The Plan and Tools for Vulnerability Testing in Information Software-Based System

Injung Kim*, Younggyo Lee, and Dongho Won****

Abstract: Although many tests for stabilization of the software have been done, vulnerability test for a system run by combination of the software of various products has not been conducted enough. This has led to increased threats and vulnerability of system. Especially, web-based software system, which is public, has inherent possibility of exposure to attacks and is likely to be seriously damaged by an accident. Consequently, comprehensive and systematic test plans and techniques are required. Moreover, it is necessary to establish a procedure for managing and handling the results of vulnerability test. This paper proposes vulnerability test plans and designs for implementing automated tools, both of which can be complied with on web-based software systems.

Keywords: Risk Analysis, Vulnerability, Asset, Threat

1. Introduction

Software testing [1] is a very much hard task. Most of software development models repeat coding, testing and correction multiple times in order to stabilize a developed software, correct any bugs found during test and check any new bugs that might be detected. This is not true of tests of software-based systems. Once a software system is built, its testing is under restrictions. Even if the software system is found to have vulnerability, it is impossible to correct it. This is because once a software system is built and operated, it cannot be corrected or stopped arbitrarily. Therefore, cost effects coming from the revision and supplementation of vulnerability should be analyzed since unlike a bug, the vulnerability does not mean that the system has functional problems.

Especially, recent developments in internet have led many organizations to establish requirements for a web-based system before starting to operate it [2]. A web-based software system can be developed easily and shorten development period since it has various and similar solutions. However, it has been found that the system has much vulnerability and has been exposed to increasingly new kinds of vulnerability since it uses TCP/IP protocol and runs on Windows or Linux platform [3]. Conventional system vulnerability testing [4][5] has been conducted in such a way that any vulnerability found by using scan tools or cracking tools is reported, followed by recom-mendations like installation of security patch or OS upgrade. This testing method may repeat same test since it does not have any proper plan and any proper knowledge of previous test cases and testing methods. Re-repeated testing for this reason increases wasted time and cost, making response to new vulnerability.

There have been standardized plans and procedures for software testing [15], and there have been many studies on them. However, any testing method for software-based systems is hardly found, except for risk analysis approach [10][11] and risk management approach [12][13][14]. These methodologies require specialized knowledge of security and lots of human resources and time. For this reason, they cannot be applied well to the situation where vulnerability should be eliminated in real time. Especially, the methodologies require a long time from risk analysis and budget design to establishment of countermeasure, disabling just-in-time establishment of countermeasure [9]. This makes it difficult to establish safe system operation. It is better to apply security risk analysis methodology when changing whole system or considering a new project. Once a system is built, vulnerability should be eliminated by vulnerability testing, as occasion calls.

Next, a tool [6] is developed and used for software testing. Since related developers and testers conduct similar jobs, proper information on the stress and load against software is set in the testing tool. Especially, a testing tool provides knowledge of predictable result of a bug. However, system vulnerability testing inspects various kinds of vulnerability of a system composed of a combination of software, in terms of rule settings, batch, composition and interlock. Vulnerability removal affects other software. Therefore, personnel, roles, methods and levels for system test planning must be clearly defined and specified, and inter-system vulnerability diffusion analysis [7][9] must be done well. To solve these problems, this paper proposes a method for software system vulnerability test planning and a design tool for its automation.

Manuscript received September 30, 2005; accepted November 8, 2005.
* Electronics and Telecommunication Research Institute, Daejeon, Korea (ciper@etri.re.kr)
** Information Security Group, Sungkyunkwan University, Suwon, Korea ({yglee, dhwon}@dosan.skku.ac.kr)

2. Vulnerability testing plan

The first thing to do in establishing a vulnerability testing plan is to understand system environment for vulnerability testing and define testing scope. To this end, the procedure illustrated in Fig. 1 will be followed.

