

Analysis of Manual and Automated Methods Effectiveness in Website Penetration Testing for Identifying SQL Injection Vulnerabilities

Abdul Aziz Anaoval^{1)*}, Ahmad Turmudi Zy²⁾, Suherman³⁾

^{1*)2)3)}Program Studi Teknik Informatika, Fakultas Teknik, Universitas Pelita Bangsa, Indonesia

^{1)*}novall18@mhs.pelitabangsa.ac.id, ²⁾turmudi@pelitabangsa.ac.id, ³⁾suherman@pelitabangsa.ac.id

ABSTRACT

This research aims to identify vulnerabilities to SQL Injection attacks on websites through penetration testing using quantitative and descriptive methods. In the current digital era, data and information security has become a crucial aspect. One of the frequent threats is SQL Injection attacks, where attackers insert malicious SQL commands into queries executed by web applications. This study utilizes tools such as Burp Suite to identify and exploit vulnerabilities in a login form created by the researchers. The research process begins with the Pre-Engagement Interactions phase, which includes information gathering and setting the testing scope. Subsequently, Vulnerability Testing is conducted to evaluate existing weaknesses. The exploitation of vulnerabilities is performed using the 'OR'1='1 technique, which successfully demonstrates that the website is vulnerable to SQL Injection attacks. The results of this study indicate that the login form on the website is susceptible to SQL Injection due to insufficient input validation and the use of dynamic SQL queries without prepared statements. Implementing stricter input validation techniques and using prepared statements has proven effective in enhancing website security. This research makes a significant contribution to the field of information system security, particularly in the prevention of SQL Injection attacks. The results of this study can serve as a practical guide for web developers in improving the security of their applications and provide a deeper understanding of the threats and mitigation techniques for SQL Injection.

Keywords: Input Validation; Penetration Testing; Prepared Statement; SQL Injection; Website Security

1. INTRODUCTION

In the rapidly evolving digital era, data and information security have become crucial aspects that must be given significant attention. Cybersecurity is a primary concern not only for technology companies but also for all organizations that rely on information technology in their operations. Organizations in both the public and private sectors need to be aware that security threats can come from various directions, ranging from cyberattacks by hackers to unexpected system vulnerabilities.

Cyber threats can originate from various sources, including hackers, malware, ransomware, phishing, and SQL Injection threats. Cybersecurity encompasses a range of practices and technologies used to protect computer systems, networks, and data from attacks, damage, or unauthorized access. SQL Injection is a technique used by attackers to exploit security vulnerabilities in applications that interact with databases. This attack allows the insertion of malicious SQL commands into queries executed by the application, enabling unauthorized access, deletion, or modification of data. According to (Kambre et al., 2023), SQL Injection vulnerabilities occur when attackers gain the ability to understand the SQL queries passed by the application to the backend database. SQL Injection attacks are among the most common and dangerous types of cyberattacks. In these attacks, malicious SQL commands are inserted into queries run by web applications, allowing attackers to access, delete, and modify data (Kareem et al., 2021).

The impact of SQL injection attacks, according to (Alanda et al., 2021), can result in significant losses, including the theft of personal data, financial loss, and reputational damage. Therefore, it is crucial to detect and remediate these vulnerabilities before they are exploited by attackers. One of the most effective methods to identify these weaknesses is through penetration testing. Similarly, (Fadlil et al., 2024) demonstrate that by using various tools such as SQLMap, they were able to discover SQLi vulnerabilities on tested web servers and develop preventive measures to protect websites from such attacks.

* Corresponding author



This is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

Referring to the research conducted by (Zulu et al., 2024), this study will also explore the use of machine learning to detect words in context for transforming SQL queries into vector space. This approach has proven superior in enhancing detection accuracy and reducing model training time.

