SQL Injection Attacks Prevention System Technology: Review

Article · July 2021

DOI: 10.9734/AJRCOS/2021/v10i330242

Fairoz Q Kareem
Duhok Polytechnic University
4 PUBLICATIONS **42** CITATIONS
SEE PROFILE

Siddeeq Yousif Ameen
Duhok Polytechnic University
95 PUBLICATIONS **397** CITATIONS
SEE PROFILE

Awder Ahmed
Sulaimani Polytechnic University
22 PUBLICATIONS **61** CITATIONS
SEE PROFILE

Azar Abid Salih
Duhok Polytechnic University
30 PUBLICATIONS **330** CITATIONS
SEE PROFILE

Asian Journal of Research in Computer Science

10(3): 13-32, 2021; Article no.AJRCOS.70376
ISSN: 2581-8260

SQL Injection Attacks Prevention System Technology: Review

Fairoz Q. Kareem[1*], Siddeeq Y. Ameen[1], Azar Abid Salih[1], Dindar Mikaeel Ahmed[1], Shakir Fattah Kak[1], Hajar Maseeh Yasin[1], Ibrahim Mahmood Ibrahim[1], Awder Mohammed Ahmed[2], Zryan Najat Rashid[2] and Naaman Omar[1]

[1]Duhok Polytechnic University, Duhok, Kurdistan Region, Iraq.
[2]Sulaimani Polytechnic University, Sulaimani, Kurdistan Region, Iraq.

Authors' contributions

This work was carried out in collaboration among all authors.. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/AJRCOS/2021/v10i330242
Editor(s):
(1) **Dr. Dariusz Jacek Jakóbczak,** Koszalin University of Technology, Poland.
Reviewers:
(1) Muralidharan J, KPR Institute of Engineering and Technology, India.
(2) Singaraju Suguna Mallika , CVR College of Engineering, India.
(3) G.S.N.Murthy, Aditya College of Engineering, India.
(4) Saroj Kumar Dash, BPUT,India.
Complete Peer review History: http://www.sdiarticle4.com/review-history/70376

Review Article

Received 01 May 2021
Accepted 06 July 2021
Published 06 July 2021

ABSTRACT

The vulnerabilities in most web applications enable hackers to gain access to confidential and private information. Structured query injection poses a significant threat to web applications and is one of the most common and widely used information theft mechanisms. Where hackers benefit from errors in the design of systems or existing gaps by not filtering the user's input for some special characters and symbols contained within the structural query sentences or the quality of the information is not checked, whether it is text or numerical, which causes unpredictability of the outcome of its implementation. In this paper, we review PHP techniques and other techniques for protecting SQL from the injection, methods for detecting SQL attacks, types of SQL injection, causes of SQL injection via getting and Post, and prevention technology for SQL vulnerabilities.

Keywords: SQL injection; PHP; database security.

**Corresponding author: E-mail: Fairoz.kareem@dpu.edu.krd;*

1. INTRODUCTION

Website applications play an essential part in daily life in today's technology-driven world. People use websites for various purposes, including internet shopping, banking, and chatting with friends. Often, websites use databases to store user data on the backend [1]. Since sensitive information, including passwords, credit card numbers, and social security numbers, is kept in such files, malicious hackers often attack them [2-4].

According to an analysis of numerous hacking events, when operating system security improves and security protection software and hardware solutions become more widespread, network attacks directly triggered by operating system vulnerabilities decrease year after year, while the usage rate of WEB application system vulnerabilities increases. Because of its easy syntax and high development performance, PHP has become the language of choice for developing all types of portal websites and Web application programs. LAMP architecture (Linux + Apache + MySQL + PHP) is a PHP language development environment with fast speed, high compatibility, free open source, and other benefits [5]. Based on the current traffic levels, about 70% of the LAMP architecture relies on the present-day network conditions [6]. Since the PHP early design process is over-efficient and transparent, some necessary security specifications in the programming language are not strictly restricted [7]. Adding delicate data is needed. It is counterproductive because the programmer lacks their security knowledge while the machine functions are jeopardized [8].

Some sections of the system face no constraints in their operational protection [9]. So many data are vulnerable, resulting in the fact that the system can present significant security risks [10]. Especially prominent among them are SQL injection vulnerabilities. For the first time since 1999, the SQL injection has been in people's minds and was ten years old. While a comprehensive preventive plan is now in place, its capacity cannot be underestimated [11]. One of the most dangerous loopholes is SQL injection [12]. OWASP Top 10 for 2010, OWASP Top 10 for 2013, and OWASP Top 10 for 2017, SQL injection was ranked as the top challenge to Web application systems three times [13,14].

SQL injection vulnerabilities compromise the protection of individual websites and the entire database infrastructure and network system that hosts related applications [15]. It is easily capable of causing large sections of web pages to hang, viruses to spread, privacy breaches, remote control of servers, and, in extreme cases, network paralysis [16]. Also, due to its simplicity, SQL injection vulnerabilities are often used as a stepping stone for network attacks, which have the objective of penetrating the target's network step by step [17]. While hackers may also use their expertise to infiltrate only an application or a single server, they want the ability to get full access to the network or get complete internal information [18].

One of the common ways for hackers to attempt databases is this SQL injection attack. More and more programmers use this mode to create apps in the development of the B/S mode application [19,20]. Given the unequal degree of programmers' experience, a significant percentage of programmers do not assess user input data's authenticity when developing code, which poses concerns for the application's security [21,22]. Users can submit a database query code and obtain data based on the program outcomes they wish to know [23]. One of the methods of database security assault is the SQL injection assault. The database security protection technology can successfully secure it [24,25].

It is not unusual nowadays to note media coverage of any serious violation of the cybersecurity of a significant firm [25,26]. Many of these infringements are attributable to software or system flaws [27]. Once a thorough study was carried out of these vulnerabilities, the development issues revealed a high number of these flaws [28,29]. More specifically, the vulnerabilities were caused by either developers or the design process. Injection attacks are a specific vulnerability triggered by developers or by an imperfect design process [30]. SQL injection attacks were mainly attributable to most of the cybersecurity violations committed by the organization. This kind of attack can harm a company or company [31,32]. These effects might include monetary loss, the disclosure of private corporate data, consumer exposure, a drop in stock, or a mix of the four [33,34]. In interactive online applications, SQL injection attacks are relatively frequent. They can not only be detected readily and reasonably easy to mitigate by SQL injection assaults [35,36].

A relatively common online vulnerability is Structured Query Language Injection Attack (SQLIA) [37,38]. To obtain data access or make unauthorized modifications to the data, the attacker adds malicious structured query language (SQLs) code to the input area of a web form [39,40]. Successful malicious SQL injection leads to significant cash losses, reputation losses, compliance and regulation violations in the targeted business [41,42]. Several research projects have been undertaken to identify and prevent SQL injections. However, a single advanced tool for identifying and mitigating SQL injection threats remains unusual [43,44].

SQL Injection may be defined as allowing hackers to use a web app to perform a malicious SQL query on the database server to access sensitive information or database [45,46]. This web-based vulnerability enables the attacker to spot the identity, delete the system's data, and modify the database records [47,48]. SQL injection's main effects include loss of Confidentiality, authentication as an attacker without providing the original user name and password to access the network successfully by manipulating the SQL command logically, loss of authorization as an attacker leaves complete system information, and lack of integrity as hacker gets access to the database [49,50].

To adapt to changing business requirements, information systems typically migrate to the Web; however, these technologies are sometimes vulnerable to an enormous variety of assaults with their security weaknesses [51,52]. According to Mitre Corporation, SQL Injection Threats (SQLIA) are among the most prevalent types of safety attacks these systems face [53,54].

The internet network has now become one of the primary everyday requirements of humans in the era of continuously changing and growing information technology [55,56]. Web applications such as online banking, web-based e-mails, instant communications networking are aimed at attackers [57,58]. Their main aim is to obtain critical user information and use it for their purposes. The SQL injective attacks are one way to target online apps and web-based information systems [59,60].

SQL Injection Assault is one of the most frequent and most damaging ways of a hacker attack. SQL injector attacks and prevention are essential and challenging subjects to teach in information system protection in our school [61,62]. SQLi-labs have different vulnerabilities to teaching help software [63]. The instructor may perform in-class SQL injection assaults via the aid of this program, helping pupils understand the SQL injection attack and preventive concept [64,65].

A primary security concern is posed by a SQL injection attack (SQLIA) in Web applications backed by a database [66]. This exploit allows attackers to easily access the application's underlying database and the potentially sensitive information contained in the databases [67,68]. A hacker can access database content by carefully crafted input, which otherwise cannot. Usually, this is done by modifying SQL statements used in online applications [69]. Researchers have investigated SQLIA detection and prevention thoroughly and created numerous approaches because of the safety of web applications [70,71].

As the internet is used to provide online services, web security risks are also drastically grown every day [72,73]. SQL injection is one of the most significant and dangerous vulnerabilities in online applications [74]. SQL injection attack occurred when a part of a malicious SQL query was inserted in the legal query statement via an invalidated user entry [75,76]. This will lead to the execution of such instructions and SQL injection [77]. The database management system's Confidentiality, integrity, and availability of information in the database have interfered with successful SQL injection attacks [78,79].

