Modelling of CWEs on the CWE-287 example

Zhanar Sartabanova¹, Vladimir Dimitrov²

¹ K. Zhubanov ARSU, 34 A. Moldagulova Prospect, 030000 Aktobe, Kazakhstan
zhana.85@mail.ru
² University of Sofia, 5 James Bourchier Blvd., 1164 Sofia, Bulgaria
cht@fmi.uni-sofia.bg

Abstract. Representing of software weaknesses in a knowledge base is an approach to support software developers in their effort to write secure code. The leading idea of this research is to represent the knowledge in CWEs as knowledge in OWL.

Keywords: software weaknesses, CWE, knowledge base, OWL.

1 Introduction

The software systems are part of our everyday life. The vulnerability risk of these systems increases with the number of the users. The weaknesses can be exposed as vulnerabilities during the system exploitation, but they are introduced during the implementation phase. That is why it is so important to prevent software weaknesses during the implementation.

The developer’s community established Mitre Corporation to create an international list of software weaknesses – CWE.

CWE (Common Weakness Enumeration - a common list of weaknesses) [1] is an international list of software weaknesses, which is freely available for public use. The use of this list ensures the creation of enhanced software, taking into account existing weaknesses.

Software weakness is any mistake made during the implementation, configuration, or design of the software. Software weakness is a defect, malfunction that if left uncorrected may be exposed as a system vulnerability for direct, or network attacks.

3 Construction of CWE-287 ontology

The common list of software weaknesses of CWE is a formal list of types of weaknesses that is created to aim the software developers and practitioners in the field of cybersecurity. CWE-287 is one of these enumerated weaknesses.

The knowledge contained as a text in CWE-287 is represented as OWL knowledge in Protégé.

Protégé is a free, open ontology editor and knowledge base building framework.
Protégé has an open, easily expandable architecture with support for modules that extend functionality.

Protégé is supported by a large community of developers and scientists, government and corporate users, who use it for knowledge-related tasks in such diverse fields as Biomedicine, knowledge gathering and corporate modeling. [2]

The class hierarchy of CWEs is shown in Fig 1.

![CWE class hierarchy](image)

**Fig. 1.** CWEs class hierarchy.

CWE entries are Bases, Classes, Compound, and Variations.

The most abstract type of weakness is the Class. It is independent of any particular language or technology.

Base is a more specific type of weakness. It is still largely independent of the resource or technology, but contains enough details to have specific detection and prevention methods.

Variant is a weakness that is described at a very low level of details. It is usually limited to a particular language or technology.

Compound is a significant collection of several weaknesses, organized in a chain or in a composite.

All other CWEEntries represented in Fig. 1 describe the CWE environment. CWE entry “Views” represents a conceptual point of view to the CWEs. It is a set of other CWEEntries. A view can contains Categories at its higher level, but it is not obligatory.

Category is a CWE entry that contains a set of other entries that share a common characteristic.

In our case CWE-1008 VIEW: Architectural Concepts is used. It contains the following categories:
Fig. 2. CWE-1008 VIEW: Architectural Concepts.

Categories contains Bases, Classes, Compound, and Variations.
The ontology represented as a graph is given in Fig. 3.
The weakness properties are:
1. Description (Description)
2. The design phase during which this weakness can occur (architecture and design, implementation) (Modes Of Introduction)
3. Applicable platforms (based on programming languages): discusses weaknesses that may occur in applications developed in the Java programming language). (Applicable Platforms)
4. General effects (individual effects associated with weakness). Scope identifies the security scope of an application that is compromised, while
Impact describes the negative technical impact that occurs if an attacker manages to exploit the vulnerability. Probability provides information about how likely it is that a particular consequence is expected with respect to other consequences in the list. For example, there may be a high probability that a weakness will be used to achieve a certain impact, but a low probability that it will be used to achieve another impact. (Common Consequences)

Fig. 3. OntoGraph representation of architectural concepts.

Fig. 4. Object properties of CWE-1010.
1. The possibility of using
2. Illustrative example
3. Related vulnerabilities
4. Potential Mitigation
5. Detection method

Almost every CWE is associated with vulnerabilities and attacks. A view of CWE-1010 object properties is given at Fig. 4. Finally, CWE-287 is shown at Fig. 5 as graph in its environment.

Fig. 5. CWE-287 graph.

5 Conclusion

Proposed knowledge representation aims software developers to write secure code offering to them different views to the possible weaknesses in all development phases.

References