Date: April 2021

OBJECT MANAGEMENT GROUP®

Automated Source Code Data Protection Measure

Version 1.0 – beta 1

OMG Document Number: ptc/2021-04-01

Standard Document URL:
https://www.omg.org/spec/ASCDPM/

Normative Machine Consumable File(s):
https://www.omg.org/spec/ASCDPM/20201102/AutomatedSourceCodeDataProtectionMeasure.xmi

This OMG document replaces the submission document (admtf/20-12-01). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome and should be directed to issues@omg.org by April 30, 2021.

You may view the pending issues for this specification from the OMG revision issues web page https://issues.omg.org/issues/lists.

The FTF Recommendation and Report for this specification will be published in June 2021. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

USE OF SPECIFICATION – TERMS, CONDITIONS & NOTICES

LICENSES

PATENTS

GENERAL USE RESTRICTIONS

DISCLAIMER OF WARRANTY

RESTRICTED RIGHTS LEGEND

TRADEMARKS

COMPLIANCE

OMG's Issue Reporting Procedure

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

Table of Contents

1 Scope

1.1 Purpose

This specification is derived from the Automated Source Code Security Measure specified in the Automated Source Code Quality Measure (ASCQM) specification (https://www.omg.org/spec/ASCQM/1.0/) to cover common weaknesses (CWEs) that affect the protection of confidential information. Specifying this measure is important as a source of evidence for complying with regulations such as the General Data Protection Regulation (GDPR) in Europe, and the Cybersecurity Maturity Model Certification (CMMC), and California Consumer Privacy Act both in the United States. This measure is calculated from detecting and counting 89 violations of good architectural and coding practices (weaknesses) in the source code that could result in unacceptable risks to the exposure or theft of confidential information. This measure will supplement ISO/IEC 25023 that provides measures of software product confidentiality (a subcharacteristic of Security) by providing a measure at the source code level for protecting confidential data.

1.2 Overview of Structural Quality Measurement in Software

Many recent Governmental regulations are requiring evidence that software-intensive systems provide protection of confidential information. Much of the evidence provided involves the process by which these systems are developed and accessed. However, these regulations are often weak on the evidence required to indicate the systems themselves are secure. This specification addresses one aspect of this problem by providing measure of the extent to which a software system is free from weaknesses that would expose confidential information to unauthorized parties. Thus, this specification provides a measure calculated from detecting weaknesses affecting data protection in the source code.

Measurement of the structural quality characteristics of software such as data protection has a long history in software engineering (Curtis, 1980). Recent advances in measuring the structural quality of software involve detecting violations of good architectural and coding practice from statically analyzing source code. 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 includes weaknesses that affect data protection.

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 data protection. 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 calculation of the Automated Source Code Data Protection Measure from their quality measure elements is then represented in the Structured Metrics Metamodel (SMM). This calculation is developed 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 shall demonstrate the following attributes to claim conformance: automated, objective, transparent, and verifiable.

- ***Automated***—The analysis of the source code and counting of weaknesses must 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 must not require further human intervention. The analysis and calculation must be able to repeatedly produce the same results and outputs on the same body of software.
- ***Transparent***—Implementations that conform to this specification must 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 state the assumptions/heuristics it uses with sufficient detail so that the calculations may be independently verified by third parties. In addition, all inputs used are required to be clearly described and itemized so that they can be audited by a third party.

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, https://www.omg.org/spec/SPMS/1.2/
- Knowledge Discovery Metamodel, version 1.4 (KDM), https://www.omg.org/spec/KDM/1.4/
- Structured Metrics Metamodel, version 1.2 (SMM), formal/2012-01-05
- MOF/XMI Mapping, version 2.5.1 (XMI), https://www.omg.org/spec/XMI/2.5.1/
- ISO/IEC 25020:2007 Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Measurement reference model and guide
- International Organization for Standards (2019). *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.

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

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

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)

Data Protection—the ability of a software product to prevent unauthorized access to confidential information contained within the product or within any software product it interacts with.

Detection Pattern—a collection of parsed program elements and their relations that constitute a weakness in the software.

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)

Data Protection 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 (adapted from ISO/IEC 25020)

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)

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

Software Data Protection Attribute—an inherent property or characteristic of software that can be distinguished quantitatively or qualitatively by human or automated means. (adapted from ISO/IEC 25020)

Software Data Protection 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.

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

Weakness— sometimes referred to as a software anti-pattern, is a pattern or structure in the code (Detection 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)

5 Symbols (and Abbreviated Terms)

AFP — Automated Function Points

ASCSM — Automated Source Code Security Measure

CWE — Common Weakness Enumeration

CISQ — Consortium for Information and Software Quality

KDM — Knowledge Discovery Metamodel

SPMS — Structured Pattern Metamodel Standard

SMM — Structured Metrics Metamodel

6 Additional Information (Informative)

6.1 Software Product Inputs

The following inputs are needed by static code analyzers to interpret violations of the software data protection rules that would be included in individual software data protection 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.

6.2 Automated Source Code Data Protection Measure Elements

The weaknesses violating software data protection rules that compose the CISQ Automated Source Code Data Protection Measure are presented in clauses 6 and 7. All weaknesses included in this measure are identified by their CWE number from the CWE repository. In most cases the description of CWEs is taken from information in the online CWE repository (cwe.mitre.org).

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 table 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. Compliance to the CISQ measures is assessed at the level of the parent weakness. A technology must be able to detect at least one of the contributing weaknesses to be assessed compliant on the parent weakness.

The data protection measure elements (weaknesses violating software data protection rules) that compose the CISQ Automated Source Code Data Protection Measure are presented in Table 1. This measure contains 36 parent weaknesses and 53 contributing weaknesses (children in the CWE) that represent variants of these weaknesses. The CWE numbers for contributing weaknesses are presented in light blue cells immediately below the parent weakness whose CWE number is in a dark blue cell. The weaknesses included in this measure are compared to those in the CISQ Automated Source Code Security and Reliability Measures in Annex C.

Table 1. Data Protection Measure Elements for the Automated Source Code Data Protection Measure

CWE #	Descriptor
CWE-22	**Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')**
CWE-23	**Relative Path Traversal**
CWE-36	**Absolute Path Traversal**
CWE-77	**Improper Neutralization of Special Elements used in a Command ('Command Injection')**

CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
CWE-88	Argument Injection or Modification
CWE-624	Executable Regular Expression Error
CWE-917	Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross Site Scripting')
CWE-89	Improper Neutralization of Special Elements used in a SQL Command ('SQL Injection')
CWE-90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')
CWE-91	XML Injection (aka Blind XPath Injection)
CWE-99	Improper Control of Resource Identifiers ('Resource Injection')
CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer
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
CWE-129	Improper Validation of Array Index
CWE-134	Use of Externally Controlled Format String
CWE-170	Improper Null Termination
CWE-213	Exposure of Sensitive Information Due to Incompatible Policies
CWE-284	Improper Access Control
CWE-285	Improper Authorization
CWE-287	Improper Authentication
CWE-288	Authentication Bypass Using an Alternate Path or Channel
CWE-639	Authorization Bypass Through User-Controlled Key
CWE-862	Missing Authorization
CWE-863	Incorrect Authorization
CWE-311	Missing Encryption of Sensitive Data
CWE-359	Exposure of Private Personal Information to an Unauthorized Actor
CWE-404	Improper Resource Shutdown or Release
CWE-761	Free of Pointer not at Start of Buffer
CWE-762	Mismatched Memory Management Routines
CWE-763	Release of Invalid Pointer or Reference

