INTERNATIONAL STANDARD

ISO/IEC 5055

First edition
2021-03

Information technology — Software measurement — Software quality measurement — Automated source code quality measures

Reference number
ISO/IEC 5055:2021(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by the Object Management Group (OMG) (as Automated Source Code Quality Measures [ASCQM], Version 1.0) and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Computers are being used in an increasing variety of application areas where their correct operation is critical for business accuracy or human safety. Developing or selecting trustworthy, dependable software products is of prime importance. Quantitative evaluation of software product quality is key to ensuring correct performance. This assurance requires rigorous measures of software product quality characteristics. Defining these measures in an international standard is important for standardizing the use of such measures by public and commercial organizations.

ISO/IEC 5055:2020 Automated Source Code Quality Measures defines measures at the software product level (source code) for four of the quality characteristics defined in the software product quality model presented in ISO/IEC 25010-2. These measures conform to the definitions of quality characteristics in ISO/IEC 25010-2 that they quantify. Each measure is calculated from counts of severe weaknesses in the source code that affect the quality characteristic being measured. Each weakness is associated with one or more detection patterns that can guide the development of automated tools for quality analysis of software products. The weaknesses included in each measure were selected by an international team of software engineering experts based on the severity of their impact on a software product quality characteristic.

ISO/IEC 25023 defines software measures for the quality model in ISO/IEC 25010-2, but most all of these measures are at the behavioral level For instance, a measure for availability is defined as mean downtime, which does not identify the problems in the software that caused the downtime. ISO/IEC 5055:2020 therefore supplements ISO/IEC 25023 by defining software measures at the level of weaknesses in the source code. Thus, availability would be measured by the existence of software weaknesses that cause downtime such as poor error handling or missing timeouts. ISO/IEC 5055:2020 adds strong product level measurement to support the ISO/IEC 25000 software product quality standards.

ISO/IEC 5055:2020 may be revised from time to time based on new severe weaknesses being added or existing weaknesses in the model becoming less severe because of advances in computer science. Thus, this will be a standard that adapts to changes in the technology of computing.

Information technology — Software measurement — Software quality measurement — Automated source code quality measures

1 Scope

1.1 Purpose

The measures in this standard were calculated from detecting and counting violations of good architectural and coding practices in the source code that could result in unacceptable operational risks or excessive costs. Establishing standards for these measures at the source code level is important because they have been used in outsourcing and system development contracts without having international standards to reference. For instance, the ISO/IEC 25000 series of standards that govern software product quality provide only a small set of measures at the source code level.

A primary objective of updating these measures was to extend their applicability to embedded software, which is especially important for the growing implementation of embedded devices and the Internet of Things. Functionality that has traditionally been implemented in IT applications is now being moved to embedded chips. Since the weaknesses included in the measures specified in this document have been found to be applicable to all forms of software, embedded software is not treated separately in this specification.

1.2 Overview of Structural Quality Measurement in Software

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

Recent advances in measuring the structural quality of software involve detecting violations of good architectural and coding practice from statically analyzing source code. Violations of good architectural and design practice can also be detected from statically analyzing design specifications written in a design language with a formal syntax and semantics. Good architectural and coding practices can be stated as rules for engineering software products. Violations of these rules will be called weaknesses in this specification to be consistent with terms used in the Common Weakness Enumeration (Martin & Barnum, 2006) which lists many of the weaknesses used in several of these measures.

The Automated Source Code Quality Measures are correlated measures rather than absolute measures. That is, since they do not measure all possible weaknesses in each of the four areas, they do not provide absolute measures. However, since they include counts of what industry experts have determined to be most severe weaknesses, they provide strong indicators of the quality of a software system in each area. In most instances they will be highly correlated with the probability of operational or cost problems related to each measure's area.

Recent research in analyzing structural quality weaknesses has identified common patterns of code structures that can be used to detect weaknesses. Many of these 'Detection Patterns' are shared across different weaknesses. Detection Patterns will be used in this specification to organize and simplify the presentation of weaknesses underlying the four structural quality measures. Each weakness will be described as a quality measure element to remain consistent with ISO/IEC 25020. Each quality measure element will be represented as one or more Detection Patterns. Many quality measure elements (weaknesses) will share one or more Detection Patterns in common.

The normative portion of this specification represents each quality attribute (weakness) and quality measure element (detection pattern) using the Structured Patterns Metamodel Standard (SPMS). The code-based elements in these patterns are represented using the Knowledge Discovery Metamodel (KDM). The score for each of the four Automated Source Code Quality Measures from their quality measure elements is calculated by counting the number of detection patterns for each weakness, and then summing these numbers for all the weaknesses included in the specific quality characteristic measure.

2 Conformance

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

- ***Automated***—The analysis of the source code and counting of weaknesses shall be fully automated. The initial inputs required to prepare the source code for analysis include the source code of the application, the artifacts and information needed to configure the application for operation, and any available description of the architectural layers in the application.

- ***Objective***—After the source code has been prepared for analysis using the information provided as inputs, the analysis, calculation, and presentation of results shall not require further human intervention. The analysis and calculation shall be able to repeatedly produce the same results and outputs on the same body of software.

- ***Transparent***—Implementations that conform to this specification shall clearly list all source code (including versions), non-source code artifacts, and other information used to prepare the source code for submission to the analysis.

- ***Verifiable***—Compliance with this specification requires that an implementation shall 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 shall be clearly described and itemized so that they can be audited by a third party.

3 Normative References

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

- Structured Patterns Metamodel Standard, formal/2017-11-01, https://www.omg.org/spec/SPMS/1.2/

- ISO/IEC 19506:2012 – Object Management Group Architecture Driven Modernization (ADM) – Knowledge Discovery Metamodel (KDM). Also, Knowledge Discovery Metamodel, version 1.4 (KDM), formal/2016-09-01, https://www.omg.org/spec/KDM/1.4/

- MOF/XMI Mapping, version 2.5.1 (XMI), https://www.omg.org/spec/XMI/2.5.1/

- ISO/IEC 25010:2011 Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models

- ISO/IEC 25020:2019 Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Measurement reference model and guide

- ISO/IEC 19515:2019, Automated Function Points. Information technology -- Object Management Group Automated Function Points (AFP), 1.0. Geneva, Switzerland. Also, Object Management Group (2014). Automated Function Points - formal/2014-01-03 https://www.omg.org/spec/AFP/ . Needham, MA: Object Management Group.

- ITU-T X.1524 – Series X: Data Networks, Open System Communications and Security – Cybersecurity information exchange – Vulnerability/state exchange – Common weakness enumeration.

4 Terms and Definitions

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

4.1
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)

4.2
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)

4.3
Contributing Weakness
a weakness that is represented as a child of a parent weakness in the Common Weakness Enumeration, that is, a variant instantiation of the parent weakness (cwe.mitre.org)

4.4
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)

4.5
Detection Pattern
an abstract representation of a set of parsed program elements and their relations described in a formal representation language that provides guidance for detecting a specific weakness in the software source code.

4.6
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)

4.7
Maintainability
capability of a product to be modified by the intended maintainers with effectiveness and efficiency (ISO/IEC 25010)

4.8
Parent Weakness
a weakness in the Common Weakness Enumeration that has numerous possible instantiations in software that are represented by its relation to child CWEs (cwe.mitre.org)

4.9
Performance Efficiency
capability of a product to use an appropriate amount of resources under stated conditions (ISO/IEC 25010)

4.10

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)

4.11
Reliability
capability a product, to perform specified functions under specified conditions for a specified period of time (ISO/IEC 25010)

4.12
Security
capability of a product to protect 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, and to defend against attack patterns by malicious actors (ISO/IEC 25010)

4.13
Software Product
a set of computer programs, procedures, and possibly associated documentation and data. (ISO/IEC 25010)

4.14
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)

4.15
Software Quality
degree to which a software product satisfies stated and implied needs when used under specified conditions. (ISO/IEC 25010)

4.16
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)

4.17
Software Quality Characteristic
a set of software quality attributes that affect a specific category of software quality outcomes. (similar to but more specific than ISO/IEC 25010)

4.18
Software Quality Characteristic Measure
a software quality measure derived from measuring the attributes related to a specific software quality characteristic. (ISO/IEC 25020)

4.19
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)

4.20
Software Quality Measurement

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

4.21
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)

4.22
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.

4.23
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)

4.24
Structural Element
a component of software code that can be uniquely identified and counted such as a token, decision, variable, etc.

4.25
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.

4.26
Weakness
is a specific structure of program elements in the software source code, sometimes referred to as a software anti-pattern, that is inconsistent with good architectural or coding practice, violates a software quality rule, and can lead to operational or cost problems. (derived from cwe.mitre.com)

4.27
Weakness Pattern
A formal description constructed from arranging the elements output from parsing software source code into a software pattern metamodel structure that is identified as a weakness.

5 Symbols (and Abbreviated Terms)

AFP — Automated Function Points

ASCMM — Automated Source Code Maintainability Measure

ASCPEM — Automated Source Code Performance Efficiency Measure

ASCQM — Automated Source Code Quality Measure

ASCRM — Automated Source Code Reliability Measure

ASCSM — Automated Source Code Security Measure

CWE — Common Weakness Enumeration

CISQ — Consortium for IT Software Quality

KDM — Knowledge Discovery Metamodel

SMM — Structured Metrics Metamodel

SPMS — Structured Pattern Metamodel Standard

6 Weaknesses Included in Quality Measures and Representation Metamodels

6.1 Purpose

The purpose of this clause is to enumerate the weaknesses constituting each measure and to provide a simple description of each measure. Clause 6.2 describes the input needed for analyzing the source code to detect the weaknesses. Clause 6.2 introduces the concept of weaknesses.as quality measure elements in calculating the measures. Clauses 6.3 to 6.6 enumerate the weaknesses included in each measure. These clauses also indicate the status of each weakness as a parent weakness or a contributor weakness that represents a specific instantiation of a parent weakness. Clause 6.8 introduces how weaknesses will be represented in formal metalanguages. Clauses 6.9 and 10 describe the metamodels that will be used for specifying weaknesses and their detection patterns. Clause 6.11 provides a reading guide for understanding the content of Clauses 6.9 and 6.10.

6.2 Software Product Inputs

The following inputs are needed by static code analyzers 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.
- All materials and information required to prepare the application for production.
- A list of vetted libraries that are being used to sanitize data against potential attacks.
- What routines/API calls are being used for remote authentication, to any custom initialization and clean up routines, to synchronize resources, or to neutralize accepted file types or the names of resources.

Static code analyzers will also need a list of the violations that constitute each quality element in the Automated Source Code Security Measure.

6.3 Automated Source Code Quality Measure Elements

The weaknesses violating software quality rules that compose the CISQ Automated Source Code Quality Measures are grouped by quality measure in the clauses 6.4 through 6.7. Some of the weaknesses are included in more than one quality measure because they can cause several types of problems. The Common Weakness Enumeration repository (CWE, Annex B) has recently been expanded to include weaknesses from quality characteristics beyond security. All weaknesses included in these measures are identified by their CWE number from the repository. In most cases the description of CWEs is taken from information in the online repository (cwe.mitre.org). The mappings of the weaknesses from the previous CISQ measures to the current measures are presented in Annex C.

Some weaknesses drawn from the CWE repository (parent weaknesses) have related weaknesses listed as 'contributing weaknesses' ('children' in the CWE). Contributing weaknesses represent variants of how the parent weakness can be instantiated in software. In the following tables the cells containing CWE IDs for parents are presented in a darker blue than the cells containing contributing weaknesses. Based on their severity, not all children were included. The parent contributor relationships are informative material and are presented for informational purposes only. All weaknesses listed in these tables, regardless of whether they are parents or contributors, are to be treated as Quality Measure Elements in calculating the measures presented in 6.4 through 6.7.

6.4 Automated Source Code Maintainability Measure Element Descriptions

The quality measure elements (weaknesses violating software quality rules) that compose the CISQ Automated Source Code Maintainability Measure are presented in Table 1. This measure contains 29 parent weaknesses and no contributing weaknesses. Thus, Maintainability contains 29 Quality Measure Elements.

Table 1 — Quality Measure Elements for Automated Source Code Maintainability Measure

CWE #	Descriptor	Weakness Description
CWE-407	Algorithmic Complexity	An algorithm in a product has an inefficient worst-case computational complexity that may be detrimental to system performance and can be triggered by an attacker, typically using crafted manipulations that ensure that the worst case is being reached.
CWE-478	Missing Default Case in Switch Statement	The code does not have a default case in a switch statement, which might lead to complex logical errors and resultant weaknesses.
CWE-480	Use of Incorrect Operator	The programmer accidentally uses the wrong operator, which changes the application logic in security-relevant ways.
CWE-484	Omitted Break Statement in Switch	The program omits a break statement within a switch or similar construct, causing code associated with multiple conditions to execute. This can cause problems when the programmer only intended to execute code associated with one condition.
CWE-561	Dead code	The software contains dead code that can never be executed. (Thresholds are set at 5% logically dead code or 0% for code that is structurally dead. Code that exists in the source but not in the object does not count.)
CWE-570	Expression is Always False	The software contains an expression that will always evaluate to false.
CWE-571	Expression is Always True	The software contains an expression that will always evaluate to true.
CWE-783	Operator Precedence Logic Error	The program uses an expression in which operator precedence causes incorrect logic to be used.
CWE-1041	Use of Redundant Code (Copy-Paste)	The software has multiple functions, methods, procedures, macros, etc. that contain the same code. (The default threshold for each instance of copy-pasted code sets the maximum number of allowable copy-pasted instructions at 10% of the total instructions in the instance, *alternate thresholds can be set prior to analysis*).

CWE #	Descriptor	Weakness Description
CWE-1045	Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor	A parent class has a virtual destructor method, but the parent has a child class that does not have a virtual destructor.
CWE-1047	Modules with Circular Dependencies	The software contains modules in which one module has references that cycle back to itself, i.e., there are circular dependencies.
CWE-1048	Invokable Control Element with Large Number of Outward Calls (Excessive Coupling or Fan-out)	The code contains callable control elements that contain an excessively large number of references to other application objects external to the context of the callable, i.e. a Fan-Out value that is excessively large. (default threshold for the maximum number of references is 5, *alternate threshold can be set prior to analysis*)
CWE-1051	Initialization with Hard-Coded Network Resource Configuration Data	The software initializes data using hard-coded values that act as network resource identifiers.
CWE-1052	Excessive Use of Hard-Coded Literals in Initialization	The software initializes a data element using a hard-coded literal that is not a simple integer or static constant element.
CWE-1054	Invocation of a Control Element at an Unnecessarily Deep Horizontal Layer (Layer-skipping Call)	The code at one architectural layer invokes code that resides at a deeper layer than the adjacent layer, i.e., the invocation skips at least one layer, and the invoked code is not part of a vertical utility layer that can be referenced from any horizontal layer.
CWE-1055	Multiple Inheritance from Concrete Classes	The software contains a class with inheritance from more than one concrete class.
CWE-1062	Parent Class Element with References to Child Class	The code has a parent class that contains references to a child class, its methods, or its members.
CWE-1064	Invokable Control Element with Signature Containing an Excessive Number of Parameters	The software contains a function, subroutine, or method whose signature has an unnecessarily large number of parameters/arguments. (default threshold for the maximum number of parameters is 7, *alternate threshold can be set prior to analysis*).
CWE-1074	Class with Excessively Deep Inheritance	A class has an inheritance level that is too high, i.e., it has many parent classes. (default threshold for maximum Inheritance levels is 7, *alternate threshold can be set prior to analysis*).
CWE-1075	Unconditional Control Flow Transfer outside of Switch Block	The software performs unconditional control transfer (such as a "goto") in code outside of a branching structure such as a switch block.

CWE #	Descriptor	Weakness Description
CWE-1079	Parent Class without Virtual Destructor Method	A parent class contains one or more child classes, but the parent class does not have a virtual destructor method.
CWE-1080	Source Code File with Excessive Number of Lines of Code	A source code file has too many lines of code. (default threshold for the maximum lines of code is 1000, *alternate threshold can be set prior to analysis*).
CWE-1084	Invokable Control Element with Excessive File or Data Access Operations	A function or method contains too many operations that utilize a data manager or file resource. (default threshold for the maximum number of SQL or file operations is 7, *alternate threshold can be set prior to analysis*).
CWE-1085	Invokable Control Element with Excessive Volume of Commented-out Code	A function, method, procedure, etc. contains an excessive amount of code that has been commented out within its body. (default threshold for the maximum percent of commented-out instructions is 2%, *alternate threshold can be set prior to analysis*).
CWE-1086	Class with Excessive Number of Child Classes	A class contains an unnecessarily large number of children. (default threshold for the maximum number of children of a class is 10, *alternate threshold can be set prior to analysis*).
CWE-1087	Class with Virtual Method without a Virtual Destructor	A class contains a virtual method, but the method does not have an associated virtual destructor.
CWE-1090	Method Containing Access of a Member Element from Another Class	A method for a class performs an operation that directly accesses a member element from another class.
CWE-1095	Loop Condition Value Update within the Loop	The software uses a loop with a control flow condition based on a value that is updated within the body of the loop.
CWE-1121	Excessive McCabe Cyclomatic Complexity	A module, function, method, procedure, etc. contains McCabe cyclomatic complexity that exceeds a desirable maximum. (default threshold for Cyclomatic Complexity is 20, *alternate threshold can be set prior to analysis*).

6.5 Automated Source Code Performance Efficiency Measure Element Descriptions

The quality measure elements (weaknesses violating software quality rules) that compose the CISQ Automated Source Code Performance Efficiency Measure are presented in Table 2. This measure contains 16 parent weaknesses and 3 contributing weaknesses (children in the CWE) that represent variants of these weaknesses. The CWE numbers for contributing weaknesses are presented in lighter gray cells immediately below the parent weakness whose CWE number is in a darker gray cell. Performance Efficiency contains 19 Quality Measure Elements.

Table 2 — Quality Measure Elements for Automated Source Code Performance Efficiency Measure

CWE #	Descriptor	Weakness Description
CWE-404	Improper Resource Shutdown or Release	The program does not release or incorrectly releases a resource before it is made available for re-use.
CWE-401	Improper Release of Memory Before Removing Last Reference ('Memory Leak')	The software does not sufficiently track and release allocated memory after it has been used, which slowly consumes remaining memory.
CWE-772	Missing Release of Resource after Effective Lifetime	The software does not release a resource after its effective lifetime has ended, i.e., after the resource is no longer needed.
CWE-775	Missing Release of File Descriptor or Handle after Effective Lifetime	The software does not release a file descriptor or handle after its effective lifetime has ended, i.e., after the file descriptor/handle is no longer needed. When a file descriptor or handle is not released after use (typically by explicitly closing it), attackers can cause a denial of service by consuming all available file descriptors/handles, or otherwise preventing other system processes from obtaining their own file descriptors/handles.
CWE-424	Improper Protection of Alternate Path	The product does not sufficiently protect all possible paths that a user can take to access restricted functionality or resources. When data storage relies on a DBMS, special care shall be given to secure all data accesses and ensure data integrity.
CWE-1042	Static Member Data Element outside of a Singleton Class Element	The code contains a member element that is declared as static (but not final), in which its parent class element is not a singleton class - that is, a class element that can be used only once in the 'to' association of a Create action.
CWE-1043	Data Element Aggregating an Excessively Large Number of Non-Primitive Elements	The software uses a data element that has an excessively large number of sub-elements with non-primitive data types such as structures or aggregated objects. (default threshold for the maximum number of aggregated non-primitive data types is 5, *alternate threshold can be set prior to analysis*).
CWE-1046	Creation of Immutable Text Using String Concatenation	This programming pattern can be inefficient in comparison with use of text buffer data elements. This issue can make the software perform more slowly. If the relevant code is reachable by an attacker, then this performance problem might introduce a vulnerability.

CWE #	Descriptor	Weakness Description
CWE-1049	Excessive Data Query Operations in a Large Data Table	The software performs a data query with many joins and sub-queries on a large data table. (default thresholds are 5 joins, 3 sub-queries, and 1,000,000 rows for a large table, *alternate thresholds for all three parameters can be set prior to analysis*).
CWE-1050	Excessive Platform Resource Consumption within a Loop	The software has a loop body or loop condition that contains a control element that directly or indirectly consumes platform resources, e.g. messaging, sessions, locks, or file descriptors. (default threshold for resource consumption should be set based on the system architecture *prior to analysis*).
CWE-1057	Data Access Operations Outside of Expected Data Manager Component	The software uses a dedicated, central data manager component as required by design, but it contains code that performs data-access operations that do not use this data manager. Notes: · The dedicated data access component can be either client-side or server-side, which means that data access components can be developed using non-SQL language. · If there is no dedicated data access component, every data access is a weakness. · For some embedded software that requires access to data from anywhere, the whole software is defined as a data access component. This condition must be identified as input to the analysis.
CWE-1060	Excessive Number of Inefficient Server-Side Data Accesses	The software performs too many data queries without using efficient data processing functionality such as stored procedures. (default threshold for maximum number of data queries is **5**, *alternate threshold can be set prior to analysis*).
CWE-1067	Excessive Execution of Sequential Searches of Data Resource	The software contains a data query against a SQL table or view that is configured in a way that does not utilize an index and may cause sequential searches to be performed. (default threshold for a weakness to be counted is a query on a table of at least 500 rows, or an alternate threshold recommended by the database vendor. No weakness should be counted under conditions where the vendor recommends an index should not be used. An *alternate threshold can be set prior to analysis*).

CWE #	Descriptor	Weakness Description
CWE-1072	Data Resource Access without Use of Connection Pooling	The software accesses a data resource through a database without using a connection pooling capability. (the use of a connection pool is technology dependent; for example, connection pooling is disabled with the addition of 'Pooling=false' to the connection string with ADO.NET or the value of a 'com.sun.jndi.ldap.connect.pool' environment parameter in Java).
CWE-1073	Non-SQL Invokable Control Element with Excessive Number of Data Resource Accesses	The software contains a client with a function or method that contains many data accesses/queries that are sent through a data manager, i.e., does not use efficient database capabilities. (default threshold for the maximum number of data queries is 2, *alternate threshold can be set prior to analysis*).
CWE-1089	Large Data Table with Excessive Number of Indices	The software uses a large data table (default is 1,000,000 rows; *alternate threshold can be set prior to analysis*) that contains an excessively large number of indices. (default threshold for the maximum number of indices is 3, *alternate threshold can be set prior to analysis*).
CWE-1091	Use of Object without Invoking Destructor Method	The software contains a method that accesses an object but does not later invoke the element's associated finalize/destructor method.
CWE-1094	Excessive Index Range Scan for a Data Resource	The software contains an index range scan for a large data table, (default threshold is 1,000,000 rows, *alternate threshold can be set prior to analysis*) but the scan can cover many rows. (default threshold for the index range is 10, *alternate threshold can be set prior to analysis).*

6.6 Automated Source Code Reliability Measure Element Descriptions

The quality measure elements (weaknesses violating software quality rules) that compose the CISQ Automated Source Code Reliability Measure are presented in Table 3. This measure contains 35 parent weaknesses and 39 contributing weaknesses (children in the CWE) that represent variants of these weaknesses. The CWE numbers for contributing weaknesses are presented in lighter gray cells immediately below the parent weakness whose CWE number is in a darker gray cell. Reliability contains 74 Quality Measure Elements.

Table 3 — Quality Measure Elements for Automated Source Code Reliability Measure

CWE #	Descriptor	Weakness description
CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.
CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.
CWE-123	Write-what-where condition	Any condition where the attacker can write an arbitrary value to be written to an arbitrary location, often as the result of a buffer overflow.
CWE-125	Out-of-bounds read	The software reads data past the end, or before the beginning, of the intended buffer.
CWE-130	Improper Handling of Length Parameter Inconsistency	The software parses a formatted message or structure, but it does not handle or incorrectly handles a length field that is inconsistent with the actual length of the associated data.
CWE-786	Access of Memory Location Before Start of Buffer	The software reads or writes to a buffer using an index or pointer that references a memory location prior to the beginning of the buffer. This typically occurs when a pointer or its index is decremented to a position before the buffer, when pointer arithmetic results in a position before the beginning of the valid memory location, or when a negative index is used.
CWE-787	Out-of-bounds Write	The software writes data past the end, or before the beginning, of the intended buffer. The software may modify an index or perform pointer arithmetic that references a memory location that is outside of the boundaries of the buffer.

CWE #	Descriptor	Weakness description
CWE-788	Access of Memory Location After End of Buffer	The software reads or writes to a buffer using an index or pointer that references a memory location after the end of the buffer. This typically occurs when a pointer or its index is decremented to a position before the buffer; when pointer arithmetic results in a position before the buffer; or when a negative index is used, which generates a position before the buffer.
CWE-805	Buffer Access with Incorrect Length Value	The software uses a sequential operation to read or write a buffer, but it uses an incorrect length value that causes it to access memory that is outside of the bounds of the buffer.
CWE-822	Untrusted Pointer Dereference	The program obtains a value from an untrusted source, converts this value to a pointer, and dereferences the resulting pointer. There are several variants of this weakness, including but not necessarily limited to: The untrusted value is directly invoked as a function call. In OS kernels or drivers where there is a boundary between "userland" and privileged memory spaces, an untrusted pointer might enter through an API or system call (see CWE-781 for one such example). Inadvertently accepting the value from an untrusted control sphere when it did not have to be accepted as input at all. This might occur when the code was originally developed to be run by a single user in a non-networked environment, and the code is then ported to or otherwise exposed to a networked environment.
CWE-823	Use of Out-of-range Pointer Offset	The program performs pointer arithmetic on a valid pointer, but it uses an offset that can point outside of the intended range of valid memory locations for the resulting pointer. While a pointer can contain a reference to any arbitrary memory location, a program typically only intends to use the pointer to access limited portions of memory, such as contiguous memory used to access an individual array. Programs may use offsets to access fields or sub-elements stored within structured data. The offset might be out-of-range if it comes from an untrusted source, is the result of an incorrect calculation, or occurs because of another error.
CWE-824	Access of Uninitialized Pointer	The program accesses or uses a pointer that has not been initialized. If the pointer contains an uninitialized value, then the value might not point to a valid memory location.
CWE-825	Expired Pointer Dereference	The program dereferences a pointer that contains a location for memory that was previously valid, but is no longer valid.
CWE-170	Improper Null Termination	The software does not terminate or incorrectly terminates a string or array with a null character or equivalent terminator.

CWE #	Descriptor	Weakness description
CWE-252	Unchecked Return Value	The software does not check the return value from a method or function, which can prevent it from detecting unexpected states and conditions.
CWE-390	Detection of Error Condition Without Action	The software detects a specific error, but takes no actions to handle the error. For instance, where an exception handling block (such as Catch and Finally blocks) do not contain any instruction, making it impossible to accurately identify and adequately respond to unusual and unexpected conditions.
CWE-394	Unexpected Status Code or Return Value	The software does not properly check when a function or operation returns a value that is legitimate for the function, but is not expected by the software.
CWE-404	Improper Resource Shutdown or Release	The program does not release or incorrectly releases a resource before it is made available for re-use.
CWE-401	Improper Release of Memory Before Removing Last Reference ('Memory Leak')	The software does not sufficiently track and release allocated memory after it has been used, which slowly consumes remaining memory.
CWE-772	Missing Release of Resource after Effective Lifetime	The software does not release a resource after its effective lifetime has ended, i.e., after the resource is no longer needed.
CWE-775	Missing Release of File Descriptor or Handle after Effective Lifetime	The software does not release a file descriptor or handle after its effective lifetime has ended, i.e., after the file descriptor/handle is no longer needed. When a file descriptor or handle is not released after use (typically by explicitly closing it), attackers can cause a denial of service by consuming all available file descriptors/handles, or otherwise preventing other system processes from obtaining their own file descriptors/handles.
CWE-424	Improper Protection of Alternate Path	The product does not sufficiently protect all possible paths that a user can take to access restricted functionality or resources. When data storage relies on a DBMS, special care shall be given to secure all data accesses and ensure data integrity.
CWE-459	Incomplete Clean-up	The software does not properly "clean up" and remove temporary or supporting resources after they have been used.
CWE-476	NULL Pointer Dereference	A NULL pointer dereference occurs when the application dereferences a pointer that it expects to be valid, but is NULL, typically causing a crash or exit.

CWE #	Descriptor	Weakness description
CWE-480	Use of Incorrect Operator	The programmer accidentally uses the wrong operator, which changes the application logic in security-relevant ways.
CWE-484	Omitted Break Statement in Switch	The program omits a break statement within a switch or similar construct, causing code associated with multiple conditions to execute. This can cause problems when the programmer only intended to execute code associated with one condition.
CWE-562	Return of Stack Variable Address	A function returns the address of a stack variable, which will cause unintended program behavior, typically in the form of a crash. Because local variables are allocated on the stack, when a program returns a pointer to a local variable, it is returning a stack address. A subsequent function call is likely to re-use this same stack address, thereby overwriting the value of the pointer, which no longer corresponds to the same variable since a function's stack frame is invalidated when it returns. At best this will cause the value of the pointer to change unexpectedly. In many cases it causes the program to crash the next time the pointer is dereferenced.
CWE-595	Comparison of Object References Instead of Object Contents	The program compares object references instead of the contents of the objects themselves, preventing it from detecting equivalent objects.
CWE-597	Use of Wrong Operator in String Comparison	The software uses the wrong operator when comparing a string, such as using "==" when the equals() method should be used instead. In Java, using == or != to compare two strings for equality compares two objects for equality, not their values.
CWE-1097	Persistent Storable Data Element without Associated Comparison Control Element	The software uses a storable data element that does not have all the associated functions or methods that are necessary to support comparison. Remove instances where the persistent data has missing or improper dedicated comparison operations. Note: * In case of technologies with classes, this means situations where a persistent field is from a class that is made persistent while it does not implement methods from the list of required comparison operations (a JAVA example is the list composed of {'hashCode()','equals()'} methods)
CWE-662	Improper Synchronization	The software attempts to use a shared resource in an exclusive manner, but does not prevent or incorrectly prevents use of the resource by another thread or process.
CWE-366	Race Condition within a Thread	If two threads of execution use a resource simultaneously, there exists the possibility that resources may be used while invalid, in turn making the state of execution undefined.

CWE #	Descriptor	Weakness description
CWE-543	Use of Singleton Pattern Without Synchronization in a Multithreaded Context	The software uses the singleton pattern when creating a resource within a multithreaded environment.
CWE-567	Unsynchronized Access to Shared Data in a Multithreaded Context	The product does not properly synchronize shared data, such as static variables across threads, which can lead to undefined behavior and unpredictable data changes.
CWE-667	Improper Locking	The software does not properly acquire a lock on a resource, or it does not properly release a lock on a resource, leading to unexpected resource state changes and behaviors.
CWE-764	Multiple Locks of a Critical Resource	The software locks a critical resource more times than intended, leading to an unexpected state in the system.
CWE-820	Missing Synchronization	The software utilizes a shared resource in a concurrent manner but does not attempt to synchronize access to the resource.
CWE-821	Incorrect Synchronization	The software utilizes a shared resource in a concurrent manner, but it does not correctly synchronize access to the resource.
CWE-1058	Invokable Control Element in Multi-Thread Context with non-Final Static Storable or Member Element	The code contains a function or method that operates in a multi-threaded environment but owns an unsafe non-final static storable or member data element.
CWE-1096	Singleton Class Instance Creation without Proper Locking or Synchronization	The software implements a Singleton design pattern but does not use appropriate locking or other synchronization mechanism to ensure that the singleton class is only instantiated once.
CWE-665	Improper Initialization	The software does not initialize or incorrectly initializes a resource, which might leave the resource in an unexpected state when it is accessed or used.
CWE-456	Missing Initialization of a Variable	The software does not initialize critical variables, which causes the execution environment to use unexpected values.
CWE-457	Use of uninitialized variable	The code uses a variable that has not been initialized, leading to unpredictable or unintended results.
CWE-672	Operation on a Resource after Expiration or Release	The software uses, accesses, or otherwise operates on a resource after that resource has been expired, released, or revoked.

