RESEARCH ARTICLE | MAY 08 2017

Research on the technology of detecting the SQL injection attack and non-intrusive prevention in WEB system FREE

Haibin Hu

AIP Conf. Proc. 1839, 020205 (2017)
https://doi.org/10.1063/1.4982570

Research on the Technology of Detecting the SQL Injection Attack and Non-Intrusive Prevention in WEB System

Haibin Hu[a)]

Education and Information Technology Center, China West Normal University, Nanchong Sichuan 637002, China

[a)]haibinhu@126.com

Abstract. Among numerous WEB security issues, SQL injection is the most notable and dangerous. In this study, characteristics and procedures of SQL injection are analyzed, and the method for detecting the SQL injection attack is illustrated. The defense resistance and remedy model of SQL injection attack is established from the perspective of non-intrusive SQL injection attack and defense. Moreover, the ability of resisting the SQL injection attack of the server has been comprehensively improved through the security strategies on operation system, IIS and database, etc.. Corresponding codes are realized. The method is well applied in the actual projects.

Key words: SQL injection attack; WEB application; non-intrusive; defense technology

INTRODUCTION

With the rapid development of Internet technology and the birth of a series of new Internet products such as Web 2.0, SNS and Weibo, the Internet application based on Web environment is being widely used. Many corporations use the Web platform in the informatization process. At the same time, many hackers are intensely attracted to the rapid development of Web business, resulting in the Web security issues. Among many Web security issues, the most notable and dangerous one is SQL injection. Injection risk ranks the first on the list of top 10 security risks that are the most probable, universal and dangerous in the web application programs according to the OWASP (Open Web Application Security Project) in 2013 and 2016. Usually, attackers attack the WEB servers and backend database to tamper the webpage contents and even steal important internal data by utilizing the SQL injection flaws. The worst situation is to embed malicious codes, which violates the interests of the visitors. For example, in 2011, 380 thousand sites were attacked by the Liza Moon SQL injection attack in April, and various sites of SONY Corporation were frequently attacked by the SQL injection in April and May. In October of the same year, 200 thousand sites were affected by the SQL invasion, and more than 1 million Web pages of ASP.NET platform were influenced. The SQL injection attacks like these mentioned above keep emerging. According to the report of "BBS for detecting hackers: ADC analysis of monthly network attack (October, 2012) " published by Inperva corporation, the topics concerning SQL injection account for 19%.

Therefore, it is very significant to study the detection and prevention technology targeting at SQL injection attack to improve the operation security of Web environment. Thus many researchers have carried out a lot of the work with respect to detection and defense of SQL injection attack, which are as follows: the program code written specifications; research of defense model, the literature[4] [16] [17][18]give diversity of ways to prevent automatic models; the detection tools & secure defense algorithm, Stephen (2009) created a prepared statement replacement algorithm and a corresponding tool for automated fix generation [12], Kanchana (2012) provide SQL-injection free (SQL-IF) secure algorithm to detect and prevent SQL-injection attacks (SQLIAs) [13]; data encryption stored, Indrani (2012) generate ASCII value in database for the sensitive data's as Userid ASCII and Password ASCII to secure the database from the attackers[14],The literature [15] presents a new practical protection mechanism against SQL injection attacks by using Base64-data-encoding; static and dynamic analysis & smart detection, Jang (2014) exhibit a kind of technique which dynamically analyzes the developer-intended query result size for any input, and detects attacks by comparing this against the result of the actual query [10], Inyong (2012) proposed an effective and simple novel method for detecting SQL injection attacks by comparing static SQL queries with dynamically generated queries after removing the attribute values [11].

In this study, the ability of resisting SQL attack was comprehensively improved from the perspective of server security and policy settings of database. Furthermore, if the server has already been attacked by SQL injection, the defense remedy model of SQL injection attack established in this study will play a role in resisting the attack. For the users who cannot modify the source codes in a short time, the model can be taken as a good remedial measure or an application scheme to prevent the SQL injection attack.

