International Journal of Advance Research in Science and Engineering
Vol. No.4, Special Issue (01), Spetember 2015
www.ijarse.com
IJARSE
ISSN 2319 - 8354

ANALYZING AND DEVELOPING AN AUTOMATED TOOL FOR DETECTION OF BOTH SQLI AND XSS ATTACKS

Swapna Mohan M[1], Divya B[2]

[1]*Computer Science and Engineering, Vimal Jyothi Engineering College, Kannur, Kerala, (India)*

[2]*Asst.Professor, Computer Science and Engineering, Vimal Jyothi Engineering College, Kannur, Kerala, (India)*

ABSTRACT

Web applications are one of the most widespread platforms for information and service delivery over the Internet. Because of the widely exposed feature of web application or services, any web security vulnerability will mostly be observed and be exploited by hackers. The Structured Query Language (SQL) Injection and Cross Site Scripting(XSS) are the greatest security risks in the world according to the Open Web Application Security Projects (OWASP) Top 10 Security vulnerabilities.

One of the most commonly found attack is SQL injection. Today many researches are done in SQL injection. There exists different tool to detect SQl injection .Also XSS is another commonly found attack .There exist method to detect XSS. In this survey, a detailed survey of different types of SQL Injection and XSS attack detection methods. Finally reach a conclusion to develop a tool to detect both SQLI and XSS.

I. INTRODUCTION

Today's Web applications can contain dangerous security flaws. The global distribution of these applications makes them prone to attacks that uncover and maliciously exploit a variety of security vulnerabilities. The two most common risks in the Web environment are SQL injection, in which attackers alter SQL queries sent to a database and cross site scripting (XSS),are also most dangerous.SQL injection is a code injection technique commonly used to attack websites in which the attacker inserts SQL characters or keywords into a SQL statement via unrestricted user input parameters to change the intended query's logic. This threat exists in any Web application that accesses a database via

SQL statements constructed with external input data. Cross Site Scripting (XSS) is a vulnerability in web applications that allows an attacker to inject HTML, typically including JavaScript code, into a web page. XSS results from the intermingling of server code and user input. If user input is not sanitized correctly, it could contain code that runs along with server code in a clients browser.XSS usually affects victims web browser on the client-side where as SQL injection related web vulnerability is involved with server side.

There exists different methodology and a tool to detect vulnerabilities and attacks in web applications. The white-box testing is one of the main technologies, the testing object of which is the source code of web applications. The black-box testing is another detection technology for which the source codes of application and any knowledge of the test object's internal structure are not required.

In this work focus on different kinds of SQL injection and XSS attacks and approaches for their detection. In fact, SQL Injection and XSS are categorized as the top-10 2013 Web application vulnerabilities experienced by Web applications according to OWASP[6] (Open Web Application Security Project) project. In this survey Section II describes the basic characteristics of web application. Section III describes overview of SQL injection, then comparison analysis .Section V describes XSS, finally reach a conclusion.

II. CHARACTERISTICS OF WEB APPLICATIONS

Web application is a client-server application that is executed over the Web platform [7].A web application is any application that uses a web browser as a client. It is an integral part of today's Web system that enables dynamic information and service delivery.

2.1 Working of Web Application

Web application enables the dynamic information and service delivery. Figure 1 shows the web application that includes client side and server side components. The client side components include static HTML pages with scripting languages are executed within the web browser. Server side are responsible for processing request by web server using dynamic HTML pages through execution part i.e. Java Servlet and gives the response to the client request. HTTP is a stateless protocol. Separate mechanism is used to maintain session state in web application. Web application is a gateway of database that holds a critical vulnerability application and asserts. Most web application store the information in databases or in file system bases.

The development of web applications also faces challenges that are not ordinarily encountered in the development of traditional applications. First, in the open web environment, user inputs are potentially dangerous and can never be trusted .Input validation is an important part of a web application to identify and sanitize un trusted user inputs .Although input validation is an essential function required by all types of applications, its implementation in web applications is much more challenging due to the unique features of web development technologies. Second, the communication between the client and the web application is carried out through the stateless HTTP protocol. As a result, multiple inputs from the same user will appear to be independent users to the web application. The web application has to employ session management in order to correlate the web requests from the same user into a web session. The computing results from the client side can never be trusted ,additional validation of these results is required on the server side. These unique features significantly complicate the logic implementation of a web application, which needs to enforce the applications control flow between the client and the server and across different modules.