CWE-772	Missing Release of Resource after Effective Lifetime
CWE-775	Missing Release of File Descriptor or Handle after Effective Lifetime
CWE-424	Improper Protection of Alternate Path
CWE-434	Unrestricted Upload of File with Dangerous Type
CWE-502	Deserialization of Untrusted Data
CWE-562	Return of Stack Variable Address
CWE-606	Unchecked Input for Loop Condition
CWE-611	Improper Restriction of XML External Entity Reference ('XXE')
CWE-643	Improper Neutralization of Data within XPath Expressions ('XPath Injection')
CWE-652	Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')
CWE-662	Improper Synchronization
CWE-667	Improper Locking
CWE-764	Multiple Locks of a Critical Resource
CWE-820	Missing Synchronization
CWE-821	Incorrect Synchronization
CWE-1058	Invokable Control Element in Multi-Thread Context with non-Final Static Storable or Member Element
CWE-1096	Singleton Class Instance Creation without Proper Locking or Synchronization
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-665	Improper Initialization
CWE-456	Missing Initialization of a Variable
CWE-457	Use of Uninitialized Variable
CWE-672	Operation on a Resource after Expiration or Release
CWE-415	Double Free
CWE-416	Use After Free
CWE-681	Incorrect Conversion between Numeric Types
CWE-194	Unexpected Sign Extension
CWE-195	Signed to Unsigned Conversion Error
CWE-196	Unsigned to Signed Conversion Error
CWE-197	Numeric Truncation Error
CWE-682	Incorrect Calculation
CWE-131	Incorrect Calculation of Buffer Size
CWE-369	Divide by Zero
CWE-703	Improper Check or Handling of Exceptional Conditions
CWE-248	Uncaught Exception
CWE-391	Unchecked Error Condition
CWE-392	Missing Report of Error Condition

CWE-704	Incorrect Type Conversion or Cast
CWE-732	Incorrect Permission Assignment for Critical Resource
CWE-798	Use of Hard-coded Credentials
CWE-259	Use of Hard-coded Password
CWE-321	Use of Hard-coded Cryptographic Key
CWE-908	Use of Uninitialized Resource
CWE-915	Improperly Controlled Modification of Dynamically-Determined Object Attributes
CWE-1051	Initialization with Hard-Coded Network Resource Configuration Data

6.3 Specification of Data Protection 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 Data Protection 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.
- Calculation of the Automated Source Code Data Protection Measure is represented in the Structured Metrics Metamodel (SMM).

6.4 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

6.5 Knowledge Discovery Metamodel (KDM)

This specification uses the Knowledge Discovery Metamodel (KDM) to represent the parsed entities whose relationships create a weakness pattern. The machine readable XMI format file attached to the current 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 chapter. However, for readers not familiar with KDM, Table 2 presents a primer which translates standard source code element terms into the KDM outline in this specification.

Table 2 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" ...

<table>
<tr><td>user input data flow</td><td><pre>
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"
        ...
</pre></td></tr>
<tr><td>execution path</td><td><pre>
ActionElement id="ae1" kind="UI"
    Flow|Calls "ae2"
ActionElement id="ae2"
    Flow|Calls "ae3"
ActionElement id="ae3"
    Flow|Calls "ae4"
</pre></td></tr>
<tr><td>RDBMS</td><td>DataModel
RelationalSchema ...</td></tr>
</table>

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"`

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

6.6 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 patterns 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.

6.7 Reading guide

Each numbered sub-clause in clause 7 represents the SPMS modeling, SMM, and detection pattern(s) associated with a specific data protection weakness. 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 and SMM) detection patterns,
  - 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

Last sub-clauses are summarizing the computation of the quality measure scores:

- (SMM) detection patterns,
  - useful for reporting of detection pattern-level findings at the quality characteristic level
  - useful for computing the score of the quality measure, 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 ASCDPM Weaknesses (Normative)

7.1 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

ASCQM Sanitize User Input used in Path Manipulation

7.2 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

ASCQM Sanitize User Input used in Path Manipulation

7.3 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

ASCQM Sanitize User Input used in Path Manipulation

7.4 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-624 Executable Regular Expression Error
CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
CWE-88 Argument Injection or Modification
CWE-917 Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')

Detection Patterns

ASCQM Sanitize User Input used in Expression Language Statement
ASCQM Sanitize User Input used in System Command

7.5 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

ASCQM Sanitize User Input used in System Command

7.6 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

ASCQM Sanitize Stored Input used in User Output
ASCQM Sanitize User Input used in User Output

7.7 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

ASCQM Sanitize User Input used in System Command

7.8 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>

Detection Patterns

ASCQM Sanitize User Input used in SQL Access

7.9 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

ASCQM Sanitize User Input used to access Directory Resources

7.10 CWE-91 — XML Injection (aka Blind XPath Injection)

Reference

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

Roles

- the <XMLHandlingExpression>
- the <TaintedValue>

Detection Patterns

ASCQM Sanitize User Input used in Document Manipulation Expression
ASCQM Sanitize User Input used in Document Navigation Expression

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

Reference

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

Roles

- the <ResourceIdentifier>
- the <TaintedValue>

Detection Patterns

ASCQM Sanitize User Input used in Path Manipulation

7.12 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

ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities
ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities
ASCQM Ban Use of Expired Pointer
ASCQM Check Index of Array Access
ASCQM Check Input of Memory Manipulation Primitives
ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities
ASCQM Check Offset used in Pointer Arithmetic
ASCQM Initialize Pointers before Use
ASCQM Sanitize User Input used as Pointer

7.13 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

ASCQM Ban Input Acquisition Primitives without Boundary Checking Capabilities
ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities

7.14 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

ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities

7.15 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

ASCQM Check Index of Array Access

7.16 CWE-129 — Improper Validation of Array Index

Reference

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

Roles

- the <ArrayAccess>
- the <TaintedIndex>

Detection Patterns

ASCQM Sanitize User Input used as Array Index

7.17 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

ASCQM Check Index of Array Access

7.18 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

ASCQM Ban Buffer Size Computation Based on Array Element Pointer Size
ASCQM Ban Buffer Size Computation Based on Bitwise Logical Operation
ASCQM Ban Buffer Size Computation Based on Incorrect String Length Value

7.19 CWE-134 — Use of Externally-Controlled Format String

Reference

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

Roles

- the <Formatting>
- the <TaintedFormatString>

Detection Patterns

ASCQM Sanitize User Input used as String Format

7.20 CWE-170 — Improper Null Termination

Reference

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

Roles

- the <BufferWithoutNULLTermination>

Detection Patterns

ASCQM NULL Terminate Output Of String Manipulation Primitives

7.21 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

ASCQM Ban Incorrect Numeric Implicit Conversion

7.22 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

ASCQM Ban Incorrect Numeric Implicit Conversion

7.23 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

ASCQM Ban Incorrect Numeric Implicit Conversion

7.24 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

ASCQM Ban Incorrect Numeric Implicit Conversion

7.25 CWE-213 — Exposure of Sensitive Information Due to Incompatible Policies

Reference

https://cwe.mitre.org/data/definitions/213

Roles

- the <SensitiveInformation>
- the <IncompatiblePath>

Detection Patterns

ASCQM Ban Unintented Paths To Sensitive Data

7.26 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

ASCQM Catch Exceptions

7.27 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

ASCQM Ban Hard-Coded Literals used to Connect to Resource

7.28 CWE-284 — Improper Access Control

Reference

https://cwe.mitre.org/data/definitions/284

Roles

- the <AccessControlStatement>

Contributing weaknesses

CWE-285 Improper Authorization
CWE-287 Improper Authentication
CWE-288 Authentication Bypass Using an Alternate Path or Channel
CWE-639 Authorization Bypass Through User-Controlled Key
CWE-862 Missing Authorization
CWE-863 Incorrect Authorization

Detection Patterns

ASCQM Ban Unintended Paths Bypassing Authentication
ASCQM Ban Unintended Paths Bypassing Authorization
ASCQM Catch Authentication Exceptions
ASCQM Catch Authorization Exceptions
ASCQM Check Return Value of Authentication Operations Immediately
ASCQM Check Return Value of Authorization Operations Immediately
ASCQM Sanitize User Input used in SQL Access to primary keys
ASCQM Sanitize User Input used in URI Building

7.29 CWE-285 — Improper Authorization

Reference

https://cwe.mitre.org/data/definitions/285

Roles

- the <AuthorizationStatement>

Parent weaknesses

CWE-284 Improper Access Control

Detection Patterns

ASCQM Ban Unintended Paths Bypassing Authorization

ASCQM Catch Authorization Exceptions
ASCQM Check Return Value of Authorization Operations Immediately

