International Journal of Advance Research in Science and Engineering
Volume No.06, Issue No. 09, September 2017
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

Detection of SQL Injection Vulnerability For Web Based Applications

Soumya R Vastrad[1],Manohar Madagi[2], Veeresh Hiremath[3]

[1]Department of CSE, KLE Institute of Technology Hubballi, (India)

[2]Department of CSE, KLE Institute of Technology Hubballi, (India)

[3]Tattva Labs, KLE Institute of Technology Hubballi, (India)

ABSTRACT

As the popularity of the web increases and web applications become tools of everyday use the role of web security has been gaining importance as well. Pervious few years have shown significant increase in the number of web based attacks. Too many known web application security vulnerabilities result from generic input validation problems. SQL injection is one of the examples. This paper discusses the implementation of the automated vulnerability scanner for detection of SQL injection attack. This tool analyzes web sites with the aim of finding exploitable SQL injection vulnerability.

Keywords: Security Vulnerabilities, SQL, SQL Injection.

I. INTRODUCTION

Web applications critical part is security. Web applications allow users to access the central resource the web server and through it others like database servers. By implementing and understanding security measures you provide secure environment as well as security to you own resource with which users can comfortably work with your application [1].

Information security has a branch called web application security which deals with security of web sites, web services and web applications.

Web threats are blocked to reduce malware infections, free up valuable IT resources and decrease incidents at helpdesk [4, 7]. They have hundreds of protocol control and web applications, 60 plus customized reports and role based access and also has more than 100 security filtering categories [6]. Web security gateway can be easily upgraded when desired to get, SSL inspection, data loss prevention (DLP), social media controls and inline, real time security from web sense ACE(advanced classification engine) [2]. Weakness or hole in application is called vulnerability which can be due to implementation bug, which allows attacker to cause harm to stakeholders of application or due to design flaw. Stakeholders are application users, application owner or other entities which depend on application. The top ten vulnerabilities [5] updated by open web application security projects are SQL injection, Broken authentication and session management, Cross Site Scripting (XSS), Insecure indirect reference object, security misconfiguration, Sensitive data exposure, Missing function level access control, cross site request forgery [7].

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

II. SECURITY TESTING

Security testing is a part of quality assurance process which is responsible for discovering and correcting flaws created during the implementation and design of the software vulnerability analysis is a art and science of discovering security problems or other weaknesses in the software system. Vulnerabilities in software introduced by poor practices or by mistake is serious problem. In most of the software today numerous flaws and errors are often located and exploited by attackers to compromise the software's security and other required properties[3]. If the internet users believe that their personal information is not secure and is available to those with intent to do harm, the future of e-commerce is in danger. Designing and testing software system to insure that they are safe and secure is a big issue facing software developers and test specialists[9]. Security testing evaluates system characteristics that relate to the availability, integrity and confidentiality of data and services server as a baseline for security requirements.

1.1 Security Testing Stratergies

The Various organization around the world audit software. This is done for quality assurance reasons as third party auditors or hackers look to find bugs for fun or for profit [3]. Security testing encompasses several strategies they are their main approaches to testing software application for the presence of bugs and vulnerabilities.

1.1.1 Black box testing

Black box testing is also known as data driven or input/output driven testing. It refers to testing that only operate by launching attacks against an application and observing its response to those attacks to use this method of testing we have to view the program as black box, the tester only knows the inputs that can be given to the system which comes into contact with the test item program or program unit only.

1.1.2 White box testing

Another testing strategy is white box testing as the name implies that the tester has access to application's source code and therefore has significant knowledge of the application under test[6]. It is easy to make sure the program will function as required as it is possible to directly test conditions in the program flow with white box testing it is possible to craft input that allows every line of code to be executed and tested from security perspective, it is important in white box testing to provide the program with input the program normally would not expect in order to see the program deals with the input.

1.1.3 Gray box testing

Gray box testing is a combination of functional and structural approaches and is usually applied during system testing. Gray box testing like black box testing is concerned with accuracy and dependability of the input and output of the program module. The test cases, test method and risk assessment involved in gray box testing are developed based on attempts to extract information about the application from the executable code by disassembling the binary files or even reverse engineering the application.

II. SQL INJECTION VULNERABILITIES

Injection weakness like SQL, LDAP and OS injection occur when trusted data is sent to an interpreter as a part of command query. The attacker can access unauthorized data by executing unintended commands. It is not mandatory for an attacker to visit the web pages using a browser to find if SQL injection is possible on the site [5]. Usually attackers build a web crawler to collect all URLs available on each and every web page of the site.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

Web crawler is also used to insert illegal characters into the query string of URL and checks for any error message sent by sever. If a server sends any error message as a result, it is a strong positive indication that the illegal special Meta character will pass as a part of SQL query and hence site is open to SQL injection attack.