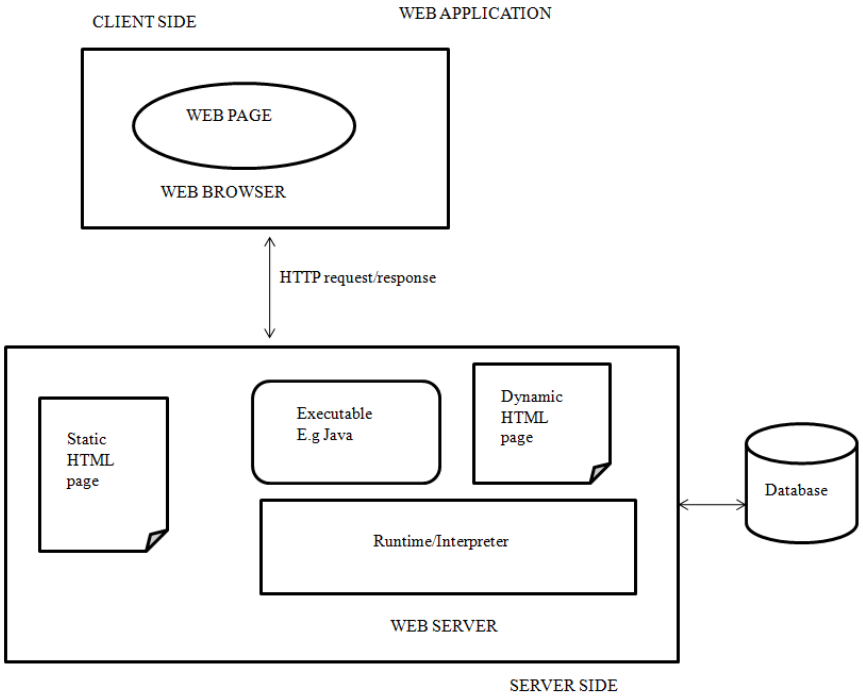

Fig. 1. Web Application

III. OVERVIEW OF SQL INJECTION

The SQL(structured query language) is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS).SQL Injection is a code injection attacks commonly used for attacking websites in which an attacker injects some SQL codes in place of the original codes to get access the database.

The main impact of SQL injection attack are confidentiality ,integrity and authentication. Confidentiality is the protection of personal information ,then loss of confidentiality is important problem.SQL databases contains sensitive information which could be viewed by unauthorized users by a successful SQL injection attack. Integrity is the successful SQL injection attack allows external source to make unauthorized modifications such as altering or even deleting information from target databases. Authentication is the SQL queries do not properly validate user names and passwords, which allows attacker to connect to the affected database or application as an authenticated user.

3.1 Classification of SQL Injection Attack

In this section cover different types of SQL Injection Attacks are described.

1) Tautology: This type of attack uses the authentication and access data through vulnerable input field using where clause .Then inject SQL query into conditional query statements which always evaluates to true.

2) Union Query: In this type of attack an unauthorized query is attached with the authorized query by using UNION clause. The UNION clause will cause to join malicious query with original query .Example is Select * from user where user=ravi union select _ from admin where id=3142 && pass = 2 = 2;.

3) Logically Incorrect queries: This type of attack collect essential information about the type and structure of the database by injecting illegal or logically incorrect SQL syntax .This will make the application return default error pages .Example is Aggregate functions applied on varchar or invalid data types.

4) Stored procedure: When a normal SQL statement (SELECT) is created as a stored procedure, an attacker can inject another stored procedure as a replacement for a normal stored procedure to performing privilege increase, create denial of service, or execute remote commands. Here is a common

form using a query delimiter(;)and the "SHUTDOWN" store procedure for this attack: normal SQL statement +"; SHUTDOWN; " <rest of injected query>.

