Securely Web-Based Application for Construction Material Testing

Nontarak, S.
Technology Information System Management
Mahidol University
Thailand

Leelawat, T.
Department of Civil and Environmental Engineering
Mahidol University
Thailand

ABSTRACT
The main aim of this paper was to develop and evaluate securely web-based application for construction material testing using object-oriented technology and parameterized queries for SQL command queries. The SQL queries for the web application of construction material testing were modified by adjusting their codes which included connection strings, authorization bypass and execute commands. Detection of SQL injection vulnerability was conducted by expertise and two automatic web vulnerability scanning tools. It was found that the parameterized queries could minimize the SQL injection flaws of the web application significantly.

General Terms
Data and Information Systems

Keywords
Construction Material Testing, Parameterized Query, Web Scanning Tool, Authorization Bypass.

1. INTRODUCTION
Nowadays, the web-based information system (IS) is mostly recognized to be a valuable tool used in many organizations. It could be applied suitably for improving the effectiveness and efficiency for quality control processes, especially for construction material testing [1]. The outcomes obtained from the use of IS are not only to minimize the overall cost and shorten the schedules of the whole processes, but the organizations which IS have been brought could also gain benefit to seek who are potential groups of customers, suppliers and possibly competitors in their business. However, many web-based IS applications developed for construction material testing are hindered by users and customers. Since the commercial databases that are currently available may not be designed suitably for their business and could not be complied with their standards. In addition, most of the business transactions and secured information have also been designed to proceed mainly through their network. Injection vulnerability of the secured information which involves in the process of transferring data between the relevant parties could be possible.

Attackers commonly inject via web application which is a popular way to attack. Several techniques attacking the web application; such as a cross-site scripting (XSS), SQL injection, parameter tampering, hidden manipulation, plain text attack or cookie poisoning, were reported [2]. However, the SQL injection was found to be the simplest method for the attack and the trends for the attack were reported to increase rapidly [3].

Spett [4] mentioned that the most common form for the attacks of web-based application was relevant to the process of bypassing logon form which was also called as "Authorization Bypass". Other common forms for the attacks could be done by adding information to the strings in order to generate a series of new commands or injecting through the stored procedures [4].

Several methods [5-12] have been proposed to prevent the SQL injection attack to the web-based application. These include adding some parameterized strings to the SQL commands, using a virtual database connectivity drive or extracting the basic structure of a SQL statement, creating the unpredictable instances of the language, and combining the static application code analysis with runtime validation. The automatic web vulnerability scanners were also proposed to be used effectively for evaluation of web-based application by detecting software fault injection techniques [13-15].

Accordingly, the paper was aimed at developing and evaluating securely web-based application for construction material testing using object-oriented technology and parameterized queries for SQL command queries. The application being developed assisted in transferring the secured data between the parties involved, storing the data in the databases which were designed to comply with the recommended format for concrete in materials property database.

2. SYSTEM ANALYSIS AND IMPLEMENTATION

2.1 System Analysis
Data was gathered by interviewing, tracking all forms of construction and material testing, and inspecting the processes of existing construction and material testing at the Department of Civil and Environmental Engineering, Mahidol University. Problem statements and user requirements defined were then used appropriately for developing the data analysis of the application program.

Details of overall data flow diagram used in the study could be separated into four parts; including the processes of data management, material testing management, verification management, and report management. The data management process covers specified levels of authorized groups, details of customers, and samples of materials to be submitted for testing (see Figure 1). The customers or secretary could select the test from the material testing lists and then follow by selecting the company.

