INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 3 - V7I3-1768)
Available online at: https://www.ijariit.com

SQL injection and XSS

Devashish Agarwala
devashishagarwala18@gmail.com
RV College of Engineering, Bengaluru, Karnataka

Bhanu Pratap Yadav
bhanupy.is18@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

Vanishree K.
vanishreek@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

ABSTRACT

SQL Injection is a vulnerability that influences the Structured Query Language (SQL) [1] queries that an application passes to a back-end database. By exploiting what is given to the database, the attacker can leverage the syntax and capabilities of SQL itself and the power and flexibility of supporting database functionality and operating system functionality available to the database. Cross-Site Scripting (XSS) is a Code Injection attack executed on the client-side of an internet Application. The foremost common method of stealing cookies or hijacking sessions is to introduce JavaScript with a browser-supported html cryptography technique. [2] Cross-site scripting vulnerabilities ordinarily enable AN aggressor to masquerade as a victim user, to hold out any actions that the user is ready to perform, and to access any of the user's knowledge. If the victim user has privileged access at intervals the applying, then the aggressor could be able to gain the full management over all the application's practicality and knowledge. [3] The CWE/SANS Top 25 software errors place SQL injection and Cross-Site Scripting at the very top. Additionally, the Open Web Application Security Project (OWASP) lists Injection Flaws (SQL injection) as the most severe security vulnerability affecting Web applications in its Top 10 list. This highlights how common and very relevant in the domain of Web security. As developers, we should be aware of the consequences and prevention measures of these attacks. This report mentions a brief survey on the real-world attacks of SQL Injection and cross-site Scripting. Additionally, the paper highlights some defenses one can use to mitigate these attacks.

Keywords: *XSS, SQL*

1. INTRODUCTION

SQL Injection is a susceptibility which impacts queries sent by an application to database through Structured Query Language (SQL) [1]. By abusing what is offered toit, the attacker can take use of SQL's syntax and capabilities, as well as the strength and suppleness of supporting databaseand OS events accessible to database.The issue not only impact only Web apps; it affectsany code that gets information from a resource and uses itto construct dynamic SQL queries. It is in top ten Web app vulnerabilities faced by Web apps in2010, corresponding to "OWASP. (OWASP) stands for Open Web Application Security Project." [4]. The SQL language is used to access MySQL [4] Server, Oracle, MySQL, Sybase, and Informix servers. The majority of Web apps require database interaction. Web programming languages like ASP, C, N ET, Java, and PHP offer a variety of ways to connect to a database and interact through it. These vulnerabilities arise whena Web app designer fails to validate input from Web forms, cookies, input parameters, and other sources beforepassing it to SQL queries executing on a database attendant. An invader can change response such that it is read as encryption along with information. In that case, the invader can implement the code on database's backend. m Developersutilize a mixture of tactics to accomplish additional objectives, and each programming language provides many options for generating and managing SQL queries. Many websites providetutorials to help developers with common coding challenges, yet many of them encourage unsafe coding methods that resultin vulnerable code. Developers can write insecure apps that areprone to SQL injection if they don't understand the database system they're working with or if they don't understand and are aware of the security flaws in the code they're building. Nonetheless, the situation has improved over time. When you Google avoid SQL injection in your preferred language or technology, you'll often discover a multitude of wonderfuland useful websites that give logical information on how todo things right. Despite this, the situation has improved over time. When you Google avoid SQL injection in some language, you'll generally discover a variety of fantastic and useful websites that give logical information on how to do things right. Apple and Android give solid guide- lines on how to design safe code to developers switching to their platforms. These include some tips on how to avoid SQLinjection concerns, and the HTML5 organizations offer several warnings and security recommendations to early adopters. Cross-website scripting works by changing frequently used scripts, allowing affiliates to perform any action that the user desires, as well as access any of the user's information. The attacker is attempting to change and add vulnerable code tothe site, exposing visitors to malicious JavaScript. Cross-site scripting attacks are

