ISSN:0975-9646



# A Survey on Web Application Attacks

Gurvinder Kaur Pannu

*University Institute of Engineering & Technology,*
*Kurukshetra University, India*

*Abstract:* - **Web applications today provide a universal way to access information. The internet has changed the lifestyle of people today. Web is being used by the organizations for providing their business. The so many uses of web applications led to security related challenges, as the number of applications over the internet increases so the number of attacks is also upsurge. Hence, security becomes one of the major concerns in web applications. Researchers have developed various approaches for the detection and prevention of various web applications including server side and client side approaches. This paper presents the various popular and common web application attacks found over the internet such as Injection attacks (SQLI), broken authentication and session management attacks and Cross-site scripting attacks (XSS), the various countermeasures taken and their respective limitations. It also proposes the future possibilities and feasible countermeasures against these attacks.**

*Keywords*: -- **Web Application Attacks, Web Security, Malicious Injection attacks, SQL Injection, Broken Authentication & Session Management Attacks, and Cross-Site Scripting Attacks.**

## I. INTRODUCTION

In today's world, Web applications have observed an overwhelming growth over the internet, which have been developed for various purposes. Now -a- days, almost each and everyone is in contact with the computer technology. Web applications openly present an interface through which clients can interact. For managing client data, service provider uses database for storage of client precise information. This database is used by the attackers by various means for obtaining client private data. Web applications are the obvious means for attackers to approach the elemental database, as they are usually vulnerable to attacks. As per Open Web Application Security Project (OWASP), code injection attack is the most familiar and fatal attacks among the top ten web application vulnerabilities followed by broken authentication and session management and cross-site scripting attacks [1]. Web applications get input from the end users by way of textboxes in the form of name, passwords, feedback etc. These input values are stored in database. Malevolent users insert SQL (Structured Query Language) query or script to do injection attacks. Web browsers execute these queries as the code, which behave abnormally as expected by the hacker. Authentication and session attacks appear when an attacker hijacks the current session or detour the authentication system. These attacks can provide the authorization of an authenticated user to the malevolent user, which can lead to serious consequences such as loss of confidentiality, integrity, authentication, authorization.

According to OWASP 2013 there are ten dominant categories specified for web application attacks in 2013 release

- A1-Injection
- A2-Broken authentication and Session management
- A3-Cross Site Scripting (XSS)
- A4-Insecure Direct Object References
- A5-Security Misconfiguration
- A6-Sensitive Data Exposure
- A7-Missing Function Level Access Control
- A8-Cross Site Request Forgery (CSRF)
- A9-Using Components with Known Vulnerabilities
- A10-Unvalidated Redirects and Forwards.

In actual world, it is very crucial to achieve entire security as some security flaws continually exist which can attack the application in distinct ways. In the next sections, the paper discusses the impact of top three-web application attacks SQL injection (SQLI), broken authentication and Session management and XSS.

## I SQL INECTION (SQLI)

SQLI is a code injection method that takes advantage of security vulnerability appearing in database tier of an application. It occurs when an attacker inserts SQL keywords as the segment of input. It executes because of incorrect or poor verification of input data. By way of SQLI, malicious user can acquire illegitimate right for database and can execute data manipulation operation on it. Broadly, injection attacks can be divided into three types. In *First Order Attack,* unions or sub SQL query is inserted to the existing statements. In *Second Order Attacks*, malignant code is forever stored in the database. In this the attacker attacks at the within application users, system heads by means of submissions, search lists, etc. In *Lateral Injection*, attacker can attack PL/SQL process that even does not get user input. It takes place when a variable whose data type or number is concatenated with text of SQL statement. When an input source is found by the malevolent user various SQLI kinds are used to accomplish attack of different kinds [2] .

### *1) TAUTOLOGY*

A tautology is a statement i.e. valid forever. This type of attack is used to bypass the verification system by using relational operators for matching operands to create a condition, which is always true. For instance

Select * from users where username = '' OR 1=1—' AND password = 'anything';

In users database, username and password are the input fields by which attack is carried out. In this ‘’OR 1=1--“will always send back true condition and usually known as tautology.

