List of Weaknesses Included in the CISQ Automated Source Code Security Measure

June 2019

Overview of Structural Quality Measurement in Software

Measurement of the structural quality characteristics of software has a long history in software engineering. These characteristics are also referred to as the structural, internal, technical, or engineering characteristics of software source code. Software quality characteristics are increasingly incorporated into development and outsourcing contracts as the equivalent of service level agreements. That is, target thresholds based on structural quality measures are being written into contracts as acceptance criteria for delivered software. This specification provides automated measures for four structural quality characteristics listed in the ISO/IEC 25010 software quality model that can be calculated from source code—Reliability, Security, Performance Efficiency, and Maintainability.

Recent advances in measuring the structural quality of software involve detecting violations of good architectural and coding practice from statically analyzing source code. Good architectural and coding practices can be stated as rules for engineering software products. Violations of these rules will be called weaknesses to be consistent with terms used in the Common Weakness Enumeration which lists the weaknesses used in these measures.

The four Automated Source Code Quality Measures are calculated from counts of what industry experts have determined to be most severe weaknesses. Consequently, they provide strong indicators of the quality of a software system and the probability of operational or cost problems related to each measure's domain.

The weaknesses comprising each CISQ Automated Source Code Quality Measure are grouped by measure in a table. This document lists the weaknesses in the Security measure. The Common Weakness Enumeration repository (an ITU standard) has recently been expanded to include weaknesses from quality characteristics beyond security. All weaknesses included in these measures are identified by their CWE number from the repository. The title and description of CWEs is taken from information in the online CWE repository (cwe.mitre.org). Each weakness will be described as a 'quality measure element' to remain consistent with the structure of software quality measures enumerated in ISO/IEC 25020.

Some weaknesses drawn from the CWE repository (parent weaknesses) have related weaknesses listed as 'contributing weaknesses' ('child weaknesses' 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 in this standard. 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.

Automated Source Code Security Measure Element Descriptions

The quality measure elements (weaknesses violating software quality rules) that compose the CISQ Automated Source Code Security Measure are presented in the table. This measure contains 36 parent weaknesses and 38 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.

Table: Quality Measure Elements for Automated Source Code Security Measure

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

CWE-79	**Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')**	The software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users. Cross-site scripting (XSS) vulnerabilities occur when: 1. Untrusted data enters a web application, typically from a web request. 2. The web application dynamically generates a web page that contains this untrusted data. 3. During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc. 4. A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data. 5. Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain. 6. This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.
CWE-89	**Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')**	The software constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.
CWE-564	**SQL Injection: Hibernate**	Using Hibernate to execute a dynamic SQL statement built with user-controlled input can allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.
CWE-90	**Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')**	The software constructs all or part of an LDAP query using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended LDAP query when it is sent to a downstream component.
CWE-91	**XML Injection (aka Blind XPath Injection)**	The software does not properly neutralize special elements that are used in XML, allowing attackers to modify the syntax, content, or commands of the XML before it is processed by an end system.
CWE-99	**Improper Control of Resource Identifiers ('Resource injection')**	The software receives input from an upstream component, but it does not restrict or incorrectly restricts the input before it is used as an identifier for a resource that may be outside the intended sphere of control.
CWE-119	**Improper Restriction of Operations within the Bounds of a Memory Buffer**	The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.