- Reflected XSS —here, the mischievous protocol is injected into existing HTTP application.
- Deposited on XSS — The mischievous code is inserted into the website's information here.
- DOM-based XSS — The susceptibility resides within the consumer component code itself in this case. [5]

How will XSS work? Cross-site scripting works by redirecting users to a vulnerable website using malicious JavaScript. The attacker has totally compromised the victim's interaction with the application after the malevolent protocol has been implemented in victim's browser. The risk of XSS attacks is increasing as additional linked machines employ various Web Apps for different tasks. By exploiting XSS vulnerabilities in Web applications, hackers can steal victims' conference facts or other sensitive info. [6]

Contributions: In this paper we make the following contributions :

- We mention some real world attacks of SQL injections and Cross Site Scripting [7]– [8]
- Some common causes of SQL injections and defenses against it

2. REAL WORLD ATTACKS

A. SQL Injection

Because firms in many countries outside of the United States are not required by law to disclose severe security breaches, it is impossible to accurately estimate how many businesses are exposed to or affected by a SQL injection vulnerability. However, several publicly available resources aid in comprehending the scope of the SQL injection problem.The 2011 CWE (Prevalent Vulnerability Evaluation)/SANS Top 25 Highly Critical Software Flaws, for example, recognises the most common flaws that might result in severe software vulnerabilities. The top 25 submissions were chosen based on comments from more than 20 distinct organisations, which assessed for each vulnerability established on dominance, relevance, and exploitation potential. SQL injection is number one "on the CWE/SANS Top 25" list of software flaws. Furthermore, inits 2010 Top 10 ranking, the OWASP rates Injection Flaws as extremely critical defense weakness shaping Web apps. [9]

Some reported SQL attacks are as follows:

- In February 2002, Jeremiah Jacks found that Guess.com was susceptible to SQL injection and gotten entry to at minimum 200,000 customer credit card numbers [2].
- Jeremiah Jacks exploited a SQL Injection Flaw to hack PetCo.com in June 2003, gaining contact to 500,000 credit cardinfo. [10]
- MasterCard said on June 17, 2005, that a hacker had ob- tained 40 million credit card credentials using SQL injection. In 2005, it was the greatest known breach of its sort. [11]
- Guidance Software identified its hacked database server via a SQL injection issue in December 2005, exposing the financial details of 3800 clients.
- TJX, a bargain retailer in the United States, was targetedin December 2006, and millions of payment details were taken from TJX systems.
- "In August 2007, the United Nations" website was vandalised owing to a SQL injection vulnerability that exhibited anti-US sentiments.
- In 2008, the Asprox botnet expanded its botnet by ex- ploiting SQL injection weaknesses for large lead-by malwareinfections on about 500,000 Web pages.
- Using SQL injection attacks, a group of Romanian hack- ers reportedly got into the websites of Kaspersky, F-Secure, RBS WorldPay, BT.com, national- lottery.co.uk, and Bit- Defender in Feb 2009.
- On 17 August, 2009, the US Judiciary Branchaccused Albert Gonzalez and two unknown Russians withstealing 130 million credit card information. Heartland Payment Systems, 7-Eleven, and Hanford Brothers were among the companies affected.
- An anonymous organisation revealed in February 2011 that hbgaryfederal.com is susceptible to SQL injection bug inside its CMS.
- An attacker disclosed database ditches that includes login records and muddled keys of the Barracuda Networks Web site in April 2011.