***2) PIGGY-BACKED QUERIES***

In this attack, an extra query is added to the original query and is executed as the part of initial query. For example

Select contact from info where login=’xyz’ AND pinnum =’123’; drop table info;

Here execution of these two queries will result in dropping of info table. So tables can be added or deleted using this type of attacks.

***3) LOGICALLY INCORRECT QUERIES***

The prime goal of this type of attack is to catch useful facts of the database by provoking error message, as number of columns of the database, database name, database version, label and type of each column. This info can be used further for misusing database.

***4) UNION QUERY***

It is also known as Statement injection attack. This attack is accomplished by imbedding union query so that database send back the dataset that is union of original query. For example

Select * from users where username =’’ UNION select * from admin—‘and password =’anything’”

Here query turn into union of both select queries. First query returns null and second returns all data from table admin.

***5) STORED PROCEDURE***

Stored procedures are the codes that are already present in the database and they are vulnerable as program code. They return true for authorized users and false for unauthorized ones. If the attacker input “,;SHUTDOWN;--“ for username then stored procedure creates following query

Select login from users where username=’’;SHUTDOWN;-- and password=’anything’.

The initial query is carried out and consequently the second query that is illegitimate is also run and lead to database shut down. Therefore, it is examined that stored procedures are as vulnerable as web application code.

***6) INFERENCE***

This type of attack is carried out to acquire information concerning the vulnerable parameter of the database. This sort of attack generates queries so that database or application reacts differently from as destined by the programmer. The two methods used in this class of attack are Blind Injection and Timing attacks. In blind injection attacker accomplish query that include result of true or false. If returned value is false then error is generated else application behaves properly. In timing attack, attacker run query in form of if-then statement and adopt WAITFOR keyword that causes database to lag its reply.

## II BROKEN AUTHENTICATION & SESSION MANAGEMENT

It happens whenever there is session hijacking or fake authentication. It involves administrartion of all sonditions of authentication and sessions which can influence the web servers, application servers, web application environment and can be source of wrong use of rights. For e.g. the attacker can alter message or can misuse data retrieved. Different types of cryptographic algorithms and session management tokens are used by developers for this type of attacks but still it is a dominant matter of concern [3].For managing verification and session issues all manner of points should beobserved in brain for instance :

***1) PASSWORD STRENGTH***

Passwords should have least possible length and proper use of alphabets, numbers and special keywords to prevent presumption.

***2) PASSWORD USE***

Number of login attempts should be restrained, as authorized users will not ever try to attempt repeatedly if he forgets the password. After a specified number of attempts user should be restrained to avoid attack.

***3) PASSWORD CHANGE CONTROL***

An individual password change method should be there to avert the security fault. The mechanism should inquire for old and new password but altering email address or modifying phone number should be prevented.

***4) PASSWORD STORAGE***

Password should be saved in with encrypted or hashed mechanism to defend them from disclosure.

***5) SESSION ID PROTECTION***

Session IDs should be long, difficult, having random numbers that cannot be simply figure out. It should be recreated repeatedly in an ongoing session to diminish session ID legitimacy. Session IDs must be changed while carrying out operations such as switching to SSL, validating users, or other major transitions. Session IDs selected by a visitor should never be accepted.

## III CROSS SITE SCRIPTING (XSS)

It is a ordinary vulnerability in web applications. XSS attack take place when HTML or JavaScript code is inserted into the database by way of inputs. If the particular inputs are not refined from server side, then serious consequences may happen such as recieving and communicating cookies, averting to third party. The sufferer of the XSS are the maximum common sources such as comments, feedback, search engines. They mark against valid users and system administrators. In XSS attacks, the mugger executes scripts to target websites for session hijacking, cookie stealing, and malicious redirection. XSS attacks are of two types [4]

***1) STORED OR PERSISTENT ATTACK***

In this the malicious code inseted by the attacker is stored permanently on the mark server. When a user requests for the stored information this script is executed in its context. Examples for raider favorite targets include web mail messages, forum, comment field, visitor log etc.