CWE #	Descriptor	Weakness description
CWE-415	Double Free	The product calls free() twice on the same memory address, potentially leading to modification of unexpected memory locations.
CWE-416	Use After Free	Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.
CWE-681	Incorrect Conversion between Numeric Types	When converting from one data type to another, such as long to integer, data can be omitted or translated in a way that produces unexpected values. If the resulting values are used in a sensitive context, then dangerous behaviors may occur. For instance, if the software declares a variable, field, member, etc. with a numeric type, and then updates it with a value from a second numeric type that is incompatible with the first numeric type.
CWE-194	Unexpected Sign Extension	The software performs an operation on a number that causes it to be sign-extended when it is transformed into a larger data type. When the original number is negative, this can produce unexpected values that lead to resultant weaknesses.
CWE-195	Signed to Unsigned Conversion Error	The software uses a signed primitive and performs a cast to an unsigned primitive, which can produce an unexpected value if the value of the signed primitive cannot be represented using an unsigned primitive.
CWE-196	Unsigned to Signed Conversion Error	The software uses an unsigned primitive and performs a cast to a signed primitive, which can produce an unexpected value if the value of the unsigned primitive cannot be represented using a signed primitive.
CWE-197	Numeric Truncation Error	Truncation errors occur when a primitive is cast to a primitive of a smaller size and data is lost in the conversion. When a primitive is cast to a smaller primitive, the high order bits of the large value are lost in the conversion, potentially resulting in an unexpected value that is not equal to the original value. This value may be required as an index into a buffer, a loop iterator, or simply necessary state data. In any case, the value cannot be trusted and the system will be in an undefined state. While this method may be employed viably to isolate the low bits of a value, this usage is rare, and truncation usually implies that an implementation error has occurred.
CWE-682	Incorrect Calculation	The software performs a calculation that generates incorrect or unintended results that are later used in security-critical decisions or resource management.
CWE-131	Incorrect Calculation of Buffer Size	The software does not correctly calculate the size to be used when allocating a buffer, which could lead to a buffer overflow.

CWE #	Descriptor	Weakness description
CWE-369	Divide by Zero	The product divides a value by zero.
CWE-703	Improper Check or Handling of Exceptional Conditions	The software does not properly anticipate or handle exceptional conditions that rarely occur during normal operation of the software.
CWE-248	Uncaught Exception	An exception is thrown from a function, but it is not caught.
CWE-391	Unchecked Error Condition	Ignoring exceptions and other error conditions may allow an attacker to induce unexpected behavior unnoticed.
CWE-392	Missing Report of Error Condition	The software encounters an error but does not provide a status code or return value to indicate that an error has occurred.
CWE-704	Incorrect Type Conversion or Cast	The software does not correctly convert an object, resource, or structure from one type to a different type.
CWE-758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior	The software uses an API function, data structure, or other entity in a way that relies on properties that are not always guaranteed to hold for that entity.
CWE-833	Deadlock	The software contains multiple threads or executable segments that are waiting for each other to release a necessary lock, resulting in deadlock.
CWE-835	Loop with Unreachable Exit Condition ('Infinite Loop')	The program contains an iteration or loop with an exit condition that cannot be reached, i.e., an infinite loop.
CWE-908	Use of Uninitialized Resource	The software uses a resource that has not been properly initialized.
CWE-1045	Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor	A parent class has a virtual destructor method, but the parent has a child class that does not have a virtual destructor.
CWE-1051	Initialization with Hard-Coded Network Resource Configuration Data	The software initializes data using hard-coded values that act as network resource identifiers.
CWE-1066	Missing Serialization Control Element	The software contains a serializable data element that does not have an associated serialization method.
CWE-1070	Serializable Data Element Containing Non-Serializable Item Elements	The software contains a serializable, storable data element such as a field or member, but the data element contains member elements that are not serializable.

CWE #	Descriptor	Weakness description
CWE-1077	Floating Point Comparison with Incorrect Operator	The code performs a comparison such as an equality test between two float (floating point) values, but it uses comparison operators that do not account for the possibility of loss of precision. Numeric calculation using floating point values can generate imprecise results because of rounding errors. As a result, two different calculations might generate numbers that are mathematically equal, but have slightly different bit representations that do not translate to the same mathematically equal values. As a result, an equality test or other comparison might produce unexpected results. (an example in JAVA, is the use of '= =' or '! =') instead of being checked for precision.
CWE-1079	Parent Class without Virtual Destructor Method	A parent class contains one or more child classes, but the parent class does not have a virtual destructor method.
CWE-1082	Class Instance Self Destruction Control Element	The code contains a class instance that calls the method or function to delete or destroy itself. (an example of a self-destruction in C++ is 'delete this')
CWE-1083	Data Access from Outside Designated Data Manager Component	The software is intended to manage data access through a particular data manager component such as a relational or non-SQL database, but it contains code that performs data access operations without using that component. Notes: 1) The dedicated data access component can be either client-side or server-side, which means that data access components can be developed using non-SQL language. 2) If there is no dedicated data access component, every data access is a violation. 3) For some embedded software that requires access to data from anywhere, the whole software is defined as a data access component. This condition must be identified as input to the analysis.
CWE-1087	Class with Virtual Method without a Virtual Destructor	A class contains a virtual method, but the method does not have an associated virtual destructor.
CWE-1088	Synchronous Access of Remote Resource without Timeout	The code has a synchronous call to a remote resource, but there is no timeout for the call, or the timeout is set to infinite.
CWE-1098	Data Element containing Pointer Item without Proper Copy Control Element	The code contains a data element with a pointer that does not have an associated copy or constructor method.

6.7 Automated Source Code Security Measure Element Descriptions

The quality measure elements (weaknesses violating software quality rules) that compose the CISQ Automated Source Code Security Measure are presented in Table 4. This measure contains 36 parent weaknesses and 37 contributing weaknesses (children in the CWE) that represent variants of these weaknesses. The CWE numbers for contributing weaknesses are presented in lighter gray cells immediately below the parent weakness whose CWE number is in a darker gray cell. Security contains 73 Quality Measure Elements.

Table 4 — Quality Measure Elements for Automated Source Code Security Measure

CWE #	Descriptor	Weakness description
CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	The software uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the software does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory.
CWE-23	Relative Path Traversal	The software uses external input to construct a pathname that should be within a restricted directory, but it does not properly neutralize sequences such as ".." that can resolve to a location that is outside of that directory.
CWE-36	Absolute Path Traversal	The software uses external input to construct a pathname that should be within a restricted directory, but it does not properly neutralize absolute path sequences such as "/abs/path" that can resolve to a location that is outside of that directory.
CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	The software constructs all or part of a command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended command when it is sent to a downstream component.
CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	The software constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component.
CWE-88	Argument Injection or Modification	The software does not sufficiently delimit the arguments being passed to a component in another control sphere, allowing alternate arguments to be provided, leading to potentially security-relevant changes.

CWE #	Descriptor	Weakness description
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	The software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users. Cross-site scripting (XSS) vulnerabilities occur when: 1. Untrusted data enters a web application, typically from a web request. 2. The web application dynamically generates a web page that contains this untrusted data. 3. During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc. 4. A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data. 5. Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain. 6. This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.
CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	The software constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.
CWE-564	SQL Injection: Hibernate	Using Hibernate to execute a dynamic SQL statement built with user-controlled input can allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.
CWE-90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')	The software constructs all or part of an LDAP query using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended LDAP query when it is sent to a downstream component.

CWE #	Descriptor	Weakness description
CWE-91	XML Injection (aka Blind XPath Injection)	The software does not properly neutralize special elements that are used in XML, allowing attackers to modify the syntax, content, or commands of the XML before it is processed by an end system.
CWE-99	Improper Control of Resource Identifiers ('Resource injection')	The software receives input from an upstream component, but it does not restrict or incorrectly restricts the input before it is used as an identifier for a resource that may be outside the intended sphere of control.
CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.
CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.
CWE-123	Write-what-where condition	Any condition where the attacker can write an arbitrary value to an arbitrary location, often as the result of a buffer overflow.
CWE-125	Out-of-bounds Read	The software reads data past the end, or before the beginning, of the intended buffer.
CWE-130	Improper Handling of Length Parameter Inconsistency	The software parses a formatted message or structure, but it does not handle or incorrectly handles a length field that is inconsistent with the actual length of the associated data.
CWE-786	Access of Memory Location Before Start of Buffer	The software reads or writes to a buffer using an index or pointer that references a memory location prior to the beginning of the buffer. This typically occurs when a pointer or its index is decremented to a position before the buffer, when pointer arithmetic results in a position before the beginning of the valid memory location, or when a negative index is used.
CWE-787	Out-of-bounds Write	The software writes data past the end, or before the beginning, of the intended buffer. The software may modify an index or perform pointer arithmetic that references a memory location that is outside of the boundaries of the buffer.

CWE #	Descriptor	Weakness description
CWE-788	Access of Memory Location After End of Buffer	The software reads or writes to a buffer using an index or pointer that references a memory location after the end of the buffer. This typically occurs when a pointer or its index is decremented to a position before the buffer; when pointer arithmetic results in a position before the buffer; or when a negative index is used, which generates a position before the buffer.
CWE-805	Buffer Access with Incorrect Length Value	The software uses a sequential operation to read or write a buffer, but it uses an incorrect length value that causes it to access memory that is outside of the bounds of the buffer.
CWE-822	Untrusted Pointer Dereference	The program obtains a value from an untrusted source, converts this value to a pointer, and dereferences the resulting pointer. There are several variants of this weakness, including but not necessarily limited to: ▯ The untrusted value is directly invoked as a function call. ▯▯▯In OS kernels or drivers where there is a boundary between "userland" and privileged memory spaces, an untrusted pointer might enter through an API or system call (see CWE-781 for one such example). ▯ Inadvertently accepting the value from an untrusted control sphere when it did not have to be accepted as input at all. This might occur when the code was originally developed to be run by a single user in a non-networked environment, and the code is then ported to or otherwise exposed to a networked environment.
CWE-823	Use of Out-of-range Pointer Offset	The program performs pointer arithmetic on a valid pointer, but it uses an offset that can point outside of the intended range of valid memory locations for the resulting pointer. ▯ While a pointer can contain a reference to any arbitrary memory location, a program typically only intends to use the pointer to access limited portions of memory, such as contiguous memory used to access an individual array. ▯ Programs may use offsets to access fields or sub-elements stored within structured data. The offset might be out-of-range if it comes from an untrusted source, is the result of an incorrect calculation, or occurs because of another error.
CWE-824	Access of Uninitialized Pointer	The program accesses or uses a pointer that has not been initialized. If the pointer contains an uninitialized value, then the value might not point to a valid memory location.
CWE-825	Expired Pointer Dereference	The program dereferences a pointer that contains a location for memory that was previously valid, but is no longer valid.

CWE #	Descriptor	Weakness description
CWE-129	Improper Validation of Array Index	The product uses untrusted input when calculating or using an array index, but the product does not validate or incorrectly validates the index to ensure the index references a valid position within the array.
CWE-134	Use of Externally Controlled Format String	The software uses a function that accepts a format string as an argument, but the format string originates from an external source.
CWE-252	Unchecked Return Value	The software does not check the return value from a method or function, which can prevent it from detecting unexpected states and conditions.
CWE-404	Improper Resource Shutdown or Release	The program does not release or incorrectly releases a resource before it is made available for re-use.
CWE-401	Improper Release of Memory Before Removing Last Reference ('Memory Leak')	The software does not sufficiently track and release allocated memory after it has been used, which slowly consumes remaining memory.
CWE-772	Missing Release of Resource after Effective Lifetime	The software does not release a resource after its effective lifetime has ended, i.e., after the resource is no longer needed.
CWE-775	Missing Release of File Descriptor or Handle after Effective Lifetime	The software does not release a file descriptor or handle after its effective lifetime has ended, i.e., after the file descriptor/handle is no longer needed. When a file descriptor or handle is not released after use (typically by explicitly closing it), attackers can cause a denial of service by consuming all available file descriptors/handles, or otherwise preventing other system processes from obtaining their own file descriptors/handles.
CWE-424	Improper Protection of Alternate Path	The product does not sufficiently protect all possible paths that a user can take to access restricted functionality or resources. When data storage relies on a DBMS, special care shall be given to secure all data accesses and ensure data integrity.
CWE-434	Unrestricted Upload of File with Dangerous Type	The software allows the upload or transfer files of dangerous types that can be automatically processed within the product's environment.
CWE-477	Use of Obsolete Function	The code uses deprecated or obsolete functions, which suggests that the code has not been actively reviewed or maintained.

CWE #	Descriptor	Weakness description
CWE-480	Use of Incorrect Operator	The programmer accidentally uses the wrong operator, which changes the application logic in security-relevant ways.
CWE-502	Deserialization of Untrusted Data	The application deserializes untrusted data without sufficiently verifying that the resulting data will be valid.
CWE-570	Expression is Always False	The software contains an expression that will always evaluate to false.
CWE-571	Expression Is Always True	The software contains an expression that will always evaluate to true.
CWE-606	Unchecked Input for Loop Condition	The product does not properly check inputs that are used for loop conditions, potentially leading to a denial of service because of excessive looping.
CWE-611	Improper Restriction of XML External Entity Reference ('XXE')	The software processes an XML document that can contain XML entities with URIs that resolve to documents outside of the intended sphere of control, causing the product to embed incorrect documents into its output.
CWE-643	Improper Neutralization of Data within XPath Expressions ('XPath Injection')	The software uses external input to dynamically construct an XPath expression used to retrieve data from an XML database, but it does not neutralize or incorrectly neutralizes that input. This allows an attacker to control the structure of the query.
CWE-652	CWE-652 Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')	The software uses external input to dynamically construct a XQuery expression used to retrieve data from an XML database, but it does not neutralize or incorrectly neutralizes that input. This allows an attacker to control the structure of the query.
CWE-665	Improper Initialization	The software does not initialize or incorrectly initializes a resource, which might leave the resource in an unexpected state when it is accessed or used.
CWE-456	Missing Initialization of a Variable	The software does not initialize critical variables, which causes the execution environment to use unexpected values.
CWE-457	Use of uninitialized variable	The software uses a variable that has not been initialized leading to unpredictable or unintended results.
CWE-662	Improper Synchronization	The software attempts to use a shared resource in an exclusive manner, but does not prevent or incorrectly prevents use of the resource by another thread or process.

CWE #	Descriptor	Weakness description
CWE-366	Race Condition within a Thread	If two threads of execution use a resource simultaneously, there exists the possibility that resources may be used while invalid, in turn making the state of execution undefined.
CWE-543	Use of Singleton Pattern Without Synchronization in a Multithreaded Context	The software uses the singleton pattern when creating a resource within a multithreaded environment.
CWE-567	Unsynchronized Access to Shared Data in a Multithreaded Context	The product does not properly synchronize shared data, such as static variables across threads, which can lead to undefined behavior and unpredictable data changes.
CWE-667	Improper Locking	The software does not properly acquire a lock on a resource, or it does not properly release a lock on a resource, leading to unexpected resource state changes and behaviors.
CWE-820	Missing Synchronization	The software utilizes a shared resource in a concurrent manner but does not attempt to synchronize access to the resource.
CWE-821	Incorrect Synchronization	The software utilizes a shared resource in a concurrent manner, but it does not correctly synchronize access to the resource.
CWE-672	Operation on a Resource after Expiration or Release	The software uses, accesses, or otherwise operates on a resource after that resource has been expired, released, or revoked.
CWE-415	Double Free	The product calls free() twice on the same memory address, potentially leading to modification of unexpected memory locations.
CWE-416	Use After Free	Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.
CWE-681	Incorrect Conversion between Numeric Types	When converting from one data type to another, such as long to integer, data can be omitted or translated in a way that produces unexpected values. If the resulting values are used in a sensitive context, then dangerous behaviors may occur. For instance, if the software declares a variable, field, member, etc. with a numeric type, and then updates it with a value from a second numeric type that is incompatible with the first numeric type.

CWE #	Descriptor	Weakness description
CWE-194	Unexpected Sign Extension	The software performs an operation on a number that causes it to be sign-extended when it is transformed into a larger data type. When the original number is negative, this can produce unexpected values that lead to resultant weaknesses.
CWE-195	Signed to Unsigned Conversion Error	The software uses a signed primitive and performs a cast to an unsigned primitive, which can produce an unexpected value if the value of the signed primitive cannot be represented using an unsigned primitive.
CWE-196	Unsigned to Signed Conversion Error	The software uses an unsigned primitive and performs a cast to a signed primitive, which can produce an unexpected value if the value of the unsigned primitive cannot be represented using a signed primitive.
CWE-197	Numeric Truncation Error	Truncation errors occur when a primitive is cast to a primitive of a smaller size and data is lost in the conversion. When a primitive is cast to a smaller primitive, the high order bits of the large value are lost in the conversion, potentially resulting in an unexpected value that is not equal to the original value. This value may be required as an index into a buffer, a loop iterator, or simply necessary state data. In any case, the value cannot be trusted and the system will be in an undefined state. While this method may be employed viably to isolate the low bits of a value, this usage is rare, and truncation usually implies that an implementation error has occurred.
CWE-682	Incorrect Calculation	The software performs a calculation that generates incorrect or unintended results that are later used in security-critical decisions or resource management.
CWE-131	Incorrect Calculation of Buffer Size	The software does not correctly calculate the size to be used when allocating a buffer, which could lead to a buffer overflow.
CWE-369	Divide by Zero	The product divides a value by zero.
CWE-732	Incorrect Permission Assignment for Critical Resource	The software specifies permissions for a security-critical resource in a way that allows that resource to be read or modified by unintended actors.
CWE-778	Insufficient Logging	When a security-critical event occurs, the software either does not record the event or omits important details about the event when logging it.

CWE #	Descriptor	Weakness description
CWE-783	Operator Precedence Logic Error	The program uses an expression in which operator precedence causes incorrect logic to be used. While often just a bug, operator precedence logic errors can have serious consequences if they are used in security-critical code, such as making an authentication decision.
CWE-789	Uncontrolled Memory Allocation	The product allocates memory based on an untrusted size value, but it does not validate or incorrectly validates the size, allowing arbitrary amounts of memory to be allocated.
CWE-798	Use of Hard-coded Credentials	The software contains hard-coded credentials, such as a password or cryptographic key, which it uses for its own inbound authentication, outbound communication to external components, or encryption of internal data.
CWE-259	Use of Hard-coded Password	The software contains a hard-coded password, which it uses for its own inbound authentication or for outbound communication to external components.
CWE-321	Use of Hard-coded Cryptographic Key	The use of a hard-coded cryptographic key significantly increases the possibility that encrypted data may be recovered.
CWE-835	Loop with Unreachable Exit Condition ('Infinite Loop')	The program contains an iteration or loop with an exit condition that cannot be reached, i.e., an infinite loop.
CWE-917	Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')	The software constructs all or part of an expression language (EL) statement in a Java Server Page (JSP) using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended EL statement before it is executed.
CWE-1057	Data Access Operations Outside of Expected Data Manager Component	The software uses a dedicated, central data manager component as required by design, but it contains code that performs data-access operations that do not use this data manager. Notes: The dedicated data access component can be either client-side or server-side, which means that data access components can be developed using non-SQL language. If there is no dedicated data access component, every data access is a weakness. For some embedded software that requires access to data from anywhere, the whole software is defined as a data access component. This condition must be identified as input to the analysis.

6.8 Introduction to the Specification of Quality Measure Elements

Clauses 7, 8, and 9 display in human readable format the content of the machine readable XMI format file attached to this specification. The content of the machine readable XMI format file represents the Quality Measure Elements with the following conventions:

- Structural elements included in a weakness pattern are represented in the Knowledge Discovery Metamodel (KDM).

- Relations among the structural elements constituting a weakness pattern are represented in the Software Patterns Metamodel Standard (SPMS) to compute measures at the weakness level.

6.9 Knowledge Discovery Metamodel (KDM)

This specification uses the Knowledge Discovery Metamodel (KDM) to represent the parsed entities whose relationships create a weakness pattern. This specification uses KDM entities in the 'KDM outline' section of the pattern definitions to represent the code elements whose presence or absence indicates an occurrence of the weakness. Descriptions try to remain as generic, yet as accurate as possible, so that the pattern can be applied to as many situations as possible: different technologies, different programming languages, etc. This means:

1. The descriptions include information such as (MethodUnit), (Reads), (ManagesResource), … to identify the KDM entities included in the pattern definition.

2. The descriptions only describe the salient aspects of the pattern since the specifics can be technology or language dependent.

Detection Patterns presented in Clause 8 use micro-KDM to provide greater granularity to their specification of weakness patterns. Additional semantic constraints are required to coordinate producers and consumers of KDM models to use the KDM Program Element layer for control- and data-flow analysis applications, as well as for providing more precision for the Resource Layer and the Abstraction Layer. Micro-KDM achieves this by constraining the granularity of the leaf action elements and their meaning by providing the set of micro-actions with predefined semantics. Micro-KDM treats the original macro-action as a container that owns certain micro-actions with predefined semantics. Thus, precise semantics of the macro-action is defined. Thus, micro-KDM constrains the patterns of how to map the statements of the existing system as determined by the programming language into KDM. KDM is helpful for reading this clause. However, for readers not familiar with KDM, Table 5 presents a primer

Table 5 — Software elements translated into KDM wording

Software element	KDM Outline
function, method, procedure, stored procedure, sub-routine etc.	`CallableUnit|MethodUnit id="ce1" ...`
variable, field, member, etc.	`StorableUnit|MemberUnit id="de1" ...`

class, interface definition and use as a type, use as base class	`ClassUnit|InterfaceUnit id="cu1" ...`
	`StorableUnit id="su1" type="cu1" ...`
	`ClassUnit id="cu2" ...`
	`  Extends "cu1" ...`

method	`ClassUnit id="cu2" ...` `  MethodUnit "mu1" ...`
field, member	`ClassUnit id="cu2" ...` `  MemberUnit "mu1" ...`

SQL stored procedures	`DataModel`
	`  RelationalSchema ...`
	`  CallableUnit id="cu1" kind="stored" ...`

return code value definition and use	`CallableUnit|MethodUnit id="ce1" type="ce1_signature" ...`
	`  Signature "ce1_signature"`
	`  ParameterUnit id="pu1" kind="return" ...`
	`Value|StorableUnit|MemberUnit id="de1" ...`
	`ActionElement id="ae1"`
	`kind="Call|PtrCall|MethodCall|VirtualCall" ...`
	`  Calls "ce1"`
	`  Reads "de1"`

exception	`CallableUnit|MethodUnit id="ce1" type="ce1_signature" ...`
	`  Signature "ce1_signature"`
	`  ParameterUnit id="pu1" kind="exception" ...`

Software element	KDM Outline
user input data flow	UIModel UIField id="uf1" UIAction id="ua1" implementation="ae1" kind="input" ReadsUI "uf1" ... CodeModel ... StorableUnit id="su1" StorableUnit id="su2" ActionElement id="ae1" kind="UI" Writes "su1" Flow "ae2" ActionElement id="ae2" Flow "ae3" Reads "su1" Writes "su2" ActionElement id="ae3" Flow "ae4" ...
execution path	ActionElement id="ae1" kind="UI" Flow|Calls "ae2" ActionElement id="ae2" Flow|Calls "ae3" ActionElement id="ae3" Flow|Calls "ae4"
RDBMS	DataModel RelationalSchema ...

Software element	KDM Outline
for loop	ActionElement id="ae5" kind="Compound" StorableUnit id="su3" ActionElement id="ae6" kind="Assign" Reads ... Writes "su3" Flows "ae7" ActionElement id="ae7" kind="LessThan|LessThanOrEqual|GreaterThan|GreaterThanOrEqual" Reads "su3" Reads "su2" TrueFlow "ae8" FalseFlow "ff1" ActionElement id="ae8" kind=... ... ActionElement id="ae9" kind="Incr|Decr" Addresses "loopVariable" Flows "ae6" ActionElement id="ff1" kind="Nop"
while loop	ActionElement id="ae5" kind="Compound" BooleanType id="booleanType" DataElement id="de1" type="booleanType" EntryFlow "tf1" ActionElement id="tf1" ... ... ActionElement id ="ae6" kind="GreaterThan|GreaterThanOrEqual|LessThan|LessThanOrEqual" Reads "su2" ... Writes "de1" ActionElement id="ae7" kind="Condition" Reads "de1" TrueFlow "tf1" FalseFlow "ff1" ActionElement id="ff1"

Software element	KDM Outline
checked	Value|StorableUnit|MemberUnit id="de1" ... ActionElement id="ae1" kind="Equals|NotEqualTo|GreaterThan|GreaterThanOrEqual|LessThan|LessThanOrEqual" ... Reads "de1"

6.10 Software Patterns Metamodel Standard (SPMS)

This specification uses the Software Patterns Metamodel Standard (SPMS) to represent weaknesses as software patterns involving code elements and their relationships in source code. In the machine readable XMI format file attached to the current specification each weakness pattern is represented in SPMS Definitions Classes as follows:

- PatternDefinition (SPMS:PatternDefinition): the pattern specification describing a specific weakness and a specific detection pattern. In the context of this document, each Quality Measure Element is the count of occurrences of the SPMS detection pattern detected in the source code for a specific weakness related to the Quality Characteristic being measured.

- 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 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. Roles for weaknesses are abstractions, while the roles for detection patterns can be linked back to the code elements.

- 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 7 different PatternSections in use:

    - "Descriptor" ("descriptor" in the XMI document) to provide pattern signature, a visible interface of the pattern.

    - "Description" ("description" in XMI document) to provide a human readable explanation of the measure.

    - "KDM Outline" ("kdm outline" in XMI document) to provide an illustration of the essential elements related to KDM, in a human readable outline.

    - "What to report" ("reporting" in XMI document) to provide the list of elements to report to claim the finding of an occurrence of a detection pattern.

    - "Reference" ("reference" in XMI document) to provide pointers to the weakness description in the CWE repository.

    - "Usage name" ("usage_name" in XMI document) to provide a more user-friendly name to the weakness, generally the case when the weakness original name was too strongly KDM-flavored for the general audience.

SPMS Relationships Classes:

- MemberOf (SPMS:MemberOf): “An InterpatternRelationship specialized to indicate inclusion in a Category”.
- RelatedPattern (SPMS:RelatedPattern) with 4 different Natures (SPMS:Nature) (“DetectedBy”, “Detecting”,” AggregatedBy”, and “Aggregating”): InterpatternRelationships used to model the relations between weaknesses and detection patterns, and between parent and child weaknesses.
- 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”
    - “Maintainability”

6.11 Reading guide

For each numbered sub-clause in clause 7:

- Sub-clause 7.x represents the Software Quality characteristic addressed by the associated weakness patterns.
- Sub-clause 7.x.y represents the SPMS modeling associated with a weakness pattern for a specific weakness associated with the Software Quality characteristic.

