ISI

Ingénierie des Systèmes d'Information

IIETA International Information and
Engineering Technology Association

Advancing the World of Information and Engineering

Editorial Board

Source details

Feedback ⟩   Compare sources ⟩

Ingenierie des Systemes d'Information

Years currently covered by Scopus: from 2012 to 2024

Publisher: Lavoisier

ISSN: 1633-1311   E-ISSN: 2116-7125

Subject area: (Computer Science: Information Systems)

Source type: Journal

View all documents ⟩   Set document alert   🖫 Save to source list

CiteScore 2023	
2.5	ⓘ

SJR 2023	
0.239	ⓘ

SNIP 2023	
0.670	ⓘ

CiteScore   CiteScore rank & trend   Scopus content coverage

CiteScore 2023 ▾

$$2.5 = \frac{1,177 \text{ Citations } 2020 - 2023}{463 \text{ Documents } 2020 - 2023}$$

Calculated on 05 May, 2024

CiteScoreTracker 2024 ⓘ

$$2.0 = \frac{978 \text{ Citations to date}}{481 \text{ Documents to date}}$$

Last updated on 05 August, 2024 • Updated monthly

IIETA International Information and Engineering Technology Association

Advancing the World of Information and Engineering

Search for

ISI

Ingénierie des Systèmes d'Information

- **ISSN:** 1633-1311 (print); 2116-7125 (online)
- **Indexing & Archiving:** Scopus, SCImago (SJR), CrossRef, Portico, EBSCOhost, Google Scholar, Publons, MIAR, ScienceOpen, Microsoft Academic, CNKI Scholar, Baidu Scholar
- **Subject:** Computer Sciences

2023 CiteScore 2.5

Scopus   SJR SCImago Journal & Country Rank   EBSCO HOST   ✓iThenticate   publons   scienceOPEN.com

Ingénierie des Systèmes d'Information (ISI) is a top-rated international journal devoted to publishing the most innovative models, algorithms, software and hardware for information systems.

The subject areas mainly include data management issues and data-related issues in the following fields: data mining, data management, information retrieval, process management, machine learning, scientific computing, data science and audiovisual information systems. We welcome implementation articles that deal with fault detection and tolerance, parallel and distributed data management, as well as general or special purpose hardware for data-intensive systems. We also welcome manuscripts from application domains like cloud platform, Internet of Things (IoT), and peer-to-peer environment. Contributions from industrial enterprises are also welcome.

All manuscripts should provide novel solutions to data management problems, namely, data models and performance enhancement, and demonstrate the application potential of these solutions. Moreover, all manuscripts should handle the research problems with solid empirical evidences from real-world or future applications. Systems papers must include rigorous experiment on prototype or robust simulations of real systems. Theoretical articles must have clear motivations from applications, develop innovative algorithms or extend existing solutions, and fully manifest the applicability of their ideas. The technical solutions must be explained clearly in a uniform notation, facilitating their applications beyond the research domain. The extremely complex details may be demonstrated with reference to previous studies.

Focus and Scope

The ISI welcomes original research papers, technical notes and review articles on various disciplines, including but not limited to:

- Data mining

Read More

Journal Content

All ▾ | Search

Browse
- By Issue
- By Title
- By Author
- Other Journals

Information

- For Readers
- For Authors
- For Librarians

JOURNAL METRICS

CiteScore 2023: 2.5 ⓘ

SCImago Journal Rank (SJR) 2023: 0.239 ⓘ

Source Normalized Impact per Paper (SNIP) 2023: 0.67 ⓘ

Top-Rated Journal

An international journal committed to the dissemination of advances in signal processing, imaging and visioning

1984-2019

TS **Traitement du Signal**

An IIETA Journal

SUBMIT NOW

Latest News & Announcement

Emerging Trends in Artificial Intelligence and Computer Engineering

---

Source details

Feedback ›   Compare sources ›

Ingenierie des Systemes d'Information

Years currently covered by Scopus: from 2012 to 2024

Publisher: Lavoisier

ISSN: 1633-1311   E-ISSN: 2116-7125

Subject area: (Computer Science: Information Systems)

Source type: Journal

View all documents ›   Set document alert   📩 Save to source list

CiteScore 2023
2.5   ⓘ

SJR 2023
0.239   ⓘ

SNIP 2023
0.670   ⓘ

CiteScore | CiteScore rank & trend | Scopus content coverage

CiteScore 2023 ▾

$$2.5 = \frac{1{,}177 \text{ Citations } 2020 - 2023}{463 \text{ Documents } 2020 - 2023}$$

Calculated on 05 May, 2024

CiteScoreTracker 2024 ⓘ

$$2.1 = \frac{1{,}106 \text{ Citations to date}}{519 \text{ Documents to date}}$$

Last updated on 05 October, 2024 • Updated monthly

Vol. 29, No. 5, October 2024

Security Analysis of SQL Injection Attacks on Multimedia and Journal-Services Sites Using Concatenated Input Validation and Parsing Method (CIVP)

Marvin Chandra Wijaya[ID]

Department of Computer Engineering, Maranatha Christian University, Bandung 40164, Indonesia

Corresponding Author Email: marvin.cw@eng.maranatha.edu

ABSTRACT

Web applications and databases continue to face grave danger from SQL injection attacks, which can result in unauthorized access, data modification, and system compromise. This report discusses the methods attackers use to exploit SQL injection vulnerabilities and emphasizes the dangers of successful attacks, such as data leaks and system compromise. This research proposes a comprehensive system for detecting SQL injection attacks using concatenated Input Validation and Parsing Method (CIVP). The site used as experimental material is the Multimedia and Journal Services Site. Based on the results of forensic analysis on the Journal Services Site, there were several attacks in cyberspace, including using SQLMAP and Python. The system created has successfully detected SQL injection attacks. Based on the test results, it was found that the use of the method proposed in this study succeeded in making processing time 15.2% more efficient. Experiments carried out with the method proposed in this study succeeded in increasing the attack detection accuracy from 96-97% to 99.5% with a p-value of 0.008446.

1. INTRODUCTION

The official site is an identity of an institution which is the identity or a mirror of the image of the institution. The official website contains the institution's identity, institution profiles, activities, internal news, and external news. Therefore, an official site must be guarded in such a way against attacks in cyberspace. The official site may be located and managed by a third party that provides website hosting services. In addition, an institution can manage its own official website. With self-management, there will be a lot of freedom and facilities that can be provided in the system. However, with self-management, the challenge of maintaining the site and information system becomes essential.

