Building an Ontology for CWE from the Point of View of Architectural Concept

Zhanar Sartabanova[1], Vladimir Dimitrov[2] [0000-0002-7441-253X], Saule Sarsimbaeva[1], Gulmira Urdabayeva[1]

[1] Aktobe Regional University "K. Zhubanov", Aktobe, Kazakhstan
[2] Sofia University "St. Kliment Ohridski", Sofia, Bulgaria

zhanarsartabanova@gmail.com

Abstract. The article deals with the issues of constructing an ontology for software weaknesses (CWE) from the point of view of the architecture concept. Analyzed are all weaknesses from the architectural concept, the weakness structure, and the process of designing the ontology using the environment Protégé.

Keywords: CWE, software weaknesses, ontology, software architect, Protégé, SPARQL.

1 Introduction

Software weakness is any mistake (defect, malfunction, and vulnerability) made during the design, coding or configuration of the software, which, if left uncorrected, can be exploited as vulnerability.

Examples of software weaknesses are buffer overflow, channel and path errors; error handling; user interface errors; path traversal and equivalence errors; authentication errors; resource management errors; insufficient data validation; code evaluation etc. for more details see [1]

CWE (Common Weakness Enumeration) has been developed by MITRE Corporation to serve as a common language describing software security weaknesses; to serve as a standard measuring rod for software defenses targeting these weaknesses; and to provide a common baseline standard for identifying, mitigating, and preventing the known weaknesses.

CWE is supported by a community that includes representatives from the major software vendors, commercial information security vendors, academia, government agencies, and research institutions. Community members participate in the development of CWE using CWE Community Research mailing list. This means that the CWE dictionary, as well as its subsequent efforts in the field as CWSS [2] and CWRAF [3], reflect the understanding and combined experience of a wide range of information technology and security professionals.

2 Common Weakness Enumeration

There is no single taxonomy in CWE, instead several point of views are used to classify weaknesses. There are three main views based on the usage concepts:

- Software Development view covers all aspects of the software development lifecycle, including architecture and implementation.
- Hardware Design view contains hardware weaknesses.
- Research Concepts view is intended to facilitate the study of weaknesses, including their relationships.

There are views that are more specialized in CWE. They are not listed as main ones and are not sub views of the above mentioned.

One of the most compact and well organized the architecture concept view that is a subject of this study.

The architectural concept organizes the weak points in accordance with the general strategy for architectural security. It is designed to help architects to identify potential errors in the software architecture development.

The architectural concept (CWE 1008) contains 12 categories, each consisting of classes, bases, variants, and composites. A brief description of each of its categories is given below:

- Audit (CWE 1009) is on audit-based system components.
- Authenticate Actors (CWE 1010) is on system's authentication components.
- Authorize Actors (CWE 1011) is on authorization components of the system.
- Cross Cutting (CWE 1012) is on the numerous cross cutting security attack tactics and their impact on the system.
- Encrypt Data (CWE 1013) is on data privacy in the system.
- Identify Actors (CWE 1014) is on identity management subsystem.
- Limit Access (CWE 1015) is on system resource usage.
- Limit Exposure (CWE 1016) is on the system entry points.
- Lock Computer (CWE 1017) is on the system locking mechanism.
- Manage User Sessions (CWE 1018) is on session management.
- Validate Inputs (CWE 1019) is on the input validation components of the system.
- Verify Message Integrity (CWE 1020) is on the system's data integrity components.

The View in CWE is a subset of CWE entries that are organized by some concept. There are two view structures are parts/slices (flat lists) and graphs (containing relationships among the entries).

The Category in CWE contains a set of other entries that share common characteristics.

The Class in CWE is a weakness that is described very abstractly, usually independently of any particular language or technology.

The Base in CWE is a weakness that is described abstractly, but with enough details to be detected. This entry description is accompanied with recommendations how to be prevented or mitigate it.

The Variant in CWE is a weakness that is described in very low level of details, usually limited to a specific language or technology.