7.30 CWE-287 — Improper Authentication

Reference

Reference https://cwe.mitre.org/data/definitions/287 Improper Authentication

Roles

- the <AuthenticationStatement>

Parent weaknesses

CWE-284 Improper Access Control

Detection Patterns

ASCQM Ban Unintended Paths Bypassing Authentication
ASCQM Catch Authentication Exceptions
ASCQM Check Return Value of Authentication Operations Immediately

7.31 CWE-288 — Authentication Bypass Using an Alternate Path or Channel

Reference

https://cwe.mitre.org/data/definitions/288

Roles

- the <AlternatePath>

Parent weaknesses

CWE-284 Improper Access Control

Detection Patterns

ASCQM Ban Unintended Paths Bypassing Authentication

7.32 CWE-311 — Missing Encryption of Sensitive Data

Reference

https://cwe.mitre.org/data/definitions/311

Roles

- the <SensitiveData>
- the <PathWithoutEncryption>

Detection Patterns

ASCQM Encrypt User Input used in SQL Access to Sensitive Data

7.33 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

ASCQM Ban Hard-Coded Literals used to Connect to Resource

7.34 CWE-359 — Exposure of Private Personal Information to an Unauthorized Actor

Reference

https://cwe.mitre.org/data/definitions/359

Roles

- the <PrivatePersonalInformation>
- the <UnahthorizedPath>

Detection Patterns

ASCQM Ban Unintented Paths To Sensitive Data

7.35 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

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

7.36 CWE-369 — Divide By Zero

Reference

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

Roles

- the <Division>

Parent weaknesses

CWE-682 Incorrect Calculation

Detection Patterns

ASCQM Check and Handle ZERO Value before Use as Divisor

7.37 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

ASCQM Ban Empty Exception Block
ASCQM Ban Useless Handling of Exceptions

7.38 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

ASCQM Ban Useless Handling of Exceptions

7.39 CWE-404 — Improper Resource Shutdown or Release

Reference

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

Roles

- the <ResourceAllocation>

Contributing weaknesses

CWE-761 Free of Pointer not at Start of Buffer
CWE-762 Mismatched Memory Management Routines
CWE-763 Release of Invalid Pointer or Reference
CWE-772 Missing Release of Resource after Effective Lifetime
CWE-775 Missing Release of File Descriptor or Handle after Effective Lifetime

Detection Patterns

ASCQM Ban Comma Operator from Delete Statement
ASCQM Implement Required Operations for Manual Resource Management
ASCQM Implement Virtual Destructor for Classes Derived from Class with Virtual Destructor
ASCQM Implement Virtual Destructor for Classes with Virtual Methods
ASCQM Implement Virtual Destructor for Parent Classes
ASCQM Release File Resource after Use in Class
ASCQM Release File Resource after Use in Operation
ASCQM Release Memory After Use
ASCQM Release Memory after Use with Correct Operation
ASCQM Release Memory after Use with Correct Reference
ASCQM Release Platform Resource after Use
ASCQM Release in Destructor Memory Allocated in Constructor

7.40 CWE-415 — Double Free

Reference

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

Roles

- the <FirstResourceRelease>
- the <SecondResourceRelease>

Parent weaknesses

CWE-672 Operation on a Resource after Expiration or Release

Detection Patterns

ASCQM Ban Double Free On Pointers

7.41 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

ASCQM Ban Free Operation on Pointer Received as Parameter
ASCQM Ban Use of Expired Pointer
ASCQM Implement Copy Constructor for Class With Pointer Resource

7.42 CWE-424 — Improper Protection of Alternate Path

Reference

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

Roles

- the <AlternatePath>

Detection Patterns

ASCQM Ban Unintended Paths
ASCQM Ban Unintended Paths Bypassing Authentication
ASCQM Ban Unintended Paths Bypassing Authorization

7.43 CWE-434 — Unrestricted Upload of File with Dangerous Type

Reference

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

Roles

- the <FileUpload>

Detection Patterns

ASCQM Sanitize User Input used in Path Manipulation

7.44 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

ASCQM Ban Allocation of Memory with Null Size
ASCQM Initialize Variables

7.45 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

ASCQM Initialize Pointers before Use
ASCQM Initialize Variables before Use

7.46 CWE-502 —— Deserialization of Untrusted Data

Reference

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

Roles

- the <Deserialization>
- the <TaintedData>

Detection Patterns

ASCQM Sanitize User Input used as Serialized Object

7.47 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

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

7.48 CWE-562 — Return of Stack Variable Address

Reference

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

Roles

- the <ReturnStatement>

Detection Patterns

ASCQM Ban Return of Local Variable Address
ASCQM Ban Storage of Local Variable Address in Global Variable

7.49 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

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

7.50 CWE-606 — Unchecked Input for Loop Condition

Reference

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

Roles

- the <LoopCondition>
- the <TaintedValue>

Detection Patterns

ASCQM Sanitize User Input used in Loop Condition

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

Reference

https://cwe.mitre.org/data/definitions/611

Roles

- the <XMLHandlingOperation>

Detection Patterns

ASCQM Secure Use of Unsafe XML Processing with Secure Parser
ASCQM Secure XML Parsing with Secure Options

7.52 CWE-624 — Executable Regular Expression Error

Reference

https://cwe.mitre.org/data/definitions/624

Roles

- the <RegularExpression>
- the <TaintedValue>

Parent weaknesses

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

Detection Patterns

ASCQM Sanitize User Input used in System Command

7.53 CWE-639 — Authorization Bypass Through User-Controlled Key

Reference

https://cwe.mitre.org/data/definitions/639

Roles

- the <AuthorizationStatement>

Parent weaknesses

CWE-284 Improper Access Control

Detection Patterns

ASCQM Sanitize User Input used in SQL Access to primary keys
ASCQM Sanitize User Input used in URI Building

7.54 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

ASCQM Sanitize User Input used in Document Navigation Expression

7.55 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

ASCQM Sanitize User Input used in Document Manipulation Expression

7.56 CWE-662 — Improper Synchronization

Reference

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

Roles

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

Contributing weaknesses

CWE-1058 Named Callable and Method Control Element in Multi-Thread Context with non-Final Static Storable or Member Element
CWE-1096 Singleton Class Instance Creation without Proper Lock Element Management
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

Detection Patterns

ASCQM Ban Creation of Lock On Inappropriate Object Type
ASCQM Ban Creation of Lock On Non-Final Object
ASCQM Ban Creation of Lock On Private Non-Static Object to Access Private Static Data
ASCQM Ban Incompatible Lock Acquisition Sequences
ASCQM Ban Incorrect Synchronization Mechanisms
ASCQM Ban Non-Final Static Data in Multi-Threaded Context
ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context
ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock
ASCQM Ban Sleep Between Lock Acquisition and Release
ASCQM Ban Use of Thread Control Primitives with Known Deadlock Issues
ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context
ASCQM Release Lock After Use
ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context

7.57 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

ASCQM Ban Allocation of Memory with Null Size
ASCQM Ban Self Assignment
ASCQM Initialize Pointers before Use
ASCQM Initialize Variables
ASCQM Initialize Variables before Use

7.58 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

ASCQM Ban Creation of Lock On Inappropriate Object Type
ASCQM Ban Creation of Lock On Non-Final Object
ASCQM Ban Creation of Lock On Private Non-Static Object to Access Private Static Data
ASCQM Ban Resource Access without Proper Locking in Multi-Threaded Context
ASCQM Ban Sleep Between Lock Acquisition and Release
ASCQM Data Read and Write without Proper Locking in Multi-Threaded Context
ASCQM Release Lock After Use

7.59 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

ASCQM Ban Double Free On Pointers
ASCQM Ban Double Release of Resource
ASCQM Ban Free Operation on Pointer Received as Parameter
ASCQM Ban Use of Expired Pointer
ASCQM Ban Use of Expired Resource
ASCQM Implement Copy Constructor for Class With Pointer Resource

7.60 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

ASCQM Ban Incorrect Numeric Implicit Conversion

7.61 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

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

7.62 CWE-703 — Improper Check or Handling of Exceptional Conditions

Reference

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

Roles

- the <ErrorHandling>

Contributing weaknesses

CWE-248 Uncaught Exception
CWE-391 Unchecked Error Condition
CWE-392 Missing Report of Error Condition

