# Cyber Shelter

**Viraj Narkar, Brijraj Menor, Sujal Dighe, Raj Jogi, Pranali Pawar**

Student, Student, Student, Student, Faculty of Shah And Anchor Kutchhi Engineering College ,Chembur
CYBER SECURITY Shah And Anchor Kutchhi Engineering College , Mumbai ,India

**Abstract** - This project aims to simplify the process of identifying and defending against cyberattacks by teaching users how to prevent sneaky scripts and SQL attacks, two major web problems. It provides tools such as routine web searches, finding people on LinkedIn, pinging websites, gathering information about websites, and understanding the meaning of various port numbers. Vulnerabilities are system defects or weaknesses that can result in a security breach. Attackers often exploit these weaknesses to launch malware attacks, steal confidential information, circumvent website restrictions, hijack sessions, deface websites, and cause denial of service attacks (DoS). There are two types of XSS attacks: persistent XSS, where malicious code is submitted to remain active or be stored, and non-persistent XSS, where user input is returned without encryption to prevent data from being rendered into the browser. The identification and mitigation of vulnerabilities in web applications and their sensitive data is a persistent challenge in cybersecurity. Cyberattacks often exploit security flaws like SQL injection, Cross-Site Scripting (XSS), data collection, and data reconnaissance, endangering the availability, integrity, and confidentiality of web-based systems. The primary objective of this assessment is to evaluate the security posture of a website and identify vulnerabilities that could be exploited by malicious actors.

**Index Terms** - Cyberattacks, Sneaky scripts, SQL attacks, Web security, Tools, Routine web searches, LinkedIn, Pinging websites, Gathering information, Port numbers, Vulnerabilities, System defects.

## I. INTRODUCTION (HEADING 1)

1.1 Overview:

This concept is like a well-organized toolkit for those who want to secure their digital assets. It teaches you how to prevent sneaky scripts and SQL attacks, two major web problems, and how to keep your websites safe from them. In addition to theory, this project provides you with helpful tools such as:

- Routine web searches (Google search)
- Routine web searches (Google search)
- Finding people on LinkedIn
- Pinging websites
- Gathering information about websites (domain search)
- Comprehending the meaning of various port numbers
- Routine web searches (Google search)
- Finding people on LinkedIn
- Pinging websites
- Gathering information about websites (domain search)
- Comprehending the meaning of various port numbers

The main objective of this project is to simplify the process of identifying and defending against cyberattacks.

1.2 Types of vulnerabilities:

System defects or weaknesses that could result in a security breach are known as vulnerabilities. After identifying a weakness, or application vulnerability, and figuring out how to get access to it, the An attacker might utilize the program to their advantage. Sensitivity. Consequently, a risk to the privacy, accuracy, or The resources that an application has available is elevated. Attackers usually depend on particular instruments or techniques find weaknesses in applications and compromise the submittal.

1.2.1 Cross-site Scripting(XSS):

Since the 1990s, there have been rumors and reports of cross-site scripting, or XSS, vulnerabilities. According to the OWASP TOP 10 web application vulnerabilities for 2010 and 2013, XSS was ranked as the third most vulnerable. A type of security flaw known as cross-site scripting (XSS) allows an attacker to insert client-side scripts into web pages that are seen by other users. At the client side, the injected code is run.

The Same Origin Policy (SOP) is frequently circumvented by the attacker by using cross-site scripting

vulnerabilities.

Vulnerabilities can be exploited by an attacker to launch malware attacks, steal confidential information and identity, circumvent website restrictions, hijack sessions, deface websites, cause denial of service attacks (DoS), and more.

According to persistence capability, there are two types of XSS attacks:

1) Persistent XSS: An attacker submits malicious code that causes an XSS attack to remain active or be stored. stored by the server in a message board, a database, comment section, visitor log, etc. Thus, a victim can obtain the web application's stored data without that information that is secured for browser rendering [6].
2) Non-Persistent XSS: This type of attack occurs when user input is immediately returned by a web application in the form of an error message, search result, or other response that contains all or part of the user's input supplied as part of the request, without the data being encrypted to prevent it from being rendered into the browser and from being permanently stored [6].

This is a common vulnerability that shows up in search boxes. When an attacker uses a non-persistent cross-site scripting attack, they fool their target victims into clicking on a specially designed URL.