5) Piggy-backed queries: In which attacker tries to add additional query to original query. By adding new query that piggy back the original query. As a result, the DBMS receives multiple SQL queries. The first is the normal query which is executed normally, while the subsequent ones are executed to satisfy the attack. Here is a common form using a query delimiter(;) for this attack. Example is Select _ from user where name= ravi and pass=1234;drop table user; The result of the example is losing the credential information of the accounts table because it would be dropped branch from database.

6) Blind Injection: This somewhat difficult type of attack, because during development process developer hides some error details which help the attacker to compromise with database.

7) Alternate Encodings: The main goal of the Alternate Encodings[9] is to avoid being identified by secure defensive coding and automated prevention mechanisms. Hence it helps the attackers to evade detection. It is usually combined SELECT * FROM users WHERE login = ” AND pass=’ ’;exec(char(Ox73687574646j776e)) In the above code the char() function and ASCII hexadecimal encoding have used. The char() function returns the actual character(s) of hexadecimal encoding of character(s). This encoded string is translated into the shutdown command by database when it is executed.

3.2 Approaches to Detect SQL Injection

There exists different approaches for detecting SQL injection either manually or automatically. Here discuss some some tool to detect SQL injection attack.

1) SQL Injection Detection System (SQL-IDS): Konstantin”s Kemalis and Theodor’s Tzouramanis, [1] suggest a specification based approach to detect SQL injection attack .It consists of three main components which interact with each other. The components are, Event Monitoring Module (EMM) is used for SQL query validation. Validated query are sent to Validity Check Module(VCM)module which check the correctness of the query with respect to the specifications that have been defined. If it is detects that the SQL query violates the specifications, then marks the query as a potential SQL

injection attack

.Disadvantages of SQL IDS are,

- It overloads operating system and degrades the performance of the web server machine when it receives a large number of requests.
- It only monitors java web based application.

2) AMNESIA: In [4], William G.J. proposes AMNESIA approach for tracing SQL input flow and generating attack input, JCrasher for generating test cases, and SQLInjectionGen for identifying hotspots. AMNESIA

consists of two parts; static analysis and runtime monitoring. Static analysis identifies hotspots and builds a model of SQL queries that could be generated by an application for each hotspot. At runtime or dynamic analysis checks the dynamically generated queries against the statically-built query model. If they do not match, the AMNESIA runtime monitor returns an error and prevents the SQL query from being executed. Otherwise, the query is sent to the database server.

The disadvantages of this approach is

- It involves no.of steps using different tools.
- It can't be used for web applications other than those built on JSP such as PHP or ASP.

3) SQLUnitGen: An approach has been proposed YongheeShin, Laurie Williams, Tao Xie[5] called SQLUnitGen: Test Case Generation for SQL Injection Detection. SQLUnitGen a tool based on static analysis, runtime and automated. It is used to find Input manipulation vulnerabilities. SQLUnitGen approach is described by three phases, such as Generate hotspot reaching test cases ,Generate attack test case and Execute test cases and generate test result summary. SQLUnitGen is based on two tools such as AMNESIA and JCrasher.

Disadvantage of SQLUnitGen is,

- Execution of initially generated test cases do not reach hotspots through every possible path due to the lack of proper input.

4) Vulnerability and Attack Injector Tool: An approach has been proposed Fonseca, J. and Vieira, M. and Madeira, H. is called Vulnerability and Attack Injector Tool(VAIT) [2].The VAIT tool is implemented for web applications. This tool was tested on web applications in two scenarios. The first to assess the effectiveness of the VAIT in generating a large number of realistic vulnerabilities for the offline assessment of security tools.The second to show how it can develop injected vulnerabilities to launch attacks, allowing the online assessment.There are mainly 4 stages in VAIT tool such as Preparation Stage,Vulnerability Injection Stage,AttackLoad Generation Stage,Attack stage.