INTRODUCTION OF SQL INJECTION

SQL injection is a database attack and a manifestation of the existence of flaws in WEB application program. The mechanism of attack is to insert the user data into the actual database manipulation language by using the peripheral interface of some database. In this way, the goal of invading the database and even the operating system can be realized. The reason of SQL injection attack is that programmers only pay attention to the realization of business logic, and do not consider robustness and security of codes due to the lack of security awareness. When editing the codes, they do not judge the validity of input data, resulting in the security problems of the application program.

SQL injection attack is the network attack implemented by SQL injection technology. It is a great and universal threat to the network. Any database server that supports the batch processing of SQL commands (such as MySQL, MSQL, Oracle, DB2 and Sybase) may be attacked. Moreover, since the attack procedure is simple and requires low technical skills, it is not necessary for attackers to have specialized knowledge of SQL injection. Without fully mastering the related knowledge, some good injection tools (such as HDSI, Sqlmap, Bobcat, BSQL, NBSI and Domain) can be used to attack and destroy the website or Web application program with SQL injection flaws. SQL injection accesses the server from the HTTP service port (usually the port 80) by requesting to visit pages. It seems that there is no difference from the normal web access. Alarms will not be raised (common network firewalls are open to HTTP/HTTPS) even without firewalls. Therefore, SQL injection is well concealed.

Analysis of the Procedure of SQL Injection

The methods of SQL injection are flexible, and the attack techniques are diverse. The injection can break the defense of most of general firewalls. The procedure of SQL injection is usually divided into the following five steps, as shown in Fig. 1.

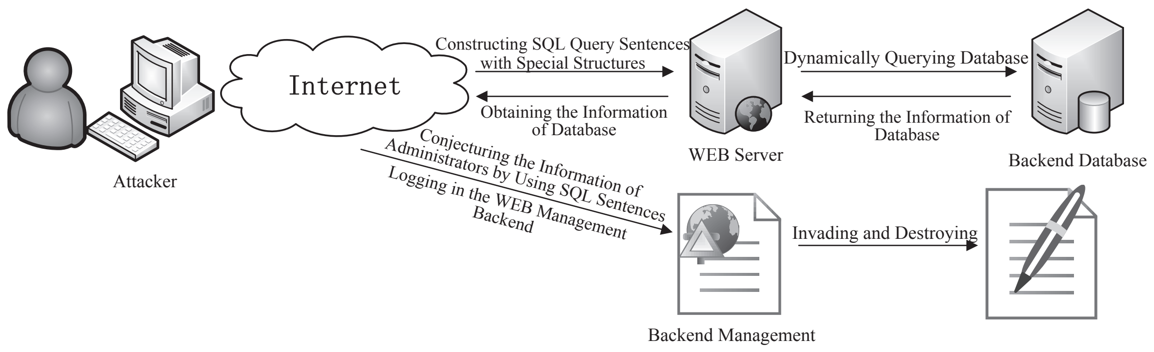
Attacker
Internet
Constructing SQL Query Sentences with Special Structures
Obtaining the Information of Database
WEB Server
Dynamically Querying Database
Returning the Information of Database
Backend Database
Conjecturing the Information of Administrators by Using SQL Sentences
Logging in the WEB Management Backend
Invading and Destroying
Backend Management

FIGURE 1. Procedure of SQL Injection

(1) Seeking Injection Points

Injection point is the web pages with injection flaws. There is no injection in the static pages of URL ending with html or shtml. The injection may only occur in the dynamic pages of URL with '?'. This is because page information is generated by obtaining information from database according to different query parameters. Hence the access to database needs to be supported by the database. Since the HTTP request has two ways, GET and POST, SQL injection attack also has two injection ways, GET injection and POST injection, respectively. It can be determined whether there is any SQL injection according to the false information returned by the browser by inputting some special characters such as 'and 1=1' and 'and 1=2' in the address bar of browser or the page table. In this way, the injection point can be found.