Apart from the official website, other websites are also very important to protect, such as e-commerce websites. Systems on e-commerce sites are also often attacked by irresponsible people [1]. Even though e-commerce is now widely used throughout the world, many are still vulnerable to attacks. Many e-commerce websites in various countries are down due to various attacks.

Websites, web applications, and web users have all been subject to severe and ongoing risks from web assaults, including SQL injection attack (SQLi), XSS, Operating System Command injection (CMDi), and Path traversal [2]. Because of the widespread usage of websites and online applications and the accessibility of web attack tools on the internet, these kinds of attacks are frequent [3]. The SQLi, XSS, CMDi, DDoS, and Path traversal (Path) web attack family is referred to as the "common web attacks" [4]. It is seen that now attacks via the "common web attacks" are

becoming more and more frequent nowadays [5]. Web attacks are becoming more massive day by day, requiring fast countermeasures [6]. To be able to deal with attacks quickly on websites, it is necessary to detect attacks properly and quickly.

There are various ways to attack a website, one of the most popular ways is SQL Injection Attack (SQLi). Website defacement is one of the biggest dangers for business, corporate, and government websites and web services.

Defacement will have negative implications for website owners, including disruption of various kinds and things that website developers will experience [7]. After the first attack step, the next step is to compromise the resources on the web server that has been attacked [8]. Therefore, the database in a web server needs strict security and resistance to attack [9].

One of the most frequent security risks to cloud-deployed web-based services is SQL injection attacks as shown in Table 1 [10]. More than 40% of attacks on the web are in the form of SQL injection, while the second largest attack is username or password disclosure only at 7%. That means the web protection against attacks is good to focus on protection against SQL Injection. SQL injection attackers can run dangerous and bad code on target databases to obtain or corrupt sensitive data by taking advantage of online software flaws.

SQL injection attacks are common online application vulnerabilities that can have serious security repercussions. SQL injection attacks can be especially harmful in the context of journal-services sites, where databases are used to store and retrieve information. By inserting malicious code into user-supplied input, an attacker can alter a SQL query, resulting in

unwanted and potentially destructive database activities.

Table 1. Most frequent attacks

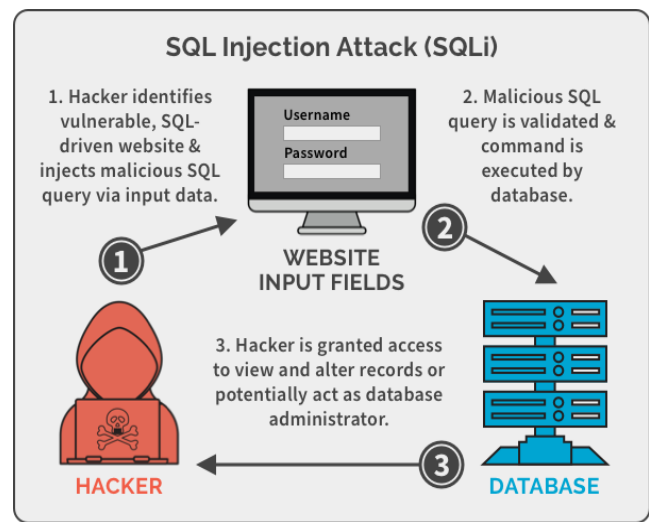| Vulnerability Types	# Vuln	#WS	Percentage
SQL Injection	*502*	*92*	*84.9%*
Possible Username or Password Disclosure	47	3	7.1%
Xpath Injection	20	2	3.1%
Possible Path Disclosure	17	5	3.1%
Possible Parameter Base Buffer Overflow	4	3	1%
Code Execution	2	2	0.6%
Total	**593**	**107**	**100%**

Figure 1. Illustration of SQL injection attacks

Figure 1 illustrates an SQL injection attack, in which a hacker identifies weaknesses in a website and injects SQL queries with input data. The server will execute malicious SQL queries to inject databases and hackers will gain access to the website.

In Indonesia, SQL injection attacks have targeted various sectors, including government websites, which are vulnerable to exploitation of sensitive data; e-commerce platforms, where attackers seek to expose customer information such as personal and financial details; and financial institutions, where banks and financial services face risks of data breaches. Several notable SQL injection incidents in Indonesia include the 2021 breach of government websites, where attackers defaced sites and leaked sensitive citizen data. E-commerce platforms have also been frequent targets, with hackers exploiting poorly secured payment systems to steal customer information, such as emails, passwords, and financial details. In the education sector, SQL injection has been used to compromise university databases, exposing student records and academic information. While specific cases are not always publicly detailed, these trends highlight the vulnerability of various sectors to such attacks.

Current research on SQL injection (SQLi) is focused on enhancing detection techniques, particularly through advanced methods like deep neural networks. One effective approach is the use of models such as recurrent neural networks (RNNs) and autoencoders, which can accurately detect SQLi by identifying patterns in database queries. These models leverage large datasets to learn the structure of both legitimate and malicious queries, significantly improving detection

accuracy compared to traditional methods. This evolving research demonstrates the growing importance of machine learning in combating SQL injection attacks.

2. LITERATURE REVIEW

SQL injection attacks can be classified based on intent: extracting data, adding data, modifying data, and others attacks. SQL injection attacks have several types: tautologies, illegal, logically incorrect queries, piggyback queries, stored procedures, and alternate encodings.

The system network has several security weaknesses because of the computer network's size and volume of information. In order to create an efficient and useful simulation model of computer network security evaluation, a system for network security evaluation must be built. Using the simulation model, network security impact can be increased. The simulation of global computer security evaluation is a novel topic in our nation since the reform and opening up. It has the ability to research network security thoroughly. Also, it can be used to construct a system for global security evaluation and study network security directly. It may assess, investigate, develop, and plan different phases in the computer network simulation system in order to play a significant role [11]. In this study, a new algorithm was implemented after analyzing the artificial network system model and addressing the neural network's weaknesses in convergence and search. Based on this analysis, a simulation model for computer network security was developed, and its performance was validated through appropriate testing. The results of the simulation highlight the model's exceptional performance and significant improvement potential.

Numerous websites access the World Wide Web using one of the many web servers that exist in the world. These websites are vulnerable to attacks, usually input validation-related ones. These attacks make website hacking simple and let anonymous users expose sensitive data. The open market is currently in a very dangerous state. The analysis carried out as previously said and on top of the computerized environment prompts us to conduct a study on SQL injection attacks and dangerous invasion approaches, that use runtime validation for detecting such assaults and tracking their event [12]. A technique for identifying and containing SQLIA issues is presented in this paper. The method involves a one-time offline process that employs stagnant application code analysis to extract an application's planned SQL query behaviour, which will take the form of a predetermined series of tokens.

