CWE Schema Documentation

Table of Contents

The following is the table of contents for the major elements in the CWE schema. The children of these fields are located subsequently to field definition. If an item you are searching for is not indexed here, navigate to its parent and the child will be located within the section. Note, however, that "Source" has been separated from Common_Attributes and placed in section 3.

1 Common Attributes

1.1 Background and Contextual Information

1.1.1 Description

This field provides a description of this Structure, whether it is a Weakness, Category or Compound Element. Its primary subelement is Description_Summary which is intended to serve as a minimalistic description which provides the information necessary to understand the primary focus of this weakness. Additionally, it has the subelement Extended_Description which is optional and is used to provide further information pertaining to this weakness.

1.1.1.1 Description_Summary

This description should be short and should limit itself to describing the key points that define this entry. Further explanation can be included in the extended description element. This is required for all entries.

1.1.1.2 Extended_Description

This element provides a place for details important to the description of this entry to be included that are not necessary to convey the fundamental concept behind the weakness or structure. This is not required for all entries and should only be included where appropriate.

1.1.2 Alternate_Terms

This element contains other names used to describe this weakness. Each alternate name should follow the same conventions as the Weakness_Name attribute and may optionally be accompanied by some descriptive text to put the alternate name in context. For example, CWE-181 Incorrect Behavior Order: Validate Before Filter, has the alternate terms value "Validate-before-cleanse". This is not required for all entries and should only be included where appropriate.

1.1.3 Demonstrative_Example

This element is the container for all of the examples associated with this weakness entry. It can contain multiple examples and each unique example should be included in its own Example_Code element. If a single example is being presented from the perspective of several different languages, all of those examples should exist in their own Example_Block inside of one Example_Code element. This should be populated for all weakness bases and variants.

1.1.3.1 Example_Code

This element contains a single weakness example where each language perspective from which this weakness is presented is contained within its own Example_Block subelement. Each unique weakness example should be given its own Example_Code element. A single example may be broken up over multiple Example_Block subelements in order to improve clarity.

1.1.3.2 PreText

The PreText element allows the example author to provide context to the reader that will make the example in its entirety clearer. This element is intended to provide the pretext from a language independent manner, language specific pretext can be provided in the Pre_Code_Comment section.

1.1.3.3 Example_Block

The Example_Block is a container for the code and description of this weakness from the perspective of one language. Each language in which the example is being demonstrated should get its own Example_Block. If an author wants to break up a single example into separate code sections with text explaining each section before and after the code, then the author should use as many Example_Block elements as are necessary, leveraging the Pre_Code_Comment and Post_Code_Comment fields in order to provide the reader with all required information.

1.1.3.4 Pre_Code_Comment

This element provides an opportunity for the example author to provide context to the reader specific to the language being demonstrated in this example block.

1.1.3.5 Example_Code_Block

This element houses the code and the language identifier for this Example_Block.

1.1.3.6 Code_Example_Language

This element identifies the language in which this Example_Block is depicting the weakness example. Valid values are C, C++, C#, Java, JSP, Javascript, ASP.NET, SQL, Python, Perl, PHP, SOAP, Ruby, Shell, PsuedoCode, All Interpreted Languages, All, .NET, and Assembly.

1.1.3.7 Code_Block

This is the element which contains the code for this depiction of the example. The code should be encapsulated by CDATA tags of the format:
<![CDATA[INSERT CODE HERE]]>.

1.1.3.8 Post_Code_Comment

This element provides an opportunity for the example author to provide an explanation of the code just presented to the reader. If more code is necessary for the same example, the

author can add another Example_Block and continue to present code and describe it in order to clearly present the example to the reader.

1.1.3.9 PostText

The PostText element allows the author to describe the preceding Example_Block(s) from a language independent perspective. Language specific explanations should be confined to the Post_Code_Comment since the example may be presented in multiple languages.

1.1.3.10 Demonstrative_Example_Reference

This element provides the example author with an opportunity to claim credit for the portions of the example for which he/she is responsible. It is free text, so multiple authors should be entered inside of a single Reference element.

1.1.4 Observed_Examples

The Observed_Examples element contains all references to specific observed instances of this weakness in the real world; typically this will be a CVE reference.

1.1.4.1 Observed_Example

This element specifies a reference to a specific observed instance of this weakness in the real-world; typically this will be a CVE reference. Each Observed_Example element represents a single example. Multiple, consecutive examples may exist for any weakness entry. This element should be filled out for as many entries as possible.

1.1.4.2 Observed_Example_Reference

This field should contain the identifier for the example being cited. For example, if a CVE is being cited it should be of the standard CVE identifier format, such as CVE-2005-1951 or CVE-1999-0046.

