Establishing the Relationship in Vulnerability Classification for a Secure Software Testing

Nor Hafeizah Hassan[1] Siti Rahayu Selamat[2] Shahrin Sahib[3]

[1]nor_hafeizah@utem.edu.my
[2]sitirahayu@utem.edu.my
[3]shahrinsahib@utem.edu.my

Abstract

Having a significance vulnerability classification is important in developing a strong confidence in choosing the damaging cases associated with testing problems. The accurate classification helps to explain the belonging of vulnerability. The current research failed to empirically describe these matters, due to the absence of generic classification for testing and assessment. The aim of this paper is to fulfill this gap by enhancing the vulnerability classification meant for a secured software testing. This enhancement, which benefited from the issues of users view point and abstraction level, is later implemented in a vulnerability report database to determine the pattern of vulnerability relationship. As a result, the patterns, which support the traceability aspect, exposed the ability to be mapped with requirement elicitation through use case notation and served as a practical tool to demonstrate the impact and priority in performing the appropriate secure software testing.

Keywords: Vulnerability Classification; Secure Software Testing; Abstraction Level; Pattern; Requirement.

1. Introduction

A secured software testing is a test to reflect that the system is having the security attributes, such as confidentiality, integrity and availability (CIA) [1]. Testing in software comes within two perspectives: a) to find defects in software and b) to increase delivered reliability [2]. However, the difference between the two is in their inner operation focus. In the conventional testing, the focus is to observe the coverage of test but in performing the secured software testing, the focus is to ensure the software is abounding with security attributes [3]. The security attributes help to increase the delivered reliability of software.
Delivered reliability is the level of dependability towards the software, and is determined by the CIA attributes. The attributes rely on the fact that the software is tested for the possible vulnerability. Due to various purposes in classifying the vulnerability, as well as it vast population, the examination of vulnerabilities (and its characters) leads to the various study of vulnerability classification [4].

2. Related Work

Vulnerability classification evolved since 1974 with assorted purposes [9]. The existing methods of vulnerability classification consist of few perspectives. A popular example is by domain and genesis as well as by origin, as discussed in [5]. A recent work by Tripathi and Kumar discussed the feature selection necessary in doing the vulnerability classification [4]. The feature selection is the common feature to group the vulnerability under a category.

Secured software testing is a state-of-the-art, a work to confirm that software is occupied with security attributes. The existing security testing is a test to detect any defects that contribute to the flaws exhibit in an application and always considered as an afterthought concerns. On top of that, the purpose of secure software testing is to increase the reliability based on confidentiality, integrity and availability (CIA). Reliability is often estimated using quantitative study [2][6].

Our initial work is to state the differences between the conventional software testing and the security testing in [7]. We focused on the latest issue: the aim of both tests. In the former one, the aim was to find errors or to verify test result with requirements while, in the later, the aim was of how to eliminate the afterthought concern by incorporate the security attributes into the whole software development from the beginning stage in the software development life cycle (SDLC)[7].

However, in order to incorporate the security attributes, the stages of SDLC were analyzed through the studies of numbers of security testing frameworks, and it was proposed that the requirement and design stages are the earliest stage that could be used of. As both stages deal with requirement specification and elicitation, a mapping of the security attributes to the existing specification must be comprehensible. An example of mapping the security attributes is using the misuse case or abuse case within the UML notation[8]. However, these cases fail to mention the traceability aspect which was important to show the impact of a flaw and to prioritize the essential testing method. Without these two, testing the security attributes for a secure system become less significance.

In this research, our goal is to model a vulnerability classification for the purpose of evaluating the relationship of vulnerability classes with respect to their traceability from the software developer's point of view. The main advantage of this classification method is it can be used to assist during the elicitation of software security requirement in software development life cycle.

2.1. Classification of Vulnerability

Vulnerability classification is a process of identifying the common properties that belong to unique categories and place them under specific label of characterization [10]. Often, the characterization is selected based on: a) the objectives of the classification and b) the perspective of the user [1][2][12]. Each label shall specifies the purpose of why vulnerability is being in that particular class.