Weakness pattern sub-clauses are summarizing the various aspects related to a weakness:

- (SPMS) usage name pattern section, if any
- (SPMS) reference pattern section
- (SPMS) roles
- (SPMS) contributing weaknesses and parent weakness, if any,
    - useful for reporting of weakness pattern-level information, aggregated or detailed
- (SPMS) detection patterns with reference numbers to subclauses in clause 8
    - useful for reporting of detection pattern-level findings at the weakness level
    - useful for counting the violations to the weakness, by summing the count of violations to its detection patterns

For each numbered sub-clause in clause 8:

- Sub-clause 8.x represents the SPMS modeling associated with a detection pattern

Detection pattern sub-clauses are summarizing the various aspects related to a detection pattern:

- (SPMS) descriptor, description, KDM outline, reporting pattern sections,
    - In description and reporting pattern sections, data between angle brackets (e.g.: <ControlElement>) identify SPMS roles

7 List of ASCQM Weaknesses

7.1 Weakness Category Maintainability

7.1.1 CWE-407 Algorithmic Complexity

Reference
https://cwe.mitre.org/data/definitions/407

Roles
- the <ControlFlow>

Detection Patterns
8.2.132 ASCQM Ban Switch in Switch Statement
8.2.117 ASCQM Limit Algorithmic Complexity via Cyclomatic Complexity Value
8.2.134 ASCQM Limit Algorithmic Complexity via Essential Complexity Value
8.2.133 ASCQM Limit Algorithmic Complexity via Module Design Complexity Value

7.1.2 CWE-478 Missing Default Case in Switch Statement

Reference
https://cwe.mitre.org/data/definitions/478

Roles
- the <SwitchStatement>

Detection Patterns
8.2.135 ASCQM Use Default Case in Switch Statement

7.1.3 Weakness CWE-480 Use of Incorrect Operator

Reference

https://cwe.mitre.org/data/definitions/480

Roles
- the <Operator>

Detection Patterns
8.2.24 ASCQM Ban Assignment Operation Inside Logic Blocks
8.2.25 ASCQM Ban Comparison Expression Outside Logic Blocks
8.2.23 ASCQM Ban Incorrect Object Comparison
8.2.26 ASCQM Ban Incorrect String Comparison
8.2.27 ASCQM Ban Logical Operation with a Constant Operand

7.1.4 CWE-484 Omitted Break Statement in Switch

Reference
https://cwe.mitre.org/data/definitions/484

Roles
- the <SwitchStatement>

Detection Patterns
8.2.64 ASCQM Use Break in Switch Statement

7.1.5 CWE-561 Dead Code

Reference
https://cwe.mitre.org/data/definitions/561

Roles
- the <DeadCode>

Detection Patterns
8.2.131 ASCQM Ban Exception Definition without Ever Throwing It
8.2.130 ASCQM Ban Logical Dead Code
8.2.125 ASCQM Ban Unreferenced Dead Code

7.1.6 CWE-570 Expression is Always False

Reference
https://cwe.mitre.org/data/definitions/570

Roles
- the <BooleanExpression>

Detection Patterns
8.2.79 ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use

7.1.7 CWE-571 Expression is Always True

Reference
https://cwe.mitre.org/data/definitions/571

Roles
- the <BooleanExpression>

Detection Patterns
8.2.79 ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use

7.1.8 CWE-783 Operator Precedence Logic Error

Reference
https://cwe.mitre.org/data/definitions/783

Roles
- the <Formula>

Detection Patterns
8.2.82 ASCQM Ban Incorrect Joint Comparison
8.2.81 ASCQM Ban Not Operator On Non-Boolean Operand Of Comparison Operation
8.2.80 ASCQM Ban Not Operator On Operand Of Bitwise Operation

7.1.9 CWE-1075 Unconditional Control Flow Transfer Outside of Switch Block

Usage name
Control transferred outside switch statement

Reference
https://cwe.mitre.org/data/definitions/1075

Roles
- the <SwitchBlock>
- the <ControlFlowTransfer>

Detection Patterns
8.2.122 ASCQM Ban Control Flow Transfer

7.1.10 CWE-1121 Excessive McCabe Cyclomatic Complexity Value

Usage name
Excessive Cyclomatic Complexity

Reference
https://cwe.mitre.org/data/definitions/1121

Roles
- the <Operation>
- the <ControlFlow>

Detection Patterns
8.2.117 ASCQM Limit Algorithmic Complexity via Cyclomatic Complexity Value

7.1.11 CWE-1054 Invocation of a Control Element at an Unnecessarily Deep Horizontal Layer (Layer-skipping Call)

Usage name
Layer-skipping calls

Reference
https://cwe.mitre.org/data/definitions/1054

Roles
- the <Layer1>
- the <Layer2>
- the <Call>

Detection Patterns
8.2.44 ASCQM Ban Unintended Paths

7.1.12 CWE-1064 Invokable Control Element with Signature Containing an Excessive Number of Parameters

Usage name
Excessive parameterization

Reference
https://cwe.mitre.org/data/definitions/1064

Roles
- the <OperationSignature>

Detection Patterns
8.2.124 ASCQM Limit Number of Parameters

7.1.13 CWE-1084 Invokable Control Element with Excessive File or Data Access Operations

Usage name
Control element with excessive data operations

Reference
https://cwe.mitre.org/data/definitions/1084

Roles
- the <Operation>
- the <DataAccesses>

Detection Patterns
8.2.118 ASCQM Limit Number of Data Access

7.1.14 CWE-1051 Initialization with Hard-Coded Network Resource Configuration Data

Usage name
Hard-coded network resource information

Reference
https://cwe.mitre.org/data/definitions/1051

Roles
- the <NetworkResourceAccess>
- the <HardCodedValue>

Detection Patterns
8.2.43 ASCQM Ban Hard-Coded Literals used to Connect to Resource

7.1.15 CWE-1090 Method Containing Access of a Member Element from Another Class

Usage name
Cross element data access

Reference
https://cwe.mitre.org/data/definitions/1090

Roles
- the <Class1>
- the <Class2>
- the <Reference>

Detection Patterns
8.2.121 ASCQM Ban Usage of Data Elements from Other Classes

7.1.16 CWE-1074 Class with Excessively Deep Inheritance

Usage name
Excessive inheritance levels

Reference
https://cwe.mitre.org/data/definitions/1074

Roles
the <ClassInheritenceTree>

Detection Patterns
8.2.120 ASCQM Ban Excessive Number of Inheritance Levels

7.1.17 CWE-1086 Class with Excessive Number of Child Classes

Usage name
Excessive child classes

Reference
https://cwe.mitre.org/data/definitions/1086

Roles
- the <Class>
- the <Children>

Detection Patterns
8.2.119 ASCQM Ban Excessive Number of Children

7.1.18 CWE-1041 Use of Redundant Code (Copy-Paste)

Usage name
Element redundancy

Reference
https://cwe.mitre.org/data/definitions/1041

Roles
- the <Operation1>
- the <Operation2>
- the <SimilarCodeElements>

Detection Patterns
8.2.116 ASCQM Limit Volume of Similar Code

7.1.19 CWE-1055 Multiple Inheritance from Concrete Classes

Usage name
Excessive inheritance from concrete classes

Reference
https://cwe.mitre.org/data/definitions/1055

Roles
- the <ClassInheritanceDeclaration>
- the <ConcreteClasses>

Detection Patterns
8.2.126 ASCQM Ban Excessive Number of Concrete Implementations to Inherit From

7.1.20 CWE-1045 Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor

Usage name
Child class missing virtual destructor

Reference
https://cwe.mitre.org/data/definitions/1045

Roles
- the <ParentClass>
- the <ParentClassVirtualDestructor>
- the <ChildClass>

Detection Patterns
8.2.35 ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor

7.1.21 CWE-1052 Excessive Use of Hard-Coded Literals in Initialization

Usage name
Hard-coded literals

Reference
https://cwe.mitre.org/data/definitions/1052

Roles
- the <Initialization>
- the <HardCodedValue>

Detection Patterns
8.2.129 ASCQM Ban Hard-Coded Literals used to Initialize Variables

7.1.22 CWE-1048 Invokable Control Element with Large Number of Outward Calls (Excessive Coupling or Fan-out)

Usage name
Excessive references

Reference
https://cwe.mitre.org/data/definitions/1048

Roles
- the <Operation>
- the <OutwardCalls>

Detection Patterns
8.2.127 ASCQM Limit Number of Outward Calls

7.1.23 CWE-1095 Loop Condition Value Update within the Loop

Usage name
Condition value update within loop

Reference
https://cwe.mitre.org/data/definitions/1095

Roles
- the <LoopCondition>
- the <LoopValueUpdate>

Detection Patterns
8.2.123 ASCQM Ban Loop Value Update within Incremental and Decremental Loop

7.1.24 CWE-1085 Invokable Control Element with Excessive Volume of Commented-out Code

Usage name
Excessive commented-out code

Reference
https://cwe.mitre.org/data/definitions/1085

Roles
- the <CommentedOutCode>

Detection Patterns
8.2.114 ASCQM Limit Volume of Commented-Out Code

7.1.25 CWE-1047 Modules with Circular Dependencies

Usage name
Circular dependencies

Reference
https://cwe.mitre.org/data/definitions/1047

Roles
- the <ModuleDependencyCycles>

Detection Patterns
8.2.113 ASCQM Ban Circular Dependencies between Modules

7.1.26 CWE-1080 Source Code File with Excessive Number of Lines of Code

Usage name
Excessively large file

Reference
https://cwe.mitre.org/data/definitions/1080

Roles
- the <Operation>
- the <SourceCode>

Detection Patterns
8.2.115 ASCQM Limit Size of Operations Code

7.1.27 CWE-1062 Parent Class Element with References to Child Class

Usage name
Parent class referencing child class

Reference
https://cwe.mitre.org/data/definitions/1062

Roles
- the <ParentClass>
- the <ChildClass>
- the <Reference>

Detection Patterns
8.2.112 ASCQM Ban Conversion References to Child Class

7.1.28 CWE-1087 Class with Virtual Method without a Virtual Destructor

Usage name
Class with virtual method missing destructor

Reference
https://cwe.mitre.org/data/definitions/1087

Roles
- the <Class>
- the <VirtualMethod>

Detection Patterns
8.2.38 ASCQM Implement Virtual Destructor for Classes with Virtual Methods

7.1.29 CWE-1079 Parent Class without Virtual Destructor Method

Usage name
Parent class missing virtual destructor

Reference
https://cwe.mitre.org/data/definitions/1079

Roles
- the <ParentClass>

Detection Patterns
8.2.36 ASCQM Implement Virtual Destructor for Parent Classes

7.1.30 Maintainability Detection Patterns

Detection Patterns
8.2.24 ASCQM Ban Assignment Operation Inside Logic Blocks
8.2.113 ASCQM Ban Circular Dependencies between Modules
8.2.25 ASCQM Ban Comparison Expression Outside Logic Blocks
8.2.122 ASCQM Ban Control Flow Transfer
8.2.112 ASCQM Ban Conversion References to Child Class
8.2.131 ASCQM Ban Exception Definition without Ever Throwing It
8.2.119 ASCQM Ban Excessive Number of Children
8.2.126 ASCQM Ban Excessive Number of Concrete Implementations to Inherit From
8.2.120 ASCQM Ban Excessive Number of Inheritance Levels
8.2.43 ASCQM Ban Hard-Coded Literals used to Connect to Resource
8.2.129 ASCQM Ban Hard-Coded Literals used to Initialize Variables
8.2.82 ASCQM Ban Incorrect Joint Comparison
8.2.23 ASCQM Ban Incorrect Object Comparison
8.2.26 ASCQM Ban Incorrect String Comparison
8.2.130 ASCQM Ban Logical Dead Code
8.2.27 ASCQM Ban Logical Operation with a Constant Operand
8.2.123 ASCQM Ban Loop Value Update within Incremental and Decremental Loop
8.2.81 ASCQM Ban Not Operator On Non-Boolean Operand Of Comparison Operation
8.2.80 ASCQM Ban Not Operator On Operand Of Bitwise Operation
8.2.132 ASCQM Ban Switch in Switch Statement
8.2.44 ASCQM Ban Unintended Paths
8.2.125 ASCQM Ban Unreferenced Dead Code
8.2.121 ASCQM Ban Usage of Data Elements from Other Classes
8.2.79 ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use
8.2.35 ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor
8.2.38 ASCQM Implement Virtual Destructor for Classes with Virtual Methods
8.2.36 ASCQM Implement Virtual Destructor for Parent Classes
8.2.117 ASCQM Limit Algorithmic Complexity via Cyclomatic Complexity Value
8.2.134 ASCQM Limit Algorithmic Complexity via Essential Complexity Value
8.2.133 ASCQM Limit Algorithmic Complexity via Module Design Complexity Value
8.2.118 ASCQM Limit Number of Data Access
8.2.127 ASCQM Limit Number of Outward Calls
8.2.124 ASCQM Limit Number of Parameters
8.2.115 ASCQM Limit Size of Operations Code

8.2.114 ASCQM Limit Volume of Commented-Out Code
8.2.116 ASCQM Limit Volume of Similar Code
8.2.64 ASCQM Use Break in Switch Statement
8.2.135 ASCQM Use Default Case in Switch Statement

7.2 Weakness Category Performance Efficiency

7.2.1 CWE-401 Improper Release of Memory Before Removing Last Reference ('Memory Leak')

Reference
https://cwe.mitre.org/data/definitions/401

Roles
- the <MemoryAllocation>

Parent weaknesses
CWE-404 Improper Resource Shutdown or Release

Detection Patterns
8.2.29 ASCQM Ban Comma Operator from Delete Statement
8.2.32 ASCQM Implement Required Operations for Manual Resource Management
8.2.34 ASCQM Release Memory After Use
8.2.31 ASCQM Release Memory after Use with Correct Operation
8.2.33 ASCQM Release Platform Resource after Use
8.2.30 ASCQM Release in Destructor Memory Allocated in Constructor

7.2.2 Weakness CWE-404 Improper Resource Shutdown or Release

Reference
https://cwe.mitre.org/data/definitions/404

Roles
- the <ResourceAllocation>

Contributing weaknesses
CWE-401 Improper Release of Memory Before Removing Last Reference ('Memory Leak')
CWE-772 Missing Release of Resource after Effective Lifetime
CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime

Detection Patterns
8.2.29 ASCQM Ban Comma Operator from Delete Statement
8.2.35 ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor
8.2.38 ASCQM Implement Virtual Destructor for Classes with Virtual Methods
8.2.36 ASCQM Implement Virtual Destructor for Parent Classes
8.2.37 ASCQM Release File Resource after Use in Operation
8.2.33 ASCQM Release Platform Resource after Use
8.2.30 ASCQM Release in Destructor Memory Allocated in Constructor

7.2.3 CWE-424 Improper Protection of Alternate Path

Reference
https://cwe.mitre.org/data/definitions/424

Roles
- the <AlternatePath>

Detection Patterns
8.2.44 ASCQM Ban Unintended Paths

7.2.4 CWE-772 Missing Release of Resource after Effective Lifetime

Reference
https://cwe.mitre.org/data/definitions/772

Roles
- the <ResourceAllocation>

Parent weaknesses
CWE-404 Improper Resource Shutdown or Release

Detection Patterns
8.2.37 ASCQM Release File Resource after Use in Operation
8.2.33 ASCQM Release Platform Resource after Use
8.2.30 ASCQM Release in Destructor Memory Allocated in Constructor

7.2.5 CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime

Reference
https://cwe.mitre.org/data/definitions/775

Roles
- the <FileDescriptorOrHandleAllocation>

Parent weaknesses
Weakness CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime

Detection Patterns
8.2.63 ASCQM Release File Resource after Use in Class
8.2.37 ASCQM Release File Resource after Use in Operation

7.2.6 CWE-1073 Non-SQL Invokable Control Element with Excessive Number of Data Resource Access

Usage name
Excessive data queries in client-side code

Reference
https://cwe.mitre.org/data/definitions/1073

Roles
- the <NonSQLOperation>
- the <DataAccesses>

Detection Patterns
8.2.109 ASCQM Ban Excessive Number of Data Resource Access from non-SQL Code

7.2.7 CWE-1057 Data Access Operations Outside of Designated Data Manager Component

Usage name
Circumventing data access routines

Reference
https://cwe.mitre.org/data/definitions/1057

Roles
- the <DataManager>
- the <DataAccess>

Detection Patterns
8.2.44 ASCQM Ban Unintended Paths

7.2.8 CWE-1043 Storable and Member Data Element Excessive Number of Aggregated Storable and Member Data Elements

Usage name
Excessively large data element

Reference
https://cwe.mitre.org/data/definitions/1043

Roles
- the <AggregationData>
- the <AggregatedData>

Detection Patterns
8.2.107 ASCQM Limit Number of Aggregated Non-Primitive Data Types

7.2.9 CWE-1072 Data Resource Access without use of Connection Pooling

Usage name
Data access not using connection pool

Reference
https://cwe.mitre.org/data/definitions/1072

Roles
- the <Connection>

Detection Patterns
8.2.101 ASCQM Ban Use of Prohibited Low-Level Resource Management Functionality

7.2.10 CWE-1060 Excessive Number of Inefficient Server-Side Data Accesses

Usage name
Excessive data queries in non-stored procedure

Reference
https://cwe.mitre.org/data/definitions/1060

Roles
- the <NonStoredSQLOperation>
- the <DataAccesses>

Detection Patterns
8.2.108 ASCQM Ban Excessive Number of Data Resource Access from non-stored SQL Procedure

7.2.11 CWE-1091 Use of Object without Invoking Destructor Method

Reference
https://cwe.mitre.org/data/definitions/1091

Roles
- the <Object>

Detection Patterns
8.2.31 ASCQM Release Memory after Use with Correct Operation

7.2.12 CWE-1046 Creation of Immutable Text Using String Concatenation

Usage name
Immutable text data

Reference
https://cwe.mitre.org/data/definitions/1046

Roles
- the <ImmutableDataCreation>

Detection Patterns
8.2.110 ASCQM Ban Incremental Creation of Immutable Data

7.2.13 CWE-1042 Static Member Data Element outside of a Singleton Class Element

Usage name
Static data outside of singleton class

Reference
https://cwe.mitre.org/data/definitions/1042

Roles
- the <StaticDataDeclaration>

Detection Patterns
8.2.111 ASCQM Ban Static Non-Final Data Element Outside Singleton

7.2.14 CWE-1049 Excessive Data Query Operations in a Large Data Table

Usage name
Complex read/write access

Reference
https://cwe.mitre.org/data/definitions/1049

Roles
- the <DataQuery>

Detection Patterns
8.2.105 ASCQM Ban Excessive Complexity of Data Resource Access

7.2.15 CWE-1067 Excessive Execution of Sequential Searches of Data Resource

Usage name
Incorrect indices

Reference
https://cwe.mitre.org/data/definitions/1067

Roles
- the <DataQuery>
- the <TableOrView>

Detection Patterns
8.2.103 ASCQM Implement Index Required by Query on Large Tables

7.2.16 CWE-1089 Large Data Table with Excessive Number of Indices

Usage name
Excessive number of indices on large tables

Reference
https://cwe.mitre.org/data/definitions/1089

Roles
- the <Table>
- the <Indexes>

Detection Patterns
8.2.104 ASCQM Ban Excessive Number of Index on Columns of Large Tables

7.2.17 CWE-1094 Excessive Index Range Scan for a Data Resource

Usage name
Excessively large indices on large tables

Reference
https://cwe.mitre.org/data/definitions/1094

Roles
- the <Table>
- the <Indexes>

Detection Patterns
8.2.102 ASCQM Ban Excessive Size of Index on Columns of Large Tables

7.2.18 CWE-1050 Excessive Platform Resource Consumption within a Loop

Usage name
Resource consuming operation in loop

Reference
https://cwe.mitre.org/data/definitions/1050

Roles
- the <Loop>
- the <ExpensiveOperation>

Detection Patterns
8.2.106 ASCQM Ban Expensive Operations in Loops

7.2.19 CWE-1060 Excessive Number of Inefficient Server-Side Data Accesses

Usage name
Excessive data queries in non-stored procedure

Reference
https://cwe.mitre.org/data/definitions/1060

Roles
- the <NonStoredSQLOperation>
- the <DataAccesses>

Detection Patterns
8.2.108 ASCQM Ban Excessive Number of Data Resource Access from non-stored SQL Procedure

7.2.20 Performance Efficiency Detection Patterns

Detection Patterns
8.2.29 ASCQM Ban Comma Operator from Delete Statement
8.2.105 ASCQM Ban Excessive Complexity of Data Resource Access
8.2.109 ASCQM Ban Excessive Number of Data Resource Access from non-SQL Code
8.2.108 ASCQM Ban Excessive Number of Data Resource Access from non-stored SQL Procedure
8.2.104 ASCQM Ban Excessive Number of Index on Columns of Large Tables
8.2.102 ASCQM Ban Excessive Size of Index on Columns of Large Tables
8.2.106 ASCQM Ban Expensive Operations in Loops
8.2.110 ASCQM Ban Incremental Creation of Immutable Data
8.2.111 ASCQM Ban Static Non-Final Data Element Outside Singleton
8.2.44 ASCQM Ban Unintended Paths
8.2.101 ASCQM Ban Use of Prohibited Low-Level Resource Management Functionality
8.2.103 ASCQM Implement Index Required by Query on Large Tables
8.2.32 ASCQM Implement Required Operations for Manual Resource Management
8.2.35 ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor
8.2.38 ASCQM Implement Virtual Destructor for Classes with Virtual Methods
8.2.36 ASCQM Implement Virtual Destructor for Parent Classes
8.2.107 ASCQM Limit Number of Aggregated Non-Primitive Data Types
8.2.63 ASCQM Release File Resource after Use in Class
8.2.37 ASCQM Release File Resource after Use in Operation
8.2.34 ASCQM Release Memory After Use
8.2.31 ASCQM Release Memory after Use with Correct Operation
8.2.33 ASCQM Release Platform Resource after Use
8.2.30 ASCQM Release in Destructor Memory Allocated in Constructor

7.3 Weakness Category Reliability

7.3.1 CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Reference
https://cwe.mitre.org/data/definitions/119

Roles
- the <BufferOperation>

Contributing weaknesses
CWE-120 Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
CWE-123 Write-what-where Condition
CWE-125 Out-of-bounds Read
CWE-130 Improper Handling of Length Parameter Inconsistency
CWE-786 Access of Memory Location Before Start of Buffer
CWE-787 Out-of-bounds Write
CWE-788 Access of Memory Location After End of Buffer
CWE-805 Buffer Access with Incorrect Length Value
CWE-822 Untrusted Pointer Dereference
CWE-823 Use of Out-of-range Pointer Offset
CWE-824 Access of Uninitialized Pointer
CWE-825 Expired Pointer Dereference

Detection Patterns
8.2.6 ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities
8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities
8.2.5 ASCQM Ban Use of Expired Pointer
8.2.1 ASCQM Check Index of Array Access
8.2.2 ASCQM Check Input of Memory Manipulation Primitives
8.2.4 ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities
8.2.7 ASCQM Check Offset used in Pointer Arithmetic
8.2.9 ASCQM Initialize Pointers before Use
8.2.8 ASCQM Sanitize User Input used as Pointer

7.3.2 CWE-120 Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')

Reference
https://cwe.mitre.org/data/definitions/120

Roles
- the <BufferCopy>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.6 ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities

8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities

7.3.3 CWE-123 Write-what-where Condition

Reference
https://cwe.mitre.org/data/definitions/123

Roles
- the <BufferWrite>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities

7.3.4 CWE-125 Out-of-bounds Read

Reference
https://cwe.mitre.org/data/definitions/125

Roles
- the <BufferRead>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.1 ASCQM Check Index of Array Access

7.3.5 CWE-130 Improper Handling of Length Parameter Inconsistency

Reference
https://cwe.mitre.org/data/definitions/130

Roles
- the <DataHandling>
- the <LengthParameter>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.1 ASCQM Check Index of Array Access

7.3.6 CWE-131 Incorrect Calculation of Buffer Size

Reference
https://cwe.mitre.org/data/definitions/131

Roles
- the <BufferSizeCalculation>

Parent weaknesses
CWE-682 Incorrect Calculation

Detection Patterns
8.2.71 ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size
8.2.70 ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation
8.2.72 ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value

7.3.7 CWE-170 Improper Null Termination

Reference
https://cwe.mitre.org/data/definitions/170

Roles
- the <BufferWithoutNULLTermination>

Detection Patterns
8.2.62 ASCQM NULL Terminate Output Of String Manipulation Primitives

7.3.8 CWE-194 Unexpected Sign Extension

Reference
https://cwe.mitre.org/data/definitions/194

Roles
- the <NumberSignExtension>

Parent weaknesses
CWE-681 Incorrect Conversion between Numeric Types

Detection Patterns
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion

7.3.9 CWE-195 Signed to Unsigned Conversion Error

Reference
https://cwe.mitre.org/data/definitions/195

Roles
- the <NumberConversionToUnsigned>

Parent weaknesses
CWE-681 Incorrect Conversion between Numeric Types

Detection Patterns
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion

7.3.10 CWE-196 Unsigned to Signed Conversion Error

Reference
https://cwe.mitre.org/data/definitions/196

Roles
- the <NumberConversionToSigned>

Parent weaknesses
CWE-681 Incorrect Conversion between Numeric Types

Detection Patterns
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion

7.3.11 CWE-197 Numeric Truncation Error

Reference
https://cwe.mitre.org/data/definitions/197

Roles
- the <NumberTruncation>

Parent weaknesses
CWE-681 Incorrect Conversion between Numeric Types

Detection Patterns
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion

7.3.12 CWE-248 Uncaught Exception

Reference
https://cwe.mitre.org/data/definitions/248

Roles
- the <ExceptionThrowDeclaration>
- the <ExceptionCatchSequence>

Parent weaknesses
CWE-703 Improper Check or Handling of Exceptional Conditions

Detection Patterns
8.2.65 ASCQM Catch Exceptions

7.3.13 CWE-252 Unchecked Return Value

Reference
https://cwe.mitre.org/data/definitions/252

Roles
- the <OperationCall>

Detection Patterns
8.2.21 ASCQM Check Return Value of Resource Operations Immediately
8.2.20 ASCQM Handle Return Value of Must Check Operations

7.3.14 CWE-366 Race Condition within a Thread

Reference
https://cwe.mitre.org/data/definitions/366

Roles
- the <Thread1>
- the <Thread2>
- the <ConflictingResource>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.57 ASCQM Ban Creation of Lock On Private Non-Static Object to Access Private Static Data
8.2.48 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Contex

7.3.15 CWE-369 Divide by Zero

Reference
https://cwe.mitre.org/data/definitions/369

Roles
- the <Division>

Parent weaknesses
CWE-682 Incorrect Calculation

Detection Patterns
8.2.56 ASCQM Check and Handle ZERO Value before Use as Divisor

7.3.16 CWE-390 Detection of Error Condition Without Action

Reference
https://cwe.mitre.org/data/definitions/390

Roles
- the <ErrorCondition>

Detection Patterns
8.2.66 ASCQM Ban Empty Exception Block
8.2.18 ASCQM Handle Return Value of Resource Operations

7.3.17 CWE-391 Unchecked Error Condition

Reference
https://cwe.mitre.org/data/definitions/391

Roles
- the <ErrorConditionProcessing>

Parent weaknesses
CWE-703 Improper Check or Handling of Exceptional Conditions

Detection Patterns
8.2.66 ASCQM Ban Empty Exception Block
8.2.22 ASCQM Ban Useless Handling of Exceptions

7.3.18 CWE-392 Missing Report of Error Condition

Reference
https://cwe.mitre.org/data/definitions/392

Roles
- the <ErrorConditionProcessing>

Parent weaknesses
CWE-703 Improper Check or Handling of Exceptional Conditions

Detection Patterns
8.2.22 ASCQM Ban Useless Handling of Exceptions

7.3.19 CWE-394 Unexpected Status Code or Return Value

Reference
https://cwe.mitre.org/data/definitions/394

Roles
- the <ReturnValue>

Detection Patterns
8.2.19 ASCQM Ban Incorrect Numeric Conversion of Return Value
8.2.20 ASCQM Handle Return Value of Must Check Operations
8.2.18 ASCQM Handle Return Value of Resource Operations

7.3.20 CWE-401 Improper Release of Memory Before Removing Last Reference ('Memory Leak')

Reference
https://cwe.mitre.org/data/definitions/401

Roles
- the <MemoryAllocation>

Parent weaknesses
CWE-404 Improper Resource Shutdown or Release

Detection Patterns
8.2.29 ASCQM Ban Comma Operator from Delete Statement
8.2.32 ASCQM Implement Required Operations for Manual Resource Management
8.2.34 ASCQM Release Memory After Use
8.2.31 ASCQM Release Memory after Use with Correct Operation
8.2.33 ASCQM Release Platform Resource after Use
8.2.30 ASCQM Release in Destructor Memory Allocated in Constructor

7.3.21 CWE-404 Improper Resource Shutdown or Release

Reference
https://cwe.mitre.org/data/definitions/404

Roles
- the <ResourceAllocation>

Contributing weaknesses
CWE-401 Improper Release of Memory Before Removing Last Reference ('Memory Leak')
CWE-772 Missing Release of Resource after Effective Lifetime
CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime

Detection Patterns
8.2.29 ASCQM Ban Comma Operator from Delete Statement
8.2.35 ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor
8.2.38 ASCQM Implement Virtual Destructor for Classes with Virtual Methods
8.2.36 ASCQM Implement Virtual Destructor for Parent Classes
8.2.37 ASCQM Release File Resource after Use in Operation
8.2.33 ASCQM Release Platform Resource after Use
8.2.30 ASCQM Release in Destructor Memory Allocated in Constructor

7.3.22 CWE-415 Double Free

Reference
https://cwe.mitre.org/data/definitions/415

Roles
- the <ResourceRelease >
- the <ResourceAccess>

- the <ResourceUse>

Parent weaknesses
CWE-672 Operation on a Resource after Expiration or Release

Detection Patterns
8.2.76 ASCQM Ban Double Free On Pointers

7.3.23 CWE-416 Use After Free

Reference
https://cwe.mitre.org/data/definitions/416

Roles
- the <ResourceRelease>
- the <ResourceUse>

Parent weaknesses
CWE-672 Operation on a Resource after Expiration or Release

Detection Patterns
8.2.14 ASCQM Ban Free Operation on Pointer Received as Parameter
8.2.5 ASCQM Ban Use of Expired Pointer
8.2.13 ASCQM Implement Copy Constructor for Class with Pointer Resource

7.3.24 CWE-424 Improper Protection of Alternate Path

Reference
https://cwe.mitre.org/data/definitions/424

Roles
- the <AlternatePath>

Detection Patterns
8.2.44 ASCQM Ban Unintended Paths

7.3.25 CWE-456 Missing Initialization of a Variable

Reference
https://cwe.mitre.org/data/definitions/456

Roles
- the <VariableDeclaration>

Parent weaknesses
CWE-665 Improper Initialization

Detection Patterns
8.2.75 ASCQM Ban Allocation of Memory with Null Size

8.2.74 ASCQM Initialize Variables

7.3.26 CWE-459 Incomplete Cleanup

Reference
https://cwe.mitre.org/data/definitions/459

Roles
- the <ResourceAllocation>
- the <ResourceRelease>

Detection Patterns
8.2.31 ASCQM Release Memory after Use with Correct Operation

7.3.27 CWE-476 NULL Pointer Dereference

Reference
https://cwe.mitre.org/data/definitions/476

Roles
- the <PointerDereferencing>

Detection Patterns
8.2.10 ASCQM Check NULL Pointer Value before Use

7.3.28 CWE-480 Use of Incorrect Operator

Reference
https://cwe.mitre.org/data/definitions/480

Roles
- the <Operator>

Detection Patterns
8.2.24 ASCQM Ban Assignment Operation Inside Logic Blocks
8.2.25 ASCQM Ban Comparison Expression Outside Logic Blocks
8.2.23 ASCQM Ban Incorrect Object Comparison
8.2.26 ASCQM Ban Incorrect String Comparison
8.2.27 ASCQM Ban Logical Operation with a Constant Operand

7.3.29 CWE-484 Omitted Break Statement in Switch

Reference
https://cwe.mitre.org/data/definitions/484

Roles
- the <SwitchStatement>

Detection Patterns
8.2.64 ASCQM Use Break in Switch Statement

7.3.30 CWE-543 Use of Singleton Pattern Without Synchronization in a Multithreaded Context

Reference
https://cwe.mitre.org/data/definitions/543

Roles
- the <SingletonUse>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.41 ASCQM Ban Non-Final Static Data in Multi-Threaded Context
8.2.46 ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context

7.3.31 CWE-562 Return of Stack Variable Address

Reference
https://cwe.mitre.org/data/definitions/562

Roles
- the <ReturnStatement>

Detection Patterns
8.2.52 ASCQM Ban Return of Local Variable Address
8.2.53 ASCQM Ban Storage of Local Variable Address in Global Variable

7.3.32 CWE-567 Unsynchronized Access to Shared Data in a Multithreaded Context

Reference
https://cwe.mitre.org/data/definitions/567

Roles
- the <SharedDataAccess>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.41 ASCQM Ban Non-Final Static Data in Multi-Threaded Context
8.2.48 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context

7.3.33 CWE-595 Comparison of Object References Instead of Object Contents

Reference
https://cwe.mitre.org/data/definitions/595

Roles
- the <ObjectReferencesComparison>

Contributing weaknesses
CWE-597 Use of Wrong Operator in String Comparison
CWE-1097 Persistent Storable Data Element without Associated Comparison Control Element

Detection Patterns
8.2.23 ASCQM Ban Incorrect Object Comparison
8.2.26 ASCQM Ban Incorrect String Comparison
8.2.28 ASCQM Implement Correct Object Comparison Operations

7.3.34 CWE-597 Use of Wrong Operator in String Comparison

Reference
https://cwe.mitre.org/data/definitions/597

Roles
- the <StringComparison>

Parent weaknesses
CWE-595 Comparison of Object References Instead of Object Contents

Detection Patterns
8.2.26 ASCQM Ban Incorrect String Comparison

7.3.35 CWE-662 Improper Synchronization

Reference
https://cwe.mitre.org/data/definitions/662

Roles
- the <Thread1>
- the <Thread2>
- the <SharedResourceAccess>

Contributing weaknesses
CWE-366 Race Condition within a Thread
CWE-543 Use of Singleton Pattern Without Synchronization in a Multithreaded Context
CWE-567 Unsynchronized Access to Shared Data in a Multithreaded Context
CWE-667 Improper Locking
CWE-764 Multiple Locks of a Critical Resource
CWE-820 Missing Synchronization
CWE-821 Incorrect Synchronization
CWE-833 Deadlock
CWE-1058 Invokable Control Element in Multi-Thread Context with non-Final Static Storable or Member Element
CWE-1096Singleton Class Instance Creation without Proper Locking or Synchronization