1.1.4.3 Observed_Example_Description

This field should contain a product independent description of the example being cited. The description should present an unambiguous correlation between the example being described and the weakness which it is meant to exemplify.

1.1.4.4 Observed_Example_Link

This field should provide a valid URL where more information regarding this example can be obtained.

1.1.5 Context_Notes

This element contains broader contextual descriptions of this weakness or structure. It should be filled out in any entry where additional context will be helpful to understanding the weakness or weaknesses involved.

1.1.6 Source_Taxonomy

This structure describes the specific taxonomy that was a source for the creation of this node. A weakness may be the result of derivations from multiple taxonomies, in which case multiple instances of the Source_Taxonomy element should exist sequentially. This should be filled out for most weaknesses.

1.1.6.1 Original_Node_Name

This element contains the original name of this weakness in the source taxonomy.

1.1.6.2 Source_Taxonomy_Name

This element contains the name of the taxonomy from which this weakness was derived.

1.1.7 Taxonomy_Mapping

This structure describes mappings to nodes of other taxonomies that are equivalent in meaning to this node. Although this may sound similar to Source_Taxonomy, Source_Taxonomy is designed to provide a history and pedigree for the entry, whereas Taxonomy_Mapping allows similar nodes in other collections to be identified as matching concepts with this weakness. For example, Taxonomy_Mapping should be used to map the CWE entries to their OWASP Top 10 equivalents. The sole attribute is "Mapped_Taxonomy_Name" which is used to identify the taxonomy to which this weakness is being mapped.

1.1.7.1 Mapped_Node_Name

This element identifies the name of the entry to which this weakness is being mapped in taxonomy Mapped_Taxonomy_Name.

1.1.7.2 Mapped_Taxonomy_Name

This attribute identifies the taxonomy to which this weakness has a similar or equivalent entry.

1.1.8 References

The References element contains one or more Reference elements, each of which provide further reading and insight into this weakness. This should be filled out for all weakness bases and some variants.

1.1.8.1 Reference

Each Reference subelement should provide a single source from which more information and deeper insight into this weakness can be obtained, such as a research paper or an excerpt from a publication. Multiple Reference subelements can exist for any weakness. The sole attribute of this element is the id. The id is optional and translates to a preceding footnote below the context notes if the author of the entry wants to cite a reference. Not all subelements need to be completed since some are designed for web references and

other are designed for book references. The fields Reference_Author and Reference_Title should be filled out for all references if possible. Reference_Section and Reference_Date can be included for either book references or online references. Reference_Edition, Reference_Publication, Reference_Publisher, and Reference_PubDate are intended for book references, however they can be included where appropriate for other types of references. Reference_Link is intended for web references, however it can be included for book references as well if applicable.

1.1.8.2 Reference_Author

This element identifies an individual author of the material being referenced. It is not required, but may be repeated sequentially in order to identify multiple authors for a single piece of material.

1.1.8.3 Reference_Title

This element identifies the title of the material being referenced. It is not required if the material does not have a title.

1.1.8.4 Reference_Section

This element is intended to provide a means of identifying the exact location of the material inside of the publication source, such as the relevant pages of a research paper, the appropriate chapters from a book, etc. This is useful for both book references and internet references.

1.1.8.5 Reference_Edition

This element identifies the edition of the material being referenced in the event that multiple editions of the material exist. This will usually only be useful for book references.

1.1.8.6 Reference_Publication

This element identifies the publication source of the reference material, if one exists.

1.1.8.7 Reference_Publisher

This element identifies the publisher of the reference material, if one exists.

1.1.8.8 Reference_Date

This element identifies the date when the reference was included in the entry. This provides the reader with a time line for when the material in the reference, usually the link, was valid. The date should be of the format YYYY-MM-DD.

1.1.8.9 Reference_PubDate

This field describes the date when the reference was published YYYY.

1.1.8.10 Reference_Link

This element should hold the URL for the material being referenced, if one exists. This should always be used for web references, and may optionally be used for book and other publication references.

1.1.8.11 Reference_ID

The id attribute is optional and is used as a mechanism for citing text in the entry. If an id is provided, it is placed between brackets and precedes this reference and the matching id should be used inside of the text for the weakness itself where this reference is applicable. All reference ids assigned within an entry must be unique.

1.1.9 White_Box_Definition

This element describes the weakness from a white box perspective, meaning that the view includes the knowledge of control flow, data flow, and all other inner workings of the software in which the weakness exists.

1.1.10 Black_Box_Definition

This element describes the weakness from an external perspective, meaning that the view includes no knowledge of how the software is processing data other than what can be inferred from observing the software's behavior.