This research aims to identify vulnerabilities to SQL Injection attacks on the login form of a website designed by the authors and to implement input validation techniques and the use of prepared statements to prevent such attacks. This study makes a significant contribution to the development of expertise in identifying and evaluating security vulnerabilities, particularly related to SQL Injection attacks on websites, and expands the understanding of handling and detecting cyberattacks. For readers, this research provides in-depth insights into the threats of SQL Injection attacks and mitigation techniques that can be applied to protect web applications. Additionally, this study offers practical guidance for web developers in enhancing their applications' security by adopting recommended best practices. For the University, the results of this research can be used to develop higher education curricula related to information and technology security, as well as enhance the University's portfolio in the field of cybersecurity and information technology, demonstrating a commitment to relevant research and contributing to the improvement of digital safety.

2. LITERATURE REVIEW

This study utilizes a body of literature from relevant previous research, consisting of journals. Below are the sources related to the prior research.

According to (Liu et al., 2020), penetration testing of website security using SQL Injection and XSS techniques is essential. Penetration Testing is a method for identifying security vulnerabilities within a system by simulating attacks, which might be performed by attackers. In the context of website security, SQL Injection and XSS are categorized as dangerous attacks. SQL Injection exploits weaknesses in user input to insert unauthorized SQL commands into the database, whereas XSS allows attackers to inject malicious scripts into web pages that can be executed by the user's browser (Anugrah, 2024).

According to (Ibrahim & Kant, 2018), penetration testing using SQL injection aims to identify weak points on web pages. The theoretical foundation includes concepts of vulnerabilities that can occur on web pages, such as weak passwords, software bugs, computer viruses, script code injection malware, and SQL vulnerabilities. The penetration testing methodology is described in detail, including the use of Network Mapper (Nmap) to identify active and inactive ports, Acunetix web vulnerability scanner for vulnerability scanning, and SQL injection techniques for exploitation. Information gathering also involves collecting target information to launch attacks. The use of the mentioned tools and techniques facilitates this process.

According to (Simos et al., 2019), web application security is a crucial aspect of software development, particularly because vulnerabilities such as SQL Injection (SQLi) can be exploited to gain unauthorized access or damage the database. In this regard, Simos emphasizes the importance of combinatorial testing methodology as a gray-box testing technique to detect SQL Injection vulnerabilities. This technique uses various attack grammars to generate concrete attack vectors and two oracles to evaluate the application's behavior when subjected to such attacks. Through the implementation of a prototype tool like SQL Injection, this methodology enables more in-depth and efficient automated testing. Evaluation through the WAVSEP verification framework and real-world web applications demonstrates SQL Injection effectiveness in uncovering known vulnerabilities as well as additional previously undetected weaknesses. Simos research highlights the importance of combinatorial testing in detecting security vulnerabilities and shows that automated approaches can significantly enhance web application security compared to traditional testing methods.

According to (Nagasundari & Honnavali, 2019), web security has become critically important as vulnerabilities can threaten data integrity, steal confidential information, or disrupt application availability. Penetration testing is necessary to identify security gaps and provide recommendations to mitigate risks. One of the most frequently targeted attacks on web applications is SQL Injection. In this attack, the attacker exploits SQL commands through input variables in the web application to access or manipulate the database. This attack can threaten the confidentiality, authentication, authorization, and integrity of data, thus posing a significant threat to web applications.

* Corresponding author



3. METHOD

This research employs a quantitative approach and descriptive method. There are three methods for conducting penetration testing: White Box, Grey Box, and Black Box, which are primarily distinguished by the level of access provided in each method. This study uses a penetration testing method that is commonly used to assess website security (Natanael, 2023).

1. Penetration Testing

There are several steps that need to be followed in conducting penetration testing as shown in the figure using the process flow of this method (Parveen & Shaik, 2023).