Detection Patterns

ASCQM Ban Empty Exception Block
ASCQM Ban Useless Handling of Exceptions
ASCQM Catch Exceptions

7.63 CWE-704 — Incorrect Type Conversion or Cast

Reference

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

Roles

- the <TypeConversion>

Detection Patterns

ASCQM Ban Incorrect Type Conversion

7.64 CWE-732 — Incorrect Permission Assignment for Critical Resource

Reference

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

Roles

- the <PermissionAssignment>

Detection Patterns

ASCQM Ban File Creation with Default Permissions

7.65 CWE-761 — Free of Pointer not at Start of Buffer

Reference

https://cwe.mitre.org/data/definitions/761

Roles

- the <ResourceRelease>

Parent weaknesses

CWE-404 Improper Resource Shutdown or Release

Detection Patterns

ASCQM Release Memory after Use with Correct Reference

7.66 CWE-762 — Mismatched Memory Management Routines

Reference

https://cwe.mitre.org/data/definitions/762

Roles

- the <MemoryAllocation>
- the <MemoryRelease>

Parent weaknesses

CWE-404 Improper Resource Shutdown or Release

Detection Patterns

ASCQM Release Memory after Use with Correct Operation

7.67 CWE-763 — Release of Invalid Pointer or Reference

Reference

https://cwe.mitre.org/data/definitions/763

Roles

- the <ResourceRelease>

Parent weaknesses

CWE-404 Improper Resource Shutdown or Release

Detection Patterns

ASCQM Release Memory after Use with Correct Operation
ASCQM Release Memory after Use with Correct Reference

7.68 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

ASCQM Ban Sequential Acquisitions of Single Non-Reentrant Lock

7.69 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

ASCQM Release File Resource after Use in Operation
ASCQM Release Platform Resource after Use
ASCQM Release in Destructor Memory Allocated in Constructor

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

Reference

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

- the <FileDescriptorOrHandleAllocation>

Parent weaknesses

CWE-404 Improper Resource Shutdown or Release

Detection Patterns

ASCQM Release File Resource after Use in Class
ASCQM Release File Resource after Use in Operation

7.71 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

ASCQM Check Index of Array Access
ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities

7.72 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

ASCQM Check Index of Array Access
ASCQM Check Input of Memory Manipulation Primitives

7.73 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

ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities
ASCQM Check Index of Array Access
ASCQM Check Input of Memory Manipulation Primitives

7.74 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

ASCQM Ban Hard-Coded Literals used to Connect to Resource

7.75 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

ASCQM Ban String Manipulation Primitives without Boundary Checking Capabilities
ASCQM Check Input of Memory Manipulation Primitives

ASCQM Check Input of String Manipulation Primitives with Boundary Checking Capabilities

7.76 CWE-820 — Missing Synchronization

Reference

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

Roles

- the <SharedResourceUse>

Parent weaknesses

CWE-662 Improper Synchronization

Detection Patterns

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

7.77 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

ASCQM Ban Incorrect Synchronization Mechanisms

7.78 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

ASCQM Sanitize User Input used as Pointer

7.79 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

ASCQM Check Offset used in Pointer Arithmetic

7.80 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

ASCQM Initialize Pointers before Use

7.81 CWE-825 — Expired Pointer Dereference

Reference

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

Roles

- the <PointerAccess>
- the <PointerRelease>

Parent weaknesses

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

Detection Patterns

ASCQM Ban Use of Expired Pointer

7.82 CWE-862 — Missing Authorization

Reference

https://cwe.mitre.org/data/definitions/862

Roles

- the <AlternatePath>

Parent weaknesses

CWE-284 Improper Access Control

Detection Patterns

ASCQM Ban Unintended Paths Bypassing Authorization

7.83 CWE-863 — Incorrect Authorization

Reference

https://cwe.mitre.org/data/definitions/863

Roles

- the <AuthorizationStatement>

Parent weaknesses

CWE-284 Improper Access Control

Detection Patterns

ASCQM Catch Authorization Exceptions
ASCQM Check Return Value of Authorization Operations Immediately

7.84 CWE-908 — Use of Uninitialized Resource

Reference

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

Roles

- the <ResourceUse>

Detection Patterns

ASCQM Initialize Resource before Use

7.85 CWE-915 — Improperly Controlled Modification of Dynamically-Determined Object Attributes

Reference

https://cwe.mitre.org/data/definitions/915

Roles

- the <StoredData>
- the <UnsanitizedPath>

Detection Patterns

ASCQM Sanitize Deserialized Object used in Stored Data

7.86 CWE-917 — Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')

Reference

https://cwe.mitre.org/data/definitions/917

Roles

- the <ExpressionLanguageStatement>
- the <TaintedValue>

Parent weaknesses

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

Detection Patterns

ASCQM Sanitize User Input used in Expression Language Statement

7.87 CWE-1051 — Storable and Member Data Element 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

ASCQM Ban Hard-Coded Literals used to Connect to Resource

7.88 CWE-1058 — Named Callable and Method 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

ASCQM Ban Non-Final Static Data in Multi-Threaded Context

7.89 CWE-1096 Singleton Class Instance Creation without Proper Lock Element Management

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

ASCQM Singleton Creation without Proper Locking in Multi-Threaded Context

8 ASCQM Weakness Detection Patterns (Normative)

8.1 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

8.2 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

8.3 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

8.4 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

8.5 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

8.6 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

8.7 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

8.8 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.

8.9 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

8.10 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

8.11 ASCQM Ban Double Release of Resource

Descriptor

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

8.11.1.1.1

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

8.12 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

8.13 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 are 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

8.14 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

8.15 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

8.16 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

8.17 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

8.18 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

8.19 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

8.20 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

8.21 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

8.22 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

8.23 ASCQM Release File Resource after Use in Operation

Descriptor

ASCQM Release File Resource after Use in Operation(FunctionProcedureOrMethod, FileResourceOpenStatement, PathToExitWithoutFileResourceClose)

8.23.1.1.1

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

8.24 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

8.25 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

8.26 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

8.27 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

8.28 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

8.29 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

8.30 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

8.31 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

8.32 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

8.33 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

8.34 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

8.35 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

8.36 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

8.37 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

8.38 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

8.39 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

8.40 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

8.41 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

8.42 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

8.43 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

8.44 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 doesn't 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

8.45 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

8.46 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

8.47 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

8.48 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

8.49 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

8.50 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

8.51 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

8.52 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

8.53 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

8.54 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

8.55 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

8.56 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

8.57 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, ...
- doesn't 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|xmlCtxtReadMemor
y|xmlCtxtUseOptions|xmlParseInNodeContext|xmlReadDoc|xmlReadFd|xmlReadFile|xmlRea
dIO|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)

8.58 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

8.59 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.

8.60 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.

8.61 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.

8.62 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.

8.63 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.

8.64 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"
...
```

What to report

Roles to report are:
- the <PathFromUserInputToStorageStatement> path
- the <UserInput> user interface input
- the <StorageStatement> data storage statement,
- the <PathFromRetrievalStatementToUserDisplay> path
- the <RetrievalStatement> data retrieval statement
- the <UserDisplay> user interface display,
- the <CrossSiteScriptingSanitizationControlElementList> list of vetted sanitization.

8.65 ASCQM Sanitize User Input used in User Output

Descriptor

ASCQM Sanitize User Input used in User Output(PathFromUserInputToUserDisplay, UserInput, UserDisplay, CrossSiteScriptingSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToUserDisplay> path
- from the <UserInput> user interface input
- 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:

```
UIModel
    UIField id="uf1"
    UIAction id="ua1" implementation="ae1" kind="input"
        ReadsUI "uf1"
    UIField id="uf2"
    UIAction id="ua1" implementation="ae5" kind="output"
        WritesUI "uf2"
...
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="UI"
 ...
```

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 <PathFromUserInputToUserDisplay> path
- the <UserInput> user interface input
- the <UserDisplay> user interface display,
- the <CrossSiteScriptingSanitizationControlElementList> list of vetted sanitization operations

8.66 ASCQM Sanitize User Input used in System Command

Descriptor

ASCQM Sanitize User Input used in System Command(PathFromUserInputToExecuteRunTimeCommand, UserInput, ExecuteRunTimeCommandStatement, ExecuteRunTimeCommandStatementSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToExecuteRunTimeCommand> path
- from the <UserInput> user interface input
- to the <ExecuteRunTimeCommandStatement> system command,
- lacks a sanitization operation from the <ExecuteRunTimeCommandStatementSanitizationControlElementList> 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
    RunTimeResource id="rtr1"
...
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 "rtr1"
 ...
```

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 <ExecuteRunTimeCommandStatement> system command,
- the <ExecuteRunTimeCommandStatementSanitizationControlElementList> list of vetted sanitization.