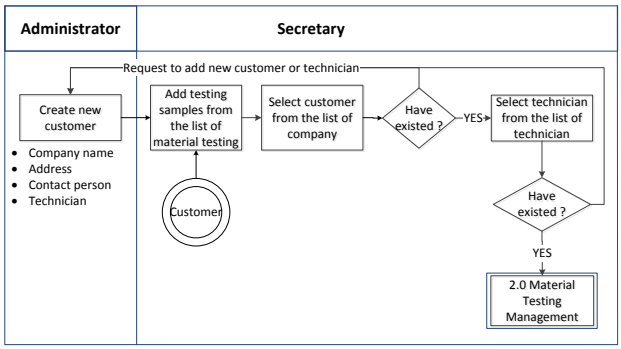

Figure 1. Data management

The material testing management process is used for checking the customers' orders obtained from the data management process or the rechecked orders specified by the cerified person. The technicians then select the project, fill the testing results and finally submit the processing results to the verification management process (see Figure 2).

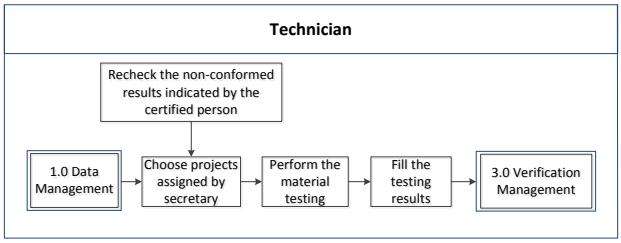

Figure 2. Material testing management

The verification management process involves with verifying the testing results obtained from the technicians (see Figure 3). This is to ensure that the testing results are correct prior to submitting the proper reports to the corresponding customers. However, if the testing results are found to be wrong, the cerified person will send the command to the technicians for rechecking the non-conformed results.

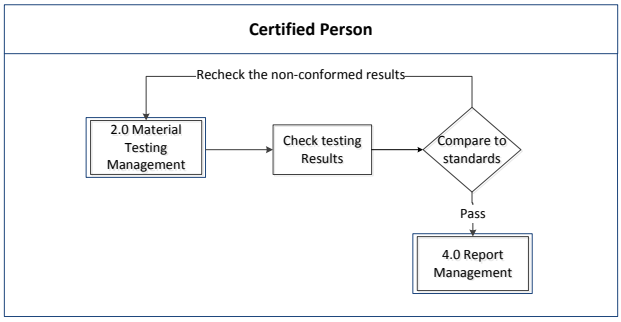

Figure 3. Verification management

The report management process involves with submitting the properly formatted reports approved by the cerified person to the corresponding customer (see Figure 4). The secretary then needs to notify to the corresponding customer via e-mail or telephone as requested.

The capability of accessing information was divided into five authorized groups; including levels of administrator, secretary, certified persons, technicians, and customers (see Figure 5). Each user could access through the specified levels of the program application via entering his/her login and password. The administrator level could access all the functions provided in the web-based application. The secretary level covers the processes of insertion, update or deletion of general information and also includes the process relating to specifying technician for material testing inquired from customers in each project. The certified level relates to the verification of the testing results of materials conducted by the technicians. Certified persons could also view all raw, processing and previous certified data. The technician level covers the process of material testing management. The testing process is provided for technicians to check the orders obtained from customers and to fill the test results which are performed by the instruments. However, the technicians could not be allowed to delete any processing data. Their personal information could, as usual, be adjusted. The customer level could send the inquiry for testing of materials, check his/her status of material testing, and review his/her testing reports.