1.2 Scope and Delimiting Factors

1.2.1 Functional_Area

This element identifies the functional area of the software in which the weakness is most likely to occur. For example, CWE-178 Failure to Resolve Case Sensitivity is likely to occur in functional areas of software related to file processing and credentials. The list of applicable functional areas should be comma delimited and standard title capitalization should be applied to each area.

1.2.2 Likelihood_of_Exploit

This element contains information regarding how likely a weakness is to be exploited if exposed. Appropriate values are one of Low, Medium, or High. This should be filled in for all weakness bases and variants.

1.2.3 Common_Consequences

This element contains the common consequences associated with this weakness. It is populated by individual Common_Consequence subelements. This should be included and completed as much as possible for all weaknesses.

1.2.3.1 Common_Consequence

This subelement contains an individual consequence associated with this weakness. One or more are required for every Common_Consequences element and should exist in all weaknesses.

1.2.4 Enabling_Factors_for_Exploitation

Conditions or factors that could increase the likelihood of exploit for this weakness. This element should contain free text with enough detail to make the enabling factors clear. This should be filled out for most weakness bases.

1.2.5 Detection_Factor

The Detection_Factor element is intended to provide information on factors which may hide the weakness or make the weakness more difficult to detect. This should be filled out for some weakness classes and bases.

1.2.6 Applicable_Platforms

This element contains a list of the individual Platform subelements which each represent a platform on which this weakness is likely to exist. This element should be included for all weaknesses.

1.2.6.1 Platform

This subelement identifies an individual platform on which this weakness is likely to exist. Valid values are:

- C
- C++
- C#
- Java
- JSP
- Javascript
- ASP.NET
- SQL
- Python
- Perl
- PHP
- SOAP
- Ruby
- Shell
- PsuedoCode
- All Interpreted Languages
- All
- .NET

- Assembly

Multiple Platform elements may be included inside of the Applicable_Platforms element.

1.2.7 Time_of_Introduction

This element identifies the point of time in the software life cycle at which the weakness may be introduced. Possible values are Policy, Requirements, Architecture and Design, Implementation, Testing, Bundling, Distribution, Installation, Patch, Documentation, Porting, System Configuration, and Operation. If there are multiple points of time at which the weakness may be introduced, then separate Time_of_Introduction elements should be included for each. This element should be populated for all weakness bases and variants.

1.3 Prescriptions and Recommendations

1.3.1 Potential_Mitigations

This element contains the potential mitigations associated with this weakness. It contains one or more mitigation subelements which each represent individual mitigations for this weakness. This should be included and completed to the extent possible for all weakness bases and variants.

1.3.1.1 Mitigation

This subelement contains a single method for mitigating this weakness. 1 or more mitigations must be included inside of a Potential_Mitigations element.

1.4 Supplemental Information

1.4.1 Weakness_Ordinality

This element describes whether this weakness exists independent of other weaknesses (Primary) or whether it is the result of the presence of some other weaknesses (Resultant). This should be filled out for all weakness bases and variants.

1.4.2 Causal_Nature

This element describes the nature of the underlying cause of the weakness. Is it an implicit underlying weakness or is it an issue of behavior on the part of the software developer? Appropriate values are either Implicit or Explicit, each accompanied by the text as required in the schema.

1.4.3 Affected_Resource

This element identifies system resources affected by this entry. Each resource affected by this weakness should be given its own Affected_Resource element. For example, CWE-249, Path Manipulation has both Memory and File/Directory listed in separate Affected_Resource elements. This should be filled out in weakness bases and variants where applicable.

1.4.4 Related_Attack_Patterns

The Related_Attack_Patterns element contains all references to CAPEC which will identify related attack patterns to this weakness. It has one or more Related_Attack_Pattern elements as children and each child will point to a single CAPEC entry which is associated with this weakness. This should be filled out to the extent possible for most weaknesses.

1.4.4.1 Related_Attack_Pattern

The Related_Attack_Pattern subelement identifies a single attack pattern that is associated with this weakness. Its only child, CAPEC_ID is required and identifies the related CAPEC entry. More than one Related_Attack_Pattern element can exist, but they must all be contained within a single Related_Attack_Patterns element.

1.4.4.2 CAPEC_ID

The CAPEC_ID stores the value for the related CAPEC entry identifier as a string. Only one CAPEC_ID element can exist for each Related_Attack_Pattern element.

1.4.5 Research_Gaps

Brief descriptions of opportunities for further research related to this weakness. This should be filled out where appropriate for weaknesses and categories.