***2) REFLECTED OR NON PERSISTENT ATTACK***

Reflected attack take place off the web server, such as it exist in an error message or in a search result that comprises part of input or all of the input directed to the server as part of the query. They are also sent to victims via redirects & forwards, such as in an e-mail message, which revert the user to different link or to an un-trusted source or server.

## IV EXISTING APPROACHES FOR SQL INJECTION PREVENTION

***1)"Injection Attack Detection using the removal of SQL Query Attribute values " (Jeom-Goo Kim) (2011)***

Jeom-Goo Kim suggested [5] a clean efficient SQL Query elimination procedure, which uses combined style of static and dynamic analysis. This mechanism examines and inspects input by removing the attribute value of SQL queries. A function F is projected which has capability to remove the attribute present in SQL queries. The attribute value of static SQL query and SQL queries generated by the runtime environment will be removed. If after deletion of attribute, the query equals the fixed SQL query then there is no attack discovered. Although if some dissimilarity is present in query then attack is considered. The limitation of this application is that it has not been put into action until now.

***2)"Research of intelligent intrusion detection system based on web data mining technology" (Chai Wenguuang, Tan Chunhui, Duan Yuting ) (2011)***

Wenguuang and Yuting proposed [6] a smart agent technology using data mining. In this method stores data is stored in local database through data acquisition agent send the data to mining agent for pre processing examination. The exposure of vulnerability is done by alarm evaluation model, which generate an alarm for every detected attack. The intelligent intrusion detection with web data mining is found more efficient than traditional intrusion detection system. Improvement of data mining algorithms is needed for enhancement of proposed mechanism.

***3) "SQLStor: Blockage of Stored Procedure SQL Injection Attack Using Dynamic Query Structure Validation" (Sruthy Mamadhan, Manesh T, Varghese Paul) (2012)***

Mamadhan *et al.* in [7] established a new mechanism for prohibiting SQL injection attacks in JSP web applications. The idea is based on validation of structure of dynamic query before execution. The process consists of different steps i.e. at first create a non-malignant query from the original query by substituting user inputs to the query with gentle inputs. After that for authorization, check the structure of the *Benign Query*. Now stacked queries are calculated in both original SQL query and produced non-malignant query. If both counts differ, then SQL injection attack is present and can be prevented from execution beyond doing semantic inspection. If they are identical then syntax trees are constructed for both the queries and are compared. If they are comparable, the query is performed otherwise attack is reported. This approach prohibits various kinds of injection attacks.

***4) "SQLIMW: a new mechanism against SQL-Injection" (Gao Jiao, Chang-Ming XU, JING Maohua) (2012)***

Jiao et al. [8] introduce a middle-ware-based prevention mechanism (SQLIMW). In this method, a middleware is added in the system's backend to demolish the SQL Injection attacks. The SQLIMW implement more safety and adaptability by renewing encryption with HASH function and preserve username, password and private key of SQLIMW all at once by XOR operation. It discards the common way for encryption and uses the hash method. Because of both encryption and hashing clear data become the "confusion" and not readable and the execution speed of hash transformation is very fast than that of the DES and other cryptographic transformations. This method is not restricted to single layer of Web application; it can lie in any layer of Web application system swapping info with the database.

***5) "Blocking of SQL Injection Attacks by Comparing Static and Dynamic Queries" (Jaskanwal Minhas and Raman Kumar) (2013)***

Minhas and Kumar suggested a straightforward, effective and efficient method [9] to disclose SQLI attacks and describe and discovered a new attack other than existing SQLI attacks named white space manipulation attack. In this proposal, static and dynamic analysis is combined. After discarding attribute values, it compares static and dynamic SQL queries. The incoming queries are compared character by character with static queries having same number of tokens to lessen response time. After removing attribute values, SQL queries become self-sufficient so this approach can be applied to any database.