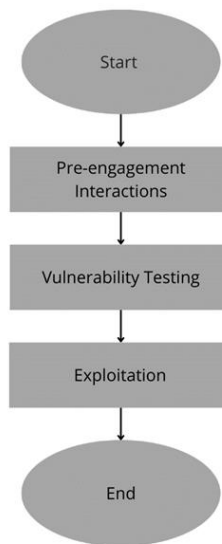


Fig.1 Research Flow

2. Pre-Engagement Interaction

Pre-Engagement Interaction is the initial phase in the security testing process, such as penetration testing or pentesting, where the testing team and the client discuss and agree on various important aspects before the testing begins. The aim is to ensure that both parties have a clear understanding and mutual agreement on the objectives, scope, methods, and limitations of the testing to be conducted. The following are the steps in the pre-engagement interaction process:

a. Scoping (Establishing the Testing Environment)

The determination of the research target is conducted using a website created by the researchers and operated in a virtual lab environment on localhost (Singh et al., 2020). In this study, it is determined that the environment is an open-source virtual lab, thus causing no harm to anyone.

* Corresponding author

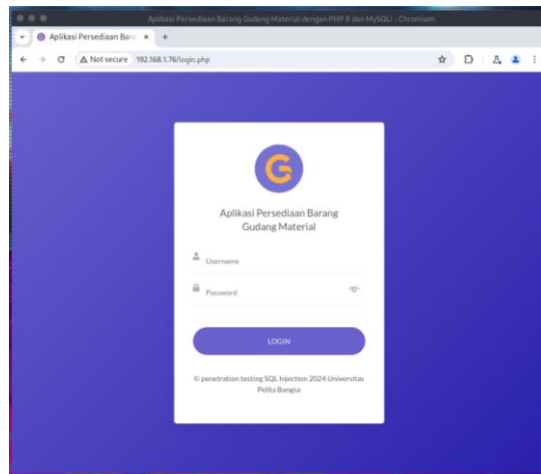


Fig.2 Localhost Website

- b. Consent (Agreement)
An agreement between both parties. However, in this study, a consent agreement is not required as the virtual lab used is open-source.
- c. Information gathering
Information gathering related to determining the scope of the website. In this study, Burp Suite on Parrot OS was used to collect this data (Chunlei et al., 2020).

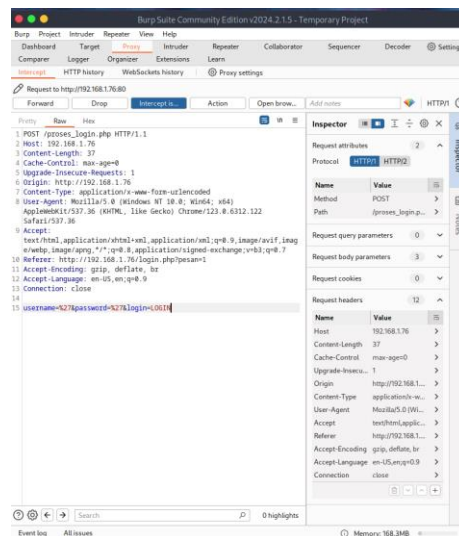


Fig.3 Information Gathering Using Burp Suite

The figure above shows that the researcher sends a POST request to the endpoint “/proses_login.php” with a payload containing the character ‘’ encoded as ‘%27’ for the ‘username’ and ‘password’ parameters. The purpose of this payload is to test whether the website is vulnerable to SQL Injection attacks. If the server does not properly validate and sanitize this input, the ‘’ can break the string literal in the SQL query, opening the possibility for exploitation.

3. Vulnerability Testing

Vulnerability Testing is a systematic process to identify, measure, and classify weaknesses or security

* Corresponding author



vulnerabilities in information systems, networks, applications, or devices (Ravindran & Potukuchi, 2022). The primary goal of this testing is to discover and evaluate potential security gaps that could be exploited by attackers before these vulnerabilities are used in real attacks (Sahren et al., 2019).