Detection Patterns
8.2.61 ASCQM Ban Creation of Lock On Inappropriate Object Type
8.2.60 ASCQM Ban Creation of Lock On Non-Final Object
8.2.57 ASCQM Ban Creation of Lock On Private Non-Static Object to Access Private Static Data
8.2.68 ASCQM Ban Incompatible Lock Acquisition Sequences
8.2.49 ASCQM Ban Incorrect Synchronization Mechanisms
8.2.41 ASCQM Ban Non-Final Static Data in Multi-Threaded Context
8.2.50 ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context
8.2.73 ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock
8.2.59 ASCQM Ban Sleep Between Lock Acquisition and Release
8.2.69 ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues
8.2.48 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context
8.2.58 ASCQM Release Lock After Use
8.2.46 ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context

7.3.36 CWE-667 Improper Locking

Reference
Reference https://cwe.mitre.org/data/definitions/667

Roles
- the <Thread1>
- the <Thread2>
- the <SharedResourceAccess>
- the <Lock>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.49 ASCQM Ban Incorrect Synchronization Mechanisms
8.2.41 ASCQM Ban Non-Final Static Data in Multi-Threaded Context
8.2.50 ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context
8.2.48 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context
8.2.46 ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context

7.3.37 CWE-672 Operation on a Resource after Expiration or Release

Reference
https://cwe.mitre.org/data/definitions/672

Roles
- the <ResourceRelease>
- the <ResourceAccess>

Contributing weaknesses
CWE-415 Double Free
CWE-416 Use After Free

Detection Patterns
8.2.12 ASCQM Ban Double Release of Resource
8.2.11 ASCQM Ban Use of Expired Resource

7.3.38 CWE-681 Incorrect Conversion between Numeric Types

Reference
https://cwe.mitre.org/data/definitions/681

Roles
- the <NumericConversion>

Contributing weaknesses
CWE-194 Unexpected Sign Extension
CWE-195 Signed to Unsigned Conversion Error
CWE-196 Unsigned to Signed Conversion Error
CWE-197 Numeric Truncation Error

Detection Patterns
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion

7.3.39 CWE-682 Incorrect Calculation

Reference
https://cwe.mitre.org/data/definitions/682

Roles
- the <Calculation>

Contributing weaknesses
CWE-131 Incorrect Calculation of Buffer Size
CWE-369 Divide by Zero

Detection Patterns
8.2.71 ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size
8.2.70 ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation
8.2.72 ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value
8.2.56 ASCQM Check and Handle ZERO Value before Use as Divisor

7.3.40 CWE-703 Improper Check or Handling of Exceptional Conditions

Reference
https://cwe.mitre.org/data/definitions/703

Roles
- the <ErrorHandling>

Contributing weaknesses
CWE-166 Improper Handling of Missing Special Element
CWE-167 Improper Handling of Additional Special Element
CWE-168 Improper Handling of Inconsistent Special Elements
CWE-228 Improper Handling of Syntactically Invalid Structure
CWE-248 Uncaught Exception
CWE-280 Improper Handling of Insufficient Permissions or Privileges
CWE-391 Unchecked Error Condition
CWE-392 Missing Report of Error Condition
CWE-393 Return of Wrong Status Code
CWE-754 Improper Check for Unusual or Exceptional Conditions
CWE-755 Improper Handling of Exceptional Conditions

Detection Patterns
8.2.22 ASCQM Ban Useless Handling of Exceptions

7.3.41 CWE-704 Incorrect Type Conversion or Cast

Reference
https://cwe.mitre.org/data/definitions/704

Roles
- the <TypeConversion>

Contributing weaknesses
CWE-843 Access of Resource Using Incompatible Type ('Type Confusion')

Detection Patterns
8.2.51 ASCQM Ban Incorrect Type Conversion

7.3.42 CWE-758 Reliance on Undefined, Unspecified, or Implementation-Defined Behavior

Reference
https://cwe.mitre.org/data/definitions/758

Roles
- the <Statement>

Detection Patterns
8.2.15 ASCQM Ban Delete of VOID Pointer
8.2.17 ASCQM Ban Reading and Writing the Same Variable Used as Assignment Value
8.2.16 ASCQM Ban Variable Increment or Decrement Operation in Operations using the Same Variable

7.3.43 CWE-764 Multiple Locks of a Critical Resource

Reference
https://cwe.mitre.org/data/definitions/764

Roles
- the <Lock1>
- the <Lock2>
- the <Resource>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.73 ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock

7.3.44 CWE-772 Missing Release of Resource after Effective Lifetime

Reference
https://cwe.mitre.org/data/definitions/772

Roles
- the <ResourceAllocation>

Parent weaknesses
CWE-404 Improper Resource Shutdown or Release

Detection Patterns
8.2.37 ASCQM Release File Resource after Use in Operation
8.2.33 ASCQM Release Platform Resource after Use
8.2.30 ASCQM Release in Destructor Memory Allocated in Constructor

7.3.45 CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime

Reference
https://cwe.mitre.org/data/definitions/775

Roles
- the <FileDescriptorOrHandleAllocation>

Parent weaknesses
CWE-404 Improper Resource Shutdown or Release

Detection Patterns
8.2.63 ASCQM Release File Resource after Use in Class
8.2.37 ASCQM Release File Resource after Use in Operation

7.3.46 CWE-786 Access of Memory Location Before Start of Buffer

Reference
https://cwe.mitre.org/data/definitions/786

Roles
- the <MemoryAccess>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.1 ASCQM Check Index of Array Access
8.2.4 ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities

7.3.47 CWE-787 Out-of-bounds Write

Reference
https://cwe.mitre.org/data/definitions/787

Roles
- the <BufferWrite>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.1 ASCQM Check Index of Array Access
8.2.2 ASCQM Check Input of Memory Manipulation Primitives

7.3.48 CWE-788 Access of Memory Location After End of Buffer

Reference
https://cwe.mitre.org/data/definitions/788

Roles
- the <MemoryAccess>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities
8.2.1 ASCQM Check Index of Array Access
8.2.2 ASCQM Check Input of Memory Manipulation Primitives

7.3.49 CWE-805 Buffer Access with Incorrect Length Value

Reference
https://cwe.mitre.org/data/definitions/805

Roles
- the <BufferAccess>
- the <LengthParameter>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities
8.2.2 ASCQM Check Input of Memory Manipulation Primitives
8.2.4 ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities

7.3.50 CWE-820 Missing Synchronization

Reference
https://cwe.mitre.org/data/definitions/820

Roles
- the <SharedResourceUse>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.50 ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context

7.3.51 CWE-821 Incorrect Synchronization

Reference
https://cwe.mitre.org/data/definitions/821

Roles
- the <SharedResourceUse>
- the <IncorrectSynchronization>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.49 ASCQM Ban Incorrect Synchronization Mechanisms

7.3.52 7.3.52 CWE-822 Untrusted Pointer Dereference

Reference
https://cwe.mitre.org/data/definitions/822

Roles
- the <PointerDereferencing>

- the <TaintedInput>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8 .2.8 ASCQM Sanitize User Input used as Pointer

7.3.53 7.3.53 CWE-823 Use of Out-of-range Pointer Offset

Reference
https://cwe.mitre.org/data/definitions/823

Roles
- the <PointerOffset>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.7 ASCQM Check Offset used in Pointer Arithmetic

7.3.54 CWE-824 Access of Uninitialized Pointer

Reference
Reference https://cwe.mitre.org/data/definitions/824

Roles
Roles:
- the <PointerAccess>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.9 ASCQM Initialize Pointers before Use

7.3.55 CWE-825 Expired Pointer Dereference

Reference
https://cwe.mitre.org/data/definitions/825

Roles
- the <PointerAccess>
- the <PointerRelease>

Parent weaknesses
CWE-672 Operation on a Resource after Expiration or Release

Detection Patterns
8.2.5 ASCQM Ban Use of Expired Pointer

7.3.56 CWE-833 Deadlock

Reference
https://cwe.mitre.org/data/definitions/833

Roles
- the <Thread1>
- the <Thread2>
- the <ConflictingLock>

Parent weaknesses
Weakness CWE-662 Improper Synchronization

Detection Patterns
8.2.68 ASCQM Ban Incompatible Lock Acquisition Sequences
8.2.69 ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues

7.3.57 CWE-835 Loop with Unreachable Exit Condition ('Infinite Loop')

Reference
https://cwe.mitre.org/data/definitions/835

Roles
- the <InfiniteLoop>

Detection Patterns
8.2.55 ASCQM Ban Unmodified Loop Variable Within Loop
8.2.54 ASCQM Ban While TRUE Loop Without Path To Break

7.3.58 CWE-908 Use of Uninitialized Resource

Reference
https://cwe.mitre.org/data/definitions/908

Roles
- the <ResourceUse>

Detection Patterns
8.2.67 ASCQM Initialize Resource before Use

7.3.59 CWE-1083 Data Access from Outside Designated Data Manager Component

Usage name
Circumventing data access routines

Reference
https://cwe.mitre.org/data/definitions/1083

Roles
- the <DataManager>
- the <DataAccess>

Detection Patterns
8.2.44 ASCQM Ban Unintended Paths

7.3.60 CWE-1058 Invokable Control Element in Multi-Thread Context with non-Final Static Storable or Member Element

Usage name
Non-final static data in a multi-threaded environment

Reference
https://cwe.mitre.org/data/definitions/1058

Roles
- the <Operation>
- the <NonFinalStaticData>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.41 ASCQM Ban Non-Final Static Data in Multi-Threaded Context

7.3.61 CWE-1096 Singleton Class Instance Creation without Proper Locking or Synchronization

Usage name
Improper locking of singleton classes

Reference
https://cwe.mitre.org/data/definitions/1096

Roles
- the <SingletonUse>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.46 ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context

7.3.62 CWE-1087 Class with Virtual Method without a Virtual Destructor

Usage name
Class with virtual method missing destructor

Reference
https://cwe.mitre.org/data/definitions/1087

Roles
- the <Class>
- the <VirtualMethod>

Detection Patterns
8.2.38 ASCQM Implement Virtual Destructor for Classes with Virtual Methods

7.3.63 CWE-1079 Parent Class without Virtual Destructor Method

Usage name
Parent class missing virtual destructor

Reference
https://cwe.mitre.org/data/definitions/1079

Roles
- the <ParentClass>

Detection Patterns
8.2.36 ASCQM Implement Virtual Destructor for Parent Classes

7.3.64 CWE-1045 Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor

Usage name
Child class missing virtual destructor

Reference
https://cwe.mitre.org/data/definitions/1045

Roles
- the <ParentClass>
- the <ParentClassVirtualDestructor>
- the <ChildClass>

Detection Patterns
8.2.35 ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor

7.3.65 CWE-1051 Initialization with Hard-Coded Network Resource Configuration Data

Usage name
Hard-coded network resource information

Reference
https://cwe.mitre.org/data/definitions/1051

Roles
- the <NetworkResourceAccess>
- the <HardCodedValue>

Detection Patterns
8.2.43 ASCQM Ban Hard-Coded Literals used to Connect to Resource

7.3.66 CWE-1088 Synchronous Access of Remote Resource without Timeout

Usage name
Synchronous call with missing timeout

Reference
https://cwe.mitre.org/data/definitions/1088

Roles
- the <SynchronousCall>
- the <TimeOutOption>

Detection Patterns
8.2.40 ASCQM Manage Time-Out Mechanisms in Blocking Synchronous Calls

7.3.67 CWE-1066 Missing Serialization Control Element

Reference
https://cwe.mitre.org/data/definitions/1066

Roles
- the <SerializableData>

Detection Patterns
8.2.42 ASCQM Ban Non-Serializable Elements in Serializable Objects

7.3.68 CWE-1070 Serializable Storable Data Element with non-Serializable Item Elements

Reference
https://cwe.mitre.org/data/definitions/1070

Roles
- the <SerializableData>
- the <NonSerialibleChildData>

Detection Patterns
8.2.42 ASCQM Ban Non-Serializable Elements in Serializable Objects

7.3.69 CWE-1097 Persistent Storable Data Element without Associated Comparison Control Element

Usage name
Persistent data without proper comparison controls

Reference
https://cwe.mitre.org/data/definitions/1097

Roles
- the <PersistentData>

Parent weaknesses
CWE-595 Comparison of Object References Instead of Object Contents

Detection Patterns
8.2.28 ASCQM Implement Correct Object Comparison Operations

7.3.70 CWE-1098 Data Element containing Pointer Item without Proper Copy Control Element

Usage name
Improper copy capabilities for data pointers

Reference
https://cwe.mitre.org/data/definitions/1098

Roles
- the <ParentData>
- the <PointerChildData>

Detection Patterns
8.2.13 ASCQM Implement Copy Constructor for Class With Pointer Resource

7.3.71 CWE-1082 Class Instance Self Destruction Control Element

Usage name
Self-destruction

Reference
https://cwe.mitre.org/data/definitions/1082

Roles
- the <SelfDestruction>

Detection Patterns
8.2.39 ASCQM Ban Self Destruction

7.3.72 CWE-1077 Floating Point Comparison with Incorrect Operator

Usage name
Improper equality comparisons of float-type numerical data

Reference
https://cwe.mitre.org/data/definitions/1077

Roles
- the <FloatNumberEqualityComparison>

Detection Patterns
8.2.45 ASCQM Ban Incorrect Float Number Comparison

7.3.73 CWE-665 Improper Initialization

Reference
https://cwe.mitre.org/data/definitions/665

Roles
- the <Initialization>

Contributing weaknesses
CWE-456 Missing Initialization of a Variable
CWE-457 Use of Uninitialized Variable

Detection Patterns
8.2.78 ASCQM Ban Self Assignment
8.2./9 ASCQM Initialize Pointers before Use
8.2.77 ASCQM Initialize Variables before Use

7.3.74 CWE-457 Use of Uninitialized Variable

Reference
https://cwe.mitre.org/data/definitions/457

Roles
- the <VariableDeclaration>
- the <VariableUse>

Parent weaknesses
CWE-665 Improper Initialization

Detection Patterns
8.2.75 ASCQM Ban Allocation of Memory with Null Size
8.2.74 ASCQM Initialize Variables

7.3.75 Reliability Detection Patterns

Detection Patterns
8.2.75 ASCQM Ban Allocation of Memory with Null Size
8.2.24 ASCQM Ban Assignment Operation Inside Logic Blocks
8.2.71 ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size
8.2.70 ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation
8.2.72 ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value
8.2.29 ASCQM Ban Comma Operator from Delete Statement
8.2.25 ASCQM Ban Comparison Expression Outside Logic Blocks
8.2.64 ASCQM Use Break in Switch Statement
8.2.62 ASCQM NULL Terminate Output Of String Manipulation Primitives
8.2.61 ASCQM Ban Creation of Lock on Inappropriate Object Type
8.2.60 ASCQM Ban Creation of Lock on Non-Final Object
8.2.57 ASCQM Ban Creation of Lock on Private Non-Static Object to Access Private Static Data
8.2.15 ASCQM Ban Delete of VOID Pointer
8.2.76 ASCQM Ban Double Free on Pointers
8.2.12 ASCQM Ban Double Release of Resource
8.2.66 ASCQM Ban Empty Exception Block
8.2.14 ASCQM Ban Free Operation on Pointer Received as Parameter
8.2.43 ASCQM Ban Hard-Coded Literals used to Connect to Resource
8.2.68 ASCQM Ban Incompatible Lock Acquisition Sequences
8.2.45 ASCQM Ban Incorrect Float Number Comparison
8.2.19 ASCQM Ban Incorrect Numeric Conversion of Return Value
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion
8.2.23 ASCQM Ban Incorrect Object Comparison

8.2.26 ASCQM Ban Incorrect String Comparison
8.2.49 ASCQM Ban Incorrect Synchronization Mechanisms
8.2.51 ASCQM Ban Incorrect Type Conversion
8.2.6 ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities
8.2.27 ASCQM Ban Logical Operation with a Constant Operand
8.2.41 ASCQM Ban Non-Final Static Data in Multi-Threaded Context
8.2.42 ASCQM Ban Non-Serializable Elements in Serializable Objects
8.2.17 ASCQM Ban Reading and Writing the Same Variable Used as Assignment Value
8.2.50 ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context
8.2.52 ASCQM Ban Return of Local Variable Address
8.2.39 ASCQM Ban Self Destruction
8.2.73 ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock
8.2.59 ASCQM Ban Sleep Between Lock Acquisition and Release
8.2.53 ASCQM Ban Storage of Local Variable Address in Global Variable
8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities
8.2.44 ASCQM Ban Unintended Paths
8.2.55 ASCQM Ban Unmodified Loop Variable Within Loop
8.2.5 ASCQM Ban Use of Expired Pointer
8.2.11 ASCQM Ban Use of Expired Resource
8.2.69 ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues
8.2.22 ASCQM Ban Useless Handling of Exceptions
8.2.16 ASCQM Ban Variable Increment or Decrement Operation in Operations using the Same Variable
8.2.54 ASCQM Ban While TRUE Loop Without Path to Break
8.2.65 ASCQM Catch Exceptions
8.2.67 ASCQM Initialize Resource before Use
8.2.40 ASCQM Manage Time-Out Mechanisms in Blocking Synchronous Calls
8.2.1 ASCQM Check Index of Array Access
8.2.2 ASCQM Check Input of Memory Manipulation Primitives
8.2.4 ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities
8.2.10 ASCQM Check NULL Pointer Value before Use
8.2.7 ASCQM Check Offset used in Pointer Arithmetic
8.2.21 ASCQM Check Return Value of Resource Operations Immediately
8.2.56 ASCQM Check and Handle ZERO Value before Use as Divisor
8.2.48 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context
8.2.20 ASCQM Handle Return Value of Must Check Operations
8.2.18 ASCQM Handle Return Value of Resource Operations
8.2.13 ASCQM Implement Copy Constructor for Class With Pointer Resource
8.2.28 ASCQM Implement Correct Object Comparison Operations

7.4 Weakness Category Security

7.4.1 Improper Restriction of Operations within the Bounds of a Memory Buffer

Reference
https://cwe.mitre.org/data/definitions/119

Roles
- the <BufferOperation>

Contributing weaknesses
CWE-120 Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
CWE-123 Write-what-where Condition
CWE-125 Out-of-bounds Read
CWE-130 Improper Handling of Length Parameter Inconsistency
CWE-786 Access of Memory Location Before Start of Buffer
CWE-787 Out-of-bounds Write
CWE-788 Access of Memory Location After End of Buffer
CWE-805 Buffer Access with Incorrect Length Value
CWE-822 Untrusted Pointer Dereference
CWE-823 Use of Out-of-range Pointer Offset
CWE-824 Access of Uninitialized Pointer
CWE-825 Expired Pointer Dereference

Detection Patterns
8.2.6 ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities
8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities
8.2.5 ASCQM Ban Use of Expired Pointer
8.2.1 ASCQM Check Index of Array Access
8.2.2 ASCQM Check Input of Memory Manipulation Primitives
8.2.4 ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities
8.2.7 ASCQM Check Offset used in Pointer Arithmetic
8.2.9 ASCQM Initialize Pointers before Use
8.2.8 ASCQM Sanitize User Input used as Pointer

7.4.2 CWE-120 Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')

Reference
https://cwe.mitre.org/data/definitions/120

Roles
- the <BufferCopy>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.6 ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities
8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities

7.4.3 CWE-123 Write-what-where Condition

Reference
https://cwe.mitre.org/data/definitions/123

Roles
- the <BufferWrite>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities

7.4.4 CWE-125 Out-of-bounds Read

Reference
https://cwe.mitre.org/data/definitions/125

Roles
- the <BufferRead>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.1 ASCQM Check Index of Array Access

7.4.5 CWE-129 Improper Validation of Array Index

Reference
https://cwe.mitre.org/data/definitions/129

Roles
- the <ArrayAccess>
- the <TaintedIndex>

Detection Patterns
8.2.94 ASCQM Sanitize User Input used as Array Index

7.4.6 CWE-130 Improper Handling of Length Parameter Inconsistency

Reference
https://cwe.mitre.org/data/definitions/130

Roles
- the <DataHandling>
- the <LengthParameter>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.1 ASCQM Check Index of Array Access

7.4.7 CWE-131 Incorrect Calculation of Buffer Size

Reference
Reference https://cwe.mitre.org/data/definitions/131

Roles
- the <BufferSizeCalculation>

Parent weaknesses
CWE-682 Incorrect Calculation

Detection Patterns
8.2.71 ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size
8.2.70 ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation
8.2.72 ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value

7.4.8 CWE-134 Use of Externally-Controlled Format String

Reference
https://cwe.mitre.org/data/definitions/134

Roles
- the <Formatting>
- the <TaintedFormatString>

Detection Patterns
8.2.96 ASCQM Sanitize User Input used as String Format

7.4.9 CWE-194 Unexpected Sign Extension

Reference
https://cwe.mitre.org/data/definitions/194

Roles
- the <NumberSignExtension>

Parent weaknesses
CWE-681 Incorrect Conversion between Numeric Types

Detection Patterns
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion

7.4.10 CWE-195 Signed to Unsigned Conversion Error

Reference
https://cwe.mitre.org/data/definitions/195

Roles
- the <NumberConversionToUnsigned>

Parent weaknesses
CWE-681 Incorrect Conversion between Numeric Types

Detection Patterns
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion

7.4.11 CWE-196 Unsigned to Signed Conversion Error

Reference
https://cwe.mitre.org/data/definitions/196

Roles
- the <NumberConversionToSigned>

Parent weaknesses
CWE-681 Incorrect Conversion between Numeric Types

Detection Patterns
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion

7.4.12 CWE-197 Numeric Truncation Error

Reference
https://cwe.mitre.org/data/definitions/197

Roles
- the <NumberTruncation>

Parent weaknesses
CWE-681 Incorrect Conversion between Numeric Types

Detection Patterns
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion

7.4.13 CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Reference
https://cwe.mitre.org/data/definitions/22

Roles
- the <PathManipulationStatement>
- the <TaintedInput>

Contributing weaknesses
CWE-23 Relative Path Traversal
CWE-36 Absolute Path Traversal

Detection Patterns
8.2.85 ASCQM Sanitize User Input used in Path Manipulation

7.4.14 CWE-23 Relative Path Traversal

Reference
https://cwe.mitre.org/data/definitions/23

Roles
- the <PathManipulation>
- the <TaintedInput>

Parent weaknesses
CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Detection Patterns
8.2.85 ASCQM Sanitize User Input used in Path Manipulation

7.4.15 CWE-252 Unchecked Return Value

Reference
https://cwe.mitre.org/data/definitions/252

Roles
- the <OperationCall>

Detection Patterns
8.2.21 ASCQM Check Return Value of Resource Operations Immediately

8.2.20 ASCQM Handle Return Value of Must Check Operations

7.4.16 CWE-259 Use of Hard-coded Password

Reference
https://cwe.mitre.org/data/definitions/259

Roles
- the <Authentication>
- the <HardCodedValue>

Parent weaknesses
CWE-798 Use of Hard-coded Credentials

Detection Patterns
8.2.43 ASCQM Ban Hard-Coded Literals used to Connect to Resource

7.4.17 CWE-321 Use of Hard-coded Cryptographic Key

Reference
https://cwe.mitre.org/data/definitions/321

Roles
- the <Authentication>
- the <HardCodedCryptographicKey>

Parent weaknesses
CWE-798 Use of Hard-coded Credentials

Detection Patterns
8.2.43 ASCQM Ban Hard-Coded Literals used to Connect to Resource

7.4.18 CWE-36 Absolute Path Traversal

Reference
https://cwe.mitre.org/data/definitions/36

Roles
- the <PathManipulation>
- the <TaintedInput>

Parent weaknesses
CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Detection Patterns
8.2.85 ASCQM Sanitize User Input used in Path Manipulation

7.4.19 CWE-366 Race Condition within a Thread

Reference
https://cwe.mitre.org/data/definitions/366

Roles
- the <Thread1>
- the <Thread2>
- the <ConflictingResource>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.57 ASCQM Ban Creation of Lock On Private Non-Static Object to Access Private Static Data
8.2.48 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context

7.4.20 CWE-369 Divide by Zero

Reference
https://cwe.mitre.org/data/definitions/369

Roles
- the <Division>

Parent weaknesses
CWE-682 Incorrect Calculation

Detection Patterns
8.2.56 ASCQM Check and Handle ZERO Value before Use as Divisor

7.4.21 CWE-401 Improper Release of Memory Before Removing Last Reference ('Memory Leak')

Reference
https://cwe.mitre.org/data/definitions/401

Roles
- the <MemoryAllocation>

Parent weaknesses
CWE-404 Improper Resource Shutdown or Release

Detection Patterns
8.2.29 ASCQM Ban Comma Operator from Delete Statement
8.2.32 ASCQM Implement Required Operations for Manual Resource Management
8.2.34 ASCQM Release Memory After Use
8.2.31 ASCQM Release Memory after Use with Correct Operation
8.2.33 ASCQM Release Platform Resource after Use
8.2.30 ASCQM Release in Destructor Memory Allocated in Constructor

7.4.22 CWE-404 Improper Resource Shutdown or Release

Reference
https://cwe.mitre.org/data/definitions/404

Roles
- the <ResourceAllocation>

Contributing weaknesses
CWE-401 Improper Release of Memory Before Removing Last Reference ('Memory Leak')
CWE-772 Missing Release of Resource after Effective Lifetime
CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime

Detection Patterns
8.2.29 ASCQM Ban Comma Operator from Delete Statement
8.2.35 ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor
8.2.38 ASCQM Implement Virtual Destructor for Classes with Virtual Methods
8.2.36 ASCQM Implement Virtual Destructor for Parent Classes
8.2.37 ASCQM Release File Resource after Use in Operation
8.2.33 ASCQM Release Platform Resource after Use
8.2.30 ASCQM Release in Destructor Memory Allocated in Constructor

7.4.23 CWE-424 Improper Protection of Alternate Path

Reference
https://cwe.mitre.org/data/definitions/424

Roles
- the <AlternatePath>

Detection Patterns
8.2.44 ASCQM Ban Unintended Paths

7.4.24 CWE-434 Unrestricted Upload of File with Dangerous Type

Reference
https://cwe.mitre.org/data/definitions/434

Roles
- the <FileUpload>

Detection Patterns
8.2.85 ASCQM Sanitize User Input used in Path Manipulation

7.4.25 CWE-456 Missing Initialization of a Variable

Reference
https://cwe.mitre.org/data/definitions/456

Roles
- the <VariableDeclaration>

Parent weaknesses
CWE-665 Improper Initialization

Detection Patterns
8.2.75 ASCQM Ban Allocation of Memory with Null Size
8.2.74 ASCQM Initialize Variables

7.4.26 CWE-457 Use of Uninitialized Variable

Reference
https://cwe.mitre.org/data/definitions/457

Roles
- the <VariableDeclaration>
- the <VariableUse>

Parent weaknesses
CWE-665 Improper Initialization

Detection Patterns
8.2.75 ASCQM Ban Allocation of Memory with Null Size
8.2.74 ASCQM Initialize Variables

7.4.27 CWE-477 Use of Obsolete Function

Reference
https://cwe.mitre.org/data/definitions/477

Roles
- the <ObsoleteFunctionCall>

Detection Patterns
8.2.93 ASCQM Ban Use of Deprecated Libraries

7.4.28 CWE-480 Use of Incorrect Operator

Reference
https://cwe.mitre.org/data/definitions/480

Roles
- the <Operator>

Detection Patterns
8.2.24 ASCQM Ban Assignment Operation Inside Logic Blocks
8.2.25 ASCQM Ban Comparison Expression Outside Logic Blocks
8.2.23 ASCQM Ban Incorrect Object Comparison
8.2.26 ASCQM Ban Incorrect String Comparison
8.2.27 ASCQM Ban Logical Operation with a Constant Operand

7.4.29 CWE-502 Deserialization of Untrusted Data

Reference
https://cwe.mitre.org/data/definitions/502

Roles
- the <Deserialization>
- the <TaintedData>

Detection Patterns
8.2.98 ASCQM Sanitize User Input used as Serialized Object

7.4.30 CWE-543 Use of Singleton Pattern Without Synchronization in a Multithreaded Context

Reference
https://cwe.mitre.org/data/definitions/543

Roles
- the <SingletonUse>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.41 ASCQM Ban Non-Final Static Data in Multi-Threaded Context
8.2.46 ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context

7.4.31 CWE-564 SQL Injection: Hibernate

Reference
https://cwe.mitre.org/data/definitions/564

Roles
- the <HibernateSQLStatement>
- the <TaintedInput>

Parent weaknesses
CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Detection Patterns
8.2.86 ASCQM Sanitize User Input used in SQL Access

7.4.32 CWE-567 Unsynchronized Access to Shared Data in a Multithreaded Context

Reference
https://cwe.mitre.org/data/definitions/567

Roles
- the <SharedDataAccess>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.41 ASCQM Ban Non-Final Static Data in Multi-Threaded Context
8.2.48 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context

7.4.33 CWE-570 Expression is Always False

Reference
https://cwe.mitre.org/data/definitions/570

Roles
- the <BooleanExpression>

Detection Patterns
8.2.79 ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use

7.4.34 CWE-571 Expression is Always True

Reference
https://cwe.mitre.org/data/definitions/571

Roles
- the <BooleanExpression>

Detection Patterns
8.2.79 ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use

7.4.35 CWE-606 Unchecked Input for Loop Condition

Reference
https://cwe.mitre.org/data/definitions/606

Roles
- the <LoopCondition>
- the <TaintedValue>

Detection Patterns
8.2.97 ASCQM Sanitize User Input used in Loop Condition

7.4.36 CWE-643 Improper Neutralization of Data within XPath Expressions ('XPath Injection')

Reference
https://cwe.mitre.org/data/definitions/643

Roles
- the <XPathExpression>
- the <TaintedValue>

Detection Patterns
8.2.88 ASCQM Sanitize User Input used in Document Navigation Expression

7.4.37 CWE-652 Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')

Reference
https://cwe.mitre.org/data/definitions/652

Roles
- the <XQueryExpression>
- the <TaintedValue>

Detection Patterns
8.2.87 ASCQM Sanitize User Input used in Document Manipulation Expression

7.4.38 CWE-662 Improper Synchronization

Reference
https://cwe.mitre.org/data/definitions/662

Roles
- the <Thread1>
- the <Thread2>
- the <SharedResourceAccess>

Contributing weaknesses
CWE-366 Race Condition within a Thread
CWE-543 Use of Singleton Pattern Without Synchronization in a Multithreaded Context
CWE-567 Unsynchronized Access to Shared Data in a Multithreaded Context
CWE-667 Improper Locking
CWE-764 Multiple Locks of a Critical Resource
CWE-820 Missing Synchronization
CWE-821 Incorrect Synchronization
CWE-833 Deadlock
CWE-1058 Invokable Control Element in Multi-Thread Context with non-Final Static Storable or Member Element
CWE-1096Singleton Class Instance Creation without Proper Locking or Synchronization