With the progress of the Web, the majority of people transact on the Web, for example, through data research, banking, shopping, management, surveillance and management of dam and commercial exchanges, etc [80]. Web apps have adapted to the daily life of many people. Web applications' dangers have extended to include enormous expansion [81]. At the moment, the more vulnerabilities are reduced every day, the greater the number of threats [82]. The SQLIA is one of the most significant hazards of online applications risks [83]. Structured queries are an injection attack [84]—the lack of SQL web-based injection attack validation flaws [85]. SQLIA is a malignancy that involves the abuse of data-driven applications through negated SQL declaration [86]. This vulnerability allows an attacker to communicate the applications' interaction with backend databases by complying with the designated input [87]. Therefore, access to the database can be obtained by the insertion, modification, or deletion without valid authorization of essential information [88].

In this paper, we discussed the SQL injection prevention methods in depth: in section 1, we introduced a general concept of SQL injection. In section 2, we explained in-depth the database security threats and SQL injection (attacks and types), in section 3, the Prevention technology for SQL vulnerabilities was discussed, in section 4, we explained the related work on SQL injection with PHP and different techniques, in section 5 we discussed the SQL injection prevention with PHP and we compared with other authors. Finally, the conclusion expounds and summarizes the attack principle and attack implementation SQL injection attack Principles and Preventive Techniques for PHP Sites.

2. DATABASE THREATS

Database security is described as a collection of measures, policies, and processes for ensuring data confidentiality, integrity, and availability and combating potential device attacks (threats) from both insiders and outsiders, both malicious and unintentional [89]. On the other hand, data security is described as using hardware or software to protect information from unwanted access, modification, or destruction [90]. In a database world, protection is achieved by defining risks and selecting appropriate policies and mechanisms, which contribute to what the security system is supposed to do and how the security system can meet the security objectives [91]. It also entails providing security system assurance, which relates to how well the security system satisfies the defense specifications and performs its functions [89].

A relational database is a set of data points with specified relationships that can be accessed easily [92]. The Structured Query Language (SQL) is a relational database's standard user and application program interface (API) (SQL).

2.1 Structured Query Language (SQL) Attacks

SQL (Structured Query Language) is a text-based language for interacting with database servers. SQL commands such as (INSERT, RETRIEVE, UPDATE, and DELETE) are utilized to operate on the Database. The programmer uses this command to manipulate data in the database server. SQL Injection is a strategy for injecting SQL commands into a web server that runs in a backend database by exploiting an invalidated input weakness.

According to the concept, a SQL Injection attack is hazardous because the attacker who successfully enters the server database will access the data already present in the database. Improper manipulation of data by an attacker can cause harm to the owner of an injected website. The leakage of data and information is fatal. Such data may be misused by irresponsible parties [93].

According to the definition, a SQL Injection attack is hazardous since an attacker who has successfully infiltrated the system database can change the data already present. An attacker's improper data modification can affect the owner of an injected website. Data and information leakage may be disastrous. Irresponsible persons may misuse this information.

2.2 SQL Injections Attacks

SQL injection is a type of code injection attack in which the user's data is placed into the SQL query, causing a portion of the user's input to be interpreted as SQL code [9]. This is a technique to use web pages as an input to insert SQL queries or orders. It happens when the data supplied by the user is not checked correctly and explicitly used in the SQL questionnaire. An attacker can directly access the database by using these vulnerabilities. There are two key SQL injection strategies, according to Sharma (2005). (Access through login page and access through URL). The first method is most easy under which login types are circumvented as passwords are used for user authentication. The attackers will do this by: 'or' state, 'getting' clause, various queries, and expanded stored procedure. An intruder may execute the second technique: manipulate the question string in a URL and use the statements 'SELECT' or UNION. This type of vulnerability indicated a serious.

Suppose the formula returns [1 = 1] or an empty row in the user's table, a user was detected. The first [‘] quote ends the string, and the characters [--] indicate the start of a SQL comment; anything after that is ignored. The database's interpretation of the problem is now a tautology that is still fulfilled. As a result, an attacker can get unlimited access to sensitive information on the server by bypassing all authentication modules. SQL injection can damage any database, regardless of software or web application. This attack can be used to steal sensitive information, circumvent authentication protocols, alter databases, and run arbitrary code. In some instances, the attacks were directed at the database server itself [94].

2.3 SQL Injection Attacks Can Take Several Types

2.3.1 properly filtered escape character

This is a kind of injection attack where the user is not bound to prevent them from writing something they do not want to write and is therefore passed on to a SQL statement. If this results in the end-user handling the SQL statements, this can only lead to complications for the user. A notable example of this issue is a code fragment that illustrates this vulnerability:

- Statement: (“SELECT * FROM users WHERE name =’” + 17username + “;”)
- The general aim of this code is to pull out a single user from its list, but if the user’s name is compromised, the action can go off in an unexpected direction.

2.3.2 Failure to handle sort

There are several attack patterns of this kind. If an attacker applies simple types in a nonconfirmed field, this attack may be made [95]. Until submitting it to the database (whether it is numeric or not). As another case:

Statement: (“SELECT * FROM data WHERE id = + a_variable +”;;)

The Author expects a variable to be a number connected to the “id” field, as can be seen by this sentence. But if the end-user selects a string, it circumvents the need for escapes. For instance, set a variable to 1; user of the DROP tab to remove the user table from the database and to the SQL statement: (SELECT * DATA WHERE ID = 1); user of the DROP Tab;

Select * from users where username = ‘ “&username “ ‘ and password =’ “&userpassword& “ ‘ ”
If the username and password as provided by the user are used , the query to be submitted to the data base takes the form
Select * from users where username=’guest’ and password=’guestpass’
If the user were to enter [; or 1=1--] and [] instead of [guest] and [guestpass], the query would take the form: select *from users where username = ‘ ’ or 1=1-‘ and password=’ ’

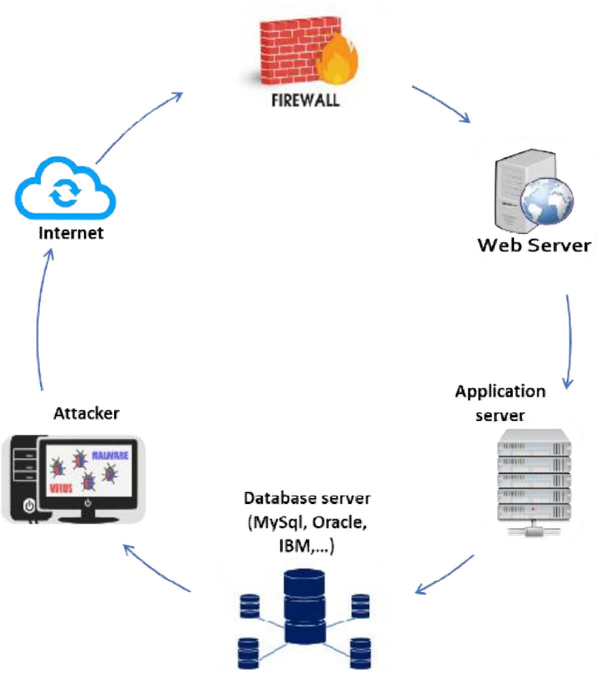


Fig. 1. the process of SQL injection

2.3.3 Database server vulnerabilities

Often the database system program has bugs like the MYSQL server real _escape_ string () feature vulnerability. An attacker will use this flaw to launch a powerful SQL injection attack using incorrect unified character encoding [96].

2.3.4 Blind SQL injection attack

A so-called blind SQL injection. An application is vulnerable to attack, but its impact is not apparent to the attacker. A SQL injection is put into the SQL becomes a "blind" one. The information could be expanded without being displayed but using valid statements, logical processes inserted into the system will lead to various developed data outcomes [97]. New bytes must be added for any byte added to the database when a unique string is tested. When you know where the flaw is located, you can use Absinthe to search for the corresponding target information.

2.3.5 Conditional response

It's worth noting that a SQL injection causes the database to calculate the value of a logical argument on a typical application screen:

- SELECT book title FROM booklist WHERE book Id
- ='00k14cd'AND 1 = 1 This leads to a standard face, while the statement SELECT book title FROM booklist WHERE book Id
- ='00k14cd'AND 1 = 2)
- The outcome can be different when a page is vulnerable to SQL injection attacks. This single injection would demonstrate that blind SQL injection is feasible, allowing an attacker to create claims that judge authenticity based on the content of a field in another table.

2·3.6 Conditional errors

If the WHERE statement is correct, the database is forced to judge a message that causes an error, resulting in a SQL error. Consider the following example:
(SELECT 1/0 FROM users WHERE username='Ralph'. If Ralph exists, dividing by zero will lead to errors.)

2.3.7 Time delay

A kind of blind SQL injection involves delaying the execution of a statement for a predetermined length of time. Under this logic, the SQL engine's implementation is likely to take a long time to execute or wait for a long queue to be completed. You may calculate the amount of time needed to load a page to decide whether or not a test statement is accurate [98].