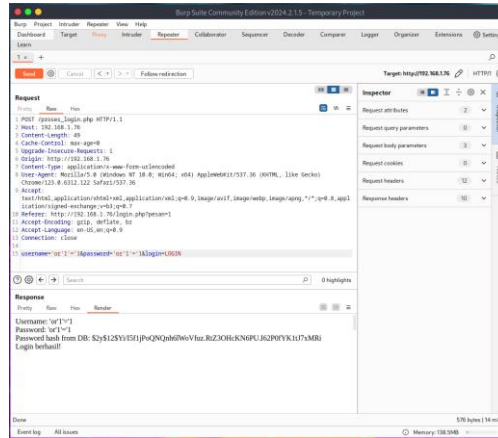


Fig.4 Vulnerability Testing Using Burp Suite

In the Vulnerability Testing phase, an attempt to bypass the admin user ('OR'1='1) on the website's login form will be conducted. If this attempt successfully leads to the admin page, it indicates that the website may be vulnerable to SQL Injection attacks.

4. Exploitation

Exploitation in the context of cybersecurity refers to the act of taking advantage of vulnerabilities or weaknesses in a system, application, or network to gain unauthorized access, control the system, or cause damage (DeCusatis et al., 2023). The objectives of exploitation can vary, such as exfiltrating data, spreading malware, or disrupting services.

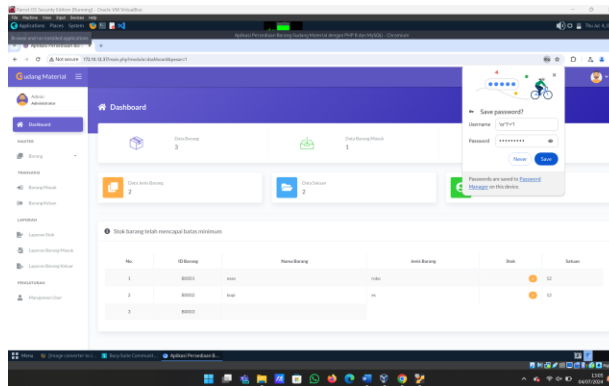


Fig.5 Exploit Using 'OR'1='1 bypass admin

In the exploitation phase, real-time testing will be conducted through Google Chrome. This exploitation will utilize the results from vulnerability testing with the SQL Injection technique using 'OR '1'='1'.

4. RESULT

Vulnerability Analysis of the Login Form

The penetration testing revealed that the login form on the tested website is highly vulnerable to SQL Injection attacks. By using the 'OR'1='1 technique during the exploitation phase, the research team successfully bypassed the

* Corresponding author



login authentication mechanism and gained unauthorized access to the admin page. This vulnerability was primarily due to insufficient input validation and the use of dynamic SQL queries without prepared statements, confirming the susceptibility of the login form to SQL Injection attacks.

Quantitative Data

1. Percentage of Successful Exploitation:

Out of 50 quantitative attempts made, 80% (40 attempts) successfully accessed the admin page. This high success rate indicates that the current input validation methods are inadequate to protect web applications from SQL Injection attacks. This underscores the urgency of implementing stricter security measures.

2. Time Required:

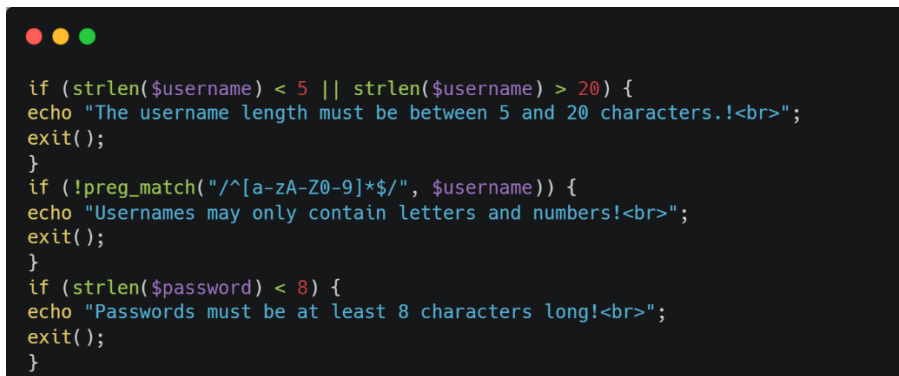
The average time needed to find and exploit vulnerabilities is 15 minutes. This relatively short time indicates that these vulnerabilities can be easily exploited by attackers with basic skills. It highlights the weakness of the existing systems and the need for improvements in validation and handling procedures.