Detection Patterns
8.2.61 ASCQM Ban Creation of Lock On Inappropriate Object Type
8.2.60 ASCQM Ban Creation of Lock On Non-Final Object
8.2.57 ASCQM Ban Creation of Lock On Private Non-Static Object to Access Private Static Data
8.2.68 ASCQM Ban Incompatible Lock Acquisition Sequences
8.2.49 ASCQM Ban Incorrect Synchronization Mechanisms
8.2.41 ASCQM Ban Non-Final Static Data in Multi-Threaded Context
8.2.50 ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context
8.2.73 ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock
8.2.59 ASCQM Ban Sleep Between Lock Acquisition and Release
8.2.69 ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues
8.2.48 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context
8.2.58 ASCQM Release Lock After Use
8.2.46 ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context

7.4.39 CWE-665 Improper Initialization

Reference
https://cwe.mitre.org/data/definitions/665

Roles
- the <Initialization>

Contributing weaknesses
CWE-456 Missing Initialization of a Variable
CWE-457 Use of Uninitialized Variable

Detection Patterns
8.2.78 ASCQM Ban Self Assignment
8.2.9 ASCQM Initialize Pointers before Use
8.2.77 ASCQM Initialize Variables before Use

7.4.40 CWE-667 Improper Locking

Reference
https://cwe.mitre.org/data/definitions/667

Roles
- the <Thread1>
- the <Thread2>
- the <SharedResourceAccess>
- the <Lock>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.61 ASCQM Ban Creation of Lock On Inappropriate Object Type
8.2.60 ASCQM Ban Creation of Lock On Non-Final Object
8.2.57 ASCQM Ban Creation of Lock On Private Non-Static Object to Access Private Static Data

8.2.50 ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context
8.2.59 ASCQM Ban Sleep Between Lock Acquisition and Release
8.2.48 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context
8.2.58 ASCQM Release Lock After Use

7.4.41 CWE-672 Operation on a Resource after Expiration or Release

Reference
https://cwe.mitre.org/data/definitions/672

Roles
- the <ResourceRelease>
- the <ResourceAccess>

Contributing weaknesses
CWE-415 Double Free
CWE-416 Use After Free

Detection Patterns
8.2.12 ASCQM Ban Double Release of Resource
8.2.11 ASCQM Ban Use of Expired Resource

7.4.42 CWE-681 Incorrect Conversion between Numeric Types

Reference
https://cwe.mitre.org/data/definitions/681

Roles
- the <NumericConversion>

Contributing weaknesses
CWE-194 Unexpected Sign Extension
CWE-195 Signed to Unsigned Conversion Error
CWE-196 Unsigned to Signed Conversion Error
CWE-197 Numeric Truncation Error

Detection Patterns
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion

7.4.43 CWE-682 Incorrect Calculation

Reference
https://cwe.mitre.org/data/definitions/682

Roles
- the <Calculation>

Contributing weaknesses
CWE-131 Incorrect Calculation of Buffer Size
CWE-369 Divide by Zero

Detection Patterns
8.2.71 ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size
8.2.70 ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation
8.2.72 ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value
8.2.56 ASCQM Check and Handle ZERO Value before Use as Divisor

7.4.44 CWE-732 Incorrect Permission Assignment for Critical Resource

Reference
https://cwe.mitre.org/data/definitions/732

Roles
- the <PermissionAssignment>

Detection Patterns
8.2.100 ASCQM Ban File Creation with Default Permissions

7.4.45 CWE-77 Improper Neutralization of Special Elements used in a Command ('Command Injection')

Reference
https://cwe.mitre.org/data/definitions/77

Roles
- the <Command>
- the <TaintedValue>

Contributing weaknesses
CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
CWE-88 Argument Injection or Modification

Detection Patterns
8.2.128 ASCQM Sanitize User Input used in Expression Language Statement
8.2.92 ASCQM Sanitize User Input used in System Command

7.4.46 CWE-772 Missing Release of Resource after Effective Lifetime

Reference
https://cwe.mitre.org/data/definitions/772

Roles
- the <ResourceAllocation>

Parent weaknesses
CWE-404 Improper Resource Shutdown or Release

Detection Patterns
8.2.37 ASCQM Release File Resource after Use in Operation
8.2.33 ASCQM Release Platform Resource after Use
8.2.30 ASCQM Release in Destructor Memory Allocated in Constructor

7.4.47 CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime

Reference
https://cwe.mitre.org/data/definitions/775

Roles
- the <FileDescriptorOrHandleAllocation>

Parent weaknesses
CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime

Detection Patterns
8.2.63 ASCQM Release File Resource after Use in Class
8.2.37 ASCQM Release File Resource after Use in Operation

7.4.48 CWE-778 Insufficient Logging

Reference
https://cwe.mitre.org/data/definitions/778

Roles
- the <SecurityExceptionOrError>

Detection Patterns
8.2.99 ASCQM Log Caught Security Exceptions

7.4.49 CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

Reference
https://cwe.mitre.org/data/definitions/78

Roles
- the <OSCommand>
- the <TaintedValue>

Parent weaknesses
CWE-77 Improper Neutralization of Special Elements used in a Command ('Command Injection')

Detection Patterns
8.2.92 ASCQM Sanitize User Input used in System Command

7.4.50 CWE-783 Operator Precedence Logic Error

Reference
https://cwe.mitre.org/data/definitions/783

Roles
- the <Formula>

Detection Patterns
8.2.82 ASCQM Ban Incorrect Joint Comparison
8.2.81 ASCQM Ban Not Operator On Non-Boolean Operand Of Comparison Operation
8.2.80 ASCQM Ban Not Operator On Operand Of Bitwise Operation

7.4.51 CWE-786 Access of Memory Location Before Start of Buffer

Reference
https://cwe.mitre.org/data/definitions/786

Roles
- the <MemoryAccess>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.1 ASCQM Check Index of Array Access
8.2.4 ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities

7.4.52 CWE-787 Out-of-bounds Write

Reference
https://cwe.mitre.org/data/definitions/787

Roles
- the <BufferWrite>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.1 ASCQM Check Index of Array Access
8.2.2 ASCQM Check Input of Memory Manipulation Primitives

7.4.53 CWE-788 Access of Memory Location After End of Buffer

Reference
https://cwe.mitre.org/data/definitions/788

Roles
- the <MemoryAccess>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities
8.2.1 ASCQM Check Index of Array Access
8.2.2 ASCQM Check Input of Memory Manipulation Primitives

7.4.54 CWE-789 Uncontrolled Memory Allocation

Reference
https://cwe.mitre.org/data/definitions/789

Roles
- the <MemoryAllocation>

Detection Patterns
8.2.95 ASCQM Check Input of Memory Allocation Primitives
8.2.94 ASCQM Sanitize User Input used as Array Index

7.4.55 CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Reference
https://cwe.mitre.org/data/definitions/79

Roles
- the <WebPageGenerationStatement>
- the <TaintedInput>

Detection Patterns
8.2.90 ASCQM Sanitize Stored Input used in User Output
8.2.91 ASCQM Sanitize User Input used in User Output

7.4.56 CWE-798 Use of Hard-coded Credentials

Reference
https://cwe.mitre.org/data/definitions/798

Roles
- the <HardCodedValue>
- the <Authentication>

Contributing weaknesses
CWE-259 Use of Hard-coded Password
CWE-321 Use of Hard-coded Cryptographic Key

Detection Patterns
8.2.43 ASCQM Ban Hard-Coded Literals used to Connect to Resource

7.4.57 CWE-805 Buffer Access with Incorrect Length Value

Reference
https://cwe.mitre.org/data/definitions/805

Roles
- the <BufferAccess>
- the <LengthParameter>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities
8.2.2 ASCQM Check Input of Memory Manipulation Primitives
8.2.4 ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities

7.4.58 CWE-820 Missing Synchronization

Reference
https://cwe.mitre.org/data/definitions/820

Roles
- the <SharedResourceUse>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.50 ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context

7.4.59 CWE-821 Incorrect Synchronization

Reference
https://cwe.mitre.org/data/definitions/821
Roles
- the <SharedResourceUse>
- the <IncorrectSynchronization>

Parent weaknesses
CWE-662 Improper Synchronization

Detection Patterns
8.2.49 ASCQM Ban Incorrect Synchronization Mechanisms

7.4.60 CWE-822 Untrusted Pointer Dereference

Reference
https://cwe.mitre.org/data/definitions/822

Roles
- the <PointerDereferencing>
- the <TaintedInput>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.8 ASCQM Sanitize User Input used as Pointer

7.4.61 CWE-823 Use of Out-of-range Pointer Offset

Reference
https://cwe.mitre.org/data/definitions/823

Roles
- the <PointerOffset>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.7 ASCQM Check Offset used in Pointer Arithmetic

7.4.62 CWE-824 Access of Uninitialized Pointer

Reference
https://cwe.mitre.org/data/definitions/824

Roles
- the <PointerAccess>

Parent weaknesses
CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Detection Patterns
8.2.9 ASCQM Initialize Pointers before Use

7.4.63 CWE-825 Expired Pointer Dereference

Reference
https://cwe.mitre.org/data/definitions/825

Roles
- the <PointerAccess>
- the <PointerRelease>

Parent weaknesses
CWE-672 Operation on a Resource after Expiration or Release

Detection Patterns
8.2.5 ASCQM Ban Use of Expired Pointer

7.4.64 CWE-835 Loop with Unreachable Exit Condition ('Infinite Loop')

Reference
https://cwe.mitre.org/data/definitions/835

Roles
- the <InfiniteLoop>

Detection Patterns
8.2.55 ASCQM Ban Unmodified Loop Variable Within Loop
8.2.54 ASCQM Ban While TRUE Loop Without Path To Break

7.4.65 CWE-88 Argument Injection or Modification

Reference
https://cwe.mitre.org/data/definitions/88

Roles
- the <Command>
- the <TaintedInput>

Parent weaknesses
CWE-77 Improper Neutralization of Special Elements used in a Command ('Command Injection')

Detection Patterns
8.2.92 ASCQM Sanitize User Input used in System Command

7.4.66 CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Reference
https://cwe.mitre.org/data/definitions/89

Roles
- the <SQLStatement>
- the <TaintedInput>

Contributing weaknesses
Weakness CWE-564 SQL Injection: Hibernate

Detection Patterns
8.2.87 ASCQM Sanitize User Input used in Document Manipulation Expression
8.2.88 ASCQM Sanitize User Input used in Document Navigation Expression

7.4.67 CWE-90 Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')

Reference
https://cwe.mitre.org/data/definitions/90

Roles
- the <LDAPQuery>
- the <TaintedInput>

Detection Patterns
8.2.89 ASCQM Sanitize User Input used to access Directory Resources

7.4.68 CWE-91 XML Injection (aka Blind XPath Injection)

Reference
https://cwe.mitre.org/data/definitions/91

Roles
- the <XMLHandlingExpression>
- the <TaintedValue>

Detection Patterns
8.2.87 ASCQM Sanitize User Input used in Document Manipulation Expression
8.2.88 ASCQM Sanitize User Input used in Document Navigation Expression

7.4.69 CWE-99 Improper Control of Resource Identifiers ('Resource Injection')

Reference
https://cwe.mitre.org/data/definitions/99

Roles
- the <ResourceIdentifier>
- the <TaintedValue>

Detection Patterns
8.2.85 ASCQM Sanitize User Input used in Path Manipulation

7.4.70 CWE-611 Improper Restriction of XML External Entity Reference ('XXE')

Reference
https://cwe.mitre.org/data/definitions/CWE-611

Roles
- the <XMLHandlingOperation>

Detection Patterns
8.2.84 ASCQM Secure Use of Unsafe XML Processing with Secure Parser
8.2.83 ASCQM Secure XML Parsing with Secure Options

7.4.71 CWE-1057 Data Access Control Element from Outside Designated Data Manager Component

Usage name
Circumventing data access routines

Reference
https://cwe.mitre.org/data/definitions/1057

Roles
- the <DataManager>
- the <DataAccess>

Detection Patterns
8.2.44 ASCQM Ban Unintended Paths

7.4.72 CWE-415 Double Free

Reference
https://cwe.mitre.org/data/definitions/415

Roles
- the <ResourceRelease >
- the <ResourceAccess>

- the <ResourceUse>

Parent weaknesses
CWE-672 Operation on a Resource after Expiration or Release

Detection Patterns
8.2.76 ASCQM Ban Double Free On Pointers

7.4.73 CWE-416 Use After Free

Reference
https://cwe.mitre.org/data/definitions/416

Roles
- the <ResourceRelease>
- the <ResourceUse>

Parent weaknesses
CWE-672 Operation on a Resource after Expiration or Release

Detection Patterns
8.2.14 ASCQM Ban Free Operation on Pointer Received as Parameter
8.2.5 ASCQM Ban Use of Expired Pointer
8.2.13 ASCQM Implement Copy Constructor for Class with Pointer Resource

7.4.74 Security Detection Patterns

Detection Patterns
8.2.75 ASCQM Ban Allocation of Memory with Null Size
8.2.24 ASCQM Ban Assignment Operation Inside Logic Blocks
8.2.71 ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size
8.2.70 ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation
8.2.72 ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value
8.2.29 ASCQM Ban Comma Operator from Delete Statement
8.2.25 ASCQM Ban Comparison Expression Outside Logic Blocks
8.2.61 ASCQM Ban Creation of Lock on Inappropriate Object Type
8.2.60 ASCQM Ban Creation of Lock on Non-Final Object
8.2.57 ASCQM Ban Creation of Lock on Private Non-Static Object to Access Private Static Data

8.2.76 ASCQM Ban Double Free on Pointers
8.2.12 ASCQM Ban Double Release of Resource
8.2.100 ASCQM Ban File Creation with Default Permissions
8.2.14 ASCQM Ban Free Operation on Pointer Received as Parameter
8.2.43 ASCQM Ban Hard-Coded Literals used to Connect to Resource
8.2.68 ASCQM Ban Incompatible Lock Acquisition Sequences
8.2.82 ASCQM Ban Incorrect Joint Comparison
8.2.47 ASCQM Ban Incorrect Numeric Implicit Conversion
8.2.23 ASCQM Ban Incorrect Object Comparison
8.2.26 ASCQM Ban Incorrect String Comparison
8.2.49 ASCQM Ban Incorrect Synchronization Mechanisms
8.2.6 ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities
8.2.27 ASCQM Ban Logical Operation with a Constant Operand
8.2.41 ASCQM Ban Non-Final Static Data in Multi-Threaded Context
8.2.81 ASCQM Ban Not Operator on Non-Boolean Operand of Comparison Operation
8.2.80 ASCQM Ban Not Operator on Operand of Bitwise Operation
8.2.50 ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context
8.2.78 ASCQM Ban Self Assignment
8.2.73 ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock
8.2.59 ASCQM Ban Sleep Between Lock Acquisition and Release
8.2.3 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities
8.2.44 ASCQM Ban Unintended Paths
8.2.55 ASCQM Ban Unmodified Loop Variable Within Loop
8.2.93 ASCQM Ban Use of Deprecated Libraries
8.2.5 ASCQM Ban Use of Expired Pointer
8.2.11 ASCQM Ban Use of Expired Resource
8.2.69 ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues
8.2.54 ASCQM Ban While TRUE Loop Without Path to Break
8.2.79 ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use
8.2.1 ASCQM Check Index of Array Access
8.2.95 ASCQM Check Input of Memory Allocation Primitives
8.2.2 ASCQM Check Input of Memory Manipulation Primitives
8.2.4 ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities
8.2.7 ASCQM Check Offset used in Pointer Arithmetic
8.2.21 ASCQM Check Return Value of Resource Operations Immediately
8.2.56 ASCQM Check and Handle ZERO Value before Use as Divisor
8.2.48 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context
8.2.20 ASCQM Handle Return Value of Must Check Operations
8.2.13 ASCQM Implement Copy Constructor for Class With Pointer Resource
8.2.32 ASCQM Implement Required Operations for Manual Resource Management
8.2.35 ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor
8.2.38 ASCQM Implement Virtual Destructor for Classes with Virtual Methods
8.2.36 ASCQM Implement Virtual Destructor for Parent Classes
8.2.9 ASCQM Initialize Pointers before Use
8.2.74 ASCQM Initialize Variables
8.2.77 ASCQM Initialize Variables before Use
8.2.99 ASCQM Log Caught Security Exceptions
8.2.63 ASCQM Release File Resource after Use in Class
8.2.37 ASCQM Release File Resource after Use in Operation
8.2.58 ASCQM Release Lock After Use
8.2.34 ASCQM Release Memory After Use
8.2.31 ASCQM Release Memory after Use with Correct Operation
8.2.33 ASCQM Release Platform Resource after Use

8 ASCQM Weakness Detection Patterns

8.1 Specification of Detection Patterns

Detection patterns provide guidance for automated detection of the weaknesses enumerated in Clause 7. Each weakness may have several different instantiations in the source code. Thus, a weakness may be associated with several different detection patterns. Each detection pattern may be associated with weaknesses in several different quality measures. There are 135 detection patterns associated with the weaknesses in Automated Source Code Quality Measures. This number will grow as more detection patterns are discovered and specified.

Detection Patterns use micro-KDM to provide greater granularity to their specification of weakness patterns. Additional semantic constraints are required to coordinate producers and consumers of KDM models to use the KDM Program Element layer for control- and data-flow analysis applications, as well as for providing more precision for the Resource Layer and the Abstraction Layer. Micro-KDM achieves this by constraining the granularity of the leaf action elements and their meaning by providing the set of micro-actions with predefined semantics. Micro-KDM treats the original macro-action as a container that owns certain micro-actions with predefined semantics. Thus, precise semantics of the macro-action is defined. Micro-KDM constrains the patterns of how to map the statements of the existing system as determined by the programming language into KDM.

8.2 ASCQM Check Index of Array Access

Descriptor

ASCQM Check Index of Array
Access(PathFromDeclarationStatementToUseAsAnIndexStatement,
VariableDeclarationStatement, ArrayAccessStatement)

Description

Identify occurrences in application model where
- the <PathFromDeclarationStatementToUseAsAnIndexStatement> path
- from the <VariableDeclarationStatement> variable declaration statement
- to the <ArrayAccessStatement> array access statement using the variable as an index,
- lacks a range check operation.

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
StorableUnit id="su1"
StorableUnit id="su2"
ArrayType id="at1"
StorableUnit id="su3" type="at1"
...
ActionElement id="ae2"
   Flow "ae3"
    Reads "su1"
    Writes "su2"
ActionElement id="ae3"
    Flow "ae4"
ActionElement id="ae4"
    Flow "ae5"
ActionElement id="ae5" kind="ArraySelect|ArrayReplace"
    Addresses "su3"
```

```
    Reads "su2"
    Reads|Writes ...
 ...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ActionElement id="ae2" kind="GreaterThan|GreaterThanOrEqual"
    Reads "su2"
    Reads ...
    ...
ActionElement id="ae3" kind="LessThan|LessThanOrEqual"
    Reads "su2"
    Reads ...
    ...
```

What to report

Roles to report are:
- the <PathFromDeclarationStatementToUseAsAnIndexStatement> path
- the <VariableDeclarationStatement> variable declaration statement
- the <ArrayAccessStatement> array access statement

8.3 ASCQM Check Input of Memory Manipulation Primitives

Descriptor

ASCQM Check Input of Memory Manipulation Primitives (MemoryManipulationCall)

Description

Identify occurrences in application model where:
- the <MemoryManipulationCall> call to a memory manipulation function, procedure, method, ... with boundary checking capabilities
- uses the length parameter without range checking its value

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PointerType id="pt1"
IntegerType id="it1"
ControlElement id="ce1" name="memcpy|..." type="ce1_signature"
    Signature id="ce1_signature"
        ...
        ParameterUnit id="pu1" type="dt1" kind="byValue"
        ParameterUnit id="pu2" type="pt1" kind="return"
    ...
...
StorableUnit id="su1"  type="it1"
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
    Reads "su1"
    Calls "ce1"
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ActionElement id="ae2" kind="GreaterThan|GreaterThanOrEqual"
    Reads "su1"
    ...
ActionElement id="ae3" kind="LessThan|LessThanOrEqual"
    Reads "su1"
    ...
```

What to report

Roles to report:
- the <MemoryManipulationCall> call to a memory manipulation function, procedure, method, ... with boundary checking capabilities

8.4 ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities

Descriptor

ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities(StringManipulationCall)

Description

Identify occurrences in application model where:
- the <StringManipulationCall> call to a string manipulation function, procedure, method, ... without boundary checking capabilities

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce1" name="strcpy|strlen|..."
    ...
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
    Calls "ce1"
```

What to report

Roles to report:
- the <StringManipulationCall> call to a string manipulation function, procedure, method, ... without boundary checking capabilities

8.5 ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities

Descriptor

ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities (StringManipulationCall)

Description
Identify occurrences in application model where:
- the <StringManipulationCall> call to a string manipulation function, procedure, method, ... with boundary checking
  capabilities
- uses the length parameter without range checking its value

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
StringType id="st1"
IntegerType id="it1"
ControlElement id="ce1" name="strncpy|strncat|..." type="ce1_signature"
    Signature id="ce1_signature"
        ParameterUnit id="pu1" type="st1"
        ParameterUnit id="pu2" type="it1" kind="byValue"
        ParamteterUnit id="pu3" type="st1" kind="return"
    ...
...
StorableUnit id="su1" type="it1"
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
    Reads "su1"
    Calls "ce1"
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ActionElement id="ae2" kind="GreaterThan|GreaterThanOrEqual"
    Reads "su1"
    ...
ActionElement id="ae3" kind="LessThan|LessThanOrEqual"
    Reads "su1"
    ...
```

What to report
Roles to report:
- the <StringManipulationCall> call to a string manipulation function, procedure, method, ... with boundary checking capabilities

8.6 ASCQM Ban Use of Expired Pointer

Descriptor
ASCQM Ban Use of Expired Pointer (PathToPointerAccessFromPointerRelease, PointerReleaseStatement, PointerAccessStatement)

Description
Identify occurrences in application model where:
- the <PathToPointerAccessFromPointerRelease> path
- from the <PointerReleaseStatement> resource release statement
- to the <PointerAccessStatement> resource access statement

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit|IntegerType|DecimalType|FloatType|StringType|VoidType|... id="dt1"
PointerType id="pt1"
    ItemUnit id="pi1" type="dt1"
StorableUnit id="su1"  type="pt1"
...
ActionElement id="ae1" name="free|delete|..."
    Addresses "pt1"
    Flows "ae2"
ActionElement id="ae2"
    Flows "ae3"
ActionElement id="ae3" kind=PtrSelect|PtrReplace|Call|PtrCall|MethodCall|VirtualCall"
    Reads|Addresses "pt1"
...

or

ClassUnit|IntegerType|DecimalType|FloatType|StringType|VoidType|... id="dt1" name="dt1"
PointerType id="pt1" name="pt1"
    ItemUnit id="iu1" type="dt1" ext="dt1 & pt1"
StorableUnit id="su1"  type="dt1"
StorableUnit id="su2"  type="pt1"
    HasType "pt1"
    HasValue "su1"
...
ActionElement id="ae1" name="free|delete|...|push_back|..."
    Addresses "su1"
    Flows "ae2"
ActionElement id="ae2"
    Flows "ae3"
ActionElement id="ae3" kind=PtrSelect|PtrReplace|Call|PtrCall|MethodCall|VirtualCall"
    Reads|Addresses "su2"
```

What to report

Roles to report:
- the <PathToPointerAccessFromPointerRelease> path
- the <PointerReleaseStatement> resource release statement
- the <PointerAccessStatement> resource access statement

8.7 ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities

Descriptor

ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities (InputAcquisitionCall)

Description

Identify occurrences in application model where:
- the <InputAcquisitionCall> call to an input acquisition function, procedure, method, ... without boundary checking capabilities

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce1" name="gets|scanf|..."
    ...
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
    Calls "ce1"
```

What to report

Roles to report:
- the <InputAcquisitionCall> call to an input acquisition function, procedure, method, ... without boundary checking capabilities

8.8 ASCQM Check Offset used in Pointer Arithmetic

Descriptor

ASCQM Check Offset used in Pointer Arithmetic (ArithmeticExpression, EvaluationStatement)

Description

Identify occurrences in application model where:
- the result of the <ArithmeticExpression> arithmetic expression,
- with an offset value which is not range checked
- is used to dererence the pointer in the <EvaluationStatement> evaluation statement

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
PointerType id="pt1"
StorableUnit id="su1" type="pt1"
...
IntegerType id="it1"
StorableUnit id="su2" type="it1"
StorableUnit id="su3" type="it1"
...
ActionElement id="ae1" kind="Add|Substract"
    Reads "su1"
    Reads "su2"
    Writes "su3"
...
ActionElement id="ae2" kind="PtrSelect|PtrReplace"
    Addresses "su3"
    ..
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ActionElement id="ae2" kind="GreaterThan|GreaterThanOrEqual"
    Reads "su2"
    Reads ...
    ...
ActionElement id="ae3" kind="LessThan|LessThanOrEqual"
    Reads "su2"
```

```
    Reads ...
    ...
```

What to report

Roles to report are:
- the <ArithmeticExpression> arithmetic expression
- the <EvaluationStatement> evaluation statement

8.9 ASCQM Sanitize User Input used as Pointer

Descriptor

ASCQM Sanitize User Input used as Pointer (PathFromUserInputToPointerDereferencing, UserInput, PointerDereferencingStatement, PointerDereferencingSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToPointerDereferencing> path
- from the <UserInput> user interface input
- to the <PointerDereferencingStatement> pointer dereferencing statement,
- lacks a sanitization operation from the <PointerDereferencingSanitizationControlElementList> list of vetted sanitizations.

The list of vetted sanitization primitives is an input to provide to the measurement process.

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
UIModel
    UIField id="uf1"
    UIAction id="ua1" implementation="ae1" kind="input"
        ReadsUI "uf1"
...
CodeModel
    ...
    StorableUnit id="su1"
    StorableUnit id="su2"
    ActionElement id="ae1" kind="UI"
        Writes "su1"
        Flow "ae2"
    ActionElement id="ae2"
        Flow "ae3"
        Reads "su1"
        Writes "su2"
    ActionElement id="ae3"
        Flow "ae4"
    ActionElement id="ae4"
        Flow "ae5"
    ActionElement id="ae5" kind="PtrSelect"
        Addresses "su2"
        Reads|Writes ...
 ...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce1" kind="sanitization"
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
        Flow "ae4"
        Calls "ce1"
        Reads "su2"
        Writes "su2"
...
```

What to report

Roles to report are:
- the <PathFromUserInputToPointerDereferencing> path
- the <UserInput> user interface input
- the <PointerDereferencingStatement> pointer dereferencing statement,
- the <PointerDereferencingSanitizationControlElementList> list of vetted sanitizations.

8.10 ASCQM Initialize Pointers before Use

Descriptor

ASCQM Initialize Pointers before Use (PathToPointerAccessFromPointerDeclaration, PointerDeclarationStatement, PointerAccessStatement)

Description

Identify occurrences in application model where:
- the <PathToPointerAccessFromPointerDeclaration> path
- from the <PointerDeclarationStatement> pointer declaration statement
- to the <PointerAccessStatement> pointer access statement
- lacks a pointer initialization statement

excluding variable and platform resources

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
PointerType id="pt1"
StorableUnit id="su1" type="pt1"
...
ActionElement id="ae2" ...
     Flows "ae3"
ActionElement id="ae3" kind="PtrSelect"
     Reads "su1"
     ...
...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
ActionElement id="ae1" kind="Assign|Ptr"
    Writes "su1"
    Flows "ae2"
...
```

What to report

Roles to report are:
- the <PathToPointerAccessFromPointerDeclaration> path
- the <PointerDeclarationStatement> pointer declaration statement
- the <PointerAccessStatement> pointer access statement

8.11 ASCQM Check NULL Pointer Value before Use

Descriptor

ASCQM Check NULL Pointer Value before Use(EvaluationStatement)

Description

Identify occurrences in application model where:
- a pointer is evaluated in the <EvaluationStatement> evaluation statement
- with no NULL comparison operation performed on the pointer immediately before

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
PointerType id="pt1"
    ItemUnit id="iu1"
StorableUnit id="su1" type="pt1"
ActionElement id="ae3" kind="PtrSelect|PtrReplace"
    Reads "iu1"
    Addresses "su1"
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
Value id="v1" name="NULL|nullptr"
StorableUnit id="su2"
ActionElement id="ae1" kind="NotEqual"
    Reads "v1"
    Reads "su1"
    Writes "su2"
    Flows "ae2"
ActionElement id="ae2" kind="Condition"
    Reads "su2"
    TrueFlow "ae3"
    FalseFlow "ff1"
...
```

What to report
Roles to report are:
- the <EvaluationStatement> evaluation statement

8.12 ASCQM Ban Use of Expired Resource

Descriptor
ASCQM Ban Use of Expired Resource (PathToResourceAccessFromResourceRelease, ResourceReleaseStatement, ResourceAccessStatement)

Description
Identify occurrences in application model where:
- the <PathToResourceAccessFromResourceRelease> path
- from the <ResourceReleaseStatement> resource release statement
- to the <ResourceAccessStatement> resource access statement excluding pointers

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    ...
    DataManager|FileResource id="pr1"
    ...
    PlatformResource id="pa1" kind="open" implementation="ae4"
        ManagesResource "pr1"
    PlatformResource id="pa2" kind="close" implementation="ae1"
        ManagesResource "pr1"

...
CodeModel
    ...
    ActionElement id="ae1" kind="PlatformAction"
        Flows "ae3"
    ActionElement id="ae3"
         Flows "ae4"
    ActionElement id="ae4" kind="PlatformAction"
     ...
...
```

What to report
Roles to report:
- the <PathToResourceAccessFromResourceRelease> path
- the <ResourceReleaseStatement> resource release statement
- the <ResourceAccessStatement> resource access statement

8.13 ASCQM Ban Double Release of Resource

Descriptor
ASCQM Ban Double Release of Resource (PathToResourceReleaseFromResourceRelease, FirstResourceReleaseStatement, SecondResourceReleaseStatement)

Description
Identify occurrences in application model where:
- the <PathToResourceReleaseFromResourceRelease> path
- from the <FirstResourceReleaseStatement> resource release statement
- to the <SecondResourceReleaseStatement> resource release statement

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    ...
    DataManager|ExecutionResource id="pr1"
    ...
    PlatformAction id="pa2" kind="close" implementation="ae1 ae4"
        ManagesResource "pr1"

...
CodeModel
    ...
    ActionElement id="ae1" kind="PlatformAction"
        Flows "ae3"
    ActionElement id="ae3"
         Flows "ae4"
    ActionElement id="ae4" kind="PlatformAction"
        ...
...
```

What to report
Roles to report:
- the <PathToResourceReleaseFromResourceRelease> path
- the <FirstResourceReleaseStatement> resource release statement
- the <SecondResourceReleaseStatement> resource release statement

8.14 ASCQM Implement Copy Constructor for Class with Pointer Resource

Descriptor
ASCQM Implement Copy Constructor for Class with Pointer Resource (Class, Pointer)

Description
Identify occurrences in application model where:
- the <Class> Class
- owns the <Pointer> pointer resource
- but lacks a copy constructor

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
PointerType id="pointerType"
    ...
ClassUnit id="cu1"
    MemberUnit id="mu1" type="pointerType"
    ...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="cu1"
    ...
    MethodUnit is="m1"
name="class|this|__construct|new|New|__new__|alloc|constructor|initialize|..."
methodKind="constructor" type="m1_signature"
        Signature id = "m1_signature"
            ParameterUnit id="p1" name="p1" type="class" kind="byReference"
            ParameterUnit id="r" name="r" type="class" kind="return"
        ...
```

What to report
Roles to report are:
- the <Class> Class
- the <Pointer> pointer resource

8.15 ASCQM Ban Free Operation on Pointer Received as Parameter

Descriptor
ASCQM Ban Free Operation on Pointer Received as Parameter (ReleaseStatement, Signature)

Description
Identify occurrences in application model where:
- the pointer is released by the <ReleaseStatement> release statement
- and was received as a parameter in the <Signature> signature

The list of release operations is technology, language dependent. For example, with C-type languages: free, delete.

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
...
PointerType id="pt1"
...
ControlElement id="ce1" name="free|delete|..."
...
CallableUnit kind="regular|external|stored" | MethodUnit id="ce2" type="ce2_signature"
    Signature id="ce2_signature"
        ParameterUnit id="pu1" kind="byReference" type="pt1"
    ...
    ActionElement id="ae1" kind="Call|PtrCall[MethodCall|VirtualCall"
        Calls "ce1"
        Reads "pu1"
    ...
```

What to report
Roles to report are:
- the <ReleaseStatement> release statement
- the <Signature> signature

8.16 ASCQM Ban Delete of VOID Pointer

Descriptor
(DeclarationStatement, ReleaseStatement)

Description
Identify occurrences in application model where:
- the pointer declared as a VOID pointer in <DeclarationStatement> declaration statement
- is released by the <ReleaseStatement> release statement
- without ever been casted into a non-VOID pointer

The list of release operations is technology, language dependent. For example, with C-type languages: delete.