2.4 SQL Injection Causes in PHP Code

A web application often expects to receive input from a user and a human to respond. The user must provide data to the server as part of the interaction procedure [99]. After receiving data from the user, the web application queries the database system to show the client conditionally. Users typically provide data to the server through the GET, POST, and Cookie methods. The GET technique involves directly writing the data to be submitted in the URL, and it is frequently used to transmit less data while browsing. The POST method is commonly used as a form submission and is frequently utilized when users need input data, which is usually a considerable amount of data. Cookies are small pieces of data that an online application saves in the user's browser buffer. They're typically used to store user IDs or track their browsing habits [100,101].

In Table 1, the three data transfer methods are described in terms of their characteristics and danger levels. The fundamental reason for SQL injection vulnerability is that the web application system does not immediately identify the transmitted data as it passes it. In the course of writing the code, the programmer uses the code directly. The information is combined with SQL commands before being sent to the database system for processing. If an attacker includes more SQL commands in the provided data, they will all be executed simultaneously, resulting in SQL injection.

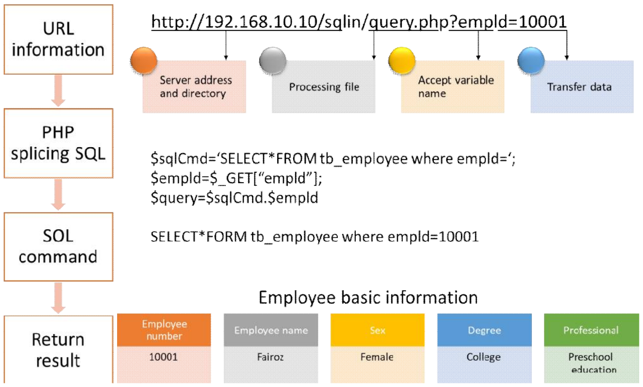
2.5 SQL Injection in the GET Mode

Since the data supplied by the GET method is written in the URL, it is straightforward to update and pass the data, and the data to be transferred is merely adjusted in the URL [102]. For example, the GET method is used to immediately transmit the employee number to the web application system during the regular use of the employee number to query employee information. Fig. 1 shows the URL and the SQL command data. Since the query's keyword is exposed explicitly in the URL, an attacker can find the passed variable name and data format and change the URL to add a SQL command to the injector's URL. For instance, inserting the "OR empId=10002" injection code can inquire and leak information from other employees. Fig. 2 shows the URL and SQL command details [103].

An attacker may receive information from all employees in extreme instances. For instance, the attack principle is the same as Fig. 3 in injections by using (OR embed > 0,) Scattered SQL controls must be updated to (SELECT * FROM tab employee when EmpId=10001 OR embed >0), and the original injection information must be substantially altered. By examining the SQL command, they noticed that it was added to the SQL command logically or operational embed information equal to or higher than 10001 or empId is taken from the tb employee table. The SQL statement is compatible with the implementation results (SELECT * FROM TB employee where the empId >0) since 10001 is more significant than 0. The integer number must be higher than 0 when the employee number is programmed in the employee number of the database. All the information of the employees may be obtained and the batch information leaked.

Table 1. Methods of web applications receive data from users

Techniques	Characteristics	Degree of Risk
Post	A large amount of data must be delivered, and it must be submitted via form forms or the AJAX post technique.	High
Get	Less data is passed by directly writing data in the URL.	High
Cookie	Sorted in the browser cache of the user's end, the data is large and sensitive	moderate



http://192.168.10.10/sqlin/query.php?empId=10001

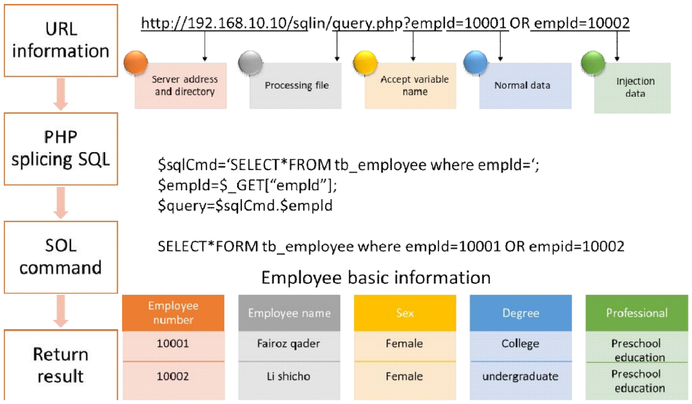
```
$sqlCmd=‘SELECT*FROM tb_employee where empId=‘;
$empId=$_GET[“empId”];
$query=$sqlCmd.$empId
```

SELECT*FORM tb_employee where empId=10001

Employee basic information

Employee number	Employee name	Sex	Degree	Professional
10001	Fairoz	Female	College	Preschool education

Fig. 2. The employee number usually is used to obtain information from the employee



http://192.168.10.10/sqlin/query.php?empId=10001 OR empId=10002

```
$sqlCmd=‘SELECT*FROM tb_employee where empId=‘;
$empId=$_GET[“empId”];
$query=$sqlCmd.$empId
```

SELECT*FORM tb_employee where empId=10001 OR empid=10002

Employee basic information

Employee number	Employee name	Sex	Degree	Professional
10001	Fairoz qader	Female	College	Preschool education
10002	Li shicho	Female	undergraduate	Preschool education

Fig. 3. The addition of information about injections leads to a lack of knowledge

2.6 SQL Injection in the POST Mode

The POST method's SQL injection vulnerabilities are the same as the GET method. After the user receives it, the web application system's data doesn't require security detection. The splicing with the SQL command is immediately the result. The POST method requires the Form, Delivery method, compared with the GET method Risk degree characteristics GET [104]—the transfer of data typed directly into the URL, less high data POST. Submitted using the form or POST method by AJAX giant cookie, a large amount of data to be transmitted. The data is vast and sensitive stored in the user end browser cache—moderate SQL Attack Principles of SQL Injection Site Prevention Technologies.

Furthermore, an annotation utilizing SQL commands, often prevalent upon user login, is one of the primary POST injection methods. For example, single-line comments ("--" and "#") commonly use a MySQL database. Once mention in the SQL command occurs, the contents of the statement will not be used as comments. If the attackers enter a login account and login code, an annotation can be used to eliminate the authentication portion and log in with the login account [105].

3. PREVENTION TECHNOLOGY FOR SQL VULNERABILITIES

The previous description of SQL injection vulnerabilities and standard methodology determined that the same features apply to all SQL injection locations. Programmer’s simply paying attention to the application function while designing application programs, have little regard for application security, and even have no basic sense of security protection [106,107]. Given the causes mentioned above, it is vital to present focused preventative measures and preventative strategies using summaries of the reasons for SQL injection vulnerabilities.

3.1 User Data Sensitive Keyword Filtering

The user receives the user data to the Web application system in PHP language ($_GET, $_POST) and other global variables. All user data received on the server-side are assumed to be "unsafe" and cannot be used straight away only after the sensitive keywords are filtered out. Mainly SQL instructions and special characters are typical filter keywords. Table 3 shows the particular filter content.

3.2 Using the Apache Server's Rewrite Module, prevent SQL Injection Attacks

To obtain search engine optimization (SEO) and concealed development technology and use it for the precaution against SQL injection, people commonly utilize the rewrite module on an Apache server. In truth, the Rewrite is a SQL injection prevention module that is relatively straightforward to use. In the settings Apache HTTPd.conf, it must be activated when using the Rewrite module. The replacement rules are configured in two ways [5]. The first is to set the virtual host when adding a replacement rule, and the second is to build an htaccess file to replace the rules file in the site directory [108]. The first benefit is that Apache automatically loads rule information when starting up, running efficiently, and replace the rules specified in the Apache Configuration File. However, after changing the replacement rules, you need to restart Apache, which causes business disruption. The second is a more flexible procedure. The replacement rules and the Apache setup file are separated. They come into effect immediately following the addition or modification of the rules. You need not restart Apache, although there is a low running efficiency [109].