SQL injection is the most common attack because of the popularity. Most of the relational databases speak SQL language. Most of the web applications both larger and the smaller ones use the database to store information which are used for operating. To control the data directly SQL injection attack uses sequence of commands from the structured query language (SQL). SQL is one of the universal languages suitable for all databases [4]. Most of the times applications use the SQL injection attack. To check for SQL injection the tool creates two test cases

1.1 Automatic insertion of 1' in every text box available in the web form.

1.2 Modifying the URL by replacing = → ='

After the test cases are prepared, each test case is passed through every link and web forms of the website which is given as input. Using these test cases, the test is performed on the website. Scanner builds the HTTP request to send it to the respective given website and in response URL will give the HTTP response.[4] The response contains the details of the website, further tests will be done on the details. If any syntax error is found per database server in response text or any internal error in a web form is found then the website or link is vulnerable to SQL injection.

To check for syntax error the tool stores all the patterns of syntax error in the pattern.xml file. If any of the patterns found in the response text matches with the patterns already present in pattern.xml file then the given website or the link is vulnerable to SQL injection.

III. PROPOSED SYSTEM

The proposed system is more innovative and provides better results than the existing system "SQL injection scanner" is basically software designed in python to scan the website for SQL injection vulnerability that sustains in the websites and providing report to the client.

This software takes URL of website as input, to test for SQL injection vulnerability. It will check whether website exists in the domain or not. If not then asks user to enter valid Input i.e. valid URL of website. The status code is checked to see whether the website really exists or not. If the status code is 404 then the URL does not exists. If the website i.e the URL is valid then the tool crawls down the website with the help of developed crawler "basic crawler" basically made in python. The crawler parses the website. The output of this phase is multiple forms details and platform used for developing web pages of respective website. Scanner tests for all forms and links of the website for SQL vulnerability after parsing the website. Scanner takes the URL of the website as an input, to test for SQL injection vulnerability. It will check whether the website is present in the domain or not. If not, system asks for valid input. It checks whether the website really exists or not by checking the status code. If the status code is 404 then the URL does not exist. For valid URL it crawls down the website. The output of this phase is multiple frameworks form details and platform used in developing web pages of respective website. After parsing website scanner tests all the forms and the links of the website for vulnerabilities.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