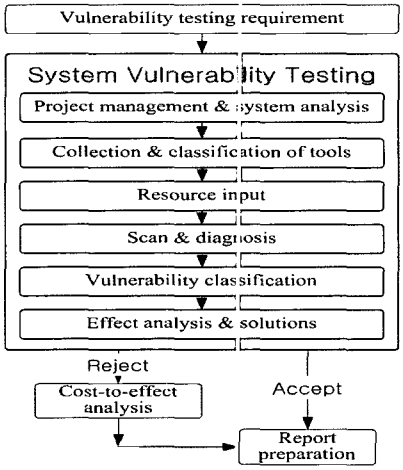

Fig. 1. Vulnerability testing flow diagram

2.1 Vulnerability testing requirement

When an organization builds a software system, it must conduct vulnerability testing to make the system secure. The testing must be told from conventional testing for improvement of system failures. Vulnerability testing requires knowledge on the specs required of software system, data of the system's testing results and development environment of the system. Generally, vulnerability testing cannot be done by in-house personnel. It is outsourced mostly to related consultants. It is necessary, therefore, to know how to sign a testing contract, budget required, information of budget designers, and the consultants' names, titles, addresses, telephone numbers, email addresses and roles.

2.2 Project management and system analysis

A team in charge of vulnerability testing must not think that a department demanding the testing understands everything about system. Explanations on testing level/scope and training must be given to the line manager of the department demanded the testing. Most of testing-demanded departments believe that vulnerability testing will get rid of vulnerability and ensure safe system operation. However, the testing is not free from occurrence of attacks. Based on project management, therefore, it is necessary to record the human resources that will be employed in vulnerability testing, their roles, testing timetable and testing conducted, and present vulnerability testing results within a specified period. The most important thing in project management is schedule management. Vulnerability testers should be managed well so that vulnerability testing timetable covering diagnosis target selection, setting check, rule check, scanning, mock attack and report preparation can proceed sequentially.

2.3 Collection and classification of testing tools

In a software system, different development languages, operating systems and hardware-based programs give and take information through protocols. To find serious vulnerability from these complicated relations, it is important to collect and classify related testing tools. Especially, level of testing results depends on the accuracy of testing tools. But, skilled testers capable of using the tools are necessary.

2.4 Resource input

Hardware for testing should be listed up and resource input items of the operations to be conducted should be decided. Since a testing team has different levels in its members' experiences, job assignment suitable for each member's experience should be specified. The most important thing is that testers are aware of the importance of security. Therefore, testers must comply with security agreement and obligations related with testing. This is because system vulnerability known to the public provides a cause of attack and makes the system exposed to serious dangers. The resource items to be input are as follows.

1. Testers : number of testers, and their experiences and specialties
2. Equipment: computer, scanner and analyzer
3. Office and room for study
4. Software: word processor and database
5. Other supplies: telephones and books

Testers should define their roles to avoid double check. Testers' roles are classified into managing, physical and technical fields. Since most of them are in technical field, it is desirable to divide them into sub-fields: server, network, data, application and PC.

2.5 Scan and diagnosis

The best thing is to do vulnerability testing by building a test bed. This requires high cost and disables modeling, and it is applied to a system that is actually run. Therefore, it is

Table 1. Classification of informal testing items and formal testing items

Items	Informal	Formal
System deployment & operation	O	
Whether each asset is upgraded & patched	O	
Privilege per asset & user access control	O	
Rule setting & control items per asset	O	
Server OS		O
DB		O
Network OS/protocol		O
Contacts of interlocked systems		O

absolutely necessary to know precise scan times and locations. Since scan cannot be done on all locations, honeynet [7] or log system should be installed in the system. For software testing, black box test or white box test is ordinarily done. For vulnerability testing, all systems cannot be scanned against vulnerability since software vulnerability and overall diagnosis of whole system should be done. Consequently, it is important to tell informal testing items from formal testing items. This classification is as shown in Table 1.

2.6 Vulnerability classification

Vulnerabilities always change and increase. Periodic vulnerability scan is required. This is one of slow-paced and difficult tasks. Therefore, diagnosis results should be classified and stored. Also, the information on testers input, tools and methods employed, and diagnosis dates should be stored in order to make it clear who should be responsible for testing results later. In subsequent re-diagnosis, vulnerability testing should be applied to new updated list of vulnerabilities. Any new vulnerabilities' found should be listed up and classified on the basis of their possible effects. Then, their seriousness should be recorded.