The meaningful labeling of this classification is considered as a security classification model when it can describe how and when security breaches occur; describe the impact on the system when they do, as well as the mechanisms, effects, and costs of system recovery, system maintenance, and defenses[13]. An analysis of eleven secure systems design methodologies made by [14] highlighted the needs for a standardized methodological approach that taking into account security attributes from the earliest stages of development till the completion.".

Nevertheless, the relationship mentioned in their research is claimed as impractical due to the unclear vulnerability classification[1]. Another issues raised in classification is the *level of abstraction* and *point of view*[15]. As an example, a single set of code can be considered as an input validation problem at one level of abstraction, or can be viewed as race condition vulnerability at another level. And at higher level of abstraction, this vulnerability can be classified as a design error. The viewpoint is also important as a designer will see it as a design error and a developer will see it as a development error. Thus, in classifying the vulnerabilities, these two issues must be taken care of as well as the purpose of classification[1].

Therefore, in this research, we proposed to classify the vulnerability into specific classes which are expressed by specific descriptions. The descriptions of the classes are adopted from the combination of the selected researcher works from [1]. We introduced the description of the classes as *"label of characterization"* as each class may represent any axes of perspectives or interests that characterized them which further let the researcher label them appropriately.

3. Identification of vulnerability classification reference model

As a beginning, we describe how we did the mapping of the classification features from existing class into generic class before using it for the purpose of secure software testing. We observed the purposes of classification from researchers that clearly stated their aims are to do the classification for the purpose of assisting in testing and maintenance as in [1]. It means that the purposes shall represent the level of abstraction. Level of abstraction implies the attempt how extensive the classification is. The target users are software developer or software designer. In existing research, the outcomes of classification consist of software development issues, location of flaws in the system and impact of flaws on the system. Each class comprises of second tier of subclasses.

The research in [1] proposes the combination of :i) cause, ii) location, iii) attack vector and iv) impact as the essential label of characterizations for testing, maintenance and assessment. Therefore, the reference model for vulnerability classification within software developer's perspective with the aim of assisting security testing, maintenance and assessment is based on this model (shown in Table 1) [4]. Any decision to specify the label of characterization is established based on the purpose of the classification and the target users.

The purpose responds to the issue of *level of abstraction* and the user's perspective counteracts the issues of *point of view*.

Table 1: The least label of characterization needed for security testing, maintenance and assessment.

Purposes	**User Perspective**	**Label of Characterization**
To assist security testing, maintenance and assessment	Software developer	Cause Location Attack vector Impact

However, the connections between the labels remain unclear. Based on the issues addressed, we proposed a new classification approach to determine the relationship between each class.

4. The Proposed Method for Vulnerability Classification in Secure Software Testing

The important aims within secure software testing are to confirm the presence of security attributes in the software and to increase the delivered reliability of the software to the user[2]. As the vulnerability population is vast, it is impossible to examine each of them individually[6] [4]. In their work, Memon and Xie, pointed out that any type of testing requires the tester to determine: i) what to test, ii) building test cases, iii) generating expected results and iv) verifying execution outputs. However, the security testing focuses on vulnerability exploit that demands the understanding of different states occurred as well as their relationship between each other when compromised. These states describe the behavior of an event that having certain attribute at any given condition. The existing studies describe how they perceive the existence (occurrence) of vulnerability in various ways such as domain, origin, operational and SDLC phases. Nevertheless, in software, the occurrence is about writing at least an executable statement of code. Then, regardless of how they are perceived, an occurrence of vulnerability consists of at least four states: i) creation, ii) discovered, exploited and iv) resolved of an executable statement of code[16]. These four states determine the characterization needed during the vulnerability classification model for a secure software testing framework. In any exploitation of vulnerability, it is essential for an attacker to identify the location of possible exploitable code using an attack mechanism and produce the targeted impact they have expected.