8.67 ASCQM Sanitize User Input used as Array Index

Descriptor

ASCQM Sanitize User Input used as Array Index(PathFromUserInputToArrayAccess, UserInput, ArrayAccessStatement)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToArrayAccess> path
- from the <UserInput> user interface input
- to the <ArrayAccessStatement> array access statement,
- 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:

```
UIModel
    UIField id="uf1"
    UIAction id="ua1" implementation="ae1" kind="input"
        ReadsUI "uf1"
...
CodeModel
    ...
    StorableUnit id="su1"
    StorableUnit id="su2"
    ArrayType id="at1"
    StorableUnit id="su3" type="at1"
    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="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 <PathFromUserInputToArrayAccess> path
- the <UserInput> user interface input
- the <ArrayAccessStatement> array access statement,

8.68 ASCQM Sanitize User Input used as String Format

Descriptor

ASCQM Sanitize User Input used as String Format(PathFromUserInputToFormatStatement, UserInput, FormatStatement, FormatStatementSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToFormatStatement> path
- from the <UserInput> user interface input
- to the <FormatStatement> formatting statement,
- lacks a sanitization operation from the <FormatStatementSanitizationControlElementList> list of vetted sanitization.

The list of vetted sanitization primitives is an input to provide to the measurement process.
The list of string format primitives is technology, framework, language dependent. For example with C-type languages: printf, snprintf.

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="printf|snprintf|..."
    ...
    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 <PathFromUserInputToFormatStatement> path
- the <UserInput> user interface input
- the <FormatStatement> formatting statement,
- the <FormatStatementSanitizationControlElementList> list of vetted sanitization.

8.69 ASCQM Sanitize User Input used in Loop Condition

Descriptor

ASCQM Sanitize User Input used in Loop Condition(PathFromUserInputToLoopCondition, UserInput, LoopConditionStatement)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToLoopCondition> path
- from the <UserInput> user interface input
- to the <LoopConditionStatement> loop condition,
- 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:

```
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="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 "ae10"
        ActionElement id="ae8" kind=...
            ...
        ActionElement id="ae9" kind="Incr|Decr"
            Addresses "loopVariable"
            Flows "ae6"
        ActionElement id="ae10" kind="Nop"
```

or

```
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="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"  ...
...
```

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 <PathFromUserInputToLoopCondition> path
- the <UserInput> user interface input
- the <LoopConditionStatement> loop condition,

8.70 ASCQM Sanitize User Input used as Serialized Object

Descriptor

ASCQM Sanitize User Input used as Serialized Object(PathFromUserInputToDeserialization, UserInput, DeserializationStatement, DeserializationStatementSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToDeserialization> path
- from the <UserInput> user interface input
- to the <DeserializationStatement> deserialization statement,
- lacks a sanitization operation from the <DeserializationStatementSanitizationControlElementList> list of vetted sanitization.

The list of vetted sanitization primitives is an input to provide to the measurement process.
The list of deserialization primitives is technology, framework, language dependent. For example in Java: XMLdecoder, readObject, readExternal.

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="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 <PathFromUserInputToDeserialization> path
- the <UserInput> user interface input
- the <DeserializationStatement> deserialization statement,
- the <DeserializationStatementSanitizationControlElementList> list of vetted sanitization.

8.71 ASCQM Ban File Creation with Default Permissions

Descriptor

ASCQM Ban File Creation with Default Permissions(FileCreationStatement, Permission)

Description

Identify occurrences in application model where:
- the <FileCreationStatement> file creation statement with permission setting capabilities
- doesn't use its <Permission> permission option

The list of file creation primitives with permission setting capabilities is technology, framework, language dependent. For example: open from fcntl.h in C, os.open in python.

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="open|..." type="ce1_signature"
    Signature id="ce1_signature"
        ParameterUnit id="pu1" name="file|..."
        ParameterUnit id="pu2" name="flags|..."
        ParameterUnit id="pu3" name="mode|..."
    ...
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Calls "ce1"
    Reads ...
    Reads ...
```

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

```
...
ActionElement id="ae1" kind="Call|PtrCall|MethodCall|VirtualCall"
    Calls "ce1"
    Reads ...
    Reads ...
    Reads ...
```

What to report

Roles to report:
- the <FileCreationStatement> file creation statement with permission setting capabilities
- the <Permission> permission option

8.72 ASCQM Ban Unintended Paths Bypassing Authentication

Descriptor

ASCQM Ban Unintended Paths Bypassing Authentication(ArchitectureModel, Relation, Caller, Callee, OriginModule, TargetModule, AuthenticationModule)

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
- bypasses the <AuthenticationModule> authentication layer, component, or subsystem
- as defined in the <ArchitectureModel> architectural blueprint defining layers, components, or subsystems

The architectural blueprint defining layers, components, or subsystems is application dependent, including the identification of the authentication layer, component, or subsystem.

KDM outline illustration

KDM elements present in the application model
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="..."
...
```

KDM elements absent from the application model
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
            "ce3"
...
Layer|Component|Subsystem id="m3" kind="authentication"
     ...
      CallableUnit callableKind="regular|external|stored" | MethodUni
      id="ce3" name="..."
         ...
```

```
        ActionElement id="ae3"
            UsesType|Reads|Writes|Creates|Addresses|Calls|Dispatches "ce2"
...
Layer|Component|Subsystem id="m2"
    ...
     CallableUnit callableKind="regular|external|stored" | MethodUnit
     id="ce2" name="..."
...
```

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
- the <AuthenticationModule> authentication layer, component, or subsystem

8.73 ASCQM Ban Unintended Paths Bypassing Authorization

Descriptor

ASCQM Ban Unintended Paths Bypassing Authorization (ArchitectureModel, Relation, Caller, Callee, OriginModule, TargetModule, AuthorizationModule)

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
- bypasses the <AuthorizationModule> authentication layer, component, or subsystem
- as defined in the <ArchitectureModel> architectural blueprint defining layers, components, or subsystems

The architectural blueprint defining layers, components, or subsystems is application dependent, including the identifaction of the authorization layer, component, or subsystem.

KDM outline illustration

KDM elements present in the application model 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="..."
...
```

KDM elements absent from the application model
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 "ce3"
...
Layer|Component|Subsystem id="m3" kind="authorization"
    ...
    CallableUnit callableKind="regular|external|stored" | MethodUnit id="ce3"
    name="..."
        ...
        ActionElement id="ae3"
            UsesType|Reads|Writes|Creates|Addresses|Calls|Dispatches "ce2"
...
Layer|Component|Subsystem id="m2"
    ...
    CallableUnit callableKind="regular|external|stored" | MethodUnit id="ce2"
    name="..."
...
```

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
- the <AuthorizationModule> authorization layer, component, or subsystem

8.74 ASCQM Ban Unintented Paths To Sensitive Data

Descriptor

ASCQM Ban Unintented Paths To Sensitive Data(PathFromUserInputToSQLStatement, UserInput, SQLStatement, PriviledgedInterfaceList, SensitveDataList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToSQLStatement> path
- from the <UserInput> user interface input,
- not identified as part of the <PriviledgedInterfaceList> list of priviledged interfaces,
- to the <SQLStatement> SQL statement,
- accessing data from the <SensitveDataList> list of sensitive data.

The list of list of sensitive data is an input to provide to the measurement process. It typically comes from data census required by data protection regulations. The list of list of priviledged interfaces is an input to provide to the measurement process. It typically comes from interface census required by data protection regulations. SQL is not limited to traditional RDBMS SQL, it covers all data management capabilities. E.g.: 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 <PriviledgedInterfaceList> list of priviledged interfaces
- the <SensitveDataList> list of sensitive data.