B. Cross Site Scripting

"Cross-site scripting (XSS) may be akind of attack within which malicious scripts are injected into internet-sites and web applications" for the aim of running on the tip user's device. all through this method, unsanitized or unvalidated inputs (user-entered data) are alterd and shown as outputs. Some XSS attacks don't have a particular target; the offender merely exploits a vulnerability within the application or website, taking advantage of anyone unlucky enough to fall victim. however in several cases, XSS is performed in a directmethod, like email message. XSS attack will flip an online application or web site into a vector for delivering malicious scripts to the online browsers of unsuspecting victims. [12] Some reported XSS attacks are as follows:

- In September 2020, a critical vulnerability was found in the business communication app called slack.
- In March 2021, Microsoft team launched bug reporting program for many applications only for the desktop users.
- In Jan 2021, Cisco customer have update their systems after the networking giant patched dozens of high-impact security vulnerabilities the first patch cycle of 2021.
- A team of researchers spent three months hacking Ap- ple's web domain in October 2020, identifying 55 flaws and receiving a large reward.
- In May 2020, a researcher discover the DOM based XSS vulnerability in email service of google. [9]

3. CAUSES AND DEFENSES

A. Dynamic string building

Dynamic string construction allows developers to create SQL statements on the fly. At runtime, a dynamic SQL statement is built, and various situations yield distinct SQL statements. It is useful to pick which fields to retrieve from, SELECT declarations, as well as the other criteria forqueries and maybe alternative lists to query centered on various factors. This provides a susceptible code where code injection might occur. To achieve the same effect in a more safe manner,
utilise parameterized queries. These are SQL statements that include more inserted factors. We cannot inject code because arguments containing client input will not be treated as instructions since parameters are sup- plied to queries at run time. As a result, parameterized queries are more efficient and secure than dynamically generated SQL statements. The PHP code sample below displays SQL string statements that are dynamically generated based on user input.

```
$query = "SELECT * FROM TABLE1 WHERE
Value = '$ GET["input"]'";
```

The difficulty with vibrant SQL assertions, like the one above, are vulnerable to code injection if the input is not validated or encoded. Here is the SQL declaration built by above example:


```
SELECT * FROM TABLE1 WHERE Value = 'input'
```

B. Incorrectly Handled Escape Characters

Anything inside the quotations is interpreted as data in SQL, while anything outside the quotes is interpreted as data. Thus, by entering a single quotation in the URL or input form, we may immediately determine if the website is susceptible. The single-quote character can cause one of the following issues when entered into an application that utilises SQL queries. Alert: mysql fetch assoc( ) : dispute is not justifiable MySQLoutcome

```
Warning: mysql fetch assoc(): case isnot justifiable MySQL outcome
error in SQL syntax;
seek the tutoring manual for correct syntax
```

The issue occurs because the single-quote is con- sidered as a distinct character. An attacker utilises the quotation to "escape" the originator's objection, allow-ing him to build his own queries. Other special characters havedifferent meanings depending on the database; for example, blank spaces in Oracle have different meanings. [13] Asa result, knowledge with the distinct characteristics of the database technology utilised is crucial.

C. Incorrectly Handled Types

We do not include numeric data in quotes in SQL because we do not want it to be regarded as a string. Here's an exampleof code that uses user input to generate dynamic SQL queries. The script accepts a number character as the user's identityand shows user information.

```
$QUERY = "SELECT * FROM table1 WHEREValue = $
GET["IDENTIFIER"]";
$RES = mysql query ($QUERY);
// check number of rows returned
$COUNT = mysql num rows ($RES);
// print all records.
$i = 1;
while ($ROW= mysql fetch assoc($RES))
{ if ($i <= $COUNT)
{print $ROW[$i]. "<BR>";
$i++; }}
```

The LOAD FILE function in MySQL reads and returns the contents of a file as a string value. This file resides on the database server, and the complete path name is supplied as input. When the following sentence is submitted as input, an attacker will be able to access the /etc/passwd file for system credentials.

```
UNION ALL SELECT LOAD FILE('/etc/passwd')
UNION SELECT "<? system ($ REQUEST['cmd
']); ?>" INTO OUTFILE "/var/www/html/victim.
com/cmd.php"
```

Because the input is immediately processed as SQL syntax, the MYSQL user should be allowed file access for the afore-mentioned instructions. As a result, it is critical to validate the numeric input received by the web application since it opens the door for an attacker to execute instructions that might resultin the leak of system rights and the creation of a remote shell to the computer.

