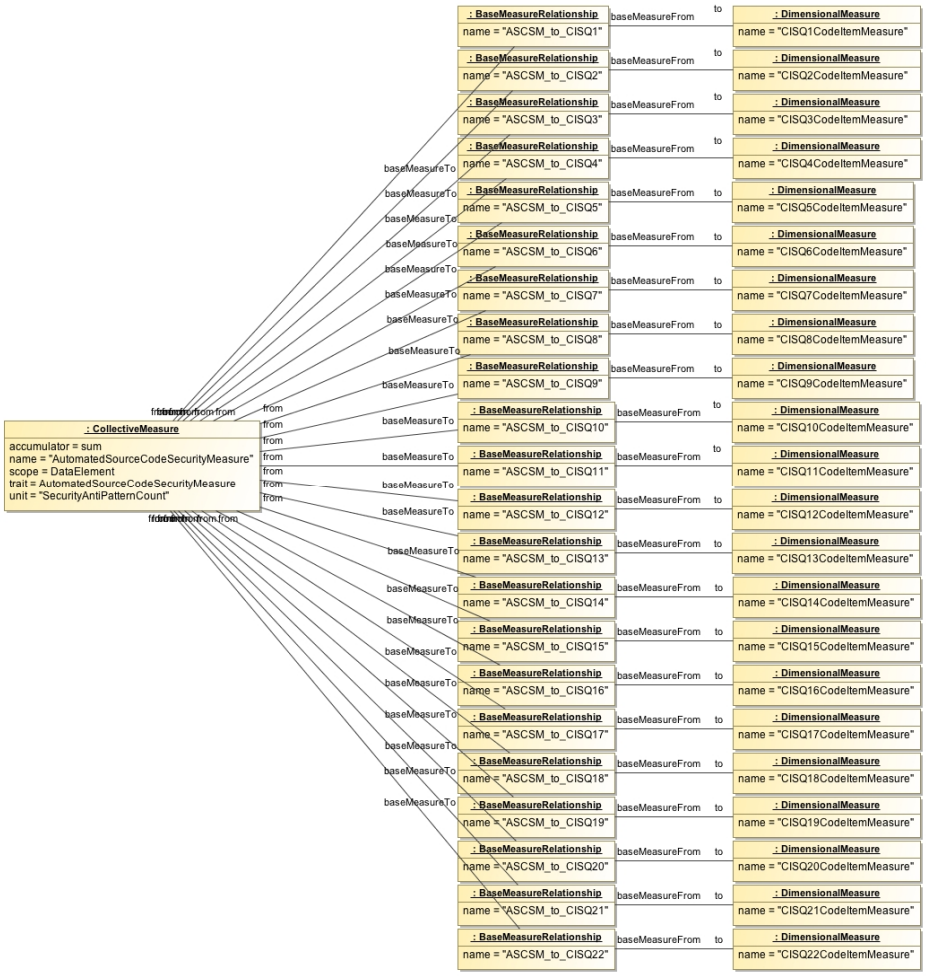


Figure 4. Class diagram for Automated Source Code Security Measure

10. References (Informative)

ASCSM-87: Shorten references - Shorten the reference section to only those related to the text and removing those whose text was deleted in revisions.

Common Weakness Enumeration. http://cwe.mitre.org

Consortium for IT Software Quality (2010). http://www.it-cisq.org

Curtis, B. (1980). Measurement and experimentation in software engineering. *Proceedings of the IEEE*, 68 (9), 1103-1119.

International Organization for Standards. ISO/IEC 25010 Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models

International Organization for Standards (2012). *ISO/IEC 25023 (in development) Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality.*

International Organization for Standards (2012). *ISO/IEC TR 9126-3:2003, Software engineering — Product quality — Part 3: Internal metrics.*

Object Management Group (2014). Automated Function Points. formal 2014-01-03 http://www.omg.org/spec/AFP/, .