In an effort to gain access to sensitive data, attackers are considering web apps as a prime target. A company may be vulnerable to different attacks if it does not implement efficient data protection mechanisms. To ensure effective data protection, government institutions in particular need to look outside the box when it comes to security measures. Therefore, it is crucial to do security testing and ensure that the system is secure before an attack occurs. One of the oldest, most common, and most dangerous online application vulnerabilities is the SQL Injection flaw because it may harm any website or web application that uses a SQL-based database. Utilizing various security systems is necessary to solve the SQL injection issues [13].

The main goal of conventional wireless application firewalls is to stop erroneous SQL requests. Few of them can rapidly assess the severity of an attack and precisely determine

whether it is truly detrimental. to make the renters more conscious of how severe a SQL injection attack is. In 2019, Gu et al. and associates presented DIAVA, a novel traffic-based SQL injection attack detection and vulnerability analysis platform that may proactively and immediately alert tenants. DIAVA can precisely identify successful SQL injection attacks from every SQL query input from bidirectional network traffic of SQL operations using the suggested multilayer regular expression model. DIAVA, meanwhile, can swiftly assess the seriousness of such SQL injection attacks and the vulnerabilities of the associated spilled data using its GPU-based dictionary attack analysis engine. According to experimental findings, DIAVA not only exceeds cutting-edge wireless application firewalls in terms of precision and recall when it comes to identifying SQL attacks, but it also offers real-time vulnerability evaluation of data leaks brought on by SQL injection [14]. SQL injection attacks (SIA) have recently grown to be a serious hazard to Web applications. Attackers can expose or control a Web application's back-end database through properly prepared user input.

Alkhathami and Alzahrani [15] in 2022 will detect SQL injection attacks using machine learning. SQL injection requests are divided into two groups by the model: attack and valid. Four machine learning algorithms are being used to train the model. After conducting data preprocessing and feature extraction. Authors used various classification methods to classify every SQL query. Figure 2 shows the steps of the model used in Jamilah's system.

In 2019, Tashenova et al. [16] conducted a study to look at various ways of SQL injection attacks. Different strategies for implementing SQL injection and techniques to prevent it were taken into consideration and experimentally used in the research effort. The author also comprehended the traits of SQL injections and how they connect to their fundamental structure. On the basis of this, it was experimentally put into practice, launching an assault on two web apps that had a similar interface but a different core structure. In other words, the second web application was secure, whereas the first web application was open to assault.

Volkova et al. [17] in 2019 studied the use of machine learning in advanced SQL injection attacks. The main goal of the research is to apply machine learning techniques for identifying injection features in the HTTP query string. Authors use various machine learning techniques. Deep Sequential Models and a Neural Network with Dropout layers were also used. The results demonstrated the benefits of using a machine-learning approach to identify harmful patterns in HTTP query strings. Figure 3 shows the steps of the SQL injection attack detection research scheme researched by Volkova et al.
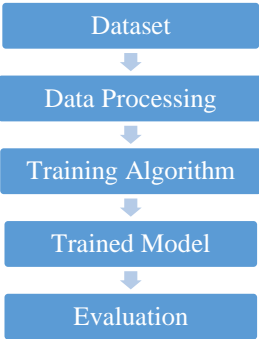
Figure 2. Jamilah's system model [15]
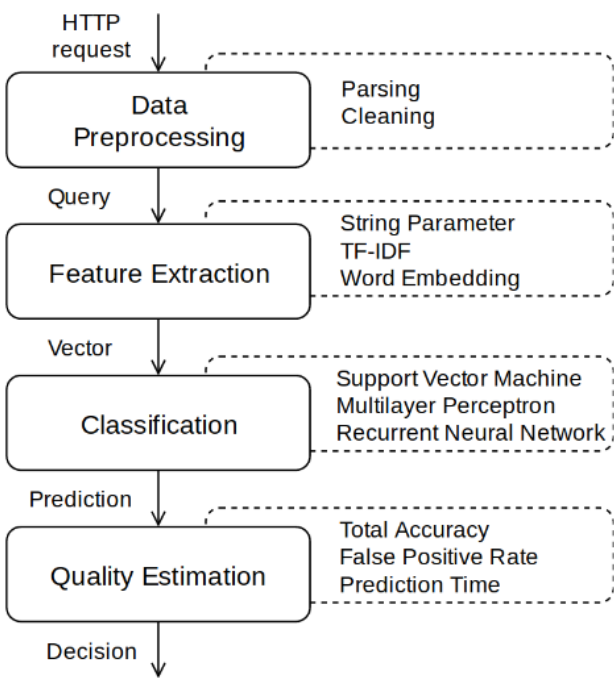
Figure 3. Marina Volkova's research scheme [17]

Bandhakavi et al. [18] studies to prevent SQL injection attacks using a technique called CANDID (candidate evaluations). The method proposed in this study for detecting SQL injection attacks focuses on comparing the query structure the programmer intended for any input with the structure of the actual query that gets executed. The authors introduce a simple and innovative approach to extract intended queries by continuously evaluating runs using well-formed candidate inputs. This theoretically robust technique operates by interpreting the symbolic query generated during program execution to infer the intended queries.

Research on SQL injection (SQLi) attacks on multimedia websites highlights significant vulnerabilities in systems handling media content, especially due to the complex nature of multimedia data and dynamic content delivery. Many of these sites rely heavily on databases to manage large volumes of user-generated content, video, and other media files, which makes them a prime target for SQLi attacks. Attackers can exploit weaknesses in these sites by injecting malicious SQL code through input fields, leading to unauthorized access, data breaches, or defacement of media content [19].

Recent studies emphasize the use of machine learning (ML) and hybrid techniques for detecting and preventing SQLi attacks. Approaches such as pattern-matching algorithms and the integration of deep learning methods like recurrent neural networks (RNNs) have shown promise in identifying malicious queries and preventing attacks in real-time. Additionally, encryption techniques (such as AES-128) and token-based authentication have been suggested to mitigate SQLi risks by securing database access and input validation. These methods aim to enhance detection accuracy while minimizing false positives, crucial for sites with heavy traffic and multimedia usage [20].

3. METHODOLOGY

A security analysis of SQL injection threats on websites using journal services is provided below.