D. Incorrectly Handled Query Assembly

Dynamic SQL statements are sometimes required and cannot be avoided. Such applications may occur when the database is dynamic and the fields are populated afterwards.In a social networking site, for example, we would wish to query the active postings. When someone adds something on their page, these postings are placed. As a result, the fieldsare not prepopulated, and we may need to conduct dynamic SQL queries to perform them. We present an example of an employee-based information system. The database provides information about an employee's login and logout activities aswell as time sheets. The programme allows the user to specify which data should be returned as the current month's day rate.Even though the programme created the timesheets and day rate, the programme is still user-controlled in this situation. This is due to the application's support for GET requests. For the application-generated values, an attacker might provide histable and field data as follows:

```
$QUERY = "SELECT". $ GET["column1"]. "
FROM". $ GET["Employee Table"];
$RES = mysql query ($QUERY);

// count the rows returned.
$COUNT = mysql num rows ($RES);

// print rows returned
$i = 1;
while ($row = mysql fetch assoc($RES))
{ if ($i <= $COUNT)
{ print $row[$i]. "<BR>";
$i++; }}
```

To reveal the credentials of all users in the database, the attacker can modify the HTTP appeal by substituting the user meaning for table name and client, password, and private values for column names. This is how the attacker can accomplish it.

```
"http://www.victim.com/user details.php?tabl
e=users&column1=user&column2=
password&column3=Super priv"
```

As a result, it is critical to validate the query and examine the payload of a GET and post request.

E. Incorrectly Handled Errors

Inappropriate error management causes a slew of security issues for a website. When external error notes, like database discards and error insyntax, are shown, a typical issue emerges. These notifications contain implementation information that should not be displayed. This information provides an at-tacker with hints about site weaknesses. To extract data from databases, database error messages are employed. The error messages provide avenues for an attacker to alter and generate code in order to circumvent the provided developer's inquiry. The following basic sample application makes use ofa Microsoft SQL Server database server. The database error messages are shown on the website using this database verboseof error. When the user picks an identification from a list, the writing constructs and executes a SQL query dynamically:

```
private void Index Change (object user ,System.
Event Args e){
// A dynamic SQL statement to find arecord wi
th a given value

string QUERY;

QUERY = "SELECT * FROM TABLE1 "; QUERY +=
```



```
”WHERE VALUE =” + UserList.
SelectedItem.Value + ””;
// Establish connection with DB

OleDbConnection connection = new
Ole Db Connection (connectionString);
OleDbCommand command = new OleDbCommand(
QUERY, connection); OleDbDataReaderreader;
// Open database and read informationtry
connection.Open();
RES = cmd.ExecuteReader();RES.Read();
Results.Text = ”<b>” + RES[”LastName”];
Results.Text += ”,” + RES[”First Name”] +”</
b><br>”;
Results.Text += ”VALUE:” + RES[”VALUE”] +”<br
>”;
reader.Close();
catch (Exception error)
Results.Text = ”Error getting data. ”;Results
.Text += errorMessage;
 }
finally
connection.Close();
} }
```

An attacker may leverage the SQL error warnings regardingthe database if they manipulated the HTTP request to swap VALUE with another SQL query. As a result, we must sanitiseand deliver generic error messages that do not reveal sensitive information to an attacker.