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
VoidType id="vt1"
PointerType id="pt1"
    ItemUnit id="iu1" type="vt1"
StorableUnit id="su1" type="pt1"
ControlElement id="ce1" name="delete|..."
    ...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Reads "su1"
    Calls "ce1"
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
...
IntegerType|DecimalType|FloatType|StringType|ClassUnit id="dt1"
PointerType id="pt2"
    ItemUnit id="iu2" type="dt1"
ActionElement id="ae2" kind="TypeCast|DynCast"
    Reads "su1"
    UsesType "pt2"
    Writes "su1"
...
```

What to report
Roles to report are:
- the <DeclarationStatement> declaration statement
- the <ReleaseStatement> release statement

8.17 ASCQM Ban Variable Increment or Decrement Operation in Operations using the Same Variable

Descriptor
ASCQM Ban Variable Increment or Decrement Operation in Operations using the Same Variable (VariableAssignment)

Description
Identify occurrences in application model where:
- the <VariableAssignment> variable assignment
- uses the outcome of increment or decrement operation on a variable
- jointly with the variable itself

e.g. : x + x++;

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
StorableUnit id="su1"
ActionElement id="ae1" kind="Compound"
    ActionElement id="ae2" kind="Incr|Decr"
        Addresses "su1"
    ...
    ActionElement id="ae3"
        ...
        Reads "su1"
...
```

What to report
Roles to report:
- the <VariableAssignment> variable assignment

8.18 ASCQM Ban Reading and Writing the Same Variable Used as Assignment Value

Descriptor
ASCQM Ban Reading and Writing the Same Variable Used as Assignment Value (VariableAssignment)

Description
Identify occurrences in application model where:
- the <VariableAssignment> variable assignment
- uses the outcome of an operation on a variable
- jointly with the assignment of the variable itself

e.g. : x = a + (a=2);

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
StorableUnit id="su1"
ActionElement id="ae1" kind="Compound"
    StorableUnit id="su2"
    ActionElement id="ae2" kind="Assign"
        ...
        Writes "su1"
    ...
    ActionElement id="ae3"
        ...
        Reads "su1"
        Writes "su2"
    ActionElement id="ae4" kind="Assign"
        Reads "su2"
```

```
        Writes ...
```

What to report

Roles to report:
- the <VariableAssignment> variable assignment

8.19 ASCQM Handle Return Value of Resource Operations

Descriptor

ASCQM Handle Return Value of Resource Operations (CallToTheOperation)

Description

Identify occurrences in application model where:
- the platform resource management function, method, procedure, ... is called in the <CallToTheOperation> call
  statement
- with no use in a conditional statement of the return value

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    ...
    DataManager|ExecutionResource|... id="pr1"
    ...
    PlatformResource id="pa1"  implementation="ae1"
        ManagesResource|ReadsResource|WritesResource "pr1"
    ...
CodeModel
...
    CallableUnit|MethodUnit id="ce1" type="ce1_signature"
        Signature id="ce1_signature"
            ParameterUnit id="pu1" kind="return"
    ...
    ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
StorableUnit id="su1"
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Writes "su1"
    Flows "ae2"
ActionElement id="ae2" kind="Switch"
    Reads "su1"
    GuardedFlow "gf1"
    GuardedFlow|FalseFlow "gf2"
...
```

or

```
StorableUnit id="su1"
StorableUnit id="su2"
```

```
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Writes "su1"
    Flows "ae2"
ActionElement id="ae2"
kind="Equal|NotEqual|LessThan|LessThanOrEqual|GreaterThan|GreatedThanOrEqual"
    Reads "su1"
    Writes "su2"
    Flows "ae3"
ActionElement id="ae3" kind="Condition"
    TrueFlow "tf1"
    FalseFlow "ff1"
...
```

What to report

Roles to report are:
- the <CallToTheOperation> call statement

8.20 ASCQM Ban Incorrect Numeric Conversion of Return Value

Descriptor

ASCQM Ban Incorrect Numeric Conversion of Return Value (FunctionMethodOrProcedure, VariableDataType, CallStatement, TargetDataType)

Description

Identify occurrences in application model where:
- the <FunctionMethodOrProcedure> function, method, procedure, ...
- declared to return a value with the <VariableDataType> numerical data type
- is called in the <CallStatement> call statement
- with assignment of its return value to a variable of the <TargetDataType> second numerical data type
- which is incompatible with the first one
- without any explicit casting

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
IntegerType|DecimalType|FloatType id="dt1"
IntegerType|DecimalType|FloatType id="dt2"
StorableUnit|ItemUnit|MemberUnit|Value id="de1" type="dt2"
...
CallableUnit|MethodUnit id="ce1" type="ce1_signature" attribute="CheckReturnValue|..."
    Signature id="ce1_signature"
        ParameterUnit id="pu1" kind="return" type="dt1"
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Calls "ce1"
    Writes "de1"
...
```

and the numeric datatypes are not compatible.

What to report

Roles to report are:
- the <FunctionMethodOrProcedure> function, method, procedure, ...
- the <VariableDataType> numerical data type

- the <CallStatement> call statement with assignment
- the <TargetDataType> second numerical data type

8.21 ASCQM Handle Return Value of Must Check Operations

Descriptor

ASCQM Handle Return Value of Must Check Operations (CallToTheOperation)

Description

Identify occurrences in application model where:
- the must-check function, method, procedure, ... is called in the <CallToTheOperation> call statement
- with no use in a conditional statement of the return value

The must-check nature of a function, method, procedure, ... is technology dependent. For example, in Java: the @CheckReturnValue annotation

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
CallableUnit|MethodUnit id="ce1" type="ce1_signature" attribute="CheckReturnValue|..."
    Signature id="ce1_signature"
        ParameterUnit id="pu1" kind="return"
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
StorableUnit id="su1"
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Writes "su1"
    Flows "ae2"
ActionElement id="ae2" kind="Switch"
    Reads "su1"
    GuardedFlow "gf1"
    GuardedFlow|FalseFlow "gf2"
...
```

or

```
StorableUnit id="su1"
StorableUnit id="su2"
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Writes "su1"
    Flows "ae2"
ActionElement id="ae2"
kind="Equal|NotEqual|LessThan|LessThanOrEqual|GreaterThan|GreatedThanOrEqual"
    Reads "su1"
    Writes "su2"
    Flows "ae3"
ActionElement id="ae3" kind="Condition"
    TrueFlow "tf1"
```

```
    FalseFlow "ff1"
...
```

What to report

Roles to report are:
- the <CallToTheOperation> call statement

8.22 ASCQM Check Return Value of Resource Operations Immediately

Descriptor

ASCQM Check Return Value of Resource Operations Immediately (CallToTheOperation)

Description

Identify occurrences in application model where:
- a platform resource management function, procedure, method, ... is called in the <CallToTheOperation> call statement
- with no operation performed immediately after on the return value

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    ...
    DataManager|ExecutionResource|... id="pr1"
    ...
    PlatformResource id="pa1"  implementation="ae1"
        ManagesResource|ReadsResource|WritesResource "pr1"
    ...
CodeModel
    CallableUnit|MethodUnit id="ce1" type="ce1_signature"
        Signature id="ce1_signature"
            ParameterUnit id="pu1" kind="return"
    ...
    ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
StorableUnit id="su1"
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Writes "su1"
    Flows "ae2"
ActionElement id="ae2"
    Reads "su1"
```

What to report

Roles to report are:
- the <CallToTheOperation> call statement8.22

8.23 ASCQM Ban Useless Handling of Exceptions

Descriptor
ASCQM Ban Useless Handling of Exceptions (CatchBlock)

Description
Identify occurrences in application model where:
- the <CatchBlock> catch block
- does not report on the error condition as a new throw or as a return value

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
...
CatchUnit id="cu1"
    ...
 ...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
...
CatchUnit id="cu1"
    ...
    ActionElement id="ae1" kind="Throw"
        Throws ...
...
```

or

```
...
CatchUnit id="cu1"
    ...
    ActionElement id="ae1" kind="Return"
        Reads ...
...
```

What to report
Roles to report are:
- the <CatchBlock> catch block

8.24 ASCQM Ban Incorrect Object Comparison

Descriptor
ASCQM Ban Incorrect Object Comparison (ObjectEqualityComparisonExpression)

Description
Identify occurrences in application model where:

- the <ObjectEqualityComparisonExpression> equality comparison expression between two objects

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="cu1"
StorableUnit|ItemUnit|MemberUnit id="de1" type="cu1"
StorableUnit|ItemUnit|MemberUnit id="de2" type="cu1"
ActionElement id="ae1" kind="Equals|NotEqual" ext="de1 == de2 | de1 != de2"
    Reads "de1"
    Reads "de2"
```

What to report

Roles to report are:
- the <ObjectEqualityComparisonExpression> equality comparison expression

8.25 ASCQM Ban Assignment Operation Inside Logic Blocks

Descriptor

ASCQM Ban Assignment Operation Inside Logic Blocks (AssignmentExpression, LogicBlock)

Description

Identify occurrences in application model where:
- the <AssignmentExpression> assignment expression
- is used within the <LogicBlock> logic block

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
...
ActionElement id="ae1" kind="Compound"
    StorableUnit|MemberUnit id="de1"
    ...
    ActionElement id="ae2" kind="Condition|Switch"
        Reads "de1"
        ActionElement id="ae3" kind="Assign"
            Writes "de1"
...
```

What to report

Roles to report are:
- the <AssignmentExpression> assighment expression
- the <LogicBlock> logic block

8.26 ASCQM Ban Comparison Expression Outside Logic Blocks

Descriptor

ASCQM Ban Comparison Expression Outside Logic Blocks (ComparisonExpression)

Description
Identify occurrences in application model where:
- the <ComparisonExpression> comparison expression
- is not used within a logic block

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
...
ActionElement id="ae1" kind="Compound"
    StorableUnit|MemberUnit id="de1"
    ...
    ActionElement id="ae3" kind="Equal"
        Reads "de1"
...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
...
ActionElement id="ae1" kind="Compound"
    StorableUnit|MemberUnit id="de1"
    ...
    ActionElement id="ae2" kind="Condition|Switch"
        Reads "su1"
        StorableUnit id="su1" type="register"
        ActionElement id="ae3" kind="Equal"
            Writes "su1"
            Reads "de1"
...
```

What to report
Roles to report are:
- the <ComparisonExpression> comparison expression

8.27 ASCQM Ban Incorrect String Comparison

Descriptor
ASCQM Ban Incorrect String Comparison (StringEqualityComparisonExpression)

Description
Identify occurrences in application model where:
- the <StringEqualityComparisonExpression> equality comparison expression between two strings

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
StringType id="st1"
StorableUnit|ItemUnit|MemberUnit id="de1" type="st1"
StorableUnit|ItemUnit|MemberUnit id="de2" type="st1"


ActionElement id="ae1" kind="Equals|NotEqual" ext="de1 == de2 | de1 != de2"
    Reads "de1"
```

```
    Reads "de2"
```

What to report
Roles to report are:
- the <StringEqualityComparisonExpression> equality comparison expression

8.28 ASCQM Ban Logical Operation with a Constant Operand

Descriptor
ASCQM Ban Logical Operation with a Constant Operand (ComparisonExpression)

Description
Identify occurrences in application model where:
- the <ComparisonExpression> comparison expression with a constant operand

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
Value id="v1"
...
ActionElement id="ae1" kind="And|Or|Xor"
    Reads "v1"
    ...
```

What to report
Roles to report are:
- the <ComparisonExpression> comparison expression

8.29 ASCQM Implement Correct Object Comparison Operations

Descriptor
ASCQM Implement Correct Object Comparison Operations (Class)

Description
Identify occurrences in application model where:
- the <Class> class
- lacking the required comparison operations

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="cu1"
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
BooleanType id="bt1"
IntegerType id="it1"
...
ClassUnit id="cu1"
    ...
    MethodUnit id="mu1" name="equals|Equals|operator==|..." type="mu1_signature"
        Signature id="mu1_signature"
            ParameterUnit id="pu1" kind="byReference" type="cu1"
            ParameterUnit id="pu2" kind="Return" type="bt1"
    ...
    MethodUnit id="mu2" name="hashCode|GetHashCode|hash|..." type="mu2_signature"
        Signature id="mu2_signature"
            ParameterUnit id="pu3" kind="byReference" type="cu1"
            ParameterUnit id="pu4" kind="Return" type="it1"
...
```

What to report

Roles to report are:
- the <Class> class

8.30 ASCQM Ban Comma Operator from Delete Statement

Descriptor

ASCQM Ban Comma Operator from Delete Statement (DeleteStatement, CommaStatement)

Description

Identify occurrences in application model where:
- the <DeleteStatement> delete statement
- compounded with the <CommaStatement> comma statement

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
...
CallableUnit id="cu1" name="delete" callableKind="operator"
CallableUnit id="cu2" name="comma" callableKind="operator"
...
ActionElement id="ae1" kind="Compound" ext="delete x, y"
    ActionElement id="ae2" kind="Call"
         Calls "cu1"
    ...
    ActionElement id="ae3" kind="Call"
         Calls "cu2"
    ...
...
```

What to report

Roles to report are:
- the <DeleteStatement> delete this statement
- the <CommaStatement> comma statement

8.31 ASCQM Release in Destructor Memory Allocated in Constructor

Descriptor
ASCQM Release in Destructor Memory Allocated in Constructor (MemoryAllocationStatement)

Description
Identify occurrences in application model where:
- the <MemoryAllocationStatement> memory allocation statement in the class constructor
- lacking a corresponding memory release statement in the class destructor

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit|IntegerType|DecimalType|FloatType|StringType|VoidType|... id="dt1"
PointerType id="pt1"
    ItemUnit id="iu1" type="dt1"
...
ClassUnit id="cu1"
    ...
    StorableUnit id="su1" type="pt1"
    ...
    MethodUnit id="mu1" MethodKind="constructor"
        ...
        ActionElement id="ae1" kind="New|NewArray"
            Creates "dt1"
            Writes "su1"
...
```

or

```
ControlElement id="ce1" name="malloc|calloc|..."
...
ClassUnit|IntegerType|DecimalType|FloatType|StringType|VoidType|... id="dt1"
PointerType id="pt1"
    ItemUnit id="iu1" type="dt1"
...
ClassUnit id="cu1"
    ...
    StorableUnit id="su1" type="pt1"
    ...
    MethodUnit id="mu1" MethodKind="constructor"
        ...
        ActionElement id="ae1" kind="Call"
            Calls "ce1"
            Writes "su1"
...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce2" name="delete|delete[]|free|..."
...
ClassUnit id="cu1"
    ...
    MethodUnit id="mu2" MethodKind="destructor"
        ...
        ActionElement id="ae2" kind="Call"
             Addresses "su1"
             Calls "ce2"
```

What to report
Roles to report:
- the <MemoryAllocationStatement> memory allocation statement

8.32 ASCQM Release Memory after Use with Correct Operation

Descriptor
ASCQM Release Memory after Use with Correct Operation (MemoryAllocationStatement, MemoryReleaseStatement)

Description
Identify occurrences in the application model where:
- the memory is allocated via the <MemoryAllocationStatement> allocation statement
- then released via the mismatched <MemoryReleaseStatement> release statement

The pairs of matching allocation/deallocation primitives and operations are technology, framework, language dependant. For example: malloc/free, calloc/free, realloc/free in C/C+, new/delete, new[]/delete[] in C+, new/Release() with COM IUnknown interface.

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit|IntegerType|DecimalType|FloatType|StringType|VoidType|... id="dt1"
PointerType id="pt1"
    ItemUnit id="iu1" type="dt1"
...
StorableUnit id="su1" type="pt1"
...
ActionElement id="ae1" kind="New"
    Creates "dt1"
    Writes "su1"
...
ControlElement id="ce2" name="delete[]|free|..."
...
ActionElement id="ae2" kind="Call"
    Addresses "su1"
    Calls "ce2"
```

or

```
ClassUnit|IntegerType|DecimalType|FloatType|StringType|VoidType|... id="dt1"
PointerType id="pt1"
    ItemUnit id="iu1" type="dt1"
...
StorableUnit id="su1" type="pt1"
...
ActionElement id="ae1" kind="NewArray"
    Creates "dt1"
    Writes "su1"
...
ControlElement id="ce2" name="delete|free|..."
...
ActionElement id="ae2" kind="Call"
    Addresses "su1"
    Calls "ce2"
```

or

```
ControlElement id="ce1" name="malloc|calloc|..."
...
ClassUnit|IntegerType|DecimalType|FloatType|StringType|VoidType|... id="dt1"
PointerType id="pt1"
    ItemUnit id="iu1" type="dt1"
...
StorableUnit id="su1" type="pt1"
...
ActionElement id="ae1" kind="Call"
    Calls "ce1"
    Writes "su1"
...
ControlElement id="ce2" name="delete|delete[]|..."
...
ActionElement id="ae2" kind="Call"
    Addresses "su1"
    Calls "ce2"
```

What to report

Roles to report are:
- the <MemoryAllocationStatement> allocation statement
- the <MemoryReleaseStatement> release statement

8.33 ASCQM Implement Required Operations for Manual Resource Management

Descriptor

ASCQM Implement Required Operations for Manual Resource Management (ObjectDeclaration)

Description

Identify occurrences in application model where:
- the <ObjectDeclaration> object declaration
- declares an object with manual resource management capabilities
- which lacks the required operation.

The manual resource management capability is technology, framework, and language dependent. For example: class inheritance from IDisposable in C#, and AutoClosable in Java, class with __enter__ in python.

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
InterfaceUnit id="iu1" name="IDisposable|AutoClosable|..."
...
ClassUnit id="cu1"
     Extends "iu1"
  …
```

of

```
...
ClassUnit id="cu1"
     MethodUnit "mu1" name="__enter__"
```

```
    ...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="cu1"
    ...
    MethodUnit "mu1" name="dispose|close|__exit__|..."
```

What to report
Roles to report:
- the <ObjectDeclaration> object declaration

8.34 ASCQM Release Platform Resource after Use

Descriptor
ASCQM Release Platform Resource after Use (FunctionProcedureOrMethod, ResourceAllocationStatement, PathToExitWithoutResourceRelease)

Description
Identify occurrences in application model where:
- the <FunctionProcedureOrMethod> function, procedure, method, ...
- uses the <ResourceAllocationStatement> resource allocation statement
- excluding memory and file resources
- while there exist the <PathToExitWithoutResourceRelease> path to exit the <FunctionProcedureOrMethod> function,
  procedure, method, ... without releasing the resource

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    ...
    DataManager|ExecutionResource id="pr1"
    ...
    PlatformAction id="pa1" kind="open" implementation="ae1"
        ManagesResource "pr1"
    PlatformAction id="pa2" kind="close" implementation="ae2"
        ManagesResource "pr1"

...
CodeModel
    ...
    CallableUnit|MethodUnit id="ce1"  name="..."
        ...
        ActionElement id="ae1" kind="PlatformAction"
            Flows "ae3"
        ActionElement id="ae3"
             Flows "ae4"
        ActionElement id="ae4" kind="Return"
        ...
        ActionElement id="ae2" kind="PlatformAction"
        ...
...
```

What to report
Roles to report
- the <FunctionProcedureOrMethod> function, procedure, method, ...
- the <ResourceAllocationStatement> file resource open statement
- the <PathToExitWithoutResourceRelease> path to exit

8.35 ASCQM Release Memory After Use

Descriptor
ASCQM Release Memory After Use (MemoryAllocationStatement)

Description
Identify occurrences in application model where :
- the <MemoryAllocationStatement> memory allocation statement
- lacking a corresponding memory release statement

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit|IntegerType|DecimalType|FloatType|StringType|VoidType|... id="dt1"
PointerType id="pt1"
    ItemUnit id="iu1" type="dt1"
...
StorableUnit id="su1" type="pt1"
...
ActionElement id="ae1" kind="New|NewArray"
    Creates "dt1"
    Writes "su1"
...
```

or

```
ControlElement id="ce1" name="malloc|calloc|..."
...
ClassUnit|IntegerType|DecimalType|FloatType|StringType|VoidType|... id="dt1"
PointerType id="pt1"
    ItemUnit id="iu1" type="dt1"
...
StorableUnit id="su1" type="pt1"
...
ActionElement id="ae1" kind="Call"
    Calls "ce1"
    Writes "su1"
...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce2" name="delete|delete[]|free|..."
...
ActionElement id="ae2" kind="Call"
    Addresses "su1"
    Calls "ce2"
```

What to report
Roles to report :
- the <MemoryAllocationStatement> memory allocation statement

8.36 ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor

Descriptor
ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor (Class, ParentClass, ParentVirtualDestructor)

Description
Identify occurrences in application model where :
- the <Class> class
- inherits from the <ParentClass> parent class
- with the <ParentVirtualDestructor> virtual destructor
- but lacks a virtual destructor

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="c1"
    ....
    MethodUnit is="m1"  methodKind="method" isVirtual="true"
        ...
ClassUnit id="c2"  InheritsFrom="c1"
    ...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="c2"
    ....
    MethodUnit is="m2" methodKind="destructor" isVirtual="true"
        ...
```

What to report
Roles to report are :
- the <Class> class
- the <ParentClass> parent class
- the <ParentVirtualDestructor> virtual destructor

8.37 ASCQM Implement Virtual Destructor for Parent Classes

Descriptor
ASCQM Implement Virtual Destructor for Parent Classes (Class, ParentClass)

Description
Identify occurrences in application model where:
- the <Class> class
- inherits from the <ParentClass> parent class
- which lacks a virtual destructor

KDM outline illustration
KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="c1"
    ....
ClassUnit id="c2"  InheritsFrom="c1"
    ...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="c1"
    ....
    MethodUnit is="m1"  methodKind="method" isVirtual="true"
        ...
```

What to report
Roles to report are:
- the <Class> class
- the <ParentClass> parent class

8.38 ASCQM Release File Resource after Use in Operation

Descriptor
ASCQM Release File Resource after Use in Operation (FunctionProcedureOrMethod, FileResourceOpenStatement, PathToExitWithoutFileResourceClose)

Description
Identify occurrences in application model where:
- the <FunctionProcedureOrMethod> function, procedure, method, ...
- uses the <FileResourceOpenStatement> file resource open statement
- while there exist the <PathToExitWithoutFileResourceClose> path to exit the <FunctionProcedureOrMethod> function,
  procedure, method, ... without releasing the file resource

The path to exit the function, procedure, method, includes calls to other functions, procedures, methods, ...

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    ...
    FileResource id="pr1"
    ...
    PlatformAction id="pa1" kind="open" implementation="ae1"
        ManagesResource "pr1"
    PlatformAction id="pa2" kind="close" implementation="ae2"
        ManagesResource "pr1"

...
CodeModel
    ...
    CallableUnit|MethodUnit id="ce1"  name="..."
        ...
        ActionElement id="ae1" kind="PlatformAction"
            Flows "ae3"
        ActionElement id="ae3"
             Flows "ae4"
        ActionElement id="ae4" kind="Return"
        ...
        ActionElement id="ae2" kind="PlatformAction"
        ...
...
```

What to report

Roles to report:
- the <FunctionProcedureOrMethod> function, procedure, method, ...
- the <FileResourceOpenStatement> file resource open statement
- the <PathToExitWithoutFileResourceClose> path to exit

8.39 ASCQM Implement Virtual Destructor for Classes with Virtual Methods

Descriptor

ASCQM Implement Virtual Destructor for Classes with Virtual Methods (Class, VirtualMethod)

Description

Identify occurrences in application model where:
- the <Class> class
- owns the <VirtualMethod> virtual method
- but lacks a virtual destructor

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="c1"
    ....
    MethodUnit is="m1" methodKind="method" isVirtual="true"
        ...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="c1"
    ....
    MethodUnit is="m2" methodKind="destructor" isVirtual="true"
        ...
```

What to report
Roles to report are:
- the <Class> class
- the <VirtualMethod> virtual method

8.40 ASCQM Ban Self Destruction

Descriptor
ASCQM Ban Self Destruction (DeleteThisStatement)

Description
Identify occurrences in application model where:
- the <DeleteThisStatement> delete this statement

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
CallableUnit id="cu1" name="delete" callableKind="operator"
...
ClassUnit id="cu1"
    ...
    StorableUnit id="su1"
    ...
    ActionElement id="ae1" kind="This"
        Writes "su1"
    ...
    ActionElement id="ae2" kind="Call"
        Addresses "su1"
        Calls "cu1"
```

What to report
Roles to report:
- the <DeleteThisStatement> delete this statement

8.41 ASCQM Manage Time-Out Mechanisms in Blocking Synchronous Calls

Descriptor
ASCQM Manage Time-Out Mechanisms in Blocking Synchronous Calls (BlockingSynchronousCall, TimeOutOption)

Description
Identify occurrences in application model where:
- the <BlockingSynchronousCall> synchronous call
- does not use its <TimeOutOption> time-out option

The list of blocking synchronous primitives is technology, framework, language dependent. For example, in Java: connect(), receive().

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce1" name="connect|receive|..." type="ce1_signature"
    Signature id="ce1_signature"
        ...
        ParameterUnit id="pu1" name="timeout|..."
    ...
Value id="v1" attribute="infinite_wait"
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
    Calls "ce1"
    Reads "v1"
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
...
Value id="v2" attribute="finite_wait"
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
    Calls "ce1"
    Reads "v2"
```

What to report
Roles to report:
- the <BlockingSynchronousCall> synchronous call
- the <TimeOutOption> time-out option

8.42 ASCQM Ban Non-Final Static Data in Multi-Threaded Context

Descriptor
ASCQM Ban Non-Final Static Data in Multi-Threaded Context (Declaration)

Description
Identify occurrences in application model where:
- the <Declaration> declaration of non-final static data
- in multi-threaded environment

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
CodeModel
    StorableUnit id="su1" isFinal="false" isStatic="true"
...
PlatformModel
    DeployedResource id="dr1"
        ExecutionResource id="er1"
            Thread id="t1"
            Thread id="t2"
    ...
```

What to report

Roles to report are:
- the <Declaration> declaration of non-final static data

8.43 ASCQM Ban Non-Serializable Elements in Serializable Objects

Descriptor

ASCQM Ban Non-Serializable Elements in Serializable Objects (SerializableClass, NonSerializableMember)

Description

Identify occurrences in application model where:
- the <SerializableClass> serializable class
- owns the <NonSerializableMember> non-serializable member, excluding final and transient members and members of
  primitive types
- without owning custom serialization / deserialization methods

The serializable nature of the element is technology dependent For example.: serializable nature comes from a serializable SerializableAttribute attribute or the inheritance from System.RuntimeSerialization.ISerializable in .NET, and the inheritance from the java.io.Serializable interface in Java.

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
Package id="java.io" name="java.io" | Namespace id="System.Runtime.Serialization"
name="System.Runtime.Serialization"
    InterfaceUnit id="iu1" name="Serializable|ISerializable"
ClassUnit id="cu1"
ClassUnit id="cu2"  Implements="iu1" | attribute="Serializable"
ClassUnit id="cu3"  Implements="iu1" | Extends="cu2" | attribute="Serializable"
    MemberUnit id="mu1" type="cu1"
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="cu1"  Implements="iu1" | Extends="cu2" | attribute="Serializable"
```

or

```
ClassUnit id="cu3" Implements="iu1" | Extends="cu2" | attribute="Serializable"
    MemberUnit id="mu1" type="cu1" storableKind="static"
```

or

```
ClassUnit id="cu3" Implements="iu1" | Extends="cu2" | attribute="Serializable"
    MemberUnit id="mu1" type="cu1" attribute="transient|NonSerialized"
```

or

```
ClassUnit id="cu3" Implements="iu1" | Extends="cu2"
    MethodUnit id="mu1" name="readObject" kind="method"
    MethodUnit id="mu2" name="readObjectNoData" kind="method"
    MethodUnit id="mu3" name="writeObject" kind="method"
```

or

```
ClassUnit id="cu3" Implements="iu1" | Extends="cu2"
    MethodUnit id="mu1" name="GetObjectData" kind="method"
    MethodUnit id="mu2" name="cu2" kind=constructor" type="mu2_signature"
        Signature id="mu2_Signature"
            ParameterUnit id="p1" name="info" type="SerializationInfo"
            ParameterUnit id="p2" name="context" type="StreamingContext"
```

What to report

Roles to report:
- the <SerializableClass> serializable class
- the <NonSerializableMember> non-serializable member

8.44 ASCQM Ban Hard-Coded Literals used to Connect to Resource

Descriptor

ASCQM Ban Hard-Coded Literals used to Connect to Resource (InitializationStatement, ResourceAccessStatement)

Description

Identify occurrences in application model where:
- the <InitializationStatement> initialization statement
- initialize a variable used in the <ResourceAccessStatement> resource access statement as parameter to call a resource
  access primitive

It covers credentials, passwords, encryption keys, tokens, remember-me keys...