IV. COMPARATIVE ANALYSIS FOR SQL INJECTION

4.1 Comparison of SQL Injection Detection Techniques With Respect to Attack Types

Detection techniques are techniques that detect attacks mostly at runtime. Table shows a chart of the schemes and their detection capabilities against various SQL injections attacks and summarize the results of this comparison.

Approaches\Attacks	Tautology	Logically Incorrect Queries	Union Query	Piggy-Backed Queries	Stored Procedure	Alternate Encoding
AMNESIA	YES	YES	YES	YES	NO	YES
SQL-IDS	YES	YES	YES	YES	YES	YES
SQLUnitGen	YES	YES	YES	YES	NO	YES

V. CROSS SITE SCRIPTING

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client side script into Web pages viewed by other users. The expression "Cross-Site Scripting" originally referred to the act of loading the attacked, third party web application from an unrelated attack site, in a manner that executes a fragment of JavaScript prepared by the attacker in
the security context of the targeted domain.

5.1. XSS Attack Types

There exists three distinct types of XSS attacks such as non persistent, persistent, and DOM-based attack[12].
1) Reflected or Non-persistent attacks: Non-Persistent cross-site scripting (XSS), also known as Reflected XSS .With Non-Persistent cross-site scripting, malicious code is executed by the victims browser, and the payload is not stored anywhere, instead, it is returned as part of the response HTML that the server sends. Therefore, the victim is being tricked into sending malicious code to the vulnerable web application ,which is then reflected back to the victims browser where the XSS payload executes. 2)Stored or Persistent attacks: The Persistent XSS condition is met when a website or web application stores user input, serves it back to other users when retrieving it at a later stage without validation before storage or before embedding stored content into HTML response pages.
3) DOM Based XSS: DOM Based XSS simply means a Cross-site scripting vulnerability that appears in the DOM(Document Object Model) instead of part of the HTML.

5.2 Approaches to Detect XSS

There are mainly three kinds to detect XSS in web application. They are: Static Analysis, Dynamic Analysis and Combination of Static and Dynamic Analysis.
1) Ardilla: Static analysis approach is to detecting XSS vulnerabilities that takes into account the semantics of input validation routines. One of the static analysis approach for detecting XSS is Ardilla. Ardilla has been used by Adam Kiezun[3] to detect both XSS and SQL injection attacks..Ardilla marks data coming from the user as potentially unsafe (tainted), tracksthe flow of tainted data in the application, and checks whether tainted data can reach sensitive sinks. It mainly consists of 4 components such as Input Generator, Executor/Taint Propagator, Attack Generator/Checker and Concrete+Symbolic Database.Input generator creates inputs for the PHP program under test. Executor/Taint Propagator runs the program on each input produced by the input generator and tracks which parts of the input flow into sensitive sinks. Attack Generator/ Checker takes a list of taint sets and creates candidate attacks by modifying the inputs in the taint sets.Then runs the program on the candidate attacks and compare it with original input. The concrete+symbolic database stores both concrete and symbolic values for each data record. Some disadvantages of this approach are it is tested on PHP based applications, If the original developer left the project it is very difficult to patch the vulnerabilities, Source code adjustment is needed ,Developer availability and learning is required. Some disadvantages also exist with this approach are:

- It is tested on PHP based applications.
- If the original developer left the project it is very difficult to patch the vulnerabilities.

- Source code adjustment is needed.
- Developer availability and learning is required.

2) SWAP: [10]P.wurzinger, C.Platzer, C.ludl, E.kirda and C.Kruegel developed a server side solution that detects and prevents cross site scripting attacks .It operates on a reverse proxy ,which relays all traffic between web server .In order to identify embedded java script content ,the proxy forwards each web response to a java detection component before sending it back to the client browser .All legitimate script calls in the web application are encoded in scriptIDs and so they are hidden from JavaScript detection component .If no scripts are found, the proxy decodes all script IDs ,and restoring all legitimate scripts, and delivers the response to the client. If the JavaScript detection component detects a script, SWAP stop delivering the response ,and notify the client as a wrong message .This approach requires only simple automated changes of original web application and is able to distinguish between legitimate
and malicious scripts.