~~10. References~~

~~Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., MacCleod, G.J., & Merrit, M.J. (1978). *Characteristics of Software Quality*. Amsterdam: North Holland.~~

~~Consortium for IT Software Quality (2010). http://www.it-cisq.org~~

~~Lewis, J.A., et al. (2010). A Human Capital Crisis in Cybersecurity: A White Paper of the Commission on Cybersecurity for the 44th Presidency. Washington, DC.: Center for Strategic and International Studies.~~

~~Curtis, B. (1980). Measurement and experimentation in software engineering. *Proceedings of the IEEE*, 68 (9), 1103-1119.~~

~~Curtis, B., Sheppard, S.B., and Milliman, P. (1979a). Third time charm: Stronger prediction of programmer performance by software complexity metrics. *Proceedings of the 4th International Conference on Software Engineering*. Washington, DC: IEEE Computer Society, 356-360.~~

~~Curtis, B., Sheppard, S.B., Milliman, P., Borst, A., & Love, T. (1979b). Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics. *IEEE Transactions on Software Engineering*, 5 (2), 96-104.~~

~~Halstead, M.E. (1976). *Elements of Software Science*. Amsterdam: Elsevier North Holland.~~

~~Henry, S. & Kafura, D. (1981). Software structure metrics based on information flow. *IEEE Transactions on Software Engineering*, 7 (5), 510-518.~~

International Organization for Standards. ISO/IEC 25010 Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models

International Organization for Standards (2012). ISO/IEC 25023 (in development) Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality.

International Organization for Standards (2012). ISO/IEC TR 9126-3:2003, Software engineering – Product quality – Part 3: Internal metrics.

International Telecommunications Union (2012). SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY Cybersecurity information exchange – Vulnerability/state exchange - Common weakness enumeration (CWE)

McCabe, T.J. (1976). A complexity measure. *IEEE Transactions on Software Engineering*, 2 (4), 308-320.

McCall, J.A., Richards, P.K., & Walters, G.F. (1977). *Factors in Software Quality – Volumes I, II, & III*. NTIS AD/A-049 014, 015, 055.

Mitre Corporation (2013). *Common Weakness Enumeration*. http://cwe.mitre.org.

Mitre Corporation (2013). *CWE/SANS Top 25*. http://cwe.mitre.org/top25/#Listing.

ASCSM-89: Add CISQ Appendix -- Add an Appendix describing CISQ

Appendix A: CISQ

The purpose of the Consortium for IT Software Quality (CISQ) is to develop specifications for automated measures of software quality characteristics taken on source code. These measures were designed to provide international standards for measuring software structural quality that can be used by IT organizations, IT service providers, and software vendors in contracting, developing, testing, accepting, and deploying IT software applications. Executives from the member companies that joined CISQ prioritized the quality characteristics of Reliability, Security, Performance Efficiency, and Maintainability to be developed as measurement specifications.

CISQ strives to maintain consistency with ISO/IEC standards to the extent possible, and in particular with the ISO/IEC 25000 series that replaces ISO/IEC 9126 and defines quality measures for software systems. In order to maintain consistency with the quality model presented in ISO/IEC 25010, software quality characteristics are defined for the purpose of this specification as attributes that can be measured from the static properties of software, and can be related to the dynamic properties of a computer system as affected by its software. However, the 25000 series, and in particular ISO/IEC 25023 which elaborates quality characteristic measures, does not define these measures at the source code level. Thus, this and other CISQ quality characteristic specifications supplement ISO/IEC 25023 by providing a deeper level of software measurement, one that is rooted in measuring software attributes in the source code.

Companies interested in joining CISQ held executive forums in Frankfurt, Germany; Arlington, VA; and Bangalore, India to set strategy and direction for the consortium. In these forums four quality characteristics were selected as the most important targets for automation—reliability, security, performance efficiency, and maintainability. These attributes cover four of the eight quality characteristics described in ISO/IEC 25010. Figure 1 displays the ISO/IEC 25010 software product quality model with the four software quality characteristics selected for automation by CISQ highlighted in orange. Each software quality characteristic is shown with the sub-characteristics that compose it.

Page 13: [1] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [2] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [3] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [4] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [5] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [6] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [7] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [8] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [9] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [10] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [11] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [12] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [13] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [14] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [15] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [16] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [17] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [18] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [19] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [20] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [21] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [22] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [23] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [24] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [25] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [26] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [27] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [28] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [29] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [30] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [31] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [32] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [33] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [34] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font

Page 13: [35] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [36] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [37] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [38] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [39] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [40] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [41] Formatted **Bill Curtis** **8/19/2015 1:52:00 PM**

Default Paragraph Font, Check spelling and grammar

Page 13: [42] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [43] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [44] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [45] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [46] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [47] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [48] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [49] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [50] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [51] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [52] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [53] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar

Page 13: [54] Formatted	Bill Curtis	8/19/2015 1:52:00 PM

Default Paragraph Font, Check spelling and grammar