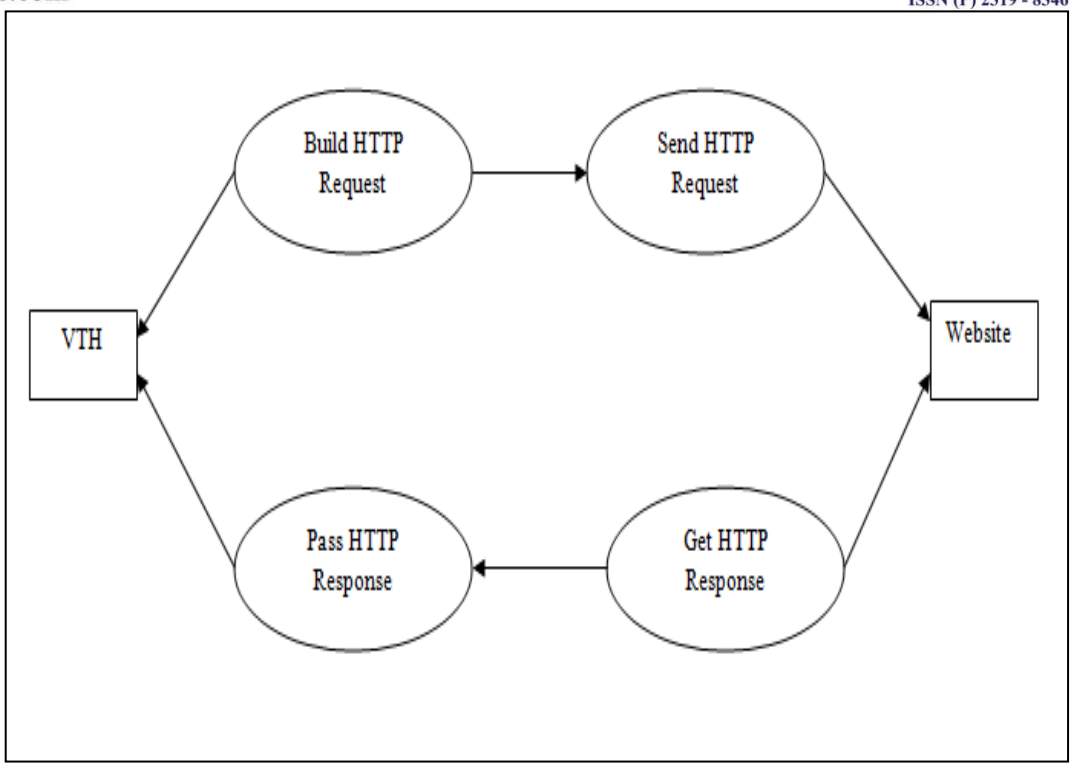

Fig. 1. Web Client Module

Fig. 2.

The system is divided into four main functional models

4.1 The Spiral Model

This model takes the function of interaction between the system and the website. It first generates web request by the given URL. When the web server receives the request, it returns a response to client. The response contains status code, cookies and message body and HTML content in the message body then the forms are extracted sequentially.

4.2 The Injector Model

This model sends invalid payload to the affected parameter in the HTTP request, the proposed method sends set of payload attacks instead of sending single payload attack to reduce the time consumption in the testing process maximum set of length is taken which is accepted by the input field, after testing all set of payload and no vulnerability found in each set and retry testing with new set reach the single character of payload. The injector method vulnerability detection can happen only by sending one request of payload rather than sending many.

4.3 The Analyzer Model

The proposed method is to use the hash value to detect if website has vulnerability; first request is sent as normal request such as http://localhost/index.php?id=1 then compute hash value for response of the web page. Next the entire malicious request will be sent one by one and for each response hash value is computed and compared.

International Journal of Advance Research in Science and Engineering
Volume No.06, Issue No. 09, September 2017
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