Some disadvantages of this approach are:

- There is performance overhead.
- Also different web browsers may have different notation on valid and invalid JavaScript.
- It is capable of detecting only JavaScript based attacks.
- It cannot defend against other malicious content.

3) Noxes: [11]Engin Kirda developed Noxes which acts like a personal firewall that either allows or blocks connections to websites based on the filter rules. Noxes provides an additional layer of protection for existing personal firewall. Noxes operates as a web proxy that fetches HTTP re quests of the users browser. Then all web connections of the browser pass through Noxes and can either be blocked or allowed .Firewall rules can be created manually in which user enters the set of rules in a database, or user can create a rule interactively whenever a request for connection is made which does not match an existing rule or user can use a snapshot mode in which Noxes tracks and collects domains that have been visited by the browser.

Some disadvantages of Noxes are

- It requires client actions whenever a connection violates the filter rules.
- It only detects exploits that send user information to a third-party server, not other exploits such as those involving Web content manipulation.

Main objective is to implement a SQL injection and XSS vulnerability detection tool which can be used to identify and analyze SQL injection and XSS vulnerability of a web application. Automatic detection of SQL injection vulnerabilities relies on heuristics of how the target application behaves in response to specially crafted queries. Detecting new injection vulnerabilities with automated tools is the best option. In this research exists a tool that detect SQL injection and XSS separately. If can detect both SQLI and XSS using same automated tool has several advantages.

VI. CONCLUSION

Security is important factor we need to keep information as confidential. There are many security and privacy issues in many web applications .Web applications reach out to a larger, less trusted user base than legacy client server applications. The methodology consists of analyzing the web application and

generating a set of potential vulnerabilities. Each vulnerability is then injected and various attacks are mounted over each one. The success of each attack is automatically assessed and reported. There are many approaches and frameworks implemented in different web applications, security is still one of the major issues.

In this survey gone through a different cases of SQL injection and XSS attacks to web applications. First section cover SQL injection attacks type ,then discuss different approaches to detect SQL injection attacks.Also discuss their disadvantages .Each approaches have their own merits and demerits. Then handle a detailed comparative study of different approaches. Cross site scripting has been a major threat for web applications and its users from past few years. Lot of work has been done to handle XSS attacks such as client side approach ,server side approach, static and dynamic approach. Different approaches have their own advantages and disadvantages .Major problems are runtime overhead, not being able to cover all types of XSS attacks ,prone to human error etc.

REFERENCES

[1]. Konstantinos Kemalis and Theodores Tzouramanis, "SQL-IDS: a specification-based approach for SQL-injection detection", ACM, 2008.

[2]. Fonseca, J. and Vieira, M. and Madeira, "Evaluation of Web Security Mechanisms Using Vulnerability & Attack Injection", 2014.

[3]. Adam Kie˙zun and Philip J. Guo and Karthick Jayaraman and Michael D.Ernst, "Automatic creation of SQL injection and cross-site scripting attacks,", ICSE'09

[4]. William G.J. Halfond and Alessandro Orso," AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks,",2005.

[5]. Yonghee Shin and Laurie Williams and Tao Xie, "SQLUnitGen: Test Case Generation for SQL Injection Detection," 2006.

[6]. title=https://www.owasp.org/index.php/Top 10 2013-Top10.

[7]. XIAOWEI LI,YUAN XUE ,"A Survey on Server-side Approaches to Securing Web Applications", Nov 2013.

[8]. Shar, L.K. and Hee Beng Kuan Tan, "DefeatingSQL Injection", 2011.

[9]. Abhishek Kumar Baranwal," Approaches to detect SQL injection and XSS in web applications,",2012.

[10]. Wurzinger, P. and Platzer, C. and Ludl, C. and Kirda, E. and Kruegel,C," SWAP: Mitigating XSS attacks using a reverse proxy",2009.

[11]. Engin Kirda and Christopher Kruegel and Giovanni Vigna and Nenad Jovanovic," Noxes: A Client-Side Solution for Mitigating Cross-Site Scripting Attacks",2006.

[12]. Wikipedia, http://wikipedia.org.