Table 2. Prevent SQL injection filtering keywords

Content of filter	The purpose of the filter
Insert	Prevent the use of additional SQL commands to insert data
Select	Prevent subquests for injections
Delete	Prevention of further SQL command update removal
Update	Prevention of further SQL command update removal
Drop	Prevent further command up SQL
Truncate	Prevent further emptying of the SQL command
Special symbols	to include features such as (>, <, =, ', space, etc.) to block any more logic
Logical Operators	(AND, OR, NOT) PREVENT the inclusion of additional query criteria

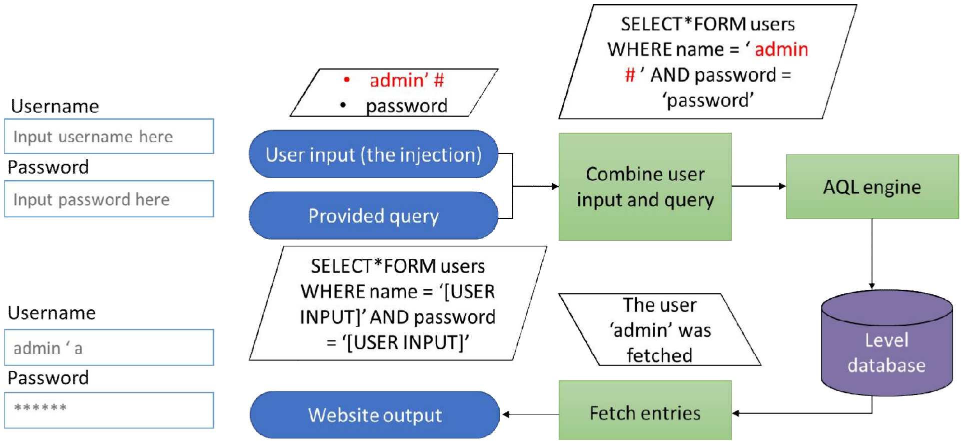

Username
Input username here
Password
Input password here
• admin' #
• password
User input (the injection)
Provided query
SELECT*FORM users WHERE name = ' admin # ' AND password = 'password'
Combine user input and query
AQL engine
SELECT*FORM users WHERE name = '[USER INPUT]' AND password = '[USER INPUT]'
Username
admin ' a
Password
******
The user 'admin' was fetched
Level database
Website output
Fetch entries

Fig. 4. an injection process that begins with user input, carries performed on the website, and concludes with a username and password interface

3.3 Using Various Techniques for Detection the Vulnerabilities of the Database Such as LDAP Injection (Lightweight Directory Access Protocol)

What is LDAP Injection and how to exploit it: LDAP or the so-called Lightweight Directory Access Protocol is one of the networking protocols, which is mainly used in the process of verifying the identity of the user in addition to his powers and the way to access any services within the network such as printers, shared files, and others. What can be done in LDAP? Through this protocol, many things can be done, such as:

Search found items (users, sources, groups, ...etc) [110,111].

It was comparing or checking whether a particular element has an input that contains a specific value or not.

Add a new item.
Delete an existing item.
Update an existing item.
Among the services or servers that deal with this protocol:
Apache Directory Server
Red Hat Directory Server
OpenLDAP
Novell eDirectory
Active Directory Microsoft
LDAP Query Language:

The LDAP protocol uses a straightforward query language. For example, if they want to query all the existing elements (user, group, printer, etc.) that contain the word Mohammad in the given name field, which is the first name, they write the following sentence: (givenName=mohammad).

In the previous example, the query clause contained one condition: the first name, so if they wanted to put more than one condition?

They put each condition separately and then put the condition type before these sentences, saying that the condition can be AND or OR. An example of this is the following: (&(givenName=mohammad)(l=amman)).

So that (l=amman) is to search in the property of the region Location and (givenName=mohammad) is to search in the first name and the sign & is to apply the two conditions together (and), in other words, all elements that have these two properties should be returned together.

If they write *amm ->, it means to search for the one ending in amm.

And if they write amm* — < it means to search for ones that start with amm.

And if they write *amm* — < it means to search for the one that contains amm.

If they write * by themselves ->, it means to return all elements that contain this property.

(givenName=*amm)
(givenName=amm*)
(givenName=*amm*)
(givenName=*)

Now that we have learned about this protocol and how to query through it and its uses, it is time to learn how and how to exploit the LDAP Injection vulnerability.

Suppose there is a login form on a site, and this form verifies the login data for users by checking within the LDAP. The query sentence will be as follows:

(&(USER=$Uname)(PASSWORD=$Pwd))

This sentence means checking both the user name and the password in LDAP. In this case, there are several ways to check whether this form is infected or not:

First method:

When entering the symbol * inside the username or password box, or both, the query sentence becomes as follows:

(&(USER=myUserName)(PASSWORD=*)) or
(&(USER=*)(PASSWORD=myPassword)) or
(&(USER=*)(PASSWORD=*))

In all of the previous cases, the query statement will return data because the condition is proper, and therefore they will bypass the entry form.

Second method:

If they enter in the UserName box the following value, for example:

AHMAD)(USER=ALI))

And they are sure that the users AHMED and ALI are present in the LDAP, but they don't know the password's value. The query statement will look like this:

(&(USER=
AHMAD)(USER=ALI)))(PASSWORD=*))

And since there is no between the first part of the sentence and (&(USER= AHMAD)(USER=ALI)))

And the second part is (PASSWORD=*))

Either & or | This will cause the first part of the sentence to be executed and the second part to be ignored, which leads to the statement returning a valid value because they knew that the users AHMAD and ALI already existed in LDAP...

Or the user name box can be injected with the following value if they already know a user name from the system users so that the result of the query statement is correct:

AHMAD)(&))

As for the injection of the password box, it is rare, because in most cases, a password HASH is done before it is sent to LDAP to be checked so that it is afraid of the type of HASH used from one server to another, some of which use MD5 and some of which use SHA and so on.

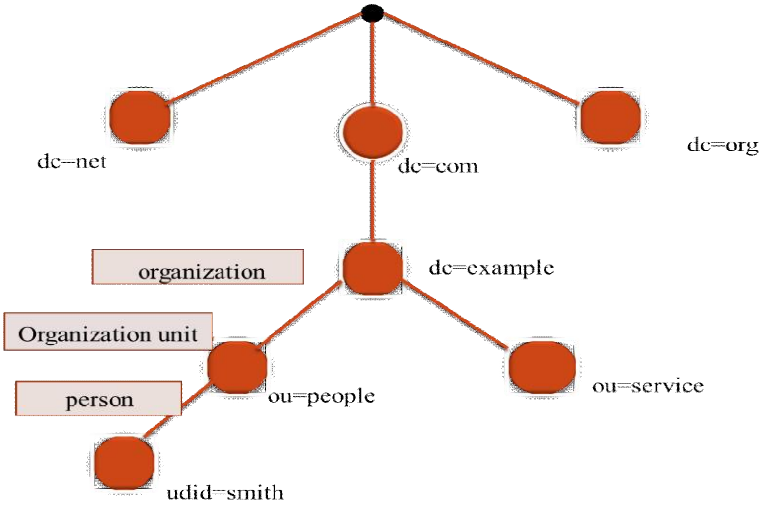


Fig. 5. flowchart of LDAP Directory Tree Where (DC= domain component, OU=OrganizationUnitName, UID=Userid)

How to protect the application from LDAP Injection?

Any LDAP web developer can protect their application from LDAP injection by doing a server-level validation of all input from browsers to the application so that only letters and numbers are allowed in.

4. RELATED WORK

Many researchers have discussed the subject of SQL injection, and in different ways, we will discuss some researchers and their techniques in this section as follows:

4.1 Prevention SQL Injection using PHP System

PHP is the most popular scripting language for web development. It is free, open-source, and server-side (the code is executed on the server). The following is the PHP code of the application that contains an SQL Injection vulnerability of some researchers.

The authors in [112] Present a way to prevent assaults by SQL on e-commerce websites. This is done by using PHP Data Object (PDO) and Prepared Statement to connect to the database, insert, update, select and enter Web forms. The approach employed in these documents is Agile Methodology, which follows planning, requirements analysis, design, coding, testing, and documentation in parallel during the production process phase. A penetration test was performed at many websites, one utilizing PHP data objects and prepared statements, and the other using a standard PHP script to see whether or not they are injectable. The perforated penetration demonstrated that SQL injection attacks were not visible on the PHP Data Object and Prepared Statement website. At the same time, other websites produced with standard PHP connectivity could be viewed and injected into SQL Injection. SQL injection attacks employing the PHP Data Object and Prepared Statement have blocked the system.

Other Author in [113] The SQL Injection Vulnerability Auto Black Box Test Proposes (SQL). This automates an SQLIV evaluation in SQLIA. Recent research has also demonstrated that existing SQLIVS has to be improved to reduce manual vulnerability inspection costs and the risk of being attacked because of inaccurate false-negative and false-positive results. This research aims to enhance the effectiveness of SQLI VS by suggesting in its development an object-orientated approach to decrease and contribute to the incidence of false-positive and false-negative results and enable potential researchers to improve a proposed scanner. As compared to the previous investigations, the results of the experimental study reveal a substantial improvement. Likewise, the analysis demonstrated that the suggested scanner could analyze the page response attacked by four different strategies.

In [114] “Neutralized SQL Injection web-based application attacks by server-side code changes," it offers a way to increase web safety by detecting Web-based SQL Injection assaults through server-based modifying code to limit vulnerability and alleviate fraudulent and damaging activities. This solution has been deployed on a simple website with a database to record users with a controlled administrator. The server utilized is local, and PHP was the background server code. MySQL was used for the front end. The scripting language on the PHP server-side was being used for code modification. 'PDO prepares' a parameter preparation to run. With its ability to block all kinds of SQL assaults, the solution provided proved efficient. Acunetix was used to test the code's vulnerability, and it was implemented with a simple database on a simple website. To validate the model, some prominent SQL injection attack tools and security data sets have been employed. The acquired results are promising, with a high precision rate for SQL injection detection.