8.75 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. E.g. 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.

8.76 ASCQM Catch Authentication Exceptions

Descriptor

ASCQM Catch Authentication Exceptions(AuthenticationMethod, Exception, MethodCall)

Description

Identify occurrences in application model where
- the <AuthenticationMethod> authentication method
- declared as throwwing the <Exception> exception
- is called in the <MethodCall> method call
- which does not catch exceptions of type <Exception> The list of authentication management function, procedure, method, ... is technology dependant.

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" kind="authentication"
	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 <AuthenticationMethod> authentication method
- the <Exception> exception
- the <MethodCall> method call

8.77 ASCQM Catch Authorization Exceptions

Descriptor

ASCQM Catch Authorization Exceptions(AuthorizationMethod, Exception, MethodCall)

Description

Identify occurrences in application model where:
- the <AuthorizationMethod> authorization method
- declared as throwwing the <Exception> exception
- is called in the <MethodCall> method call
- which does not catch exceptions of type <Exception>

The list of authorization management function, procedure, method, ... is technology dependant.

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" kind="authorization"
      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 <AuthorizationMethod> authorization method
- the <Exception> exception
- the <MethodCall> method call

8.78 ASCQM Check Return Value of Authentication Operations Immediately

Descriptor

ASCQM Check Return Value of Authentication Operations Immediately(CallToTheOperation)

Description

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

The list of authentication management function, procedure, method, ... is technology dependent.

KDM outline illustration

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

```
...
CodeModel
        CallableUnit|MethodUnit id="ce1" type="ce1_signature" kind="authentication"
              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 statement

8.79 ASCQM Check Return Value of Authorization Operations Immediately

Descriptor

ASCQM Check Return Value of Authorization Operations Immediately(CallToTheOperation)

Description

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

The list of authorization management function, procedure, method, ... is technology dependant.

KDM outline illustration

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

```
...
CodeModel
      CallableUnit|MethodUnit id="ce1" type="ce1_signature" kind="authorization"
            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 statement

8.80 ASCQM Encrypt User Input used in SQL Access to Sensitive Data

Descriptor

ASCQM Encrypt User Input used in SQL Access to Sensitive Data(PathFromUserInputToSQLStatement, UserInput, SQLStatement, EncryptionControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToSQLStatement> path
- from the <UserInput> user interface input
- to the <SQLStatement> SQL statement,
- lacks an encryption operation from the <EncryptionControlElementList> list of vetted encryption.

The list of list of sensitive data is an input to provide to the measurement process. It typically comes from data census required by data protection regulations. The list of vetted encryption primitives is an input to provide to the measurement process. SQL is not limited to traditional RDBMS SQL, it covers all data management capabilities. E.g.: 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="encryption"
...
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 <EncryptionControlElementList> list of vetted encryption.

8.81 ASCQM Release Memory after Use with Correct Reference

Descriptor

ASCQM Release Memory after Use with Correct Reference(MemoryAllocationStatement, AllocationReference, MemoryReleaseStatement, ReleaseReference)

Description

Identify occurrences in the application model where;
- the memory is allocated via the <MemoryAllocationStatement> allocation statement
- using the <AllocationReference> reference
- then released via <MemoryReleaseStatement> release statement
- using the mismatched <ReleaseReference> reference

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

```
ControlElement id="ce1" name="malloc|calloc|...|New|NewArray|..."
...
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"
...
StorableUnit id="su2" type="pt1"
...
ActionElement id="ae2" type="add"
      Reads "su1"
      ...
      Writes "su2"
...
ControlElement id="ce2" name="free|...|delete|delete[]|..."
...
ActionElement id="ae3" kind="Call"
      Addresses "su2"
      Calls "ce2"
```

What to report

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

8.82 ASCQM Sanitize Deserialized Object used in Stored Data

Descriptor

ASCQM Sanitize Deserialized Object used in Stored Data(PathFromObjectDeserializationToStorage, StorageStatement, DeserializationStatement, DeserializationStatementSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromObjectDeserializationToStorage> path
- from the <DeserializationStatement> deserialization statement,
- to the <StorageStatement> storage statement,
- lacks a sanitization operation from the <DeserializationStatementSanitizationControlElementList> list of vetted sanitization.

The list of vetted sanitization primitives is an input to provide to the measurement process. The list of deserialization primitives is technology, framework, language dependent. E.g. in Java: XMLdecoder, readObject, readExternal.

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="ae4"
            WritessColumnSet ...
            WritesResource "pr1"
...
CodeModel
      ...
      StorableUnit id="su1"
      StorableUnit id="su2"
      ActionElement id="ae1" kind="deserialization"
            Writes "su1"
            Flow "ae2"
      ActionElement id="ae2"
            Flow "ae3"
            Reads "su1"
            Writes "su2"
      ActionElement id="ae3"
            Flow "ae4"
      ActionElement id="ae4" kind="Data"
            ManagesResource|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 <PathFromObjectDeserializationToStorage> path
- the <StorageStatement> storage statement,
- the <DeserializationStatement> deserialization statement,
- the <DeserializationStatementSanitizationControlElementList> list of vetted sanitization.

8.83 ASCQM Sanitize User Input used in Expression Language Statement

Descriptor

ASCQM Sanitize User Input used in Expression Language Statement(UserInput, TransformationSequence, ExpressionLanguageExpression, ExpressionLanguageSanitizationControlElementList)

Description

Identify occurrences in application model where:
- an external value is entered into the application through the <UserInput> user interface input,
- transformed throughout the application along the <TransformationSequence> sequence,
- and ultimately used in <ExpressionLanguageExpression> EL expression,
- none of the callable or method control element of the transformation sequence being a vetted sanitization operation from the <ExpressionLanguageSanitizationControlElementList> list of vetted sanitization operations.

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"
      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 <UserInput> user interface input action
- the <TransformationSequence> sequence
- the <ExpressionLanguageExpression> EL expression
- the <ExpressionLanguageSanitizationControlElementList> list of vetted sanitization operations

8.84 ASCQM Sanitize User Input used in SQL Access to primary keys

Descriptor

ASCQM Sanitize User Input used in SQL Access to primary keys(PathFromUserInputToSQLStatement, UserInput, SQLStatement, PrimaryKey, SQLStatementSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToSQLStatement> path
- from the <UserInput> user interface input
- to the <SQLStatement> SQL statement,
- which accesses the <PrimaryKey> primary key,
- 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.

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"
                  UniqueKey id="uk1" implementation="iu1"
                  ItemUnit id="iu1"
      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"
            Reads|Writes to="iu1"
...
```

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 <PrimaryKey> primary key,
- the <SQLStatementSanitizationControlElementList> list of vetted sanitization.

8.85 ASCQM Sanitize User Input used in URI Building

Descriptor

ASCQM Sanitize User Input used in URI Building(PathFromUserInputToURIBuildingStatement, UserInput, URIBuildingStatement, URIBuildingStatementSanitizationControlElementList)

Description

Identify occurrences in application model where:
- the <PathFromUserInputToURIBuildingStatement> path
- from the <UserInput> user interface input
- to the <URIBuildingStatement> SQL statement,

- lacks a sanitization operation from the <URIBuildingStatementSanitizationControlElementList> list of vetted sanitization.

The list of vetted sanitization primitives is an input to provide to the measurement process. The list of URI building function, method, ... is technology dependent.

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="URI"
...
```

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 <PathFromUserInputToURIBuildingStatement> path
- the <UserInput> user interface input
- the <URIBuildingStatement> SQL statement,
- the <URIBuildingStatementSanitizationControlElementList> list of vetted sanitization.

9 Calculation of Quality and Functional Density Measures (Normative)

9.1 Calculation of the Base Measure

After reviewing several alternatives, a count of total weaknesses in an application was selected as the best option for a base measure for the Automated Source Code Data Protection Measure. Software quality measures have frequently been scored at the component level and then aggregated to develop an overall score for the application. However, scoring at the component level was rejected because many weaknesses cannot be isolated to a single component, but rather involve interactions among several components. Therefore, the Automated Source Code Data Protection Measure score is computed as the sum of its quality measure elements (weaknesses) counted across an entire application.