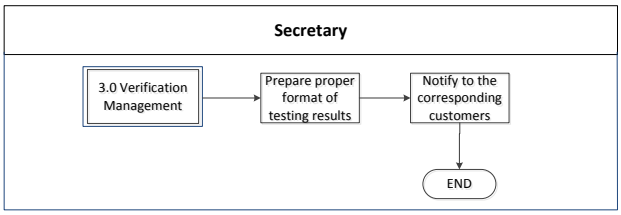

Figure 4. Report management

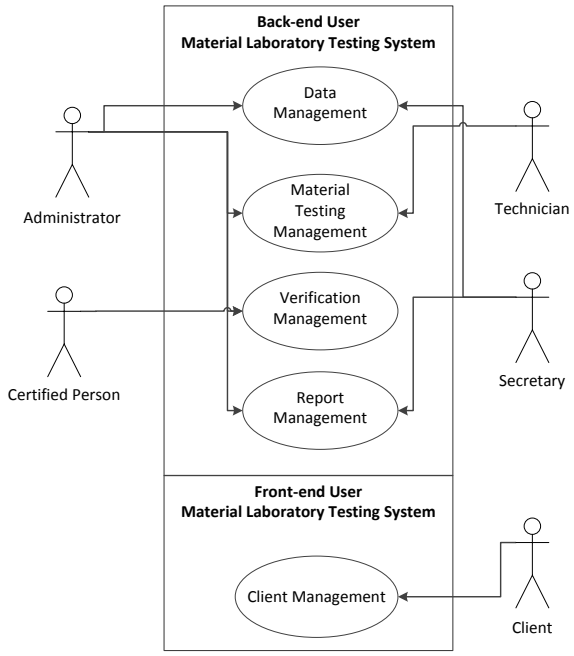

Figure 5. Capability of assessing information for each authorized group

2.2 System Implementation

Two web-based applications were developed in this study. The first web-based application was simply designed following all of the requirements specifying in the development tools (see Figure 6). Whereas the second web-based application additionally adjusted the SQL queries from the first web-based application by modifying their codes which included connection strings, authorization bypass and execute commands

(see Figure 6). Typical connection strings which were converted into object-oriented SQL coding are given in Table 1.

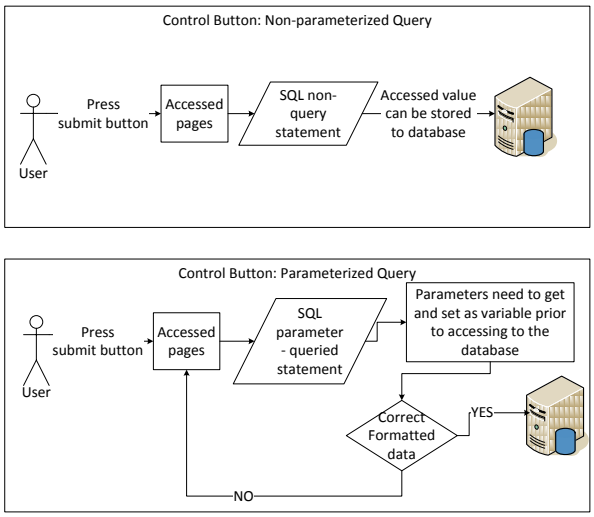

Figure 6. Non-parameterized and parameterized web-based application

Table 1. SQL Coding for connection string used

Web-based Application	Connection Strings
1st web-based application (none Object-Oriented)	strConnString = "SERVER= localhost; USER ID= root; PASSWORD= root; DATABASE=DBQC; POOLING=FALSE";
2nd web-based application (Object-Oriented)	#region string ConfigDatabase private const string SERVERNAME = " localhost "; private const string DATABASENAME = " dbqc "; private const string USERNAME = "ROOT"; private const string PASSWORD = "ROOT"; public const string ConnString = "Data Source=" + SERVERNAME + ";" + "Initial Catalog=" + DATABASENAME + ";" +"Persist Security Info=True;" + "User ID=" + USERNAME + ";" +"Password=" + PASSWORD + ";"; #endregion

The authorization bypass was also adjusted to minimize the SQL injection attack via login page. The password was encrypted by using getMD5 which is commonly used for encrypting password for register or reset password. The confirmation of user authentication for every transaction was required. Typical examples of SQL coding for the web-based applications studied are given in Table 2.

In addition, the SQL codings of the execute commands for the second web-based application were adjusted for objected-oriented and parameterized queries. This is to minimize the transmission values through variables directly. The examples of execute commands adjusted are given in Table 3.