1.4.6 Relevant_Properties

This structure contains one or more Relevant_Property elements. Each Relevant_Property element identifies a property that is required by the code or a resource in order to function as specified. Correctly labeling all of the relevant properties can help to figure out what the root cause of a vulnerability might be.

1.4.4.2 Relevant_Property

Each Relevant_Property element identifies a property that is required by the code or a resource in order to function as specified. Correctly labeling all of the relevant properties can help to figure out what the root cause of a vulnerability might be.

1.4.7 Relationships

The Relationships structure contains one or more Relationship elements, each of which identifies an association between this structure, whether it is a Weakness, Category, or Compound_Element and another structure.

1.4.7.1 Relationship

Each Relationship identifies an association between this structure, whether it is a Weakness, Category, or Compound_Element and another structure. The relationship also identifies the views under which the relationship is applicable.

1.4.7.1.1 Relationship_Views

This element contains a list of the individual Views to which this relationship pertains.

1.4.7.1.1.1 Relationship_View_ID

Specifies the unique ID of an individual view element to which this relationship pertains.

1.4.7.1.2 Relationship_Chains

This element contains a list of the individual Chains to which this relationship pertains.

1.4.7.1.2.1 Relationship_Chain_ID

Specifies the unique ID of an individual chain element to which this relationship pertains.

1.4.7.1.3 Relationship_Type

The Relationship_Type element defines the target type of this relationship, such as Category, Weakness, View or Compound_Element.

1.4.7.1.4 Relationship_Nature

The Relationship_Nature element defines the nature of the relationship between this element and the target element.

1.4.7.1.5 Relationship_Target_ID

The Relationship_Target_ID specifies the unique ID of the target element of the relationship.

2 Structural Schema Elements

2.1 Weakness_Catalog

The Weakness_Catalog structure represents a collection of software security issues (flaws, faults, bugs, vulnerabilities, weaknesses, whatever). The name used by CWE is usually "CWEC", however if this collection is a subset of CWE then a more appropriate name should be used.

2.1.1 Views

The Views structure contains zero or more View elements. Each View element represents a perspective with which one might look at the weaknesses in CWE. CWE-630 Weaknesses Examined by SAMATE and CWE-658 Weaknesses found in the C Language are two examples of Views.

2.1.1.1 View

Each View element represents a perspective with which one might look at the weaknesses in CWE. CWE-630 Weaknesses Examined by SAMATE and CWE-658 Weaknesses found in the C Language are two examples of Views.

2.1.1.1.1 View_ID

The View_ID is the unique identifier assigned to this view which allows relationships to specify under which views they are applicable.

2.1.1.1.2 View_Name

The View_Name is a descriptive named used to give the reader an idea of what perspective this view represents.

2.1.1.1.3 View_Status

The View_Status attribute defines the status level for this view.

2.1.2 Categories

The Categories structure contains zero or more Category elements. Each Category element represents what used to be referred to in CWE as a "Grouping" entry. That is, a category is now a collection of weaknesses based on a common attribute, such as CWE-310 Cryptographic Issues or CWE-355 User Interface Security Issues.

2.1.2.1 Category

Each Category element represents what used to be referred to in CWE as a "Grouping" entry. That is, a category is now a collection of weaknesses based on a common attribute, such as CWE-310 Cryptographic Issues or CWE-355 User Interface Security Issues.

2.1.2.1.1 Category_ID

The Category_ID is the unique identifier assigned to this category which allows other categories and weaknesses to refer to it.

2.1.2.1.2 Category_Name

The Category_Name is a descriptive name used to give the reader an idea of what the commonality is amongst the children of this category.

2.1.2.1.3 Category_Status

The Category_Status attribute defines the status level for this category.

2.1.3 Weaknesses

The Weaknesses structure contains zero or more Weakness elements. Each Weakness element represents an actual weakness entry in CWE, such as CWE-311 Failure to Encrypt Sensitive Data or CWE-326 Weak Encryption.

2.1.3.1 Weakness

Each Weakness element represents an actual weakness entry in CWE, such as CWE-311 Failure to Encrypt Sensitive Data or CWE-326 Weak Encryption.

2.1.3.1.1 Weakness_ID

This attribute provides a unique identifier for the entry. It will be static for the lifetime of the entry. In the event that this entry becomes deprecated, the ID will not be reused and a pointer will be left in this entry to the replacement. This is required for all Weakness entries.

2.1.3.1.2 Weakness_Name

This attribute is the string which identifies the entry. The name should focus on the weakness being described in the entry and should avoid focusing on the attack which exploits the weakness or the consequences of exploiting the weakness. All words in the entry name should be capitalized except for articles and prepositions unless they begin or end the name. Sequential words in a hyphenated chain are also not capitalized. This is required for all Weakness entries.

