A Component-based Architecture for Software Vulnerability Management

Chia-Hwa Liu
Institute of Applied Information Technology, Hsing Wu University

Abstract — **Recently, the increasing usages of software system in different web applications are threaten and attacked for security vulnerabilities. It impacts the existing information infrastructure seriously. Thus, how to identify, classify, remediate and mitigate the vulnerabilities of software had refereed as an important step to improve the software system's assurance. Basically the vulnerabilities are weaknesses in software that enable an attacker to compromise the integrity, availability, or confidentiality of that software or the data it processes. Thus to secure the software, it is necessary to collect all the related vulnerabilities in a system before identifying and removing them. Since component concepts in software development are expected to exhibit certain behaviors and characteristics that let them interact with its environment and other components. These attributes will fulfill the cyclical practice requirements of vulnerability management. Therefore, in this paper, the use of a component based strategy to create a comprehensive software vulnerability management system is presented. (CBASVMS) Which embed the vulnerability rule base for reasoning vulnerability attributes and the vulnerability knowledge base for possible settle methods are explained. Since the component representation can separate of concerns in respect of the wide-ranging functionality available for software applications, which have advantages of sharing with other applications and reusability for other new systems, that will characterize the knowledge of a domain for vulnerability settlement. According to the prototyping test, the CBASVMS are suitable for application in various types of software security service. The process of identifying and remediation of software vulnerabilities based on the costs and benefits associated with it will improve the security breach, meanwhile, it will reduce the impact or likelihood of security risk in the future.**

Index Terms — **software vulnerability, component-based architecture, secure software**

I. INTRODUCTION

Recently, the increasing numbers of wireless devices and network applications around world has led to a great demand for the usage of network software systems. However, there is a lot of software services in different web applications are threaten and attacked for security vulnerabilities as well. Statistics from Computer Emergency Response Team (CERT), Open Source Vulnerability Database (OSVDB) and Common Vulnerability Exposure (CVE) database confirm that the average number of reported vulnerabilities increase over time. (see Figure1). In other words, the occurrences and the evidences of the security vulnerabilities in

software service development will cause serious economic impact and hazards. It high-impact and high-likelihood challenged the existing information infrastructure. Thus, how to identify, classify, remediate and mitigate the vulnerabilities of software had refereed as an important step to improve the software system's assurance.

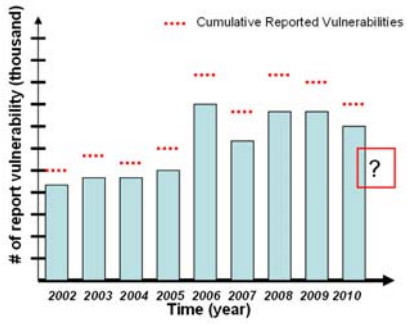

Figure1: The average number of reported vulnerabilities statistics

Basically the software vulnerabilities can be seen weaknesses in software that enable an attacker to compromise the integrity, availability, or confidentiality of that software or the data it processes. Thus to secure the software, it is necessary to collect all the related vulnerabilities in a system before identifying and removing them. Typical vendor treatments for users to identify and classify known vulnerabilities are using vulnerability scanners. These fragmental tools look for vulnerabilities known and reported by the security community, and which generally are already fixed by relevant vendors with patches and security updates, thus it is troublesomely and impractically. Since components are expected to exhibit certain behaviors and characteristics that let them participate in the component structure and interact with its environment and other components. When a component offers services to the rest of the system, it adopts a provided interface that specifies the services that other components can utilize, and how they can do so. Meanwhile, that a component can replace another at different time, if the successor component meets the requirements of the initial components. These attributes will fulfill the cyclical practice requirements of vulnerability management. Therefore, in this paper, the use of a component based strategy to create a comprehensive software vulnerability management system is presented. (CBASVMS) Which embed the vulnerability rule base for reasoning vulnerability attributes and the vulnerability knowledge base for possible settle methods are explained.[2,3] Since the component representation can separate of concerns in respect of the wide-ranging functionality available for software applications, which have advantages of sharing with other applications and reusability for other new systems, that will characterize the knowledge of a domain for vulnerability settlement and it have been used in a variety of areas as a support for creating more intelligent systems. According to the prototyping test, the CBASVMS are suitable for application in various types of software security service. The process of identifying and remediation of software vulnerabilities based on the costs and benefits associated with it will improve the security breach, meanwhile, it will reduce the impact or likelihood of security risk in the future.

This paper address some of the concerns raised for software vulnerability management architectures and proposes a component-based framework that supports; (i) generic interfaces to

support heterogeneity and portability, (ii) policy management to handle stakeholder and user policies, and (iii)a behavior abstraction to support extensibility and modularize the definition of individual optimization/control goals and their coordination.