## V EXISTING APPROACHES FOR BROKEN AUTHENTICATION & SESSION MANAGEMENT PREVENTION

***1) "Root cause analysis of session management and broken authentication vulnerabilities" (D.Huluka and O.Popov) (2012)***

Huluka and Popov [10] have explored the root cause behind the session management and broken authentication vulnerabilities. As per them, there are numerous ways to exploit vulnerabilities. Therefore, it is very important to have a deep and clear understanding of problem. Therefore, they analyzed the root causes behind these vulnerabilities and presented how it can be used to improve the security aspects of web applications. They identified 11 root causes of session management and nine root causes behind broken authentication vulnerabilities. Identification of these root causes led to efficient solutions that can limit repetition of these attacks on web applications.

***2) "Automatic Detection Of Session Fixation Vulnerabilities in Web Applications"(Yusuki Takamastu, Yuji Kosuga, Kenji Kono) (2012).***

In [11] Takamastu *et al.* introduced a method to examine session fixation attack. Session fixation is an attack in which attacker compel visitor to use session ID send by him and can use application as a guest. In this method, they created a structure that works in three steps. First step is Packet Capturing catch all the packets notice the diversity in session-IDs. The structure exists between the user browser and the web application server. In second step, initial inspection is done for examining the vulnerability for session ID. The third step is Attack Simulation in which system initiate attack simulator that automatically generates the identical environment as that of real attacker and a victim. At this time virtual attacker approach and login to web application with Session ID (SID) that attacker get. The virtual attacker confirms if he can login with the obtained SID or not. The reply is analyzed for keywords. For example, "welcome victim" if such type of keyword is

acquired then web application is treated as vulnerable. The flaw of this method is that it is hard to handle such kind of virtual environment for real world application.

## VI EXISTING APPROACHES FOR CROSS-SITE SCRIPTING PREVENTION

***1) "CIDT : Detection of malicious code injection attacks on web application" (Atul S. Choudhary and M.L Dhore) (2012)***

Choudhary & Dhore [12] presented a Code Injection Detection Tool based on a Proxy Agent that organize the request as scripted request, or query based request. There are two components: Query Detector and Script Detector. The HTTP request approaching from client side is first forward to the CIDT within which the request is passed to both components one by one. Firstly, the Query detector verifies the request and the query are dropped if any invalid character is found. Only the authentic requests are passed to the next module, the Script detector that also refine the request for invalid tags and encodes it before sending it to the web server. The limitation of this approach is it requires more time to reply, that is the delay time is more.

***2)" Cross Site Scripting Attacks Detection Algorithm Based on the Appearance Position of Characters" (Takeshi Matsuda, Daiki Koizumi) (2012).***

Matsuda & Koizumi [13] presented a detection method against cross-site scripting attacks by drawing out an attack feature of XSS by seeing the appearance position and frequency of symbols. It study the attack features from given attack specimens. There are three modules in this approach.. The first one is the classification module in which sample of 32 characters is collected based upon the characters that frequently occurs in attacks. The second module will calculate the important degree of characters. The final module will detect the attack-by-attack feature value & threshold that is taken as 15 for the proper detection of the attack in proposed approach. The main disadvantage is it is calculation based and it does not tackle the new attacks effectively.

***3)" Developing a Security Model to Protect Websites from Cross-site Scripting Attacks Using Zend Framework Application", (Yousra Faisal Gad Mahgoup Elhakeem, Bazara I. A. Barry) (2013)***

Elhakeem & Barry [14] bring up the issues related to XSS attacks and providing a simple and useful security model to prevent websites from such attacks with the help of ZEND framework. The security model is based on a chain of levels and is built using a combination of tools. It is divided into four levels as: Security Awareness, Server Security, Client Security, & Design Guidelines. The framework defined by them is Zend Framework (ZF), which is an open source framework for developing web applications and services with PHP. This loosely coupled architecture allows developers to use components individually and offers a robust Model View Controller (MVC) implementation. The MVC paradigm is a way of breaking an application, or even just a piece of an application's interface, into three parts which are: Model: The model part of the application is the part that is concerned with the specifics of the data to be displayed, View: The view consists of bits of the application that are concerned with the display to the user, & Controller: The controller ties together the specifics of the model and the view to ensure that the correct data is displayed on the page. It accepts input from the user and instructs the model and view port to perform actions based on that input. It requires lot of tools to be combined so compatibility issues are there.