CWE-120	**Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')**	The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.
CWE-123	**Write-what-where condition**	Any condition where the attacker has the ability to write an arbitrary value to an arbitrary location, often as the result of a buffer overflow.
CWE-125	**Out-of-bounds Read**	The software reads data past the end, or before the beginning, of the intended buffer.
CWE-130	**Improper Handling of Length Parameter Inconsistency**	The software parses a formatted message or structure, but it does not handle or incorrectly handles a length field that is inconsistent with the actual length of the associated data.
CWE-786	**Access of Memory Location Before Start of Buffer**	The software reads or writes to a buffer using an index or pointer that references a memory location prior to the beginning of the buffer. This typically occurs when a pointer or its index is decremented to a position before the buffer, when pointer arithmetic results in a position before the beginning of the valid memory location, or when a negative index is used.
CWE-787	**Out-of-bounds Write**	The software writes data past the end, or before the beginning, of the intended buffer. The software may modify an index or perform pointer arithmetic that references a memory location that is outside of the boundaries of the buffer.
CWE-788	**Access of Memory Location After End of Buffer**	The software reads or writes to a buffer using an index or pointer that references a memory location after the end of the buffer. This typically occurs when a pointer or its index is decremented to a position before the buffer; when pointer arithmetic results in a position before the buffer; or when a negative index is used, which generates a position before the buffer.
CWE-805	**Buffer Access with Incorrect Length Value**	The software uses a sequential operation to read or write a buffer, but it uses an incorrect length value that causes it to access memory that is outside of the bounds of the buffer.

CWE-822	**Untrusted Pointer Dereference**	The program obtains a value from an untrusted source, converts this value to a pointer, and dereferences the resulting pointer. There are several variants of this weakness, including but not necessarily limited to: □ The untrusted value is directly invoked as a function call. □□□In OS kernels or drivers where there is a boundary between "userland" and privileged memory spaces, an untrusted pointer might enter through an API or system call (see CWE-781 for one such example). □ Inadvertently accepting the value from an untrusted control sphere when it did not have to be accepted as input at all. This might occur when the code was originally developed to be run by a single user in a non-networked environment, and the code is then ported to or otherwise exposed to a networked environment.
CWE-823	**Use of Out-of-range Pointer Offset**	The program performs pointer arithmetic on a valid pointer, but it uses an offset that can point outside of the intended range of valid memory locations for the resulting pointer. □ While a pointer can contain a reference to any arbitrary memory location, a program typically only intends to use the pointer to access limited portions of memory, such as contiguous memory used to access an individual array. □ Programs may use offsets to access fields or sub-elements stored within structured data. The offset might be out-of-range if it comes from an untrusted source, is the result of an incorrect calculation, or occurs because of another error.
CWE-824	**Access of Uninitialized Pointer**	The program accesses or uses a pointer that has not been initialized. If the pointer contains an uninitialized value, then the value might not point to a valid memory location.
CWE-825	**Expired Pointer Dereference**	The program dereferences a pointer that contains a location for memory that was previously valid, but is no longer valid.
CWE-129	**Improper Validation of Array Index**	The product uses untrusted input when calculating or using an array index, but the product does not validate or incorrectly validates the index to ensure the index references a valid position within the array.
CWE-134	**Use of Externally Controlled Format String**	The software uses a function that accepts a format string as an argument, but the format string originates from an external source.
CWE-252	**Unchecked Return Value**	The software does not check the return value from a method or function, which can prevent it from detecting unexpected states and conditions.
CWE-404	**Improper Resource Shutdown or Release**	The program does not release or incorrectly releases a resource before it is made available for re-use.