The organization of the paper will divide into five sections. While in section one, the basic research background is introduced. In section 2, we survey and explain the related researches. Section 3 describe the proposed software vulnerability management system and explain the needed details of element, and section 4 will describe current practices of simulation results. Finally, in the section 5, the overall conclusion and future perspectives are given.

II. RELATED WORKS

Software vulnerabilities are caused by some kinds of programming errors and flaws that give rise to exploit techniques or particular attack patterns. Some related research about vulnerabilities management as [1,4] are presented, but not mention the component architecture for dynamic expansion. In paper [6], the optimal policy for software vulnerability disclosure is discussed. The vulnerability trend and classification statistics are presented in [8,9], but without mentioning system development issues. In paper [12], the researcher propose a quest for a framework to improve software security via vulnerability black markets scenario, which give a good suggestion for the policy adoption in the field of software vulnerability disclosure and vulnerability countermeasure in black markets.

III. BASIC COMPONENT-BASED ARCHITECTURE DESIGN

The component-based architecture is the basis design for distributed component embraces mechanisms and techniques for developing basic yet reusable business implementation units that is environment aware. It emphasizes the separation of concerns in respect of the wide-ranging functionality available throughout a given software system. It is a reuse-based approach to defining, implementing and composing loosely coupled independent components into systems.

Component-based systems encompass both commercial off-the-shelf (COTS) products and components acquired through other means.[5,6] The Major elements of a component will include (1) basic specification (2) One or more implementing methods (3) component model (4) packaging approach (5) deployment approach. And the component can communicate each other as shown in Figure2. Thus we can use the component architecture to construct a system, based on the component model which defining the rules of linking components together and suitable component interface with related platform environment, the system can operate to fulfill the management purpose. The architecture proposed in this work is based on component models for the blocks of the system shown in Figure 3

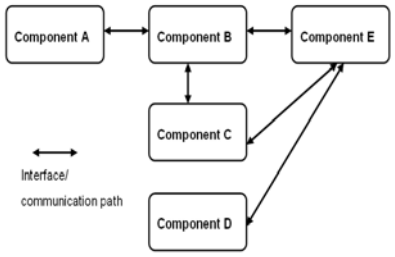

Figure 2: The software components may communicate each other to fulfill the management function

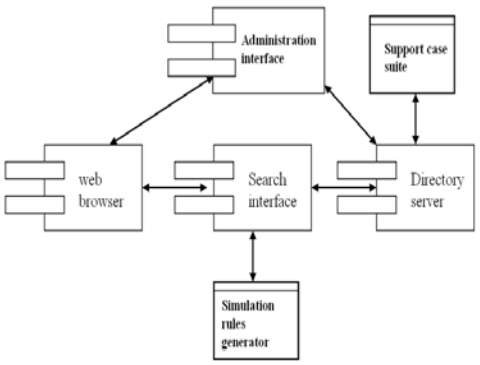

Figure3: The system component architecture

IV. SOFTWARE VULNERABILITY MANAGEMENT SYSTEM DESIGN

Vulnerabilities are weaknesses in software that enable an attacker to compromise the integrity, availability, or confidentiality of that software. Thus, the basic vulnerability management initiatives should be designed to indicate where the vulnerabilities will occur first, and then according to the degree of hazard to set the vulnerabilities up and follow to check the way of controls are not being applied adequately and finally to verify that vulnerabilities have been remediate.

The vulnerability management must start out by determining what the desired security state for their environment, which is the policy of the vulnerability management and associated process. According the policy to implement vulnerability management processes are shown in Figure4.

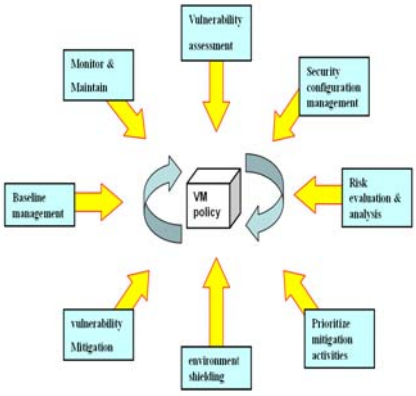

Figure4: According the policy to implement vulnerability management process

There are two types of vulnerabilities: known and unknown. The known vulnerabilities are the hazard events have already been found and reported. The best way to keep up with known vulnerabilities is to subscribe to regular security updates from comprehensive vulnerability databases. However the unknown vulnerabilities are vulnerabilities that have not yet been found. Especially new technologies and proprietary code extensions are frequently infested with unknown vulnerabilities. Unknown vulnerability management is not restricted to testing during development. The instance of managing security updates, verifying patches, system integration and network monitoring are all essential parts of unknown vulnerability management strategies. The Overall framework of CBASVMS with related the operational components and managing functions are shown in Figure5.