- Impact on Data Confidentiality

SQL injection attacks may threaten the confidentiality of private data kept in the database. Attackers can create malicious SQL queries to retrieve data that they are not allowed to access. This situation could include user personal information from journal-services websites, such as names, email addresses, or research data.

- Impact on Data Integrity

Attacks using SQL injection can also change or manipulate database data. Attackers have the ability to alter the database's structure, add harmful data, or modify or delete records. This could result in the unlawful change or deletion of published papers, research data, or user accounts on sites that provide journal services.

- Impact on Availability

By establishing the database or the entire application unusable or crashing, SQL injection attacks can lead to denial-of-service scenarios. Attackers may take advantage of SQL query flaws to exhaust system resources or carry out laborious tasks, disrupting service for authorized users.

- Privilege Escalation

Attackers may be able to increase their privileges within the program through SQL injection attacks. Attackers can get around access controls and obtain administrator or superuser rights by inserting specially crafted SQL queries. As a result, the application and underlying database may be entirely under the control.

The procedures and steps proposed to mitigate SQL injection attacks in this study are shown in Figure 4. These procedures will be experimented on the Multimedia and journal service site. This procedure is designed to ensure the security of the application on the targeted website.

Before utilizing it in SQL queries, every user-supplied input should be checked for accuracy and cleaned up. In order to make sure that user input is regarded as data rather than executable code, prepared statements or parameterized queries should be utilized. By doing this, attackers are unable to inject malicious SQL code. The application's database user accounts should have the bare minimum of permissions. Avoid using privileged accounts or giving application users unauthorized access.
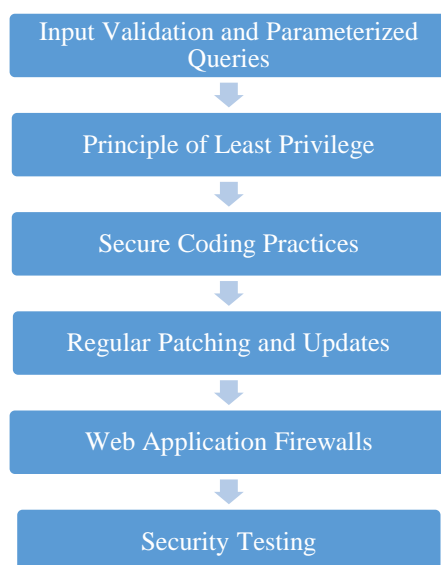
Developers should adhere to secure coding standards and refrain from concatenating user input into SQL queries. Instead, they ought to make use of the appropriate query-creation techniques offered by the employed programming language or framework. Update the application with the most recent security patches, upgrades, and the underlying database management system. This situation aids in defending against weaknesses that attackers might use. Install a web application firewall (WAF) to recognize and stop SQL injection threats. A WAF can offer an extra layer of security by scrutinizing incoming requests and denying those that display suspected SQL injection patterns. Conduct regular security audits, such as penetration tests, to find and fix the application's weaknesses. Potential SQL injection vulnerabilities can be found using automated tools and manual testing methods.

The theoretical analysis of SQL injection threats on websites, particularly those providing multimedia and journal services, highlights several critical impacts and mitigation strategies. SQL injection attacks can severely compromise data confidentiality by enabling unauthorized access to sensitive information, such as personal user details and research data. These attacks also pose a risk to data integrity, as they can alter, add, or delete database records, potentially tampering with published papers and user accounts. Furthermore, SQL injection can impact availability by disrupting the service through database crashes or resource exhaustion, leading to denial-of-service scenarios. Attackers might also exploit SQL injection to escalate privileges, bypassing access controls and gaining administrative rights, thereby gaining complete control over the application and database.

To counter these threats, the proposed procedures include validating and sanitizing all user-supplied input to ensure it is treated as data rather than executable code, employing prepared statements or parameterized queries to prevent code injection, and limiting database user permissions to the minimum required. Adherence to secure coding practices, regular updates, and the use of a web application firewall (WAF) are recommended to detect and block SQL injection attempts. Additionally, conducting regular security audits, including penetration tests, helps identify and address potential vulnerabilities in the application.

3.1 Input validation

An attack known as SQL injection takes advantage of websites' carelessness in allowing users to enter specific data without filtering out dangerous characters. Typically, users submit information into the search box or other areas of the website that communicate with the site's SQL database. The command that the attacker enters is typically a piece of information containing a specific link that takes the victim to a particular website that the attacker uses to retrieve the victim's personal information.

Developers can use programs like NoScript, an add-on for the Firefox web browser, to prevent dangerous links from websites subjected to SQL injection attacks. With SQL Injection, an attacker can access the database by sending commands to the server via URIs or form fields. As an example of a vulnerability in accessing a username:

```
statement="SELECT*FROM    users    WHERE
name=""+userName+";"
```


Figure 4. Proposed procedures to mitigate SQL injection attacks

(Figure contents, top to bottom:)
Input Validation and Parameterized Queries
Principle of Least Privilege
Secure Coding Practices
Regular Patching and Updates
Web Application Firewalls
Security Testing

The "userName" variable can be abused by careless users, even if the SQL code is intended to get the user's table records with a specific username. Setting the "userName" variable and executing the altering SQL statement with:

```
SELECT*FROM users WHEREname='' OR '1'='1';
```

Input validation's implementation for the Multimedia and Journal Services site is as follows:

- Sanitize and Validate User Input:

Allowing only specific characters, formats, or values, also known as whitelisting input, helps ensure that the input adheres to the expected format, such as restricting an email field to valid email formats. It's also crucial to perform type checking to confirm that the input matches the required data type, such as integers or dates.

- Use Prepared Statements and Parameterized Queries:

It is advisable to use prepared statements instead of inserting raw user input into SQL queries, as this approach ensures that the input is treated as data rather than part of the query itself.

```
query="SELECT * FROM users WHERE
username=%s AND password=%s"

cursor.execute(query, (username, password))
```

- Escape Special Characters

If parameterized queries are not possible, escape special characters in user input before including them in SQL queries.

```
$username=mysqli_real_escape_string ($connection,
$username);
```

- Enforce Strong Input Validation Rules:

For numeric inputs, it is important to ensure that the input is validated as numeric using appropriate language-specific methods, such as is_numeric() in PHP. String inputs should have any potentially harmful characters removed or encoded to prevent misinterpretation by the database engine, including characters like " ; " , " ” , " - " , " ' ", and " " ". Additionally, date inputs should align with the required format, which can be verified using regular expressions or built-in date parsing libraries.