(2) Obtaining the Information of Database Type

To obtain the information such as the database type is an essential part of SQL injection. The database type can be judged according to the error messages generated by SQL sentences with special structures. Suppose there is a URL address: http: / / www.***. Com/show.aspx? id= 6. If a single quotation mark is added to the end of it, i.e., http: / / www.***. Com/show. aspx? id= 6', the type of backend data can be judged according to the returned result. This step prepares for subsequent injection. If there is any error such as 'Microsoft JET Database Engine Error '80040e14'' on the pages, the ACCESS database can be judged as the backend database. If the result of Microsoft OLE DB Provider for SQL Server error '80040e14' is returned, i.e., there is an unclosed quotation mark before the string, it means that the single quotation mark is not filtered and the database type is SQLSERVER. In this type of database, "sa" is taken as the default account of the administrator of database system. If the attacker confirms that the backend database of web application is SQLSERVER, the default methods for cracking the password can be used to obtain the account information of system administrator.

(3) Conjecturing about the information such as Table Names, Field Names, User Names and Passwords of database

There are regulations on the table names and field names stored in the database [1]. For example, the information of other databases available in the MySQL server is stored in the information database information schema that comes with the MySQL database. In the current MySQL instance, all the database information and table information are stored in the SCHEMATA table and TABLES table, respectively. Moreover, all the column information is stored in the COLUMNS table.

It is not simple to conjecture the table names and field names of database [2]. To promote the success rate of conjecturing, table names and field names to be cracked need to be further judged after the lengths of them are confirmed. Attackers can query the lengths and contents of table name, field name, user name and password successively in the database by establishing special database access sentences. Specifically, the corresponding SQL test sentences are added to the end of normal URL. Then after the request is submitted, judgments are made according to the response of web server. If normal webpage contents can be obtained after submitting, it means that the table names or field names conjectured by attackers are correct, otherwise they are wrong. Then attackers should continue making further assumptions and submitting test sentences until the web page contents are shown successfully. This conjecturing process can be soon realized through a large amount of existing injection tools on the Internet, and the user's passwords are cracked by cracking the sites.

(4) Searching for the Entry to Management Backend of Web System

In general, registered users of web system can only access the Web server by logging in the fore-end. The management interface of Web backend system is not open to normal users. To find the login path to backend, scan tools can be used to search quickly, and all the possible login addresses are tried successively. After the entry address for management backend is known, the previous administrator information is used to login in the Web system. This means that the administration authorities of the web system are taken over.

(5) Invading and Destroying

After logining in the backend management, the attackers can destroy the system, such as illegally publishing information, distorting webpages, uploading Trojan virus, and revising, stealing and leaking users' profiles. Then they invade the database servers.

SQL INJECTION VULNERABILITY DETECTION

Manually finding SQL injection vulnerabilities

According to the difficulty of leak detection, SQL injection vulnerabilities in Web application system is divided into three levels, as shown in Table. 1.

(1)The first level of SQL injection vulnerabilities

The simplest approach to detect SQL injection vulnerabilities is to construct SQL clause dynamically by using as single quotes, double quotation marks and ‘ and 1=1’,for example, if there is a web login page, the login validation SQL code as follows:

select * from admin where username=’"+ userInput.Text + "’ and user Pass=’ "+ passInput.Text +’"

When input "user' or l=l--" in the textbox, we are able to pass the authentication successfully. Because "--" is SQL comment, the codes following it are commented, so "l=l" becomes identity, the condition of SQL clause is always true.