## VII FUTURE WORK

To provide better services over the internet it is important to have security over the web applications. The main aim of application developer is to provide their business over the web but the web applications may have various types of vulnerabilities and prone to various dangerous attacks that can harm an organization data and reputation. My future work will incorporate the web security tool that can detect and prevent maximum number of web application attacks. The proposed tool will mainly detect and prevent the attacks considered in this paper with minimum response delay and with zero false negatives and positives.

## VII CONCLUSION

SQLI, Broken authentication and session management, and XSS are considered as the most dangerous and common web application attacks found over the internet. In this paper, we have reviewed many techniques against that suffer from various types of limitations such as:

- Built-in limitations
- Partial implementations
- Complicated framework
- Developer's ability
- Run-time overhead
- False positives and false negatives
- No secure channel between the web server and web browser
- Response delay
- Additional infrastructure
- Cost of deployment

Many industries are employing web services for their benefits on the World Wide Web but for relieving themselves from the additional cost, they do not go for the security of the websites they created. Eventually it harms the users and company too. With the expansion of web applications, it is need of the hour to have an inclusive and logical framework for the prevention of various web application attacks so that better services can be provided to end users.

## REFERENCES

[1] https://www.owasp.org/index.php/Top_10_2013-Top_10
[2] Nilesh Kochre, Satish Chalukar, Santosh Kakde, "Survey On SQL Injection Attacks And Their Countermeasures ", International Journal Of Computational Engineering And Management, Vol -14, October 2011
[3] https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management
[4] Hossain Shaihriar and Mahammad Zulkernine, "$S^2XS^2$: A Server Side Approach To Automatically Detect XSS Attacks", Ninth International Conference on Dependable, Automatic Secure Computing, IEEE, 2011 PP.7-17

[5] Jeom-Goo Kim, "Injection Attack Detection Using Removal of SQL Query Attribute Values ", IEEE 2011
[6] Chai Wenguuang, Tan Chunhui, Duan Yuting, "Research Of Intelligent Intrusion Detection System Based On Web Data Mining Technology", IEEE 4th International Conference On Business Intelligence And Financial Engg. 2011, PP. 14-17
[7] Sruthy Mamadhan, Manesh T, Varghese Paul, "SQLStor: Blockage of Stored Procedure SQL Injection Attack Using Dynamic Query Structure Validation" 12th International Conference on Intelligent Systems Design and Applications (ISDA), IEEE, Nov. 2012, PP. 240-245
[8] Gao Jiao, Chang-Ming XU, JING Maohua "SQLIMW: a new mechanism against SQL-Injection" in Proc. Of 2012 International Conference on Computer Science and Service System, 2012, PP. 1178-1180
[9] Jaskanwal Minhas, Raman Kumar, "Blocking of SQL Injection Attacks by Comparing Static and Dynamic Queries" in International Journal Computer Network and Information Security, vol.2, 2013 PP.1-9
[10] D.Huluka, O.Popov, "Root cause analysis of session management and broken authentication vulnerabilities", IEEE World Congress on Internet Security, 2012, PP. 82-86
[11] Yusuki Takamastu, Yuji Kosuga, Kenji Kono, "Automatic Detection Of Session Fixation Vulnerabilities ", 2012 Tenth Annual International Conference on Privacy, Security and Trust IEEE, PP-112-119
[12] Atul S. Choudhary and M.L Dhore, "CIDT: Detection Of Malicious Code Injection Attacks On Web Application", International Journal Of Computing Applications Volume-52-N0.2, August 2012, PP. 19-25
[13] Takeshi Matsuda, Daiki Koizumi, "Cross Site Scripting Attacks Detection Algorithm Based on the Appearance Position of Characters", 5th International Conference on Communications, Computers and Applications, IEEE, October 2012, PP. 65-70
[14] Yousra Faisal Gad Mahgoup Elhakeem , Bazara I. A. Barry," Developing a Security Model to Protect Websites from Cross-site Scripting Attacks Using Zend Framework Application", International Conference on Computing, Electrical and Electronics Engineering (ICCEEE), August 2013, PP. 624-629