- Use ORM or Framework-Level Protections:

The risk of SQL injection is mitigated when frameworks abstract query construction, as this approach reduces direct interaction with raw SQL.

```
if (filter_var($email,
FILTER_VALIDATE_EMAIL) &&
preg_match("/^[a-zA-Z0-9]*$/", $username)) {
    $stmt = $conn->prepare("SELECT * FROM
users WHERE email = ? AND username = ?");
    $stmt->bind_param("ss", $email, $username);
    $stmt->execute();
} else {
    echo "Invalid input.";
}
```

3.2 Principle of least privilege

The Principle of Least Privilege (PoLP), a fundamental concept in computer security, suggests that individuals, processes, or systems should be granted only the minimum level of access or permissions necessary to perform their specific tasks or functions. Key elements of the least privilege principle include:

- Access Control
- Privilege Separation
- Regular Review
- Principle of Fail-Safe Default
- Segmentation and Isolation
- Least Privilege

3.3 Secure coding practices

The technique of developing software code in a way that minimizes vulnerabilities and lowers the risk of security threats and attacks is known as secure coding. To create applications that are resistant to common security concerns, security considerations must be incorporated into the development process. Several fundamental ideas and recommended methods for secure coding:

- Input Validation
- Parameterized Queries
- Secure Authentication
- Avoid Hardcoding Sensitive Information
- Secure Error Handling
- Protect Against Cross-Siste Scripting
- Secure File Handling
- Regularly Update
- Secure Coding Frameworks
- Security Testing and Code Reviews

3.4 Regular patching and updates

Patching and updating often is essential for preserving the security and reliability of software systems. Consider the following best practices for managing patches and updates:

- Implement a Patch Management Process
- Prioritize Critical Update
- Automate Patch Deployment
- Maintain System Documentation

3.5 Multimedia and web applications and firewalls

A security tool called a Web Application Firewall (WAF) is made to shield web applications against different kinds of assaults. Between the web application and the internet, it serves as a firewall, examining incoming and outgoing traffic to spot and stop dangerous or suspicious activity. Key characteristics and advantages of web application firewalls include:

- Application Layer Protection
- Attack Detection and Prevention
- Web Application Hardening
- DDoS Mitigation
- Logging and Auditing

3.6 Security testing

A crucial phase in the software development life cycle is security testing, which aims to identify weaknesses, vulnerabilities, and security issues within a system or application. An initial test often involves inserting a single quote or semicolon into the field or parameter being examined. The single quote acts as a string terminator in SQL, and if not properly filtered by the application, can lead to a faulty query. Similarly, the semicolon is used to terminate an SQL statement, and if not filtered, is likely to trigger an error. The output from a vulnerable field may appear as follows on a Microsoft SQL Server:

```
Microsoft OLE DB Provider for ODBC Drivers error
'8OO4OeO4'
{Microsoft}{ODBC SQL Server Driver][SQL Server}
Unclosed quotation mark before the
character string '_'.
/folder/file.php, line 254
```

To try to alter the query, use comment delimiters (/* */, --, or others) as well as additional SQL keywords such as AND and OR. A straightforward yet sometimes effective technique is to input a string where a number is expected, which can result in the following error:

```
Microsoft OLE DB Provider for ODBC Drivers error
'8OO4OeO8'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax
error converting the
varchar value 'book' to a column of data type float.
/folder/folder.asp, line 254
```

Monitor all web server responses and review the JavaScript or HTML source code, as issues may be present but not visible to the user. Detailed error messages, like those in the examples, can provide attackers with valuable information to execute a successful injection attack. However, applications often reveal minimal information, such as a generic '500 Server Error' or a custom error page, which may require the use of blind injection techniques. Regardless, it is essential to test each field individually, ensuring that only one variable is altered at a time, to accurately identify which parameters are more vulnerable than others.

3.7 Parsing PCAP implementations

Parsing PCAP (Packet Capture) files can be implemented using various programming languages and libraries designed to read and analyze network traffic data.

- Parse PCAP Files with Scapy

```
from scapy.all import rdpcap
packets = rd pcap('file.pcap')
for packet in packets:
print(packet.summary())
```

- Parse PCAP Files with PyShark

```
import pyshark
# Load the PCAP file
```

```
cap = pyshark.FileCapture('example.pcap')
# Iterate through packets and display information
for packet in cap:
    print(packet)
```

- Additional Steps

Effective handling of PCAP files involves filtering packets with BPF (Berkeley Packet Filter) to focus on specific traffic types, extracting and analyzing protocols such as TCP, UDP, and HTTP along with their metadata, and using tools like Wireshark for visual inspection of the traffic, or alternatively, developing a custom tool for detailed analysis.

4. RESULTS

The experiment for this study was carried out on a multimedia and journal service website. Firstly, need to know the original query is always required to achieve union-based injection. The object of this research is a multimedia website and service journal that has been verified for accuracy [21]. The content of the multimedia and journal services site is as follows:

- Research Papers and Articles
- Abstracts and Summaries
- Author Profiles
- Citations References
- Downloaded Content
- Video/Audio Content
- Images and Graphics
- Content Descriptions
- Metadata

Table 2 shows the steps to retrieve the original query using the default DBMS tables.

Extracting and analyzing network traffic data that has been recorded in the PCAP format is what is involved in parsing file log PCAP (Packet Capture). Pcap files preserve captured packets, payloads, and headers, enabling offline analysis or post-event research. The developer can adhere to the general methods listed below to parse a PACAP file:

- Select a tool for PCAP parsing.
- Open up the PCAP File.
- Extract Information from Packets.
- Analyze and Filter Packets.
- Examine the headers and payloads of packets.
- Conduct a protocol analysis.
- Extract Relevant Information.
- Produce reports or visuals.