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
Value id="hcv" name="hcv"
...
StorableUnit|ItemUnit|MemberUnit id="su1"
...
ActionElement id="ae1" kind="Assign
    Reads "hcv"
```

```
    Writes "su1"
...
MarshalledResource|MessagingResource|DataManager|ExecutionResource id="nwr"
...
ControlElement id="ce1"
    ...
    ActionELement id="ae2" kind="Platform"
        ManagesResource|ReadsResource|WritesResource "nwr"
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
    Reads "su1"
    ...
    Calls "ce1"
```

What to report

Roles to report are:
- the <InitializationStatement> initialization statement
- the <ResourceAccessStatement> resource access statement

8.45 ASCQM Ban Unintended Paths

Descriptor

ASCQM Ban Unintended Paths (ArchitectureModel, Relation, Caller, Callee, OriginModule, TargetModule)

Description

Identify occurrences in the application model where:
- the <Relation> call-type, data, use relations
- between the <Caller> caller
- grouped in the <OriginModule> origin layer, component, or subsystem
- and the <Callee> callee
- grouped into the <TargetModule> target layer, component, or subsystem
- as defined in the <ArchitectureModel> architectural blueprint defining layers, components, or subsystems
- where relations from the <OriginModule> layer, component, or subsystem to the <TargetModule> layer, component, or
  subsystem are not intended

The architectural blueprint defining layers, components, or subsystems is application dependent.

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
...
Layer|Component|Subsystem id="m1"
    ...
    CallableUnit callableKind="regular|external|stored" | MethodUnit id="ce1" name="..."
        ...
        ActionElement id="ae1"
            UsesType|Reads|Writes|Creates|Addresses|Calls|Dispatches "ce2"
...
Layer|Component|Subsystem id="m2"
    ...
    CallableUnit callableKind="regular|external|stored" | MethodUnit id="ce2" name="..."
...
```

With "m1" not intended to reference "m2"

What to report
Roles to report are:
- the <ArchitectureModel> architectural blueprint
- the <Relation> relation
- the <Caller> caller
- the <Callee> callee
- the <OriginModule> origin layer, component, or subsystem
- the <TargetModule> target layer, component, or subsystem

8.46 ASCQM Ban Incorrect Float Number Comparison

Descriptor
ASCQM Ban Incorrect Float Number Comparison (FloatEqualityComparisonExpression)

Description
Identify occurrences in application model where
- the <FloatEqualityComparisonExpression> equality comparison expression
- between two float numbers

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
FloatType id="ft1"
StorableUnit|ItemUnit|MemberUnit id="de1" type="ft1"
StorableUnit|ItemUnit|MemberUnit id="de2" type="ft1"
ActionElement id="ae1" kind="Equals|NotEqual" ext="de1 == de2 | de1 != de2"
    Reads "de1"
    Reads "de2"
```

What to report
Roles to report are:
- the <FloatEqualityComparisonExpression> equality comparison expression

8.47 ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context

Descriptor
ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context (SingletonClass, InitializationStatement)

Description
Identify occurrences in application model where:
- the <SingletonClass> singleton class
- with the <InitializationStatement> self-reference initialization statement
- not properly locked
- while it operates in a multi-threaded environment

The proper locking is technology, framework, and language dependent.
The detection of multi-threading capability is technology, framework, and language dependent.

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ExecutionResource id="er1"
            Thread id="t1"
        ...
        PlatformAction id="pa1" implementation="ae1"
            ManagesResource "t1"
    ...
...
CodeModel
    ActionElement id="ae1"
    ...
    ClassUnit id="singleton" exportKind="public"
        MemberUnit id="reference" isStatic="true" exportKind="private" type="singleton"
        MethodUnit id="c" kind="constructor" exportKind="private" type="c_signature"
            Signature
                ParameterUnit id="r1" kind="return" type="singleton"
        ...
        MethodUnit id="refget" kind="method" storableKind="static" exportKind="public"
type="refget_signature"
            Signature id="refget_signature"
                ParameterUnit id="r2" kind="return" type="singleton"
            ActionElement id="a2" name="a2" kind="Return"
                Writes "r2"
                Reads "reference"
        ...
...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ...
        LockResource id="lr1"
        ...
        PlatformAction id="pa2" kind="lock" implementation="ae3"
            ManagesResource|ReadsResource|WritesResource "lr1"
        PlatformAction id="pa3" kind="unlock" implementation="ae5"
            ManagesResource|ReadsResource|WritesResource "lr1"
...
CodeModel
    ClassUnit id="singleton" exportKind="public"
        ...
            ActionElement id="ae2" kind="Compound"
                EntryFlow "ae3"
                ActionElement id="ae3" kind="PlatformAction"
                    Flows "ae4"
                ActionElemeent id="ae4"
                    Writes "reference"
                    Flows "ae5"
                ActionElement id="ae5" kind="PlatformAction"
...
```

What to report
Roles to report are:
- the <SingletonClass> singleton class
- the <InitializationStatement> initialization statement

8.48 ASCQM Ban Incorrect Numeric Implicit Conversion

Descriptor
ASCQM Ban Incorrect Numeric Implicit Conversion (Variable, VariableDataType, VariableAssignmentStatement, Data, TargetDataType)

Description
Identify occurrences in application model where:
- the <Variable> variable is declared with the <VariableDataType> numerical data type
- then updated is the <VariableAssignmentStatement> assignment statement
- with the <Data> data of the <TargetDataType> second numerical data type
- which is incompatible with the first one
- and without any range check or explicit casting

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
IntegerType|DecimalType|FloatType id="dt1"
StorableUnit|ItemUnit|MemberUnit id="de1" type="dt1"
IntegerType|DecimalType|FloatType id="dt2"
StorableUnit|ItemUnit|MemberUnit|Value id="de2" type="dt2"
ActionElement id="ae1" kind="Assign"
    Writes "de1"
    Reads "de2"
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ActionElement id="ae2" kind="LessThan|LessThanOrEqual"
    Reads "de2"
ActionElement id="ae3" kind="GreaterThan|GreaterThanOrEqual"
    Reads "de2"
```

or

```
ActionElement id="ae1" kind="TypeCast"
    Reads "de2"
    UsesType "dt1"
    Writes "de1"
```

and the numeric datatypes are not compatible.
Compatibility comes from storage size and primary types. For example.: char and int8, wchar and int16, 64-bit pointers and 64-bits long integers, ...

What to report

Roles to report are:
- the <Variable> variable
- the <VariableDataType> numerical data type
- the <VariableAssignmentStatement> assignment statement
- the <Data> data
- the <TargetDataType> second numerical data type

8.49 ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context

Descriptor

ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context (InitializationStatement)

Description

Identify occurrences in application model where:
- the <WriteOrReadStatement> write or read statement
- of variable with the <NonAtomicDataType> non-atomic data type
- is not properly locked,
- while it operates in a multi-threaded environment

The proper locking is technology, framework, and language dependent.
The detection of multi-threading capability is technology, framework, and language dependent.
The list of non-atomic data types is technology, framework, and language dependent.

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ExecutionResource id="er1"
            Thread id="t1"
        ...
        PlatformAction id="pa1" implementation="ae1"
            ManagesResource "t1"
    ...
...
CodeModel
    ActionElement id="ae1"
    ...
    DataType id="dt1" isAtomic="false"
    StorableUnit id="su1" type="dt1"
    ...
    ActionElement id="ae4" kind="Assign|Select|..."
        Reads|Writes "su1"
...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ...
        LockResource id="lr1"
        ...
        PlatformAction id="pa2" kind="lock" implementation="ae3"
            ManagesResource|ReadsResource|WritesResource "lr1"
        PlatformAction id="pa3" kind="unlock" implementation="ae5"
            ManagesResource|ReadsResource|WritesResource "lr1"
...
CodeModel
    ...
    ActionElement id="ae2" kind="Compound"
        EntryFlow "ae3"
    ActionElement id="ae3" kind="PlatformAction"
        Flows "ae4"
    ActionElement id="ae4" kind="Assign|Select|..."
        Reads|Writes "su1"
        Flows "ae5"
    ActionElement id="ae5" kind="PlatformAction"
...
```

What to report

Roles to report are:
- the <InitializationStatement> initialization statement

8.50 ASCQM Ban Incorrect Synchronization Mechanisms

Descriptor

ASCQM Ban Incorrect Synchronization Mechanisms (IncorrectSynchronizationPrimitiveCall)

Description

Identify occurrences in application model where:
- the <IncorrectSynchronizationPrimitiveCall> call to incorrect synchronization primitive
- while it operates in a multi-threaded environment

The list of incorrect synchronization primitives is technology, framework, language dependent. For example.: java.lang.Thread.run() in Java; getlogin() in C; synchronization primitives with EJBs.
The detection of multi-threading capability is technology, framework, and language dependent.

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
CodeModel
    ControlElement id="ce1" name="run|getlogin|..."
        ...
    ...
    ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
        ...
        Calls "ce1"
...
PlatformModel
    DeployedResource id="dr1"
        ExecutionResource id="er1"
```

```
            Thread id="t1"
            Thread id="t2"
    ...
```

What to report

Roles to report are:
- the <IncorrectSynchronizationPrimitiveCall> call to incorrect synchronization primitive

8.51 ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context

Descriptor

ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context (ResourceAccessStatement)

Description

Identify occurrences in application model where:
- the <ResourceAccessStatement> access statement to a resource
- not properly locked
- while it operates in a multi-threaded environment

The proper locking is technology, framework, and language dependent.
The detection of multi-threading capability is technology, framework, and language dependent.

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ExecutionResource id="er1"
            Thread id="t1"
        ...
        PlatformAction id="pa1" implementation="ae1"
            ManagesResource "t1"
        ...
        StreamResource|FileResource|... id="pr1"
        ...
        PlatformAction id="pa2" implementation="ae2"
            ManagesResource|ReadsResource|WritesResource "pr1"

    ...
...
CodeModel
    ActionElement id="ae1" kind="PlatformAction"
    ...
    ActionElement id="ae2" kind="PlatformAction"
    ...
...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ...
        LockResource id="lr1"
```

```
        ...
        PlatformAction id="pa2" kind="lock" implementation="ae4"
            ManagesResource|ReadsResource|WritesResource "lr1"
        PlatformAction id="pa3" kind="unlock" implementation="ae5"
            ManagesResource|ReadsResource|WritesResource "lr1"
...
CodeModel
    ClassUnit id="singleton" exportKind="public"
        ...
            ActionElement id="ae3" kind="Compound"
                EntryFlow "ae4"
                ActionElement id="ae4" kind="PlatformAction"
                    Flows "ae2"
                ActionElement id="ae2"
                    Flows "ae5"
                ActionElement id="ae5" kind="PlatformAction"
...
```

What to report

Roles to report are:
- the <ResourceAccessStatement> access statement to a resource

8.52 ASCQM Ban Incorrect Type Conversion

Descriptor

ASCQM Ban Incorrect Type Conversion (Variable, VariableDataType, VariableAssignmentStatement, Data, TargetDataType)

Description

Identify occurrences in application model where:
- the <Variable> variable is declared with the <VariableDataType> non-numerical data type
- then updated is the <VariableAssignmentStatement> assignment statement
- with the <Data> data is of the <TargetDataType> second non-numerical data type
- which is incompatible with the first one

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
StringType|ClassUnit|... id="dt1"
StorableUnit|ItemUnit|MemberUnit id="de1" type="dt1"
StringType|ClassUnit|... id="dt2"
StorableUnit|ItemUnit|MemberUnit|Value id="de2" type="dt2"
ActionElement id="ae1" kind="Assign"
    Writes "de1"
    Reads "de2"
```

or

```
StringType|ClassUnit|... id="dt1"
PointerType id="pt1"
StorableUnit|ItemUnit|MemberUnit id="de1" type="pt1"
StringType|ClassUnit|... id="dt2"
PointerType id="pt2"
ActionElement id="ae1" kind="TypeCast"
    Reads "de1"
    UsesType "pt2"
```

Where the non-numeric datatypes are not compatible.
Compatibility comes from inheritence links between objects, and, when numeric types are concerned, from storage size and primary types. For example: char and int8, wchar and int16, 64-bit pointers and 64-bits long integers, ...

What to report
Roles to report are
- the <Variable> variable
- the <VariableDataType> data type
- the <VariableAssignmentStatement> assignment statement
- the <Data> data
- the <TargetDataType> second data type

8.53 ASCQM Ban Return of Local Variable Address

Descriptor
ASCQM Ban Return of Local Variable Address (LocalVariable, Operation)

Description
Identify occurrences in application model where:
- the address of the <LocalVariable> local variable
- is returned by the <Operation> operation

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
...
PointerType id="pt1"
CallableUnit callableKind="regular|external|stored" | MethodUnit id="ce1" name="..."
type="ce1_signature"
    Signature id="ce1_signature"
        ...
        ParameterUnit id="pu1" kind="return" type="pt1"
    ...
    StorableUnit id="su1" kind="register"
    StorableUnit id="su2" kind="local"
    ActionElement id="ae1" kind="Ptr"
        Writes "su1"
        Addresses "su2"
    ActionElement id="ae2" kind="Return"
        Reads "su1"
...
```

What to report
Roles to report are:
- the <LocalVariable> local variable address
- the <Operation> operation

8.54 ASCQM Ban Storage of Local Variable Address in Global Variable

Descriptor
ASCQM Ban Storage of Local Variable Address in Global Variable (LocalVariable, StorageStatement, GlobalVariable)

Description
Identify occurrences in application model where:
- the address of the <LocalVariable> local variable
- is stored by the <StorageStatement> statement
- into the <GlobalVariable> global variable

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
StorableUnit id="su1" kind="global"
...
CallableUnit callableKind="regular|external|stored" | MethodUnit id="ce1"
    ...
    StorableUnit id="su2" kind="register"
    StorableUnit id="su3" kind="local"
    ActionElement id="ae1" kind="Ptr"
        Writes "su2"
        Addresses "su3"
    ActionElement id="ae2" kind="Assign"
        Reads "su2"
        Writes "su3"
...
```

What to report
Roles to report are:
- the <LocalVariable> local variable address
- the <StorageStatement> statement
- the <GlobalVariable> global variable

8.55 ASCQM Ban While TRUE Loop Without Path to Break

Descriptor
ASCQM Ban While TRUE Loop Without Path To Break (WhileTrueLoop)

Description
Identify occurrences in the application model where:
- the <WhileTrueLoop> "while true" loop
- lacks a control flow to a break statement out of the loop

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
BooleanType id="booleanType"
Value id="true" name="true" type="booleanType"
ActionElement id="ae1" kind="Compound"
    ActionElement id="ae2" kind="Condition"
```

```
        Reads "true"
        TrueFlow "tf1"
        FalseFlow "ff1"
    ActionElement id="tf1" ...
    ...
    Flows "ae2"
ActionElement id="ff1" ...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ActionElement id="ae1" kind="Compound"
    ActionElement id="ae2" kind="Condition"
        ...
        TrueFlow "tf1"
        ...
        ActionElement id="tf1" ...
            Flows "ae3"
        ActionElement id="ae3"
            Flows "e1"
        ActionElement id="e1" kind="Goto"
            Flows "ff1"
    ...
ActionElement id="ff1" ...
```

What to report

Roles to report:

- the <WhileTrueLoop> "while true" loop

8.56 ASCQM Ban Unmodified Loop Variable Within Loop

Descriptor

ASCQM Ban Unmodified Loop Variable Within Loop (WhileLoop)

Description

Identify occurrences in the application model where:

- the <WhileLoop> while loop
- lacks an update of the condition value within the loop

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
BooleanType id="booleanType"
StorableUnit id="su1"  type="booleanType"
ActionElement id="ae1" kind="Compound"
    ...
    ActionElement id="ae2" kind="Condition"
        Reads "su1"
        ...
    ...
...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ActionElement id="ae1" kind="Compound"
    ...
    ActionElement id="ae3" kind="Assign|Incr|Decr"
        Writes "su1"
        ...
    ...
```

What to report
Roles to report:
- the <WhileLoop> while loop

8.57 ASCQM Check and Handle ZERO Value before Use as Divisor

Descriptor
ASCQM Check and Handle ZERO Value before Use as Divisor (DivisionStatement)

Description
Identify occurrences in application model where:
- the <DivisionStatement> division statement
- uses a variable which is not checked and handled before use as divisor immediately before

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
StorableUnit id="su1"
StorableUnit id="su2"
ActionElement id="ae3" kind="Divide"
    Reads "su1"
    Reads "su2"
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
...
Value id="v1" name="0"
StorableUnit id="su3"
ActionElement id="ae1" kind="NotEqual"
    Reads "v1"
    Reads "su2"
    Writes "su3"
    Flows "ae2"
ActionElement id="ae2" kind="Condition"
    Reads "su3"
    TrueFlow "ae3"
    FalseFlow "ff1"
...
```

What to report
Roles to report are:
- the <DivisionStatement> division statement

8.58 ASCQM Ban Creation of Lock on Private Non-Static Object to Access Private Static Data

Descriptor

ASCQM Ban Creation of Lock On Private Non-Static Object to Access Private Static Data (PrivateNonStaticLock, DataAccess, PrivateStaticData)

Description

Identify occurrences in application model where:
- the <PrivateNonStaticLock> private non-static lock object
- is used to lock a block including the <DataAccess> data access
- to the <PrivateStaticData> private static data

The locking mechanism is technology, framework, language dependent.

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ...
        LockResource id="lr1"
        ...
        PlatformAction id="pa2" kind="lock" implementation="ae1"
            ManagesResource|ReadsResource|WritesResource "lr1"
...
CodeModel
    ...
    StorableUnit id="su1" isStatic="false" exportKind="private"
    StorableUnit id="su2" isStatic="true" exportKind="private"
    ...
    ActionElement id="ae1" kind="PlatformAction"
        Reads "su1"
        Flows "ae2"
    ActionElement id="ae2"
        Flows "ae3"
    ActionElement id="ae3" kind="Assign|PtrReplace|ArrayReplace|PtrSelect|ArraySelect|..."
        Reads|Writes "su2"
        ...
    ...
```

What to report

Roles to report:
- the <PrivateNonStaticLock> private non-static lock object
- the <DataAccess> data access
- the <PrivateStaticData> private static data

8.59 ASCQM Release Lock After Use

Descriptor

ASCQM Release Lock After Use (FunctionProcedureOrMethod, LockAcquisitionStatement, PathToExitWithoutLockRelease)

Description

Identify occurrences in application model where:
- the <FunctionProcedureOrMethod> function, procedure, method, ...
- uses the <LockAcquisitionStatement> lock acquisition statement
- while there exist the <PathToExitWithoutLockRelease> path to exit the <FunctionProcedureOrMethod> function,
  procedure, method, ... without releasing the lock resource

The path to exit the function, procedure, method, includes calls to other functions, procedures, methods, ...
The locking mechanism is technology, framework, and language dependent.

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ...
        LockResource id="lr1"
        ...
        PlatformAction id="pa2" kind="lock" implementation="ae1"
            ManagesResource|ReadsResource|WritesResource "lr1"
        PlatformAction id="pa3" kind="unlock" implementation="ae2"
            ManagesResource|ReadsResource|WritesResource "lr1"
...
CodeModel
    ...
    CallableUnit|MethodUnit id="ce1"  name="..."
        ...
        ActionElement id="ae1" kind="PlatformAction"
            Flows "ae3"
        ActionElement id="ae3"
             Flows "ae4"
        ActionElement id="ae4" kind="Return"
        ...
        ActionElement id="ae2" kind="PlatformAction"
        ...
...
```

What to report

Roles to report:
- the <FunctionProcedureOrMethod> function, procedure, method, ...
- the <LockAcquisitionStatement> lock acquisition statement
- the <PathToExitWithoutLockRelease> path to exit

8.60 ASCQM Ban Sleep Between Lock Acquisition and Release

Descriptor

ASCQM Ban Sleep Between Lock Acquisition and Release (PathFromLockAcquisitionToLockRelease, LockAcquisitionStatement, LockReleaseStatement, SleepStatement)

Description

Identify occurrences in application model where:
- the <PathFromLockAcquisitionToLockRelease> path
- from the <LockAcquisitionStatement> lock acquisition statement
- to the <LockReleaseStatement> lock release statement
- contains the <SleepStatement> sleep statement

The path includes calls to other functions, procedures, methods, ...
The locking mechanism is technology, framework, and language dependent.

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ...
        LockResource id="lr1"
        ...
        PlatformAction id="pa2" kind="lock" implementation="ae1"
            ManagesResource|ReadsResource|WritesResource "lr1"
        PlatformAction id="pa3" kind="unlock" implementation="ae5"
            ManagesResource|ReadsResource|WritesResource "lr1"
        ...
    ExecutionResource id="er1"
        ...
        Thread id="t1"
        ...
        PlatformAction id="pa3" kind="sleep" implementation="ae3"
            ManagesResource "t1"
...
CodeModel
    ...
    CallableUnit|MethodUnit id="ce1"  name="..."
        ...
        ActionElement id="ae1" kind="PlatformAction"
            Flows "ae2"
        ActionElement id="ae2"
             Flows "ae3"
        ActionElement id="ae3" kind="PlatformAction"
             Flows "ae4"
        ActionElement id="ae4"
             Flows "ae5"
        ActionElement id="ae5" kind="PlatformAction"
        ...
...
```

What to report

Roles to report:
- the <PathFromLockAcquisitionToLockRelease> path
- the <LockAcquisitionStatement> lock acquisition statement
- the <LockReleaseStatement> lock release statement
- the <SleepStatement> sleep statement

8.61 ASCQM Ban Creation of Lock on Non-Final Object

Descriptor
ASCQM Ban Creation of Lock On Non-Final Object (NonFinalObjectDeclaration, LockingAcquisitionStatement)

Description
Identify occurrences in application model where:
- the <NonFinalObjectDeclaration> non-final object declaration
- declares an object used as a lock in the <LockingAcquisitionStatement> locking acquisition statement

The locking mechanism is technology, framework, language dependent.

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ...
        LockResource id="lr1"
        ...
        PlatformAction id="pa2" kind="lock" implementation="ae1"
            ManagesResource|ReadsResource|WritesResource "lr1"
...
CodeModel
    ...
    StorableUnit id="su1" isFinal="false"
    ...
    ActionElement id="ae1" kind="PlatformAction"
        Reads "su1"
    ...
```

What to report
Roles to report:
- the <NonFinalObjectDeclaration> non-final object declaration
- the <LockingAcquisitionStatement> locking acquisition statement

8.62 ASCQM Ban Creation of Lock on Inappropriate Object Type

Descriptor
ASCQM Ban Creation of Lock On Inappropriate Object Type (ObjectDeclaration, LockingAcquisitionStatement)

Description
Identify occurrences in application model where:
- the <ObjectDeclaration> object declaration
- declares an object used as a lock in the <LockingAcquisitionStatement> locking acquisition statement
- while its type is not suitable for locking

The list of proper locking object types is technology, framework, language dependent. For example, in C# and Java: Reference Types, excluding Boxed Types, Strings

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ...
        LockResource id="lr1"
        ...
        PlatformAction id="pa2" kind="lock" implementation="ae1"
            ManagesResource|ReadsResource|WritesResource "lr1"
...
CodeModel
    ...
    StorableUnit id="su1"
    ...
    ActionElement id="ae1" kind="PlatformAction"
        Reads "su1"
    ...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
CodeModel
    ...
    ClassUnit|InterfaceUnit|... id="dt1"
    StorableUnit id="su1" type="dt1"
    ...
```

What to report

Roles to report:
- the <ObjectDeclaration> object declaration
- the <LockingAcquisitionStatement> locking acquisition statement

8.63 ASCQM NULL Terminate Output of String Manipulation Primitives

Descriptor

ASCQM NULL Terminate Output Of String Manipulation Primitives (StringManipulationCallStatement)

Description

Identify occurrences in application model where:
- the <StringManipulationCallStatement> string manipulation call statement
- is not immediately followed by adding a NULL termination to the resulting string

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
StringType id="string"
StorableUnit id="su1" type="string"
...
ControlElement id="ce1" type="ce1_signature"
    Signature id="ce1_signature"
         ParameterUnit id="pu1" kind="Return|byReference" type="string"
...
```

```
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Calls "ce1"
    Writes "su1"
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
Value id="null"
ActionElement id="ae2" kind="PtrReplace|ArrayReplace"
    Reads "null"
    Addresses "su1"
```

What to report

Roles to report:
- the <StringManipulationCallStatement> string manipulation call statement

8.64 ASCQM Release File Resource after Use in Class

Descriptor

ASCQM Release File Resource after Use in Class (Class, FileResourceOpenStatement)

Description

Identify occurrences in application model where:
- the <Class> class, ...
- uses the <FileResourceOpenStatement> file resource open statement
- without releasing the file resource in any of its methods

The path to exit the function, procedure, method, includes calls to other functions, procedures, methods, ...

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    ...
    FileResource id="pr1"
    ...
    PlatformAction id="pa1" kind="open" implementation="ae1"
        ManagesResource "pr1"
    PlatformAction id="pa2" kind="close" implementation="ae2"
        ManagesResource "pr1"

...
CodeModel
    ...
    ClassUnit id="cu1"
        ...
        ActionElement id="ae1" kind="PlatformAction"
        ...
...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="cu1"
    ...
    ActionElement id="ae2" kind="PlatformAction"
...
```

What to report
Roles to report:
- the <Class> class
- the <FileResourceOpenStatement> file resource open statement

8.65 ASCQM Use Break in Switch Statement

Descriptor
ASCQM Use Break in Switch Statement (Switch, ControlFlowBranch)

Description
Identify occurrences in application model where:
- the <ControlFlowBranch> control flow branch
- of the <Switch> switch
- does not contain a break statement

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
StorableUnit id="su1"
StorableUnit id="su2"
StorableUnit id="su3"
ActionElement id="ae1"  kind="Switch"
    Reads "su1"
    GuardedFlow "gf1"
    GuardedFlow "gf2"
    ...
    FalseFlow "ff1"
ActionElement id="gf1" kind="Guard"
    Reads "su2"
    Flows "f1"
ActionElement id="gf2"  kind="Guard"
    Reads "su3"
    Flows "f2"
...
ActionElement id="ff1"  kind="Compound"
    ...
    ActionElement id="g1"  kind="Goto"
        Flows "e1"
ActionElement id="f1" kind="Compound"
    ...
ActionElement id="f2"  kind="Compound"
    ...
    ActionElement id="g1"  kind="Goto"
        Flows "e1"
...
ActionElement id="e1"  ...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
StorableUnit id="su1"
StorableUnit id="su2"
ActionElement id="ae1"  kind="Switch"
    Reads "su1"
    GuardedFlow "gf1"
    ...
ActionElement id="gf1" kind="Guard"
    Reads "su2"
    Flows "f1"
...
ActionElement id="f1" kind="Compound"
    ...
    ActionElement id="g1" kind="Goto"
        Flows "e1"
...
ActionElement id="e1"  ...
```

What to report

Roles to report are:
- the <Switch> switch
- the <ControlFlowBranch> control flow branch

8.66 ASCQM Catch Exceptions

Descriptor

ASCQM Catch Exceptions (Method, Exception, MethodCall)

Description

Identify occurrences in application model where:
- the <Method> method
- declared as throwwing the <Exception> exception
- is called in the <MethodCall> method call
- which does not catch exceptions of type <Exception>

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
ClassUnit id="cu1"
...
MethodUnit id="mu1" type="mu1_signature"
    Signature id="mu1_signature"
        ParameterUnit id="pu1" type="cu1" kind="throws"
    ...
...
ActionElement id="ae1" kind="MethodCall"
    Calls "mu1"
...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
...
TryUnit id="t1"
    ...
    ActionElement id="ae1" kind="MethodCall"
        Calls "mu1"
    ...
    ExceptionFlow "c1"
...
CatchUnit id="c1"
    ParameterUnit id="pu2" type="cu1"
    ...
...
```

What to report
Roles to report are:
- the <Method> method
- the <Exception> exception
- the <MethodCall> method call

8.67 ASCQM Ban Empty Exception Block

Descriptor
ASCQM Ban Empty Exception Block (CatchBlock)

Description
Identify occurrences in application model where:
- the <CatchBlock> catch block
- is empty

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
...
CatchUnit id="cu1"
    ActionElement id="ae1" kind="Nop"
...
```

What to report
Roles to report are:
- the <CatchBlock> catch block

8.68 ASCQM Initialize Resource before Use

Descriptor
ASCQM Initialize Resource before Use (PathToResourceAccessFromResourceDeclaration, ResourceDeclarationStatement, ResourceAccessStatement)

Description

Identify occurrences in application model where:
- the <PathToResourceAccessFromResourceDeclaration> path
- from the <ResourceDeclarationStatement> resource declaration statement
- to the <ResourceAccessStatement> resource access statement
- lacks a resource initialization statement

excluding pointers and variables

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    ...
   PlatformResource id="pr1"
    ...
    PlatformResource id="pa1" kind="read|write" implementation="ae6"
        ReadsResource|WritesResource "pr1"
...
CodeModel
    ...
    StorableUnit id="su1"
    ActionElement id="ae1" kind="Assign"
        Writes "su1"
        Flows "ae3"
    ActionElement id="ae3" ...
         Flows "ae4"
    ActionElement id="ae4" ...
         Flows "ae5"
    ActionElement id="ae5" ...
         Flows "ae6"
    ActionElement id="ae6" kind="PlatformAction"
         Reads "su1"
     ...
...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    ...
    PlatformResource id="pa2" kind="open" implementation="ae4"
        ReadsResource|WritesResource "pr1"
...
CodeModel
    ...
    ActionElement id="ae4" kind="PlatformAction"
         Reads "su1"
         Flows "ae5"
     ...
...
```

What to report

Roles to report:
- the <PathToResourceAccessFromResourceDeclaration> path
- the <ResourceDeclarationStatement> resource declaration statement
- the <ResourceAccessStatement> resource access statement

8.69 ASCQM Ban Incompatible Lock Acquisition Sequences

Descriptor
ASCQM Ban Incompatible Lock Acquisition Sequences (LockAcquisitionSequence, ReverseLockAcquisitionSequence)

Description
Identify occurrences in application model where:
- the <LockAcquisitionSequence> sequence of lock acquisition
- is the reverse of the <ReverseLockAcquisitionSequence> sequence of lock acquisition

The locking mechanism is technology, framework, and language dependent.

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ...
        LockResource id="lr1"
        LockResource id="lr2"
        ...
        PlatformAction id="pa1" kind="lock" implementation="ae1 ae12"
            ManagesResource|ReadsResource|WritesResource "lr1"
        PlatformAction id="pa2" kind="lock" implementation="ae3 ae10"
            ManagesResource|ReadsResource|WritesResource "lr2"
...
CodeModel
    ...
    ActionElement id="ae1" kind="PlatformAction"
        Flows "ae2"
    ActionElement id="ae2" ...
        Flows "ae3"
    ActionElement id="ae3" kind="PlatformAction"
        Flows "ae4"
    ActionElement id="ae4" ...
    ...
    ActionElement id="ae10" kind="PlatformAction"
        Flows "ae11"
    ActionElement id="ae11" ...
        Flows "ae12"
    ActionElement id="ae12" kind="PlatformAction"
        Flows "ae13"
    ActionElement id="ae13"  ...
```


What to report
Roles to report are:
- the <LockAcquisitionSequence> sequence of lock acquisition
- the <ReverseLockAcquisitionSequence> sequence of lock acquisition

8.70 ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues

Descriptor
ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues (ThreadControlPrimitiveCall)

Description
Identify occurrences in application model where:
- the <ThreadControlPrimitiveCall> call to a thread control function, procedure, method, ... with known deadlock issues.

The list of primitives is technology, framework, language dependant. For example, in Java: java.lang.Thread.suspend(), java.lang.Thread.resume(), java.lang.ThreadGroup.suspend(), java.lang.ThreadGroup.resume() and dependent methods java.lang.ThreadGroup.allowThreadSuspension().