The Automated Source Code Data Protection Measure score is calculated as follows:

- The score for each weakness is the count of its detection patterns, and
- The Automated Source Code Data Protection Measure score is the sum of its weakness scores.

That is,

Score for Weakness x_i = Σ (Occurrences of ASCQM-x_{ij})
Where x = an Automated Source Code Data Protection Measure weakness
i = 1 to 85
ASCQM-x_{ij} = the jth detection pattern associated with weakness x_i
and

Automated Source Code Data Protection Measure score = Σ (Weakness X_i scores) for weaknesses 1 to 85.

9.2 Functional Density of Weaknesses

To compare quality results among different applications, the Automated Source Code Data Protection Measure can be normalized by size to create a density measure. There are several size measures with which the density of quality violations can be normalized, such as lines of code and Function Points. These size measures, if properly standardized, can be used for creating a density measure for use in benchmarking the quality of applications. OMG's Automated Function Points (AFP) measure (ISO, 2019) offers an automatable size measure that, as an OMG Supported Specification, is standardized. AFP was adapted from the International Function Point User Group's (IFPUG) counting guidelines, and is commercially supported. Although other size measures can be used to evaluate the density of security violations, the following density measure for weaknesses is derived from the OMG supported specification for Automated Function Points. Thus, the functional density of Data Protection weaknesses is a simple division expressed as follows.

ASCDPM-density = ASCDPM / AFP

Where ASPCM-density = the density of ASCDPM weaknesses per Automated Function Point,
ASCDPM = Automated Source Code Data Protection Measure score, and
AFP = Automated Function Points

10 References (Informative)

Common Weakness Enumeration. http://cwe.mitre.org . Bedford, MA: MITRE Corporation.

Consortium for IT Software Quality (2010). http://www.it-cisq.org . Needham, MA: Object Management Group, Consortium for IT Software Quality (CISQ).

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

International Organization for Standards (2007). ISO/IEC 25020 Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality – Measurement reference model and guide. Geneva, Switzerland.

International Organization for Standards (2011). ISO/IEC 25010:2011 Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models. Geneva, Switzerland.

International Organization for Standards (2012). ISO/IEC 25023 Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality. Geneva, Switzerland.

International Organization for Standards (2019). *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 http://www.omg.org/spec/AFP/ . Needham, MA: Object Management Group.

International Telecommunications Union (2012). ITU-T X.1524 – Series X: Data Networks, Open System Communications and Security – Cybersecurity information exchange – Vulnerability/state exchange – Common weakness enumeration. Geneva:, Switzerland.

Martin, R.A. & Barnum, S. (2006). *Status update: The Common Weakness Enumeration*. NIST Static Analysis Summit, Gaithersburg, MD Jun 29, 2006.

Annex A: Consortium for IT Software Quality (CISQ) (Informative)

The Consortium for IT Software Quality (CISQ), a consortium managed by OMG, was formed in 2010 to create international standards for automating measures of size and structural quality characteristics from source code. These measures were designed to provide international standards for measuring software structural quality that can be used by IT organizations, IT service providers, and software vendors in contracting, developing, testing, accepting, and deploying IT software applications. Executives from the member companies that joined CISQ prioritized the quality characteristics of Reliability, Security, Performance Efficiency, and Maintainability to be developed as measurement specifications.

CISQ strives to maintain consistency with ISO/IEC standards to the extent possible, and in particular with the ISO/IEC 25000 series that replaces ISO/IEC 9126 and defines quality measures for software systems. In order to maintain consistency with the quality model presented in ISO/IEC 25010, software quality characteristics are defined for the purpose of this specification as attributes that can be measured from the static properties of software, and can be related to the dynamic properties of a computer system as affected by its software. However, the 25000 series, and in particular ISO/IEC 25023 which elaborates quality characteristic measures, define very few of these measures at the source code level. Thus, this and other CISQ quality characteristic specifications supplement ISO/IEC 25023 by providing a deeper level of software measurement, one that is rooted in measuring software attributes in the source code.

An international team of experts drawn from CISQ's 24 original companies formed into working groups to define CISQ measures. Weaknesses that had a high probability of causing reliability, security, performance efficiency, or maintainability problems were selected for inclusion in the four measures. The original CISQ members included IT departments in Fortune 200 companies, system integrators/ outsourcers, and vendors that provide quality-related products and services to the IT market. The experts met several times per year for two years in the US, France, and India to develop a broad list of candidate weaknesses. This list was pared down to a set of weaknesses they believed had to be remediated to avoid serious operational or cost problems. These 86 weaknesses became the foundation of the original specifications of the automated source code measures for Reliability, Security, Performance Efficiency, and Maintainability. In 2018 these measures were extended to include weaknesses related to embedded software. There are now 133 weaknesses in the 4 CISQ measures that are collectively referred to as CISQ's Automated Source Code Quality Measures (ASCQM).

This specification of weaknesses related to data protection extends the CISQ Security measure to the specific domain of and the protection of confidential data. It is directly related to the ISO 25010 subcharacteristic of Confidentiality, which is categorized under Security.

Annex B: Common Weakness Enumeration (CWE) (Informative)

The Common Weakness Enumeration (CWE) repository (http://cwe.mitre.org/) maintained by MITRE Corporation is a collection of over 800 weaknesses in software architecture and source code that malicious actors have used to gain unauthorized entry into systems or to cause malicious actions. The CWE is a widely used industry source (http://cwe.mitre.org/community/citations.html) that provides a foundation for the ITU-T X.1524 and ISO/IEC standard, in addition to 2 ISO/IEC technical reports:

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

The CWE/SANS Institute Top 25 Most Dangerous Software Errors is a list of the 25 most widespread and frequently exploited security weaknesses in the CWE repository. The previous version of the CISQ Automated Source Code Security Measure (ASCSM) was based on 22 of the CWE/SANS Top 25 that could be detected and counted in source code. In this revision, the number of security weaknesses is being expanded beyond the CWE/SANS Top 25 since there are other weaknesses severe enough to be incorporated in the CISQ measure. In addition, many CWEs also cause reliability problems and are therefore included in the CISQ reliability measure. Wherever a CWE is included in any of the 4 CISQ structural quality measures, its CWE identifier will be noted.

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

Annex C: Comparison of Weaknesses Included in the CISQ Automated Source Code Security, Reliability, and Data Protection Measures (Informative)

This annex displays a comparison of the weaknesses in CISQ's Automated Source Code Security, Reliability, and Data Protection Measures. There are 26 weaknesses in the CISQ Data Protection measure that are not in the CISQ Security measure. However, of these 26 weaknesses, 11 weaknesses are included in the CISQ Reliability measure. There are 11 weaknesses included in the CISQ Security measure that are not included in the CISQ Data Protection measure.

Table C1: Comparison of Weaknesses in CISQ Security, Reliability, and Data Protection Measures

CWE	Security	Reliability	Data Protection
22	X		X
23	X		X
36	X		X
77	X		X
78	X		X
79	X		X
88	X		X
89	X		X
90	X		X
91	X		X
99	X		X
119	X	X	X
120	X	X	X
123	X	X	X
125	X	X	X
129	X		X
130	X	X	X
131	X	X	X
134	X		X
170		X	X
194	X	X	X
195	X	X	X
196	X	X	X
197	X	X	X
213			X
248		X	X
252	X	X	
259	X		X
284			X
285			X
287			X
288			X
311			X

321	X		X
359			X
366	X	X	X
369	X	X	X
390		X	
391		X	X
392		X	X
394		X	
401	X	X	
404	X	X	X
415	X	X	X
416	X	X	X
424	X	X	X
434	X		X
456	X	X	X
457	X	X	X
459		X	
476		X	
477	X		
480	X	X	
484		X	
502	X		X
543	X	X	X
562		X	X
564	X		
567	X	X	X
570	X		
571	X		
595		X	
597		X	
606	X		X
611	X		X
624			X
639			X
643	X		X
652	X		X
662	X	X	X
665	X	X	X
667	X	X	X
672	X	X	X
681	X	X	X
682	X	X	X
703		X	X
704		X	X
732	X		X
758		X	