Also, the Author used the PHP system in [103]. They introduced a hybrid technique in PHP, the popular server-side programming language, for preventing SQL assaults. This strategy is more effectively used without extensive string analyzer logic to prevent an attack of SQL injected into a dynamic Web content context. Initially, the application is built in safe mode to construct the query model for each hotspot. Dynamically produced queries are validated in the production environment. The results and analysis suggest the recommended solution prevent common SQL injection vulnerabilities is simple and effective.

In [5], researchers Highlighted carrying out a PHP language-based safety check. This tool may uncover SQL and cross-site scripting vulnerabilities to Cross-Site Scripting (XSS) by constructing a tree-like query structure before passing it to the parser. It helps extract invalid or a weak point easier by identifying sink points for the data flow information through target application sources. The presence of the sink-point is validated when a language-dependent parser compares the structure of the resulting tree query with queries. It leads to the classification of the vulnerability class according to their patterns when the sink point is found. The program may also analyze incoming user queries or ask for an applied tree-like structure using taught attack patterns. It allows the device to statistically assess the questions and decide how they might be fixed in the list of susceptible questions. Many Whitebox safety control instruments that may provide somesthetic or syntactic elements of a legitimate query may be expanded into a security assessment language but are typical with significant false alarms.

In [115] Proposes a vulnerability assessment technique to address the issue of SQL injection attacks. A pattern matching technique is used to categorize input and function variables into a design with attack vector properties. Then, a code analyzer is used to verify incorrect queries with sensory inputs. Tools to apply this method employ an innovative rule model to decide on possibly susceptible scripts for SQLI and XSS attacks. The experimental assessment reveals consistent results for PHP test boards and minor false alarms. A unique code is used to train data sets and test datasets of attack patterns to eliminate false alarms. The training dataset is divided into five log partitions of the repository and is used to test each test dataset using SQLI assault patterns and vectors.

In [116], a little method based on an EQA encoding concerning traffic metrics such as request and response time and message length is described. EQA conceals SQL related to the database and avoids some common SQL injection kinds (Tautology, Piggybacked, and Comment). Using MySQL and PHP environment and Wireshark platform, EQA is developed and tested. The findings show that the proposal has high safety performance and reduces HTTP demand, response time, and message length.

The Author in [117] Developed a new framework for developing a web-based application based on the Model View Controller's architectural design and Ajax technology (MVC). Ajax technology with its built-in library is implemented in the framework. Their findings revealed that the new PHP web application framework might assist users in constructing dynamic and real-time web applications as a web application development tool.

4.2 Prevention SQL Injection Using Various Techniques

In [118], Using multiple assumptions, the researchers proposed a software component engineering technique for discovering vulnerabilities in SQL injection attacks. Researchers developed a method to decrease false warnings, and this work aims to address flaws in currently available SQL injection vulnerability scanners. For valid and erroneous query requests, the established SQL detection algorithm is based on grammatical, structural patterns. The test results show that the SQL detection is highly accurate and false-positive when a valid query request contains a phrase (e.g., Orton, Fernando, Hernando, Armando, etc.) that is considered suspicious in the grammatical tree model built.

Table 3. Prevention SQL injection using PHP system

Techniques	Advantages	Disadvantages
PHP Data Object, Prepared Statement	the PHP Data Object and Prepared Statement could block the system.	SQL IDs, phrases, and keywords cannot be used as arguments in prepared statements.
PHP, MySQL, Acunetix	using these three techniques has a lot of advantages one of them is has a high precision rate for SQL injection detection.	MySQL disadvantages are not very efficient in handling the same number of databases.
PHP, MVC, Ajax	The PHP and MVC techniques have a positive result for the security of web technology	But the Ajax technique disadvantages is not worked on all browsers, and security is more diminutive.
PHP, EQ	The advantages of EQA, it has high safety performance, reduce HTTP demand, response time, and the length of the message	EQA does not measure routine performance and has a high workload for (NRL).

Also, the authors in [119] The three most often used content management systems have compared SQLi vulnerabilities utilizing Nikto, a vulnerability scanner called SQLMAP for penetration testing. This was implemented on the default pages of WordPress, Drupal, and Joomla on the LAMP server (Localhost). Results demonstrated that all content management systems were not susceptible to assaults via SQLi, but provided alerts on other potential flaws.

The Author in [54] Applied a Query Tokenization to express each SQL query, use the Word2vec Skip-gram model to create word embedding for each query, and then train the SVM classification system using eigenvector to identify fraudulent requests. Tokenization Experimental results indicate the efficiency and low overhead performance of all forms of SQL assaults, in particular tautological assaults.

In [113], They suggested a concept of SQL injection attack detection by comparing the language and the syntax tree. They tried to overcome formidable obstacles and many forms of SQL injection vulnerabilities not before uncovered. The grammar correspondence is based on rules-based pattern mining, collecting and analyzing signature patterns in the SQLIA mode stored. The methodology also employs attack patterns, signature and feature sets that have been learned as SQL injections, and trained data set evaluated in different techniques for machine learning. For testing and evaluating prototypes in two testbed situations, SVM, Naive Bayesian, Random Forest, ID3, and K-means are employed. Experimental findings reveal that BTestSet1dataset results in all five algorithms more accurately than GTestSet1 in ATTAR. On the other side, by having more SQLIA assaults, the flawed alarm of the model can be enhanced.

The researchers in [120] Proposed A-Gap Weighted String Subsection Algorithm for categorizing genuine and malicious queries based on the extraction properties of a string match. The methodology proposed utilizes trained and tested examples to detect and halt the SQL injections tautology type dynamically. The model test cases for current or future tautological SQL injection attacks are prepared employing attack extraction features using the support vector machine (SVM). The investigation reveals that the suggested solution can block existing SQL injection tautology attacks and launch new SQL assaults with comparable assault patterns and payloads with the current one.

In [121] Suggested a free safe SQL injection technique to identify and prevent SQL assaults. Java and algorithm implementation outlines the way they follow SQL Injection Prevention Procedures. Comparison of similar assault types with the characteristics Lastly, the assessment shows that the algorithm detects SQLIAs with high efficiency.

On the other hand, the authors in [122] The new DIAVA Traffic-based SQLIA Vulnerability Analysis and Detection Framework has been launched, allowing tenants proactively to provide alerts quickly. DIVA can detect successful SQLIAs among all suspects correctly by evaluating the two-way network traffic of SQL operations and using their suggested regular multi-level model for expression. Meanwhile, the severity of such SQLIAs may be promptly assessed by DIAVA based on its GPU-based dictionary attack analysis engine as well as the vulnerabilities of the related disclosed data. Research shows that DIAVA conducts advanced WAFs for SQLA detection from the point of view of precision and recall and allows the assessment of leaked data causing SQL injection in real-time.

The authors in [123] suggest that the suggested system was evaluated using an SQlMapproject attack tool utilizing two safety measures, namely the use of an intrusion detection system as a sensor to identify an attack occurring on SQL and the use of a web-based firewall (Mod secure) as a security system to stop assaults. SQLMapproject is used before and after protecting web applications. The findings suggest that the safety system presented works properly and can successfully safeguard the web-based database system, high performance, and efficiency.

In [124], the researchers proposed a new technique for accurate SQL injection detection based on neural networks. Their approach is genuine, effective, and practical since they start by obtaining Internet Service Provider (ISP) user URL access log data and ensure that it is true, accurate, and valuable. For data analysis, they perform a statistical study on regular data and SQL injection data. Based on the findings shown in the chart, plan to use eight different characteristics and train an MLP model. The model retains an extremely high accuracy of

Table 4. Prevention SQL injection using various techniques

Techniques	Advantages	Disadvantages
Nikto, SQLMAP	all content management systems were not susceptible to assaults via SQLi but provided alerts on other potential flaws.	The most common disadvantages of SQLMAP are the pilferage of data.
SQL injection with (SVM)	launch new SQL assaults with comparable assault patterns and payloads with the current one.	SVM methods do not make suitable with the large number of data set.
SQL injection with neural network	its advantages are the accuracy of neural networks for detecting SQL injection is superior to the relevant machine learning algorithms.	These algorithms often demand far more data than the classic machine learning techniques.
SQLA, DIAVA, SQL injection	DIAVA not only conducts advanced WAFs for SQLA detection from the point of view of precision and recall but also allows the assessment of leaked data causing SQL injection in real-time	The more significant false-negative rate of this input process due to DIAVA: a Traffic-based Framework for SQL Injection Attacks Detection

around 99 percent. Meanwhile, they assess and compare various machine learning algorithms (LSTM, for example). The findings show that the technique provides better outcomes than these related machine learning techniques.