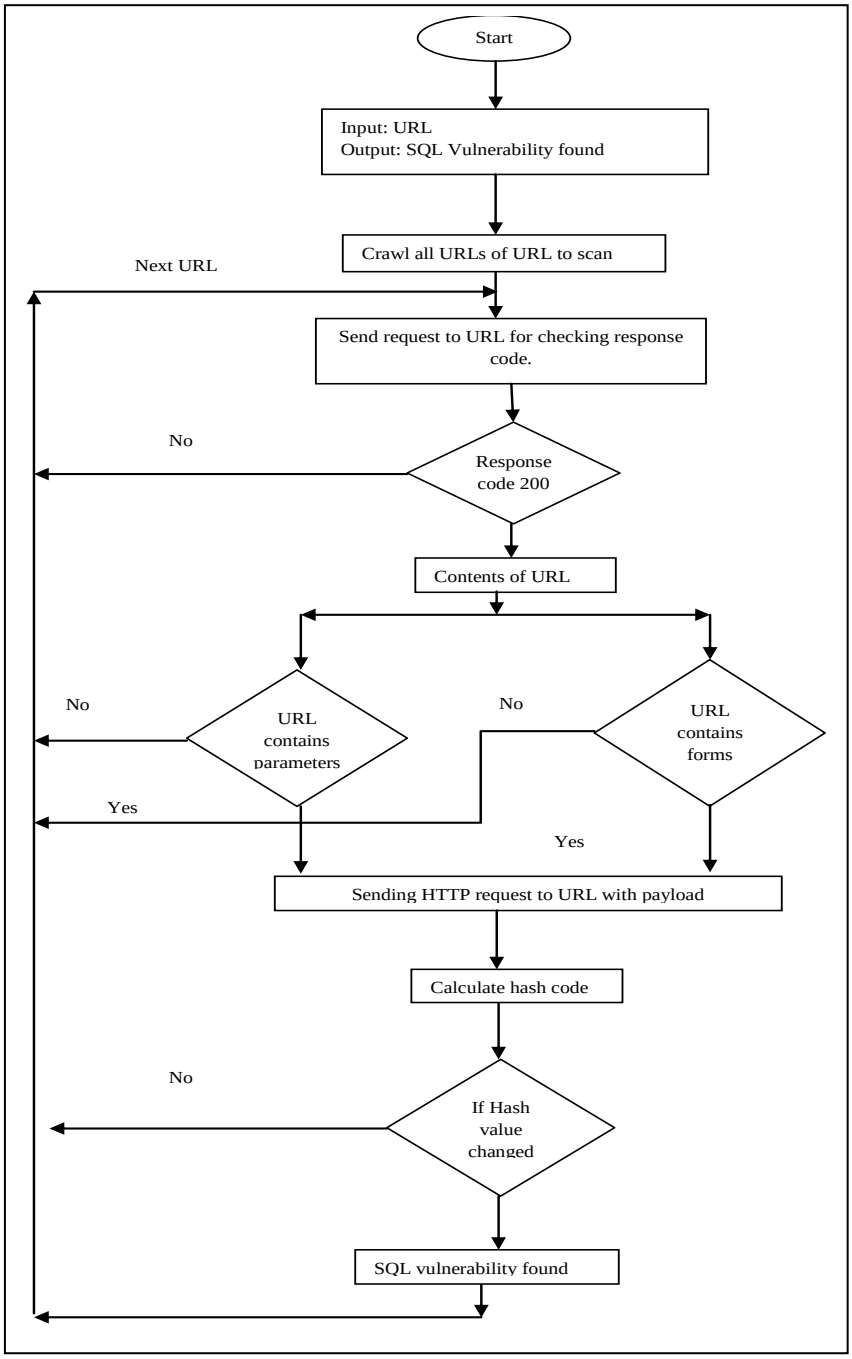

Fig 2. Proposed system

V. CONCLUSION

SQL injection is threat to any system that is connected to internet with database backend. An intelligent attacker will find flaws in the code and/or in the server security and leverage them to their maximum advantage. An SQL injection is easy to guard against and should be considered for any project before a line of code is written. In this project we have discusses various types of vulnerabilities among which SQL injection is more commonly used. We have also discussed ways of detection perform this we have discussed how SQL injection works and its root cause. It does highlight several limitations which form the foundation for future work in this area. The future

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

scope of our application is first, we are not providing the patch for the detected vulnerable URL to overcome this we are planning to generate patches and providing solutions in future. Second, rich area for future work is exploring many more vulnerabilities, detecting them and providing solutions. Third, is finding best techniques among the various methods for their evaluation.

REFERENCES

[1] Maki Mahdi, Ahmed Hashim mohammad, *using hash algorithm to detect Sql injection vulnerability*, International journal of research in computer applications and robotics *volume 4 issue 1* january 2016.

[2] Rahul Kumar, Indraveni K, Aakash Kumar, *Automation Of Detection Of Security Vulnerabilities In Web Services Using Dynamic Analysis*, International Conference For Internet Technology And Secured Transactions 2014.

[3] Nor Fatimah awing, Azizah Abd Manaf and siti Fatimah abidins, *Test input generation for detecting SQL injection vulnerabilitu in web application*, International journal of soft computing *11(2),103-106*, 2016.

[4] Parveen Sadotra, *Hashing Technique Sql Injection Attack Detection And Prevention*, International Journal of Innovative Research In Computer And Communication Engineering May 2015.

[5] Surya Pratap Singh, Upendra Nath Tripati, Manish Mishra, *Detection And Prevention Of SQL Injection Attack Using Hashing Technique*, International Journal Of Modern Communication Technologies And Research *volume 2, issue 9*, September 2014.

[6] Jose Fonseca, Macro Vieira, *Mapping Software Faults with Web Security Vulnerabilities*, International Conference on Dependable Systems And Networks *24-27* June 2008.

[7] Vieira, *Using Web Security Scanners to Detect Vulnerabilities in Web Services*, IEEE/IFIP International Conference on Dependable Systems and Networks *DSN 2009*, June 2009.

[8] Wei, K., Muthuprasanna, M., & Suraj Kothari, *Preventing SQL Injection attacks in stored procedures*, Software Engineering IEEE Conference. Retrieved November 2007

[9] Thomas, Stephen, Williams, & Laurie, *Using Automated Fix Generation to Secure SQL Statements*, Software Engineering for Secure Systems IEEE CNF. Retrieved November 2007

[10] Merlo, Ettore, Letarte, Dominic, Antoniol & Giuliano, *Automated Protection of PHP Applications Against SQL-injection Attacks* ,Software Maintenance and Reengineering, 11th European Conference IEEE CNF. Retrieved November 2007

[11] Wassermann Gary, Zhendong Su, *Sound and precise analysis of web applications for injection vulnerabilities*, ACM SIGPLAN conference on Programming language design and implementation PLDI, November 2007

[12] Friedl's Steve Unixwiz.net Tech Tips, *SQL Injection Attacks by Example*, Retrieved November 2007

[13] Yuji Kosuga, Kenji Kono, Miyuki Hanaoka, *Syntactic and Semantic Analysis for Automated Testing against SQL Injection*, IEEE Computer Society. November 12, 2007