761			X
762			X
763			X
764		X	X
772	X	X	X
775	X	X	X
778	X		
783	X		
786	X	X	X
787	X	X	X
788	X	X	X
789	X		
798	X		X
805	X	X	X
820	X	X	X
821	X	X	X
822	X	X	X
823	X	X	X
824	X	X	X
825	X	X	X
833		X	
835	X	X	
862			X
863			X
908		X	X
915			X
917	X		X
1045		X	
1051		X	X
1057	X		
1058		X	X
1066		X	
1070		X	
1077		X	
1079		X	
1082		X	
1083		X	
1087		X	
1088		X	
1096		X	X
1097		X	
1098		X	

Annex D: Relationship of the CISQ Automated Source Code Data Protection Measure to ISO 25000 Series Standards (SQuaRE) (Informative)

ISO/IEC 25010 defines the product quality model for software-intensive systems (Figure 1). This model is composed of 8 quality characteristics, four of which are the subject of CISQ structural quality measures (indicated in blue). Each of ISO/IEC 25010's eight quality characteristics consists of several quality sub-characteristics that define the domain of issues covered by their parent quality characteristic. CISQ structural quality measures conform to the definitions in ISO/IEC 25010. The sub-characteristics of each quality characteristic were used to ensure the CISQ measures covered the domain of issues in each of the four areas. The CISQ Automated Source Code Data Protection Measure is conformant to the subcharacteristic measure of Confidentiality under the quality characteristic of Security. ISO/IEC 25010 is currently undergoing revision with CISQ participation.

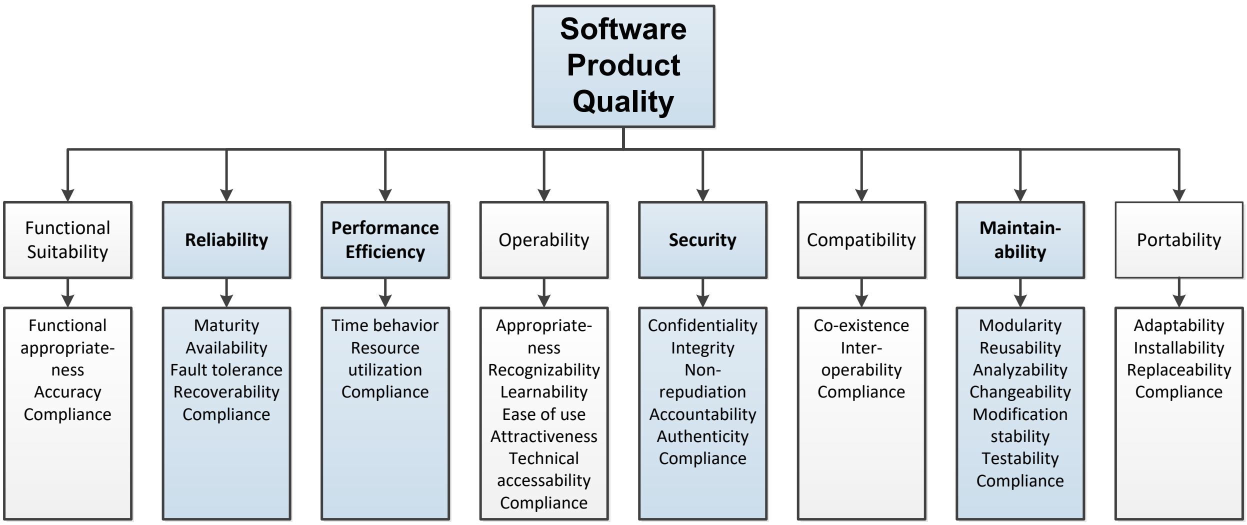
Software Product Quality
Functional Suitability
Reliability
Performance Efficiency
Operability
Security
Compatibility
Maintain-ability
Portability
Functional appropriate-ness Accuracy Compliance
Maturity Availability Fault tolerance Recoverability Compliance
Time behavior Resource utilization Compliance
Appropriate-ness Recognizability Learnability Ease of use Attractiveness Technical accessability Compliance
Confidentiality Integrity Non-repudiation Accountability Authenticity Compliance
Co-existence Inter-operability Compliance
Modularity Reusability Analyzability Changeability Modification stability Testability Compliance
Adaptability Installability Replaceability Compliance

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

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

Figure 2 displays the hierarchical relationships indicating how CISQ conforms to the reference measurement structure established in ISO/IEC 25020 that governs software quality measures in ISO/IEC 25023. This structure is presented using the CISQ Security measure as an example. The CISQ measures only use ISO's quality subcharacteristics for ensuring that the CISQ weaknesses covered the measurable domain of an ISO quality characteristic as defined in ISO/IEC 25010. CISQ's weaknesses (CWEs) correspond to ISO's quality attributes. CISQ weaknesses are represented as one or more detection patterns among structural code elements in the software. Variations in how a weakness may be instantiated are represented by its association with several different detection patterns. Each occurrence of a detection pattern represents an occurrence of a weakness in the software. Occurrences of these detection patterns in the software correspond to ISO's quality measure elements and are the elements calculated in the CISQ measures.

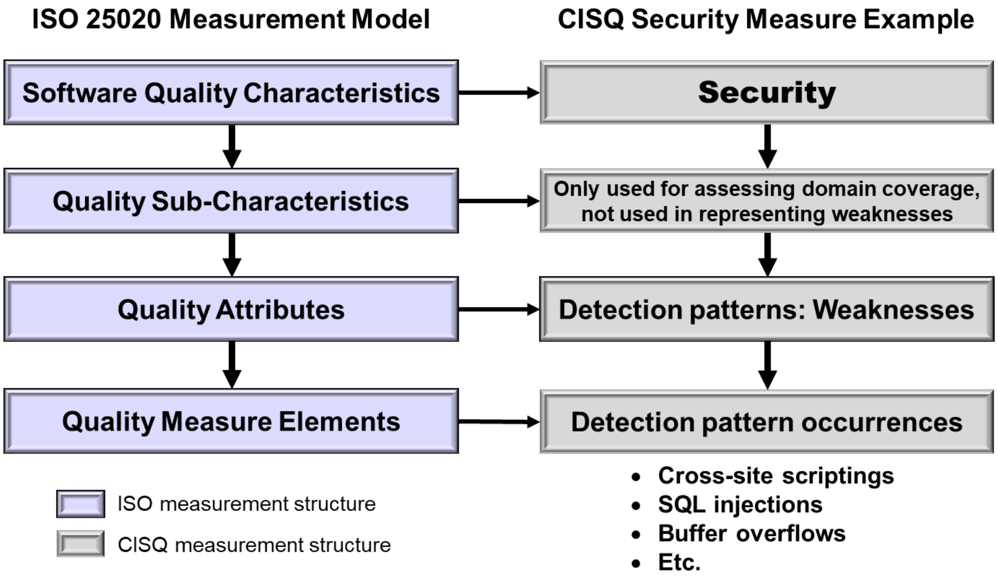
ISO 25020 Measurement Model
CISQ Security Measure Example
Software Quality Characteristics
Security
Quality Sub-Characteristics
Only used for assessing domain coverage, not used in representing weaknesses
Quality Attributes
Detection patterns: Weaknesses
Quality Measure Elements
Detection pattern occurrences
Cross-site scriptings
SQL injections
Buffer overflows
Etc.
ISO measurement structure
CISQ measurement structure

Figure 2 ISO/IEC 25020 Framework for Software Quality Characteristics Measurement

Clause 6 of this specification lists weaknesses that correspond to ISO/IEC 25020's quality attributes. A weakness is detected by identifying patterns of code elements in the software (called detection patterns) that instantiate the weakness. Each detection pattern equates to a quality measure element used in calculating the CISQ quality measures. In Clause 7, quality attributes (weaknesses) are transformed into the KDM and SPMS-based detection patterns that represent them. The CISQ quality measures are then calculated by detecting and counting occurrences of detection patterns, each of which indicates the existence of a weakness in the software. These calculations are represented in the Structured Metrics Metamodel (SMM).