2.7 Effect analysis and solutions

Any predictable effects of vulnerabilities found should be analyzed and solutions for these problems should be decided. This is the most necessary thing of what testers must do. A software tester completes his/her mission only by writing a bug report and forwarding it on a coder. A vulnerability tester finding a vulnerability should predict the possible effects and damages that may come from it, and provide prediction results, because he/she knows best about seriousness of the concerned vulnerability. Vulnerabilities affect various software systems as well as an asset. It is most effective to provide a solution for reducing overall vulnerabilities. Therefore, it is best to establish a countermeasure for comprehensive decrease in vulnerabilities of testing items.

2.8 Cost-to-effect analysis

Even if a vulnerability is found, it may not be able to remove or decrease it. When elimination of the vulnerability requires high cost or provides little effectiveness, it should be ignored. Highly cost-effective methods for vulnerability elimination should be implemented first, and other vulnerabilities requiring high cost should be left to be removed later during system upgrade.

2.9 Report preparation

A report on detected vulnerabilities should be prepared in such a way that each of vulnerabilities, analysis of its effects and its solution are matched. Showing facts about vulnerabilities and results of attacks against vulnerabilities increases level of report.

3. Case and procedure of vulnerability testing

Test case and procedure should be prepared after vulnerability testing is planned. Control items as shown in BS7799[14] or command sentences written in exploit code should be used. Vulnerability test procedure should be established by recording objectives and requirements for detection of each vulnerability and making new testers know the precise target and method of testing. The case and procedure of vulnerability testing is very effective for most of web-based software systems since these systems have a similar scope of vulnerability. Based on the example and procedure, effect analysis and solution should be provided. Results should be stored in database for use in subsequent analysis of system vulnerability so that job volume needed by analysis may be reduced.

4. Vulnerability testing tools

It is very difficult to prepare vulnerability test plan and procedure by using one's memory or paper/document. For this reason, this study has developed a tool that can automate the proposed vulnerability test plan and procedure. This tool is divided into two parts: one for project management and other for informal diagnosis and formal diagnosis. For formal diagnosis, the tool is designed to classify vulnerabilities on the basis of diagnosis item, put solutions for related vulnerabilities into table, and place solutions immediately for any vulnerabilities found. Solutions are categorized into period and cost, ensuring that best countermeasures can be established at optimal cost.

Using these functions, this study designed a tool as shown in Fig. 2. The items for implementing the tool are as follows in Fig 3.

- Server: Windows 2000 server : MS-SQL database
- Apache web server, .Net based software system
- Client: Windows XP/2000 professional : Web browser

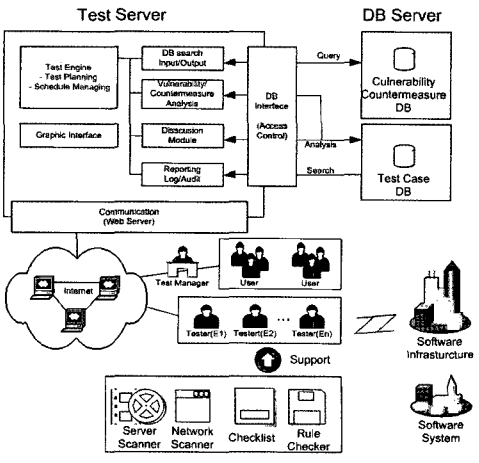

Fig. 2. K-check configuration

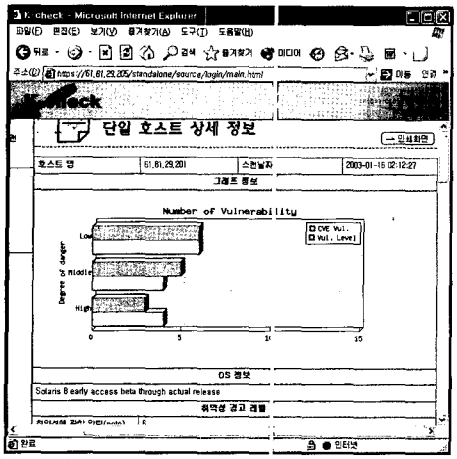

Fig. 3. Illustrating a screen of K-Check shows the result of software system vulnerability check