F. Incorrectly Handled Multiple Submissions

Whitelisting is a procedure in which all characters are forbidden except those on the white list. It is suggested thatwe utilise a whitelist rather than a blacklist. Blacklisting is a strategy that states that all characters except those onthe blacklist should be permitted. The blacklist approach to input validation is compiling a list of all potential characters that may be used maliciously and prohibiting their input. Adequately maintaining such a list is a tough endeavour because there are several assault classes that be able to be characterizedin a variety of methods. The danger of utilizing directory of undesirable letters to examine an erroneouscharacter while compiling the list, or to neglect additional depictions of that terrible character. This isa common issue in big Web development projects, when not all developers adhere to the same code requirements. For example, during an application evaluation, it is not uncommonto discover that virtually all of the information submitted has been verified; yet, with patience, you can frequently unearthan effort which a designer has neglected to check. Developers, for example, assume that if a client reaches third type ina sequence of forms, they ought to have finished previous two forms. In fact, however, it is frequently easy to avoid the intended data movement by accessing supplies quickly via their URLs. This creates a gap and vulnerability, making the application susceptible to SQL injection. As a result, we should validate every page and form, and we shouldwhitelist our inputs.

G. Insecure Database Configuration

We should diminish gain access to attacker can get into the database by controlling the amount of data retrieved, the access to interconnected systems. Initially, we must secure the application code, but we should also look at the underlying database. Databases usually come with many default users. For instance, Microsoft SQL Server uses “sa,” and MYSQL uses ”root” as a system administrator account. These accounts have default and well-known passwords which need to bereset. We should ensure that the database service does notrun in privileged mode. As stated before, an attacker can use that even to get a shell into the system and read files with system credentials. Oracle on Windows, on the other hand, does not allow this and must be executed with SYSTEM rights.A database server enforces an entrance monitor paradigm, assigning different levels of access to different user accounts and permitting the implementation of collected processes,functions. Every database server offers by default functionality that is usually superfluous and might be abused “by an attacker (xp cmdshell, OPENROW SET, LOAD FILE, ActiveX, Java” assistance, etc.).

To meet the demands of their applications, application developers typically employ the built-in confidential accounts rather than building a user account. These strong accounts have the ability to do various activities on the database that are not required orrequired by the application. If an attacker leverages a “SQL injection vulnerability in an” app which links to database together with a confidential account, he be able to implement protocol by permissions of the specific account. Web application makers must operate together with database supervisors to build a “least-privilege architecture for the application’s database access.”

4. IMPLEMENTATION

A. SQL Injection Demonstration

We created a web application for SQL injection. Because we updated the /etc/hosts file to match our URL to our system’s local IP address, the URL to the application is only accessible from our system (127.0.0.1). To host the web sites, we utilised the Apache server on our PC. The configuration directives are stored in a configuration file entitled 000-default.config in the directory ”/etc/apache2/sites-available.” The Mysql databaseis set up with a database for employees and a super usercalled admin. Figure 1 shows a snapshot of the database.Employees can access and update their personal information in the database using the online application. The followingare the two primary responsibilities in this web application: Administrators have the ability to manage each individual employee’s profile information; individuals can read or change their own profile information. We will use the login page for this job. The login page is displayed in Figure 2 . It requests for a username and password. The web app logs in and validates users centered on their input. As an attacker, weget access to the online application without knowing any of the employees’ credentials. The code below demonstrates how users are authenticated in our application.

B. Cross-site Scripting Demonstration

“XSS (cross-site scripting)”: Hacker information is not deposited on website in this case. RXSS is only active on the target's side. Cross-site scripting is reflected in this situation. A hacker sends an input script to a website, which is then sent to the victim’s browser, where the hacker executes mischievous JavaScript code.

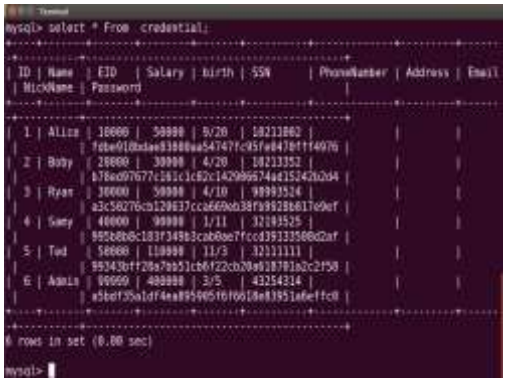

Fig. 1: The database created in our example application with the employee details and credentials along with a super user called "admin"

Fig. 2: Login page with the vulnerable php code which doesn't do input sanitization in the Username and Password fields. We demonstrate an exampleattack where we use # which is a special symbol for comment in php codeto bypass and login without credentials.