5. ASSESSMENT AND RECOMMENDATION

PHP remains the most popular server-side language for websites and web applications. According to the latest data from w3techs, it is used by 79% of websites whose server-side language is known. Therefore, secure PHP programming and configuration are of critical importance. SQL Injection vulnerabilities have been on the OWASP Top 10 list since its beginning. They may appear in all languages, including the Web's two most popular languages, PHP and Java. SQL Injection vulnerabilities pose a serious threat to sensitive data and web application security in general. Attackers may use malicious code to get complete control of the system. In this study, we can compare PHP results with other techniques, such as the authors in [36] used a PHP system to prevent SQL injection attacks. The results and analysis show the approach is simple and effective to avoid common SQL injection vulnerabilities. Also, we can see the authors in [115] used PHP Data Object (PDO) to preventing SQL injection attacks on e-commerce websites. The penetration test showed that the system was entirely controlled from SQL injection attacks using PHP Data Object and Prepared Statement. Using various techniques for detection SQL injection, the authors in [53] Proposed A-Gap Weighted String Subsection Algorithm to categorize genuine and malicious queries based on the extraction properties of a string match. The methodology proposed utilizes trained and tested examples to detect and halt the SQL injections tautology type dynamically. The model test cases for current or future tautological SQL injection attacks are prepared to utilize attack extraction features using the support vector machine (SVM). The investigation reveals that the suggested solution can block existing SQL injection tautology attacks and launch new SQL assaults with comparable assault patterns and payloads with the current one.

6. CONCLUSION

There are many dangerous functions in PHP that can cause high-risk vulnerabilities in PHP web applications if web application developers misuse them. PHP Web application developers must do stringent filtering of user input so that most exposures can be avoided. In general, the more convenient it is for developers, the more security risks it may mean. This paper expounds and summarizes the attack principle and attack implementation SQL injection attack Principles and Preventive Techniques for PHP Sites. Process of SQL injection and demonstrates the SQL injection vulnerability exploiting method by manual injection. Because the attack principle has certain universality, the Web application systems developed in other languages can also use the practices described in this article for security testing. With the continuous improvement of SQL injection technology, as long as the Web is still used in programs or source code, there are still vulnerabilities and hidden dangers. Based on the summary of the

SQL injection, this article proposes a variety of suggestions for preventing SQL injection during the development of Web application systems.

DISCLAIMER

The products used for this research are commonly and predominantly use products in our area of research and country. There is absolutely no conflict of interest between the authors and producers of the products because we do not intend to use these products as an avenue for any litigation but for the advancement of knowledge. Also, the research was not funded by the producing company rather it was funded by personal efforts of the authors.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Antunes N, Vieira M. "Comparing the effectiveness of penetration testing and static code analysis on the detection of sql injection vulnerabilities in web services," in 2009 15th IEEE Pacific Rim International Symposium on Dependable Computing. 2009;301-306.
2. Halfond WG, Viegas J, Orso A. "A classification of SQL-injection attacks and countermeasures," in Proceedings of the IEEE international symposium on secure software engineering. 2006;13-15.
3. Patel N, Mohammed F, Soni S. SQL injection attacks: techniques and protection mechanisms. International Journal on Computer Science and Engineering. 2011;3:199-203.
4. Ntagwabira L, Kang SL. "Use of query tokenization to detect and prevent SQL injection attacks," in 2010 3rd International Conference on Computer Science and Information Technology. 2010;438-440.
5. Zhang H, Zhang X. "SQL injection attack principles and preventive techniques for PHP site," in Proceedings of the 2nd International Conference on Computer Science and Application Engineering. 2018;1-9.
6. Stobart S. Vassileiou M. "MySQL database and PHPMy admin installation," in PHP and MySQL Manual, ed.
7. Abdullah RM, Ameen SY, Ahmed DM, Kak SF, Yasin HM, Ibrahim IM, et al. Paralinguistic speech processing: An overview. Asian Journal of Research in Computer Science. 2021;34-46.
8. Ibrahim IM, Ameen SY, Yasin HM, Omar N, Kak SF, Rashid ZN, et al. Web server performance improvement using dynamic load balancing techniques: A review. Asian Journal of Research in Computer Science. 2021;47-62.
9. Zebari IM, Zeebaree SR, Yasin HM. "Real time video streaming from multi-source using client-server for video distribution," in 2019 4th Scientific International Conference Najaf (SICN). 2019;109-114.
10. Ahmed DM, Ameen SY, Omar N, Kak SF, Rashid ZN, Yasin HM, et al. A state of art for survey of combined iris and fingerprint recognition systems. Asian Journal of Research in Computer Science. 2021;18-33.
11. Maulud DH, Ameen SY, Omar N, Kak SF, Rashid ZN, Yasin HM, et al. Review on natural language processing based on different techniques. Asian Journal of Research in Computer Science. 2021;1-17.
12. Thiyab RM, Ali M, Basil F. "The impact of SQL injection attacks on the security of databases," in Proceedings of the 6th International Conference of Computing & Informatics. 2017;323-331.
13. Kaur N, Kaur P. SQL injection–anatomy and risk mitigation. Cover Story What, Why and How of Software Security 7 Cover Story Developing Secure Software. 2014;9:27.
14. Yasin HM, Zeebaree SR, Zebari IM. "Arduino based automatic irrigation system: Monitoring and SMS controlling," in 2019 4th Scientific International Conference Najaf (SICN). 2019; 109-114.
15. Salih AA, Ameen SY, Zeebaree SR, Sadeeq MA, Kak SF, Omar N, et al. Deep learning approaches for intrusion detection. Asian Journal of Research in Computer Science. 2021;50-64.
16. Kindy DA, Pathan ASK. "A survey on SQL injection: Vulnerabilities, attacks, and prevention techniques," in 2011 IEEE 15th international symposium on consumer electronics (ISCE). 2011;468-471.
17. Al Janaby AO, Al-Omary A, Ameen SY, Al-Rizzo H. Tracking and controlling high-speed vehicles via CQI in LTE-A systems. International Journal of Computing and Digital Systems. 2020;9:1109-1119.

18. Mohammed K, Ameen S. Performance investigation of distributed orthogonal space-time block coding based on relay selection in wireless cooperative systems; 2019.
19. Fawzi LM, Alqarawi SM, Ameen SY, Dawood SA. Two levels alert verification technique for smart oil pipeline surveillance system (SOPSS). International Journal of Computing and Digital Systems. 2019;8:115-124.
20. Zeebaree S, Yasin HM. Arduino based remote controlling for home: Power saving, security and protection. International Journal of Scientific & Engineering Research. 2014;5:266-272.
21. Al-Sultan MR, Ameen SY, Abduallah WM. Real time implementation of stegofirewall system. International Journal of Computing and Digital Systems. 2019;8:498-504.
22. Zeebaree S, Zebari I. Multilevel client/server peer-to-peer video broadcasting system. International Journal of Scientific & Engineering Research. 2014;5:260-265.
23. Al Janaby AO, Al-Omary A, Ameen SY, Al-Rizzo HM. "Tracking high-speed users using SNR-CQI mapping in LTE-A networks," in 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT). 2018;1-7.
24. Hasan BMS, Ameen SY, Hasan OMS. Image authentication based on watermarking approach. Asian Journal of Research in Computer Science. 2021;34-51.
25. Taher KI, Saeed RH, Ibrahim RK, Rashid ZN, Haji LM, Omar N, et al. Efficiency of semantic web implementation on cloud computing: A review. Qubahan Academic Journal. 2021;1:1-9.
26. Zebari S, Yaseen NO. Effects of parallel processing implementation on balanced load-division depending on distributed memory systems. J. Univ. Anbar Pure Sci. 2011;5:50-56.
27. Othman A, Ameen SY, Al-Rizzo H. Dynamic switching of scheduling algorithm for. International Journal of Computing and Network Technology. 2018;6,.
28. Hamed ZA, Ahmed IM, Ameen SY. Protecting windows OS against local threats without using antivirus. Relation. 2020;29:64-70.
29. Kareem FQ, Zeebaree SR, Dino HI, Sadeeq MA, Rashid ZN, Hasan DA, et al. A survey of optical fiber communications: challenges and processing time influences. Asian Journal of Research in Computer Science. 2021;48-58.
30. Ameen SY, Ali ALSH. A comparative study for new aspects to quantum key distribution. Journal of Engineering and Sustainable Development. 2018;11:45-57.
31. Fawzi LM, Ameen SY, Alqaraawi SM, Dawwd SA. Embedded real-time video surveillance system based on multi-sensor and visual tracking. Appl. Math. Infor. Sci. 2018;12:345-359.
32. Omer MA, Zeebaree SR, Sadeeq MA, Salim BW, Mohsin SX, Rashid ZN, et al. Efficiency of malware detection in android system: A survey. Asian Journal of Research in Computer Science. 2021;59-69.
33. Ali ZA, Ameen SY. Detection and prevention cyber-attacks for smart buildings via private cloud environment. International Journal of Computing and Network Technology. 2018;6:27-33.
34. Rashid ZN, Zeebaree S, Sengur A. Novel remote parallel processing code-breaker system via cloud computing. ed: TRKU; 2020.
35. Haji SH, Ameen SY. Attack and Anomaly detection in iot networks using machine learning techniques: A review. Asian Journal of Research in Computer Science. 2021;30-46.
36. Rashid ZN, Zeebaree SR, Shengul A. "Design and analysis of proposed remote controlling distributed parallel computing system over the cloud," in 2019 International Conference on Advanced Science and Engineering (ICOASE). 2019;118-123.
37. Mohammed BA, Ameen SY. A comparison of adaptive equalization techniques for MIMO-OFDM system; 2017.
38. Rashid ZN, Zebari SR, Sharif KH, Jacksi K. "Distributed cloud computing and distributed parallel computing: A review," in 2018 International Conference on Advanced Science and Engineering (ICOASE). 2018;167-172.
39. Fawzi LM, Ameen SY, Dawwd SA, Alqaraawi SM. Comparative study of ad-hoc routing protocol for oil and gas pipelines surveillance systems. International Journal of Computing and Network Technology. 2016;4.
40. Rashid ZN, Sharif KH, Zeebaree S. Client/Servers clustering effects on CPU

execution-time, CPU usage and CPU Idle depending on activities of Parallel-Processing-Technique operations. Int. J. Sci. Technol. Res. 2018;7:106-111.