The browser will transmit the injected code to the server when the user clicks the link, and the server will then reflect the attack back to the victim's browser, allowing the browser to execute the code.

1.2.2 Sql Injection (SQLi):

Through the use of a web page's input into a SQL statement, users can insert SQL commands using a technique known as SQL injection. An inserted SQL command modifies a SQL statement and jeopardizes a web application's security [1]. SQL Injection is a type of code injection where SQL statements are put into an entry field to attack data-driven systems. SQL Injection takes advantage of a software security flaw in an application. An attacker can read and alter sensitive data, perform database administration tasks, retrieve file content from the DBMS file system, and more by using SQL injection exploitation [6].

Here are some common SQL injection types:

1) Classic SQL Injection:In classic SQL injection, an attacker injects malicious SQL code directly into user-input fields. The attacker might append SQL commands to input data, such as login forms or search boxes, to manipulate the database.
2) Blind SQL Injection (Boolean-based): Blind SQL injection doesn't display the database's data directly but relies on true/false responses from the server. Attackers inject SQL code and infer information based on whether the application's response is true or false. This technique is often used to extract data bit by bit.
3) Blind SQL Injection (Time-based): In this type of attack, an attacker injects SQL queries that cause the database to delay its response if a condition is met. By observing the time it takes for the application to respond, attackers can infer whether the injected condition is true or false.
4) Out-of-Band SQL Injection: Unlike traditional SQL injection, this attack doesn't rely on the application's response but instead sends data through a different channel, like DNS requests or HTTP requests, to extract information from the database.
5) Union-Based SQL Injection: Union-based SQL injection occurs when an attacker leverages the SQL UNION operator to combine the results of the original query with results from their own query. This can be used to extract data from the database.
6) Time-Based Blind SQL Injection: Attackers manipulate the SQL query to delay the application's response, effectively creating a time-based blind SQL injection attack. By measuring the time it takes for the application to respond, they can infer whether a condition is true or false.
7) Second-Order SQL Injection: In this type of attack, the initial injection occurs in one part of the application, but the malicious payload is not immediately executed. Instead, it's stored for later use, and when it's used in a different context, it can lead to an SQL injection vulnerability.
8) Error-Based SQL Injection: Attackers inject SQL code that intentionally causes errors in the application. The error messages generated by the application can reveal valuable information about the database structure, which can then be used in subsequent attacks.
9) Double SQL Injection: In this type of attack, an attacker injects malicious SQL code that targets multiple parameters or multiple vulnerable points within the application.
10) Time-Based Blind SQL Injection: Attackers manipulate the SQL query to delay the application's response, effectively creating a time-based blind SQL injection attack. By measuring the time it takes for the application to respond, they can infer whether a condition is true or false.

11) Stored (Persistent) SQL Injection: In stored SQL injection, the malicious payload is stored in the application's database, typically in a comment or user profile. When other users access this stored data, the payload is executed.
12) Reflected (Non-Persistent) SQL Injection: In reflected SQL injection, the malicious payload is embedded in a URL or form input, and it's not stored in the application's database. It's usually delivered via a link or tricking a user into clicking it.
13) DOM-Based SQL Injection: This type of injection occurs in client-side code (JavaScript) that manipulates the Document Object Model (DOM). Attackers can inject malicious code into the client-side script, causing SQL queries to be executed on the server.
14) It's crucial for developers to implement security measures such as input validation, prepared statements, and parameterized queries to prevent SQL injection attacks. Regular security testing and code review can help identify and mitigate these vulnerabilities.

1.3 Problem statement:

The identification and mitigation of vulnerabilities that expose web applications and the sensitive data they manage to potential exploitation is a persistent and evolving challenge in the cybersecurity landscape as we rely more and more on these applications for a variety of services and functionalities. Cyberattacks now frequently use security flaws like SQL injection, Cross-Site Scripting (XSS), data collecting, and data reconnaissance as entry points, endangering the availability, integrity, and confidentiality of web-based systems.

1.4 Objectives

The primary objective of this assessment is to evaluate the security posture of [website name or URL] and identify vulnerabilities that could be exploited by malicious actors. The assessment encompasses a variety of security testing techniques, including Cross-Site Scripting (XSS), SQL injection, data collection, and data reconnaissance. These techniques are commonly used by attackers to compromise web applications and the sensitive data they manage.

