**An Ontology-based Approach to Model Common Vulnerabilities and Exposures in Information Security**

(1) **Vulnerability**: Vulnerability refers to the flaws, defects, or mistakes in software that can be directly used by a hacker to gain access to a system or network. CVE considers a mistake a vulnerability if it allows an attacker to use it to violate a reasonable security policy for that system. This excludes those "open" security policies in which all users are trusted, or where there is no consideration of risk to the system. In order to better satisfy the properties of taxonomy [28] and our design objectives, we choose the CWE research view, a classification system in CWE, to be our vulnerability classification scheme. We take each weakness in CWE research view as a type of vulnerability, and thus a CVE vulnerability will become the instance of one or more types of vulnerabilities.

(2) **Introducation\_Phase**: Introduction\_Phase refers to the phases in the software development life cycle (SDLC), such as: requirements specification, design, coding, testing, integration, deployment, maintenance, etc, during which the vulnerability can be introduced. The taxonomy of this concept is based on the flaws by time of introduction from NRL taxonomy [30]. For instance, the vulnerabilities of the type “CWE-399: Resource Management Errors” can be introduced in the phase of implementation in the SDLC. With further details of this concept, the relationship between vulnerability and development lifecycle can be further investigated.

(3) **Active\_Location**: Active\_Location refers to the locations of the software system where the flaw manifests itself. For instance, one active location could be the system configuration files, where the vulnerability will be active during the system initialization. The taxonomy of this concept is based on the “flaws by location” from the NRL taxonomy. For example, the vulnerabilities of the type “CWE-399: Resource Management Errors” can be activated in the memory management module of the operating system.

(4) **IT\_Product**: IT\_Product is the concept that subsumes an enumeration of IT products encoded in CVE vulnerability descriptions. Each instance of IT\_Product can be reflected to an external entity of IT system in the real word. We differentiate the software products from the hardware products, and divide the software products into two types – operating systems and applications.

(5) **IT\_Vendor**: IT\_Vendor is the supplier of the IT\_Product, who produces the instances of software or hardware products. The vendor can be a commercial IT company, an open source project, an academic institution, or an individual programmer. IT\_Product together with IT\_Vendor are the concepts that compose the targeted IT system that are described in the CVE vulnerability <desc> element. For example, the targeted IT system in CVE-2008-5044 is Microsoft Windows Server 2003 and Vista. We construct both concepts based on the Common Platform Enumeration (CPE) [29], which is a structured naming scheme for information technology systems, platforms, and packages, in order to provide an unambiguous naming for each targeted IT system. The CPE name structure is listing as follows:

**cpe:/{part}:{vendor}:{product}:{version}:{update}:{edition}:{language}**

The underlying idea is that best practices have greater utility when all participants share common names for the entities described and use of consistent and meaningful names can speed application development, foster interoperability, improve correlation of test results, and ease gathering of metrics. Figure 3 below uses the targeted IT systems in CVE-2008-5112 to present design of the structure of these two concepts. Here Windows\_Server\_2003\_SP1, Windows\_Server\_2003\_SP2 and Windows\_2000\_SP4 in Figure 3 are all instances of Windows products, while Microsoft is an IT vendor which is an instance of **Commercial\_Company**. The relationship “**producedBy**” is used to express the production relationship between an IT product and its vendor.

(6) **Attack**: Attack refers to a description of a common approach that can be taken by attackers to compromise the IT system through exploiting the vulnerabilities. In our ontology, we design the attack classification scheme based on the effort of Common Attack Pattern Enumeration and Classification (**CAPEC**) which divides the attacks into eleven categories including Abuse of Functionality, Spoofing, Exploitation of Authentication, Injection, etc. An attack pattern in CAPEC is identified by CAPEC ID. We take a single attack pattern as a type of attacks. For example, the vulnerability “CVE-2008-5070” is an instance of SQL Injection (CWE-89), and the attack that can exploit this vulnerability can be an instance of SQL Injection Attack (CAPEC-66).

(7) **Attack\_Intent**: Attack\_Intent is the concept to describe the generalized purpose of the attack. This class is comprised of several subclasses, each of which represents one type of purposes, including Exploitation, Penetration, and Obfuscation.

(8) **Attack\_Method**: Attack\_Method is the concept to describe the methods used in the attacks. This class is comprised of several subclasses, each of which represents one type of attack methods, including Injection, Spoofing, Protocol\_Manipulation Analysis, Modification\_of\_Resources, API\_Abuse, Social\_Engineering, Time\_and\_State , Brute\_Force.

(9) **Attacker**: Attacker is the concept to describe the human or agents that conduct the attack by exploiting the vulnerability to cause the consequences. This class is comprised of several subclasses, each of which represents one type of attackers, including Remote\_Attacker, Local\_Attacker. For example, CVE-2008-5070 can be exploited by an instance of Remote\_Attacker.

(10) **Consequence**: Consequence refers to the technical part of impact that could be caused by the Attack conducted by Attackers through exploiting certain Vulnerabilities. This class is comprised of several subclasses, each of which represents one type of consequences, including Denial\_of\_Service, Privilege\_Escalation, Data\_Modification Information\_Leakage, Arbitrary\_Code \_Execution. Take CVE-2008-5070 for example, the consequence that it might cause can be an instance of Arbitrary\_SQL\_Command\_Execution, which is a subclass of Arbitrary\_Code\_Execution.

(11) **Countermeasure**: Countermeasure is the concept to describe actions or approaches that can potentially prevent or mitigate the risk of this attack. These solutions and mitigations are targeted to improve the resistance of the target software and thus reduce the likelihood of the attacker’s success or to improve the resilience of the target software and thereby reduce the impact of the attack if it is successful. This concept is comprised of several sub concepts, each of which represents one type of countermeasures, including Patching and Advisory.



Figure 4. The rule of the similarVulnerability in SWRL