Table 2. Default DBMS table

DBMS	Table
My SQL	information_schema. processlist
Postgres SQL	pg. stat activity
Microsoft SQL Server	sys.dm exec cached plans
Oracle	V$ SQL

Reading log files is implemented on a network forensics

server. The log file is examined, which helps observe the flow of packet headers that move around the network.

```
My_data@my_data:~$perl parsing_pcap.pl

Time: 03-12 17:40:11.152692
IP Address Source: aaa.aaa.aaa.aaa Mac Address Source:
03134f601983 Port Numbers: 456O2
IP Address Destination: aaa.aaa.aaa.aaa Mac Address
Destination: OO3462758dda Port Numbers: 8O

Time: 03-11 19:16:31.123545
IP Address Source: aaa.aaa.aaa.aaa Mac Address Source:
OO124a41f375 Port Numbers: 123704
IP Address Destination: aaa.aaa.aaa.aaa Mac Address
Destination: OOO9dfd4343 Port Numbers: 8O
```

A tool to determine which ports are open or closed on a server or host is a port scanning application. The developer can use it by entering perl portscan.pl, followed by the required port number and the IP address of the server or host they wish to analyze.

```
root@my_data:/folder/my_data#      perl      port_scan.pl
aaa.aaa.aaa.aaa 2)-26

The Results are ... ...
Target aaa.aaa.aaa.aaa: Port_2O is closed

Target aaa.aaa.aaa.aaa: Port_21 is open

Target aaa.aaa.aaa.aaa: Port_22 is closed

Target aaa.aaa.aaa.aaa: Port_23 is closed

Target aaa.aaa.aaa.aaa: Port_24 is closed

Target aaa.aaa.aaa.aaa: Port_25 is closed

Target aaa.aaa.aaa.aaa: Port_26 is closed
```

In order to get the answers to forensic queries like what IP address attacked a server, what port did the attacker use to access a system, and other things, log files that have been retrieved from IDS are analyzed using parsing logs and port scans. In the third script, the log files are analyzed using SQLite. They were calling the pkts2db.pl script, opening the logfileall.pcap file, specifying the name of the new database log file, and then typing-d (to create a database) completes the process of converting a log file into a database.

```
My_data@my_data:~$ perl_log_kedb.pl-r data_log.pcap-d
data_log.db

sqlite>select s addr, d addr, count(*) as count
   __>from ip
   __>group by s addr, d addr
   __>order by count desc; s addr      d addr count
          -              aaa.aaa.aaa.aaa   aaa.aaa.aaa.aaa
256
aaa.aaa.aaa.aaa aaa.aaa.aaa.aaa 41
aaa.aaa.aaa.aaa aaa.aaa.aaa.aaa 35
```

```
aaa.aaa.aaa.aaa aaa.aaa.aaa.aaa 2O
aaa.aaa.aaa.aaa aaa.aaa.aaa.aaa 19
aaa.aaa.aaa.aaa aaa.aaa.aaa.aaa 13
aaa.aaa.aaa.aaa aaa.aaa.aaa.aaa 18
aaa.aaa.aaa.aaa aaa.aaa.aaa.aaa 14
```

Only a small number of IP addresses will be examined by the attacker's analysis tool. An attacker believed to be located in Asia Pacific is identified by the IP address 125.201.71.aaa. The website for Multimedia and Journal Services was attacked using sqlmap.

```
My_data@my_data:~$ perl_logkedb.pl-r data_log.pcap-d
data_log.db

sqlite>select s addr, d addr, count(*) as count
   __>from ip
   __>group by s addr, d addr
   __>order by count desc; s addr      d addr count

Time            s addr        d addr
---------------- -------------- -----------
2022-08-10 11:18 8O.255.47.aaa aaa.aaa.aaa
2022-08-10 11:18 8O.255.47.aaa aaa.aaa.aaa
2022-08-10 11:18 8O.255.47.aaa aaa.aaa.aaa

8O.255.47.aaa 1O.13.254.42 Python_urllib/2.8
8O.255.47.aaa 1O.13.254.42 Python_urllib/2.8
```

An attacker who is known to be in Europe is identified by the IP address 80.255.47.aaax. Python is used by the attacker to target the Journal Services Site.

```
My_data@my_data:~$ perl_logkedb.pl-r data_log.pcap-d
data_log.db

sqlite>select s addr, d addr, count(*) as count
   __>from ip
   __>group by s addr, d addr
   __>order by count desc; s addr      d addr count

Time            s addr        d addr
---------------- -------------- -----------
2022-08-09 10:13 125.201.71.aaa aaa.aaa.aaa
2022-08-09 10:13 125.201.71.aaa aaa.aaa.aaa
2022-08-09 10:14 125.201.71.aaa aaa.aaa.aaa

125.201.71.aaa aaa.aaa.aaa.aaa sqlmap/1.O_dev (rNone)
(http://www.sqlmap.org)
```

The process of identifying SQL injection attacks using input validation and parsing methods requires quite a long processing time. Figure 5 is the result of measuring the time required for the input validation process. It can be seen that in the input validation process, the processing time starts to look stable at around 2250 users.

Figure 6 shows the results of measuring the time required for the parsing method process. As with input validation, it can be seen that in the parsing method the processing time starts to look stable at a number of users around 500 users.

The concatenated method process proposed in this study

succeeded in making the processing time more efficient, as shown in Table 3. The concatenated method processing time succeeded in reducing the processing time to be 15.2% more efficient than the sum of the processing times of the two methods separately.
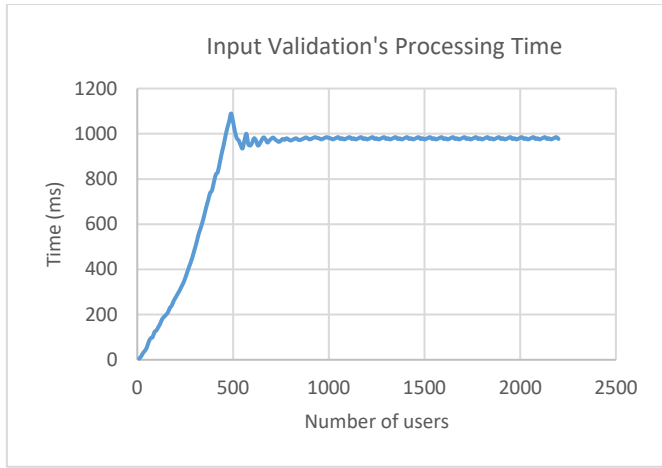

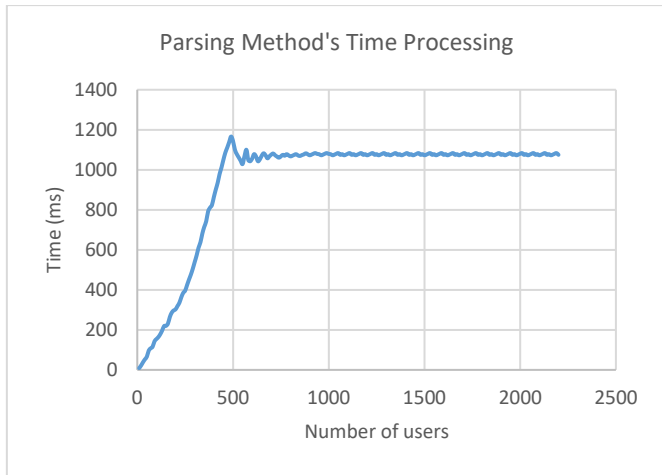**Figure 5.** Input validation's processing time graph