1.5 Goals

The primary goal of this security assessment is to identify, evaluate, and mitigate vulnerabilities, such as Cross-Site Scripting (XSS) and SQL injection, within [website name or URL], while also assessing data collection and reconnaissance activities, in order to enhance the website's security, protect sensitive data, and promote proactive security measures to maintain user trust.

1.6 Existing System:

1. Attack via Email, Stealing user‟s cookies,
2. Sending an unauthorized request and XSS attack in comment field are some ways to attack a website using Cross Side Scripting.
3. A server side XSS Attack can ruin or steal important information from all users of thewebsite.
4. There is a heavy server load on the system
5. because of lightweight virtualization and a robust mechanism is needed for detection and prevention of SQL injection and XSS attacks. So the researchers have implemented a system that reduces the server load and implement faster mapping patterns to remove attacks such as SQL injection and XSS in multitier web applications.

1.7 Proposed System:

1. N- Stalker, Acunetix, Paros, Hackbar and XSS ME are some tools to be used for detection of XSS vulnerabilities.
2. Also, escape all HTML, CSS and JavaScriptattributes before performing any new operation.
3. Based on the testing done and results obtained, our approach is efficient considering web server overhead. Using httperf it is found that, upto 90 requests per seconds our system performed similar compared to vanilla system. But above thatthe system started showing degradation after 110 requests per second. The overhead in the system defined in was 10.3% to 26.2%. But in the researchers approach they have found out that their system's over head is between 4.5% to 12.2%. Also the time required for execution in some system that maintains bulk data is more than their present approach. It is mostly because of the bulk data loading time the system consumes each and every time the request is made by the user. Theirapproach uses dedicated web server approach with basic format of SQL query comparison i.e. skeleton comparison that reduces time requirement for execution.Considering the other overheads on

the system in a network, our approach reducestime requirement for the whole system to be completed by nearly 52% on an average.

## II. LITERATURE SURVEY

Web application security is a critical concern in today's digital landscape, given the increasing dependence on web-based services and the ever-evolving threat landscape. To provide context and insights into the security assessment of [3], this literature review explores key topics related to web application security, common vulnerabilities like Cross-Site Scripting (XSS) and SQL injection, and best practices for safeguarding web applications.

The field of web application security has gained prominence due to the growing number of online services. Web applications are susceptible to various security threats, making it essential to implement robust security measures. According to McGraw and Viega (2002), security needs to be an integral part of the software development life cycle (SDLC). This concept, known as "Software Security Assurance," emphasizes proactive security measures, secure coding practices, and regular security testing throughout the application's development stages.

Cross-Site Scripting (XSS) remains a prominent vulnerability in web applications. XSS attacks allow malicious actors to inject client-side scripts into web pages viewed by other users. This vulnerability poses a substantial threat to user data and the integrity of web applications. The Open Web Application Security Project (OWASP) has consistently ranked XSS among the top web application vulnerabilities (OWASP, 2021).

Existing research has offered insights into different types of XSS attacks. The [6]provides an extensive overview of various XSS attack vectors and mitigation techniques. Their research highlights the need for input validation, output encoding, and secure coding practices to prevent XSS vulnerabilities.

SQL injection is another critical vulnerability in web applications. Attackers exploit this vulnerability by manipulating user inputs to insert malicious SQL commands into application queries, potentially compromising the database's security. Research by Aditya Rai; MD. Mazharul Islam Miraz; Deshbandhu Das; Harpreet Kaur; Swati (2021) [11] has emphasized the importance of robust input validation and output encoding to prevent

Various types of SQL injection attacks have been identified, including classic SQL injection, blind SQL injection, and time-based blind SQL injection. [11]Explores the intricacies of classic SQL injection and the potential consequences of such attacks, underlining the necessity for prepared statements and parameterized queries as countermeasures.

Best Practices for Web Application Security:

Several best practices are recommended to enhance web application security. The integration of input validation, output encoding, and prepared statements is consistently emphasized in the literature to mitigate XSS and SQL injection vulnerabilities. Additionally, the regular application of security assessments, code reviews, and security testing is vital to identify and remediate vulnerabilities in a proactive manner [2].