5. Conclusion

Software testing is a very important task that should be done periodically and continuously. But for this testing, a software system may be attacked easily. This causes replacement cost and gives damages to users. This paper describes software vulnerability testing plan and procedure, and method for design using a tool. Vulnerability test using K-check showed higher efficiency than when any tool is not used. The testing could provide immediately testing result. So the result could be used as objective data in establishing urgent countermeasures. Future study will subdivide software test plans and procedures, and propose testing plans based on vulnerability and threat and effective algorithms for effect analysis.

References

[1] *ANSI/IEEE Standard 829/1983 for Software Test Documentation*. 1988.

[2] *IEEE Standard 830 for Recommended Practice for Software Requirements Specifications*, 1998.

[3] Cert Coordination Center, http://www.cert.org.

[4] Shu Xiao; Lijun Deng; Sheng Li; Xiangrong Wang, "Integrated TCP/IP protocol software testing for vulnerability detection, Computer Networks and Mobile Computing," *ICCNMC 2003*. pp.311-319, Oct. 2003

[5] Thompson, H.H. "Why security testing is hard," *Security & Privacy Magazine*, IEEE , Volume: 1 , Issue: 4 , Pages:83 - 86, July-Aug. 2003

[6] Satoh, I., "Software testing for mobile and ubiquitous computing," *Autonomous Decentralized Systems, 2003. ISADS 2003. The Sixth International Symposium on* , Pages:185 - 192, 9-11 April 2003.

[7] Injung Kim, el. "The Design and Implementation for the Practical Risk Analysis Tools," *IFIP2004 Summer Conference*, Aug. 2003.

[8] Injung kim, el, "Security Honey-Net in Risk Analysis," Oct. *PosterSession COMPSEC2003.*

[9] Injung Kim, el, "A Study on Security Risk Modeling over Information and Communication", *SAM2004.*

[10] CSE, *Threat and Risk Assessment Working Guide*, Government of Canada, Communications Security Establishment, 1999.

[11] GAO, *Information Security Risk Assessment - Practices of Leading Organizations, Exposure Draft*, U.S. General Accounting Office, August 1999.

[12] ISO/IEC JTC 1/SC27, *Information technology - Security technique - Guidelines for the management of IT security (GMITS) - Part 3*: Techniques for the management of IT security, ISO/IEC JTC1/SC27 N1845, 1997. 12. 1.

[13] Solm, R., "Information Security Management(2): Guidelines to The Management of Information Technology Security (GMITS)", *Information Management & Computer Security*, Vol. 6, No. 5, 1998, pp.221-223.

[14] BSI, *BS7799 - Code of Practice for Information Security Management, British Standards Institute*, 1999.

[15] A. Fredlein, *Web Project management*, 2000.

Injung Kim
He received his M.E. degree in Electronic Engineering from Chungnam National University, Korea, in 1992. He joined ADD (Korea Agency for Defense Development) in 1992 and remained until 2000. In addition, since 2000, he has working in NSRI (Korea National Security Research Institute). He is currently a Ph.D. candidate of Computer Engineering in Sungkyunkwan University. His current research interest is in the area of information security, network security, risk analysis.

Younggyo Lee
He received the B.E. degree in Electronic Engineering from Hanyang University, Korea, in 1986 and the M.S. degree in Electronic Engineering from Hanyang University, Korea, in 1991. He is currently a Ph.D. candidate of Computer Engineering in Sungkyunkwan University. His current research interest is in the area of cryptography, particularly regarding the design of secure PKI protocols.

Dongho Won
He received his B.E., M.E., and Ph.D. degrees from Sungkyunkwan University in 1976, 1978, and 1988, respectively. After working at ETRI (Electronics & Telecommunications Research Institute) from 1978 to 1980, he joined Sung kyunkwan University in 1982, where he is currently Professor of School of Information and Communication Engineering. His interests are on cryptology and information security. Especially, in the year 2002, he was occupied the president of KIISC (Korea Institute of Information Security & Cryptology).