TABLE 1. Three levels of SQL injection vulnerabilities

level	description	example
first level	error triggering	’
	always true condition	1’ or ‘1’=’1
	Always false condition	1’ and ‘1’=’2
	no condition	value’ or ‘1’=’2
	SQL comment	- -
	Microsoft SQL Server concatenation	1’ or ‘ab’=‘a’+’b
second level	variant keyword	uPdate as update
	ASCII conversion	a=chr(97)
	restructuring	aandnandd as and
	UNICODE coding	and%201%3Dl as and 1=1
third level	Use stored procedures to complete access to the database	’;exec master.xp_cmdshell ’ping 127.0.0.1’- -

(2)The second level of SQL injection vulnerabilities

Besides the basic method of detecting SQL injection vulnerabilities by constructing dynamically SQL clause, these methods can also achieve such as variant keyword, ASCII conversion, restructuring and UNICODE coding and so on. For example, partly or all, using ASCII to represent the input characters, it is able to bypass Web application filter based on keywords or special characters. Because Characters and ASCII is one to one correspondence relation, such as a=chr(97) and b=chr(98).If represent "or l=l", the method of ASCII conversion is "chr(111)chr(114) l=l". As another example, "and%201%3Dl" is equivalent to "and 1=1". When using Unicode to encode special characters in URL, the web server could decode automatically, such as Space’ Unicode is "%20", equal sign’ Unicode is "%3D". It is obvious that UNICODE coding is much more concealed.

(3)The third level of SQL injection vulnerabilities

This level do mainly test some specific SQL procedures stored to complete access to the database, For example "’; exec master.xp_cmdshell ’ping 127.0.0.1’- -" can execute ping command.

Automatically Finding SQL Injection

When Web application has a lot of codes, it is very hard and inefficient to manually find SQL injection vulnerabilities. Automated tools are systematic and thorough. They don’t understand the Web application logic, but they can test very quickly a lot of potential injection points which is something that a human cannot do thoroughly and consistently [9]. There are some commercial tools and open source tools such as HP Web Inspect, IBM Rational AppScan, HP Scrawl, SQLiX Paros Proxy and so on. Although automatic discovery tools may not identify all the existing vulnerabilities, they provide security assessment of a Web site, including testing for SQL injection vulnerabilities.

DETECTION AND DEFENSE OF SQL INJECTION ATTACK

Detection of the SQL Injection Attack

There are two ways of SQL injection attack. One is the manual way of establishing some abnormal inputs, and the other one is to use tools [3]. Regardless of which attack means is used, traces will be left in the system after the successful SQL injection attack. There are four effective ways for judging whether the Web system has been attacked by the SQL injection or not [4].

(1) To check if there is any abnormal data table in the backend database. Some temporary tables will be generated in the database after the SQL injection attack is performed by using the software’s such as HDSI and NBSI.

(2) Logs of backend database are checked. After the log management is started by the backend database, accesses to the database will be recorded. In particular, SQL injection can be found from the logs if the falsely executed SQL sentences are recorded.

(3) To check the IIS logs. In the Web server, detailed information such as IP addresses, access time, access files and access ways of the visitor is recorded in the IIS logs. For the SQL injection attack, some page files with SQL injection point are usually frequently accessed. The log file is large, hence it can be judged whether there is any SQL injection attack by checking the size and content of log files.

(4) It can also be determined whether there is any intrusion by checking the information such as account number of system administrator, status of port opening, the files generated lately in the system, virus and logs of firewalls.

Prevention of the Non-Intrusive SQL Injection Attack

A Web system administrator should develop the habit of data backup and check the log information of database, IIS and firewall regularly. If there is any unusual behavior, the administrator should locate the injection point quickly and analyze the reason. If the failure is caused by a security loophole of the operating system, then update and patch should be installed on the operating system in time. If it is the problem of the Web system program itself, the administrator should carry out defence remedy measure immediately, then modify the codes. When the program codes are modified and improved, a programmer can conduct strict filtering, detection and parametric SQL query on the dynamically constructed SQL statements with codes. The approach involves source code modification. It is necessary to have the source code and modify it. Hence, it is an intrusive solution [5], which is not considered in this study. On the contrary, by taking Windows+SQL SERVER+ASP.NET as an example, a non-intrusive approach for preventing SQL injection attack is proposed from the aspect of security configuration on server and defensive remedy.