Conclusively, web application security is a multifaceted discipline that requires ongoing attention and the integration of security measures throughout the software development process. Addressing vulnerabilities like XSS and SQL injection is crucial to protect user data and maintain user trust in web applications.

| Serial no. | Paper Name | Author Name | Year published |
|---|---|---|---|
| 1. | Cross-Site Scripting (XSS) | Voo Teck En, Vinesha Selvarajah | 2021 |
| 2. | A Survey on SQL Injection Attack: Detection and Challenges | Zain Marashdeh, Khaled Suwais, Mohammad Alia | 2022 |
| 3. | SQL Injection: Classification and Prevention | Aditya Rai; MD. Mazharul Islam Miraz; Deshbandhu Das; Harpreet Kaur; Swati | 2021 |
| 4. | Detection and Prevention of Cross-site Scripting Attack with Combined Approaches | Hsing-Chung Chen, Aristophane Nshimiyimana, Cahya Damarjati, Pi-Hsien Chang | 2021 |

Table 2.1.1 Literature Review

## III. CONCLUSIONS

The project outlines a security assessment on a specific web application, highlighting the need for improved security measures due to the rising reliance on web applications and their associated vulnerabilities. It identifies critical issues, including XSS and SQL injection, and proposes a comprehensive system with various tools and strategies to enhance security, ultimately aiming to fortify the web application's trustworthiness and data protection in a dynamic digital environment.

## IV. REFERENCES

[1] Mehul Singh, Prabhishek Singh, Dr. Pramod Kumar "An Analytical Study on Cross-Site Scripting" , 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)

[2] Hsing-Chung Chen, Aristophane Nshimiyimana, Cahya Damarjati, Pi-Hsien Chang, "Detection and Prevention of Cross-site Scripting Attack with Combined Approaches", 2021 International Conference on Electronics, Information, and Communication (ICEIC)

[3] Voo Teck En, Vinesha Selvarajah, "Cross-Site Scripting (XSS)", 2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications (ICMNWC).

[4] Hanan Alsobhi, Reem Alshareef, "SQL Injection Countermeasures Methods", 2020 International Conference on Computing and Information Technology, University of Tabuk, Kingdom of Saudi Arabia.

[5] Nikita Joshi, Tejas Sheth, Varshil Shah, Jaya Gupta, Prof. Sofiya Mujawar , "A Detailed Evaluation of SQL Injection Attacks, Detection and Prevention Techniques", 2022 5th International Conference on Advances in Science and Technology (ICAST)

[6] Zain Marashdeh, Khaled Suwais, Mohammad Alia , "A Survey on SQL Injection Attack: Detection and Challenges", 2021 International Conference on Information Technology (ICIT)

[7] Alya Aiman Salsabila Arif, Rahmat Purwoko, Nurul Qomariasih, Hermawan Setiawan "Analysis of SQL Injection Attack Detection and Prevention on MySQL Database Using Input Categorization and Input Verifier" , 2022 IEEE 8th Informa tion Technology International Seminar (ITIS) Surabaya, Indonesia, October 19 – 21, 2022

[8] Kashish Gaur; Manoj Diwakar; Kaamya Gaur; Prabhishek Singh; Tanya Sachdeva; Neeraj Kumar Pandey , "SQL Injection Attacks and Prevention" , 2023 6th International Conference on Information Systems and Computer Networks (ISCON)

[9] Tao Zhang; Xi Guo , "Research on SQL Injection Vulnerabilities and Its Detection Methods" , 2020 4th Annual International Conference on Data Science and Business Analytics (ICDSBA)

[10] Eman Hosam; Hagar Hosny; Walaa Ashraf; Ahmed S. Kaseb, "SQL Injection Detection Using Machine Learning Techniques", 2021 8th International Conference on Soft Computing & Machine Intelligence (ISCMI)

[11] Aditya Rai; MD. Mazharul Islam Miraz; Deshbandhu Das; Harpreet Kaur; Swati , "SQL Injection: Classification and Prevention" , 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)

[12] Zar Chi Su Su Hlaing; Myo Khaing , "A Detection and Prevention Technique on SQL Injection Attacks", 2020 IEEE Conference on Computer Applications(ICCA)