Table 2. SQL Coding for user authentication

Web-based Application	SQL Coding
1st web-based application (none-parameterized queries)	sqlUserName = "SELECT eusername,epsswd FROM employee " sqlUserName += " WHERE eusername ='" + username + "'" sqlUserName += " AND epsswd ='" + pwd + "'";

2nd web-based application (parameterized queries)	sqlUserName = @"SELECT eusername,epsswd FROM employee WHERE eusername =@Username
	AND epsswd =@Password";
	cmd.Parameters.AddWithValue("@Username",Username);
	cmd.Parameters.AddWithValue("@Password", vMD5(pwd));

Table 3. SQL coding for select command

Web-based Application	SQL Coding for select command
1st web-based application (none-parameterized queries)	strSQLcontract = "SELECT COUNT(*) FROM samplemng WHERE sample_id = '" + Me.TxtContract.Value + "'"
2nd web-based application (parameterized queries)	string strSQLcontract = @"SELECT COUNT(*) FROM samplemng WHERE sample_id = @CtrValue"; cmd.Parameters.AddWithValue("@CtrValue",CtrValue);

3. TESTS FOR WEB APPLICATION

The web security of two web-based applications was tested by both manual test, which was performed by expert, and two web scanning tools [16, 17].

The manual test was conducted to examine the process of bypassing login form whether the messages resulting from SQL command error occur. Typical examples of commands; such as adding 'OR "=' on both username and password in login form, or inserting single quote (') into the password and adding "admin" into the username in login form (see Figure 7), were checked. Both web applications were also conducted to examine the errors occurring on registry page due to improper codes on execute commands.

Whereas the web scanning tools were used for detecting the vulnerabilities of the application program in five main areas; including injection flaw, information leakage and improper error handing, insecure cryptographic storage, broken authentication and session management, and insecure communication.

Figure 7. Login form

Table 4 shows the results of SQL injection attacks for both web-based applications evaluated by expert. After the login form of the first web-based application was injected as mentioned previously, the second page of the first web application is shown in Figure 8. However, the second web-based application being injected appeared only "Invalid User" on the login page. It can be seen that the modifying object-oriented technology and parameterized queries could manually prevent SQL injection penetrating through both authorization bypass and execute commands of the web application being tested.

Table 4. Manual Evaluation of SQL injection attacks on the web-based applications

Investigated Parameters	Injection Capability	
	1st web-based application (none-parameterized queries)	2nd web-based application (parameterized queries)
Process of bypassing login	Yes	No
Errors occurring on registry page	Show error messages	No error message

Figure 8. Directly access to the main page

Whereas the number of alerts detected from the attacks using the web scanning tools is given in Table 5. The alert results obtained from two automatic web scanning tools also indicated that injection flaw, information leakage and improper error handing, and insecure cryptographic storage were substantially reduced. The 1st web scanning tool detected that SQL injection vulnerability for the parameterized web-based application developed in the study was minimized by 62%. Whereas the SQL injection vulnerability significantly reduced up to 84% was reported by the 2nd web scanning tool.

Table 5. Evaluation of SQL injection attacks on the web-based applications using web scanning tools

Categories	Number of Alerts Detected	
	1st Web Scanning Tool (2nd Web Scanning Tool)	
	1st web-based application (none-parameterized queries)	2nd web-based application (parameterized queries)
Injection Flaws	22 (1)	0 (0)
Information leakage and improper error handling	0 (6)	0 (0)
Broken Authentication and Session Management	5 (4)	4 (1)
Insecure cryptographic storage	0 (4)	0 (1)
Insecure Communications	5 (4)	4 (1)

The alerts of the injection flaws for the 2nd web-based application were found to significantly decrease for both web scanning tools, compared to those for the 1st web-based application. Since all queries were transformed to parameterized queries in order to get the variable value prior to accessing the database and also filter out hazardous characters from user input.

The information leakage and improper error handling was detected only by the 2nd web scanning tool. This is due to parameter setting of debug error message for .NET framework prior to modifying queries into parameterized queries of the 2nd web-based application. The test was conducted by disabling debug error message, removing default scripts, and deleting old version of file obtained from the virtual directory. It was found that no alert of the information leakage and improper error handling was observed.

However, the number of alert results for broken authentication and session management, insecure cryptographic storage, and insecure communications for the 2nd web-based applications were still reported. Since these three categories are relevant to communication and network security of the web-based application. Even adjusting the codes of the queries for the web-based application was still found to unchange the number of the alerts. Since the alerts detected by the web scanning tools could be ignored in the case that the connection usually SSL is encrypted.