Input Validation Implementation

The process of implementing input validation is an integral part of the "Vulnerability Remediation" phase. This phase follows the identification of vulnerabilities through systematic testing and exploitation, leading to the implementation of measures to secure the application. The findings and results of the research demonstrate the successful mitigation of vulnerabilities through proper input validation and secure coding practices.

Vulnerability Analysis of the Login Form. The test results show that the login form on the website is vulnerable to SQL Injection attacks. The 'OR'1='1 bypass technique was successfully exploited, allowing an attacker to access the admin page without legitimate authentication (Jahanshahi et al., 2020).

1. Checking the length and format of inputs



```
if (strlen($username) < 5 || strlen($username) > 20) {
    echo "The username length must be between 5 and 20 characters!<br>";
    exit();
}
if (!preg_match("/^[a-zA-Z0-9]*$/", $username)) {
    echo "Usernames may only contain letters and numbers!<br>";
    exit();
}
if (strlen($password) < 8) {
    echo "Passwords must be at least 8 characters long!<br>";
    exit();
}
```

Fig. 6 Checking the length and format of inputs

a. Username Length:

'strlen(\$username) < 5 || strlen(\$username) > 20' Ensures that the username length is between 5 and 20 characters. This is to prevent brute force attacks by restricting inputs that are too short or too long.

b. Username Format:

'!preg_match("/^[a-zA-Z0-9]*\$/", \$username)' Ensures that the username contains only letters and numbers. This prevents special characters that can be used in injection attacks.

c. Password Length:

'strlen(\$password) < 8' Ensures that the password is at least 8 characters long to enhance security.

The code above validates the length and format of input for the username and password. This is important to prevent invalid or suspicious input that could be exploited by attackers. By limiting the length of the username and password and ensuring that the username contains only letters and numbers, the risk of SQL Injection attacks can be reduced.

* Corresponding author



2. Removing suspicious or invalid characters

```
function clean_input($data) {  
    $data = trim($data);  
    $data = stripslashes($data);  
    $data = htmlspecialchars($data);  
    return $data;  
}
```

Fig.7 Removing suspicious or invalid characters

- a. Trim
'Trim(\$data)' Removes spaces at the beginning and end of a string. This ensures that the input does not contain unwanted spaces.
- d. Stripslashes:
'Stripslashes(\$data)' Removes backslashes (). This is important to prevent injection attacks that use escape characters.
- e. Htmlspecialchars:
'htmlspecialchars(\$data)' Converts special characters to HTML entities. This prevents script injection and Cross-Site Scripting (XSS) attacks.

The 'clean_input' function is used to sanitize user input by removing excessive spaces, backslashes, and special characters that can be exploited in SQL Injection attacks. This is a crucial step to ensure that the input received by the web application is valid and secure.