In [16], the states are given as lifecycle events of vulnerability and was represented as a waterfall sequence. However, the relationships between the events are not explained. For example, whether there exists any kind of source and target events, or if there exists bidirectional relationship between them. In order to identify the type of relationship that exists, the study of the vulnerability class must be able to map with the respective states. This relationship indicates the traceability aspect.

These states are further perceived as label of characterization in the Table 2 by adopting the labeling of cause – as vulnerability point (to refer to the vulnerable code statement), location - as application (to refer to the application container of where the code statement reside), attack vector – as attack (to refer to the attack mechanism use on the code) and impact – as target (to refer to the expected result of the exploitation). As the purpose is to determine the relationship between each label, the component relationship is added.

Table 2: The proposed label of characterization for vulnerability classification in secure software testing

Objective	**User Perspective**	**Labels and relationship of Characterization**
To assist security testing	Software developer	Label of characterization 1. Vulnerability point 2. Attack 3. Application 4. Target Relationship between label

5. Implementation: Using the Proposed Labeling Characterization

The purpose of this implementation is to investigate the efficiency of the vulnerability classification using the proposed label of characterization as in Table 2. The ultimate objective of this experiment was to increase the delivered reliability, therefore a quantitative approach is used as suggested by [2][6].

The example used the reported vulnerability database of Common Vulnerability Exposure (CVE). A CVE database consists of an entry and confirmed case. For this experiment, the confirmed case was extracted. The relationship of the class is identified through the text of the reported data. For example, in CVE-2000-0002, "Buffer overflow in ZBServer Pro 1.50 allows remote attackers to execute commands via a long GET request." Upon placing the keyword associated with each labeling, the relationship is shown by the available word left ("allows", "via", "execute").

Upon examined 50 random extracted data, the labeling and the associated word of the data is given as in Table 3 that summarized as object and relationship and presented in Fig. 1.

Table 3: The sample of object and relationship from proposed labeling of characterizations

Object	Relationship
Vulnerability point	allow
Attack	execute
Application	has
Target	caused
	via
	in

In Table 3, the object refers to each class identified in Table 2. The relationship is extracted from the reports to show the generic flow of vulnerability. This object and the relationship are adopted from the Tsipenyuk classification [11].

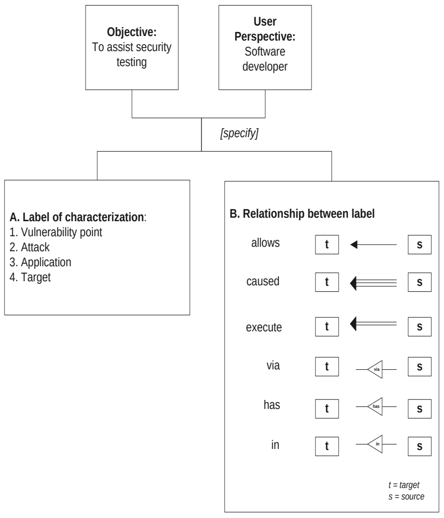

Fig. 1: Proposed objects and relationships

Fig.1 simplifies the object and relationship from Table 3. We identify that guided with the purpose and user's perspective, the label of characterizations determines the vulnerability classification and there exist a relationship between them. Upon analyzing the generic flow, three dominant flows were encountered as in Table 4.

Based on Table 4, the first pattern shows that the object relationship of "application-attack-target-vulnerability point" is the most common that occurred in the database, followed by relationship of "vulnerability point-application-attack-target-vulnerability point" and "vulnerability point-attack-target".

Table 4: The main generic flow from the proposed labeling

Generic Flow	%(out of 50 data)
1. application "allows" attack "execute" target "using	51
2. vulnpoint "in" apps "allows" attack "to execute" target	26
3. vulnpoint "allows" attack "to execute/perform" target	23

In order to show the relationship between the labels, some notations are introduced as in Fig. 1. Source and target are from the label of characterization. Source represents the origin labels and target represents effect of the intention that results into the target labels. From the CVE sample, a generic flow of vulnerability is identified as: "Application *allows* attack *execute* target *using /via* vulnerability point" or "Vulnerability point *allows* attack *to execute* target *through* application" and another five flows

6. Conclusion

The aim of this paper is to introduce the label of characterization and the relationship between them to be used in vulnerability classification for a sustainable secure software testing model. An implementation was conducted on a CVE database and the result showed that there was a significant generic flow of relationship from the outcome. This generic flow is beneficial to be expanded as a vulnerability flow graph.