Security Configuration of Servers

(1) Settings of Operating System and IIS

The basic environment running the Web system is the operating system and IIS. Their security is fundamental for the whole Web system to defend SQL injection attack. For the operating system, system update and patch should be installed regularly. Necessary Internet firewalls and virus firewalls should be installed, unnecessary services and ports should be shut down, and related dangerous components should be disabled. Moreover, Web applications should be operated in the mode with the least authority. The directories where static and dynamic web pages are located should be set with different authorities. Directory should be configured with no authority. For example, for the directory o files uploaded by the management center on website backend, the execute authority should be set as "none". This will limit the execute authority of uploaded files. Even when a Trojan program is uploaded, it will not be operated.

The function of HTTP error message prompt in IIS makes it very convenient for Web application developers to debug. However, it also brings opportunities for SQL injection attackers. The error prompt can be disabled or redirected. Analyses on the types of programs, such as CGI, ASP and ASPX, are enabled by IIS. SQL injection attacks can also be reduced by maintaining necessary extension of Web services according to the demand of Web system application. In IIS, several ISAPI such as Rewrite and URLScan should be configured. URL rewrite technique is adopted in Rewrite to realize the pseudo static of the webpage, so that the real URL path is concealed. Therefore, the brute-force on string by the attacker can be prevented, and the security of the webpage is improved. URLScan can monitor HTTP server requests in real-time, and shield the execution of malicious codes on server. In this way, SQL injection attack is prevented.

(2) Settings of Database

The core of SQL injection is database, hence the security of database itself is very important. The following problems should be considered in the security configuration of the database.

1) Account Management

The default account of SQL Server database is sa, and it is the authority of system administrator. The authority of default account sa of the system can be decreased, such as only owning the authority of read or write, or the default system account is modified. Moreover, strong password should be set, and redundant accounts are deleted. Besides, the specific user authority of handling with tables in SQL Server database should be carefully set up according to different demand and function of application.

2) Remote Service Management

If the database and Web program are on the same server, external connection of port TCP 1433/ UDP 1434 should be closed. That is to say, to close the TCP/IP remote access of database; if the database and Web program are not on the same server, IP visit can be restricted by revising the security policies of server where the database exists.

3) Management of Extended Storage Procedure

The extended storage procedure that is not or rarely used are deleted [6]. For example, xp_cmd_shell procedure can call Windows CMD process to execute the Windows script and dangerous operations of adding users and revising the authorities. As another example, xp_regwrite procedure and xp_regdeletevalue procedure are able to modify or delete windows registry.

4) Role Management [7]

The lowest authority for accessing the web application program of database is assigned. The Web application program is forbidden to connect database by using the account of database administrator.

Remedy

It takes some time for the Web application program in the process from the functional demand analysis, design, and development to delivery. If any SQL injection due to the defects of source code is found during use, the program should be returned to the developer for revision. It takes much time for analyzing the reason of injection, searching for the flaws of source code and making revision plans. During this period, servers should be closed, and hence adverse influences will be brought to the users.

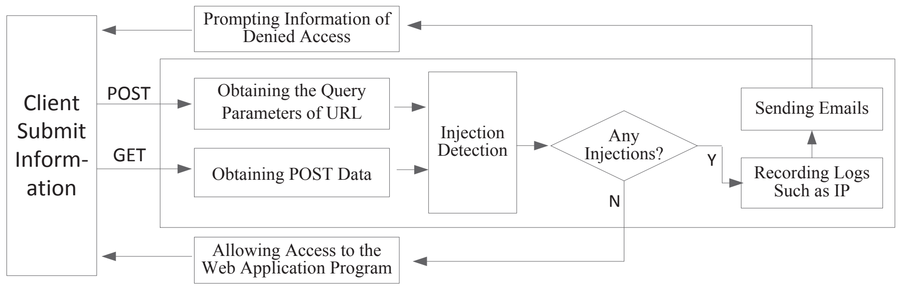
Prompting Information of Denied Access
Client Submit Inform-ation
POST
GET
Obtaining the Query Parameters of URL
Obtaining POST Data
Injection Detection
Any Injections?
Y
N
Sending Emails
Recording Logs Such as IP
Allowing Access to the Web Application Program