4. CONCLUSIONS

The web-based application of construction material testing could be securely developed by modifying object-oriented

technology and parameterized queries. The databases of the web-based application were designed to support appropriate information for all of the parties involved and to encrypt password for registered authentication users of every transaction prior to being verified and accessed to the system management. Whereas parameterized queries were modified by removing malicious characters that could possibly modify the actual SQL query structures and changing contexts relating to the execute commands in order to prevent the queries from manipulating the server to return records other than those intended.

Evaluation of the SQL injection vulnerability conducted by the expert indicated that modifying object-oriented technology and parameterized queries could prevent SQL injection penetrating through both authorization bypass and execute commands of the web application being tested. Whereas two web scanning tools detected that the SQL injection vulnerability was significantly reduced to 62 and 84%, compared to the web-based application using non-parameterized queries.

5. REFERENCES

[1] Arıoz, O., et al. 2007. Web-based quality control of ready mixed concrete. Building and Environment. 42, 1465-1470.

[2] Kost, S. 2007. An Introduction to SQL Injection Attacks for Oracle Developer. White Paper.Integrigy Corporation.

[3] Strom, D. 2006. An Anatomy of a Web Hack: SQL Injection explained. White Paper.Breach Security Inc.

[4] Spett, K. 2005. SQL Injection: Are your web applications vulnerable? Technical Report.SPI Dynamics Inc.

[5] Amirtahmasebi, K., Jalalinia, S.R., and Khadem, S. 2009. A survey of SQL injection defense mechanisms. in International Conference for Internet Technology and Secured Transactions, ICITST. London. IEEE.

[6] Anley., C. 2002. More Advanced SQL Injection. White Paper.Next Generation Security Software Ltd.

[7] Bandhakavi, S., et al. 2007. CANDID: Preventing SQL Injection Attacks using Dynamic Candidate Evaluations. in Proceedings of the 14th ACM conference on Computer and communications security. New York. ACM.

[8] Boyd, S.W. and Keromytis, A.D. 2004. SQLrand: Preventing SQL Injection Attacks. in Proceedings of the 2nd Applied Cryptography and Network Security (ACNS) Conference. Yellow Mountain.

[9] Nystrom, M.G. 2007. SQL Injection Defenses. Short Cuts.O'Reilly Media.

[10] Obimbo, C. and Ferriman, B. 2011. Vulnerabilities of LDAP as an Authentication Service. Journal of Information Security. 2, 151-157.

[11] Sam, M.S.N. 2005. SQL Injection Protection by Variable Normalization of SQL Statement.; Available from: http://www.securitydocs.com/library/3388.

[12] Wei, K., Muthuprasanna, M., and Suraj, K. 2006. Preventing SQL injection attacks in stored procedures. in Software Engineering Conference. IEEE.

[13] Fonseca, J., Vieira, M., and Madeira, H. 2007. Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks, in PRDC. 13th Pacific Rim International Symposium on Dependable Computing., IEEE: Melbourne. 365 - 372.

[14] Fu, X. and Qian, K. 2008. SAFELI: SQL injection scanner using symbolic execution. in Proceedings of the 2008 workshop on Testing, Analysis, and Verification of web services and applications. New York. ACM.

[15] Khoury, N., et al. 2011. An Analysis of Black-Box Web Application Security Scanners against Stored SQL Injection. in IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and IEEE Third International Confernece on Social Computing (SocialCom) Boston. IEEE.

[16] Shinder, D. 2005. Acunetix Web Vulnerability Scanner. Product Review 2005; Available from: www.windowsecurity.com/articles/product-review-acunetix-wvs.html.

[17] Anon 2008. The Power of AppScan: A Hands-On Review of IBM Rational AppScan Standard Edition. EMA™ IMPACT BRIEF 2008; Available from: www.ibm.com/software/awdtools/appscan/.