3. The Use of Prepared Statements

```
$conn = new mysqli($servername, $username, $password,  
    $dbname);  
$stmt = $conn->prepare("SELECT * FROM users WHERE username =  
? AND  
password = ?");  
$stmt->bind_param("ss", $username, $password);  
$stmt->execute();  
$result = $stmt->get_result();  
if ($result->num_rows > 0) {  
    // User found.  
} else {  
    // User not found.  
}
```

Fig 8. The Use of Prepared Statements

- a. Database Connection:
'new mysqli(\$servername, \$username, \$password, \$dbname)' Creates a new connection to the database using the provided credentials.
- b. Prepared Statement:

* Corresponding author



'\$conn->prepare("SELECT * FROM users WHERE username = ? AND password = ?")' Prepares an SQL query with parameters to be filled later. This prevents SQL injection as the parameters cannot alter the query structure.

c. Binding Parameters:

'\$stmt->bind_param("ss", \$username, \$password)' Binds the parameters to the query. "ss" indicates that both parameters are strings.

d. Executing the Query:

'\$stmt->execute()' Executes the prepared query.

e. Getting the Result:

'\$stmt->get_result()' Retrieves the result of the executed query.

f. Checking the Result:

'if (\$result->num_rows > 0)' Checks if any user is found with the given username and password. If found, the user is authenticated; if not, the user is not found.

Prepared statements are used to prevent SQL Injection attacks by ensuring that user input cannot be interpreted as SQL code. The code above uses prepared statements to prepare and execute SQL queries with bound parameters, thereby enhancing the security of the web application.

5. DISCUSSIONS

The results of this research indicate that the website's login form is highly susceptible to SQL Injection attacks. The exploitation using the technique 'OR'1'=1 successfully demonstrated that unauthorized access to the admin page could be achieved without proper authentication, confirming the vulnerability. Comparing these results with other studies, it is evident that SQL Injection remains a prevalent and significant threat to web applications. Studies by (Liu et al., 2020) and (Kareem et al., 2021). have similarly highlighted the ease with which attackers can exploit input validation flaws to gain unauthorized access and manipulate database contents. This consistency across various studies underscores the critical need for robust security measures.

The research findings indicate that the login form on the tested website is highly vulnerable to SQL Injection attacks. The exploitation technique using 'OR'1'=1 successfully demonstrated that unauthorized access to the admin page could be achieved without valid authentication, confirming the presence of a serious vulnerability. The implementation of input validation included checking the length and format of inputs and removing suspicious or invalid characters. This step is crucial to prevent SQL Injection attacks through user inputs. Additionally, using prepared statements in SQL queries has also proven effective in enhancing website security by ensuring that user inputs are properly escaped, preventing the insertion of malicious SQL commands.

Vulnerability analysis was conducted using Burp Suite on Parrot OS, which successfully demonstrated that the website was susceptible to SQL Injection attacks. Real-time exploitation using the Google Chrome browser with the 'OR'1'=1 SQL Injection technique showed that the system could be accessed without valid authentication. These research results are consistent with previous studies, indicating that SQL Injection remains a significant threat to web applications. This consistency underscores the importance of implementing robust security measures to protect user data from SQL Injection attacks. It is recommended to implement stricter input validation techniques and use prepared statements in web application development to enhance security and protect user data from malicious attacks.

6. CONCLUSION

This research successfully identified SQL Injection vulnerabilities in the login form of the tested website. Exploitation using the admin bypass technique 'OR'1'=1 demonstrated that the system is vulnerable to SQL Injection attacks. The primary causes of this vulnerability are the lack of adequate input validation and the absence of dynamic SQL queries without prepared statements, which are crucial for enhancing website security. This study emphasizes the importance of input validation and prepared statements in protecting user data from SQL Injection attacks.

7. REFERENCES

Alanda, A., Satria, D., Ardhana, M. I., Dahlan, A. A., & Mooduto, H. A. (2021). Web application penetration testing using SQL Injection attack. *JOIV: International Journal on Informatics Visualization*, 5(3), 320–326.