FIGURE 2. Defense Remedy Model of the Non-Intrusive SQL Injection Attack

The process of remedy for non-intrusive SQL injection attack is as follows (Fig.2): injection detections for the POST and GET request are added to the front end of Web application program that is officially accessed; For any access with injection, the log information such as IP, access methods and access pages are recorded; Then an email is sent to the Web system administrator to forbid the client to access further, which is a reminding and warning. The model does not involve source code modification of Web application program. After the server is attacked by the SQL injection, the SQL injection detection can be added to the front end of the Web application program, and thus plays a role in resisting the SQL injection attack. For the users who cannot or need a longer time to modify the source code, this method can be used as a good remedial measure or application scheme for preventing the SQL injection attack.

TABLE 2. Five basic events in the Global. Sax file

NO.	event	description
1	Application Start	Triggered when the application starts
2	Application End	Triggered when the application ends
3	Application Error	Triggered when the application sends an error
4	Session Start	Triggered when the application starts each session
5	Session End	Triggered when the application ends each session

Under the ASP.NET environment, Global. Sax is a text file, which provides globally available codes. Definition of the codes consists of global events such as event handling procedure, session event, method and static variables. In these codes, five basic events are defined for Application object and Session object [8], as shown in Table 2. Besides, other events such as Application_BeginRequest and Application_EndRequest are also included, where Application_BeginRequest is triggered when a request is sent by a webpage (a POST or GET request). The key to the non-intrusive defence and remedy model for SQL injection attack proposed in this study is to perform extended programming on these events.

The following content shows the main code segments of Application_BeginRequest in Global.asax.

```
Void Application_BeginRequest(object sender, Eventers e)
{Bool isSqlInjection = false;
If (Request.RequestType.ToUpper() == "POST")  //check POST data
isSqlInjection = SQLInjection.ValidPostData();
Else if (Request.RequestType.ToUpper() == "GET")  //check GET data
isSqlInjection = SQLInjection.ValidGetData();
If (isSqlInjection)
{SQLInjection.WriteLog (); //record SQL injection logs
SQLInjection.SendEmail (); //send emails to administrator
Response. Write ("exist SQL injection or malicious characters.");
Response. End () ;}}
```

An App_Code directory is built under the root directory of Web application program, and SQLInjection.cs is created. The class of SQLInjection is built and methods such as ValidPostData and ValidGetData are realized. The main codes are shown as following.

```
Public class SQLInjection
{     Private const string StrRegexEx = @".*(drop table|update|select|insert|delete|from|count(|
 truncate|asc(|mid(|char(|xp_cmdshell|exec master|:|net user|’|or|and).*";
 Public static bool ValidPostData() // validate the data by POST request
 {    Bool isSqlInjection = false;
 For (int i = 0; i < HttpContext.Current.Request.Form.Count; i++)
 {    IsSqlInjection=ValidData (HttpContext.Current.Request.Form[i].ToString());
 if (isSqlInjection)  break;  // exist SQL injection }
 Return isSqlInjection;}
 Public static bool ValidGetData() //validate the data by GET request
 {    Bool isSqlInjection = false;
 for (int i = 0; i < HttpContext.Current.Request.QueryString.Count; i++)
 {    isSqlInjection=ValidData(HttpContext.Current.Request.
 QueryString[i].ToString());
 if (isSqlInjection)  break;        //exist SQL injection }
 Return isSqlInjection;}
 Public static bool ValidData(string strData) // By regex to determine whether exist  SQL injection
 {    if (Regex.IsMatch(strData, StrRegexEx))  return true; Else return false ;}
 }
```