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce1" name="java.lang.Thread.suspend|java.lang.Thread.resume|..."
    ...
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
    Calls "ce1"
```

What to report
Roles to report:
- the <ThreadControlPrimitiveCall> call to a thread control function, procedure, method, ... with known deadlock issues.

8.71 ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation

Descriptor
ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation (MemoryAllocationCall, BitwiseOperation)

Description
Identify occurrences in application model where:
- the <MemoryAllocationCall> call to a memory allocation primitive
- uses the length parameter based on the <BitwiseOperation> bitwise operation

The list of memory allocation primitives is technology, framework, language dependent. For example with C-type languages: malloc, calloc, realloc.

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
IntegerType id="it1"
...
ControlElement id="ce1" name="malloc|calloc|realloc|..." type="ce1_signature"
    Signature id="ce1_signature"
        ParameterUnit id="pu1" type="it1" kind="byValue"
        ParameterUnit id="pu1" type="pt1" kind="return"
    ...
...
StorableUnit id="su1"  type="it1"
```

```
StorableUnit id="su2"  type="it1"
StorableUnit id="su3"  type="it1"
...
ActionElement id ="ae1" kind="BitAnd|BitOr|BitXor"
    Reads "su1"
    Reads "su2"
    Writes "su3"
ActionElement id="ae2" kind="Call|PtrCall|MethodCall|VirtualCall"
    Reads "su3"
    Calls "ce1"
```

What to report

Roles to report:
- the <MemoryAllocationCall> call to a memory allocation primitive
- the <BitwiseOperation> bitwise operation

8.72 ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size

Descriptor

ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size (MemoryAllocationCall)

Description

Identify occurrences in application model where:
- the <MemoryAllocationCall> call to a memory allocation primitive
- uses the length parameter based on datatype pointer size

The list of memory allocation primitives is technology, framework, language dependent. For example with C-type languages: malloc, calloc, realloc.

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
DataType id="dt1"
PointerType id="pt1"
    ItemUnit id="iu1" type="dt1"
IntegerType id="it1"
ControlElement id="ce1" name="malloc|calloc|realloc|..." type="ce1_signature"
    Signature id="ce1_signature"
        ParameterUnit id="pu1" type="it1" kind="byValue"
        ParameterUnit id="pu1" type="pt1" kind="return"
    ...
...
StorableUnit id="su1"  type="it1"
StorableUnit id="su2"  type="pt1"
StorableUnit id="su3"  type="it1"
...
ActionElement id="ae1" kind="Sizeof"
    Writes "su1"
    Reads "su2" | UsesType "pt1"
ActionElement id="ae2" kind="Multiply"
    Reads "su1"
    Reads ...
    Writes "su3"
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Reads "su3"
    Calls "ce1"
```

What to report
Roles to report:
- the <MemoryAllocationCall> call to a memory allocation primitive

8.73 ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value

Descriptor
ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value (MemoryAllocationCall, LengthComputation)

Description
Identify occurrences in application model where:
- the <MemoryAllocationCall> call to a memory allocation primitive
- uses the length parameter based on the incorrect <LengthComputation> string length computation where 1 is added to
  the string address and not the result of the call

The list of memory allocation primitives is technology, framework, language dependent. For example with C-type languages: malloc, calloc, realloc.
The list of string length computation primitives is technology, framework, language dependent. For example with C-type languages: strlen.

e.g.: new_name = (char*)malloc(strlen(name+1));

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
StringType id="st1"
PointerType id="pt1"
IntegerType id="it1"
...
ControlElement id="ce1" name="strlen|..." type="ce2_signature"
    Signature id="ce2_signature"
        ParameterUnit id="pu3" type="pt1"
        ParameterUnit id="pu4" type="it1" kind="return"
    ...
ControlElement id="ce2" name="malloc|calloc|realloc|..." type="ce1_signature"
    Signature id="ce1_signature"
        ParameterUnit id="pu1" type="it1" kind="byValue"
        ParameterUnit id="pu1" type="pt1" kind="return"
    ...
...
Value id="v1" name="1" type="it1"
StorableUnit id="su1"  type="st1"
StorableUnit id="su2"  type="pt1"
StorableUnit id="su3"  type="it1"
...
ActionElement id ="ae1" kind="Add"
    Reads "su1"
    Reads "v1"
    Writes "su2"
ActionElement id="ae2" kind="PtrCall|Call|MethodCall|VirtualCall"
    Reads "su1"
    Writes "su3"
    Calls "ce1"
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
```

```
    Reads "su3"
    Calls "ce2"
```

What to report

Roles to report:
- the <MemoryAllocationCall> call to a memory allocation primitive
- the <LengthComputation> string length computation

8.74 ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock

Descriptor

ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock (FirstLockAcquisitionStatement, SecondLockAcquisitionStatement)

Description

Identify occurrences in application model where:
- the <FirstLockAcquisitionStatement> lock acquisition statement
- is followed by the <SecondLockAcquisitionStatement> lock acquisition statement
- on a single lock
- without any lock release statement in between

The locking mechanism is technology, framework, and language dependent.
Reentrant locks are excluded.

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ...
        LockResource id="lr1"
        ...
        PlatformAction id="pa2" kind="lock" implementation="ae1 ae5"
            ManagesResource|ReadsResource|WritesResource "lr1"
...
CodeModel
    ...
    ActionElement id="ae1" kind="PlatformAction"
        Flows "ae2"
    ActionElement id="ae2" ...
        Flows "ae3"
    ActionElement id="ae3" ...
        Flows "ae4"
    ActionElement id="ae4" ...
        Flows "ae5"
    ActionElement id="ae5" kind="PlatformAction"
...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DeployedResource id="dr1"
        ...
```

```
        LockResource id="lr1"
        ...
        PlatformAction id="pa2" kind="lock" implementation="ae1 ae5"
            ManagesResource|ReadsResource|WritesResource "lr1"
        PlatformAction id="pa3" kind="unlock" implementation="ae3"
            ManagesResource|ReadsResource|WritesResource "lr1"
...
CodeModel
    ...
    ActionElement id="ae1" kind="PlatformAction"
        Flows "ae2"
    ActionElement id="ae2" ...
        Flows "ae3"
    ActionElement id="ae3" kind="PlatformAction"
        Flows "ae4"
    ActionElement id="ae4" ...
        Flows "ae5"
    ActionElement id="ae5" kind="PlatformAction"
...
```

What to report

Roles to report are:
- the <FirstLockAcquisitionStatement> lock acquisition statement
- the <SecondLockAcquisitionStatement> lock acquisition statement

8.75 ASCQM Initialize Variables

Descriptor

ASCQM Initialize Variables (PathFromVariableDeclaration, VariableDeclarationStatement)

Description

Identify occurrences in application model where:
- the <PathFromVariableDeclaration> path
- from the <VariableDeclarationStatement> variable declaration statement
- lacks a variable initialization statement

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
StorableUnit id="su1"
...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
ActionElement id="ae1" kind="Assign"
  Writes "su1"
  Flows "ae2"
...
```

What to report
Roles to report are
- the <PathFromVariableDeclaration> path
- the <VariableDeclarationStatement> variable declaration statement

8.76 ASCQM Ban Allocation of Memory with Null Size

Descriptor
ASCQM Ban Allocation of Memory with Null Size (MemoryAllocationCall)

Description
Identify occurrences in application model where:
- the <MemoryAllocationCall> call to a memory allocation primitive
- uses a zero length parameter

The list of memory allocation primitives is technology, framework, language dependent. For example with C-type languages: malloc, calloc, realloc.

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
PointerType id="pt1"
IntegerType id="it1"
Value id="v1" type="it1" name="0"
ControlElement id="ce1" name="malloc|calloc|realloc|..." type="ce1_signature"
    Signature id="ce1_signature"
        ParameterUnit id="pu1" type="it1" kind="byValue"
        ParameterUnit id="pu1" type="pt1" kind="return"
    ...
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Reads "v1"
    Calls "ce1"
```

What to report
Roles to report
- the <MemoryAllocationCall> call to a memory allocation primitive

8.77 ASCQM Ban Double Free on Pointers

Descriptor
ASCQM Ban Double Free On Pointers (PathToPointerReleaseFromPointerRelease, FirstPointerReleaseStatement, SecondPointerReleaseStatement)

Description
Identify occurrences in application model where:
- the <PathToPointerReleaseFromPointerRelease> path
- from the <FirstPointerReleaseStatement> pointer release statement
- to the <SecondPointerReleaseStatement> pointer release statement

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit|IntegerType|DecimalType|FloatType|StringType|VoidType|... id="dt1"
PointerType id="pt1"
    ItemUnit id="pi1" type="dt1"
StorableUnit id="su1"  type="pt1"
...
ActionElement id="ae1" name="free|delete|..."
    Addresses "pt1"
    Flows "ae2"
ActionElement id="ae2"
    Flows "ae3"
ActionElement id="ae3" name="free|delete|..."
    Addresses "pt1"
...
```

or

```
ClassUnit|IntegerType|DecimalType|FloatType|StringType|VoidType|... id="dt1" name="dt1"
PointerType id="pt1" name="pt1"
    ItemUnit id="iu1" type="dt1" ext="dt1 & pt1"
StorableUnit id="su1"  type="dt1"
StorableUnit id="su2"  type="pt1"
    HasType "pt1"
    HasValue "su1"
...
ActionElement id="ae1" name="free|delete|...|push_back|..."
    Addresses "su1"
    Flows "ae2"
ActionElement id="ae2"
    Flows "ae3"
ActionElement id="ae3"  name="free|delete|...|push_back|..."
    Addresses "su1"
```

What to report

Roles to report:
- the <PathToPointerReleaseFromPointerRelease> path
- the <FirstPointerReleaseStatement> pointer release statement
- the <SecondPointerReleaseStatement> pointer release statement

8.78 ASCQM Initialize Variables before Use

Descriptor

ASCQM Initialize Variables before Use (PathToVariableAccessFromVariableDeclaration, VariableDeclarationStatement, VariableAccessStatement)

Description

Identify occurrences in application model where:
- the <PathToVariableAccessFromVariableDeclaration> path
- from the <VariableDeclarationStatement> variable declaration statement
- to the <VariableAccessStatement> variable access statement
- lacks a variable initialization statement

excluding pointers and platform resources

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
StorableUnit id="su1"
...
ActionElement id="ae2" ...
     Flows "ae3"
ActionElement id="ae3"
     Reads "su1"
     ...
...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
ActionElement id="ae1" kind="Assign"
    Writes "su1"
    Flows "ae2"
...
```

What to report

Roles to report are:
- the <PathToVariableAccessFromVariableDeclaration> path
- the <VariableDeclarationStatement> variable declaration statement
- the <VariableAccessStatement> variable access statement

8.79 ASCQM Ban Self Assignment

Descriptor

ASCQM Ban Self Assignment (SelfAssignmentStatement)

Description

Identify occurrences in application model where:
- the <SelfAssignmentStatement> assignment statement
- assign one's variable to itself

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
...
StorableUnit id="su1"
...
ActionElement id="ae1" kind="Assign"
    Reads "su1"
    Writes "su1"...
```

What to report

Roles to report:
- the <SelfAssignmentStatement> assignment statement

8.80 ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use

Descriptor

ASCQM Check Boolean Variables are Updated in Different Conditional Branches before Use (Boolean, Condition)

Description

Identify occurrences in application model where:
- the <Boolean> variable
- is used in the <Condition> condition
- but its value is never assigned in different branches of conditional statements

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
BooleanType id="booleanType"
...
StorableUnit id="su1"  type="booleanType"
...
ActionElement id="ae1"  kind="Condition"
    Reads "su1"
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ActionElement id="ae2" kind="Condition"
    ...
    TrueFlow "tf1"
    FalseFlow "ff1"
...
ActionElement id="tf1" kind="Compound"
    ...
    ActionElement id="ae3" kind="Assign"
        Writes "su1"
    ...
ActionElement id="ff1" kind="Compound"
    ...
    ActionElement id="ae4" kind="Assign"
        Writes "su1"
    ...
```

or

```
ActionElement id="ae2" kind="Switch"
    ...
    GuardedFlow "gf1"
    GuardedFlow | FalseFlow "gf2"
...
ActionElement id="gf1" kind="Compound"
    ...
    ActionElement id="ae3" kind="Assign"
        Writes "su1"
    ...
ActionElement id="gf2" kind="Compound"
    ...
```

```
    ActionElement id="ae4" kind="Assign"
        Writes "su1"
    ...
```

What to report
Roles to report:
- the <Boolean> variable
- the <Condition> condition

8.81 ASCQM Ban Not Operator on Operand of Bitwise Operation

Descriptor
ASCQM Ban Not Operator On Operand Of Bitwise Operation (BitwiseExpression)

Description
Identify occurrences in application model where:
- the <BitwiseExpression> bitwise expression with a not operator on one of the operand

KDM outline illustration
KDM outline illustrating only the essential elements related to micro KDM:

```
IntegerType|BitstringType|OctetstringType id="dt1"
StorableUnit id="su1" type="dt1"
...
StorableUnit id="su2" kind="register"
ActionElement id="ae1" kind="Not"
    Reads "su1"
    Writes "su2"
    Flows "ae2"
ActionElement id="ae2" kind="BitAnd|BitOr|BitXor"
    Reads "su2"
    Reads ...
```

What to report
Roles to report are:
- the <BitwiseExpression> bitwise expression

8.82 ASCQM Ban Not Operator on Non-Boolean Operand of Comparison Operation

Descriptor
ASCQM Ban Not Operator On Non-Boolean Operand Of Comparison Operation (ComparisonExpression)

Description
Identify occurrences in application model where:
- the <ComparisonExpression> comparison expression with a not operator on one of the non-boolean operand

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
IntegerType|DecimalType|FloatType|StringType|PointerType|ClassUnit|... id="dt1"
StorableUnit id="su1" type="dt1"
...
StorableUnit id="su2" kind="register"
ActionElement id="ae1" kind="Not"
    Reads "su1"
    Writes "su2"
    Flows "ae2"
ActionElement id="ae2"
kind="Equals|NotEqualTo|GreaterThan|GreaterThanOrEqual|LessThan|LessThanOrEqual"
    Reads "su2"
    Reads ...
```

What to report

Roles to report are:
- the <ComparisonExpression> comparison expression

8.83 ASCQM Ban Incorrect Joint Comparison

Descriptor

ASCQM Ban Incorrect Joint Comparison (JointComparisonExpression)

Description

Identify occurrences in application model where
- the <JointComparisonExpression> joint comparison expression is one of the following: != || != or == || != or == && == or == && !=

KDM outline illustration

KDM outline illustrating only the essential elements related to micro KDM:

```
StorableUnit id="su1" kind="register"
StorableUnit id="su2" kind="register"
...
ActionElement id="ae1" kind="NotEqual"
    ...
    Writes "su1"
ActionElement id="ae2" kind="NotEqual"
    ...
    Writes "su2"
ActionElement id="ae3" kind="Or"
    Reads "su1"
    Reads "su2"
    ...
```

or

```
StorableUnit id="su1" kind="register"
StorableUnit id="su2" kind="register"
...
ActionElement id="ae1" kind="Equals"
    ...
    Writes "su1"
ActionElement id="ae2" kind="NotEqual"
```

```
    ...
    Writes "su2"
ActionElement id="ae3" kind="Or"
    Reads "su1"
    Reads "su2"
    ...
```

or

```
StorableUnit id="su1" kind="register"
StorableUnit id="su2" kind="register"
...
ActionElement id="ae1" kind="Equals"
    ...
    Writes "su1"
ActionElement id="ae2" kind="Equals"
    ...
    Writes "su2"
ActionElement id="ae3" kind="And"
    Reads "su1"
    Reads "su2"
    ...
```

or

```
StorableUnit id="su1" kind="register"
StorableUnit id="su2" kind="register"
...
ActionElement id="ae1" kind="Equals"
    ...
    Writes "su1"
ActionElement id="ae2" kind="NotEqual"
    ...
    Writes "su2"
ActionElement id="ae3" kind="And"
    Reads "su1"
    Reads "su2"
    ...
```

What to report

Roles to report are:
- the <JointComparisonExpression> joint comparison expression

8.84 ASCQM Secure XML Parsing with Secure Options

Descriptor

ASCQM Secure XML Parsing with Secure Options (XMLParsingCall, DTDProcessingDisablingOption)

Description

Identify occurrences in application model where:

- the <XMLParsingCall> call to an XML parsing method, function, procedure, ...
- does not use its <DTDProcessingDisablingOption> DTD processing disabling capability

The list of XML parsing primitives is technology, framework, language dependent. For example, in Java: SchemaFactory, JAXP DocumentBuilderFactory, SAXParserFactory, XMLReader.

The list of option(s) to disable DTD processing is primitive dependent. E.g. with XMLReader: set disallow-doctype-decl feature to true and external-general-entities and external-parameter-entities features to false.
Cf. https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce1"
name="xmlCtxtReadDoc|xmlCtxtReadFd|xmlCtxtReadFile|xmlCtxtReadIO|xmlCtxtReadMemory|xmlCtxt
UseOptions|xmlParseInNodeContext|xmlReadDoc|xmlReadFd|xmlReadFile|xmlReadIO|xmlReadMemory|
..." type="ce1_signature"
    Signature id="ce1_signature"
        ...
        ParameterUnit id="pu1" name="options|..."
    ...
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
    Calls "ce1"
```

or

```
ClassUnit id="cu1"
name="SchemaFactory|DocumentBuilderFactory|SAXParserFactory|XMLReader|...
    MethodUnit id="mu1" name="newSchema|..."
        ...
    MethodUnit id="mu2" name="setProperty|setAttribute|setFeature|..."
type="mu2_signature"
        Signature id="mu2_signature"
            ParameterUnit id="pu1" name="name|property|attribute|feature|..."
            ParameterUnit id="pu2" name="value|..."
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
    Calls "mu1"
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
...
StorableUnit id="su1" attribute="DTD_disable"
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
    Calls "ce1"
    ...
    Reads "su1"
```

or

```
...
StorableUnit id="su1" attribute="DTD_processing"
StorableUnit id="su2" attribute="disable"
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
    ...
    Calls "mu2"
    Reads "su1"
    Reads "su2"
...
```

What to report
Roles to report:
- the <XMLParsingCall> call to an XML parsing function, procedure, method, ...
- the <DTDProcessingDisablingOption> DTD processing disabling option(s)

8.85 ASCQM Secure Use of Unsafe XML Processing with Secure Parser

Descriptor
ASCQM Secure Use of Unsafe XML Processing with Secure Parser (XMLProcessingCall)

Description
Identify occurrences in application model where:
- the <XMLProcessingCall> call to an XML processing method, function, procedure, ... without DTD processing
  disabling capabilities
- is not preceded by a call to a secure XML parser

The list of XML processing primitives without DTD processing disabling capabilities is technology, framework, language dependent. For example in Java: JAXB Unmarshaller, XPathExpression.
The list of XML parsing primitives with DTD processing disabling capabilities is technology, framework, language dependent. For example in Java: DocumentBuilder.
The list of option(s) to disable DTD processing is primitive dependent. For example with SAXParserFactory: set external-general-entities, external-parameter-entities, and load-external-dtd features to false.
Cf. https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="cu1" name="Unmarshaller|XPathExpression|...
    MethodUnit id="mu1" name="unmarshall|evaluate|..."
        ...
...
StorableUnit id="su1"
...
ActionElement id="ae2" kind="MethodCall"
    Reads "su1"
    Calls "mu1"
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ClassUnit id="cu2" name="DocumentBuilder|...
    MethodUnit id="mu2" name="parse|..."

...
ActionElement id="ae1" kind="MethodCall"
    ...
    Calls "mu2"
    Writes "su1"
    Flows "ae2"
...
```

What to report

Roles to report:
- the <XMLProcessingCall> call to an XML processing method, function, procedure, ... without DTD processing
  disabling capabilities

8.86 ASCQM Sanitize User Input used in Path Manipulation

Descriptor

ASCQM Sanitize User Input used in Path Manipulation (PathFromUserInputToPathManipulation, UserInput, PathManipulationStatement, PathManipulationStatementSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToPathManipulation> path
- from the <UserInput> user interface input
- to the <PathManipulationStatement> file path manipulation statement,
- lacks a sanitization operation from the <PathManipulationStatementSanitizationControlElementList> list of vetted
  sanitization.

The list of vetted sanitization primitives is an input to provide to the measurement process.
The list of file manipulation primitives is technology, framework, language dependent. For example with C-type languages: File, FileInputStream, open.

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    FileResource id="fr1"
...
UIModel
    UIField id="uf1"
    UIAction id="ua1" implementation="ae1" kind="input"
        ReadsUI "uf1"
...
CodeModel
    ...
    StorableUnit id="su1"
    StorableUnit id="su2"
    ActionElement id="ae1" kind="UI"
        Writes "su1"
        Flow "ae2"
    ActionElement id="ae2"
        Flow "ae3"
        Reads "su1"
        Writes "su2"
    ActionElement id="ae3"
        Flow "ae4"
    ActionElement id="ae4"
        Flow "ae5"
    ActionElement id="ae5" kind="Data"
        ManagesResource|ReadsResource|WritesResource "fr1"
 ...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce1" kind="sanitization"
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
        Flow "ae4"
        Calls "ce1"
        Reads "su2"
        Writes "su2"
...
```

What to report
Roles to report are:
- the <PathFromUserInputToPathManipulation> path
- the <UserInput> user interface input
- the <PathManipulationStatement> file path manipulation statement,
- the <PathManipulationStatementSanitizationControlElementList> list of vetted sanitization.

8.87 ASCQM Sanitize User Input used in SQL Access

Descriptor
ASCQM Sanitize User Input used in SQL Access (PathFromUserInputToSQLStatement, UserInput, SQLStatement, SQLStatementSanitizationControlElementList)

Description
Identify occurrences in application model where:
- the <PathFromUserInputToSQLStatement> path
- from the <UserInput> user interface input
- to the <SQLStatement> SQL statement,
- lacks a sanitization operation from the <SQLStatementSanitizationControlElementList> list of vetted sanitization.

The list of vetted sanitization primitives is an input to provide to the measurement process.
SQL is not limited to traditional RDBMS SQL, it covers all data management capabilities. For example: NoSQL databases.

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    DataManager id="dm1"
         HasContent "rs1"
...
DataModel
    RelationalSchema id="rs1"
        RelationTable|RelationalView id="rtv1"
    PlatformAction id="pa1" implementation="ae5"
        ReadsColumnSet|WritesColumnSet "rtv1"
        ReadsResource|WritesResource "dm1"
...
UIModel
    UIField id="uf1"
    UIAction id="ua1" implementation="ae1" kind="input"
```

```
        ReadsUI "uf1"
...
CodeModel
    ...
    StorableUnit id="su1"
    StorableUnit id="su2"
    ActionElement id="ae1" kind="UI"
        Writes "su1"
        Flow "ae2"
    ActionElement id="ae2"
        Flow "ae3"
        Reads "su1"
        Writes "su2"
    ActionElement id="ae3"
        Flow "ae4"
    ActionElement id="ae4"
        Flow "ae5"
    ActionElement id="ae5" kind="Data"
 ...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce1" kind="sanitization"
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
        Flow "ae4"
        Calls "ce1"
        Reads "su2"
        Writes "su2"
...
```

What to report

Roles to report are:
- the <PathFromUserInputToSQLStatement> path
- the <UserInput> user interface input
- the <SQLStatement> SQL statement,
- the <SQLStatementSanitizationControlElementList> list of vetted sanitization.

8.88 ASCQM Sanitize User Input used in Document Manipulation Expression

Descriptor

ASCQM Sanitize User Input used in Document Manipulation Expression (PathFromUserInputToDocumentManipulation, UserInput, DocumentManipulationExpression, DocumentManipulationSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToDocumentManipulation> path
- from the <UserInput> user interface input
- to the <DocumentManipulationExpression> document manipulation expression,
- lacks a sanitization operation from the <DocumentManipulationSanitizationControlElementList> list of vetted
  sanitization.

The list of vetted sanitization primitives is an input to provide to the measurement process.

The list of document manipulation primitives is technology, framework, and language dependent. For example: XQuery

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
UIModel
    UIField id="uf1"
    UIAction id="ua1" implementation="ae1" kind="input"
        ReadsUI "uf1"
...
CodeModel
    ...
    StorableUnit id="su1"
    StorableUnit id="su2"
    StringType id="st1"
    StorableUnit id="su3"
    ControlElement id="ce1" name="..."
    ...
    ActionElement id="ae1" kind="UI"
        Writes "su1"
        Flow "ae2"
    ActionElement id="ae2"
        Flow "ae3"
        Reads "su1"
        Writes "su2"
    ActionElement id="ae3"
        Flow "ae4"
    ActionElement id="ae4"
        Flow "ae5"
    ActionElement id="ae5" kind="Call|PtrCall|MethodCall|VirtualCall"
        Calls "ce1"
        Reads "su3"
        Reads "su2"
        ...
 ...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce2" kind="sanitization"
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
        Flow "ae4"
        Calls "ce2"
        Reads "su2"
        Writes "su2"
...
```

What to report

Roles to report are:

- the <PathFromUserInputToDocumentManipulation> path
- the <UserInput> user interface input
- the <DocumentManipulationExpression> document manipulation expression,
- the <DocumentManipulationSanitizationControlElementList> list of vetted sanitization.

8.89 ASCQM Sanitize User Input used in Document Navigation Expression

Descriptor

ASCQM Sanitize User Input used in Document Navigation Expression
(PathFromUserInputToDocumentNavigationEvaluation, UserInput,
DocumentNavigationEvaluationExpression, DocumentNavigationSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToDocumentNavigationEvaluation> path
- from the <UserInput> user interface input
- to the <DocumentNavigationEvaluationExpression> document navigation evaluation expression,
- lacks a sanitization operation from the <DocumentNavigationSanitizationControlElementList> list of vetted
  sanitization.

The list of vetted sanitization primitives is an input to provide to the measurement process.
The list of document navigation expression evaluation primitives is technology, framework, language dependent. For example with Java language: javax.xml.xpath.evaluate, javax.xml.xpath.XPath.evaluateExpression.

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
UIModel
    UIField id="uf1"
    UIAction id="ua1" implementation="ae1" kind="input"
        ReadsUI "uf1"
...
CodeModel
    ...
    StorableUnit id="su1"
    StorableUnit id="su2"
    StringType id="st1"
    StorableUnit id="su3"
    ControlElement id="ce1" name="evaluate|evaluateExpression|..."
    ...
    ActionElement id="ae1" kind="UI"
        Writes "su1"
        Flow "ae2"
    ActionElement id="ae2"
        Flow "ae3"
        Reads "su1"
        Writes "su2"
    ActionElement id="ae3"
        Flow "ae4"
    ActionElement id="ae4"
        Flow "ae5"
    ActionElement id="ae5" kind="Call|PtrCall|MethodCall|VirtualCall"
        Calls "ce1"
        Reads "su3"
        Reads "su2"
        ...
 ...
```

KDM elements absent from the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce2" kind="sanitization"
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
        Flow "ae4"
        Calls "ce2"
        Reads "su2"
        Writes "su2"
...
```

What to report
Roles to report are:
- the <PathFromUserInputToDocumentNavigationEvaluation> path
- the <UserInput> user interface input
- the <DocumentNavigationEvaluationExpression> document navigation evaluation expression,
- the <DocumentNavigationSanitizationControlElementList> list of vetted sanitization.

8.90 ASCQM Sanitize User Input used to access Directory Resources

Descriptor
ASCQM Sanitize User Input used to access Directory Resources (PathFromUserInputToExecuteRunTimeCommand, UserInput, DirectoryAccessStatement, DirectoryAccessStatementSanitizationControlElementList)

Description
Identify occurrences in application model where:
- the <PathFromUserInputToExecuteRunTimeCommand> path
- from the <UserInput> user interface input
- to the <DirectoryAccessStatement> directory access statement,
- lacks a sanitization operation from the <DirectoryAccessStatementSanitizationControlElementList> list of vetted
  sanitization.

The list of vetted sanitization primitives is an input to provide to the measurement process.

KDM outline illustration
KDM elements present in the application model
KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    NamingResource id="nr1"
...
UIModel
    UIField id="uf1"
    UIAction id="ua1" implementation="ae1" kind="input"
        ReadsUI "uf1"
...
CodeModel
    ...
    StorableUnit id="su1"
    StorableUnit id="su2"
    ActionElement id="ae1" kind="UI"
```

```
        Writes "su1"
        Flow "ae2"
    ActionElement id="ae2"
        Flow "ae3"
        Reads "su1"
        Writes "su2"
    ActionElement id="ae3"
        Flow "ae4"
    ActionElement id="ae4"
        Flow "ae5"
    ActionElement id="ae5" kind="Data"
        ManagesResource|ReadsResource|WritesResource "nr1"
 ...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce1" kind="sanitization"
...
ActionElement id="ae3" kind="Call|PtrCall|MethodCall|VirtualCall"
        Flow "ae4"
        Calls "ce1"
        Reads "su2"
        Writes "su2"
...
```

What to report

Roles to report are:
- the <PathFromUserInputToExecuteRunTimeCommand> path
- the <UserInput> user interface input
- the <DirectoryAccessStatement> directory access statement,
- the <DirectoryAccessStatementSanitizationControlElementList> list of vetted sanitization.

8.91 ASCQM Sanitize Stored Input used in User Output

Descriptor

ASCQM Sanitize Stored Input used in User Output (PathFromUserInputToStorageStatement, UserInput, StorageStatement, PathFromRetrievalStatementToUserDisplay, RetrievalStatement, UserDisplay, CrossSiteScriptingSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToStorageStatement> path
- from the <UserInput> user interface input
- to the <StorageStatement> data storage statement,
- and the <PathFromRetrievalStatementToUserDisplay> path
- from the <RetrievalStatement> data retrieval statement
- to the <UserDisplay> user interface display,
- lacks a sanitization operation from the <CrossSiteScriptingSanitizationControlElementList> list of vetted sanitization.

The list of vetted sanitization primitives is an input to provide to the measurement process.

KDM outline illustration

KDM elements present in the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
PlatformModel
    FileResource|DataManager id="pr1"
        HasContent "rr1"
...
DataModel
    RecordFile|RelationalSchema id="rr1"
    DataAction id="da1" implementation="ae3"
        WritessColumnSet ...
        WritesResource "pr1"
    DataAction id="da2" implementation="ae4"
        ReadsColumnSet ...
        ReadsResource "pr1"
...
UIModel
    UIField id="uf1"
    UIAction id="ua1" implementation="ae1" kind="input"
        ReadsUI "uf1"
    UIAction id="ua1" implementation="ae5" kind="output"
        ReadsUI "uf1"
...
CodeModel
    ...
    StorableUnit id="su1"
    StorableUnit id="su2"
    StorableUnit id="su3"
    ActionElement id="ae1" kind="UI"
        Writes "su1"
        Flow "ae2"
    ActionElement id="ae2"
        Flow "ae3"
        Reads "su1"
        Writes "su2"
    ActionElement id="ae3" kind="Data"
        Reads "su2"
        Flow "ae4"
    ...
    ActionElement id="ae4" kind="Data"
        Writes "su3"
        Flow "ae5"
    ActionElement id="ae5"
        Flow "ae6"
    ActionElement id="ae6" kind="UI"
        Reads "su3"
 ...
```

KDM elements absent from the application model

KDM outline illustrating only the essential elements related to micro KDM:

```
ControlElement id="ce1" kind="sanitization"
...
ActionElement id="ae5" kind="Call|PtrCall|MethodCall|VirtualCall"
        Flow "ae6"
        Calls "ce1"
        Reads "su3"
        Writes "su3"
...
```