CWE-401	**Improper Release of Memory Before Removing Last Reference ('Memory Leak')**	The software does not sufficiently track and release allocated memory after it has been used, which slowly consumes remaining memory.
CWE-772	**Missing Release of Resource after Effective Lifetime**	The software does not release a resource after its effective lifetime has ended, i.e., after the resource is no longer needed.
CWE-775	**Missing Release of File Descriptor or Handle after Effective Lifetime**	The software does not release a file descriptor or handle after its effective lifetime has ended, i.e., after the file descriptor/handle is no longer needed. When a file descriptor or handle is not released after use (typically by explicitly closing it), attackers can cause a denial of service by consuming all available file descriptors/handles, or otherwise preventing other system processes from obtaining their own file descriptors/handles.
CWE-424	**Improper Protection of Alternate Path**	The product does not sufficiently protect all possible paths that a user can take to access restricted functionality or resources. When data storage relies on a DBMS, special care shall be given to secure all data accesses and ensure data integrity.
CWE-434	**Unrestricted Upload of File with Dangerous Type**	The software allows the upload or transfer files of dangerous types that can be automatically processed within the product's environment.
CWE-477	**Use of Obsolete Function**	The code uses deprecated or obsolete functions, which suggests that the code has not been actively reviewed or maintained.
CWE-480	**Use of Incorrect Operator**	The programmer accidentally uses the wrong operator, which changes the application logic in security-relevant ways.
CWE-502	**Deserialization of Untrusted Data**	The application deserializes untrusted data without sufficiently verifying that the resulting data will be valid.
CWE-570	**Expression is Always False**	The software contains an expression that will always evaluate to false.
CWE-571	**Expression Is Always True**	The software contains an expression that will always evaluate to true.
CWE-606	**Unchecked Input for Loop Condition**	The product does not properly check inputs that are used for loop conditions, potentially leading to a denial of service because of excessive looping.
CWE-611	**Improper Restriction of XML External Entity Reference ('XXE')**	The software processes an XML document that can contain XML entities with URIs that resolve to documents outside of the intended sphere of control, causing the product to embed incorrect documents into its output.

CWE-643	**Improper Neutralization of Data within XPath Expressions ('XPath Injection')**	The software uses external input to dynamically construct an XPath expression used to retrieve data from an XML database, but it does not neutralize or incorrectly neutralizes that input. This allows an attacker to control the structure of the query.
CWE-652	**CWE-652 Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')**	The software uses external input to dynamically construct an XQuery expression used to retrieve data from an XML database, but it does not neutralize or incorrectly neutralizes that input. This allows an attacker to control the structure of the query.
CWE-665	**Improper Initialization**	The software does not initialize or incorrectly initializes a resource, which might leave the resource in an unexpected state when it is accessed or used.
CWE-456	**Missing Initialization of a Variable**	The software does not initialize critical variables, which causes the execution environment to use unexpected values.
CWE-457	**Use of uninitialized variable**	The software uses a variable that has not been initialized leading to unpredictable or unintended results.
CWE-662	**Improper Synchronization**	The software attempts to use a shared resource in an exclusive manner, but does not prevent or incorrectly prevents use of the resource by another thread or process.
CWE-366	**Race Condition within a Thread**	If two threads of execution use a resource simultaneously, there exists the possibility that resources may be used while invalid, in turn making the state of execution undefined.
CWE-543	**Use of Singleton Pattern Without Synchronization in a Multithreaded Context**	The software uses the singleton pattern when creating a resource within a multithreaded environment.
CWE-567	**Unsynchronized Access to Shared Data in a Multithreaded Context**	The product does not properly synchronize shared data, such as static variables across threads, which can lead to undefined behavior and unpredictable data changes.
CWE-667	**Improper Locking**	The software does not properly acquire a lock on a resource, or it does not properly release a lock on a resource, leading to unexpected resource state changes and behaviors.
CWE-820	**Missing Synchronization**	The software utilizes a shared resource in a concurrent manner but does not attempt to synchronize access to the resource.
CWE-821	**Incorrect Synchronization**	The software utilizes a shared resource in a concurrent manner but it does not correctly synchronize access to the resource.
CWE-672	**Operation on a Resource after Expiration or Release**	The software uses, accesses, or otherwise operates on a resource after that resource has been expired, released, or revoked.