CONCLUSION

SQL injection attack is easy to be carried out with high concealment. It has become a serious threat to WEB application security. In this study, the characteristics, process and procedures of SQL injection are analyzed, and the approaches for the detection of SQL injection attack are explained and summarized. Taking the environment of Windows+SQL SERVER+ASP.NET as an example, the security of Web application is fully improved with the security strategies by configuring the operating system, IIS and data base, from the perspective of prevention non-intrusive SQL injection attack. Moreover, the model for protecting and remedying SQL injection attack is established, and the related code is implemented. By applying the program on actual projects (several servers in the university), it is proved that the proposed approach is able to protect the network from SQL injection. All the debug works are well performed under Win7+VS.net 2008 environment.

ACKNOWLEDGMENTS

The work is supported by the Scientific Research Project of Sichuan Provincial Department of Education of China (16ZB0184), the Foundation Project of China West Normal University of China (15D002).

REFERENCES

1. Introduction of SQL Injection Attack and Defense, http://www.h3c.com.cn/About_H3C/Company_Publication/IP_Lh/2012/06/Home/Catalog/201212/769059_30008_0.htm，2012.
2. Review of the Security of Microsoft SQL Server, [EB/OL]. http://wwvv'.microsoft.com/china/cte/ctc/Newsletter/04/ ctc2.htm.2004 3 25.
3. Wang Shihui and Zhang Xiyun. Research of cause, detection and defense measures of SQL injection. Journal of Hubei University (Natural Sciences). 2010, 32 (3): 269-272.
4. Chen Xiaobing, Zhang Hanyu, Luo Liming and Huang He. Research of SQL injection attack and technology of defense and detection. Computer Engineering and Applications, 2007, 43 (11): 150-152.
5. Qi Mingxingchen. White Paper of Defense Technology for SQL Injection and XSS Attack, 2010: 3-9.
6. Wang Zhihu. Research of SQL injection attack and prevention measures. Coal Technology, 2011, 30 (1): 95-97.
7. Wang Yun, Guo Waiping and Chen Chenghuan. Research on the SQL injection problem and defense methods in Web projects. Computer Engineering and Design, 2010, 31 (5): 976-978.
8. Li Yulin and Wang Yan. Mastering ASP.NET 2.0 Network Programming [M]. Beijing: Tsinghua University Press, 2006.9.
9. Justin Clarke. SQL Injection Attacks and Defense [M]. Elsevier.2012.
10. Young-Su Jang, Jin-Young Choi.Detecting SQL injection attacks using query result size.Computers&Security, 2014, 44:104-118.
11. Inyong Lee, Soonki Jeong, Sangsoo Yeo, Jongsub Moon. A novel method for SQL injection attack detection based on removing SQL query attribute values. Mathematical and Computer Modelling, 2012, 55:58-68.
12. Stephen Thomas, Laurie Williams, Tao Xie. On automated prepared statement generation to remove SQL injection vulnerabilities. Information and Software Technology.2009, 51:589-598.
13. Kanchana Natarajan, Sarala Subramani.Generation of SQL-injection free secure algorithm to detect and prevent SQL-injection attacks. Procedia Technology, 2012, 4:790-796.
14. Indrani Balasundaram, E. Ramaraj. An Efficient Technique for Detection and Prevention of SQL Injection Attack using ASCII Based String Matching. Procedia Engineering, 2012, 30:183-190.
15. XUE Yu-chun, HUANG Dong. A new approach for preventing SQ L injection attacks. Computer Knowledge and Technology, 2006, 2:121-122.
16. Witt Yi Win, Hnin Hnin Htun. A Simple and Efficient Framework for Detection of SQL Injection Attack. International Journal of Computer & Communication Engineering Research (IJCCER).2013, 1(2):26-29.
17. LIU Shuai. Research on SQL injection Attack Detection and Prevention Technology. Computer Knowledge and Technology, 2009, 5(28): 7870-7872, 7898.
18. ZHANG Y ong, LI Li, XUE Qian. Detection and Defense against SQL Injection Attacks.2004, 15:103-105,108.