The Composite in CWE is a composite entry that combines of two or more other weaknesses that have to be available the weakness to occur. Eliminating any of the weaknesses eliminates or dramatically reduces the risk.

3 Structure of CWE entries

In general, all CWE entries have the same structure by its XML schema, but in reality, it is not true. The view is structured as follows:

1. view ID – each CWE entry has a unique number;
2. type (graph or list);
3. audience that can benefits from the view;
4. relationships with other CWE entries;
5. notes;
6. recommendations;
7. content of the view;
8. history;
9. changes.

CWE entries that are not views are structured as follows:

- name of the weakness;
- description;
- alternative terms for the weakness;
- behavior description of this weakness;
- weakness exploit description;
- probability for exploit;
- description of exploit consequences;
- potential mitigation of the exploit;
- relationships with other CWEs;
- source taxonomies;
- code examples for languages / architectures
- IDs of CVE vulnerabilities for the weakness;
- links.

4 The CWE ontology

CWE ontology has been specified in OWL 2 Manchester Syntax [6] and developed in Protégé [5].

The ontology has been developed in the next steps:

- Classes and subclasses have been designed.
- The class CWEEntries has been instantiated with individuals.
- The class CVEEntries has been instantiated with individuals.
- The class CAPECEntries has been instantiated with individuals.
- The ontology has been tested.

In the ontology, under the class Thing are placed: CWEEntries – the main class under study, CAPECEntries, and CVEEntries. CWEEntries contains CWE weaknesses for the architectural view, CAPECEntries – CAPEC attack patterns, and CVEEntries – CVE vulnerabilities. The last two classes are only sketched for the integrity purpose only.

CWEEntries has as subclasses Views, Classes, Bases, Variants, and Composites that corresponds to the CWE entries types. There is one instance of class Views. It corresponds to the architectural concept view in CWE, which has 12 instances of class Categories. In Fig. 1 is presented the class hierarchy.

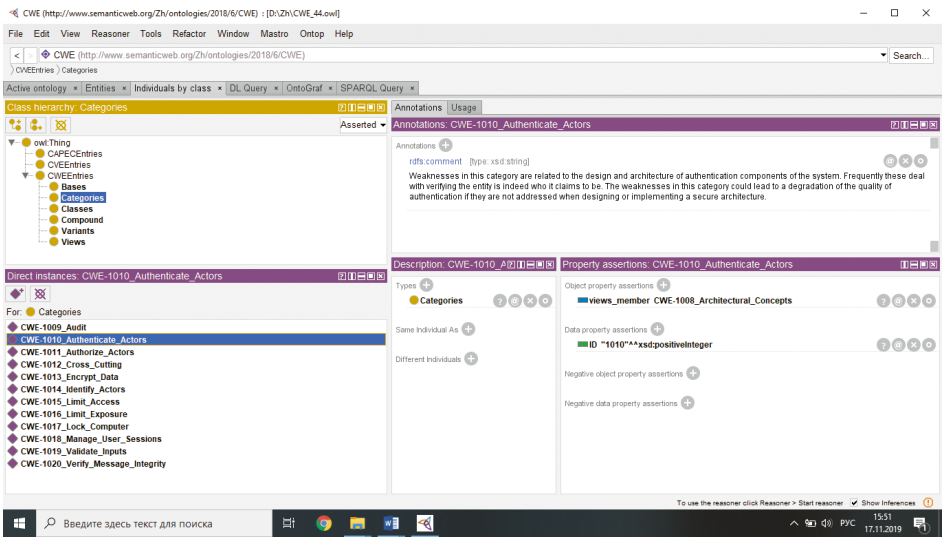

Fig. 1. Class hierarchy.

The object properties are shown in Fig. 2. These are:

- references refers to CVE vulnerabilities of this CWE weakness;
- categories_member links each base, variant, class, compound with its categories;

- categories_includes is inverse of categories_member;
- related_attack_patterns links the weaknesses with CAPEC attack pattern;
- views_includes links the view with its contents;
- views_member is inverse of views_includes.