The advantage of this generic flow graph is that it can be used as a testing aid for secure software testing model. The testing aid is gained by mapping the flow graph notation with the use case notation in requirement elicitation which disclosed the possible high risk vulnerability points. Even though there is a limitation of this study as it was run on a CVE database, it is strongly believed that the flow graph can be used on other vulnerability database that uses natural language as it report type basis.

Acknowledgement

This work was sponsored by the Minister of Higher Education of Malaysia (MOHE) and Universiti Teknikal Malaysia Melaka (UTeM).

References

[1] Tripathi, A. and U.K. Singh. "Taxonomic analysis of classification schemes in vulnerability databases". *6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT),* 2011.

[2] Piessens, F., "A Taxonomy (with Examples) of Causes of Software Vulnerabilities in Internet Software.", Department of Computer Science, K.U, Leuven: Belgium. p. 12, 2002.

[3] Memon, A.M., "A Comprehensive Framework for Testing Graphical User Interfaces, in Faculty of Arts and Sciences." University of Pittsburgh. p. 139, 2001,

[4] Tripathi, A. and U.K. Singh, "A Proposal for Common Vulnerability Classification Scheme Based on Analysis of Taxonomic Features in Vulnerability Databases." *International Journal of Computer Science and Information Security (IJCSIS), 9(6): p. 6, 2011.*

[5] Carl, E.L., R.B. Alan, P.M. John, and S.C. William, "A Taxonomy Of Computer Program Security Flaws." *ACM Comput. Surv., 26(3): p. 211-254. 1994.*

[6] Chen, Z., Y. Zhang, and Z. Chen, "A Categorization Framework for Commom Vulnerabilities and Exposures." *Computer Journal Advance Access, 53(5): p. 30, 2010.*

[7] Hassan, N.H., S.R. Selamat, S. Sahib, and B. Hussin, "Towards Incorporation of Software Security Testing Framework in Software Development", *Software Engineering and Computer Systems, Springer Berlin Heidelberg. p. 16-30,2011.*

[8] Alexander, I., "Misuse Cases : Use Cases with Hostile Intent." *Software, IEEE, 20(1): p. 58-66, 2003.*

[9] Gharibi, W. and A. Mirza, "Software Vulnerabilities, Banking Threats, Botnets and Malware Self-Protection Technologies." *IJCSI International Journal of Computer Science Issues, 8(1): p. 6, 2011.*

[10] Igure, V.M. and R. D.Williams, "Taxonomies of Attacks and Vulnerabilities in Computer Systems." *IEEE Communication Surveys & Tutorials, 10(1): p. 14, 2008.*

[11] Tsipenyuk, K. and B. Chess, "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors." *IEEE Security and Privacy, 3(6): p. 4,2005..*

[12] Saraydaryan, J., F. Benali, S. Ubeda, and V. Legrand, "Comprehensive Security Framework for Global Threats Analysis." I*JCSI International Journal of Computer Science Issues, 2: p. 14, 2009.*

[13] Nicol, D.M., W.H. Sanders, and K.S. Trivedi, "Model-Based Evaluation: From Dependability To Security." *IEEE Transactions on Dependable and Secure Computing, 1(1): p. 48-65,2004.*

[14] Villarroel, R., E. Fernández-Medina, and M. Piattini, "Secure Information Systems Development - A Survey And Comparison." *Computers & Security, 24(4): p. 308-321,2005.*

[15] Bishop, M. and D. Bailey, "A Critical Analysis of Vulnerability Taxonomies." *INFOSEC Computer Security,1996,*

[16] Al-Fedaghi, S. "System-based Approach to Software Vulnerability." *IEEE Second International Conference on Social Computing (SocialCom), 2010.*