41. Farhan FY, Ameen SY. "Improved hybrid variable and fixed step size least mean square adaptive filter algorithm with application to time varying system identification," in 2015 10th System of Systems Engineering Conference (SoSE). 2015;94-98.
42. Jijo BT, Zeebaree SR, Zebari RR, Sadeeq MA, Sallow AB, Mohsin S, et al. A comprehensive survey of 5G mm-wave technology design challenges. Asian Journal of Research in Computer Science. 2021;1-20.
43. Izadeen GY, Ameen SY. Smart android graphical password strategy: A review. Asian Journal of Research in Computer Science. 2021;59-69.
44. Sadeeq MA, Zeebaree S. Energy management for internet of things via distributed systems. Journal of Applied Science and Technology Trends. 2021;2:59-71.
45. Othman A, Ameen SY, Al-Rizzo H. A new channel quality indicator mapping scheme for high mobility applications in LTE systems. Journal of Modeling and Simulation of Antennas and Propagation. 2015;1:38-43.
46. Maulud DH, Zeebaree SR, Jacksi K, Sadeeq M.AM, Sharif KH. State of art for semantic analysis of natural language processing. Qubahan Academic Journal. 2021;1:21-28.
47. Othman A, Othman SY, Al-Omary A, Al-Rizzo H. Comparative performanceof subcarrier schedulers in uplink LTE-A under high users' mobility. International Journal of Computing and Digital Systems. 2015;4.
48. Sadeeq MM, Abdulkareem NM, Zeebaree SR, Ahmed DM, Sami AS, Zebari RR. IoT and cloud computing issues, challenges and opportunities: A review. Qubahan Academic Journal. 2021;1:1-7.
49. Haji SH, Zeebaree SR, Saeed RH, Ameen SY, Shukur HM, Omar N, et al. Comparison of software defined networking with traditional networking. Asian Journal of Research in Computer Science. 2021;1-18.
50. Shukur H, Zeebaree SR, Ahmed AJ, Zebari RR, Ahmed O, Tahir BSA, et al. A state of art survey for concurrent computation and clustering of parallel computing for distributed systems. Journal of Applied Science and Technology Trends. 2020;1:148-154.
51. Zeebaree S, Ameen S, Sadeeq M. Social media networks security threats, risks and recommendation: A case study in the kurdistan region. International Journal of Innovation, Creativity and Change. 2020;13:349-365.
52. Jacksi K, Ibrahim RK, Zeebaree SR, Zebari RR, Sadeeq MA. "Clustering documents based on semantic similarity using HAC and K-mean algorithms," in 2020 International Conference on Advanced Science and Engineering (ICOASE). 2020;205-210.
53. Hassan RJ, Zeebaree SR, Ameen SY, Kak SF, Sadeeq MA, Ageed ZS, et al. State of art survey for iot effects on smart city technology: challenges, opportunities, and solutions. Asian Journal of Research in Computer Science. 2021;32-48.
54. Sadeeq MA, Abdulazeez AM. "Neural networks architectures design, and applications: A review," in 2020 International Conference on Advanced Science and Engineering (ICOASE). 2020;199-204.
55. Othman A, Ameen SY, Al-Rizzo H. An energy-efficient MIMO-based 4G LTE-A adaptive modulation and coding scheme for high mobility scenarios. International Journal of Computing and Network Technology. 2015;3.
56. Sulaiman MA, Sadeeq M, Abdulraheem AS, Abdulla AI. Analyzation study for gamification examination fields. Technol. Rep. Kansai Univ. 2020;62:2319-2328.
57. Ameen SY. Advanced encryption standard (AES) enhancement using artificial neural networks. Int J of Scientific & Engineering Research. 2014;5.
58. Sadeeq M, Abdulla AI, Abdulraheem AS, Ageed ZS. Impact of electronic commerce on enterprise business. Technol. Rep. Kansai Univ. 2020;62:2365-2378.
59. Yasin HM, Zeebaree SR, Sadeeq MA, Ameen SY, Ibrahim IM, Zebari RR, et al. IoT and ICT based smart water management, monitoring and controlling system: A review. Asian Journal of Research in Computer Science. 2021;42-56.
60. Alzakholi O, Shukur H, Zebari R, Abas S, Sadeeq M. Comparison among cloud technologies and cloud performance.

Journal of Applied Science and Technology Trends. 2020;1:40-47.

61. Dino HI, Zeebaree S, Salih AA, Zebari RR, Ageed ZS, Shukur HM, et al. Impact of process execution and physical memory-spaces on OS performance. Technology Reports of Kansai University. 2020;62:2391-2401.

62. Ageed Z, Mahmood MR, Sadeeq M, Abdulrazzaq MB, Dino H. Cloud computing resources impacts on heavy-load parallel processing approaches. IOSR Journal of Computer Engineering (IOSR-JCE). 2020;22:30-41.

63. Sallow A, Zeebaree S, Zebari R, Mahmood M, Abdulrazzaq M, Sadeeq M. Vaccine tracker. SMS reminder system: Design and implementation; 2020.

64. Abdullah SMSA, Ameen SYA, Sadeeq MA, Zeebaree S. Multimodal emotion recognition using deep learning. Journal of Applied Science and Technology Trends. 2021;2:52-58.

65. Sadeeq MA, Zeebaree SR, Qashi R, Ahmed SH, Jacksi K. "Internet of Things security: A survey," in 2018 International Conference on Advanced Science and Engineering (ICOASE). 2018;162-166.

66. Salih AA, Zeebaree S, Abdulraheem AS, Zebari RR, Sadeeq M, Ahmed OM. Evolution of mobile wireless communication to 5G revolution. Technology Reports of Kansai University. 2020;62:2139-2151.

67. Abdulraheem AS, Salih AA, Abdulla AI, Sadeeq M, Salim N, Abdullah H, et al. Home automation system based on IoT; 2020.

68. Abdulazeez AM, Zeebaree SR, Sadeeq MA. Design and implementation of electronic student affairs system. Academic Journal of Nawroz University. 2018;7:66-73.

69. Abdulla AI, Abdulraheem AS, Salih AA, Sadeeq M, Ahmed AJ, Ferzor BM, et al. Internet of things and smart home security. Technol. Rep. Kansai Univ. 2020;62:2465-2476.

70. Aziz ZAA, Ameen SYA. Air pollution monitoring using wireless sensor networks. Journal of Information Technology and Informatics. 2021;1:20-25.

71. Sallow AB, Sadeeq M, Zebari RR, Abdulrazzaq MB, Mahmood MR, Shukur HM, et al. An investigation for mobile malware behavioral and detection techniques based on android platform. IOSR Journal of Computer Engineering (IOSR-JCE). 2020;22:14-20.

72. Shukur H, Zeebaree S, Zebari R, Zeebaree D, Ahmed O, Salih A. Cloud computing virtualization of resources allocation for distributed systems. Journal of Applied Science and Technology Trends. 2020;1:98-105.

73. Dino HI, Zeebaree SR, Hasan DA, Abdulrazzaq MB, Haji LM, Shukur HM. "COVID-19 diagnosis systems based on deep convolutional neural networks techniques: A review," in 2020 International Conference on Advanced Science and Engineering (ICOASE). 2020;184-189.

74. Abdulqadir HR, Zeebaree SR, Shukur HM, Sadeeq MM, Salim BW, Salih AA, et al. A study of moving from cloud computing to fog computing. Qubahan Academic Journal. 2021;1:60-70.

75. Ageed ZS, Zeebaree SR, Sadeeq MA, Abdulrazzaq MB, Salim BW, Salih AA, et al. A state of art survey for intelligent energy monitoring systems. Asian Journal of Research in Computer Science. 2021;46-61.

76. Mohammed SM, Jacksi K, Zeebaree SR. "Glove Word Embedding and DBSCAN algorithms for Semantic Document Clustering," in 2020 International Conference on Advanced Science and Engineering (ICOASE). 2020;1-6.