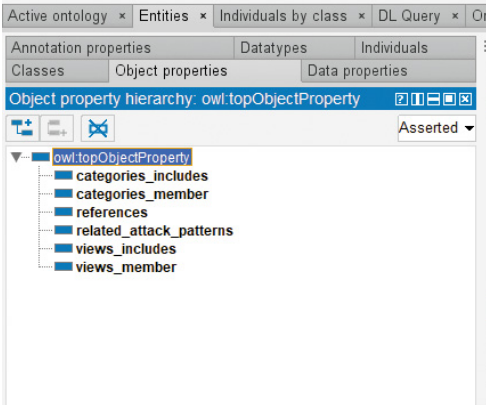

Fig. 2. Object properties.

The data properties are shown in Fig. 3. They are organized in accordance with CWE XML Schema. These are:

- ID – CWE identifier;
- applicable_platforms – on which platforms the CWE is available:
  - class;
  - languages;
  - operating_system;
  - paradigms;
  - technologies.
- common_consequences –consequences associated with the weakness: access_control;
  - accountability;
  - authentication;
  - availability;
  - confidentiality;
  - integrity;
  - non-repudiation;
  - other.
- demonstrative_examples – examples of the weakness.
- detection_methods for weakness discovery:
  - architecture_or_design_review;
  - automated_analyses;

- Black_Box;
- dynamic;
- manual_analyses;
- static.

- likelihood_of_exploit – the probability for exploit.
- modes_of_introduction – the phase in which the weakness can be introduced:
  - architecture_and_design;
  - buld_and_compilation;
  - distribution;
  - documentation;
  - implementation;
  - installation;
  - operation;
  - policy;
  - requirements;
  - system_configuration;
  - testing.
- weakness_ordinalities – order of weakness in composite weaknesses.

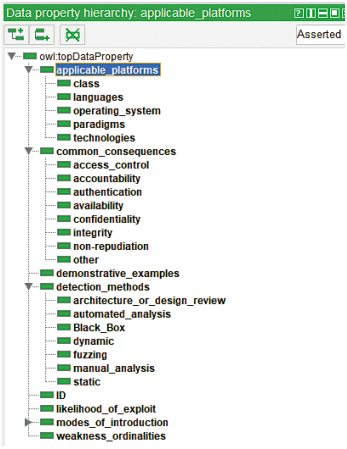

Fig. 3. Data properties.

5 Conclusion

In this paper, CWE weaknesses have been briefly analyzed. An ontology based on the architectural view has been presented.

This ontology is useful for software architects, developers, researchers in the field of software design and cybersecurity, as well for lecturers of on software

development technology and information security. For example, for the developers, this ontology can serve as an assistant and handbook for secure software design, since weaknesses are organized by known security strategies, helping the designer to embed security during the design process instead of detecting weaknesses after the software has been created.

6 Acknowledgements

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

This research is supported by the National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", financed by the Ministry of Education and Science.

References

1. MITRE Corporation, Common Weakness Enumeration (CWE), https://cwe.mitre.org, last accessed 2021/01/22.
2. MITRE Corporation, Common Weakness Scoring System (CWSS), https://cwe.mitre.org/cwss/cwss_v1.0.1.html, last accessed 2021/01/22.
3. MITRE Corporation, Scoring CWEs, Common Weakness Risk Analysis Framework (CWRAF), https://cwe.mitre.org/cwraf, last accessed 2021/01/22.
4. CWE VIEW: Architectural Concepts, https://cwe.mitre.org/data/definitions/1008.html, last accessed 2021/01/22
5. Musen, M.A. The Protégé project: A look back and a look forward. AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), June 2015. DOI: 10.1145/2557001.25757003.
6. W3C, OWL 2 Web Ontology Language, Manchester Syntax (Second Edition), https://www.w3.org/TR/owl2-manchester-syntax, last accessed 2021/01/22.