2.1.3.1.3 Weakness_Abstraction

The Weakness_Abstraction attribute defines the abstraction level for this weakness. Acceptable values are "Class", which is the most abstract type of Weakness such as CWE-362 Race Conditions, "Base" which is a more specific type of weakness that is still mostly independent of a specific resource or technology such as CWE-567 Unsynchronized Access to Shared Data, and "Variant" which is a weakness specific to a particular resource, technology or context.

2.1.3.1.4 Weakness_Status

The Weakness_Status attribute defines the status level for this weakness.

2.1.4 Compound_Elements

The Compound_Elements structure contains zero or more Compound_Element elements. Each Compound_Element represents a meaningful aggregation of several weaknesses, as in a chain like CWE-690 Unchecked Return Value to NULL Pointer Dereference or as in a composite like CWE-352 Cross-Site Request Forgery.

2.1.4.1 Compound_Element

Each Compound_Element represents a meaningful aggregation of several weaknesses, as in a chain like CWE-690 Unchecked Return Value to NULL Pointer Dereference or as in a composite like CWE-352 Cross-Site Request Forgery.

2.1.4.1.1 Compound_Element_ID

The Compound_Element_ID is the unique identifier assigned to this Compound_Element which allows other categories and weaknesses to refer to it.

2.1.4.1.2 Compound_Element_Name

The Compound_Element_Name is a descriptive name used to give the reader an idea of the meaning behind the compound weakness structure.

2.1.4.1.3 Compound_Element_Abstraction

The Compound_Element_Abstraction defines the abstraction level for this weakness. The abstractions levels for Compound_Elements and Weaknesses are the same. For example, if the Compound_Element is a chain, and all elements of the chain are Class level, then the Compound_Element Abstraction attribute is Class.

2.1.4.1.4 Compound_Element_Structure

The Compound_Element Structure attribute defines the structural nature of this compound element (e.g. composed of other weaknesses concurrently, as in a composite, or consecutively, as in a chain).

2.1.4.1.5 Compound_Element_Status

The Compound_Element_Status attribute defines the status level for this compound element.

2.1.5 Catalog_Name

Catalog_Name is a required attribute of Weakness_Catalog which identifies the collection of Weaknesses, Views, Categories and Compound_Elements represented by this XML document.

2.1.6 Catalog_Version

Catalog_Version is a required attribute of Weakness_Catalog which identifies what version of @Catalog_Name this XML document represents.

3 Administrative Information

3.1 Source

This element is used to keep track of the author of the weakness entry and anyone who has made modifications to the content. This provides a means of contacting the authors and modifiers for clarifying ambiguities, merging overlapping contributions, etc. This should be filled out for all entries.

3.1.1 Submission

This element houses the subelements which identify the submitter and his comments related to this entry. This element has a single attribute, Submission_Type, which tells the CWE team whether or not this submission information should be kept in the public XML or stripped from the XML and kept only internally to the MITRE team.

3.1.2 Submitter

This element should contain the name of the author for this entry.

3.1.3 Submitter_Organization

This element should identify the author's organization.

3.1.4 Submission_Date

This element should provide the date on which this content was authored in YYYY-MM-DD format.

3.1.5 Submission_Comment

This element provides the author with a place to store any comments regarding the content of this weakness entry, such as assumptions made, reasons for omitting elements, contact information, pending questions, etc.

3.1.6 Submission_Type

This attribute identifies whether the submission information is for the CWE team only or if it should be kept in the public XML. Valid values are “Internal” and “External” for stripped from the public XML and left in the public XML respectively.

3.1.7 Modification

This element houses the subelements which identify the modifier and his comments related to this entry. A new Modification element should exist for each modification of the entry content. This element has a single attribute, Modification_Type, which tells the CWE team whether or not the modifier's information and comments should be kept in the public XML or stripped from the XML and kept only internally to the MITRE team.

3.1.8 Modifier

This element should contain the name of the person modifying this entry.

3.1.9 Modifier_Organization

This element should contain the modifier's organization.

3.1.10 Modification_Date

This element should contain the date of the modifications.

3.1.11 Modification_Comment

This element provides the modifier with a place to store any comments regarding the content of this weakness entry, such as assumptions made, reasons for omitting elements, contact information, pending questions, etc.

3.1.12 Modification_Type

This attribute identifies whether the modifier's information is for the CWE team only or if it should be kept in the public XML. Valid values are Internal and External for stripped from the public XML and left in the public XML respectively.