77. Ageed ZS, Zeebaree SR, Sadeeq MM, Kak SF, Rashid ZN, Salih AA, et al. A survey of data mining implementation in smart city applications. Qubahan Academic Journal. 2021;1:91-99.

78. Amanuel SVA, Ameen SYA. Device-to-device communication for 5G security: A review. Journal of Information Technology and Informatics. 2021;1:26-31.

79. Zebari RR, Zeebaree SR, Sallow AB, Shukur HM, Ahmad OM, Jacksi K. "Distributed denial of service attack mitigation using high availability proxy and network load balancing," in 2020 International Conference on Advanced Science and Engineering (ICOASE). 2020;174-179.

80. Yahia HS, Zeebaree SR, Sadeeq MA, Salim NO, Kak SF, Adel AZ, et al. Comprehensive survey for cloud computing based nature-inspired algorithms optimization scheduling. Asian Journal of Research in Computer Science.2021;1-16.

81. Sharif KH, Ameen SY. "A review of security awareness approaches with special emphasis on gamification," in 2020 International Conference on Advanced Science and Engineering (ICOASE). 2020;151-156.
82. Ismael HR, Ameen SY, Kak SF, Yasin HM, Ibrahim IM, Ahmed AM, et al. Reliable communications for vehicular networks. Asian Journal of Research in Computer Science. 2021;33-49.
83. Ageed ZS, Zeebaree SR, Sadeeq MM, Kak SF, Yahia HS, Mahmood MR, et al. Comprehensive survey of big data mining approaches in cloud systems. Qubahan Academic Journal. 2021;1:29-38.
84. Khalid LF, Ameen SY. Secure Iot integration in daily lives: A review. Journal of Information Technology and Informatics. 2021;1:6-12.
85. Abdulrahman LM, Zeebaree SR, Kak SF, Sadeeq MA, Adel AZ, Salim BW, et al. A state of art for smart gateways issues and modification. Asian Journal of Research in Computer Science. 2021;1-13.
86. Yazdeen AA, Zeebaree SR, Sadeeq MM, Kak SF, Ahmed OM, Zebari RR. FPGA implementations for data encryption and decryption via concurrent and parallel computation: A review. Qubahan Academic Journal. 2021;1:8-16.
87. Malallah H, Zeebaree SR, Zebari RR, Sadeeq MA, Ageed ZS, Ibrahim IM, et al. A comprehensive study of kernel (issues and concepts) in different operating systems. Asian Journal of Research in Computer Science. 2021;16-31.
88. Abdullah DM, Ameen SY. Enhanced Mobile Broadband (EMBB): A review. Journal of Information Technology and Informatics. 2021;1:13-19.
89. Asmawi A, Sidek ZM, Razak S. "System architecture for SQL injection and insider misuse detection system," in IEEE Conference; 2008.
90. Asmawi A, Sidek ZM, Abd Razak S. "System architecture for SQL injection and insider misuse detection system for DBMS," in 2008 International Symposium on Information Technology. 2008;1-6.
91. Olson IM, Abrams MD. Computer access control policy choices. Computers & Security. 1990;9:699-714.
92. Ibrahim IM. Task scheduling algorithms in cloud computing: A review. Turkish Journal of Computer and Mathematics Education (TURCOMAT). 2021;12:1041-1053.
93. Caesarano AR, Riadi I. Network forensics for detecting SQL injection attacks using NIST method. Int. J. Cyber-Security Digit. Forensics. 2018;7:436-443.
94. Nasser A, Daher R. Detecting and preventing SQL injection attacks.
95. Wright JL, Larsen JW, McQueen M. "Estimating software vulnerabilities: A case study based on the misclassification of bugs in MySQL server," in 2013 International Conference on Availability, Reliability and Security. 2013;72-81.
96. Muscat I. Web vulnerabilities: identifying patterns and remedies. Network Security. 2016;5-10.
97. Wang Y, Guo WP, Chen CH. Research on SQL injection attacks and guard method in web project [J]. Computer Engineering and Design. 2010;5.
98. Nagpal B, Chauhan N, Singh N. A survey on the detection of SQL injection attacks and their countermeasures. Journal of Information Processing Systems. 2017;13:689-702.
99. Kromann FM, Beginning PHP. MySQL: from novice to professional: Apress; 2018.
100. Merlo E, Letarte D, Antoniol G. "Automated protection of php applications against SQL-injection attacks," in 11th European Conference on Software Maintenance and Reengineering (CSMR'07). 2007;191-202.
101. Boyd SW, Keromytis AD. "SQLrand: Preventing SQL injection attacks," in International Conference on Applied Cryptography and Network Security. 2004;292-302.
102. Som S, Sinha S, Kataria R. Study on sql injection attacks: Mode detection and prevention. International Journal of Engineering Applied Sciences and Technology, Indexed in Google Scholar, ISI etc., Impact Factor: 1.494. 2016;1:23-29.
103. Sadalkar K, Mohandas R, Pais AR. "Model based hybrid approach to prevent SQL injection attacks in PHP," in International Conference on Security Aspects in Information Technology. 2011;3-15.
104. Patel N, Wimmer H, Powell LM. PHPBB3 bulletin board security testing. Issues in Information Systems. 2020;21.
105. Jurásek P. PHP wander: A static vulnerability analysis tool for PHP; 2018.
106. Alwan ZS, Younis MF. Detection and prevention of sql injection attack: A survey. International Journal of Computer Science and Mobile Computing. 2017;6:5-17.

107. Qian L, Zhu Z, Hu J, Liu S. "Research of SQL injection attack and prevention technology," in 2015 International Conference on Estimation, Detection and Information Fusion (ICEDIF). 2015;303-306.
108. Sadegh MSS, Zarafshan F, Safari M, Rahimian A, Markazi I. Optimization of multi-agent security solution for prevent web-based system of SQL injection attack; 2018.
109. Zhang L, Zhang D, Wang C, Zhao J, Zhang Z. ART4SQLi: The ART of SQL injection vulnerability discovery. IEEE Transactions on Reliability. 2019;68:1470-1489.
110. Win SS, Thwin MMS. Preventive mechanism for potential security threats and attacks on virtual cloud ldap server.
111. Cueva-Hurtado M, Figueroa-Diaz R, Aguilar-Soto W, Armijos-Ordoñez M. Systematic literature review on the LDAP protocol as a centralized mechanism for the authentication of users in multiple systems revisión sistemática de literatura sobre el protocolo LDAP como mecanismo centralizado.
112. Rahman A, Islam MM, Chakraborty A. Security assessment of PHP web applications from SQL injection attacks. Journal of Next Generation Information Technology. 2015;6:56.
113. Aliero MS, Ghani I, Qureshi KN, Rohani MFA. An algorithm for detecting SQL injection vulnerability using black-box testing. Journal of Ambient Intelligence and Humanized Computing. 2020;11:249-266.
114. Sarjitus O, El-Yakub M. Neutralizing SQL injection attack on web application using server side code modification. Int J Sci Res Comput Sci Eng Inf Technol. 2019;5.
115. Shar LK, Tan HBK. "Predicting common web application vulnerabilities from input validation and sanitization code patterns," in 2012 Proceedings of the 27th IEEE/ACM international conference on automated software engineering. 2012;310-313.
116. Shwaish AK, Hussain MA, Al-Kashoash HA. Encoding Query Based Lightweight Algorithm for Preventing SQL injection attack. Journal of Basrah Researches ((Sciences)). 2020;46.
117. Adam SI, Andolo S. "A new PHP web application development framework based on MVC architectural pattern and ajax technology," in 2019 1st International Conference on Cybernetics and Intelligent System (ICORIS). 2019;45-50.
118. Gao H, Zhu J, Liu L, Xu J, Wu Y, Liu A. "Detecting SQL injection attacks using grammar pattern recognition and access behavior mining," in 2019 IEEE International Conference on Energy Internet (ICEI). 2019;493-498.
119. Ojagbule O, Wimmer H, Haddad RJ. "Vulnerability analysis of content management systems to SQL injection using SQLMAP," in Southeast Con. 2018;1-7.
120. McWhirter PR, Kifayat K, Shi Q, Askwith B. SQL Injection Attack classification through the feature extraction of SQL query strings using a gap-weighted string subsequence kernel. Journal of information security and applications. 2018;40:199-216.
121. Natarajan K, Subramani S. Generation of SQL-injection free secure algorithm to detect and prevent SQL-injection attacks. Procedia Technology. 2012;4:790-796.
122. Gu H, Zhang J, Liu T, Hu M, Zhou J, Wei T, et al. DIAVA: a traffic-based framework for detection of SQL injection attacks and vulnerability analysis of leaked data. IEEE Transactions on Reliability. 2019;69:188-202.
123. Dalimunthe RA, Sahren S. "Intrusion detection system and modsecurity for handling sql injection attacks," in International Conference on Social, Sciences and Information Technology. 2020;187-194.
124. Tang P, Qiu W, Huang Z, Lian H, Liu G. Detection of SQL injection based on artificial neural network. Knowledge-Based Systems. 2020;190:105528.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sdiarticle4.com/review-history/70376