Figure 6. Parsing method's processing time graph

Table 3. Processing time comparison

Users	Time (ms)			Efficiency
	Input validation	Parsing method	Concatenated	
10	6	10	14	12.50%
20	16	21	32	13.51%
30	32	41	63	13.70%
40	42	53	81	14.73%
50	55	64	101	15.12%
…				
…				
…				
2230	986	1084	1749	15.51%
2240	997	1103	1772	15.62%
2250	999	1105	1775	15.64%

The next test is to measure the success rate of identification if an attack occurs on the website. Testing will use a confusion matrix. Testing is carried out by measuring the success of attack identification using input validation, parsing methods and concatenated methods. The formulas for the confusion matrix are in Eqs. (1)-(3) and Table 4.

Table 4. Confusion matrix

Matrix		Actual Class	
		Attack	Not Attack
Prediction Class	Attack	TP (True Positive)	FP (False Positive)
	Not Attack	FN (False Negative)	TN (True Negative)

Table 5. Confusion matrix for input validation

Matrix		Actual Class	
		Attack	Not Attack
Prediction Class	Attack	TP=94	FP=0
	Not Attack	FN=6	TN=100

Table 6. Confusion matrix for parsing method

Matrix		Actual Class	
		Attack	Not Attack
Prediction Class	Attack	TP=92	FP=0
	Not Attack	FN=8	TN=100

Table 7. Confusion matrix for concatenated method

Matrix		Actual Class	
		Attack	Not Attack
Prediction Class	Attack	TP=99	FP=0
	Not Attack	FN=1	TN=100

$$\text{Precision} = \frac{TP}{TP + FP} \times 100\% \qquad (1)$$

$$\text{Recall} = \frac{TP}{TP + FN} \times 100\% \qquad (2)$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \qquad (3)$$

Based on the data in Table 5, the results of the attack detection experiment with input validation are as follows:

Precision=(94)/(94)x100%=100%

Recall=(94)/(100)x100%=94%

Accuracy=(94+100)/(200)=97%

Based on the data in Table 6, the results of the attack detection experiment with the parsing method are as follows:

Precision=(92)/(92)x100%=100%

Recall=(92)/(100)x100%=92%

Accuracy=(92+100)/(200)=96%

Based on the data in Table 7, the results of the attack detection experiment with concatenated method are as follows:

Precision=(99)/(99)x100%=100%

Recall=(99)/(100)x100%=99%

Accuracy=(99+100)/(200)=99.5%

In order to calculate the confidence interval of the efficiency carried out, the formula used is

$$CI = X \pm \frac{s}{\sqrt{n}} \qquad (4)$$

Sample size (amount)=2250
Sample mean (average)=15.2%
Standard deviation=1.5%
Confidence Level=95%

CI=15.2±0.062

The statistical calculation of the Confidence Level of the efficiency of using the concatenated method is 15.2±0.062.

The next statistic used is to calculate the p-value using an analysis of variance (ANOVA).

Based on Table 8 and Table 9, The f-ratio value is 6.99968. The p-value is 0.008446. The result is significant at $p < 0.05$. Based on the ANOVA statistical results, it was found that the proposed method significantly improved efficiency.

Table 8. Summary of ANOVA data

	Treatment		
	1	**2**	**Total**
N	250	250	500
Σ X	377974	349241	727215
Mean	1718.06	1587.46	1652.76
Σ X²	712734536	608464667	1321199203
Std. Dev.	537.85	496.83	521.27

Table 9. Results

Source	SS	DF	MS
Between-treatments	1876330.20	1	1876330.20
Within-treatments	117410017.75	448	268059.40
Total	119286347.94	449	

Table 10. Comparative study with other methods

Method	Strengths	Weaknesses	Best Used For
Concatenated Input Validation and Parsing (CIVP)	Batch processing, simplicity in certain scenarios	Parsing complexity, difficult error isolation, security risks	Systems where inputs are combined before validation
Regular Expressions (RegEx)	Granular control, efficient pattern matching	Hard to maintain, limited logic, potential security issues	Simple, well-defined input fields
Whitelisting	High security, simple and effective	Restrictive, frequent updates required	Systems with strict input rules
Blacklisting	Easy to implement for basic cases	Insecure, complex to maintain for evolving threats	Blocking specific known malicious inputs
Structured Validation (JSON/XML)	Strong data integrity, wide validation rules	Performance overhead, complexity	Structured data formats, such as APIs and services

Table 10 is a comparison between the Concatenated Input

Validation and Parsing (CIVP) method and other methods. Other methods used for comparison are Regular Expression (RegEx), Whitelisting, Blacklisting, and Structured Validation (JSON/XML).

Based on the results of statistical calculations, several further analyses can be taken as follows:

• The narrow range of the confidence interval (CI) suggests that the sample mean is a good estimate of the population mean, indicating high precision in the study's estimate of efficiency.

• F-Ratio: The F-value of 6.99968 indicates that there is variability between the treatment means that is larger than what we would expect due to random chance. A higher F-value indicates more substantial differences between group means.

• p-Value: The p-value is 0.008446, which is less than the common significance level of 0.05. This indicates that the differences between the two-treatment means are statistically significant. In other words, there is strong evidence that efficiency improvement is seen with the concatenated method.

• The Concatenated Input Validation and Parsing Method offers efficiency in certain batch processing scenarios but may introduce significant security risks and error-handling challenges, particularly if parsing is not well-defined. Other methods like RegEx, Whitelisting, and Structured Validation provide more granular control, but each comes with trade-offs in complexity, flexibility, and security. Whitelisting is usually the most secure method, whereas Structured Validation excels in complex data formats.

5. CONCLUSIONS

Network forensic investigations are carried out to trace the traces of the attacker. The log files can be used to look for evidence of unauthorized network activity. The information is derived from IDS Snort, a network-based intruder detection system. IDS Snort uses a number of rules (rules) to identify network intruders, and enforcing these rules is crucial to identifying attacks.