Fig. 3: Employee details from the database before the SQL attack. Observer that Alice's salary is 30,000

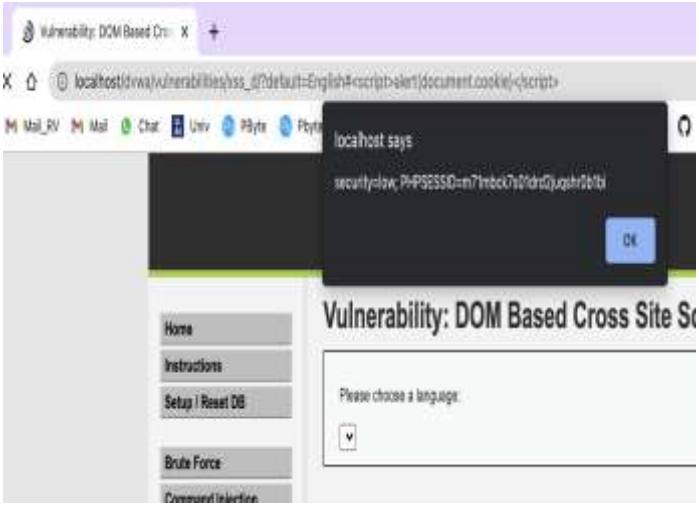

Fig. 4: SQL injection attack on the update form based web page. The page is making a dynamic SQL statement and the input is not sanitized. We exploit this vulnerability to update Alice's slary as shown in the figure.

Fig. 5: Reflected XSS . The output should be a normal text saying " Hello hacked by RXSS. We altered this to be as heading tag shown in the figure.

In this part, we'll show you how to set up a virtual environment for cross-site scripting.

Prerequisites:

- Xampp
- "DVWA (Damn vulnerable web application) [14]"
- "Browser as Firefox, explorer, Chrome e.t.c XSS (Cross-Site Scripting):SXSS Cross-site scripting" that is saved (XSS) In this situation, the hacker's malevolent malware has infiltrated the target machine as well as the internet server.when the attacker attempts to deliver malicious JavaScript code and run it on another user's machine "Stored XSS (Cross site scripting):SXSS Stored cross-site scripting (XSS) Here the hacker code" is intruded on target system and also the internet server. where invader try to send mischievous JavaScript code and try to execute it on alternative users system. [15]
- "DOM BASE XSS: Dom base (XSS) cross-site scriptingis a" cross-site scripting attack that uses the document object model as its foundation. That is, the website's answer does not

Fig. 6. DOM XSS . The malicious code is injected into the URL.

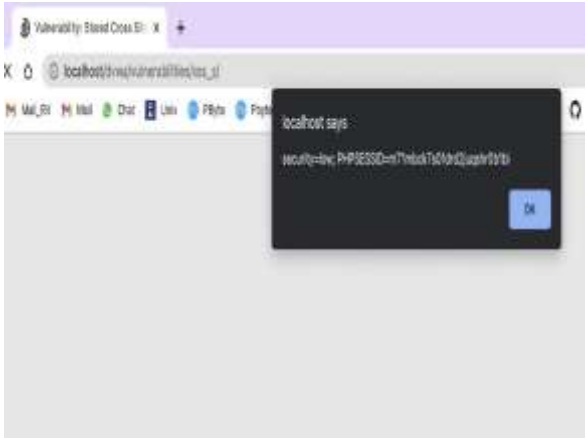

Fig. 7: Stored XSS . On page relode nothing should be shown, but here the stored value is being shown.

change; instead, the attacker injects the script into the URL and stores it, and when the user refreshes the page, the scriptis performed. [16]

5. CONCLUSION

SQL injection attack comprises of code injection where the SQL is added into application and then go on to the backend server for implementation. Typically it includes inclusion of code keen on considerations that concate-nated "with SQL commands." A modified SQL statement will operate with similar advantage (naturally high) as the element that killed the control.