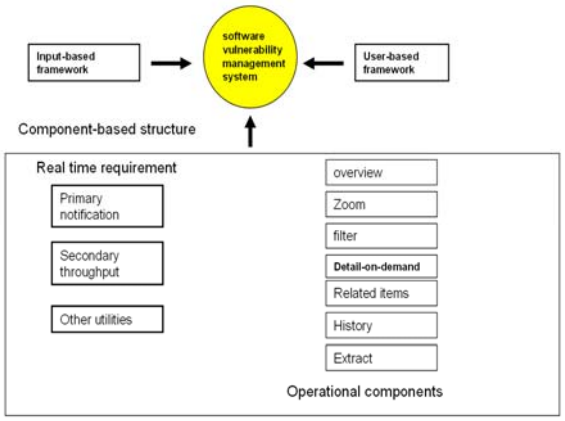

Figure 5: Overall framework of CBASVMS which **shown the operational components and managing functions**

Finally, a simulation example in OWASP top 10 vulnerability list is presented for demonstration and analyzing. Vulnerability naming Cross-Site Scripting (XSS) for example, it is recommending for the vulnerability rule base for reasoning vulnerability attributes and the vulnerability knowledge to identify flaw and possible attacking behaviors. The specification

of vulnerability in knowledge base is shown as below:

Vulnerability name : Cross-Site Scripting (XSS)	
Attacking Behavior	Defense Strategy
• to execute scripts in victim's browser • to hijack user sessions • to deface web sites • to redirect user to malicious sites	• Escape all untrusted data based on the HTML context • Positive or "whitelist" input validation • use NoScript browser,

Here the component based strategy is used to create separating method of the software vulnerability. The vulnerability rule for reasoning vulnerability attributes will use to retrieve the vulnerability knowledge from database. Thus, the countermeasure strategy can be used for possible settlement.

V. CONCLUSION

This paper uses a component based strategy to create a comprehensive software vulnerability management system. Which embed the vulnerability rule base for reasoning vulnerability attributes and the vulnerability knowledge base for possible settle methods are explained. Based on the risk management framework of CBASVMS are suitable for application in various types of software security service. The component process of identifying and remediation of software vulnerabilities based on the costs and benefits associated with it will improve the security breach, meanwhile, it is hope to reduce the impact or likelihood of security risk in the future.

REFERENCES:

[1] Anna-Maija Juuso and Ari Takanen *Unknown Vulnerability Management*, http://www.codenomicon.com/resources/whitepapers/codenomicon-wp-unknown-vulnerability-management-20101019.pdf

[2] Andy Luse, Keven P. S, Anthony M.T" A Component-Based Framework for Visualization of Intrusion Detection Events", *Information Security Journal* 17, 2008, 95-107

[3] Jurriaan S and Martin van M," A Component Based Architecture for Web Content Management: Runtime Deployable WebManager Component Bundles", proceedings of 8th international conference on web engineering, 2008

[4] John P. P," Key Elements of a Threat and Vulnerability Management Program", ISACA, online journal, http://*www.isaca.org*. 2006

[5] Diego Caberlon Santini, Walter Fetter Lages_ "A component based architecture for robot control", Revista Controle & Automação,Vol.22 no.4, 2011

[6] Arora, Ashish, Rahul Telang, and Hao Xu. "Optimal Policy for SoftwareVulnerability Disclosure". *Management Science* 54 (4):642-656. 2008.

[7] Browne, H. K., William. A. Arbaugh, John M, and William. L.F.."A Trend Analysis of exploitations", http ://www.cs.umd. edu/~waa/pubs/CS-TR-4200.pdf.

[8] " Symantec Global Internet Threat Report: Trend for July - Dec 07" http://eval.symantec.com/mktginfo/enterprise/white_papers/bwhitepaper-internet_security_threat_report_xiii_04-2008.en-us.pdf.

[9] "Symantec Report on Underground Economy."http://eval.symantec.com/mktginfo/enterprise/white_papers/bwhitepaper_underground_economy_report_11-2008-14525717.en-us.pdf.

[10]Arora, Ashish, Anand Nandkumar, and Rahul Telang.. "Does Information Security Attack Frequency Increase With Vulnerability Disclosure? An EmpiricalAnalysis.", *Information System Frontiers* 8 (5):350-362. 2006

[11] Arora, Ashish, Rahul Telang, and Hao Xu." Optimal Policy for SoftwareVulnerability ",Disclosure. *Management Science* 54 (4):642-656. 2008

[12] Jaziar Radianti, Jose J.G, Eliot Rich," A Quest for a Framework to Improve Software Security: Vulnerability Black Markets Scenario", proceedings of ISDC Albuquerque, USA, 2009