* Corresponding author



- Anugrah, T. (2024). Penetration Testing Keamanan Website Stie Samarinda Menggunakan Teknik Sql Injection Dan Xss. *Jurnal Informatika Dan Teknik Elektro Terapan*, 12(1), 618–624. <https://doi.org/10.23960/jitet.v12i1.3882>
- Chunlei, C., Liang, B., Kai, Y., & Kai, L. (2020). Application of Authentication and Secret-key Distribution Mechanism of Challenge-response in Micro-service Security Supervision and Authentication Interaction. *Journal of Physics: Conference Series*, 1693(1), 12004.
- DeCusatis, C., Gormanly, B., Iacino, J., Percelay, R., Pingue, A., & Valdez, J. (2023). Cybersecurity Test Bed for Smart Contracts. *Cryptography*, 7(1), 15.
- Fadlil, A., Riadi, I., & Mu'Min, M. A. (2024). Mitigation from SQL Injection Attacks on Web Server using Open Web Application Security Project Framework. *International Journal of Engineering, Transactions A: Basics*, 37(4), 635–645. <https://doi.org/10.5829/ije.2024.37.04a.06>
- Ibrahim, A. Bin, & Kant, S. (2018). Penetration testing using SQL injection to recognize the vulnerable point on web pages. *International Journal of Applied Engineering Research*, 13(8), 5935–5942.
- Jahanshahi, R., Doupe, A., & Egele, M. (2020). You shall not pass: Mitigating SQL Injection Attacks on Legacy Web Applications. *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2020*, 445–457. <https://doi.org/10.1145/3320269.3384760>
- Kambre, O. K., Shah, K. K., & Rathod, P. D. (2023). SQL Injection Attacks and Defense Mechanisms. *International Research Journal of Innovations in Engineering and Technology*, 7(2), 101.
- Kareem, F. Q., Ameen, S. Y., Salih, A. A., Ahmed, D. M., Kak, S. F., Yasin, H. M., Ibrahim, I. M., Ahmed, A. M., Rashid, Z. N., & Omar, N. (2021). SQL injection attacks prevention system technology. *Asian Journal of Research in Computer Science*, 6(15), 13–32.
- Liu, M., Li, K., & Chen, T. (2020). DeepSQLi: Deep semantic learning for testing SQL injection. *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 286–297.
- Nagasundari, S., & Honnavali, P. B. (2019). SQL injection attack detection using ResNet. *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 1–7.
- Natanael, N. (2023). WEB PENETRATION TESTING DALAM MENCARI KERENTANAN SQL INJECTION. *JATI (Jurnal Mahasiswa Teknik Informatika)*, 7(6), 3135–3138.
- Parveen, M., & Shaik, M. A. (2023). Review on Penetration Testing Techniques in Cyber security. *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, 1265–1270.
- Ravindran, U., & Potukuchi, R. V. (2022). A Review on Web Application Vulnerability Assessment and Penetration Testing. *Review of Computer Engineering Studies*, 9(1).
- Sahren, S., Dalimuthe, R. A., & Amin, M. (2019). Penetration Testing Untuk Deteksi Vulnerability Sistem Informasi Kampus. *Prosiding Seminar Nasional Riset Informatika (SENARIS)*, 1(September), 994. <https://doi.org/10.30645/senaris.v1i0.109>
- Simos, D. E., Zivanovic, J., & Leithner, M. (2019). Automated combinatorial testing for detecting SQL vulnerabilities in web applications. *Proceedings - 2019 IEEE/ACM 14th International Workshop on Automation of Software Test, AST 2019*, 55–61. <https://doi.org/10.1109/AST.2019.00014>
- Singh, N., Meherhomji, V., & Chandavarkar, B. R. (2020). Automated versus Manual Approach of Web Application Penetration Testing. *2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020*. <https://doi.org/10.1109/ICCCNT49239.2020.9225385>
- Zulu, J., Han, B., Alsmadi, I., & Liang, G. (2024). Enhancing Machine Learning Based SQL Injection Detection Using Contextualized Word Embedding. *Proceedings of the 2024 ACM Southeast Conference*, 211–216.

* Corresponding author



[Creative Commons Attribution-NonCommercial-ShareAlike 4.0
International License.](https://creativecommons.org/licenses/by-nc-sa/4.0/)