On the network forensic server, PERL scripts are used to decipher log files according to the time of the attack, the IP address, the Mac address, and the port. The script for log file analysis using SQLite and the ports scanning script are then used to discover open ports on a server. A port scan script aims to determine which ports are open if an attacker successfully breaches a system using SQL Injection or exploiting online vulnerabilities with databases. Then the log file is examined using the SQLite script. The three scripts and the employed modules are uploaded to the forensic network server.

By having network forensic research available via the Journal Services Site, people are believed to realize how challenging it is to defend networks from intrusions. It is possible to take steps to stop it from happening again or lessen the harm the attack will do.

Based on the test results, it was found that the use of the method proposed in this study succeeded in making processing time 15.2% more efficient. Experiments carried out with the method proposed in this study succeeded in increasing the attack detection accuracy from 96-97% to 99.5%.

SQL attack prevention is very limited by the form of data to be protected. The method in this study has limitations because it is specifically for data contained in the Multimedia and Journal Services Site which consists of research article data

including multimedia files such as video and audio.

REFERENCES

[1] Chala, O., Novikova, L., Chernyshova, L., Kalnitskaya, A. (2020). Method for detecting shilling attacks based on implicit feedback in recommender systems. EUREKA: Physics and Engineering, 5: 21-30. https://doi.org/10.21303/2461-4262.2020.001394

[2] Hoang, X.D., Nguyen, T.H. (2021). Detecting common web attacks based on supervised machine learning using web logs. Journal of Theoretical and Applied Information Technology, 99(6): 1339-1350.

[3] Szczypiorski, K. (2020). Cyber (in) security. International Journal of Electronics and Telecommunications, 6(1): 243-248. https://doi.org/10.24425/ijet.2020.131870

[4] Wiśniewski, P., Sosnowski, M., Burakowski, W. (2022). On implementation of efficient inline DDoS detector based on AATAC algorithm. International Journal of Electronics and Telecommunications, 68(4): 889-898. https://doi.org/10.24425/ijet.2022.143899

[5] Kumar, H.T.| R. (2021). Attack and anomaly detection in IoT networks using supervised machine learning approaches. Revue d'Intelligence Artificielle, 35(1): 11-21. https://doi.org/10.18280/ria.350102

[6] Dasari, K.B., Devarakonda, N. (2022). TCP/UDP-Based exploitation DDoS attacks detection using ai classification algorithms with common uncorrelated feature subset selected by pearson, spearman and kendall correlation methods. Revue d'Intelligence Artificielle, 36(1): 61-71. https://doi.org/10.18280/ria.360107

[7] Hoang, X.D., Nguyen, N.T. (2019). Detecting website defacements based on machine learning techniques and attack signatures. Computers, 8(2): 35. https://doi.org/10.3390/computers8020035

[8] Challa, R., Rao, K.S. (2022). Resource based attacks security using RPL protocol in internet of things. Ingénierie des Systèmes d'Information, 27(1): 165-170. https://doi.org/10.18280/isi.270120

[9] Murty, M.S., Rao, N.N. (2020). Stalking the resources for security in linked data applications using resource description framework. Ingénierie des Systèmes d'Information, 25(6): 793-801. https://doi.org/10.18280/isi.250609

[10] Antunes, N., Vieira, M. (2009). Detecting SQL injection vulnerabilities in web services. In 2009 Fourth Latin-American Symposium on Dependable Computing, Joo Pessoa, Brazil, pp. 17-24. https://doi.org/10.1109/LADC.2009.21

[11] Nagabhooshanam, N., Ganapathy, N.B.S., Ravindra Murthy, C., Mohammed Saleh, A.A., CosioBorda, R.F. (2023). Neural network based single index evaluation for SQL injection attack detection in health care data. Measurement: Sensors, 27: 100779. https://doi.org/10.1016/j.measen.2023.100779

[12] Dubey, A.M.S., Mehra, N. (2023). A review on SQL injection, detection and preventions techniques. Journal of Pharmaceutical Negative Results, 1: 1068-1073. https://doi.org/10.47750/pnr.2023.14.S01.148

[13] Maraj, A., Rogova, E., Jakupi, G., Grajqevci, X. (2017). Testing techniques and analysis of SQL injection attacks. In 2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA), London, UK, pp. 55-59. https://doi.org/10.1109/ICKEA.2017.8169902

[14] Gu, H., Zhang, J., Liu, T., Hu, M., Zhou, J., Wei, T., Chen, M. (2019). DIAVA: A traffic-based framework for detection of SQL injection attacks and vulnerability analysis of leaked data. IEEE Transactions on Reliability, 69(1): 188-202. https://doi.org/10.1109/TR.2019.2925415

[15] Alkhathami, J.M., Alzahrani, S.M. (2022). Detection of SQL injection attacks using machine learning in cloud computing platform. Journal of Theoretical and Applied Information Technology, 100(15): 5446-5459.

[16] Tashenova, Z., Nurlybaeva, E., Tulegulov, A., Abdugulova, Z. (2021). SQL-Attack research and protection. Journal of Theoretical and Applied Information Technology, 99(19): 4536-4545. http://www.jatit.org/volumes/Vol99No19/8Vol99No19.pdf

[17] Volkova, M., Chmelar, P., Sobotka, L. (2019). Machine learning blunts the needle of advanced SQL injections. In MENDEL, 25(1): 23-30. https://doi.org/10.13164/mendel.2019.1.023

[18] Bandhakavi, S., Bisht, P., Madhusudan, P., Venkatakrishnan, V.N. (2007). CANDID: Preventing SQL injection attacks using dynamic candidate evaluations. In Proceedings of the 14th ACM Conference on Computer and Communications Security, New York, United States, pp. 12-24. https://doi.org/10.1145/1315245.1315249

[19] Johny, J.H.B., Nordin, W.A.F.B., Lahapi, N.M.B., Leau, Y.B. (2021). SQL Injection prevention in web application: A review. In Advances in Cyber Security: Third International Conference, ACeS 2021, Penang, Malaysia, Revised Selected Papers, Springer, Singapore, 3: 568-585. https://doi.org/10.1007/978-981-16-8059-5_35

[20] Demilie, W.B., Deriba, F.G. (2022). Detection and prevention of SQLI attacks and developing compressive framework using machine learning and hybrid techniques. Journal of Big Data, 9(1): 124. https://doi.org/10.1186/s40537-022-00678-0

[21] Wijaya, M.C., Maksom, Z., Abdullah, M.H.L. (2021). Two verification phases in multimedia authoring modeling. Journal of Information and Communication Convergence Engineering, 19(1): 42-47. https://doi.org/10.6109/jicce.2021.19.1.42