SQL injection vulnerabilities are most common in Web applications. The developer fails to ensure the validation and encoding of input values from fields, cookies being passedto the SQL queries executed on the database. An invadercan monitor and manipulate the input forwarded to SQL query to implement code on backend. Devoid Of a good background on the systems and technologies like the database used and recognition of safety concerns, the code being created can create a considerable impact and produce insecure applications vulnerable to SQL Injection.

The XSS attack is among the recent internet website Application violence, although even, several appls have XSS susceptibilities due to the inappropriate execution of safety procedures in internet appl enhancement. In our analysis of 412 XSS connected documents from 2002 to 2019, plentyof analysis specialize in the sole client or server-side XSS results, however mixture results are successful in avoiding XSS strikes. In current times investigators accepting ML procedures to prevent XSS strikes, these methods are successful in detection undetermined outbreaks. [14]

6. REFERENCES

[1] C. J. Date, *A guide to the SQL Standard: a user's guide to the standard relational language SQL*. Addison-Wesley Longman Publishing Co., Inc., 1987.

[2] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna,"Cross site scripting prevention with dynamic data tainting and static analysis." in *NDSS*, vol. 2007, 2007, p. 12.

[3] A. Shrivastava, S. Choudhary, and A. Kumar, "Xss vulnerability assess- ment and prevention in web application," in *2016 2nd International Con-ference on Next Generation Computing Technologies (NGCT)*. IEEE, 2016, pp. 850–853.

[4] A. MySQL, "Mysql," 2001.

[5] D. A. Kindy and A.-S. K. Pathan, "A survey on sql injection: Vul- nerabilities, attacks, and prevention techniques," in *2011 IEEE 15th international symposium on consumer electronics (ISCE)*. IEEE, 2011, pp. 468–471.

[6] J. Clarke-Salt, *SQL injection attacks and defense*. Elsevier, 2009.

[7] D. Wichers, "Owasp top-10 2013," *OWASP Foundation, February*, 2013.

[8] P. Kumar and R. Pateriya, "A survey on sql injection attacks, detection and prevention techniques," in *2012 Third International Conference on Computing, Communication and Networking Technologies (ICC-CNT'12)*. IEEE, 2012, pp. 1–5.

[9] A. Neagos and S. Motogna, "Security analysis regarding cross-site scripting on internet explorer." in *BCI (Local)*. Citeseer, 2012, pp. 125–128.

[10] 2021. [Online]. Available: https://www.securityfocus.com/news/6194

[11] Feb 2006. [Online]. Available: https://www.ftc.gov/enforcement/cases- proceedings/052-3148/cardsystems-solutions-inc-solidus-networks-inc- dba-pay-touch

[12] J. Garcia-Alfaro and G. Navarro-Arribas, "A survey on cross-site script- ing attacks," *arXiv preprint arXiv:0905.4850*, 2009.

[13] G. Wassermann and Z. Su, "Static detection of cross-site scripting vulnerabilities," in *2008 ACM/IEEE 30th International Conference on Software Engineering*. IEEE, 2008, pp. 171–180.

[14] G. Kaur, "Study of cross-site scripting attacks and their countermea- sures," *International Journal of Computer Applications Technology and Research*, vol. 3, no. 10, pp. 604–609, 2014.

[15] S. Gupta and B. B. Gupta, "Cross-site scripting (xss) attacks and defensemechanisms: classification and state-of-the-art," *International Journal ofSystem Assurance Engineering and Management*, vol. 8, no. 1, pp. 512–530, 2017.

[16] 2021. [Online]. Available: https://www.securityfocus.com/news/346