CWE-415	**Double Free**	The product calls free() twice on the same memory address, potentially leading to modification of unexpected memory locations.
CWE-416	**Use After Free**	Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.
CWE-681	**Incorrect Conversion between Numeric Types**	When converting from one data type to another, such as long to integer, data can be omitted or translated in a way that produces unexpected values. If the resulting values are used in a sensitive context, then dangerous behaviors may occur. For instance, if the software declares a variable, field, member, etc. with a numeric type, and then updates it with a value from a second numeric type that is incompatible with the first numeric type.
CWE-194	**Unexpected Sign Extension**	The software performs an operation on a number that causes it to be sign-extended when it is transformed into a larger data type. When the original number is negative, this can produce unexpected values that lead to resultant weaknesses.
CWE-195	**Signed to Unsigned Conversion Error**	The software uses a signed primitive and performs a cast to an unsigned primitive, which can produce an unexpected value if the value of the signed primitive cannot be represented using an unsigned primitive.
CWE-196	**Unsigned to Signed Conversion Error**	The software uses an unsigned primitive and performs a cast to a signed primitive, which can produce an unexpected value if the value of the unsigned primitive cannot be represented using a signed primitive.
CWE-197	**Numeric Truncation Error**	Truncation errors occur when a primitive is cast to a primitive of a smaller size and data is lost in the conversion. When a primitive is cast to a smaller primitive, the high order bits of the large value are lost in the conversion, potentially resulting in an unexpected value that is not equal to the original value. This value may be required as an index into a buffer, a loop iterator, or simply necessary state data. In any case, the value cannot be trusted and the system will be in an undefined state. While this method may be employed viably to isolate the low bits of a value, this usage is rare, and truncation usually implies that an implementation error has occurred.
CWE-682	**Incorrect Calculation**	The software performs a calculation that generates incorrect or unintended results that are later used in security-critical decisions or resource management.
CWE-131	**Incorrect Calculation of Buffer Size**	The software does not correctly calculate the size to be used when allocating a buffer, which could lead to a buffer overflow.
CWE-369	**Divide By Zero**	The product divides a value by zero.

CWE-732	**Incorrect Permission Assignment for Critical Resource**	The software specifies permissions for a security-critical resource in a way that allows that resource to be read or modified by unintended actors.
CWE 778	**Insufficient Logging**	When a security-critical event occurs, the software either does not record the event or omits important details about the event when logging it
CWE-783	**Operator Precedence Logic Error**	The program uses an expression in which operator precedence causes incorrect logic to be used. While often just a bug, operator precedence logic errors can have serious consequences if they are used in security-critical code, such as making an authentication decision.
CWE-789	**Uncontrolled Memory Allocation**	The product allocates memory based on an untrusted size value, but it does not validate or incorrectly validates the size, allowing arbitrary amounts of memory to be allocated.
CWE-798	**Use of Hard-coded Credentials**	The software contains hard-coded credentials, such as a password or cryptographic key, which it uses for its own inbound authentication, outbound communication to external components, or encryption of internal data.
CWE-259	**Use of Hard-coded Password**	The software contains a hard-coded password, which it uses for its own inbound authentication or for outbound communication to external components.
CWE-321	**Use of Hard-coded Cryptographic Key**	The use of a hard-coded cryptographic key significantly increases the possibility that encrypted data may be recovered.
CWE-835	**Loop with Unreachable Exit Condition ('Infinite Loop')**	The program contains an iteration or loop with an exit condition that cannot be reached, i.e., an infinite loop.
CWE-917	**Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')**	The software constructs all or part of an expression language (EL) statement in a Java Server Page (JSP) using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended EL statement before it is executed.

CWE-1057	**Data Access Operations Outside of Expected Data Manager Component**	The software uses a dedicated, central data manager component as required by design, but it contains code that performs data-access operations that do not use this data manager. Notes: • The dedicated data access component can be either client-side or server-side, which means that data access components can be developed using non-SQL language. • If there is no dedicated data access component, every data access is a weakness. • For some embedded software that requires access to data from anywhere, the whole software is defined as a data access component. This condition must be identified as input to the analysis.

The cells containing CWE IDs for parents are presented in a dark blue.
The cells containing contributing weaknesses are presented in a light blue.

Master list of quality measure weaknesses: https://www.it-cisq.org/coding-rules/index.htm
Master list PDF: https://www.it-cisq.org/pdf/cisq-weaknesses-in-ascqm.pdf